A SMART CITY INTELLIGENT ROUTING APPLICATION FOR WHEELCHAIR USERS USING K-MEANS CLUSTERING

A THESIS SUBMITTED TO
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
KOCAELİ UNIVERSITY

BY

FIRDAWS BAI FARUKH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

KOCAELİ 2020

A SMART CITY INTELLIGENT ROUTING APPLICATION FOR WHEELCHAIR USERS USING K-MEANS CLUSTERING

A THESIS SUBMITTED TO
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
KOCAELİ UNIVERSITY

BY

FIRDAWS BAI FARUKH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

Prof.Dr. Nevcihan DURU		
Supervisor,	**Kocaeli University**	**.....**
Assoc.Prof.Dr. Ahmet SAYAR		
Jury member,	**Kocaeli University**	**.....**
Assoc.Prof.Dr. Tamer DAĞ		
Jury member,	**Kadir Has University**	**.....**

Thesis Defense Date: 17.08.2020

PREFACE AND ACKNOWLEDGMENTS

The main subject of this thesis was to provide a software solution for the transportation and movement problems faced by the wheelchair users in Izmit. The idea stemmed from my passion of both smart city applications and the inclusivity through ubiquitous accessibility phenomenon of our world.

I could not have successfully completed my research without the confidence, support and guidance of my scholarly advisor, Prof. Dr. Nevcihan DURU for whom I have my most sincere gratitude.

I would also like to acknowledge the contributions made by the Disabled and Elderly Services Branch Office of Izmit as well as by Gizem AYDIN, who both incorporated me into their daily activities, collaborated with me and allowed me the chance to ask, learn, observe, and understand the hardships of wheelchair users for my research. I also appreciate the scholarship and financial support provided by the Turks Abroad and Related Communities (YTB) institution of Turkey.

And most importantly, I am grateful for my resilient mother, Shamim SULEIMAN who has been the greatest source of strength in my life and my sisters and brother whom we have braced this life together and have supported, guided, and advised me with love and understanding. Thank you all for your unwavering support.

July, 2020 Firdaws FARUKH

CONTENTS

TABLE OF FIGURES

TABLES DIRECTORY

SYMBOLS AND ABBREVIATIONS

Abbreviations

AI	: Artificial Intelligence
API	: Application Programming Interfaces
CIMI	: Cities In Motion Index
GIS	: Geographic Information Systems
GSV	: Google Street View
GTFS	: General Transit Feed Specification
ICT	: Information and Communication Technologies
IoT	: Internet of Things
İSBAK	: İstanbul Bilişim ve Akıllı Kent Teknolojileri A.Ş. (Istanbul IT, and Smart City Technologies Inc.)
ITS	: Intelligent Transport Systems
LED	: Light-Emitting Diode
MBaaS	: Mobile Backend as a Service
QGIS	: Quantum Geographical Information System
SSE	: Sum of Squared Errors
SVM	: Support Vector Machines
UI	: User Interface

TEKERLEKLİ SANDALYE KULLANICILARI İÇİN K-MEANS ALGORİTMASIYLA AKILLI ŞEHİR ROTA UYGULAMASI

ÖZET

Akıllı şehirlerin gelişimi, mevcut teknolojik dünyadaki en kritik konulardan biridir. Akıllı ulaşım sistemleri, akıllı otobüs durakları, gerçek zamanlı araç konum takibi, rota planlama, kamu araçlarında gerçek zamanlı kalabalık tahminine kadar pek çok iyileştirme yapılan alanlardır.

Tez çalışmamızda, tekerlekli sandalye kullanıcılarına, İzmit şehrinde farklı otobüs güzergahlarını kullanma deneyiminin belirsizliğini azaltma ve bu güzergahlar hakkında bilinçli kararlar almalarına yardımcı olacak bir platform sağlamaya odaklanılmıştır. Akıllı şehirlerde, ulaşımla ilgili özelliklerden birini gerçekleştirmek amacıyla, İzmit'teki tekerlekli sandalye kullananlar için hem doğrudan hem de otobüsle ilgili güzergahları bilgilendiren bir sistem önerilmiştir. Sistem, yolun yürüme ve kaldırım bölümlerini haritalamak için ve eğimleri göstermek için renk kodlu çizgiler kullanır. Ayrıca şehrin General Transit Feed Specification (GTFS) verilerinden sorgulanan otobüs güzergahları için gri renkli bir çoklu çizgi kullanır.

Geliştirilen uygulamada, doğrudan yürüyüş yolları ve bölümleri, Google Haritalar API'ları aracılığıyla alınmış, kaldırım verilerinin eksikliği nedeniyle, Kuantum Coğrafi Bilgi Sistemi (QGIS) kullanılarak şehirde belirlenmiş bir alanın kaldırım haritası çizilmiştir. Haritalanan alan içindeki en kısa yol daha sonra Dijkstra algoritması kullanılarak hesaplanmıştır.

Ayrıca, K-Means kümeleme algoritması ve yıldızla puanlandırma sistemi, küme başına ortalama puana göre yolların akıllı önerilerini üretmek için kullanılmıştır. Yaş, cinsiyet ve tekerlekli sandalye tipi kullanıcılardan oluşturulan gruplar, farklı gruplar içindeki tutarlı kalıpları tasvir etmektedir.

Kullanıcılara sunulan öneriler, öneri motorunun önemli ve güvenilir olduğunu daha da doğrulamıştır.

Anahtar Kelimeler: Akıllı Şehirler, GTFS, K-Means Kümeleme, Tekerlekli Sandalye Yönlendirme.

A SMART CITY INTELLIGENT ROUTING APPLICATION FOR WHEELCHAIR USERS USING K-MEANS CLUSTERING

ABSTRACT

The development of smart cities is one of the most critical issues in the current technological world. Intelligent transportation systems are areas with many improvements, from smart bus stops, real-time vehicle location tracking, route planning, to real-time crowd forecasting in public vehicles.

Our research focuses on providing wheelchair users with a platform that would help them make informed decisions on routes to take and reduce the uncertainty of the experience of using the different bus routes in the city of Izmit. With the objective of being an inclusive smart city application, we propose a system that maps informative routes, both direct and bus-related routes, for wheelchair users in Izmit. The system uses color-coded gradient polylines to map walking and sidewalk sections of the path and a grey polyline for bus routes queried from the city's General Transit Feed Specification (GTFS) data.

Direct walk paths and the gradient of its segments are retrieved via Google Maps APIs. Due to the lack of sidewalks data, the sidewalk map of a designated area in the city using Quantum Geographical Information System (QGIS) was drawn. The shortest path within the mapped area was then computed using the Dijkstra algorithm in the pgRouting extension.

Furthermore, K-Means clustering and a star rating system was employed to generate intelligent recommendations of paths based on the average score per cluster. The created groups from the age, gender, and wheelchair type of users depict consistent patterns within different groups. Suggestions offered to users further confirmed that the recommendation engine is significant and reliable.

Key words: Smart Cities, GTFS, K-Means Clustering, Wheelchair Routing.

INTRODUCTION

Smart cities, accessibility, and inclusive societies are growing trends in the current world. Governments around the globe have employed different techniques in building better cities for all citizens, and in recent times, for the residents with special needs. Turkey is at the forefront of ensuring that its metropolitans are accessible to all citizens regardless of disability (Göçümlü, 2020).

Many cities and researchers have looked for different ways to improve the life of wheelchair users. Installation of ramps, addition of wheelchair friendly facilities like restrooms, innovations of wheelchairs that can climb stairs, and development of path finding applications specifically for wheelchair users are some of the diverse solutions that aim to improve the quality of life of the mobility-impaired people in our societies. The importance of having an inclusive society with accessible amenities for all citizens drives researchers and innovators to propose different solutions every day. We investigated the problems faced by the mobility-impaired citizens of Izmit as well as the solutions that are currently implemented for them. We also proposed a mobile application that would help recommend the accessible paths using the scores collected from other wheelchair users of the app.

Steep slopes, obstacles on the paths, inaccessible ramps and sidewalks, fights with bus drivers, and frequently missing the bus were some of the complaints we received from our interviewees. The Kocaeli Municipality provides dedicated phone lines for receiving complaints and suggestions from the elderly and disabled citizens as well as a wheelchair-friendly taxi service dedicated to the mobility-impaired citizens in a bid to ameliorate the lives of its elderly and disabled citizens.

We explored the development of an intelligent routing application that displays paths with color-coded polylines based on the slope value and also provides route recommendations to users of similar profiles based on the route scores submitted by users of the app. Our study is in line with other researchers (Gani et al., 2019; Mora et al., 2017; Prandi et al., 2017) who used the power of crowdsourcing to gather data on

accessibility issues and wheelchair-friendly paths to better route wheelchair users. These studies go a long way in helping mobility-impaired users traverse new areas without much difficulty since the widely used routing services in software such as Google Maps do not provide options specifically for them (Kozievitch et al., 2017).

Most of the studies that we found mostly provided direct paths in their routing features. We used the Google Maps Directions API to retrieve direct walk paths and the Elevation API to get the elevation value at each segment of the paths. We then tried to incorporate public transport vehicles in the routing using the city's official General Transit Feed Specification (GTFS) data. The bus routes were added to enable users to get path options that would assist them move uphill through the help of a public vehicle. On top of that, a map of the sidewalk and ramps in a selected area of the city was created since we could not find an existing database of these features. The routes fetched by our application gave precedence to the sidewalks before the normal walk paths retrieved from Google Maps APIs.

To produce the intelligently recommended routes, we first clustered the users using K-Means clustering algorithm using their age, gender, and wheelchair type. Then using the scores submitted by users in similar clusters, we calculated the average score of routes and presented the user with those that had at least a score of 3.0. The clustering of users was set to automatically trigger after a specific number of users joined the app so as to keep the clusters updated thus ensuring realistic and more accurate suggestions were presented.

Using four frequently travelled routes by our interviewees and a total of 20 users, we tested the clustering and route recommendation functionality of the application. Our analysis of the results prove that the automated clustering keeps the groups revised which trickles down to the suggested routes. The virtual collaboration platform provided by the app and the intelligent route suggestions presented to a user achieved the main objective of our research.

An overview of the meaning of smart cities, their characteristics, how different researchers and organizations index them, and a few examples of smart cities in both the world and Turkey is presented in chapter 1. Since our main research area touches

on accessibility in cities, we present our report on the accessibility issues faced by wheelchair users in Izmit, a literature review of past studies on the wheelchair routing dilemma, and the solutions currently in use in Izmit within chapter 2. After which chapter 3 details the technical details of the frameworks and technologies used in our research.

The fourth chapter then looks into the different modules of our system. We describe the importation of GTFS data into the database, the use of Google Maps Application Programming Interfaces (API) to get directions and elevation data, the mapping of sidewalks in a designated area of the city, pgRouting using Dijkstra algorithm, a brief description of the dataset, and the implementation of K-Means algorithm. Chapter 5 reports the results of testing with five users that includes an analysis of the clustering outcomes and examples of the recommended routes displayed on the application. Finally, chapter 6 presents our conclusions based on the results we found and provides insights and recommendations for future development. Our final intended product was a working mobile application that displayed routes with slope information and offered direct paths, bus routes, and recommended routes as results of the routing requests.

1. SMART CITIES

Smart cities are a trend in the technological world of our time. They are the visions of many governments, municipalities, organizations, researchers in all professions, and citizens in general. The world is working towards transforming cities into smart cities in all areas of life with a vision to improve the quality of life in a greener way.

1.1. Defining Smart Cities

To understand what a city needs for it to be called a smart city, we researched the definition of a smart city. While we could not find a clear and precise definition of a smart city, various researchers and organizations have expressed their interpretation of a smart city and what aspects of a city enable it to deserve the 'smart' label. Some smart city definitions that encompass a wide area of life mostly emphasize: the use of computerized technology; environmental protection or greener cities, integration of utilities like energy, water, transport; better safety, and security measures (Nevado Gilet al., 2020) while other definitions put more emphasis on certain aspects.

As cited in (Albino et al., 2015), there were more than 20 definitions by different researchers from as early as the year 2000. Researchers (Albino et al., 2015; Nevado Gil et al., 2020; Yin et al., 2015) compiled a list of the definitions by other researchers. Albino et al., (2015) revealed that the term 'smart city' was first used in the 1990s but was diffused in the early 2000s as an 'urban labeling' phenomenon. In later years, the term 'intelligent' was compared to the term 'smart' and smart was considered a more friendly term that is also used in marketing which also entails the inclusion of people which is not clear when using the word intelligent city (Albino et al, 2015). They referenced many terms that were investigated in the past like the ubiquitous city, digital city, virtual city, and knowledge city.

Numerous entities in Turkey also defined smart cities. According to İstanbul Bilişim ve Akıllı Kent Teknolojileri A.Ş. (İSBAK, İstanbul Information Technology, and Smart City Technologies Inc.), a company in İstanbul which works on Intelligent

Transport Systems (ITS) in the city, a smart city is defined as a sustainable city that integrates all city administration and stakeholders to improve the life of its residents while using the highest level of its technologies and resources effectively and efficiently (Güven, 2019). The Ministry of Environment and Urbanization also started a National Smart Cities Strategy and Action Plan for the years 2019 to 2022 (Güven, 2019). Under this plan, they too provided a definition of a smart city in their context. They described a smart city as a sustainable city that uses new and innovative technological approaches based on big data and experts to predict future problems and requirements for creating solutions that add value to life in the city (Güven, 2019).

From all these definitions, it is clear that the global aim of smart cities is to improve the quality of life of its citizens, protect the environment, devise innovative ideas, use the highest technological advancements effectively and efficiently, involve the government, stakeholders, and citizens in smart city solutions implementation, and most importantly, make the cities sustainable. In a smart city, all of the various subsystems are related and integrated, leading to smart cities being referred to as a 'system of systems'.

Pellicer et al., (2013) explained that the emergence of smart cities was due to the drastic population growth together with the shortage of natural resources important in the sustenance of life in the cities. Smart cities are a project to try and reduce the strain on nature while providing the essential needs for life in the city to all living things (Pellicer et al., 2013). This is one of the major reasons why the world is tirelessly working on innovative ideas to make cities smarter. Smart cities also aim at reducing environmental pollution by minimizing the use of natural resources and promoting renewable energy use (Pellicer et al., 2013) like solar and wind power.

1.2. Characteristics of Smart Cities

Without a standard description of what a smart city is and what it entails or how it is measured, researchers published categorizations of components that make up a smart city. Most of the elements presented touch on common areas of life which demonstrates the importance of smart cities on the quality of life in general.

Similar to the compilation of smart city definitions presented before, Albino et al., (2015) also compiled the categories given by various researchers from as early as 1999. They reported that the reviewed researchers' common factors include interconnectivity, transparent governance, innovative creations, the inclusion of urban residents, and natural environment protection.

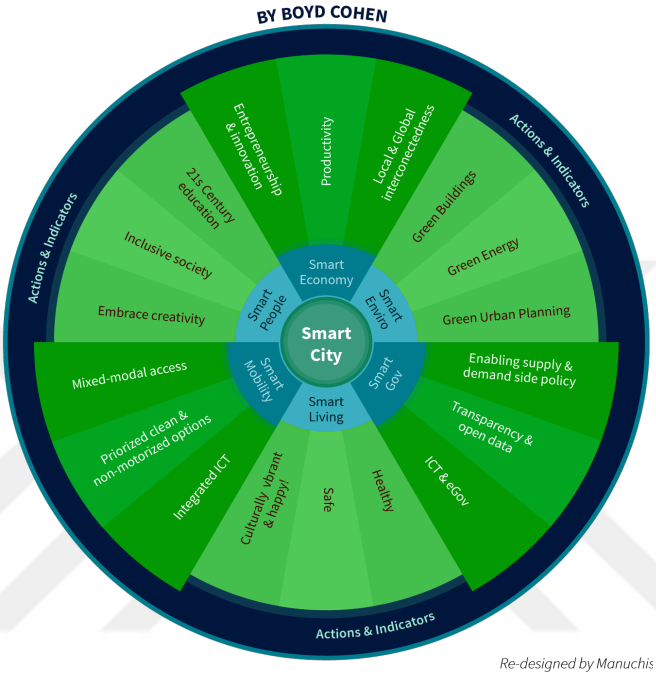


Figure 1.1. Boyd Cohen's smart city wheel (Cohen, 2012)

As Boyd Cohen explained in (Cohen, 2012; IoMob, 2018), there was no specific definition of smart cities or what they should be for a long time. Hence his decision to publish a graphical model which he called the 'smart city wheel' depicting what he believed encompassed the main components of a smart city in 6 major categories: smart economy, smart environment, smart government, smart living, smart mobility, and smart people as depicted in Figure 1.1. These characteristics are consistent with those developed by the European Smart Cities Research group (Giffinger et al., 2007). Hence definitions and details of each group overlap and can be used to complement each other. In this research, we will use these six categories to categorize better and visualize the diverse applications in smart cities of the world and Turkey.

1.2.1. Smart economy

Smart economy describes all aspects of a city's economic growth and prosperity in local, national, and international markets. It focuses on improving the business climate through new creative innovations, the attraction of start-ups and investors, utilization of technological solutions, productivity, and flexibility of labor markets and trademarks (Giffinger et al., 2007; Pellicer et al., 2013). Smart economies experience the loss of some jobs and the creation of new jobs. As has been experienced in the past, jobs lost during the Industrial Revolution consisted of those that made humans redundant since their jobs could be automated using machines. The same is predicted of the technological revolution that may come due to smart city implementations. More jobs with regards to technology like robotics, data scientists, software engineers, technicians, Artificial Intelligence (AI) friendly health workers, AI-driven urban planning, and AI-centric lawyers and laws are foreseen (URL-25).

1.2.2. Smart environment

Protecting the natural environment, reducing pollution, managing natural resources in a smart way, and achieving energy efficiency are some aspects that constitute a smart environment (Giffinger et al., 2007; Pellicer et al., 2013). They also compose of governments striving to manage the man-made developed and natural environment in a bid to improve the quality of life (URL-6). Most smart environment initiatives focus on energy efficiency, resource management, smart city planning, and the creation of a resilient community (URL-6).

1.2.3. Smart government

One of the major importance of smart cities is to improve the quality of life of its citizens. This can be achieved by making the decisions and steps being undertaken by the governing body open to discussion and participation from its citizens as well as public and private stakeholders. Transparency, thus, would play a role in ensuring that the government offers quality, diverse, and a large variety of services to its citizens and stakeholders (Giffinger et al., 2007; URL-6; URL-25; Pellicer et al., 2013).

1.2.4. Smart living

Smart living encompasses culture, improved healthcare, better education systems, tourist attractiveness, social cohesion, safety and security, and better housing quality. Smart cities use the Internet of Things (IoT) based on Wi-Fi services to improve accessibility and social engagement (URL-6). Most smart living initiatives leverage the power of sensors, the IoT, and big data available to provide different kinds of solutions such as automatic emergency requests, remote monitoring of home appliances, easily accessible city information to visitors, automated services, and remotely accessible educational programs.

People living in smart cities need to understand and be able to use the technical services and smart infrastructure put in place for their betterment. This can only be made possible if the citizens are aware of the services installed. To raise awareness, most countries, cities, organizations, and even educational institutions hold congresses, symposiums, and forums that are open to the public and broadcast to the citizens to inform them of smart solutions and include them in smartening their cities. Most cities advertise their works and even enforce the use of smart solutions by making it a requirement for residents to use the technological systems. The best example of this scenario is online appointment systems used in all areas, including government offices, hospitals, banks, and companies. These institutions usually require someone to book a slot using their websites or mobile applications before showing up at the offices. Hence makes it imperative for residents to learn and use smart solutions.

In this regard, (Lytras and Visvizi, 2018) conducted research to find out the users and their perspective of smart city services. Their research targeted a demographic that had the technical knowledge and were avid users of the smart city services. They reported that even though the participants in the exercise were tech-savvy and well-informed people in smart city solutions, the feedback given had a lot of serious concerns especially in the safety, accessibility, and efficiency of the services. This demonstrates the importance of including citizens in the implementation of smart city initiatives so as to make sure the implemented solutions tackle issues in an acceptable manner, and efforts do not go to waste.

1.2.5. Smart mobility

Transport is an essential service for the prosperity of a city. However, most transportation modes used in the 20th century use a non-renewable natural resource and contribute to noise and most of the carbon dioxide emissions to the atmosphere which harms the environment. Smart mobility aims at reducing this harm and protecting the environment as well as reducing congestions and road accidents (Pellicer et al., 2013). Furthermore, smart mobility involves efficient, cheaper, faster, environmentally friendly, customer-centric, and inclusive transportation options for residents, businesses, and visitors or tourists (URL-6.). New inventions in smart mobility include electric cars, bike-sharing, carpooling, (URL-6) smart traffic lights, automated highway payment systems, and even solar-powered roads.

1.2.6. Smart people

To accompany the evolution in smart cities, citizens need the appropriate education to enable them to support the creation of an accessible and inclusive environment (URL-6). Education is important for people to provide them the skills and thinking capacity to encourage innovativeness, participation in public life, and flexibility and openness to the outside world (Giffinger et al., 2007; URL-6.; Pellicer et al., 2013) This also entails that citizens of smart cities need to be equipped with Information and Communication Technologies (ICT) training like programming, engineering, and mathematics (URL-25). We can see these trends in academia where all departments in universities have made it mandatory for students to take and pass a course in C++ programming language before graduating. Other countries also offer Microsoft Office packages training to provide the basic skills of using a computer for any task.

1.3. Indexing/Ranking of Smart Cities

How do we know if a city has become smart? And how can we tell the level of smartness of cities? These are some of the questions tackled by researchers of smart cities. Indexing or ranking of smart cities lacks a standard worldwide formula. However, researchers and institutions have devised their own measurements and presented rankings of smart cities using different formulas in a bid to help devise a standard measure. The significance of ranking smart cities can have different

interpretations and effects based on perspective. Rankings published by different entities focus on different aspects of a city since they use different categories and indicators. Hence interested entities may pay more attention to the overall rank itself yet categorical ranks and details in the ranks should be more important (Giffinger et al, 2007). On the other hand, city rankings draw public attention to issues in their areas which stimulates discussions on regional developments to better decide where to focus time and resources. (Albino et al., 2015). They force municipalities to make their plans and decisions transparent since most rankings depend on readily available data and also contribute to foster the need to learn in (Giffinger et al., 2007).

European Smart Cities research group published their first ranking of medium-sized cities in Europe in 2007 (Giffinger et al., 2007). In their research, they selected European cities: with a population between 100,000 and 500,000; with accessible and relevant databases; and those that are of medium-size but with a dense population. From the criteria, they remained with a list of 70 cities. Luxembourg emerged as the top smart city in the 2007 European medium-sized smart cities ranking, with an above-average score in each of the six characteristics, with the smart economy being the topmost followed by smart people (Giffinger et al., 2007).

A smart city ranking index called Cities In Motion Index (CIMI) was designed by IESE Business School, which is the business school of the University of Navarra, Spain (Alkadhim, 2016). “The CIMI aims to help the public and governments to understand the performance of 10 fundamental dimensions for a city: governance, urban planning, public management, technology, the environment, international outreach, social cohesion, mobility and transportation, human capital, and the economy” (Alkadhim, 2016, p. 8). This index was used to rank cities across the world by IESE Business School and other researchers. In 2019, the CIMI merged public management into the governance dimension since their definitions overlap (Berrone et al., 2019). In the 2018 ranking reported by (Berrone et al., 2019), London, New York, and Amsterdam topped the list with İstanbul being in the 118th and Ankara in the 135th ranks out of 174 cities.

In (Nevado Gil et al., 2020), researchers used the CIMI data of 73 ranked European cities to rank cities in three groups of highest, medium, and low smart city levels using

the 10 CIMI dimensions of a smart city. Their research focused on analyzing political factors that influence the development of smart cities using multivariate analysis. They investigated the significance of the region a smart city belongs, the gender of the governing person, and the ideology of its political rulers on the development of the city. Their report showed that cities in the western region of Europe and those governed by women achieved better ranking, while the ideology of the political body did not seem to have a definite impact on the city development.

In the year 2016, a project was undertaken by top organizations in Turkey to analyze the energy, water, and transportation fields within 30 of Turkey's metropolitan municipalities and provide an overview of the state and roadmap of the development of smart cities in the country (URL-14). They reported challenges faced in the implementation of smart city applications that included financial constraints, non-collaboration with NGOs and educational institutions, and non-inclusion of citizens. Important factors detailed in the report encompass the need for innovation, skilled resources in ICT, security and privacy consideration, infrastructure for Geographic Information Systems (GIS), and the inclusion of citizens. The researchers also observed that the most common applications that were initially employed by every 2 out of 3 municipalities were in the transportation sector where traffic monitoring, electronic payments, and smart bus stops were the top applications.

Apart from the ranking systems and research mentioned above, Albino et al., (2015) reported techniques used by various entities in ranking smart cities. They concluded by expressing their belief that even though the formulas presented endeavored to create an all-encompassing ranking system, the assessments should be tailored to a city's vision since they all have different priorities and capabilities in achieving the smart city objectives.

1.4. Examples of Smart Cities in Turkey

Turkey is one of the countries in the world that attributes great significance to smart cities. The Minister of Industry and Technology, Mustafa Varank, expressed his belief in technology to enable people to live happily in cities (URL-21). He also explained

that an economic analysis of smart cities on the global economy would be massive hence the acceleration in their study of smart city applications (URL-21).

Smart city projects in Turkey were first launched as early as the year 2000 (Akiner, 2016) and have been gaining momentum ever since. Cities in Turkey are transforming into smart cities by implementing different diverse projects. Since it is not possible to list every project implemented in every city under the smart city vision, we looked at the latest projects published in the bulletin provided by the National Smart Cities Strategy and Action Plan, which was a plan devised by Turkey's Ministry of Environment and Urban Planning shaped by the public, private, and academia sectors with common viewpoints to ensure that projects are prioritized and implemented in the most productive way and in line with the determined policies under the smart city ecosystem (Bilgin, n.d.).

From the bulletin published in 2019, several smart city projects that had a significant impact on the livelihood of residents were analyzed in the six categories of smart cities that were described in Boyd Cohen's smart city wheel (Cohen, 2012). It was observed that most projects implemented in Turkey's smart cities were related to mobility and the environment. While the category with the least implementation was the smart economy category.

Under smart mobility, most cities had implemented more or less similar projects with a few extra projects in some cities. One of the most common ITS was the mobile application, and in some cities its website app, meant to simplify public transportation in the city. These apps are available for free and display all relevant public transportation information for the specific city it is designed for. Some cities with these apps include İstanbul (IBB CepTrafik), Ankara (EGO Cepte), Konya (ATUS), and Kayseri (Güven, 2019) among many others. In liaison with the apps, cities such as Mardin, Eskişehir, Gaziantep, Yalova, Kars, Konya, Kayseri, Gaziantep, and Kahramanmaraş (Güven, 2019) added smart stops that had Quick Response (QR) codes or stop codes which when searched in the app, displayed information like the nearest buses and their estimated time of arrival to the stop. The public transport vehicles were also equipped with e-payment solutions in cities like Manisa, İstanbul, Konya, Antalya (Güven, 2019). For pedestrians, İSBAK explained the crosswalk

request button which had illumination, audio, and haptic feedback specially designed for the disabled citizens.

The Electronic Detection System (EDS) is a system that uses sensors, cameras, and intelligent software to detect violations and recognizes license plates of offenders with an aim to instill positive driving culture thereby reducing road accidents while increasing the security of people and property as explained by(URL-4). It was adopted in many cities like İstanbul, Konya, Antalya, and Gaziantep (Güven, 2019). URL-4 also explained the fully Adaptive Traffic Management System (ATAK) which used cameras to monitor traffic density and intervene with calculated optimum signal times to help ease traffic adopted in cities such as Mardin, Konya, İstanbul, Ankara, Kayseri, Bursa, and Kahramanmaraş (Güven, 2019) and others. They further described the variable Light-Emitting Diode (LED) screens used to redirect, inform, and warn motorists in real time.

Other solutions documented included intelligent parking systems that displayed car park occupancy status, offered e-payment options, used number plate recognition for automatic barrier control, and mobile apps that directed motorists to available car parks. Furthermore, the taxis in İstanbul used smart signals with color-coded occupancy information, parts of the tramway line in Konya used non-catenary lines to help preserve historical monuments, and the İstanbul Grand Airport (IGA) was equipped with self-check-in systems, intelligent lighting for aircraft guidance, and biometric screening at passport control points (Güven, 2019).

To protect the natural environment, Turkey's smart environment solutions consist of: using renewable resources, recycling waste, and constantly monitoring the environmental conditions in cities. The 2019 bulletin (Güven, 2019) cited some examples of smart environment solutions. Güven (2019) described smart renewable energy solutions like the one in İstanbul which made use of renewable energy through a floating solar power plant installed on Lake Büyükçekmece while in Antalya, solar power plants in agriculture that were set up in Döşemealtı-Mellidağ took advantage of the geographical location of Antalya to use solar energy to provide free electricity to over 7 thousand farmers in 47 irrigation units. He also explained Antalya's electric generating stadium equipped with 12 thousand m^2 of solar panels which met the

electrical needs of 575 homes and prevented the emission of 1200 tons of CO_2 into the atmosphere and the smart solar poles installed in 19 points in parks and public areas in Kahramanmaraş that used solar panels for energy to provide free internet access to residents.

Güven (2019) also described a few solutions implemented to instill a recycling culture and environmental awareness in cities such as İstanbul which has smart recycling containers at metro stations and the zero-waste program in Ankara where cartridge-toners, fluorescents, waste batteries, paper, plastic, glass, and metal are collected separately. In most cities, Güven (2019) explained the energy generation facility from domestic waste incineration. He gave examples of the facility in İstanbul which generated 77MWs of electricity through incineration of 3,000 tons of waste daily; in Ankara, an integrated solid waste management system collected around 5,500 tons of waste daily and produced around 1,289MWs of energy daily from the disposed waste; in Konya, electricity generation from methane gas in a solid waste plant met the daily electricity needs of 26,000 homes, and a greenhouse of 1,200 m^2 was installed to take advantage of the heat produced resulting in the production of around 30 tons of tomatoes annually; and in Antalya, a solid waste integrated recycling and disposal facilities separated around 3000 tons of domestic waste daily and met the electricity needs of 60,000 households.

Examples of environmental control centers described by Güven (2019) are found in İstanbul and Konya which had air quality monitoring, waste tracking, noise tracking, fuel control, and vehicle tracking modules allowing the monitoring of pollution levels and instant access to the data enabled detection and solution of urban and regional environmental problems. Güven (2019) also gave an example of Ankara's Electric Energy Tracking System (ETS Elektrik Enerji Takip Sistemi) that monitored energy quality, active-reactive power, fault conditions, and instantaneous voltage-current values in the municipal facilities. He also talked about the intelligent lighting systems in parks, roads, and gardens around the city of Antalya which used LED lights for efficient use of resources which contributed to up to 80% of energy savings in 2018.

When it comes to intelligent systems that help preserve water as well as provide better and faster distribution of water to residents in the city, Güven (2019) described the

intelligent water management systems implemented in cities such as Ankara, Gaziantep, and Kahramanmaraş which also monitored tanks, water purification facilities, valves, pressure, water quality, and the water distribution network at large for any faults and had remote counter reading systems. Cities such as Antalya, Kayseri, and Gaziantep also had intelligent watering systems in public parks and gardens. Güven (2019) explained that the systems monitored the moisture condition of the soil and in the atmosphere, and when in need of water, it checked the weather forecast and devised a plan to water the soil in accordance to the weather forecast, the needs of the soil and plants, and automatically operated the water valves at the time of watering hence reducing water wastage and protecting the quality of the soil. Similar to this, is the e-desen project in Konya which analyzed soil and climate, identified the most suitable products to grow, stored this information in a central database, and informed citizens via a website (Güven, 2019).

Shared bicycle systems for the public is considered a green transportation alternative in smart city projects. This system can be classified under the ITS as well as smart environment solutions. The intelligent bicycle system was implemented in many cities in Turkey such as Antalya, Izmir, Erzincan, Kocaeli, Yalova, Konya, and Kayseri (Güven, 2019). The cities have dedicated bicycle lanes on roads and the systems use the transport e-cards to take or return a bicycle from a rental point (Güven, 2019).

Smart living solutions mainly focus on making an individual's life easier, safer, and better. The solutions focus on safety and security, healthcare, cultural exposition, and shelter. In terms of security, video surveillance cameras have been used widely in cities and buildings all over the world. The surveillance in smart cities also includes public areas as parks and roads. Güven (2019) highlighted some of the cities that have implemented security-enhancing projects. He gave an example of the smart park projects in Ankara and Antalya that detected issues such as theft, kidnapping, and suspicious-looking items and sent instant notifications through a Long-Term Evolution (LTE) wireless system to the authorities in the operation center. As for Antalya, Güven (2019) explained the trusted circle project which provided wireless low-energy wristbands to children, the elderly with Alzheimer's, and even pets, then they were monitored using solar-powered poles in parks which sent a notification to the mobile

application when it detected a wristband leaving the wristband's trust circle hence allowing free movement without a lot of worries.

Smart healthcare solutions also provide a sense of safety for smart city citizens. Güven (2019) pointed out some healthcare solutions that helped health workers and families keep track of patients remotely. He explained a chronic patient tracking project which measured patients' glucose, blood sugar, and pulse values remotely and sent the data via a mobile application to a central database where health workers checked the values and contacted the patients in case of more analysis. The project also distributed a panic button to patients with chronic illnesses which instantly called for an ambulance and informed the family at the push of the button. In Bursa, Güven (2019) talked of the 'love chip' project that made it possible for citizens with Alzheimer's and mental illnesses who carried the chip to communicate freely with their relatives who could check in on them at any time of day in real-time.

Some cities use smart building systems to aid in monitoring the elderly citizens. In Kahramanmaraş for example, Güven (2019) pointed out that the smart elderly care and coordination center in the city used remote smart tracking systems installed in the homes of elderly people who lived alone. The elderly were given a handheld button that connected to call centers directly and provided communication with personnel on duty. The smart tracking systems in the homes used sensors to automatically detect events such as fires, gas poisoning, smoke, and water overflow then instantly transferred that data to the call centers who simultaneously informed the relevant units such as ambulance, fire brigade, and police hence contributing to a life of peace and security for the elderly (Güven, 2019).

In the cultural and tourism aspect of smart cities, Güven (2019) highlighted the 3D mobile tourism atlas project implemented in Bursa which introduced the historical, cultural and natural values of Bursa using 3D models, photography, and introduction text of 100 places around the city plus video adverts of 10 other places. Kahramanmaraş had a city guide which was a website that advertised the city by providing information on events, local products, meals, and accommodation options (Güven, 2019). And in Konya, Güven (2019) pointed out the "Mevlana ve Mesnevi"

mobile app that shared the most famous works and quotes of Mevlana, known as Mesnevi, in 20 different languages as well as photographs of the Mevlana museum.

Smart cities are also enhancing services to be more inclusive and have more accessible solutions. In Antalya for example, Güven (2019) shared the audio steps project implemented in the municipality city hall. He explained that the project entailed the use of audio commands to enable the visually and hearing-impaired citizens to navigate the facility with limited need for support. Improving the lives of animals in the city was one of Konya's visions. Güven (2019) explained this by providing examples of projects in the city that improved animal care in Konya. He talked of the volunteer animal friends project where rehabilitated dogs were appropriated to verified volunteers and an e-paw mobile application that was used to monitor the health of the dogs, their vaccinations, food supplements, and living conditions.

In the spirit of having a transparent government, cities in Turkey have developed mobile applications and websites that offer information on the undergoing and planned works, plus offer a forum where citizens can submit their opinions, views, feedback, and complaints directly to the municipality. Güven (2019) explained that the municipality applications like the ones in Ankara, Bursa, Kayseri, and Kahramanmaraş provided access to all city services and information within the scope of transportation, city guide, municipal studies, and blue table applications. He also explained that cities like Ankara and Kahramanmaraş had a cemetery information system mobile application (MEBİS, Mezarlık Bilgi Sistemi Mobil Uygulaması) which had daily burial data, previous years' burial information, and easily searchable locations for graves. In Bursa, however, Güven (2019) pointed out that the e-municipality application shows the definitions, transactions, and reports, carried out by the departments of 34 main modules. The Infrastructure License Control Program (Altyapı Ruhsat Denetim Programı [ARUDEP]) enabled all applications for infrastructure excavation licenses and follow-up procedures in Bursa to be done via the web-based GIS system. Furthermore, Bursa had an online advertisement auditing system that calculated signage taxation transactions online, reliably, quickly and efficiently, and sped up business and transactions in the city (Güven, 2019).

Smart cities in Turkey also provide free internet access for its residents in public areas. In Antalya, Güven (2019) mentioned the city information kiosks which had been set up in 20 different areas of the city, were air-conditioned, had internet access, had smartphone charging points, and offered information and announcements about the city to both residents and visitors.

In a bid to make citizens of cities fit into the smart people definition, cities have employed different means of providing the needed education and skills in creative ways. For example in İstanbul, Güven (2019) pointed out "hayal kart" which was a gamified project to help children learn and love coding as well as increase their productivity. He also explained the living lab in Başakşehir, İstanbul which was a center opened to the public for research, innovation, and experience where new products and services were developed and tested with real users. While in Konya, he mentioned the Konya science center equipped with intelligent building systems and established to provide a fun and interactive environment to stimulate creativity, curiosity, togetherness, and learning in its citizens of all ages.

Alongside providing a center for education and mentorship, İstanbul also has Zemin İstanbul which is a social and institutional structure that aims to attract and produce innovative products and services with citizens, instill an understanding of smart cities in the people, and provide professional advice about commercialization to entrepreneurs (Güven, 2019). This project focused more on entrepreneurship and commercialization efforts hence can be classified under smart economy solutions.

The examples provided in this section are mainly concentrated on the solutions published in the 2019 White Bulletin. There are very many examples that have been implemented, currently being implemented, and planned for implementation that have not been covered. More examples can always be found on the annually published bulletins in the cities as well as the country.

1.5. Examples of Smart Cities in The World

Most countries of the developed world have, at one time or another, considered the possibility of developing smart cities. While some smart cities have been designed from scratch, others have gradually evolved from normal cities to smart cities.

According to Calderoni (2015), smart cities are close to becoming a common reality, especially with the various advancements that have been made in the fields of traffic management, smart payments, automated billing services and automated waste management. The advancement of the IoT has also improved the smartness of individual homes, which in effect increases the adaptability of fully smart cities.

Cities such as Portland in Oregon, USA and Santander in northern Spain are examples of normal cities that are being deliberately transformed into smart cities. Portland, with a population of half a million inhabitants, has been a model city in the United States for sustainable development (Pellicer et al., 2013).

Figure 1.2. 20 minute neighborhood concept (URL-22)

In the first iteration, they enhanced the mobility and transport systems to reduce emissions to the environment. New tramways were created and an extensive network of bike lanes. In the second iteration, they focused on the construction of sustainable buildings with minimal environmental impact and energy savings. Finally, they implemented the concept of '20 minute neighborhoods', where all basic services can be reached within 20 minutes, thus reducing traffic and air pollution (Pellicer et al., 2013). The concept can be summarized by Figure 1.2 which shows the main elements incorporated such as health, educations, sports, transport, and housing facilities.

In Santander, the city council had launched measures to turn it into a smart city. These measures included smart traffic management, interactive mobile apps to support tourism, tracking and monitoring of people with disability or disease, emergency alerts

to citizens, monitoring of security via a centralized system and an intelligent waste collection system.

Smart cities are made smarter by extensive use of ICT in all public facing services. By improving ICT infrastructure, these innovative architectures bring a more efficient service delivery model (Pellicer et al., 2013). Some of the services that, by just being properly automated and improved, can transform a city to smart status are: video control and intelligent urban surveillance, security and geocoding in emergency call service, energy saving, remote control of industrial facilities, remote control of utility networks, logistics, safety at work, personal and community healthcare, waste management, public transport, automated traffic control, intelligent parking infrastructures, monitoring of air and noise pollution, wired and wireless broadband internet, NFC and other integrated payment methods, tourist facilities and secure digital voting for smart governance (Calderoni, 2015).

Transport is also essential in today's society, in which we move while performing functions vital to the advancement of society (Pellicer et al., 2013). In Zaragoza, Spain, a wide sensor network controls approximately 90% of urban routes (Calderoni, 2015). There, a centralized control point monitors the entire city's traffic in real time and helps the government to improve road networks with adequate policies. To improve transport systems in a city, one of the most important aspects is smart parking. By automating parking services, people can move around faster and more efficiently, and revenue collection is centralized leading to increased income to the city, which in turn leads to better services being provided to residents.

According to Al-Turjman and Malekloo (2019) smart parking systems rely on analyzing and processing the real time data gathered from vehicle detection sensors and Radio Frequency Identification (RFID) systems in parking lots to report the status of a parking slot. Smart parking systems can also be combined with other systems in order to improve the accessibility and efficiency of a city's traffic system. In Santander, Spain, for example, a system is in place to monitor, control and manage traffic in various aspects such as information about parking (slot availability, disabled slots, etc.), road traffic management for emergency lane clearance and proposal of alternative routes to avoid traffic (Pellicer et al., 2013). Figure 1.3 displays an example

of the smart parking systems implemented in Santander. The systems use radar technology to detect the occupancy status of a parking slot and communicates with drivers via ethernet (URL-18).

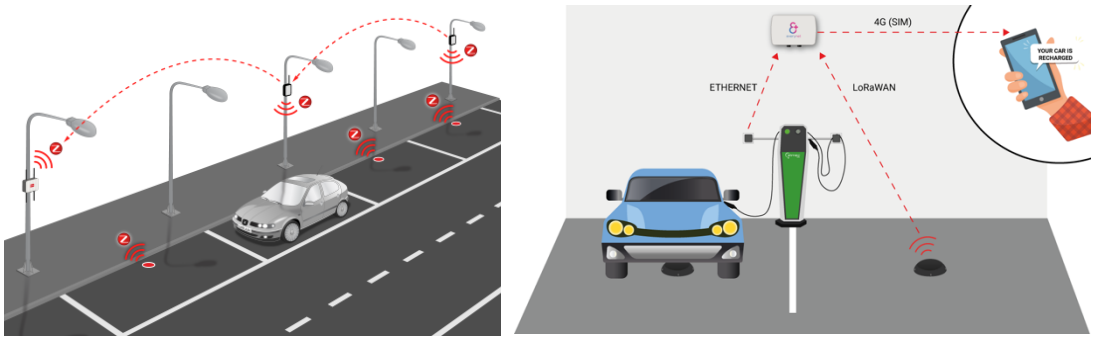


Figure 1.3. Smart parking systems in Santander (URL-18)

According to Pellicer et al., (2013), the parking system implemented in Copenhagen is fully automated and integrated. Drivers can park their cars anywhere in the city and get automated bills at the end of every month. This system used Automatic Number Plate Recognition (ANPR) systems to monitor all roads and public parking spaces, and it calculated the duration a vehicle has been parked at each available slot. Drivers who had good driving and payment records earned discounts in their parking rates, while drivers who committed traffic offences such as running red lights, parking in prohibited spaces or over speeding got steeper rates. This system has increased accountability and safety, since drivers are more careful while driving in order to enjoy the discounted rates.

In Malaga, Spain, the city provided free car charging stations for all residents who owned electric vehicles (Pellicer et al., 2013). In this program, owners of electric vehicles were also exempt from parking levies in the city, and got tax breaks. This increased the adoption of electric vehicles, greatly decreasing air and noise pollution and increased the standing of Malaga as a tourist destination (Pellicer et al., 2013). Cities such as Paris, Copenhagen, Stockholm, Frankfurt and Vancouver have implemented free or cheap bicycle exchange programs (Pellicer et al., 2013). In these programs, citizens can pick a bicycle from a parking station, ride it to another part of the city and then leave it in another parking station. These programs improve the overall physical and mental health of citizens, while reducing air pollution and traffic in the cities.

In a city with a smart economy, all other aspects of life in the city are greatly improved because of increased automation in systems such as payment systems, utility management and revenue collection. According to Mahizhnan (1999), Singapore is moving into a smart economy by adoption of electronic commerce (e-commerce). For example, Singapore's legal system is fully configured to recognize electronic identification for individuals, contracts and transactions. From as early as 1996, Singapore already set up the electronic commerce hotbed to support e-commerce. It lists guidelines that have guided the city's adoption of e-commerce. Among these guidelines, the most important one states that the government, through joint ventures with the public sector, should expedite the growth and development of e-commerce (Mahizhnan, 1999). This ensured that any projects and programs that are geared towards e-commerce are prioritized and get all the required approvals without delays.

In New York City, USA, the city launched a program called 'Startup Development for Social Web' (Pellicer et al., 2013). In this program, any technological innovations that were geared towards financial startups were exempt from some levies and taxes, and were usually placed at the top of the queue in any approval processing. Since New York, and especially Manhattan, is a financial hub, this increased investment in the financial startup scene, greatly increasing the ability of normal residents to access advanced financial services such as stock and forex trading. (Pellicer et al., 2013).

In order to improve the quality of life for its residents, Singapore initiated the Singapore one network for everyone (Mahizhnan, 1999). This was an initiative jointly driven by the governing council and Singapore's National Science and Technology Board. In this initiative, the Singapore ONE network offered more than a hundred and twenty web and mobile applications to more than 100,000 users. Mahizhnan (1999) stated that the official support for this system had increased the adoption of automated systems instead of manual ones, a process that has seen even simple establishments such as food kiosks acquire automated teller systems that are integrated to the owner's bank accounts and the tax collection agency's systems.

One of the key drivers for smart cities is reduced pollution and a cleaner environment. In Vienna, Austria, the city implemented a smart green street lighting program that used very low energy consuming LED street lights that are solar powered (Pellicer et

al., 2013). Vienna also employed a green program where new buildings that were energy efficient got quicker approvals.

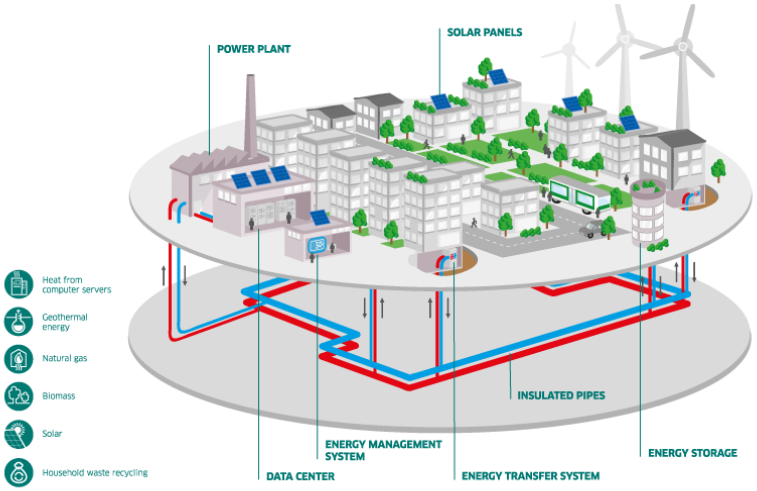


Figure 1.4. An example of a centralized heating system (Engie, 2013)

Buildings with passive heating/cooling, solar lighting and those that featured improved natural lighting usually got tax breaks and tenants in such buildings paid reduced levies (Pellicer et al., 2013). This increased the demand of green buildings and led to an explosion in the conversion of existing buildings to be more energy efficient. Pellicer et al., (2013) also gave the example of Barcelona, Spain, a city that implemented a centralized heating and cooling network. In this system, hot water and steam for heating buildings came from a centralized network which drew its heat from incinerators that burned the city's waste, as seen in Figure 1.4. This combination of waste management and heating led to zero demand for landfills, and the improvement of waste management.

In some cities, such as Paris, France, Manila, Philippines and Cyberjaya, Malaysia, a centralized cooling system is in place as seen in Figure 1.5. As mentioned in (Engie, 2013), delivering conditioned air services to buildings connected to a central cooling network utilized chilled water production and distribution facilities that operate as a closed circuit, supplying the chilled water to the connected homes and back to the production plants for further cooling.

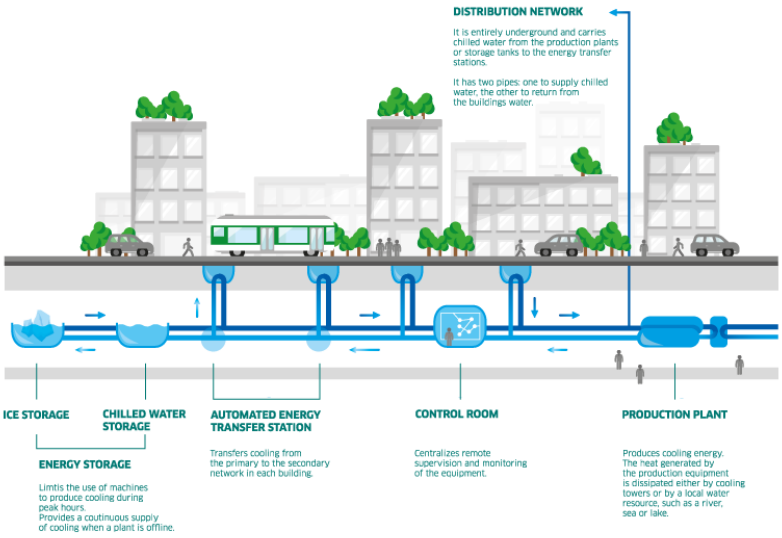


Figure 1.5. An example of a centralized cooling system (Engie, 2013)

The underground train system in London, England is one of the oldest in the world. It been modernized and optimized to ensure that there are zero emissions in the entire system (Pellicer et al., 2013). The London underground runs completely from clean energy, coming from wind electricity and electricity generated from waste burning incinerators. The lighting systems used in the London underground and in the trains used energy saving LED bulbs, that had light sensors which adjusted the brightness automatically according to the ambient light levels.

2. RELATED WORK

2.1. Accessibility Issues for Wheelchair Users

The struggle to improve the accessibility of smart cities is a growing phenomenon. But what exactly is accessibility in cities? Accessibility refers to the ability of all individuals regardless of any disability to access information and resources any time and without much difficulty or help (Tavares et al., 2016). It is the main key to making cities and technologies inclusive while improving the lives of people. Accessibility problems refer to the issues faced by people with disabilities or special needs while going about their normal daily activities. These problems do not come naturally to a person without disabilities nor special needs. Researchers, therefore, use other means such as interviewing the people with the demographic under examination and reporting their findings to help other researchers, the general public, and the problem-solvers understand them.

When it comes to mobility, most people with special needs such as the elderly, blind, deaf, and wheelchair users all have different experiences. Of these groups, mobility is most challenging for wheelchair users since they are constricted to a wheelchair. Wheelchair users have mobility limitations that depend on factors such as the surface of the path, the slope of the path, the width of the path, the abilities or power of the wheelchair itself, the obstacles on the path, and the shape of the trail itself.

Many wheelchair users in societies feel left out and feel that their needs are ignored as reported by F Bromley et al., (2007). One of the mobility-impaired students in the computer engineering department of Kocaeli University expressed feeling that the ramps and curbs installed in the city center were designed for mothers with baby strollers and not wheelchair users. It is also apparent that the needs of wheelchair users differ (F Bromley et al., 2007) depending on factors such as the type of wheelchair, the age, weight, energy of the wheelchair user, and the power of the wheelchair itself. Some wheelchair users also feel discomfort while in public places due to the gazes they get from others (F Bromley et al., 2007). Some of them also complain about the

heavy doors, cluttered aisles, inaccessible shelves, and narrow checkouts in shops (F Bromley et al., 2007).

We sought to find out the mobility problems faced by wheelchair users in the city of Izmit that is in Kocaeli, Turkey. We interviewed a student from the computer engineering department at Kocaeli University. From our exchange, we found out that the most problems faced are related to slopes because Izmit is a hilly place. For example, they expressed that when visiting an area for the first time, they have to find a path through trial and error or use google maps to find alternate paths. In order to stay out of other motorists' way and be safe, they would decide to use the sidewalks on roads but were frequently faced with problems where a car was parked obstructing the ramp that should have been used to climb the sidewalk. Most times they encountered ramps and sidewalks that were not accessible either due to obstacles, damages on the surface, its narrowness, or its steepness.

In situations where crossing the road or taking a turn was inevitable, they explained that the situation became a challenge due to the absence of ramps at the location forcing them to go back to where they started and find an alternative route mostly by rechecking the mapping applications. They also recounted how frequently they encountered steps on the sidewalks on slopes since most steps on roads were constructed to assist pedestrians. Due to all these problems, they stated that they developed a habit of checking the photos of the roads, ramps, entrances, and sidewalks of any areas that wheelchair users planned to visit beforehand to check their accessibility.

Our interviews further revealed that manual wheelchair users had difficulty going up a steep incline and on a steep decline, they had a high probability of sliding which would sometimes not be possible to stop. We were also informed that since most electric wheelchairs have good brakes, the brakes helped against the non-controllable sliding.

With regards to the public transportation in Izmit, wheelchair users expressed the common brawling with the bus drivers that was mostly because the drivers lamented opening the wheelchair ramp installed on the bus, and drivers would not tilt the bus to

make sure the ramp was not too steep. They recollected how drivers, would decide that a faster solution was to carry the passenger by their wheelchair into or out of the bus, which made the impaired feel belittled and sometimes end up with broken parts of their wheelchair in the process. They further conveyed their dislike when bus drivers stopped too far from the sidewalk, making it more difficult for wheelchair users to board.

They continued explaining that they would sometimes miss the bus while waiting at the bus stop because the driver was unable to see them signaling for the bus to stop. One user stated he would prefer a direct communication line with the bus instead of having to call the municipality office to inform the driver of their waiting. For wheelchair users to be able to board a crowded bus, they need to call in advance to book a place on the bus. They check the approaching bus's number through the e-komobil application and call 153 to inform them to notify the driver to reserve a spot for them. The interviewees complained that without doing this, drivers do not allow them to get on the bus, and they end up waiting long hours to catch a bus.

These issues end up discouraging wheelchair users from wandering outside their neighborhoods and comfort zones. It is their right to have transportation options without bias, free to explore the world and have solutions that make their lives better and easier as every other citizen. Since improving the quality of life of citizens is the main objective of smart cities, we wanted to be a part of the solution providers by creating a mobile application that would offer more information on paths, sidewalks, and bus routes to wheelchair users in Izmit.

2.2. Literature Review on Accessibility Solutions for Wheelchair Users

Turkey has started its journey to make the country accessible to all citizens as was pointed out by President Recep Tayyip Erdoğan on January 9, 2020, as cited in the "Bakanlık 'Erişilebilir Türkiye' için harekete geçti" news article (Göçümlü, 2020). The Ministry of Family, Labor and Social Services in Turkey also pointed out that the ministry will ensure "...necessary criteria are identified and included in the accessibility control process for the elderly to benefit from urban living spaces safely

and without assistance" (Göçümlü, 2020). The President also announced that 2020 was declared as the "Year of Accessibility" (Göçümlü, 2020).

With regards to public transport in cities, most smart cities have solutions that they implement to help wheelchair users move around with ease. Public transport stops in intelligent cities are constructed with accessibility in mind. Adding elevators and wheelchair ramps that give access to the stops are the main improvements taken into account. However, some of these ramps may not be well maintained hence not accessible to all users. Buses are also fitted with wheelchair ramps, which are opened for wheelchair users whenever they need to get on or off the bus. Some smart cities such as London and Paris (Brown, 2016; URL-16) have buses with an automated wheelchair ramp that opens and closes with the press of a button. These buses usually also have prioritized areas for wheelchair users with belts that hold them in place while the bus maneuvers through its route.

Efforts to make public transport accessible in Turkey have been in progress for quite some time. On August 23, 2017, the Ministry of Family, Environment and Social Services, the Ministry of Environment and Education, the Ministry of Industry and Technology, universities, non-governmental boards, and private sector members launched an ‘Accessibility Project of Passenger Transportation Services in Turkey’ (URL-24).

The public transportation and public areas have seen improvements in terms of accessibility thanks to this project. The Ministry of Family, Environment, and Social Services pointed out some of the solutions in their 2019 edition of ‘Erişilebilir Ulaşim Ve İletişim’ that have been implemented to improve the accessibility situation in the country. They explained some of the solutions for the visually impaired which include tactile surfaces in all kinds of bus stops and sidewalks; braille embossed floor plans that direct one to the essential service stations such as the washrooms, information desks, disabled parking lots, and ramps; and special kennels in places such as Antalya and Erzincan airport for the visually impaired aid dogs.

To aid wheelchair users and citizens with mobility problems, they mentioned the ramps and elevators installed at the entrances and exits of buildings, trains, stations,

boat docks, and flyovers, special ticket counters at the train stations, wheelchair-friendly toilets, reserved accessible parking spaces and reserved wheelchair-friendly spaces on the trains and buses. They also pointed out that the staff at stations in cities such as Ankara, Eskişehir, Konya, Adana, Aydın, Mersin, Alsancak, Basmane, Balıkesir, and Manisa have been given sign language training to aid the hearing-impaired passengers. They mentioned that the ministry also launched a project that encourages the employment and inclusion of people with special needs in the staff in areas such as call centers, data entry staff, and even kitchen staff.

In their 2019 edition of 'Erişilebilir Ulaşim Ve İletişim', the ministry described a mobile application that could detect voice commands and direct a user using audio cues to their destination. They explained that the mobile operators provided the visually impaired with a free service where they could be informed on their current location using reference points. Subscription contracts and invoices were also offered in Braille alphabets or as voice messages. They also described call centers with video calls for the hearing impaired, free transport tariffs for the elderly and disabled, and bill payment services where a civil servant went to the homes of the elderly and processed their bills while at home.

In terms of software solutions, researchers, as well as organizations in cities, have implemented a variety of mobile and web applications to try and make it easier for wheelchair users to move around more swiftly. In Berlin, a software solution named wheelmap was developed to provide a map of accessible areas around the city using the traffic light color system. Mobasheri et al., (2017) explained that green marked fully accessible areas, yellow meant partially accessible areas with the probability of a small step, red were sections not accessible, while grey signified areas not yet analyzed. Wheelmap used the power of crowdsourcing, simplicity, and free price to get information about the accessibility of facilities around the city (Mobasheri et al., 2017). It also emphasized on checking wheelchair-accessible toilets in a facility before marking it fully accessible (Mobasheri et al., 2017). Residents of Berlin could all contribute and use the Android, iOS, or web version of the application with ease (Mobasheri et al., 2017).

Researchers also used the power of machine learning algorithms to automate and help in the detection of accessibility issues around their cities. Some researchers combined both machine learning and crowdsourcing to make the process much more efficient. In 2013, Google Street View (GSV) images were used to detect whether an image has a curb ramp or not (Hara et al., 2013). The Support Vector Machines (SVM) with Histogram of Oriented Gradient (HOG) feature vectors classifier was trained using manually cropped 82*40 pixel images of curb ramps at intersections and random non-curb ramps (Hara et al., 2013). Using the sliding window detector technique, images from GSV were classified as having curb ramps or not (Hara et al., 2013). Results showed that the classifier managed to correctly classify 77.3% of the curb ramps but does not misclassify any non-curb ramp image as a curb ramp (Hara et al., 2013).

Hara et al. (2013) then expanded on this research and used GSV, computer vision, machine learning, and crowdsourcing to detect curb cuts on streets. They collected sidewalk data of four north American cities: Washington DC, Baltimore, Los Angeles, and Saskatoon, Saskatchewan. The collected dataset included GSV images and GIS-based intersection data of curb ramps from the four cities. The researchers collected data using a program they developed and called svCrawl as well as through a physical survey of the neighborhoods. The detection of curb ramps was first done automatically using a developed algorithm called svDetect which used SVM as a binary classifier to confirm the existence of a ramp from the results of the Deformable Part Model (DPM). The detected ramps together with the confidence score were then evaluated by a machine learning module called svControl that predicted computer vision performance and decided whether to assign the detection results for verification or for manual re-labeling using a web interface. If labeling was paramount, it was manually done using a developed tool called svLabel where a researcher looked at a 360-degree panoramic image of the sidewalk and drew where the ramp was while the verification relied on the crowds to examine and verify the correctness of the results. They reported a lower cost in using the automated method which they called Tohme, and accuracies of around 85% that were similar to the accuracy of manual labeling of the ramps.

In 2015, Neis developed an algorithm that took factors that affected the reliability and accessibility of a path into account. The algorithm consisted of slope, width, surface,

smoothness, sloped curb, and lighting as the considered parameters which had specified maximum thresholds. Each parameter was assigned an importance score value based on a user's preference. For each segment of a path, Neis (2015) calculated the impedance score using the six parameters. Only the segments with score values greater than or equal to a user's preferences were included in the routing tree. The proposed algorithm proved to generate reliable paths for wheelchair users, however, the length of the path was longer with length differences between 20% and 55%, and sometimes no path was found due to incomplete data in the OSM datasets (Neis, 2015).

Iwasawa et al., (2016) ventured into developing a prototype that would help with the collection of accessibility information without relying on a lot on manpower. Using an iPod touch that was mounted on a user's wheelchair they collected 3-D acceleration signals, location information, and annotation data during a one-hour wheelchair ride in a designated route in Tokyo. SVM, Random Forest (RF), and K-Nearest Neighbors (KNN) were used to classify the different action types into movement on a curb, movement on tactile indicators, climb on a slope, at a stop, and others (Iwasawa et al., 2015, 2016).

Mora et al., (2017) used a different approach to create a system that automatically tried to identify accessibility issues on routes in the city and also provided a way for the public to contribute to the identification of accessibility problems. They implemented a system that automatically tried to identify accessibility issues that wheelchair users would face by comparing the path and the length of the paths used by people without disabilities versus those with disability (e.g. wheelchair users). They described that the system would identify an accessibility issue on a path at the point where the path followed by a person without a disability and the wheelchair user path diverged if the latter was longer. But if the length of the paths were identical, they stated that the frequency of usage was used to identify accessibility issues. Furthermore, they pointed out that if a path always used by a person without a disability was seldom used by a wheelchair user, the path was marked as inaccessible. Apart from that, the system provided an interface that allowed reporting of accessibility issues by users themselves hence involving the disabled people (Mora et al., 2017).

Kozievitch et al., (2017) decided to use base maps that had road and building information in order to derive simplified maps that had sidewalk and curb ramps information to be used in the routing algorithm. The base map integrated a spatial database with OpenStreetMap data. They later simplified the base map sidewalk polygon, then derived a map with the average distance among streets and sidewalks, then added triangles and rectangles that represented crosswalks in road intersection areas to the map, and finally derived a weighted map, which set the weight between two points as the distance of the edge, if there were no barriers, and infinite, if there were barriers. The shortest path from one point to another was selected using the Dijkstra algorithm, with the cost of the edges that had been calculated before (Kozievitch et al., 2017).

With population aging being an important topic of current societies, providing accessible transport for the elderly and wheelchair users is paramount. Rahaman et al., (2017) proposed an algorithm they called contour-based path routing algorithm that retrieved accessible routes for the elderly and wheelchair users. The contour-based graph consisted of a network of roads that were segmented based on the different elevation values which made it easier to evaluate the accessibility of the paths using the total energy consumed and the force needed to climb up the slopes along the path (Rahaman et al., 2017). For their path routing algorithm, they used the multi-objective A* search algorithm which was initially proposed in Ref 25. Case studies were done in hilly cities of San Francisco (USA) for its grid-like street layout, Lisbon (Portugal), and Singapore for their complex street layouts. They reported that the paths output by the contour-based path routing algorithm were consistent and sometimes more than the routes found via Google maps. Thus, the system demonstrated its ability to get more diverse routes with diverse options that a user would choose based on their personal preferences.

Prandi et al., (2017) designed a mobile pervasive accessibility social sensing (i.e. mPASS) application that could be used to get personalized paths based on a personalized user's profile with options such as color blindness or low vision or contribute to data collection. It allowed "...users to (i) configure their profile; (ii) receive notifications to validate the presence/absence of accessibility

barriers/facilities; (iii) spontaneously insert a report; (iv) view the past report logs; (v) display the report localized in OpenStreetMap; (vi) search the route that better fits the user's preferences and needs, computed by a customized version of OpenTripPlanner (http://www.opentripplanner.org/), an open-source multimodal trip planning" (Prandi et al., 2017, pp. 9-10).

Another group of researchers worked on classifying buildings as accessible or not by using convolutional neural networks and image processing to identify ramps at the building entrances (Wu et al., 2019). They took pictures of building entrances from random places to create convolutional neural networks with 2, 3, and 5 hidden layers. They reported that convolutional neural network 5 gave the best results with a model accuracy of 95.6% (Wu et al., 2019). The researchers also acknowledged that they concentrated on just detecting the existence of a ramp in a building entrance and that they did not consider the slope of the ramp or width of the ramp and entrance, which are major factors in the inaccessibility of a building (Wu et al., 2019).

Cáceres et al., (2020) tried to use a playful way to gain the public's help in contributing to data collection. The Android application they developed was called "Access 'n' Go!" where users worked in teams to update accessibility information about the stations of public transport around the city. The app prompted a user to upload information of a station they were closest to at any specific time which gained the team points and badges (Cáceres et al., 2020). The researchers' idea was to collect accessibility information about public transport stations in order to provide users with disabilities enough knowledge to pre-plan and choose routes that were accessible according to their needs.

Edinger et al., (2019) researched on using vibrations from a wheelchair to classify the accessibility of a path. They developed an application called wheelshare that gathered vibration data from geo-tagged accelerometer and gyroscope data of the smart device attached to a wheelchair traveling on different types of surfaces. They then used machine learning algorithms to classify the paths as either accessible or inaccessible. The wheelshare application was further enhanced by (Gani et al., 2019) to compute possible accessible routes when a user sent a request. The computed routes were evaluated using the accessibility classification and other factors such as existence of

curb ramps and crosswalks. The routing algorithm worked by getting the top 10 routes within a radius of the origin and destination points then scores were assigned to each edge of a route using the length of the edge and the detected road surface score that was pre-set. The top three routes were then displayed to the user on a web interface with a map.

2.3. Solutions for Wheelchair Users in Izmit

The Kocaeli metropolitan municipality has provided some solutions to ease the burden of mobility for its mobility-impaired citizens as a way to demonstrate the significance these citizens possess in the city (URL-1, URL-3).

At the time of research, the Kocaeli municipality provided a free taxi service called engelsiz Kocaeli taxi to its mobility-impaired citizens. Citizens were required to complete a registration process to be eligible for the service (URL-3). Once the paperwork and registration process was completed, a wheelchair user could request transportation services by calling and making arrangements ahead of time. The services were available for errands such as going to or coming back from hospitals, schools, and even social and cultural events (URL-1, URL-3). The municipality also made clear that since the service worked on an appointment basis, any emergency or last-minute requests were catered to depending on the availability of the vehicles. They specified that there was a vehicle dedicated to the Gebze district and four other vehicles for the other regions within the municipality. A wheelchair user expressed that they preferred the municipality vehicles more than the regular vehicles because it had a ramp at the back that enabled them to board and disembark the vehicle without much difficulty as well as travel with their friends as it had enough seats (URL-1). As of January 2020, the Disabled and Elderly Services Branch Office in Izmit informed us that they had a total of 442 active registered users of the engelsiz taxi service.

Further discussions with the Disabled and Elderly Services Branch Office in Izmit revealed that they provided an open telephone line for requests and complaints that affected their disabled and elderly citizens. They narrated incidents when citizens would call and ask for the Kocaeli state hospital entrance to be made more accessible through renovating the ramps and removing a trash can that was making it difficult to

maneuver the entrance. More renovation requests had become frequent when the public saw that complaints were being acted upon. They also stated that citizens had reported accessibility issues in places such as universities, bus stops, flyovers, dormitories, and even the city center.

However, the officers revealed that they sometimes clashed with the Traffic Police Department in the city regarding some bollards on sidewalks and ramps. While the Traffic Police Department installed the bollards on the ramps to prevent cars from climbing and parking on the sidewalk, the Disabled and Elderly Services Branch Office perceived it as an accessibility problem for wheelchair users. The case was the same for bollards installed at traffic light crossings. Likewise, some street lights, street signs, and advertisement boards in the city were placed obstructively on sidewalks as shown in Figure 2.1.

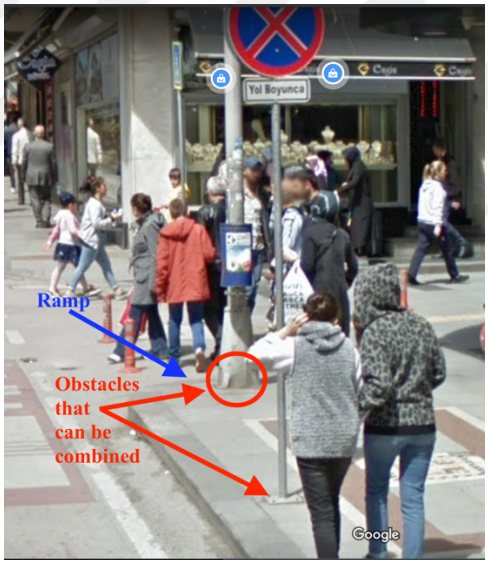


Figure 2.1. An example of an obstructed ramp (URL-23)

The officers in charge of improving accessibility proved that the position of the signage poles could be rectified. They demonstrated that the trash cans, street name signs, and streetlamps could be combined into one pole and placed on the edge of the sidewalk instead of obstructing the ramps (e.g. in Figure 2.1), and the billboards could be attached to the building walls instead of adding more poles on sidewalks. However, some cases were more complicated to resolve like the one in Figure 2.2 where the ramp was completely obstructed from one side of the street. The officers expressed that these

kinds of problems could be avoided by citizens who had prior knowledge of the accessibility situations.

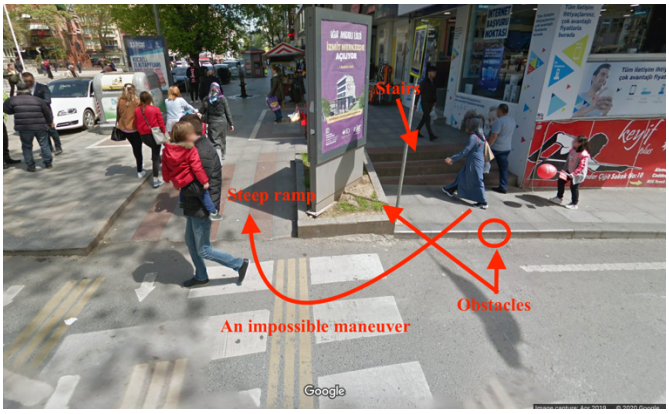


Figure 2.2. An inaccessible ramp from one side in Izmit (URL-23)

Most ramps in the city center were too steep or too damaged for a wheelchair user's safety. The steep ramps were meant to prevent cars from ascending onto the sidewalk, and the damages were due to lack of maintenance. The Disabled and Elderly Services Branch Office declared their efforts in trying to restore the damaged ramps as well as get the steep ramps chamfered for better usability. The officers assured that they regularly monitor any construction work being done in the city that was accessibility related, with their elderly and disabled citizens' best interests at heart. All their efforts were directed to ensuring a more accessible and inclusive city.

Aside from these, the metropolitan municipality of Kocaeli also offered the elderly and those with special needs a unique transportation card that gave them complimentary use of public transport vehicles in the citY. The municipality also announced the installation of wheelchair charging ports on the main walkway in the city center at the celebration of World Disabled day on December 3, 2017 (URL-8). However, these charging stations did not last as reports of their vandalization were reported less than a year after (URL-2).

Efforts to make Izmit more accessible and inclusive were apparent from all the solutions summarized. The Disabled and Elderly Services Branch Office also added that there were more projects underway to improve the district's accessibility situation. Some of the projects described include a mobile application that would replace the

calls for booking an appointment with engelsiz taksi and a partnership with Kocaeli University to increase the public's awareness of the existence of disabled people in life and their ability to be independent when given the opportunity and necessary attention (URL-7). The Information Technologies department of Kocaeli was also working on adding features in the available e-komobil transport application to aid the disabled and elderly citizens in informing approaching drivers of their location so that the drivers assist them in getting on the bus and they were assured of not missing the bus.

We also presented our research idea to The Disabled and Elderly Services Branch Office and they provided their maximum support. Our research proposed a mobile application that computed and recommended routes for wheelchair users. These routes were computed from walking paths retrieved from Google, official bus routes in Izmit, and a sidewalk and ramps map of a designated area of Izmit. The walking paths were retrieved from Google Maps directions API while the bus routes are queried from the official GTFS data of the district. The sidewalk and ramps map of a chosen area of Izmit was manually created using Quantum Geographical Information System (QGIS) mapping software and exported to a database where the querying was done. Furthermore, the application recommended routes by checking the best-rated routes from people with similar age, gender, and wheelchair type.

Our research idea was arrived at after observing the hurdles a wheelchair user faced when using the public buses in the district of Izmit. Easing the uncertainty of: catching a bus, getting space on the bus, and getting the necessary help in a wheelchair user's life was the main drive behind our research. Making them feel independent and included, instill confidence, and encourage them to move out of their comfort spaces was our goal. Moreover, being a part of the smart cities researchers and solutions innovators also played a big role in the significance of our research.

We brainstormed on ideas that would make the wheelchair users' mobility easier or at least more informed. Because Izmit is a hilly place, one of the features that we deemed important was the slope information of paths. Displaying the slope information as it varies on a route would help a user have a mental picture of the way. It would therefore help a user make an informed decision on whether they would be able to go through a path or not hence saving them time and energy. When slope information was

displayed, safety was improved because a user could avoid very steep paths that made a wheelchair harder to control. The slope information on our mobile app was displayed using color-coded polylines. The red areas of the polyline indicated an incline of greater than 7 degrees, blue areas indicated a descent of more than 7 degrees while the green parts showed areas with less than 7 degrees of incline or no more than 7 degrees of descent which meant that a wheelchair could pass that part of the route without much difficulty.

One of the most popular navigation apps is Google maps, which also has slope information. However, one of the downsides of Google maps was that it did not show the slope info of the different parts of a path. It instead used a graph that showed an overview of how the slope changed. This made it hard for wheelchair users to deduce whether specific sections of the path were impassable or not. Even though Google maps is used by millions of users daily, the only wheelchair information available was displayed when a user was in transit mode, which was used to show public transport options that were wheelchair accessible. Our application improved on this by using crowdsourced data to show routes that were preferred by wheelchair users with similar profiles.

The inclusion of bus routes in our research was to add and provide more route options for wheelchair users. By having public transport integrated in our routing algorithms, our application allowed wheelchair users to comfortably search for a route for longer distances than they usually would. A user could easily get route information for two points on opposite sides of the town, and if there were bus routes connecting those points then the app would suggest feasible routes.

We interviewed a wheelchair user in our school, who had to commute several times a week to attend her classes in the school. She also had frequent hospital visits, and she was enthusiastic about getting an informative app. She asked for a way to communicate with the driver of a bus because sometimes bus drivers failed to notice her at a bus stop. We intended to add this feature to our application, but the department of Information Technology at the municipality which was responsible for developing the e-komobil application mentioned that they were already working on this feature for an upcoming release of the application.

She also mentioned that she would get very concerned whenever she had to visit an unfamiliar location, often spending hours looking at maps and photos of the destination and route in order to map any areas that she had to avoid. Sometimes she encountered obstacles along the way, such as steps on the sidewalk that were too high to maneuver forcing her to turn back and look for more accessible routes, which led longer travel times than planned.

One of the most important features she was interested in was the mapping of sidewalks, along with the locations of ramps and their actual accessibility status. Her requests guided our search for sidewalk data, since we could not get reliable data from OpenStreetMap (OSM) so we went to the municipality with our idea to learn if they could help. The transport office redirected us to The Disabled and Elderly Services Branch Office and the Information Technology office of the district. Even though the municipality was very helpful and provided us with answers to all our questions, we could not find a database of the sidewalk, ramps, and elevators data of the district.

We decided to create a map of a small designated area in the city center that would become our pilot project. The chosen area consisted of junctions, a mall, a park, and a major bus stop. Armed with her inclinometer and tape measure devices, the internal architect at The Disabled and Elderly Services Branch Office took us around the town and described the obstacles, steep ramps, damaged ramps, manholes and drainage features that would hinder smooth movement of wheelchairs within the area. The architect also explained a few solutions that could be implemented to rectify these issues with time, such as combining street light and road sign poles to reduce obstacles on the sidewalk and chamfering ramps to make them gentler and more accessible.

Using the data collected in the field, we created sidewalk map data for the pilot area using QGIS software. We added the slope, width, length, surface type, and accessibility state information to the map. This map was then exported to a PostGIS database where the routing was implemented. We used the Dijkstra algorithm that is supplied in pgRouting to get the route from one place to another within the sidewalk map. After this, we updated our logic to first check for sidewalk routes before checking for the google map walk path, to ensure that users always get the most optimal route available.

Another feature added on our application was the rating of routes. Users could rate a route as many times as they wanted, and when subsequent searches were made, the application suggested highly-rated routes first. We added a smart recommendation for queried routes, whereby users who registered in the system were clustered based on their age, gender, and wheelchair type using k-means clustering machine learning algorithm. The system would check if a route with the same origin and destination had ever been rated by the users in the cluster, and if available it calculated the average rating of the routes and returned the ones with the at least a score of three as the recommended routes. On top of that, the system also searched and appended the Google walk paths, the bus routes, and the sidewalk paths to the results to give the users the power of choice. Finally, the clusters were set to auto-update after every five new users joined. The optimal cluster number was selected using the silhouette coefficient method.

3. TECHNICAL DETAILS

The intended output of our research was to create a smart city mobile application specifically for routing wheelchair users of Izmit. Most smartphone users had either Android or iOS devices which required the use of a framework that could ease creation of a cross-platform mobile application. Most of the utilized technologies in our research were selected based on different criteria. We explain a few of the details of the main technologies used to create our wheelchair routing application.

3.1. Flutter Framework

Flutter is Google's User Interface (UI) toolkit for building natively compiled applications for mobile (iOS, Android), web, and desktop from a single codebase. It is a free open source project meaning it is open to contributions from anyone who is interested. Flutter allows fast application development, and the resulting application can be published to both the Google Play Store and the Apple App Store without changing the code or having to write separate applications. It also helps implement an intuitive user interface due to its layered architecture that gives developers control over every pixel on the screen, as explained in Flutter's official documentation page .

Development in flutter is made quick, easy and intuitive through its hot reload feature. It is able to update the UI on the device or an emulator on-the-fly after saving the changes made in the code, without having to restart the app. It also works seamlessly with popular integrated development environments such as Visual Studio Code, IntelliJ, and Android Studio. The framework provides a large number of packages and plugins to choose from for any development needs. The packages and plugins are developed by people in the flutter community which include integration tools and simplified utilities for all needs of any app. Flutter also integrates well with the chosen backends (Google Cloud Platform for authentication, and Node.js as the API endpoint).

It uses the object oriented Dart programming language. Dart can be used to write software for all platforms such as mobile, web, desktop, command-line, and server side, whether a script or a full-featured app. The language has a C style syntax, it is class based, and garbage collected. The flutter framework is powered by Dart native which has a Dart VM (Virtual Machine) and an ahead-of-time compiler for producing machine code. The Dart VM is a virtual machine that provides an execution environment for the dart language.

Development of mobile apps can also be done using Java or Kotlin languages for Android and Objective-C or Swift for iOS devices. These are native languages designed to develop apps for the specific platforms. Development using native languages entails different codebases for different devices. For example, an android app written in Java would need the same code to be written in iOS specific languages to get its iOS equivalent app which doubles the development work needed.

Other cross-platform mobile app development tools include React Native, Ionic, and PhoneGap. React is a javascript library for developing UI created by Facebook. It has most of the pros as flutter like the hot reload feature, many packages and plugins, free and open source, and great documentation. However, it has issues with memory leakage, slow app launch, and large app size in Android (Hossain, 2020). Ionic also uses javascript with HTML and CSS when creating mobile apps. The mobile apps created using Ionic run in a web view which is basically a browser with a hidden browser header (Netkow, 2019). PhoneGap was developed by Adobe and was similar to Ionic. Their performance is moderate compared to flutter apps, and was more suited in creating simple applications.

We used the flutter framework and dart programming language due to the convenience of being able to compile the app for both Android and iOS devices from one codebase. Flutter also provided many UI packages and plugins such as Google Maps, slide-up panel, star rating, and excellent animations needed to enhance the experience of using the app. The flutter framework was also an emerging trend in the development world at the time of development, which was an excellent opportunity to learn. It integrated seamlessly with other Google libraries and APIs, such as firebase.

3.2. Firebase

Firebase is a mobile and web application development platform owned by Google. It offers user management, authentication, real time databases, push notifications, and performance and crash analytics, which earns it the position of a mobile backend as a service (MBaaS). MBaaS are service providers that offer backend service needs of a mobile application such as data management, data storage, analytics reports, push notifications, and API endpoints without writing any code. Therefore, developers can create mobile and web apps without developing the backend for apps and instead use the MBaaS services according to the needs of their applications.

Other BaaS solutions that exist are Apple's Cloudkit and Kinvey. Cloudkit flaunts its seamless integration with iOS, Mac, and web apps (Ashwini, 2020). It provides data storage functionality, authentication using iCloud, and push notifications services (Ashwini, 2020). It, however, only supports iOS apps (Ashwini, 2020). On the other hand, Kinvey and Firebase support iOS, Android, and web apps. Kinvey has a database, push notifications, authentication, and location services (Ashwini, 2020).

Firebase also provides anonymous user authentication by assigning each user a unique id whenever needed. Firebase offers a tree-view real time database and a nested documents and field database named cloud firestore. These databases can be used to synchronize data across clients or any features that occur in real time such as chats. Firebase has analytics and crashlytics that enable developers to monitor the app usage and view errors that occurred when their apps were in use.

We used the anonymous authentication feature in our application, which created anonymous accounts using unique ID assigned automatically by Firebase. Each ID was unique for each installation of the app. We also used the cloud firestore database to synchronize the user's profile information with our own database in PostgreSQL. The synchronization between firebase and PostgreSQL was made possible via cloud functions.

Cloud functions are javascript codes deployed to a firebase project. The functions can either be triggered via HTTPS requests or events such as database document creation, update, and deletion. We wrote cloud functions that would be triggered by the cloud

firestore database whenever a user was created, deleted, or updated, the function triggered and invoked the corresponding PostgreSQL query that synchronized the data. We used Firebase because of our previous knowledge of the system, its real time database functionality, and the anonymous login feature that enabled our app to assign users a unique ID without having to demand them to sign up.

3.3. Google APIs

Google APIs are a set of endpoints which enable communication between apps and Google Services. Google services are endpoints that could be used to extend the functionality of applications. There is a vast number of Google Services available. However, we used only a few services under the Google Maps Platform.

To use Google APIs, it was essential to obtain an API key from the developers console which was used as credentials when accessing any Google service. The developers console was also used to enable the APIs as needed. In our application, we used the Maps, Places, Elevation, and Directions APIs. Each API had excellent documentation that explained the form of the request for each API, its required and optional parameters, and what each parameter meant. The documentation also provided extensive examples that showed the expected response, its format, and the meaning of returned fields.

There were other services that provided mapping APIs with directions, place search, and elevation endpoints like openrouteservice. All requests made through these services needed credentials. The openrouteservice APIs had a free tier program with a limited number of requests per API per day. For example, direction accepted 2,000 requests per day, elevation points and linestrings were limited to 200 requests per day, and geocoding accepted only 1,000 requests per day. There were many other open source routing software with different capabilities and tiers. A list of them can be found under (URL-15). Each option had its own capabilities, syntax and requirements. We used Google Services since it was a familiar ground and it included the APIs needed for our application.

The Google Places API basically returns details of an area. It has different request types. We used the places autocomplete request to get suggestions of place names that

a user was searching. Each query in the autocomplete request passes a search string which responds with five places, having its name and place ID, that matched the search string. We also used the place details request to get the details of a place using the place ID. The place ID sent to the request was retrieved from the places autocomplete request.

The Google Directions API is a service that returned directions between two locations depending on the mode specified. The different modes include driving, walking, or cycling. The locations sent to the API request can either be in the form of text, latitude and longitude coordinates, or place IDs. The response returned includes a routes array with a list of the legs in each route. The routes list also contains an overview polyline which is an encoded polyline of the locations in the route. Using the polyline, it can be decoded to get the latitudes and longitudes in the path of the route which can then be drawn on the Google map.

The Google elevations API request returns the elevation data of the locations in the query. The locations can either be passed as one point location or as a path. The response from the API includes the latitude and longitude of the point and its elevation in meters. Each of the Google APIs require a key parameter within the request for authentication.

3.4. PostgreSQL Database

PostgreSQL is an object-oriented database that is open source and stable. It has the spatial database PostGIS extension which adds support for geographic objects in the database. The extension also makes it possible to have geography related SQL queries that are easy to write. It adds extra spatial data types to the database such as geometry, geography, and raster data types that refer to shapes such as point, line, and polygon. Functions, operators, and indexes are also included to enhance the usability of the spatial data types.

Other spatial databases that compete with PostGIS are MySQL and Oracle Spatial. A study done to compare the performance between Oracle Spatial and PostGIS database showed that PostGIS was faster in querying spatial data than Oracle Spatial (Shukla et al., 2016). While performance measurement was not entirely dependent on the time

constraints, time taken to execute a database query was one of the most important factors that we considered. Shukla et al. (2016) also reported that query execution was costlier in PostGIS than in Oracle Spatial in terms of computer resources used.

PostgreSQL database also has many other extensions that can be enabled as needed. PgRouting extension was the extension we used to compute the shortest distance between two locations. With the fast performance, spatial database extension, pgRouting extension, QGIS integration, and understandable documentation, PostgreSQL PostGIS database was deemed the best choice for our application needs.

3.5. General Transit Feed Specification (GTFS)

Google developed the GTFS as a format for sharing public transport information. Using a standard format, public transit agencies can publish their transit data and developers can consume the data in any of the diverse variety of applications that use public transit information without much hassle. As explained on the official website, GTFS is split into static and real time components. The static components comprise the necessary information on stops, routes, and schedules, while the real time version includes arrival predictions, vehicle positions, and service advisories (URL-5). Thousands of cities have published their GTFS data openly to the public. OpenMobilityData is a project that aimed to collect open transit data from around the world. It showed that it had data from 1251 providers in 673 locations worldwide as of July 11, 2020. From the hosted feeds, only two cities of Turkey published their GTFS data: Izmit in the municipality of Kocaeli and Mersin in Mersin's municipality. We acquired the bus information for our application from the Izmit GTFS feeds hosted on OpenMobilityData.

GTFS is intended to be a standard format. It contains specifications on the naming and requirements of the files, the number of fields in each record and their names, and the format of each field type. The field types have to follow specific guidelines that are explicitly documented in the official GTFS website. For example, colors are written in the hexadecimal format without the leading # symbol, dates follow the four-digit year, two-digit month, then two-digit day format and the latitude and longitudes use the WGS84 decimal degrees that are expected to be between -90 and 90 degrees inclusive.

The included files in a GTFS dataset are also conventional. The current dataset comprises five required, three conditionally required, and nine optional files. All files in the dataset are text comma-separated files. The most significant files in the collection include the stops, routes, trips, and stop times files. The stops file carries information about public transit bus stops in the area, the stop codes, names, and locations represented as latitude and longitude. It also includes a field named wheelchair_boarding, which indicates whether a stop is wheelchair accessible or not. The stops file retrieved from the Izmit dataset had the value 0 for wheelchair_borading, which meant that no accessibility information was available about any of the stops.

In the context of the GTFS dataset, a trip is defined as "...a sequence of two or more stops that occur during a specific time period" (URL-13) while a route is "a group of trips that are displayed to riders as a single service" (URL-13). The routes file contains the names of all routes with their short or long names. The routes in the city of Izmit are known mainly through unique bus numbers together with the first and last stop names. Hence the route_short_name field in the file was assigned the bus number while the route_long_name field was assigned the origin and destination names. The trips file consisted of multiple records of trips for each route. Each record of a trip had the name of the destination as the trip_headsign field, while the trip_short_name field was assigned the start time of a trip in the Izmit dataset. The trips file also had wheelchair_accessible, which denoted whether a trip was suitable for wheelchair users. However, this field in the Izmit file were all marked as 0, which meant the accessibility status was inconclusive.

And lastly, the stop times file describes the expected arrival and departure time of buses at each stop of a trip. This file primarily draws a picture of each recorded trip. The stop sequence is used to describe which stops a bus travelled through on a trip. The time between stops is available using the arrival and departure time fields. And finally, the stops where a bus can only pick up but not drop off and vice versa are defined using the pickup_type and drop_off_type fields, respectively. With the required five files, it is possible to draw a map of all the routes in a city, among many other potential use cases, as long as the data provided is accurate. More information

on the contents of all files as well as their meaning can be found in Google's Static Transit reference documentation (URL-13).

3.6. Quantum Geographical Information System Desktop

QGIS is an open source, free desktop GIS application. It supports viewing, editing, and analyzing geospatial data. GIS applications enables one to open a digital map, create or edit spatial information, customize maps, print maps, and perform spatial analysis. There are many GIS applications that are both available freely and on payment. ArcGIS is one of the closed source GIS software that runs on Windows only. It is mostly similar to QGIS functionality except that it does not integrate with open source GIS plugins. However, it has better spatial topological and analytical capabilities than QGIS. QGIS also works hand in hand with other GIS applications like GRASS GIS. According to the GRASS wiki page, the app focuses more on analysis while QGIS focuses on map making which makes them complement each other.

We mainly used QGIS to draw a map of sidewalks in a designated area of the city. With its extensive documentation filled with examples, QGIS was fairly easy to learn. As a first step to drawing the sidewalk map, we added a base map layer. The base map is a reference map such as Google Maps, OSM, and Bing Maps. QGIS had a QuickMapServices plugin that made adding any base map possible. We decided to use Google Map as our base map since our mobile application used it too.

QGIS also has a QuickOSM plugin that can be used to query and visualize OSM data. The plugin has a graphical user interface that can be used to generate an OSM query. The OSM queries use key value pairs as described in the OSM wiki page. QuickOSM also requires the query to specify an area to search. It provides a few options such as a name of a place, the canvas extent, or the layer extent. The queries are run through Overpass API which is a read-only API that returns OSM data based on the query sent.

To draw a new map, an empty layer was first created. Using the properties settings of the layer, the fields of the drawn elements could be defined. The fields' names, object type, and length were specified just as is done in a database. Using the relevant add feature option, a new point, line, or any other shape could be used to add items on the

relevant layer. After each feature addition, the necessary fields were entered. These layers were also exportable in many different formats such as GeoJSON, SQLite, Shapefile, and even as an excel spreadsheet.

QGIS incorporates a database manager plugin. It is used to integrate with spatial databases that are supported by QGIS like PostGIS and Oracle Spatial databases. The database menu in the plugin allows connection to existing spatial databases. The tables and schemas can then be viewed and edited directly from QGIS. The plugin also has an SQL window which allows writing and execution of SQL queries on the connected database instances. The result of the queries can also be visualized by exporting it as a layer to the project. Using the database manager plugin and query execution capabilities, it is possible to run routing algorithms and visualize the returned routes directly on the map. This made it easier to debug and confirm that the drawn map was correct.

3.7. PgRouting

The PostGIS PostgreSQL database has a geospatial routing extension called pgRouting. It is easily enabled on any PostGIS enabled PostgreSQL database instance. PgRouting extension is bundled with many famously known routing algorithms. It was first called pgDijkstra which had the Dijkstra algorithm methods. It was then extended and renamed to pgRouting and has since grown to include a variety of routing functionality. The extension also has documentation with examples which helped when learning to use it.

We used pgRouting to find the shortest path from one point to one destination. Some of the available algorithms used for this purpose include A* and Dijkstra algorithms. Other functions with other purposes are also available. For example, the travelling salesperson problem computes the shortest route that passes through each node only once and returns to the start node. There is also Kruskal's and Prim's algorithms that finds the graph with minimum weight edges that spans all nodes in the original graph.

The Dijkstra algorithm is a greedy algorithm that finds the shortest path in a graph from a defined start and end node using positively weighted edges. The A* algorithm also has the same principle of Dijkstra with an addition of a heuristic function. The

heuristic is an intelligent estimation of the cost between a node and the intended destination. PgRouting defines five different heuristic functions that can be used in its implementation of A* algorithm. These include the Manhattan, diagonal, and Euclidean distances. It also has an option to set the heuristic to a value of 0 which would basically make its logic similar to the Dijkstra algorithm.

Rachmawati and Gustin (2020) experimented the differences in execution time and loop count between A* and Dijkstra algorithm in a path finding problem. They created a map of their university and used the distance between intersections as the cost of the edges in the graph. Their reports showed that A* executed faster than Dijkstra in both execution time and loop count variables. However, the returned path and shortest distance were the same. They made the conclusion that either of the algorithms could be used on a town or regional scale. However, the A* algorithm was a better option for large scale maps. They also stated that Dijkstra's slow execution was caused by its exploration of options in a much larger area that expanded out equally in every direction contrary to how A* works.

The use of an algorithm that went through the maximum number of options to find the shortest path was considered a major advantage for our problem. This was because of the small area in question. When executing the pgRouting functions, a cost value is expected to be defined in the first parameter of the functions. The A* and Dijkstra functions returns a list of nodes that correspond to the shortest path in the graph passed to the function. The details of the nodes returned include the sequence of the node in the path, the identifier to the node's location and name details in the original graph, identifier of the edge between the current node and the next, the cost of the edge, and the aggregated cost that equaled the total cost from the start node to the node in the row. This result set can then be used to display the found path and the last row's aggregated column would equal the total cost of the path.

3.8. K-Means Clustering and The Scikit-Learn Library

K-Means is an unsupervised iterative clustering algorithm. Its main aim is to cluster samples based on their features into a predefined number of clusters. It uses the Euclidean distance metric to get the distances between samples and their assigned

cluster centers (Yuan and Yang, 2019). K-Means works to minimize the distance within cluster samples and maximize the distance between clusters. The algorithm workflow is iterative but it starts by randomly selecting k cluster centers from the dataset (Arthur and Vassilvitskii, 2007; Li and Wu, 2012). The downside is that the correctness of the clusters is sometimes influenced by the first assigned cluster centers (Li and Wu, 2012). Such that varying initial points may sometimes result in different clusters.

Each sample is then assigned to the nearest centroid which is the centroid with the least Euclidean distance value between the sample and the centroid. The next step in the algorithm is the centroid update. The centroid is recalculated using its distance to each sample in the cluster to reposition it in the center of its cluster. The algorithm then iterates over the cluster assignment and centroid update operations until either a maximum number of iterations is reached or the centroids do not change location.

K-Means algorithm expects a predefined number of clusters as a parameter. Defining the optimal number of clusters may not be straightforward in some cases. Hence there exists some algorithms that could be used to estimate the optimal number of clusters. Some of these algorithms include the elbow method and the silhouette coefficient algorithm. The elbow method is one of the most commonly used techniques to estimate the optimal k value (Trevino, n.d.). In this method, the Sum of Squared Errors (SSE) is calculated for values of k ranging from one to a defined maximum number. Smaller SSE values show that the samples in the cluster are closer together (Yuan and Yang, 2019). A plot of SSE against k is usually used to visualize the decline in the SSE values as k increases. The curve rapidly declines as it approaches the optimal k value and then becomes steady or slower as it exceeds the optimal k (Yuan and Yang, 2019). This results in a curve that resembles an arm with an elbow at the optimum k value.

The silhouette coefficient method provides a rough measurement of how good the resulting clusters are. It gives an overview of how far apart the clusters are from each other. The coefficient values range between -1 and 1. A value of -1 implies that a sample has been assigned to the wrong cluster, while values near 1 depicts correct clustering because the sample is near its cluster centroid (Yuan and Yang, 2019). Values near 0 indicate almost or overlapping clusters (URL-17). Yuan and Yang

(2019) provided a concise and understandable description of the calculation process of the silhouette coefficient of a cluster. They explained that it is calculated by first selecting a sample i in the cluster, say a. The average distance of i to all other samples in the cluster a is then calculated which they called a(i) the intra-cluster dissimilarity of sample i. Then they stated that the average distance of i to all samples in the other clusters, b(i) is calculated and the least of these distances, i.e. b(i) = min{bi1, bi2, ..., bik}; yields the inter-cluster dissimilarity value, b(i). The silhouette coefficient of the sample i is finally computed using Formula (3.1) as provided in (Yuan and Yang, 2019).

$$s(i)=\frac{b(i)-a(i)}{\max\{a(i),b(i)} = \begin{cases} 1-\frac{a(i)}{b(i)}, & a(i)<b(i) \\ 0, & a(i)=b(i) \\ \frac{b(i)}{a(i)}-1, & a(i)>b(i) \end{cases} \quad (3.1)$$

The cluster silhouette coefficient would then be calculated by getting the average of coefficients of all samples in the cluster. Silhouette coefficient values that are nearer to one indicates that the samples are assigned to the correct cluster.

A study was conducted to analyze four algorithms used to define the optimal value of k. The researchers used the iris dataset to compute its optimal number of clusters and analyzed the efficiency of each of four methods: elbow, gap statistic, silhouette coefficient, and canopy methods (Yuan and Yang, 2019). Their results showed that all four methods yield the same k value however the execution times differed with gap statistic and silhouette coefficient taking over eight seconds for 100 samples. The elbow and canopy methods were a bit faster. The researchers concluded that silhouette coefficient and gap statistic methods were not suitable for large scale datasets and that canopy was the best algorithm for large and complex datasets (Yuan and Yang, 2019). However, they also reported that all methods were suitable for small datasets (Yuan and Yang, 2019).

The implementation of machine learning algorithms has gained a lot of traction in the past few years. This has brought up many machine learning libraries and systems that

help students to learn as well as simplify the development of machine learning applications. These include the scikit-learn library and the WEKA software. The scikit-learn library relies fully on the Python ecosystem (Pedregosa et al., 2011) while the WEKA software has a wrapper called python-weka-wrapper that allows it to be used in Python projects.

The scikit-learn library can be used to implement a wide variety of both the supervised and unsupervised machine learning algorithms. Its documentation explains all the parameters of each algorithm and provides examples. Furthermore, they have an extensive user guide that explains each algorithm and function with examples, charts, figures, and code snippets. The WEKA software is an open source graphical user interface desktop application with a number of machine learning algorithms developed at the University of Waikato in New Zealand. It can be used to run many data mining algorithms on one dataset at a go enabling easier and faster comparisons of different algorithms and their outcomes. Since our implementation of the backend was a Node.JS project, implementing a python scikit-learn project to handle the clustering was considered a better option. The availability of examples, documentation, and a wide user community added to its appeal.

The K-Means function in the scikit-learn library expects a certain number of parameters. These have been explained in their documentation (URL-17). Each parameter has a default option to reduce the number of errors. We will look into some of the parameters expected. The number of clusters n to generate is required, and if not passed, it defaults to eight clusters. The centroid initialization uses different methods. The random option chooses n random samples as the initial centroids. The k-means++ option initializes centroids generally far from each other. The user guide in scikit-learn explains that the k-means++ algorithm was proposed by (Arthur and Vassilvitskii, 2007) as a smart selection of initial centroids to speed up convergence and curb the issue of getting different clusters with different initial centroids (Arthur and Vassilvitskii, 2007).

In kmeans++, the first centroid is first selected randomly from the data points. Then the distance of each data point to the selected centroid is calculated. The next centroid that is chosen is such that the chosen data point is far from the previous centroid with

high probability. This results in initial centroids that are widely spaced from each other. And since the chosen centroids are data points, there is a high likelihood that there are other points related to it. The scikit-learn K-Means algorithm also uses Elkan's algorithm when calculating distances. The Elkan's algorithm uses triangle inequality to reduce the number of distance computations hence making the overall K-Means algorithm faster (Elkan, 2003). Other parameters include the maximum number of iterations for computing centers with a default of 300, the number of times the clustering should be done with different centroids whose default is ten, a flag that signifies whether distances should be precomputed, and the number of threads to use during the executions.

The results from the K-Means scikit-learn algorithm includes an array of the cluster center locations, an array of the cluster labels, the number of iterations run, and the inertia which is the "sum of squared distances of samples to their closest cluster center" (URL-17). The K-Means model can then be created using the fit method which computes the cluster of the samples passed in the first parameter. The model can later be used to get the cluster index of each sample in the dataset using the predict function. To easily get the cluster a specific sample belongs to, the computed K-Means clusters can be cached using Python's joblib tool. It persists a Python object to a file and can be used to re-read the model and directly use it to perform cluster predictions.

The use of K-Means clustering algorithm was to enable the generation of intelligent recommendations for all users even on a first request. Other recommendation algorithms such as collaborative filtering require prior knowledge of a user's rating to generate recommendations. While K-Means clustering is the most widely used clustering algorithm (Elkan, 2003), other clustering algorithms can also be studied and compared in future research.

3.9. Node.JS

The server-side API calls for the mobile application we developed were handled by Node.JS. It is defined an asynchronous and event-driven JavaScript runtime environment. Written in C/C++ and JavaScript, Node.JS is designed to build scalable network applications in which many connections and requests can be handled

concurrently. This is very different from more conventional models, in which each request is handled in its thread. Node.JS users do not have to worry about deadlocking of processes, files, and other resources because of its asynchronous nature. Web protocols such as HTTP/S are built into Node.JS, making it well suited for use in a web library or framework.

For the backend API server for our mobile application, we considered several options before settling on Node.JS. Among the considered options were Python, Ruby on Rails, Django and Laravel. Even though Python is an excellent choice for a backend API server, Node.JS has better memory optimization (URL-9), making it a better choice for our application, which ran on an online virtual server where minimal memory utilization was key. Our application made extensive use of asynchronous calls to database functions and to web APIs, which was not an area where Python is strong at. According to URL-9, Node.JS is better at asynchronous process management than Python. Ruby on Rails is one of the most popular frameworks for backend and web programming. However, Ruby is much more opinionated (expects files and folders to be structured in a certain way) than Node.JS, which makes it suitable for larger projects that have many files and processes. For the purpose of our study, Node.JS was more appropriate since we did not envision a large API server, but a compact one with a minimal set of functions. Django (Python Based) and Laravel (PHP) were also considered, but based on our previous experience of working with Node.JS we opted not to use them because we felt we would be more productive in a familiar environment.

In order to handle Representational State Transfer (REST) API requests, we opted to use restify, a Node.JS web service framework optimized for building semantically correct RESTful web services ready for production use at scale. Restify optimizes for introspection and performance. There are several alternatives to restify, the most popular being flask and expressjs. We considered using each of these, but chose restify because it can easily handle many concurrent requests without throttling them.

4. METHOD

Our research was mainly focused on providing route options for wheelchair users. However, we also believe that an informed decision is more important than trial and error attempts. In this regard, our first step was to provide routes with information critical to wheelchair users such as the slope, distance, and all available pathways. Moreover, we added in the public transport options that would further help wheelchair users traverse hilly areas more efficiently. Finally, we attempted to get more intelligent route recommendations using the power of crowdsourcing.

Our perception of the research was to provide wheelchair users a virtual collaboration platform. Whereby a user would rate a route and their rating would be used to provide recommendations to others. To achieve this, we used K-Means algorithm which clustered users based on profile similarity and recommended routes based on the feedback from fellow cluster members. A brief overview of the application workflow is depicted using flowcharts.

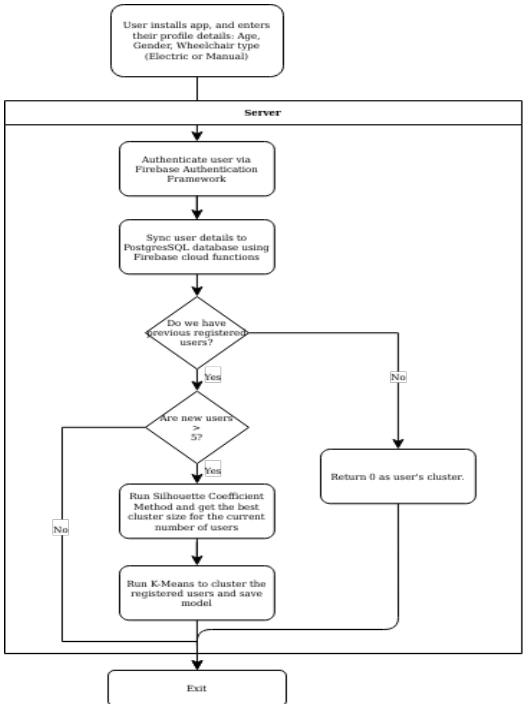


Figure 4.1. User registration flowchart

After every 5 new user registrations on the app, the clustering functionality was triggered, as shown in the flowchart in Figure 4.1. All previously-recorded users were used in the silhouette coefficient method to get the most appropriate number of clusters and avoid overfitting the data. The optimal number of clusters was used to fit the data in the K-means clustering algorithm and find the cluster a new user belonged to. The cluster number returned also provided all the users in a certain cluster. The cluster of the user that is output in Figure 4.1 was used in the first step in the API server section of Figure 4.2.

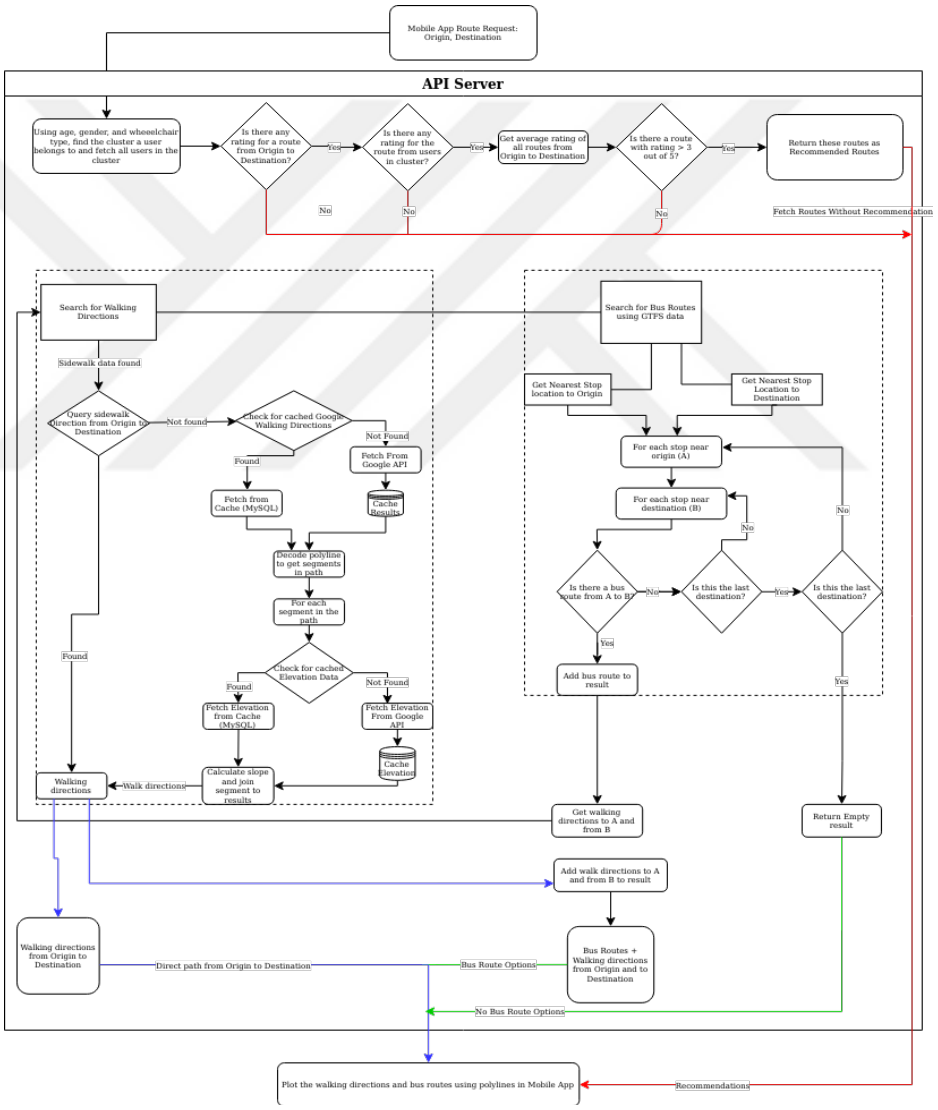


Figure 4.2. Route request process flowchart

Figure 4.2 displays the logical flow of the routing algorithm we used in our research. Whenever a user posted a request for a route from an origin to a destination, the server

first checked if there were any rated routes for this point. The routes that had been rated were then filtered based on users in the same cluster as the user requesting the route. If any users in the cluster had rated any of the routes, the average rating of each route was calculated and routes with an average rating of above three out of five were added to the result set as the recommended route.

The system then continued to search for the walking directions and bus routes for the requested origin and destination as it would when there was no rated route. Since the app was developed with wheelchair users in mind, we queried a direct path from the sidewalk map using the Dijkstra algorithm. The algorithm used the length of a segment calculated from the geopoint of its start and end location as the cost of the segment. If a sidewalk path was found, we skipped querying the walking path from Google, but if no path was found, we used Google directions API to get a direct walking path in segments. The elevation of each segment's start and end location was then retrieved via Google elevation API, and the overall slope of the segment was calculated. These segments were added to the result set as walking directions and displayed on the app using polylines. Each segment of the polyline was given a color based on the slope value of the path's segment.

Bus routes were simultaneously queried from the PostgreSQL GTFS database of the district of Izmit. Querying bus routes required the use of bus stops as parameters. The nearest bus stop within one kilometer to both the origin and destination locations were fetched using PostgreSQL's ST_Distance_Sphere function. Each pair of the first and last stops were queried for a direct bus route. If a bus route was found, walking directions from the start position to the first stop and walking directions from the last stop to the destination was fetched using the previous steps outlined. The route was then added to the bus route options result set.

After the process described above was complete, the result set finally contained a direct path, bus route options, and smart recommendation options. App users could also rate each segment of each route separately from the routes list. These ratings were re-used and re-calculated with every request made.

4.1. Mobile Application Development using Flutter Framework

The mobile application in our research was designed using Flutter, Google's UI toolkit for building natively compiled applications for mobile (iOS, Android), web, and desktop from a single codebase. Flutter was chosen because it allows for fast application development, and the resulting application could be published to both the Google Play Store and the Apple App Store without changing the code or having to write separate applications. Flutter also integrates well with the chosen backends (Google Cloud Platform for authentication, and Node.js as the API endpoint). Another reason for choosing Flutter over native app development was the speed and ease of producing a fully working application with an intuitive user interface.

The source code for the application is publicly hosted in an online repository on GitHub and is freely available for any future researchers who may wish to reproduce or build on this research. This also ensures that the API can be easily re-deployed to a new server in case redundancy is needed or in case there are problems with the current server. Amongst the considered options for developing the mobile application (including native applications), flutter was chosen because of its fast development methodology. When using flutter, changes are seen on the test device as soon as the application source code is saved. This greatly reduced development time since the developer does not have to wait for the application to recompile every time code is changed.

4.2. Node.JS Backend Route Recommendation Processing

The API developed in Node.JS was hosted on an online Linux server running Ubuntu Linux. All communication between the application and the API server was transferred over HTTPS, thus making it secure and private. To further increase security, all requests to the API server were routed via Cloudflare's DNS service. This ensured that the API server delivered fast results while remaining protected from DNS attacks such as Denial of Service (DoS) attacks.

In order to reduce the number of calls being made to Google APIs, caching was implemented in the Node.JS API server source code. Whenever a call was made to a Google API, for example, the elevation between two points, this data was cached in

the PostgreSQL database. When a new request was made for the exact same coordinates, instead of calling the Google API again, the net result was fetched from PostgreSQL. This greatly reduced the number of calls to Google APIs and also improved the speed at which the responses were served back to the application. Overall, caching Google API requests reduced the cost incurred from the API requests. The sidewalk map data was stored in a PostgreSQL database, in order to fully utilize algorithms such as pgRouting that were better suited to be run on PostgreSQL. Using Node.JS as a centralized API host ensured that the application always called one specific route for all API calls, and the API server then routed these calls to the different providers for each request

4.2.1. Google Firebase

Since Flutter is a Google tool, it integrates with Firebase seamlessly, ensuring proper differentiation of different users and leading to improved and targeted route suggestions. Firebase also includes built-in metrics and performance analytics, which can be enabled to track the app's usage, performance, and crashes (if any) to ensure the best user experience possible.

In the mobile application that we developed, we use firebase for authentication and user information tracking. Even though users did not have to explicitly enter credentials to log in, the application conducted an anonymous login via firebase that tracked each installation with a unique id. This id was used to represent the user and was used across the application when storing preferences, rating routes or getting recommendations for various routes. Firebase's flutter library is robust enough to do this authentication seamlessly in the background, and if conventional username/password authentication will be needed in the future then it can be added.

In order to sync user details between Google Firebase and PostgreSQL, Firebase Cloud Functions were used. Cloud Functions for Firebase is a serverless framework that lets developers automatically run backend code in response to events triggered by Firebase features and HTTPS requests. JavaScript or TypeScript code is stored in Google's cloud and runs in a managed environment, eliminating the need to manage and scale your own servers. In our PostgreSQL database, we had a table that holds users

(izmit.users), which was populated by the Firebase Cloud Functions. The user functions, createUser, updateUser, and deleteUser were deployed to keep the PostgreSQL table in sync with the users table in Firebase.

4.2.2. Bus route querying

The data for mapping bus routes and displaying bus schedules came in the form of a GTFS data set. GTFS is a data specification that outlines guidelines regarding how public transit agencies can organize and publish transit data in a standardized format that can be used by various software applications (URL-20).

A GTFS dataset for Izmit was obtained from an officially published feed by the municipality of Kocaeli at URL-10. This data set contained route information, bus stop details including their coordinates and bus schedules for each route. In order to make it easier to consume this data in the API, we converted it from the original collection of text files to database tables. We tried to use Firebase's real time database first by converting the text files to NoSQL and uploading them to Firebase. However, the result was over 3 million requests which ended up costing us 250 Turkish liras. Since the GTFS data had exceeded the free tier provided by Firebase, we could not continue using it as our database due to the high cost that would have been incurred by subsequent queries to the database. We then dropped the GTFS data from Firebase and shifted to migrating the GTFS data to a relational PostgreSQL database.

Our research was not focused on getting ways of importing the bus routes data into a relational database. We, therefore, used an open-source tool developed in Python to import the data to our database. The tool is called gtfsdb and can be found in GitHub under the OpenTransitTools account (Young, 2019). The developers of the tool mentioned that the software was developed to help jump-start software developers in their projects that need GTFS data in a relational database. They provide a short tutorial on how to use the tool to migrate the official GTFS data to PostgreSQL, Oracle, MySQL, and SQLite databases. To import the data, we created an empty schema in the database and downloaded the GTFS data from the link provided by the Information Technology department of Izmit. We later passed the schema name and relative location of the downloaded data to gtfsdb which created the tables for all text files

included in the GTFS folder. Using this tool saved us a significant amount of time and gave us the assurance of having correctly imported data on the database.

Once the GTFS data was in a relational database table, it was easy to come up with SQL queries to fetch data on demand based on the various bus routes and bus schedules. In the GTFS dataset for Izmit that was acquired, each bus stop record contained the stop's GPS coordinates. Using this data, given a set of GPS coordinates, it was now possible to figure out the closest bus stop, and all of the bus routes that pass through it.

In case of extension or continuation of this study, the application can easily be improved to work with other transit types such as train and boat transit. GTFS's structure is very standardized, meaning that the stops in a bus transit dataset can represent stations in a train transit dataset. If, in the future, there's a need to extend this application to other areas, all it will take is downloading the GTFS dataset for that area, and running the script to convert it into SQL statements that can be executed to insert the data into the PostgreSQL database, and the application will work as expected. This will also be done in case the GTFS dataset for Izmit's bus system is updated.

4.2.3. Walk directions using Google APIs

Walking directions and slope information (elevation) were retrieved from Google using the directions API and elevation API respectively. Whenever a user searched for directions, say from location A to location B, this request was passed on to the API server. The server then checked the GTFS database and selected the bus stops nearest to A and B, which were also in bus routes linking the two points. Since the bus stops would not always be exactly at the same coordinates as points A and B, the user had to be provided with directions from their origin to the first bus stop. They also had to be provided with directions from the last bus stop to their destination.

Google directions API was used to provide walking directions between these points. Since the target users used wheelchairs, a way had to be found to map a safe route that wheelchairs could navigate without any difficulties. Google's elevation API was quite suited for this since it could provide slope/elevation details between two given points.

By taking the route given by the directions API and splitting it into multiple segments, elevation data was obtained between the start and endpoints of each segment. If the slope between two segments was too steep, it was highlighted in red for steep incline and in blue for steep descent, and thus the user would have an overview of the suggested route and how usable it would be with a wheelchair. In order to reduce external traffic, to reduce costs and to improve speed, the collected elevation and direction data was cached in a PostgreSQL database on the server, and subsequent requests for the same data were served from this cache instead of making fresh requests to Google.

4.3. Sidewalk Mapping

On top of having bus routes and slope detailed walking routes displayed, we added fetching of sidewalk data. However, the available sidewalk map was only of a designated pilot area in the city center. The designated area was chosen with a major bus stop, a public park, a shopping mall, and road junctions in mind. The inclusion of these was to enable more straightforward and more realistic test case simulation.

Since the municipality did not have a ready database of sidewalks and ramp locations in the district, we sought ways of obtaining the data from online open source solutions. We checked OpenStreetMap using the QGIS mapping application. QGIS is a free GIS that can be used to visualize, manage, edit, analyze data, and print customized maps. It has an OpenStreetMap plugin that can be used to download and visualize OSM data using layers. We downloaded the footway and pedestrian data under the highway data type of OSM. The footways and pedestrian paths displayed an overview of their locations in a few areas of the city.

In Figure 4.3, it is clear that the map only contained a rough location of the footways. Pedestrian paths are described in the OSM wiki as a road that is exclusively meant for pedestrians and vehicles may be allowed at specific times only. The pedestrian paths in the city center shown in Figure 4.3 can be seen clearly marked for areas prominently known by citizens.

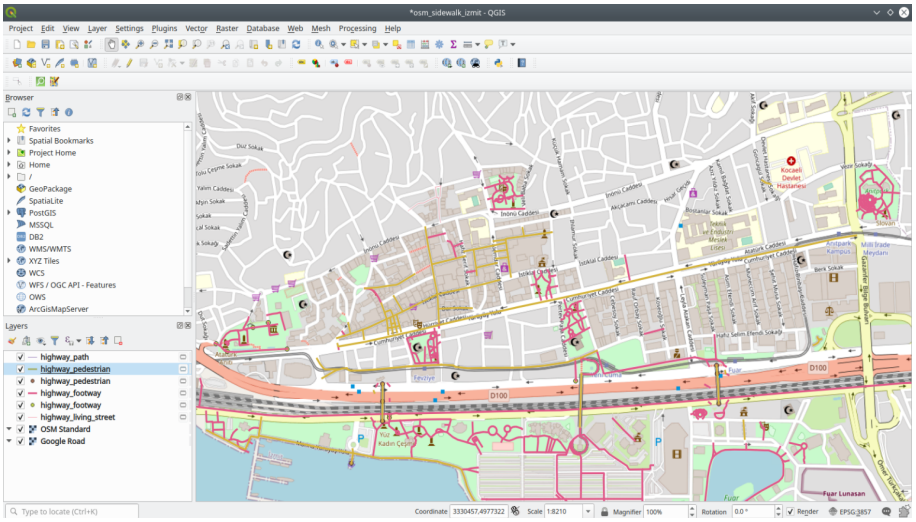


Figure 4.3. Sidewalk and ramps map of Izmit from OSM

Footways in the context of OSM describe paths that are mainly used by pedestrians. OSM provides an option to mark a footway as a sidewalk that can further be marked as being present on the left, right, or both sides of the road. However, the OSM footways map of Izmit only showed footways in a few areas mostly in parks. At the time of our research, we could not find a map that had details of sidewalks and ramps that are on roads, both major and minor, in the city of Izmit. We, therefore, selected a pilot area to map and use in our research.

In our pilot area, we selected a central location in the city. In this area, a mall, a park, a major bus stop, roads with sidewalks and ramps, and a pedestrian path were located. We surveyed the area with the help of one of the interior architects from the Disabled and Elderly Services Branch Office of Izmit. The architect used a tape measure to measure the widths of sidewalks, and a digital inclinometer to measure the elevation of ramps on our path. We also noted down the many physical and structural obstacles that would pose a threat or difficulty to a wheelchair user. We later mapped the area using QGIS mapping software and entered the length, width, elevation, and surface type details of sidewalks and ramps collected from the survey.

Our mapping of the sidewalk followed the paths on either side of roads with sidewalks. The selected area's mapping can be seen in Figure 4.4. The map details and mapping procedure was inspired by Accessmap, a wheelchair accessibility map of Seattle that displays the incline, color-coded wheelchair accessibility status, and type of road

(sidewalk, curb, or crossing) of segments on either side of a road. An example of the map can be seen in Figure 4.5.

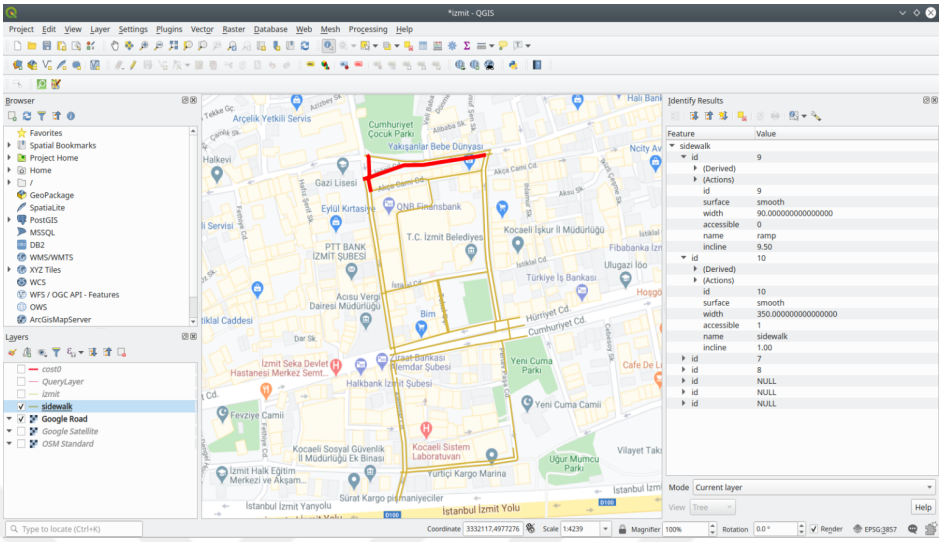


Figure 4.4. Sidewalk mapping of pilot area

Accessmap shows the details of a clicked location. In the example, the clicked location is displayed by the location pointer and the details show that it is a sidewalk made of concrete with a 1.2% incline. The accessibility of the location is partially accessible. The red dashed lines communicate inaccessible paths (Bolten and Caspi, 2019). The accessibility of the paths is measured using the uphill and downhill steepness settings on the left panel and the colors on the map are updated in real-time (Bolten and Caspi, 2019).

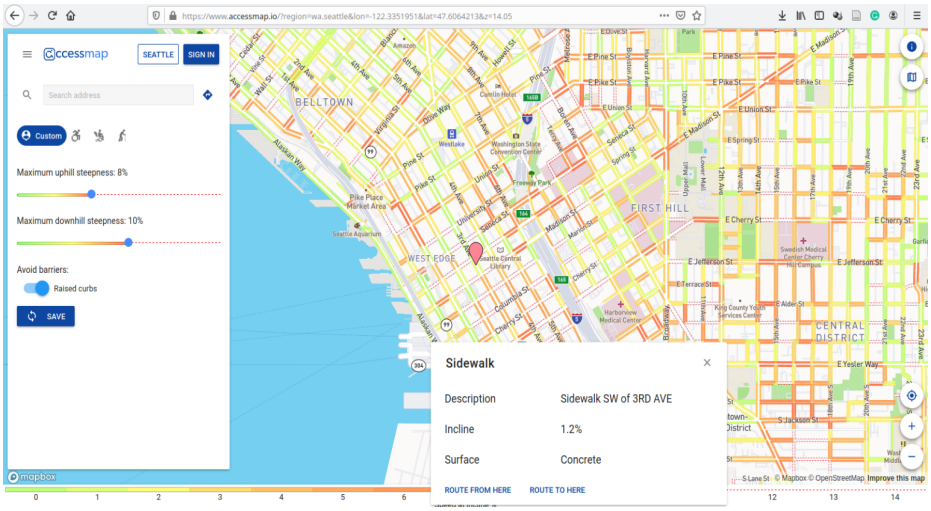


Figure 4.5. An example screenshot of AccessMap

In Figure 4.4, the roads with only one line show pedestrian paths where vehicles were only allowed in specific times of day. The roads with lines on either side shows the presence of sidewalk on roads with an active flow of vehicles. The lines on these roads are segmented into paths with different information on each segment. The information included in the segments included the width of the path, the surface type (smooth or ragged), the average incline of the stretch of the path, the accessibility status (whether a wheelchair user would be able to pass through with difficulty or not), and the name of the path. The name given to the path segment mentioned whether it was a sidewalk, a curb, or a crossing. The map was later exported to a GeoJSON file with all its details. The file was imported into a PostGIS PostgreSQL database for routing functionality.

4.4. Smart Sidewalk Routing using PGRouting

The routing functionality through the manually plotted sidewalk map was implemented using a PostGIS database with pgRouting extension. PostGIS is an extension of the PostgreSQL database with support for GIS spatial objects. It is freely available, open-source, and has geospatial object types and functions on top of the basic geometry types already available in the PostgreSQL database. Spatial databases like the PostGIS database are optimized for handling data that represent objects in a geographic sense. Furthermore, the pgRouting extension of the PostGIS database provides geospatial routing and network analysis functions. The extension contains a vast number of routing functions that use algorithms widely known, for example, Dijkstra, A*, and the bidirectional A* and Dijkstra algorithms. We used the Dijkstra algorithm in our research.

Upon getting the GeoJSON file of the sidewalk map from QGIS software, we imported the map into our PostGIS database using a tool called ogr2ogr, which is part of the geospatial data abstraction library. It is used to convert vector data between different file formats like GeoJSON, PostGIS, and Environmental Systems Research Institute (ESRI) shapefile.

The import tool created a table of our map. Figure 4.6 shows the table columns that corresponded to the details we added to each segment. Each row in the table referenced

an edge on the map. Each edge corresponded to a line drawn on the map that referenced sidewalks, ramps, crossings, or obstacles.

	id [PK] integer	surface character varying	width double precision	accessible integer	name character varying	incline double precision	wkb_geometry geometry	source bigint	target bigint
1	2	smooth	200.04	1	sidewalk	6	0102000020E610000002...	19	20
2	3	smooth	70	1	ramp	2	0102000020E610000002...	20	21
3	4	smooth	90	0	ramp	2	0102000020E610000002...	22	23
4	5	smooth	90	1	sidewalk	7	0102000020E610000006...	24	23
5	6	drainage	90	0	crossing	2	0102000020E610000002...	24	25
6	7	ragged	120	0	ramp	4	0102000020E610000002...	26	26
7	8	none	120	0	barrier	20	0102000020E610000002...	27	28
8	9	smooth	90	0	ramp	9.5	0102000020E610000002...	29	30
9	10	smooth	350	1	sidewalk	1	0102000020E61000000F...	30	31
10	11	smooth	80	0	ramp	6	0102000020E610000002...	31	32
11	12	smooth	150	0	sidewalk	12	0102000020E610000004...	33	34
12	13	smooth	150	1	ramp	[null]	0102000020E610000002...	35	34
13	14	smooth	500	1	pedestrian	[null]	0102000020E610000007...	36	35
14	15	smooth	80	1	ramp	3	0102000020E610000002...	37	38
15	16	smooth	300	1	sidewalk	4	0102000020E610000002...	39	37
16	17	smooth	50	1	ramp	1	0102000020E610000002...	40	41
17	18	smooth	90	1	sidewalk	[null]	0102000020E610000003...	41	10
18	19	smooth	140	1	ramp	1	0102000020E610000002...	10	11
19	20	smooth	80	1	sidewalk	1	0102000020E610000002...	19	11
20	21	smooth	120	1	sidewalk	8	0102000020E610000002...	42	27
21	22	smooth	150	1	pedestrian	[null]	0102000020E610000004...	43	44

Figure 4.6. A snapshot of the PostGIS table contents of imported sidewalk map

The columns for each section of the path included information about the width of the path, the surface type (smooth or ragged), the average incline of the stretch of the trail, the accessibility status (whether a wheelchair user would be able to pass through with difficulty or not), and the name of the segment (whether a sidewalk, ramp, crossing, or obstacle). Additionally, it created a geometry column that stored the geolocation of the segment. The start and end latitude and longitudes of the segment could be derived from the geometry column.

To have a network that correctly maps intersections of roads, we first created a noded system of the map using the pgr_nodeNetwork function as is recommended for pgRouting. This function uses the geometry column of the map table to intersect all edges to find probable junctions. It then splits the intersecting sides into different segments while assigning the edges to a node that link the new segments. An example best depicting this provided by the official pgRouting documentation can be seen in Figure 4.7 (URL-11). The figure shows an example of a graph with crossing edges. After running the pgr_nodeNetwork function, it shows that the resulting network is rid of crossing edges that do not have a node at the intersection.

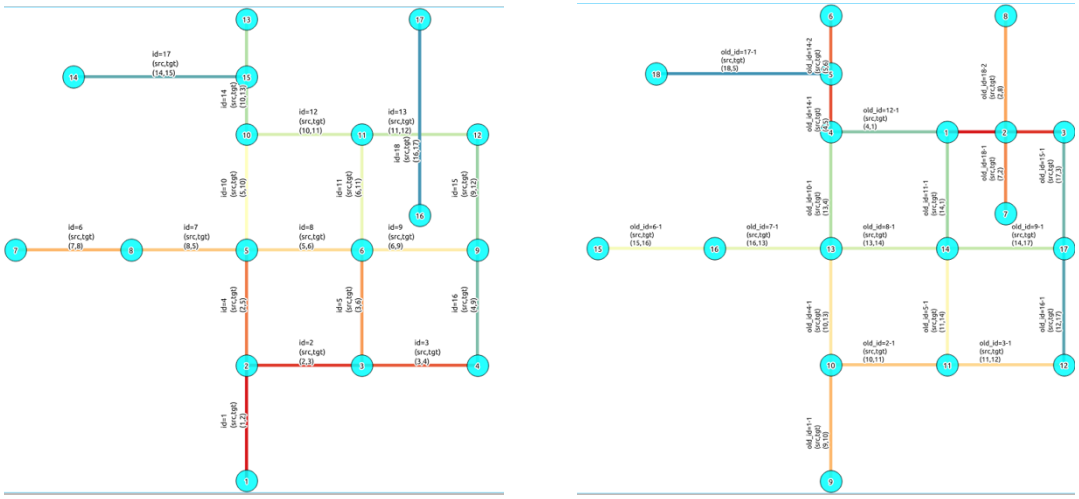


Figure 4.7. Before node and After node (URL-11)

We then built a topology of the map using the noded graph and pgRouting's pgr_createTopology function. This function creates a vertex table of the noded edges. This means that each end of an edge was assigned a unique vertex (URL-12). The edges with connections to each other were assigned the same vertices, which resulted in a network topology of our original map (URL-12). The creation of the topology brings us to the routing functionality.

Getting a path from one point of the map to another was made possible by the available greedy algorithms functions supplied in the pgRouting extension. We used the pgr_dijkstra function to get the shortest path from one point to another using Dijkstra greedy algorithm. The function expects a query for the edges to search through, a starting vertex, and an ending vertex as its arguments. The inner query for the edges to search through contains a cost parameter, which we set as the length of the segment. This ensured that the path returned was the shortest possible path since the Dijkstra algorithm uses the cost of an edge to calculate the shortest route.

The source and end vertex arguments of the pgr_dijkstra function are ids of an edge in a topology map. Since when querying routes, we have the latitude and longitude details of the start and end locations, we first need to get the ids of the edges closest to these locations. In our query, we used the ST_Distance method in PostGIS to find any edges within 500meters of the LatLng points. Moreover, we used the <-> PostGIS operator to get the nearest neighbor edge to the location.

The result set returned from the pgr_Dijkstra method contained an array of edges in sequence that lead from the start to end coordinates. We appended the edges into the response and sent it back to the client app. The edges were then displayed on the mobile app using the same color-coded polylines with the incline column from the table as the slope variable.

4.5. Application Workflow

We created a mobile application using the Flutter framework so that it would be easy to compile for both android and iOS devices. We will however use screenshots from an Android device to exhibit the interface of our application. On a fresh installation, we invoked firebase anonymous authentication function in the firebase library. The function assigns a user a unique firebase id. And for subsequent runs of the app, the same id was used per installation. This id was later used to create a user object in the firestore database. The firebase user object consisted of age, gender, wheelchair type, and accepted fields. The accepted field was set to true when a user accepted the privacy policy.

The user object in firebase was then synced to the PostGIS database which later used it to link ratings to a user. The sync functionality was made possible via the cloud functions in the firebase cloudstore database. The cloud functions updated the users table in the PostGIS database every time a record was created, updated or deleted in the cloud firestore database. Deleted users in the PostGIS database were marked using an is_deleted flag in the users table to maintain integrity of the database. It also ensured that previous rated routes by a deleted user were still available for use in the recommendation generation engine.

With their firebase identifier, the user was directed to the profile page where it was mandatory to accept the privacy policy as can be seen in Figure 4.8. The privacy policy asked for permission to use the user's details that they saved for research purposes only. The user profile screen also had the age, gender, and wheelchair type fields which were saved and used for the smart route recommendations. The profile fields were however optional since the app was foreseen to be used by non-wheelchair users for other purposes such as getting walking directions, or contributing to route ratings.

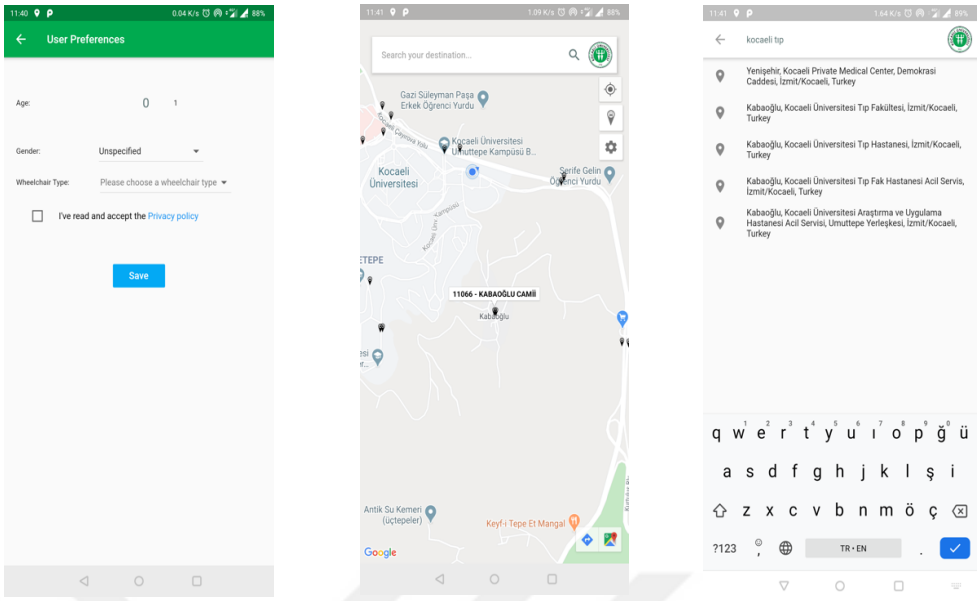


Figure 4.8. User profile, Main screen, and Search screen examples

On accepting the privacy policy, the user was directed to the main screen of the app. The user's current location was first fetched. We assumed that the user's route request origin would be their current location. Sometimes the location takes long to be fetched but this did not hinder a user to continue using the app. The user could manually set the location of origin when performing their route request.

The main screen had a full google map view centered at the user's current location. The map showed all nearby bus stops in the area by default. Clicking on a bus stop icon opened a window with the code and name of the stop. The bus stops were automatically fetched and displayed any time the map was moved. However, the bus stops display icons could be switched off using the location pointer button situated on the top right-hand side of the map view.

The buttons on the top right also included a settings icon button which redirected a user back to the profile page in case they would like to make any changes in their profile. The location button centered the map on a user's current location, if found, or last known location, if current location was not found. At the top of the main screen shown in Figure 4.8, a search box allowed a user to search for a destination to be mapped. The search box opened up a search screen. The search used Google's place autocomplete API to get areas that matched the text typed. The API returns a maximum of 5 results which were displayed in the search screen. A user was then expected to

select a search result so as to set the location as the destination. Selecting a location from the search results triggered another request to the Google place details API. The API returns a place's full details like the address, reviews, and location which were not included in the autocomplete results. The location coordinates and address description extracted from the place details API was then passed back to the main screen where a red location icon was added at the destination location selected as can be seen in Figure 4.9.

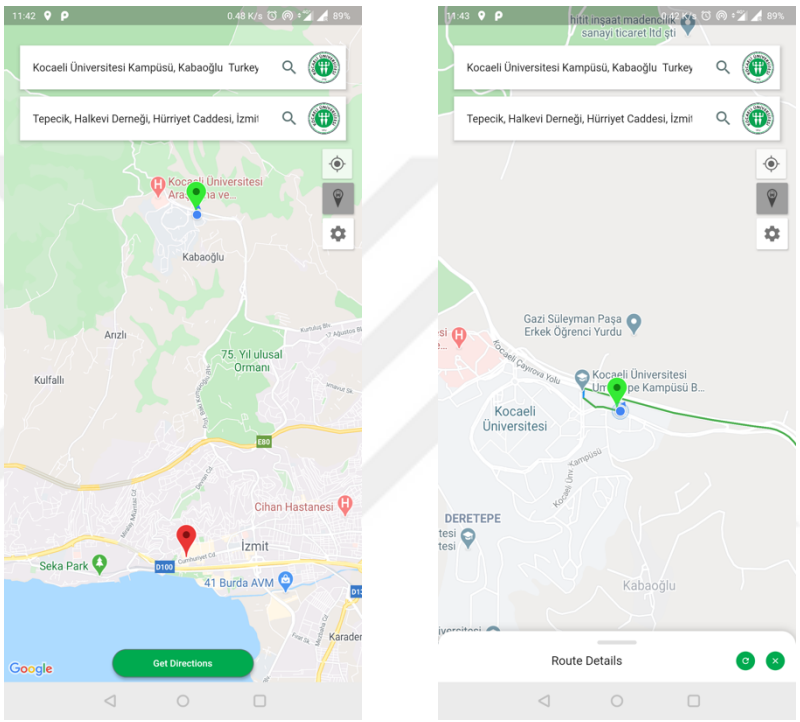


Figure 4.9. Before and After route request

Setting the destination location automatically set the origin location to the current location, or to the last known location if the current one was not available. This also added a green location icon at the retrieved location. The user was also free to change or manually reset the origin location using the origin text box that was displayed on top of the destination one. Clicking on the origin text box opened the search screen in Figure 4.8 and the same process as the destination setting process is followed. Figure 4.9 shows an example of the origin set to the Kocaeli University hospital and the destination as Halkevi in Izmit city center. When both the destination and origin were set, the route request could then be triggered using the get directions button visible at the bottom of the screen as seen in Figure 4.9.

The route request was sent to the API server which searched for the direct walking directions from origin to destination. It then looked for the nearest bus stops within 1 km to the origin and destination. Using these bus stops, it queried for all bus routes that went through them. Since a bus route with different schedule times considered each time as a different trip, the results sometimes returned the same route but with different arrival times as different routes. Therefore we added a time limit in our bus route query. The limit queried for all bus routes that would be at the particular stop within the next 30 minutes of the querying time. For all the returned bus routes, we searched for either the sidewalk path or walking path, if the sidewalk path was not found, to the start stop and from the last stop. The walking and sidewalk paths both included slope details of different segments of the path. All these results were then bundled together and sent back to the client as seen in Figure 4.9. The figure also shows the collapsed version of the slide-up panel that is used to display the different route options returned from the server.

The bottom panel with the returned route results can be seen in Figure 4.10. The panel had three sections. The first section displayed the smart recommendations, if any were found. The second section showed direct routes (where no bus route is provided). And the third section showed the wheelchair and bus routes (wheelchair navigation to the first bus stop, then bus navigation to the last bus stop, and finally wheelchair navigation to the destination). The panel also had a refresh icon button which re-fetched routes for the same locations. A close icon button closed the panel, removed the origin and destination locations and reset the map to the state shown in Figure 4.8. It could be used when a user needed to reset the map and restart a new route request.

The polylines drawn on the map always coincided with the selected route option in the slide-up panel. This meant that whenever a user selected a different option, the polylines on the map were redrawn to show the selected route. Figure 4.10 shows an example of a selected bus route. The map has been zoomed out to fit the origin and destination in the visible canvas.

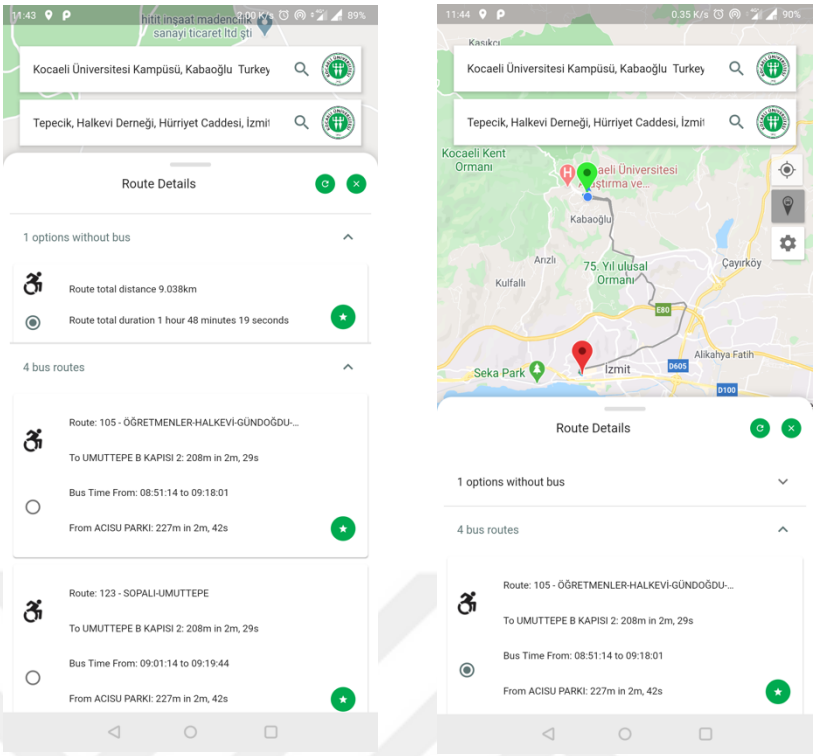


Figure 4.10. Route request outcome

The bus route selected in Figure 4.10 is represented by the grey polyline on the map. The gray polyline meant that the elevation information was not available. We did not display the elevation details of a bus route because the vehicle would still pass through all areas it was designated to pass without a problem. The details of a route displayed to a user included the bus number, the distance and average time from the origin to first bus stop, the expected departure time at the first bus stop and arrival time at the last bus stop, and the distance and average time from the last stop to the destination. The distance between a location and a bus stop was found from the Google directions API. The average time from a location to a stop was however calculated using the assumption that a wheelchair user travels at an average speed of 1.4m/s.

The sidewalk or walking directions were color coded using the elevation value of each segment of the path. Figure 4.11 shows an example of a path from a bus stop to the destination. The colors on the path show different elevation values which translate to paths that would be either hard or safe to navigate for wheelchair users.

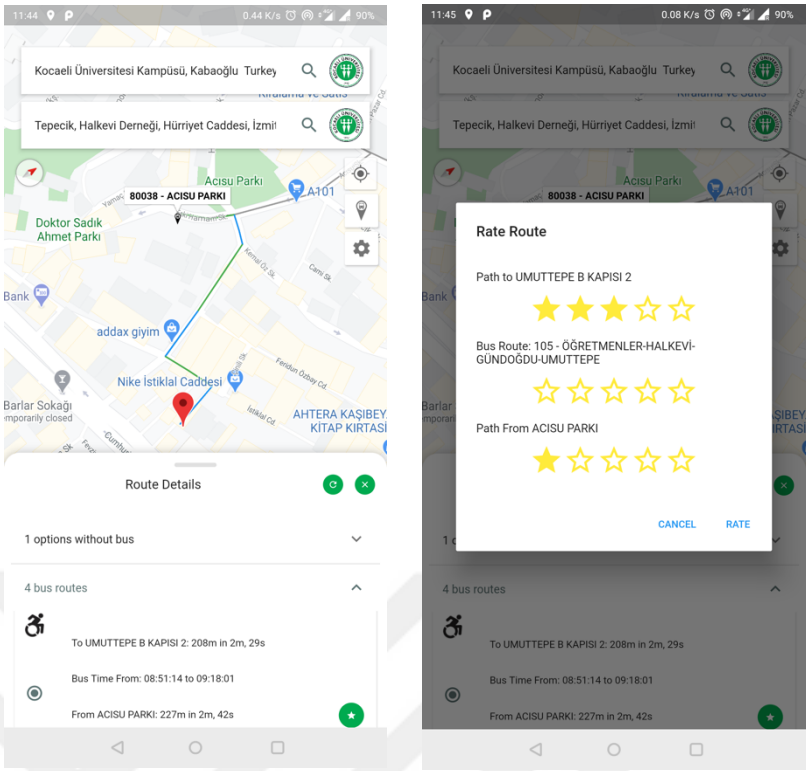


Figure 4.11. Color-coded Directions and Rating Dialog

The red polylines indicated an incline of greater than seven degrees while the blue polylines indicated a descent of greater than seven degrees. The green areas showed sections that had elevation of between zero and seven degrees uphill and downhill which were considered safe for wheelchair users . Even though we learnt that a safe elevation value is seven degrees, future development of the application could add a custom minimum and maximum elevation value for a user to manipulate according to their personal needs and preferences.

Finally, the app provided a rating option for each route returned in the results via a dialog visible in Figure 4.11. A user would rate a route without actually going through it because we learnt that wheelchair users usually learnt and followed the same paths in their daily life. Hence users could provide their feedback on routes that they had prior knowledge on. We used a five star rating system in our application. A user could fill the number of stars and submit their rating as many times as they want. When retrieving routes, the average rating of a route from all users was returned. The rating returned could be seen on opening the rating dialog.

Once the application was almost fully developed, it was released to the Google Play store. This process involved compiling the code into an Android Package (APK) and submitting it for review. After the first review, Google rejected the application due to a lack of privacy policies, terms and conditions, and fatal bugs that had to be fixed before they could accept it again. Privacy policy and terms and conditions were included in the application, alongside a request for the user's permission to use their details for this research.

4.6. Collection of Data

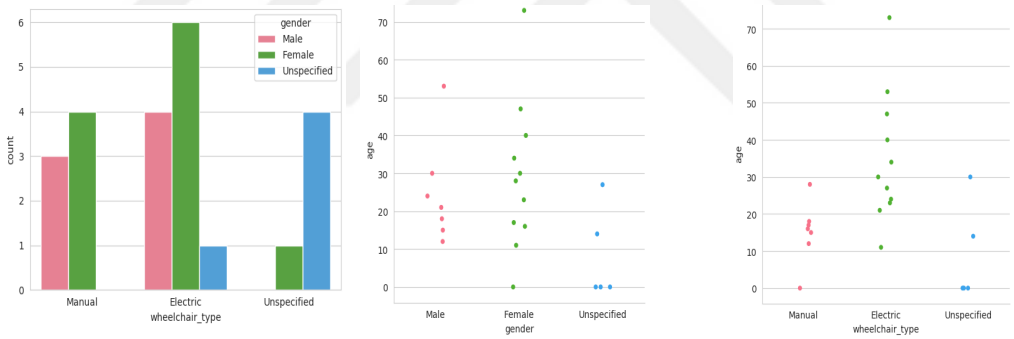
Our study's route recommendation process was majorly dependent on the clustering of users and the ratings each user submitted. The average rating of a route was calculated the mean of the points assigned to the route by users in the same cluster. Routes recommended to a user making a query were based on a few criteria. The cluster rating of the route was supposed to be at least three points, and the origin and destination points of a route were supposed to be within 100 meters of the origin and destination points of the rated routes.

The grouping of users started when a user installed the application on their device. Subsequent clustering of users was triggered after a specified number of new users were registered into the system. We used a small amount of five new users to trigger clustering in order to deliver a working demo of the system. However, studies that were planned to be conducted with the special needs citizens of the city of Izmit were distracted due to a global pandemic.

The pandemic brought about by the infectious coronavirus disease, also known as Covid-19, led to closing schools and public areas in Turkey in March, 2020. The epidemic led the country, and the world at large, to an era of quarantine, lockdown, social distancing, and use of sanitizers and masks. Observing distance between people was of paramount importance to minimize the spread of the virus and the disease that was responsible for the loss of many lives worldwide. In Turkey, the number of cases increased sharply to 13,531 infected people and the death toll to 214 people by the end of March after the first case was reported on 10th March 2020 (URL-19).

The sharp increase in cases translated to stricter measures and a stay-at-home policy that lasted for almost four months instead of an originally announced three-week quarantine duration. All students went back to their homes, while student dormitories were converted to in-patient hospital wards. Foreign students were all moved to one designated dormitory per number of cities. Going out for non-essential workers, inter-city travel, and international travel was prohibited with the increasing number of cases and deaths caused by the pandemic.

These measures also affected schools, which retaliated by converting the classes to online video platforms. As a result of all these measures and restrictions, we could not carry out our field study to completion as has been planned. Therefore, we updated the app to accept ratings of a route without actually going through the route for simulation purposes. The data used to simulate the workflow of the app was also collected using feedback from the interviews with and observation of the wheelchair users and the architects conducted at the beginning of the research.



(a) (b) (c)

Figure 4.12. a) Number of users based on gender and wheelchair type b) Distribution of age per gender c) Distribution of age per wheelchair type

A portion of the users in the final dataset included wheelchair users who did a few tests before the onset of the pandemic. The other portion was created using previously collected information about the users, their hardships, experiences, and recommendations. Consequently, our simulation of the application used profiles of 20 people. A graph of the distribution of the profiles based on the three features we looked at in our study can be seen in Figure 4.12. The first graph shows the number of people based on their gender and wheelchair type. The unspecified gender and wheelchair types denote users who decided not to share details of their gender and type of

wheelchair they use. This was added to enable non-wheelchair users to submit ratings on routes that may benefit wheelchair users using the system. The graph communicates that there were mostly female users with electric wheelchairs and in overall most users had electric chairs.

Table 4.1. User profiles

Gender	Age	Wheelchair Type
Male	18	Manual
Male	30	Electric
Male	24	Electric
Male	21	Electric
Male	12	Manual
Male	15	Manual
Female	16	Manual
Female	11	Electric
Unspecified	14	Unspecified
Female	34	Electric
Female	23	Electric
Female	47	Electric
Female	28	Manual
Female	73	Electric
Female	17	Manual
Male	53	Electric
Female	30	Unspecified
Unspecified	27	Electric
Unspecified	0	Unspecified
Unspecified	0	Unspecified

The second and third graphs in Figure 4.12 shows the age brackets of users in both gender and wheelchair type. It is clear that there were a few people older than 50 years of age and the larger population of wheelchair users in our dataset are users between 10 and 40 years of age. We also understand that there were mostly female users and most of the users used electric chairs. Table 4.1 shows details of each user profile used in our research. A few observations that can be made are that there were 9 females, 7 male, and 4 with unspecified gender; 10 electric, 6 manual, and 4 unspecified wheelchair type users. Of these females, 5 used electric chairs, 3 used manual chairs, and 1 did not specify. In the case of males, 4 used electric while 3 used manual chairs.

Three users did not specify either gender nor wheelchair type while one electric wheelchair user did not specify their gender. Apart from the user profiles, we also had routes that were used in the simulation.

Table 4.2. Overview of routes used for simulation

Route Number	Route Origin	Route Destination
1	Kocaeli Tepekoy KO-MEK	Belsa Plaza
2	Tepecik Halkevi	Kocaeli University, Faculty of Medicine
3	Kocaeli Toki Konutlari	Kocaeli Municipal Hospital
4	Umuttepe Bus stop	Kocaeli University, Faculty of Engineering, Blok A

These routes were also selected based on the frequented areas of our study subjects. A summary of the start and end point of the chosen routes can be seen in Table 4.2. The first route was one that started from a public institution named KO-MEK in the Tepekoy neighborhood to the Belsa Plaza at the center of the city square. The destination was chosen specifically because it lay within the area with the sidewalk mapping. The second route led to the Kocaeli University faculty of medicine building from a popular area in the city called Halkevi. This route imitated one of the most frequented routes with many citizens going to the university from town.

The third route originated from a residential complex named Toki to the municipality hospital. Many wheelchair users had frequent visits to the hospital which was covered by this route. And finally, a route from the bus stops at Umuttepe to block A of the engineering faculty in Kocaeli University was chosen to replicate the daily route of one of our test subjects. The last route did not have bus options since it was within the university where no bus routes pass through. It was chosen to also simulate the recommendation of routes that do not rely on buses.

Each of the users went through each route and assigned the rating that best fit their normal experience on a five star scale. We did not put a limitation on the number of routes or the bus routes that needed rating. The ratings were given based on the returned routes and bus options at the time of querying.

At the end of the data collection phase, we had a total of 15 routes with ratings both good and bad as displayed in Table 4.3. The table also shows the number of users who

rated each route and the overall rating of each route based on all the points it received. Routes with a high frequency of rating showed that there were not many alternatives of buses at the location that went to the destination queried.

Table 4.3. Total number of ratings per route

Route Short Name	Number of Users	Overall Rating
bus-105	1	4
bus-111	14	3
bus-115	2	3
bus-13	1	4
bus-145	5	4.8
bus-24	1	1
bus-255	1	2
bus-33	4	2.5
bus-43	1	5
bus-70	1	1
bus-72	7	3.29
bus-85	2	3.5
bus-92	10	2.9
sidewalk	17	3.18
walk	4	2.17

For example, bus number 111 had 14 ratings because there were no other bus choices when querying from Toki housing to Kocaeli municipality hospital. Bus number 92 also had a high rating frequency because there were not many frequent buses from Tepekoy KO-MEK to Belsa Plaza. Even though bus 72 was also an option for this route, the bus was dispatched once in an hour as opposed to bus 92 which was dispatched two to four times an hour depending on the time of day and day of week.

It was also important to note that the average rating of each route calculated during recommendation was dependent on the queried origin and destination locations. Since bus routes passed through multiple different stations, we reserved a rating made for a route based on the start and last stop of the route. For example, bus number 33 is famously known for its town to university route that is notoriously crowded during school days. Some users preferred to alight at gate B of the university which is one stop from gate A, the last stop of the route at the university. Let us assume we are dealing with users in the same cluster. If a user searched for a route to gate A and gave

bus 33 a good rating, and another user searched for a route to gate B and gave bus 33 a bad rating, its overall rating would be average. However, if another user searched for a route to gate B, bus 33 will not be recommended because the bus had a bad rating when the destination was gate B. But if they set gate A as the destination, bus 33 would be recommended since it had a good rating.

In summary, our study did not consider the same rating for overlapping routes. Each rating is exclusive to the origin and destination locations of a route. The main reason behind this was that the paths to and from the bus stops mattered more than the bus itself. However further development of the app can add the ability to rate buses independently of the route and stops in the query.

4.7. K-means Clustering and Smart Route Recommendation

With the routing functionalities of sidewalks and bus routes in place, we added an intelligent recommendation for our target users. The suggestion was deduced by using the average scores of routes made by users in similar clusters. The user's group was, in turn, determined via clustering users based on their profile using the K-Means clustering algorithm.

The K-Means clustering algorithm is a type of unsupervised learning algorithm. This means that it is not given prior data with known output to train and learn from. Instead, it uses data and features to look for patterns in a dataset, which provides meaningful insights about the data. The main objective of K-Means algorithms is to look for underlying patterns in the observations of a dataset and output k number of groups from the dataset based on the underlying similarity.

To cluster data points, K-Means assigns k number of random centroids. The average distance of each data point to each centroid is then calculated. The data points with the least distances to a centroid are assigned to that centroid's cluster. The algorithm then recalculates its new center point in its group by averaging its distance to each data point for every feature being used in the clustering functionality. This is the first iteration of the K-Means algorithm. With the newly calculated centroid locations, the algorithm then calculates the distance of each data point to each centroid again and assigns it to the centroid's cluster with the least distance. The centroid's central position

is then recalculated again. This iteration continues until either the centroid's spot in the group does not change or the iterations specified are reached.

To get the optimum number of k clusters for our dataset, we used the silhouette coefficient method. This method assigns scores based on the average cohesion and separation distances. The cohesion distances of samples within their clusters and the separation distance is the minimum distance between the samples and other clusters. The score is then calculated using Formula (3.1). Silhouette values close to a value of 1 indicate good clustering while values close to -1 indicate wrong clustering. Since comparison of silhouette scores are manageable using programming, using this score made it easier and more reliable than using the elbow method. The elbow method is mostly used to output a graph of SSE against the number of clusters. Since the SSE values percentage differences are not uniform nor predictable, it is not easy to determine the optimal k value using mathematical formulas in code. Hence its automation proved to be challenging and we opted to use it as a validation option instead.

In our research, we used three features to cluster registered users. The features were age, gender, and wheelchair type. Since the K-Means clustering algorithm worked by calculating distances to the centroid, we had to convert the categorical features into numeric values. The gender feature had male and female categories, while the wheelchair type had manual and electric categories. Gender and wheelchair type columns were assigned numeric values using the one-hot encoder function available in the scikit learn python library. The one-hot encoder function was used to encode nominal or named features like gender types into numerical categories. It basically creates a new column for each category in the feature. It assigns either a one (for samples with the category) or a zero (for samples without the category) to the samples. For example, Table 4.4 shows a sample of the first five users who joined the application. For the gender category, the encoder created a gender_Male, gender_Female, and gender_Unspecified features. It then assigned gender_Male to one and gender_Female to 0 for male users and vice versa for female users. The resulting encoded data for the sample raw data in Table 4.4 can be seen in Table 4.5. The vital point to note is that the category labels have to be precisely the same so that

the new numeric columns are equal to the number of categories in the feature, and we do not have redundant columns.

Table 4.4. Sample of user profile raw data

user_id	gender	age	wheelchair_type
85	Female	34	Electric
87	Female	23	Electric
88	Female	28	Manual
89	Female	17	Manual
90	Female	30	Unspecified

Table 4.5. Example output of encoded and standardized user profile data

gender_ Female	gender _Male	gender_ Unspecified	wheelchair_ type_ Electric	wheelchair_ type_ Manual	wheelchair_ type_ Unspecified	age	user_id
1	0	0	1	0	0	1.291293	85
1	0	0	1	0	0	-0.577684	87
1	0	0	0	1	0	0.271851	88
1	0	0	0	1	0	-1.597125	89
1	0	0	0	0	1	0.611665	90

After having categorical features converted to numeric values, we standardized the data. Standardization of data in K-Means clustering improves performance by giving all features in a dataset equal weight as explained by (Sharma, 2019). He continued to explain that not normalizing features led to bias when clustering data points because the algorithm's unit of measure is the distance between a data point and the center in terms of feature values. He also showed with example that the algorithm is biased towards variables with greater variance. Hence normalizing data helped to reduce this bias.

We used scikit-learn's standard scaler to standardize our age variable. The scaler calculates the mean and standard deviation of the age feature, then for each observation, it calculates its new value by subtracting the mean value from the

observation value and dividing the result by the standard deviation using Formula (4.1). The result of the standardized age values can be seen in Table 4.5.

$$z=\frac{x-\mu}{\sigma} \tag{4.1}$$

With our standardized data ready, we passed our dataset through the silhouette coefficient method to get the optimum number of clusters that would best group our users. We used scikit-learn's K-Means function to perform the clustering in our research. In the silhouette coefficient method, we calculated the silhouette score of clusters ranging from 2 to a maximum of 20. The optimal number of clusters chosen became the one with the highest score. The silhouette score values for our 20 test users can be seen in Figure 4.13. It is clear that the optimal k is 10 with a score of 0.522. Hence that is what we used for our clustering.

```
For n_clusters = 2 The average silhouette_score is : 0.3049869077463677
For n_clusters = 3 The average silhouette_score is : 0.30984542488629974
For n_clusters = 4 The average silhouette_score is : 0.37031292025004997
For n_clusters = 5 The average silhouette_score is : 0.39256359576992317
For n_clusters = 6 The average silhouette_score is : 0.47082727055382884
For n_clusters = 7 The average silhouette_score is : 0.49616125947960404
For n_clusters = 8 The average silhouette_score is : 0.4774087531377318
For n_clusters = 9 The average silhouette_score is : 0.507085169305788
For n_clusters = 10 The average silhouette_score is : 0.522135335304332
For n_clusters = 11 The average silhouette_score is : 0.4554161176571201
For n_clusters = 12 The average silhouette_score is : 0.45066661170018907
For n_clusters = 13 The average silhouette_score is : 0.44062710359478235
For n_clusters = 14 The average silhouette_score is : 0.4283742742747775
For n_clusters = 15 The average silhouette_score is : 0.3748270757695048
For n_clusters = 16 The average silhouette_score is : 0.2746212121212117
For n_clusters = 17 The average silhouette_score is : 0.24962121212121163
For n_clusters = 18 The average silhouette_score is : 0.1912878787878784
```

Figure 4.13. Silhouette Coefficient method output.

The dataset was then passed through the K-Means clustering algorithm, with k=10, where the clustering of all registered users was completed, and the cluster model preserved using the joblib library. The process of getting the optimal k number of clusters, clustering users, and preserving the model was triggered whenever a new user registers on the app. However before any new clustering was performed, we checked whether the number of new users since the last clustering had reached its threshold. For our research, we set this threshold at five users. Meaning that after every five new

user registration, the clusters were updated. Hence the cluster of users were kept up to date and offered more realistic recommendations.

Whenever a user requested for routing, we first tried to find a smart recommendation using the ratings of other users in their cluster. The user's cluster was found using the saved K-Means model, which enabled us to get the other users in the same cluster as well. We then checked for any ratings for routes that had the same origin and destination as the requested route. We filtered these ratings based on the users in the cluster and got the average score of each unique route. If the average rating of a route exceeded the minimum threshold of a rating of three out of five, we consider this route a good recommendation and passed it back to the requesting client along with the other options.

The display of the route on the mobile app followed the same procedure as all the others. The route’s incline or slope value in each segment of the path was used to decide the color of each section. Users were also able to rate the recommended route, as well as all the other routes as many times as they wished.

5. FINDINGS AND DISCUSSION

As with the majority of studies, the design of the current study was subject to limitations and assumptions. Most of the information on the details of a path were retrieved from Google API. The place search, directions, and elevation details were all reliable on Google API. Even though the APIs had a free tier provided by Google, the free credits were easily drained by the many requests from users. This can however be worked around by using the free and open source APIs such as OpenStreetMap. The mapping of sidewalk also involved a lot of tedious manual work even for a small area of the city that we chose. While our study only focused mapping the sidewalks on a small area of the city, it provided insights on the results that could be reached if it was done on a larger scale. The mapping could be crowd sourced as implemented by some researchers (Hara et al., 2013, 2014) to reduce the manual work. The sidewalk maps would at least provide wheelchair users some more helpful information on routes to use in different areas of the city without much difficulty.

When displaying the bus route options, we used the approximated times at bus stops to query the buses that were likely to arrive within half an hour which might not be the case. Even though most buses in the city adhere to the timetables set for them, other issues on the ground such as traffic, accidents, and social events may cause differences in the times and routes. Use of the real time locations of buses which would provide more accurate times would solve this limitation. Furthermore, users can double check the bus times and locations using the official E-komobil application developed by the city municipality.

One of the main areas of our research was to get the safe and easy to maneuver route for a wheelchair user. However, temporary obstacles on a user's route such as construction works, parked cars, closed roads, and trash cans would pose a problem for a wheelchair user. We focused mainly on the road elevation and sidewalk status to provide route recommendations. Some researchers used the surface types, Google Street View images , and crowdsourced applications to gather data on such obstacles

(Cáceres et al., 2020; Edinger et al., 2019; Gani et al., 2019; Hara et al., 2014; Prandi et al., 2017). Some of these can be incorporated to make the app better suited for everyday use.

Our research depended on ratings of routes by users to generate smart recommendations. These kinds of implementation usually suffer from cold start problems because there is no prior data from which to generate a recommendation. We therefore provided the ability to rate routes just by searching for them and rating them without having to travel through it first. Thus enabling users to provide a rating for previously learnt routes that would help others instantly.

5.1. Clustering Results

K-Means clustering of our users was triggered after every fifth new user. The clustering automatically updated the K-Means model saved in a file on the system as a Python object. The number of clusters was also automatically selected by comparing the silhouette score of clusters between 2 and 20. This meant that a function loops between 2 and a maximum of 20 clusters and calculated the silhouette score of each clustering result. Finally the number of clusters that got the highest silhouette score was assigned the k value which coincided with the optimal number of clusters.

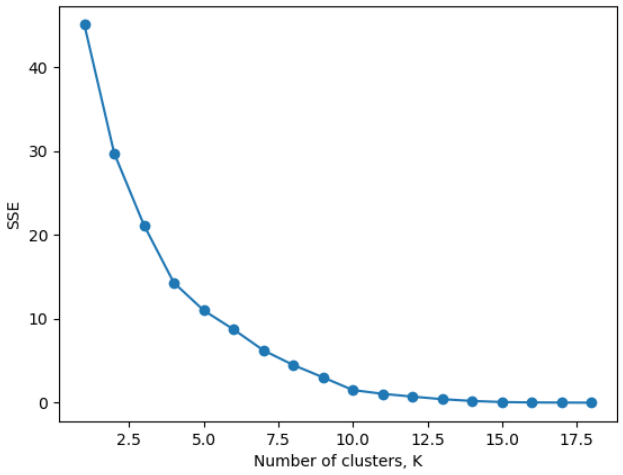


Figure 5.1. Elbow method graph of SSE against K for 20 users

We analyzed the k value returned by the silhouette coefficient method for the 20 users with the graph of SSE against k values, popularly known as the elbow method. The

graph yielded by the elbow method for the 20 users is shown in Figure 5.1. The graph shows that 10 clusters would be an ideal number, similarly to what the silhouette coefficient returned as the optimal number of clusters.

Using k=10 as the number of clusters and the encoded and standardized data of the 20 user profiles collected, the output of the clustering can be visualized in Figure 5.2. The first chart shows the graph of the standardized age values against the cluster numbers while the second graph shows the original age values against the cluster numbers for easier interpretation. The first chart also indicates the location of each cluster centroid. The resulting 10 clusters visible on the graphs show that each cluster had a specific wheelchair type and gender. Furthermore, the clusters were seen to be separated based on the age brackets.

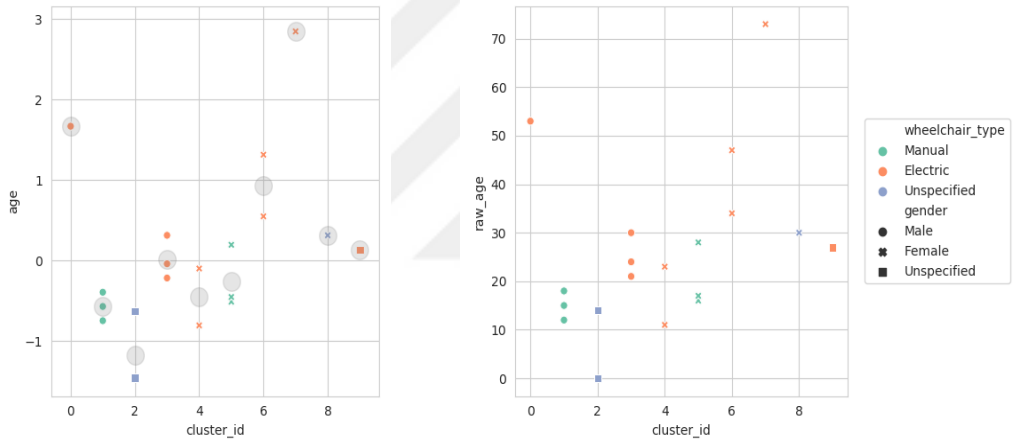


Figure 5.2.K-Means clustering result

For example, the users in cluster 4, 6, and 7 were all females with electric chairs. However, they were grouped in different clusters because of their age. Users in cluster 4 can be described as females with electric wheelchairs who were below 30 years of age, those in cluster 6 were females with electric wheelchairs who were between 30 and 50 years of age, while those in cluster 7 were females with electric wheelchairs who were above 50 years of age. The same explanation applied for male users with electric wheelchairs in clusters 0 and 3. The rest of the clusters had users of uniquely separated gender and wheelchair types.

To further analyze the clusters of the users in our dataset, graphs of gender against the cluster number and wheelchair type against cluster number were drawn as exhibited in

Figure 5.3. These graphs show that each cluster had a specific type of gender as well as wheelchair type. And as for clusters that have similar combinations of the gender and wheelchair type groups, the graph in Figure 5.2 confirms that the clusters were differentiated based on the age values.

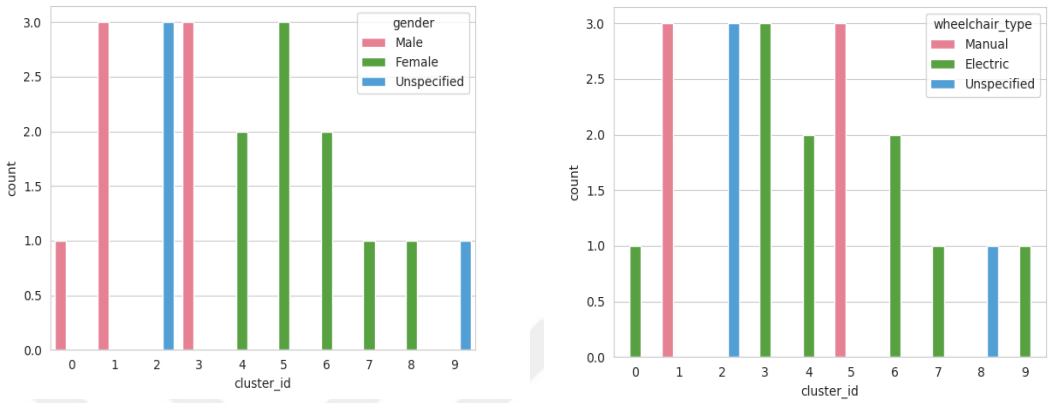


Figure 5.3. Distribution of gender and wheelchair types per cluster

Since the number of users in our dataset was limited due to unavoidable circumstances, we decided to run a test on a dataset with generic data of around 1400 user profiles. This data was generated by assigning random gender and wheelchair type from the respective pre-defined options. The age value was also selected at random in the range of between 10 and 80 years.

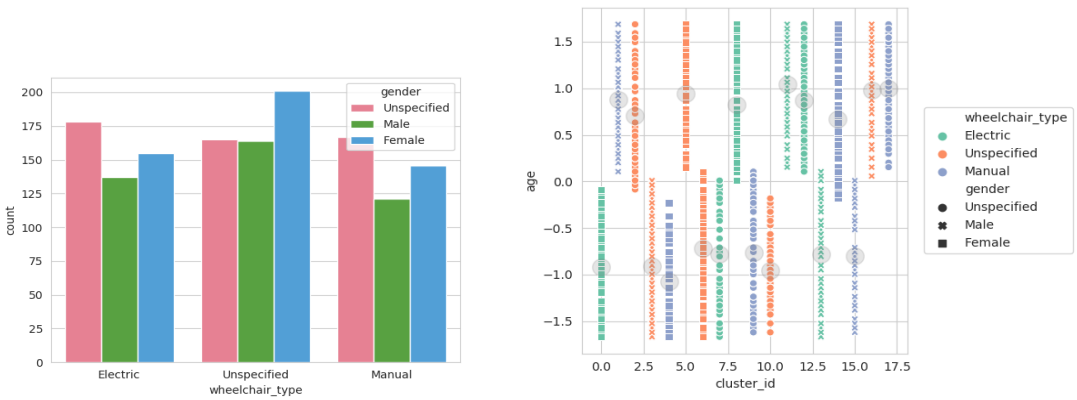


Figure 5.4. Test data distribution and clustering result

Figure 5.4 shows the number of each gender per wheelchair type and the resulting 18 clusters. The outcome is consistent to the result found with our test users but on a larger scale. It shows that the groups each have a specific gender and wheelchair type within a specific age group. During a route query, a user's cluster was first retrieved to

determine the route average rating in their cluster. For the new users that requested a route before being updated in the clusters, the previously clustered model was used to determine the cluster that best suited them. A test user who was a 32-year-old female with an unspecified type of wheelchair was used to test the correctness of the cluster model created using the 20 original users. The outcome of her cluster is as depicted in Figure 5.5. The grey circle shows the test user location who was assigned to cluster 9. This cluster previously had one member who was a 30-year-old female with an unspecified wheelchair type. The routes recommended to the test user would therefore be good rated routes as per the points received from the members of cluster 9.

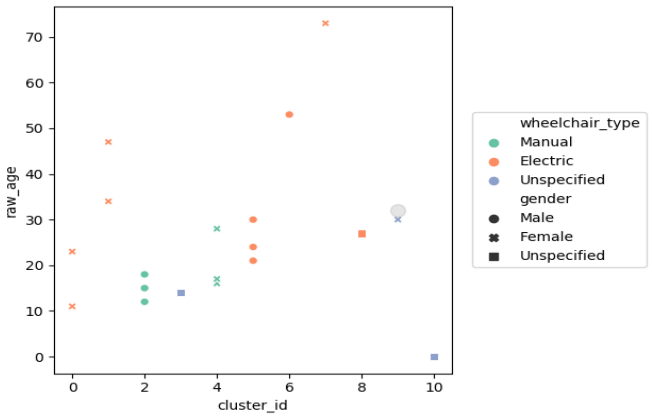


Figure 5.5. Clustering of a test user

In an attempt to confirm the clustering functionality of our model, we used five user profiles as described in Table 5.1. Three of the test users had similar characteristics to previous users in the clusters while two of them were unique. The unique profiles were both male, one was 25 and the other was 63 years of age, and one had a manual chair while the other was not specified. The aftermath of clustering the test users is indicated by the graph in Figure 5.6. The markers depicting the new users are a bit larger in size than the markers of the previous users.

The results showed that the users with similar profiles as some of the previous users were assigned to the cluster that matched their profile. For example, the user assigned to cluster 0 is similar to the others since the user was a 40 year old female with an electric chair. Similarly the users assigned to cluster 9 and 2 both fit in perfectly. Also the two unique users were assigned to the cluster that best fit their profile.

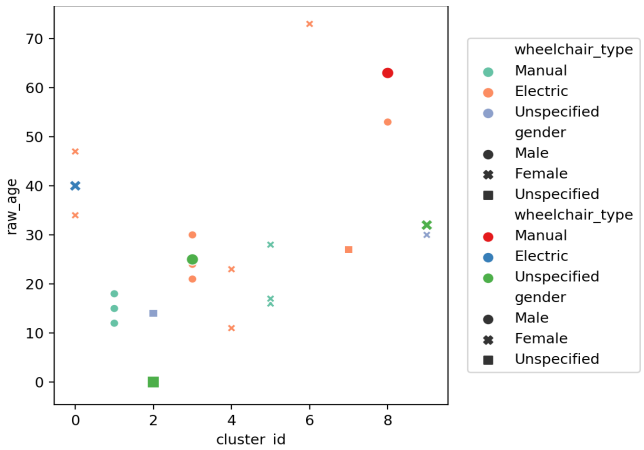


Figure 5.6. Clustering of test users

Table 5.1. Test user profiles

Age	Gender	Wheelchair Type
63	Male	Manual
40	Female	Electric
32	Female	Unspecified
25	Male	Unspecified
0	Unspecified	Unspecified

For the 63 year old male with a manual chair, he was assigned to cluster 8 which had males with electric chairs who were greater than 50 years of age. A closer look at the clusters revealed that there was a cluster 1 that had males with manual chairs. However, cluster 1 users were all below 20 years of age. Thus showing that the cluster based its decision on the age factor. Similarly, the 25 year old male with an unspecified wheelchair type was assigned to cluster 3 instead of cluster 8 or 1 because of his age. The males in cluster 1 were below 20 years while the ones in cluster 8 were above 50 years. And since there was no cluster with males with unspecified wheelchair types, the decision was established by his age.

After these five test users registered on the app, the clustering was triggered and the outcome is depicted in Figure 5.7. The three users with similar characteristics as previous users can be seen to be retained in the previous clusters that they were assigned even though the cluster numbers were updated.

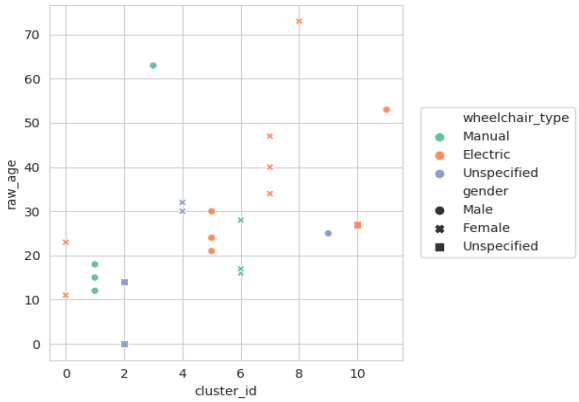


Figure 5.7. Result after clustering the test users

However, the two unique profiles resulted in two new clusters, bumping the number of clusters from 10 to 12. The cluster 4 and 9 in the 12-cluster graph were the new clusters added with each having the unique characteristic of their members. This translates to mean that the recommendation returned by the system would update accordingly based on the new clusters formed after the addition of every five new users, or whatever threshold is set.

5.2. Route Recommendation Analysis

The wheelchair router app that was the product of our study had one main functionality which was to return available routes from one point to another in the district of Izmit in the municipality of Kocaeli in Turkey. The start and end locations were entered by searching their names as identified on Google Maps. A button that exclusively submitted a query to the server would then appear after both the origin and destination was set. The query would be posted to our servers where routes were queried in three phases. First, a direct route from the start to end position was queried using the Google Maps direction API which would later be reconstructed by querying and adding the elevation value to each segment of the path. Second, bus stops within a kilometer of either location were searched. For every combination of the bus stops found, we explored the GTFS database of the bus routes to look for any available direct bus routes. For every route found, a sidewalk would then be queried using the Dijkstra pgRouting functionality to and from either bus stop, and if none was found, the walking path from the directions API which would be added to the bus route. And

finally, we filtered the routes whose rating was made when a user had searched for an origin and destination within 100 meters of the current query locations. These routes were further filtered based on the clusters in which the user who rated them belonged. The average rating of the routes submitted by users in similar clusters as the current user were then calculated and routes with an average score of three or greater were returned as recommendations.

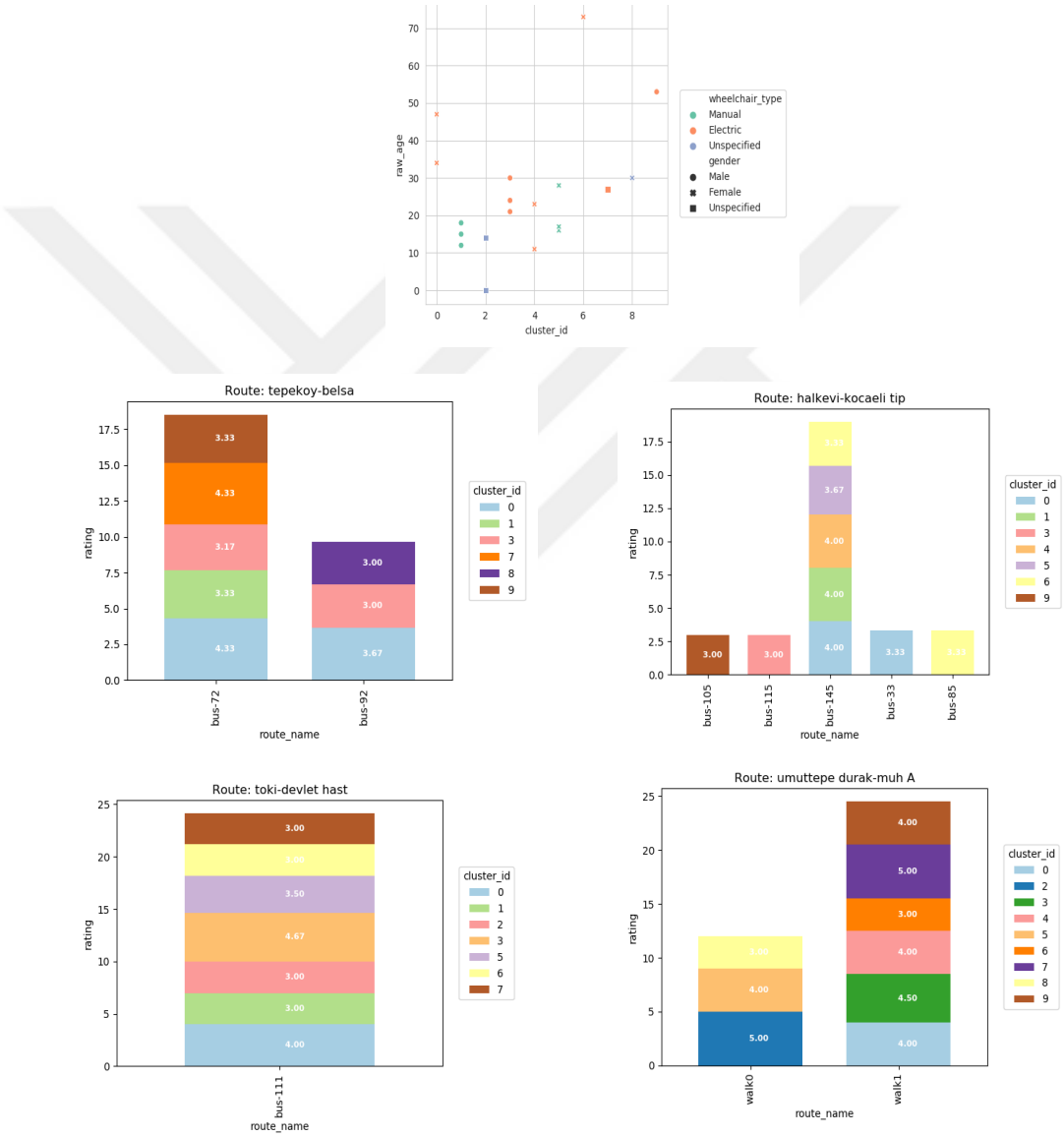


Figure 5.8. Route rating per cluster and corresponding clusters

We chose four combinations of origin and destination locations that were mostly frequented by our test subjects as shown in Table 4.2. The buses and walking routes returned after querying the test locations were then rated by each user depending on

their normal experiences and as many as they wanted. Some users differed from some routes by using different destination or origin locations. For example some users entered their route 2's destination as Kocaeli University Gate B instead of the faculty of medicine. And others set the start location of route 4 at Kocaeli University hospital instead of the bus stops at Umuttepe. Nevertheless, the average rating of each route as per the initial test locations that each bus got per cluster can be seen in Figure 5.8.

Since the cluster numbers changed after every cluster iteration, the cluster graph is shown beside the route score chart to communicate the demographic of users in the clusters represented in the rate score graph. The graphs show the average points of buses for each route as rated by users in the different clusters. We immediately understand that there were more bus options when a user searches from Halkevi to the university. It is also possible to deduce the bus numbers that had good ratings across different groups. For example, buses 72, 145, and 111 are all good options for at least five groups of people. For the route with no bus options, it is possible to deduce that there were at least two options which cater to the needs of most groups. Furthermore, we can predict the buses that would be recommended to a user based on their profile, cluster, and the rating of the buses for a particular origin and destination location.

To confirm that routes were recommended as expected, we used the 63 year old male with a manual chair. He made requests of each route as defined in the origin and destination locations in our previously selected test routes. Since we know that the cluster he was assigned to was cluster 9 as in Figure 5.8, we expected routes 1, 2, and 4 to have at least one recommended route because cluster 9 appears in all the routes charts with an acceptable rating. It should also be noted that the recommended routes depend on the schedule of the bus itself within the next 30 minutes of querying. The outcome of querying the test locations using the first test user are presented in Figure 5.9. The first and fourth screenshots display the recommended routes which are bus 72 and walking route respectively. The second screenshot shows that the query did not result in any recommended routes yet we expected bus 105 to be shown. However, this is explained by the fact that bus 105 had just gone past the stop in question meaning the bus was missed due to its schedule as confirmed by the screenshot from e-komobil, the official bus tracking app of the city.

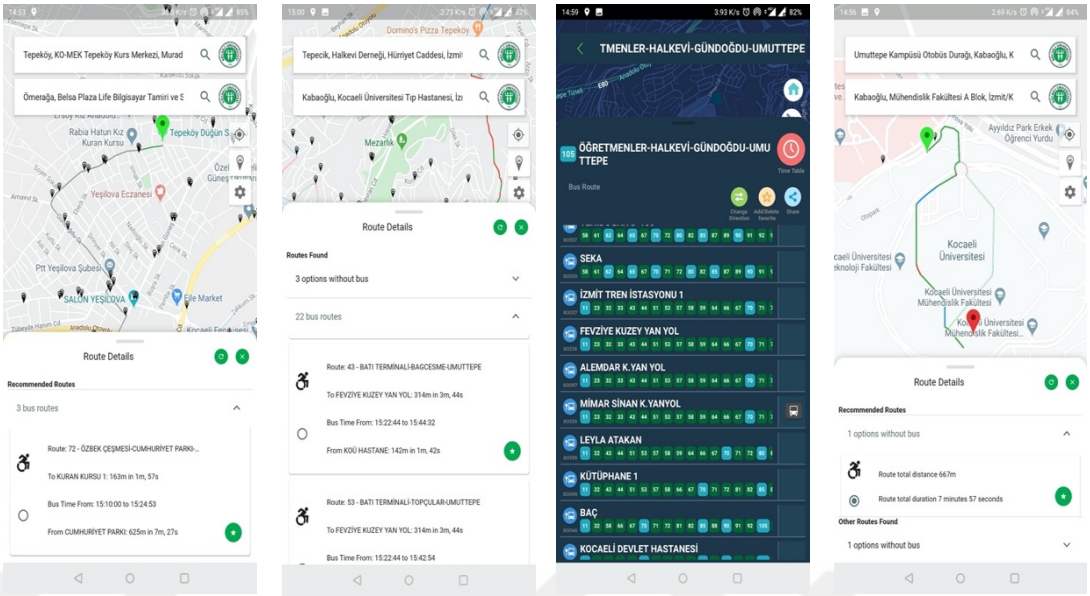


Figure 5.9. Request results for 63 year old male with manual wheelchair

We performed the routing with all the five test user profiles and compared the results with the charts' outputs. We observed that the recommended routes were consistent to the expected results, similar to the described example. After these confirmations, the test users were signed up to the application which triggered a re-clustering. We then re-analyzed the rating of the predefined four routes with the new clusters. The output of the redrawn charts are as shown in Figure 5.10. The number of buses rated for each route, and the number of clusters that rated each bus did not change because no new rating was done during the analysis phase. However, since two of the new users had unique profiles, the number of clusters changed from 10 to 12 as explained in the clustering analysis section. The two new clusters added each had one of the new test users with the unique profile. These clusters are 7 and 9 according to the cluster output shown in Figure 5.10.

Having been assigned new clusters that were different from their previous assigned groups, the 63 and 25 year old male users were unable to get the recommendations they were presented before. While lack of recommendations from previous clusters can be considered a disadvantage, the update made to the clusters keeps the system up to date by learning new patterns within the users.

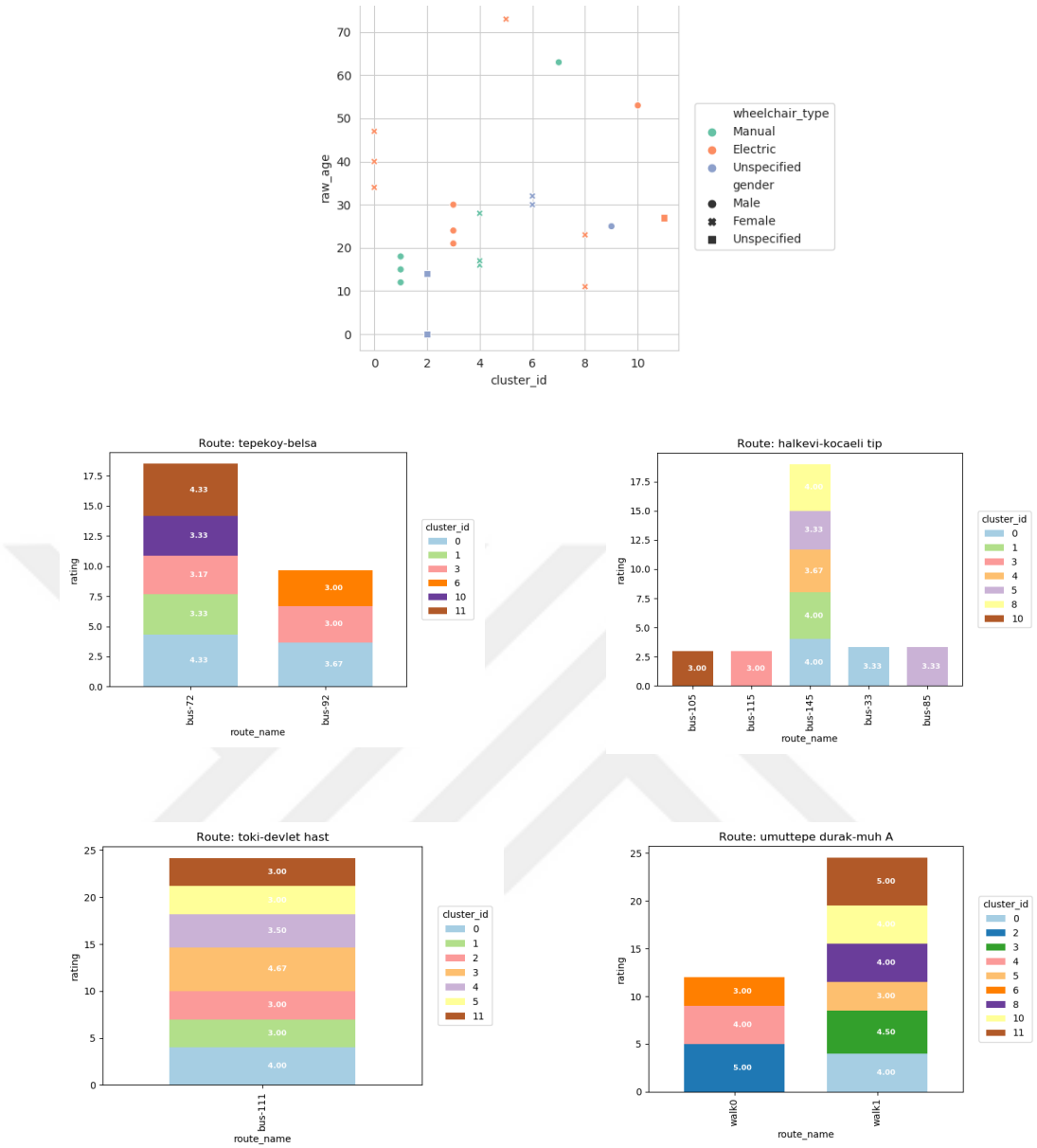


Figure 5.10. Route rating per cluster including the new test users

Furthermore, any rating submitted by users of different or unique demographics would translate to recommendations for future users who might share the profile in question.

5.3. Discussion

Results obtained and observed from the performance of the app were plausible for the cases we studied. Users with similar demographics were grouped in clusters of similar profiles, and the presented suggestions of routes were as expected. Our research majorly relies on the scores submitted by users of the app, making it a crowdsourced

app similar to previous studies as well as currently available applications in the device stores like wheelmap, route4u, and accessibility map (found in Google play store).

Researchers have tackled the issue of offering accessible routes to wheelchair users in many different forms. Some (Edinger et al., 2019; Gani et al., 2019; Iwasawa et al., 2016) used vibrations of surfaces to classify the accessible ways. Others (Hara et al., 2014; Prandi et al., 2017; Wu et al., 2019) used image processing to detect the ramps and obstacles. Others (Cáceres et al., 2020; Mobasheri et al., 2017) created apps that use crowdsourcing to collect information on accessibility issues in their cities. All the studies that we found which ventured into routing wheelchair users more or less delivered their routes on direct paths by considering ramps, crossings, slopes, surface types, and sidewalks. We tried to incorporate bus routes using the standard GTFS system of public transportation.

The city of Izmit is situated in a hilly geographic location. Most of its areas have slopes and hills that sometimes prove difficult for pedestrians, let alone wheelchair users. The bus routes that run through the city cover most of the residential areas in the city. However, there exist complexes that are not accessible by public transport. The idea behind our research was to provide wheelchair users with a platform to facilitate their making informative decisions on the routes they chose to travel.

The visualization of routes on a map is imperative for routing applications. Most researchers (Kozievitch et al., 2017; Mobasheri et al., 2017; Neis, 2015) use the OSM map and its data because it is free and open source. However, data is not always readily available for all cities (Neis, 2015). An exploration of the sidewalk, ramps, directions, and elevation data proved difficult for the town of Izmit. Therefore, we decided to use the Google Maps APIs and create our own map of the sidewalks and ramps in a designated area. It should also be noted that the free tier provided by Google APIs on a new account ran out before the end of the study. Some solutions to counter and reduce the number of requests made to the APIs would be to either use a free service or create a cache database for the API requests early on. Elevation data or maps can also be downloaded and set up on self-hosted databases to further reduce the cost incurred from paid services.

Researchers who have worked on providing accessible routes for wheelchair users employed many factors and variables. Most studies applied a score for directions using different variables. For example, (Gani et al., 2019) used the length and a pre-set score for the detected surface type, and Neis (2015) defined a reliability factor that was generated from values of slope, width, surface, smoothness, curb length, and lighting. We implemented a score for each route based on the average rating submitted by users in similar clusters.

The clustering of users via K-Means helps get users' recommendations of routes without relying on their rating. The constant update of clusters, in extension recommendations, also provides more intelligent information that enables more informed decisions on the part of the users. Overall, the users of the system would be collaborating and sharing information about their travel experiences smartly. Hence attaining the goal of a smart city application.

6. CONCLUSION AND RECOMMENDATIONS

Our study's main aim was to create a smart city application that would recommend either direct or bus routes to wheelchair users residing in a smart city. We attempted to provide an application with informative paths and intelligent recommendations for Izmit's mobility-impaired residents. We used the K-Means clustering method to get similar users based on their age, gender, and wheelchair type. Intelligent recommendations of routes were then generated from the average score of routes from users in the same group.

Based on the recommended routes presented to the users of both unique and shared demographics, it can be concluded that the proposal of routes from average cluster scores provide meaningful suggestions. The study shows that automated clustering functionality keeps the model updated on the different groups of users, thereby offering suitable and renewed route recommendations. Moreover, creating a platform that enhances collaboration and sharing of route scores between wheelchair users in the city promotes our research in the fields of smart city applications.

Further advancement of the application to greater heights of intelligence can entail broadening the features utilized in clustering and supplementing the recommendation engine with more machine learning algorithms such as collaborative filtering. The application can be made more dynamic by providing users with options to set some of the variables as per their needs. For example, the minimum and maximum gradients a wheelchair can accommodate usually differ. Users can also be allowed to explicitly set the time of day a user queries for bus routes for pre-planned trips. Similarly, the radius of searching the nearest bus stops and the minimum score of recommended ways can be adjustable from the presets used in our research.

Additionally, the predicted buses and their times can use the real time location of buses in the query instead of using the static bus schedules stated in the GTFS data. By rating the accessibility of each bus route, bus stop, and path segments separately, the scores and sections can be used to create a graph that uses greedy algorithms to get other

varieties of recommendations. The shortest path with the highest rating from a combination of its sections can be explored. Bus routes with transfers to other public transport options such as tramways can also be weaved in.

While creating the maps of sidewalks, ramps, and crossings was labor-intensive, the ability to compute routes that went through features designed explicitly for mobility-impaired users can be considered safe. Further improvements are, however, needed to visualize the returned path from the Dijkstra algorithm correctly. The manual mapping of the accessibility features in future research can be done via crowdsourcing as past researchers have proved reliable. The reporting of obstacles and damages currently undertaken via a dedicated phone line may also be made more efficient through mobile apps, as has been done by (Cáceres et al., 2020).

Incorporating more features into the user profile like the minimum and maximum preferred gradients, types of public transport used or queried, weather conditions and time of day during the rating of a route, and surface types preferred can be investigated to give the intelligent engine a broader perspective for appropriate suggestions. Our study outputs proposals based on users with similar profiles, facilitating the output of tips even for new users and those who have not submitted any feedback. Another category of recommendations that can be examined for users who have offered scores is presenting well-rated routes by other users who rated similar routes as them. Different machine learning algorithms, such as collaborative filtering, can complement the clustering to give a more diverse kind of recommendation.

With the popularity of technology and smart cities being at their hype, further development and investigation into making informative applications for all citizens, especially those with special needs, is fundamental. Our study shows that it is possible to have apps that can help work around the problem of accessibility in our cities as the municipalities and governments improve their efforts to establish and repair accessibility issues in public sectors.

REFERENCES

Akiner M. E., Smart Cities Transformation in Turkey, *International Journal of Contemporary Architecture The New ARCH*, DOI: 10.14621/tna.20160302.

Al-Turjman F., Malekloo A., Smart parking in IoT-enabled cities: A survey, *Sustainable Cities and Society*, DOI: 10.1016/j.scs.2019.101608.

Albino V., Berardi U., Dangelico R. M., Smart Cities: Definitions, Dimensions, Performance, and Initiatives, *Journal of Urban Technology*, DOI: 10.1080/10630732.2014.942092.

Alkadhim S. A. S., IESE Cities in Motion Index 2016 Center for Globalization and Strategy, DOI: 10.13140/RG.2.2.12141.4912.

Arthur D., Vassilvitskii S., K-Means++: The Advantages of Careful Seeding, *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, DOI: 10.1145/1283383.1283494.

Ashwini A., How To Choose The Best Mobile Backend As A Service (MBaaS), https://medium.com/swlh/how-to-choose-the-best-mobile-backend-as-a-service-mbaas-5534e1fc33f4 (Accessed: 9 July 2020).

Berrone P., Ricart J. E., Duch A., Carrasco C., IESE Cities in Motion Index 2019, DOI: 10.15581/018.ST-509.

Bilgin G., Taslak Eylem Planı—2020-2023 Ulusal Akıllı Şehirler Stratejisi, https://www.akillisehirler.gov.tr/eylemplani/ (Accessed: 22 May 2020).

Bolten N., Caspi A., AccessMap Website Demonstration: Individualized, Accessible Pedestrian Trip Planning at Scale, *The 21st International ACM SIGACCESS Conference on Computers and Accessibility,* DOI: 10.1145/3308561.3354598.

Brown L., Bus accessibility – more than just a ramp, https://www.intelligenttransport.com/transport-articles/20454/bus-accessibility/ (Accessed: 1 June 2020).

Cáceres P., Cuesta C. E., Vela B., Cavero J. M., Sierra A., Smart data at play: Improving accessibility in the urban transport system, *Behaviour & Information Technology,* 2020, 39(6), 681-694.

Calderoni L., Distributed Smart City Services for Urban Ecosystems, DOI: 10.6092/unibo/amsdottorato/6858.

Cohen B., Blockchain Cities and the Smart Cities Wheel, https://medium.com/iomob/blockchain-cities-and-the-smart-cities-wheel-9f65c2f32c36, (Accessed: 5 May 2020).

Cohen B., What Exactly Is A Smart City? https://www.fastcompany.com/1680538/what-exactly-is-a-smart-city, (Accessed: 7 May 2020).

Edinger J., Hofmann A., Wachner A., Becker C., Raychoudhury V., Krupitzer C., WheelShare: Crowd-Sensed Surface Classification for Accessible Routing, *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops),* DOI: 10.1109/PERCOMW.2019.8730849.

Elkan C., Using the triangle inequality to accelerate kmeans, *Proceedings of the Twentieth International Conference on Machine Learning (ICML),* Washington, DC, USA, 21-24 August 2003.

Engie, District heating and cooling systems, https://www.engie.com/en/businesses/district-heating-cooling-systems (Accessed: 28 July 2020).

F Bromley R. D., Matthews D. L., Thomas C. J., City centre accessibility for wheelchair users: The consumer perspective and the planning implications, *Cities*, 2007, 24(3), 229–241.

Gani O., Raychoudhury V., Edinger J., Mokrenko V., Cao Z., Smart Surface Classification for Accessible Routing through Built Environment—A Crowd-sourced Approach, *BuildSys '19: Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation,* DOI: 10.1145/3360322.3360863.

Giffinger R., Fertner C., Kramar H., Kalasek R., Pichler-Milanovic N., Meijers E., Smart Cities - Ranking of European medium-sized cities, *Vienna University of Technology,* 2007.

Göçümlü B. Ç., Bakanlık "Erişilebilir Türkiye" için harekete geçti, https://www.aa.com.tr/tr/turkiye/bakanlik-erisilebilir-turkiye-icin-harekete-gecti-/1726013, (Accessed: 1 June 2020).

Güven H., Akıllı Şehirler Beyaz Bülteni, *TC Çevre ve Şehircilik Bakanlığı*, 2019.

Hara K., Le V., Sun J., Jacobs D., Froehlich J. E., (2013). Exploring Early Solutions for Automatically Identifying Inaccessible Sidewalks in the Physical World using Google Street View, *Human Computer Interaction Consortium (2013),* Pacific Grove, California, USA, 22-27 June 2013.

Hara K., Sun J., Moore R., Jacobs D., Froehlich J., Tohme: Detecting curb ramps in google street view using crowdsourcing, computer vision, and machine learning, *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology - UIST '14*, DOI: 10.1145/2642918.2647403.

Hossain A., 7 Reasons to Choose React Native for Mobile App Development, https://geekflare.com/react-native-for-mobile-app/, (Accessed: 9 July 2020).

Iwasawa Y., Yairi I., Matsuo Y., Combining Human Action Sensing of Wheelchair Users and Machine Learning for Autonomous Accessibility Data Collection, *IEICE Transactions on Information and Systems,* DOI: 10.1587/transinf.2015EDP7278.

Kozievitch N. P., Almeida L. D. A., Silva R. D., Minetto R., A Smarter Sidewalk-Based Route Planner for Wheelchair Users: An Approach with Open Data, *Smart Cities, Green Technologies, and Intelligent Transport Systems*, DOI: 10.1007/978-3-319-63712-9_11.

Li Y., Wu H., A Clustering Method Based on K-Means Algorithm, *Physics Procedia*, 2012, 25, 1104–1109.

Lytras M., Visvizi A. Who Uses Smart City Services and What to Make of It: Toward Interdisciplinary Smart Cities Research, *Sustainability*, 2018, 10(6), 1998.

Mahizhnan A., (1999). Smart cities: The Singapore case, *Cities*, 1999, 16(1), 13–18.

Mobasheri A., Deister J., Dieterich H., Wheelmap: The wheelchair accessibility crowdsourcing platform, *Open Geospatial Data, Software and Standards,* 2017, 2(1), 27.

Mora H., Gilart-Iglesias V., Pérez-del Hoyo R., Andújar-Montoya M., A Comprehensive System for Monitoring Urban Accessibility in Smart Cities, *Sensors*, 2017, 17(8), 1834.

Neis P., Measuring the Reliability of Wheelchair User Route Planning based on Volunteered Geographic Information, *Transactions in GIS*, 2015, 19(2), 188–201.

Netkow M., Ionic Article: Ionic vs Flutter, https://ionicframework.com//resources/articles/ionic-vs-flutter-comparison-guide, (Accessed: 9 July 2020).

Nevado Gil M. T., Carvalho L., Paiva I., Determining factors in becoming a sustainable smart city: An empirical study in Europe, *Economics & Sociology*, 2020 13(1), 24–39.

Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E., Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, 2011, 12, 2825--2830.

Pellicer S., Santa G., Bleda A. L., Maestre R., Jara A. J., Skarmeta A. G., A Global Perspective of Smart Cities: A Survey, *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, DOI: 10.1109/IMIS.2013.79.

Prandi C., Mirri S., Ferretti S., Salomoni P., On the Need of Trustworthy Sensing and Crowdsourcing for Urban Accessibility in Smart City, *ACM Transactions on Internet Technology,* 2017, 18(1), 1–21.

Rachmawati D., Gustin L., Analysis of Dijkstra's Algorithm and A* Algorithm in Shortest Path Problem, *Journal of Physics: Conference Series*, 2020, 1566, 012061.

Rahaman M. S., Mei Y., Hamilton M., Salim F. D., CAPRA: A contour-based accessible path routing algorithm*, Information Sciences,* 2017, 385(C), 157–173.

Sharma P., Why is scaling required in KNN and K-Means?, https://medium.com/analytics-vidhya/why-is-scaling-required-in-knn-and-k-means-8129e4d88ed7, (Accessed 20 June 2020).

Shukla D., Shivnani C., Shah D., Comparing Oracle Spatial and Postgres PostGIS, *International Journal of Computer Science & Communication*, 2016, 7(2), 95-100.

Tavares J., Barbosa J., Cardoso I., Costa C., Yamin A., Real R., Hefestos: An intelligent system applied to ubiquitous accessibility, *Universal Access in the Information Society,* 2016, 15(4), 589–607.

Trevino A., Introduction to K-means Clustering, https://blogs.oracle.com/datascience/introduction-to-k-means-clustering, (Accessed: 20 June 2020).

URL-1: https://www.kocaeli.bel.tr/tr/main/birimler/saglik-ve-sosyal-hizmetler-dairesi-baskanligi/76 , (Accessed: 4 June 2020).

URL-2: https://www.kocaeligazetesi.com.tr/haber/1563522/akulu-tekerlekli-sandalye-sarj-istasyonunu-bu-hale-getirdiler, (Accessed: 4 June 2020).

URL-3: https://www.kocaeli.bel.tr/tr/main/news/haberler/3/bu-taksi-her-engeli-asiyor/30362, (Accessed: 4 June 2020).

URL-4: https://www.isbak.istanbul/intelligent-transportation-systems/electronic-detection-system/, (Accessed: 22 May 2020).

URL-5: https://gtfs.org/, (Accessed: 11 July 2020).

URL-6: https://hub.beesmart.city/smart-city-indicators/, (Accessed: 10 May 2020).

URL-7: http://fef.kocaeli.edu.tr/, (Accessed: 4 June 2020).

URL-8: http://www.ozgurkocaeli.com.tr/izmitte-tekerlekli-sandalye-sarj-istasyonu-kuruldu-338908h.htm, (Accessed: 4 June 2020).

URL-9: https://www.guru99.com/node-js-vs-python.html, (Accessed: 20 July 2020).

URL-10: https://transitfeeds.com/p/kocaeli-buyuksehir-belediyesi, (Accessed: 27 July 2020).

URL-11: https://docs.pgrouting.org/2.0/en/src/common/doc/functions/node_network.html, (Accessed: 26 July 2020).

URL-12: https://docs.pgrouting.org/latest/en/pgRouting-concepts.html#getting-started, (Accessed: 26 July 2020).

URL-13: https://developers.google.com/transit/gtfs/reference, (Accessed: 11 July 2020).

URL-14: https://www.thesmartcityjournal.com/en/articles/1064-road-smart-city-strategy-turkey, (Accessed: 5 May 2020).

URL-15: https://wiki.openstreetmap.org/wiki/Routing/online_routers, (Accessed: 11 July 2020).

URL-16: https://landtransportguru.net/sbs-transit-man-a95-with-auto-ramp/, (Accessed: 1 June 2020).

URL-17: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans, (Accessed: 14 July 2020).

URL-18: http://www.libelium.com/smart_parking/, (Accessed: 28 July 2020).

URL-19: https://covid19.saglik.gov.tr/, (Accessed: 21 July 2020).

URL-20: https://blog.transloc.com/blog/what-is-gtfs-why-does-it-matter-public-transit, (Accessed: 7 May 2020).

URL-21: https://www.aa.com.tr/en/economy/turkey-attaches-great-importance-to-smart-cities/1626053, (Accessed: 5 May 2020).

URL-22: https://www.planmelbourne.vic.gov.au/__data/assets/pdf_file/0003/509736/Brochure-January-20-min-neighbourhood-2019.pdf, (Accessed: 28 July 2020).

URL-23: http://maps.google.com, (Accessed: 25 July 2020).

URL-24: https://www.uab.gov.tr/bakanlik-yayinlari, (Accessed: 23 May 2020).

URL-25: https://iotsummiteurasia.com/en/trends/legacy-of-the-smart-cities/, (Accessed: 10 May 2020).

Wu J., Hu W., Coelho J., Nitu P., Paul H. R., Madiraju P., Smith R. O., Ahamed S. I., Identifying Buildings with Ramp Entrances Using Convolutional Neural Networks. *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC),* 2019, 2, 74–79.

Yin C., Xiong Z., Chen H., Wang J., Cooper D., David B., A literature survey on smart cities, *Science China Information Sciences*, 2015, 58(10), 1–18.

Young M., OpenTripPlanner—Creating and querying your own multi-modal route planner.

Yuan C., Yang H., Research on K-Value Selection Method of K-Means Clustering Algorithm, *J — Multidisciplinary Scientific Journal*, 2019, 2(2), 226–235.

PUBLICATIONS AND WORKS

Farukh F., Duru N. (2019) Automatic Detection of Elderly on Touch Screen Interfaces Using Touch Gesture Measurements, *10th International Conference on Image Processing, Wavelet and Applications, IWW2019,* Kocaeli, Turkey, 18-20 October 2019.

BIOGRAPHY

Firdaws Farukh was born in 1989 in the town of Kisumu in Kenya. She completed her high school education in Kisumu Girls' High School in the year 2007. She then joined Gazi University for a one year Turkish language course in preparation for her university major. She went on to graduate third of her class from Selçuk University in 2013 which earned her a degree in Bsc. Computer Engineering.

Later that year she started her career in software development as an intern at Allied Technique Inc. in Nairobi, Kenya. She was promoted to team lead status in 2016 surpassing many of her colleagues. She then ventured into a UI/UX position at Finance in Motion GmbH in 2018 which lasted for 6 months. She took a break from her career to pursue a degree in Msc. Computer Engineering after securing a scholarship with the Turks Abroad and Related Communities (YTB) institution of Turkey. She joined the department of Computer Engineering in Kocaeli University in 2018 where she researched topics on accessibility and wrote a thesis on a smart city application that offered intelligent route suggestions for wheelchair users in Izmit.

In addition, she is still working as a part time software engineer at Megvel Cartons Ltd. where she joined in 2020 during her masters course of study.