AN APPLICATION THAT OFFERS AI-BASED RECOMMENDATIONS DURING ELECTRIC CAR CHARGING

by

Nilhan Süer

Engineering Project Report

Yeditepe University

Faculty of Engineering

Department of Computer Engineering

2024

AN APPLICATION THAT OFFERS AI-BASED RECOMMENDATIONS DURING ELECTRIC CAR CHARGING

APPROVED BY:

Assoc. Prof. Dr. Mert Özkaya (Supervisor)	. . . .
Prof. Dr. Şebnem Baydere	. . . .
Assist. Prof. Dr. Mustafa B. Mutluoğlu	. . . .

DATE OF APPROVAL: .../.../2024

ACKNOWLEDGEMENTS

First of all I would like to thank my advisor Associate Professor Mert Özkaya for his guidance and support throughout my project.

Also I would like to thank my parents for their support and encouragement throughout my education up to the present.

ABSTRACT

AN APPLICATION THAT OFFERS AI-BASED RECOMMENDATIONS DURING ELECTRIC CAR CHARGING

Electric car users often need to charge their cars on in different places and they spend a lot of time there during car charging. However, they sometimes can't find any activity to do there such as having a coffee, shopping, walking or fitness… And then they decide to change their charging location or they want to leave before charging process is completed. So, this application solves this problem. It offers various activity recommendations for electric car users according to their habits and demands while charging their cars. For example, if the user has children and always prefers the locations that have playgrounds, then the application will automatically search for the nearest available charging station that is suitable for the families having children and recommend that location where the previous users also have a good time with their families during charging. The recommendations are decided using AI, machine learning and decision support algorithms based on the opportunities in that location and preferences of both the current user and the previous users who have visited the related charging stations. In addition, the application will be user-friendly, so that the users can easily access the application whenever they need.

ÖZET

ELEKTRİKLİ ARAÇ ŞARJI SIRASINDA AI TABANLI ÖNERİLER SUNAN UYGULAMA

Elektrikli otomobil kullanıcıları çoğu zaman araçlarını farklı yerlerde şarj etmek zorunda kalıyor ve araç şarjı sırasında orada çok fazla zaman harcıyorlar. Ancak bazen orada kahve içmek, alışveriş yapmak, yürüyüş yapmak, spor yapmak gibi bir aktivite bulamıyorlar… Sonra şarj yerlerini değiştirmeye karar veriyorlar ya da şarj işlemi bitmeden ayrılmak istiyorlar. Yani bu uygulama bu sorunu çözüyor. Elektrikli otomobil kullanıcılarına, araçlarını şarj ederken alışkanlıklarına ve taleplerine göre çeşitli aktivite önerileri sunuyor. Örneğin kullanıcı çocukluysa ve her zaman oyun alanı olan lokasyonları tercih ediyorsa uygulama otomatik olarak çocuklu ailelere uygun en yakın şarj istasyonunu arayacak ve önceki kullanıcıların da keyifli vakit geçirebileceği lokasyonu önerecektir. şarj sırasında aileleriyle birlikte. Önerilere yapay zeka, makine öğrenimi ve karar destek algoritmaları kullanılarak, o lokasyondaki fırsatlar ve hem mevcut kullanıcının hem de ilgili şarj istasyonlarını ziyaret eden önceki kullanıcıların tercihleri dikkate alınarak karar veriliyor. Ayrıca uygulama kullanıcı dostu olacak, böylece kullanıcılar ihtiyaç duydukları anda uygulamaya kolaylıkla ulaşabilecekler.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . x
LIST OF SYMBOLS/ABBREVIATIONS . . . xi
1. INTRODUCTION . . . 1
  1.1. Electric Vehicle (EV) Charging . . . 1
    1.1.1. EV Charging Stations . . . 1
    1.1.2. EV Charging Applications . . . 1
    1.1.3. Recommendations Based on User Habits . . . 2
  1.2. Terms . . . 2
  1.3. Motivation . . . 3
    1.3.1. Accessibility . . . 4
    1.3.2. As a Basis for Other Applications . . . 4
  1.4. Scope and Limitations . . . 4
  1.5. Problem Definition . . . 4
  1.6. Requirements . . . 6
    1.6.1. Functional Requirements . . . 7
    1.6.2. Non- Functional Requirements . . . 9
    1.6.3. Software Requirements . . . 10
    1.6.4. Hardware Requirements . . . 12
2. BACKGROUND . . . 13
  2.1. Previous works . . . 13
    2.1.1. Electric Vehicles (EVs) . . . 13
    2.1.2. Machine Learning in Recommendation Systems . . . 13
    2.1.3. EV Recommendation Systems . . . 14
3. ANALYSIS . . . 16
  3.1. User Side of The Application . . . 16
    3.1.1. User Data . . . 16
  3.2. Collaborative Filtering in Recommendation Systems . . . 18
  3.3. Back End Side of The Application . . . 22
  3.4. Communication Between Front and Back End . . . 24
  3.5. Nearby Stations . . . 26
4. DESIGN AND IMPLEMENTATION . . . 30

4.1. API Calls 32
4.2. Generating Recommendations 36
5. TEST AND RESULTS 38
6. CONCLUSION 44
6.1. Future Work 44
Bibliography 45

LIST OF FIGURES

Figure 3.1. Class Diagram of User Side 17
Figure 3.2. Use Case Diagram of User Side 18
Figure 3.3. E-R Model for Database 19
Figure 3.4. Flow Chart of User Side 20
Figure 3.5. Collaborative Filtering Work Flow 21
Figure 3.6. Machine Learning Collaborative Filtering Algorithm 23
Figure 3.7. Use Case Diagram of Back End Side 24
Figure 3.8. Flow Chart of Back End Side 25
Figure 3.9. Algorithm For Communication Between Front and Back End 26
Figure 3.10. Algorithm For Displaying Nearby Stations 27
Figure 3.11. State Diagram of User Side 28
Figure 3.12. State Diagram of Back End Side 29
Figure 4.1. Firebase Database 30
Figure 4.2. Activity Pop Up Algorithm 31
Figure 4.3. Activity Selection Algorithm 32
Figure 4.4. REST API Architecture 33
Figure 4.5. User Interface1 34
Figure 4.6. User Interface2 35

Figure 4.7. Sequence Diagram 36

Figure 4.8. Recommendation Logic 37

Figure 5.1. Similarity Heat Map 39

Figure 5.2. Top Recommended Activities for The Current Preferences of User . . 40

Figure 5.3. Recommendation Output 40

Figure 5.4. Speed Test for Machine Learning Algorithm 41

Figure 5.5. Usability Test 41

Figure 5.6. Average Result of Usability Test 42

Figure 5.7. Response Time Between Front End and Back End 42

Figure 5.8. CPU Utilization 43

LIST OF TABLES

Table 5.1. Example Dataset 38

LIST OF SYMBOLS/ABBREVIATIONS

EV	Electric Vehicle
AI	Artificial Intelligence
UI	User Interface
ABI	Application Binary Interface
CLI	Command Line Interface

1. INTRODUCTION

1.1. Electric Vehicle (EV) Charging

With the popularity of sustainable transportation, the usage of electric vehicles (EVs) are increasing day by day. According to the [1] International Energy Agency's Global EV Outlook 2023, EV sales have been growing exponentially, contributing to both economic savings for users and significant reductions in greenhouse gas emissions. Car users begin to prefer EV selection to make their budget more economic and contribute to the sustainability trend which aims a more green world with the decreased usage of harmful ingredients like gasoline, diesel engine or LPG on vehicles.

1.1.1. EV Charging Stations

Although, EV cars offer lots of advantages on user life and sustainability of the world; there are some issues and challenges that EV car users face with. One significant problem is *the charging process* due to the requirement of long period of time. Users have to wait long hours at the charging stations and sometimes they can not find anything to do because of the lack of conveniences at some *EV charging stations*. So, this problem makes users; especially the families with children or users that have limited time; to give up charging and look for another station option. This issue makes them waste time more and also lead the preference rate of that station brand decrease.

1.1.2. EV Charging Applications

The most of the *EV charging applications* on the market, does not find a solution to the problem of using time efficiently. They only displays the location of the charging stations belong to their brand and they do not show the location information of other brands. And the biggest shortcoming is that they do not have a system for providing recommendations based on user behaviour. [2] In the article written by Lanyun, Rebeccaa and Tracy, it is emphasized that the user-oriented feature of mobile application development in the field of EV charging. According to this article, many EV charging applications today do not provide user-specific services. So that, the electric car users face with several challenges during charging due to lack of user-centered structure.

1.1.3. Recommendations Based on User Habits

By *keeping track of the user's habits*, the application database for providing recommendations will be updated in real time to improve the user experience. The station information that user prefers will also be stored in the database to display the charging history. In addition, the user can add the desired stations as favorite to access them more easily. These detailed information collection makes the application give more accurate and current recommendations.

The mobile application will get help from *artificial intelligence (AI)* tools, *machine learning (ML)* algorithms and *decision support systems* to provide personalized activity recommendations and some other features to EV car users during charging. [3] The article ”Data-driven smart charging for heterogeneous electric vehicle fleets” explains the importance of machine learning algorithms in the field of EV charging. This article highlights the fact that efficiency and user experience will increase much more by integrating AI and machine learning technologies into EV charging systems. So that, I decided to use the benefits of AI and machine learning algorithms while creating recommendations on my mobile application. These recommendations will be created with respect to the users' preferences on station locations, nearby charging locations and user needs. For example, a user prefers taking coffee and going to a restaurant during charging; another user also prefers taking coffee, going to a restaurant and additionally, she/he prefers going to a library. Then, in the next use of the application by the first user; the library option will be recommended. And also, the nearby stations that can be preferred by that user will be displayed to that user.

1.2. Terms

- *Artificial Intelligence (AI)* the technology and science to make the machines think and behave like humans.
- *Machine Learning (ML)* is a sub field of artificial intelligence, which is the thing that the capability of a machine to imitate human behavior.
- *Decision Support Systems* is the technology that have the capability of making decisions based on the human's behaviour, system constraints or some rules.

1.3. Motivation

The rapid and significant rise of the electric car industry continues to meet our expectations for a sustainable and environmentally friendly world in the future. Although electric vehicles, which are increasingly coming to the fore in our environment, have many positive aspects, an important problem arises; charging problem. Range and charging times vary depending on the type and brand of electric vehicles. And brands are constantly making new developments in this regard. But despite this, the charging problem is still seen as a bigger problem. Users complain about long charging times and the fact that the applications of charging brands are not user-oriented. According to a report by [4] Pareek, long wait times at charging stations and inefficient use of time are common complaints among EV users .

My priority in this project is to eliminate the passiveness of this experience by making the electric vehicle charging experience more dynamic and interactive, making it a user-oriented application. My goal is to create a mobile application that helps electric vehicle users find nearby charging stations and offers personalized recommendations based on users' tendencies, needs and surroundings.

With this mobile application, users will make the most of their charging time. While doing this, my application will get help from algorithms such as AI and machine learning. Thanks to the application, charging will not be a burden, but will be a tool for users to spend their time more efficiently and enjoyable. For example, for someone working remotely outside, station suggestions will be made with a cafe nearby, for families with children, station suggestions with a playground nearby will be made, and for those who prefer to exercise, station suggestions with walking opportunities will be made.

My goal in doing all this is to increase EV usage by making the electric car charging process more enjoyable and personalized, thereby contributing to users and the environment and promoting a more environmentally friendly form of transportation.

My project's strategy will be based on collecting various data from users in real time and using advanced algorithms to provide personalized recommendations to these users. By investigating the shortcomings of existing applications and solutions, I aim to create a user-focused EV experience and make charging time more enjoyable and efficient.

1.3.1. Accessibility

Electric vehicle owners who can not use time efficiently or can not find any activity during EV charging so, leaving the station unhappy may find an AI based activity recommendation system during EV charging useful.

1.3.2. As a Basis for Other Applications

Recommendation systems are created by machine learning and AI algorithms. These recommendation systems are used in various fields. Retrieved information can be used for any field that requires prediction based on the previous preferences such as music, book or film recommendations. All of these makes people do easy choices and provide a better user experience.

1.4. Scope and Limitations

- One of the common problems of machine learning algorithms is the possibility of missing user data, so that couldn't provide the accurate outputs.
- Machine Learning and AI technology include lots of algorithms based on their needs and requirements, therefore it is difficult to choose and implement the suitable algorithm for the current problem.
- In terms of getting the user preferences, cleaning the data obtained and creating the appropriate recommendations to the user is a challenge because of the increased number of stations and demands to the electric vehicles.
- Creating recommendations using collaborative filtering is problematic because of the large amount of data obtained from user.

1.5. Problem Definition

Especially today, when busy users need to use their time much more efficiently, we are far from platforms that enable them to use this time in the most efficient way. For example, there is no suggestion system for electric vehicle users about what activities they can do during charging time. This may cause users to waste time or leave the charging station unhappy. However, even 1-2 hours spent at the charging station can be significant for a user. The article "Enhancing User Experience in Electric Vehicle Charging Applications (EVCA)" [5] emphasizes the importance of improving the user experience at charging stations and the need for user-centered solutions to increase satisfaction and efficiency.

Electric vehicle charging stations are slightly different from conventional gasoline locations. For example, while traveling on long roads, we come across many gas stations. And these stations are often located near places like restaurants, rest areas or markets. And since the gas run takes such a short time, we don't pay much attention to what options are near this station. We just fill up with gas and leave. But electric vehicle charging is not like this. It can sometimes take hours to charge your vehicle. Therefore, what you do and what you can do during this period is of great importance. At this point, user experience is very important. While stopping and taking a break for a gas-powered vehicle requires a very short time; charging an electric vehicle requires a much wider range of activity possibilities. This difference is underscored by Aiman Albatayneh and Mohammad N. Assaf [6], which notes that the extended time required for EV charging necessitates a broader range of activity options.

Users with busy schedules, families with young children, electric vehicle users who go on long journeys, or apartment residents who do not have charging facilities in their buildings will of course face with the problem of what activity opportunities a station offers during charging. Long waits at charging stations, the user choosing a station that is not suitable for him can result in boredom; and all of these may reduce the use of electric vehicles. This reduction in EV usage could hinder the hope of a sustainable and greener environment, as noted in a study by Levinson [7] on the impact of charging infrastructure on EV adoption.

With the increased use of electric vehicles, bad charging experiences also came. An awareness of spending time beyond just plugging in and charging the vehicle has emerged. Finding the right and closest station has become of great importance for users. Fabianek [8] also asserts that user happiness and the general perception of EV charging infrastructure are highly influenced by the customer experience at EV charging stations, in their articles. Because users realized that finding the right station is directly proportional to using the time correctly.

EV users have great difficulty finding the right station. The reason for this is that there is no information about what activities you can do at the stations. While some stations only have parking areas and chargers, some also have extensive facilities such as restaurants and cafes. According to a study by Carlton and Sultana [9], the availability of amenities has a considerable impact on consumer preferences and happiness, highlighting the crucial role that amenities play in improving user satisfaction at EV charging stations.

In discussing the difficulties of existing EV charging applications; Wang, Peng, Jia and Zhao [10] point out that one major drawback is the lack of user-centered features and information about station facilities, which negatively impacts the user experience. Since the activities cannot be seen through existing mobile applications, users have difficulty while planning their travels or days. In this case, it reduces the preference rate of the charging station and the relevant mobile application.

Although many attempts have been made to solve the current problems related to electric vehicles, these solutions are generally inadequate. The Electric Power Research Institute (EPRI) [11] highlights that while there have been advancements in EV charging infrastructure, many solutions still fall short in meeting user expectations . And according to that article, lots of improvements are needed in these areas to reduce the dissatisfaction of charging process. Many of the existing applications do not offer special services to electric vehicle users and do not care about the needs or preferences of these users. Therefore, non-user-friendly platforms that are far from the user's interests emerge. In fact, the biggest source of this situation is that these platforms receive a limited amount of data from the user. This prevents that platform from providing personalized recommendations to the user. These applications do not change according to user behavior or preferences, but provide a standard service to every user. As a result, certain requirements are ignored in these kinds of mobile applications.

The issue of improving the EV charging experience has been noticed by many developers, and different strategies have been explored to address this issue. For example, a study published in ResearchGate (2023) discussed the implementation of a recommendation system based on different algorithms and Boltzmann machines to create a special experience for EV users and contribute to them finding suitable stations. At the same time, a research was conducted on the application of blockchain technology to increase the efficiency of EV charging systems, and the results of this research were published by IEEE in the "A Secure Blockchain-Enabled EV Charging System" article [12]. With this study, a significant increase in user satisfaction was observed.

1.6. Requirements

There are many requirements to develop a mobile application that makes AI-based recommendations to users during the charging process of EV vehicles. I divided these requirements into 4: functional, non-functional, software and hardware requirements.

1.6.1. Functional Requirements

- User Authentication
  - (i) Requirement: Users must be able to be authenticated by their email addresses and passwords through the application.
  - (ii) Description: The authentication process should be secured and easy.
  - (iii) Priority: High
- User Registration
  - (i) Requirement: Users must be able to sign up and log in to their accounts.
  - (ii) Description: The required information should be gathered by the users while signing up and they should be saved into database.
  - (iii) Priority: High
- User Profile
  - (i) Requirement: Users can display the profile section.
  - (ii) Description: Profile section should include the nearby stations, favorite stations and station history.
  - (iii) Priority: High
- EV Charging Station Location Search
  - (i) Requirement: The application must provide a map interface for users to search the nearby charging stations.
  - (ii) Description: Using Google Maps SDK, the application should display the charging stations which have the activity options based on the user preferences within a specific radius.
  - (iii) Priority: High
- Display EV Charging Stations
  - (i) Requirement: The EV charging stations that have the recommended activities must be displayed on the map.
  - (ii) Description: When user press the related button to display the EV charging stations near the user location, the stations must be displayed and marked by icons.
  - (iii) Priority: High

- Activity Recommendations
  (i) Requirement: Based on the user preferences and the similar users' preferences, activities must be recommended.
  (ii) Description: Using collaborative filtering algorithm ,which is a algorithm of the machine learning, should be used to create recommendations.
  (iii) Priority: High
- Charging Station Information
  (i) Requirement: The application should display the activity options for each charging station.
  (ii) Description: Charging station information consists of activity names.
  (iii) Priority: Medium
- User Activity Selection
  (i) Requirement: Users should be able to choose the activity they want.
  (ii) Description: There should be a section where users can mark their activities comfortably and easily.
  (iii) Priority: High
- AI and Machine Learning Section
  (i) Requirement: Recommendations should be generated by AI and machine learning algorithms.
  (ii) Description: A decision-making mechanism should be created using the data obtained from user choices and AI and machine learning algorithms.
  (iii) Priority: High
- Favorite Stations
  (i) Requirement: User should mark the favorite stations.
  (ii) Description: The favorite stations of that user should be marked and they should be displayed in another section if user wants.
  (iii) Priority: Low
- Charging History
  (i) Requirement: The charging history of the user can be displayed through the application.

(ii) Description: Every time the user starts the charging process, the charging station and the activities that the user done should be stored.

(iii) Priority: Low

1.6.2. Non- Functional Requirements

- Usability

  (i) Requirement: The user can use system easily.

  (ii) Description: The UI components should be easy to use and the directions should be understandable, user should easily access the desired features.

  (iii) Priority: High

- Security

  (i) Requirement: The user information must be secured.

  (ii) Description: The information belong to that user should be protected, especially the authentication information such as email and password should be secured.

  (iii) Priority: High

- Performance

  (i) Requirement: The application should response fast and minimized amount of time..

  (ii) Description: Optimized machine learning algorithms should be used in the application to perform more efficiently.

  (iii) Priority: Medium

- Compatibility

  (i) Requirement: The mobile application should be available in various Android versions and devices.

  (ii) Description: Compatibility of the application should be varied and should be prepared for new version updates.

  (iii) Priority: Low

1.6.3. Software Requirements

- Development Environment for the User Side
  - (i) Requirement: Kotlin programming language should be used with Android Studio IDE.
  - (ii) Description: Kotlin language is suitable to develop Android applications and Android Studio also provides an environment for developing project with its several features such as providing Android emulator to test the application.
  - (iii) Priority: High
- Development Environment for the Server Side
  - (i) Requirement: Python programming language should be used.
  - (ii) Description: AI and machine learning operations should be handled by the python programming language.
  - (iii) Priority: High
- Database Integration
  - (i) Requirement: A database should be used to store user data, recommendation data, user's favorite stations data and user's charging history.
  - (ii) Description: Firebase database should be used for real-time data storage and update process.
  - (iii) Priority: High
- User Authentication Infrastructure
  - (i) Requirement: A secure user authentication infrastructure should be used.
  - (ii) Description: Firebase authentication should be used to authenticate user and it should be integrated with the android studio project.
  - (iii) Priority: High
- Networking
  - (i) Requirement: The necessary libraries must be added to the project to enable connection between front end and back end.
  - (ii) Description: 'okhttp' and 'retrofit' libraries should be added to the Kotlin project to make request and get response from the back end project.
  - (iii) Priority: High

- AI and Machine Learning Algorithms
    - (i) Requirement: Integrate AI/ML algorithms for recommendation part.
    - (ii) Description: Collaborative filtering algorithm should be used to make recommendations to a user based on that user's preferences and the similar users' preferences.
    - (iii) Priority: High
- Data Analysis Frameworks
    - (i) Requirement: Integrate data analysis frameworks to analyze data gathered by the user preferences.
    - (ii) Description: Data analysis frameworks should be used to make the data clean and usable such as Pandas library.
    - (iii) Priority: High
- Map Services
    - (i) Requirement: Mapping and geolocation APIs should be used.
    - (ii) Description: Google Maps API should be used for location services and displaying maps.
    - (iii) Priority: High
- Back End Services (API)
    - (i) Requirement: Back end services must be implemented for handling data, AI logic and enabling connection between two project.
    - (ii) Description: Django should be used for server-side operations and to an API should be generated to get request and send response to the Kotlin project.
    - (iii) Priority: High
- UI Components
    - (i) Requirement: Suitable UI components must be used to provide users a user-friendly application.
    - (ii) Description: Android UI components such as linear, relative or constraint layouts should be used with respect to the related page needs.
    - (iii) Priority: High

1.6.4. Hardware Requirements

- Device Compatibility
  - (i) Requirement: The application should be compatible with various Android devices.
  - (ii) Description: The application should be tested on various types of Android devices in different screen sizes.
  - (iii) Priority: Medium
- Development and Testing for User Side
  - (i) Requirement: It should be used a suitable device for development and testing for the user side operations.
  - (ii) Description: Android Studio ,which provides emulator devices to test the application, should be used to carry out the development and debugging operations on the user side.
  - (iii) Priority: High
- Development and Testing for Server Side
  - (i) Requirement: It should be used a suitable device for development and testing for the server side operations.
  - (ii) Description: Visual Studio Code should be used to carry out the development, testing and debugging on the server side.
  - (iii) Priority: High
- Servers
  - (i) Requirement: Server is required to handle the requests and posts.
  - (ii) Description: Appropriate servers should be implemented to accept the request that is sent by the user side and to sent response to the user side.
  - (iii) Priority: High
- Network Connection
  - (i) Requirement: Stable network connections should be provided.
  - (ii) Description: Network connections are required to carry out the data transmission between two projects (server and user side).
  - (iii) Priority: High

2. BACKGROUND

While more and more applications are being created for electric vehicle (EV) charging, there is still a dearth of research focusing on recommendation systems for EV charging. By examining the potential of recommendation systems to enhance the EV charging experience through the identification of surrounding charging stations and customized recommendations, this project seeks to close this gap.

2.1. Previous works

Various studies have been conducted to increase the use of electric vehicles and user experience today and before. However, most of these studies have been lacking in some points and have not reached the idea of user-oriented applications.

2.1.1. Electric Vehicles (EVs)

The world is transforming for a sustainable and greener environment. One of the important and major steps of this transformation is the widespread use of electric vehicles. The increasing interest in electric vehicles cannot be ignored. Its popularity is increasing day by day due to the many benefits and convenience it provides. But besides its advantages, it also has difficulties. Fayez Alanazi [13] has a detailed paper on this subject. While EVs are seen as a sustainable transportation option, their widespread adoption addresses their shortcomings in issues such as the number of charging stations, battery performance, and range anxiety. Alanazi's research reveals the evolution of electric vehicles and their difficulties as well as their advantages. To overcome these challenges, solutions such as improving charging infrastructures, increasing battery power, and diversity of charging stations are offered. It is stated in the article that stakeholders such as public services, governments and private sectors should be involved in order for these solutions to be implemented and thus the difficulties posed by EVs can be prevented.

2.1.2. Machine Learning in Recommendation Systems

As electric vehicles become more widespread, providing appropriate charging infrastructures is of great importance. At this point, the article prepared by Abdelhameed Ibrahim [14] and his team focuses on a recommendation system to help EV users find suitable charging stations. This system aims to increase the usability of the charging station by providing personalized recommendations to users. It uses real-time data about the user's previous preferences and habits when creating these personalized recommendations. While creating the recommendation system, a machine learning algorithm, whose parameters are fine-tuned using a waterwheel plant optimization algorithm called Boltzmann machine (RBM) is used. The complex structure of the system is handled with the Boltzmann machine algorithm and the accuracy of the suggestions provided by the system is demonstrated. This recommendation system is also based on parameters such as charging speed and cost. Thus, the user is offered a more user-oriented system where dissatisfaction is minimized and efficiency in the charging process is maximized. The study conducted by Abdelhameed Ibrahim reveals how well this recommendation system matches the user's wishes.

Another recommendation system proposal using machine learning algorithms is discussed in S. Bhaskaran's [15] article. With this recommendation system design, the dataset is grouped using collaborative filtering and ranks these groups according to adjusted cosine similarity. When the collaborative filtering method and other advanced techniques are compared, it turns out that the collaborative filtering application performs much better and more accurate solutions. With this and similar machine learning algorithms, hesitations about making a choice are eliminated and personalized suggestions are offered to the user. Thus, the interaction between the user and the user interface is improved. In his article, S. Bhaskaran reveals the relationship of personalized recommendation systems with machine learning and argues that such recommendation systems increase efficiency and user interaction.

2.1.3. EV Recommendation Systems

As EVs become more and more popular, recommendation systems that are capable of providing useful information to users in order to assist them in selecting an EV will become more and more important. The EV models that best suit the needs of the user are recommended by EV recommendation systems utilizing machine learning and artificial intelligence techniques to analyze user preferences. The applications, difficulties, and most recent developments of EV recommendation systems are covered in this review of the literature. In the article written by João Ferreira [16], it is written about an information system that is developed specifically for electric vehicle (EV) users. With this system, it is aimed for users to obtain station information about charging stations, optimize energy costs and reduce the anxiety of running out of range. While providing all these, it is envisaged to use a suggestion system. It aims to minimize energy costs by using the driver's preferences and choices to recommend the most suitable stations to the user. With the system mentioned in this article, EV drivers are provided with real-time access to station information, and it is thought that charging costs will be optimized by recommending the most suitable stations to the user by monitoring the battery status.

In addition to the Abdelhameed Ibrahim’s article, another paper that aims to solve the problem of insufficient EV charging infrastructure in smart cities is discussed by Suanpang [17]. A new method for recommending EV charging stations using reinforcement learning algorithms is mentioned. This strategy aims to improve user experience, increase the adoption of electric vehicles and create more sustainable transportation networks in smart cities. Although optimized solutions are offered, some technical difficulties are encountered. For example, it becomes quite difficult to train a model consisting of many electric vehicles and charging stations. In this case, it shows the complexity of the scalability principle. Another problem is predicted to be increased charging competition in the future. Unpredictability of future charging demands and problems such as increased waiting times or charging errors mean that this competition will be a challenging process. Coordination is another challenge. In large-scale charging station companies, it becomes much more important to meet charging demands and provide better recommendations to users.

Another problem of electric vehicles, the long charging time requirement and the random distribution of charging stations, is also explained by Liu [18]. Liu’s article proposes a solution that works with reservation logic, which puts EVs in a certain order and prioritizes them according to their charging needs and remaining time. In this solution, the most suitable charging station is selected for the user’s electric vehicle, taking into account the charging time, and it is aimed to optimize the total journey time for these vehicles. This solution also uses the reservation system to estimate the density of charging stations. The reservation system prevents situations such as the user getting bored, giving up on the charging station, or leaving the station dissatisfied, and aims to increase the user experience.

3. ANALYSIS

The complete process of user side and back end side can be seen on Figure 3.12 and Figure 3.11

3.1. User Side of The Application

First, the EV owner should sign up and then login. When user enters these data, they should be saved into the database. And then, user encounters with the profile page which has the displaying nearby locations, favorite locations and station history options.

When the user selects the displaying nearby stations option, a map view is displayed and any location can be searched by the user. Also, the user can see the EV charging stations near that location. When user selects a station, the station information is provided and the recommended activities is displayed. If user prefers to charge her/his car, then it is necessary to indicate that the charging is done. So that, the data, belongs to the user preferences is saved to the csv file to use that data while creating personalized recommendations. And while the creating csv file or parsing data to get them in suitable formats, several methods are used to achieve the desired outputs. All the required functions and the classes are like in the Figure 3.1.

While the EV Owner realizes the process of charging such as searching location or marking the stations as favorite; all the data are saved to the related collections of database by the developer side as in the Figure 3.2.

3.1.1. User Data

Since new users do not have enough data, collecting detailed user data and personalizing this data is a difficult process at first. Therefore, it is very important to collect the correct data from the user and store it in the right place. Another difficulty, which is the user not being able to interact with the application correctly, i.e. not using the application correctly, can also cause problems in collecting user data. For all these reasons, the steps required to collect user data should be simplified and the user should be able to use the application in the most interactive way. These steps are shown in the Figure 3.4.

The EV user enters the user information such as email, password and then a validation

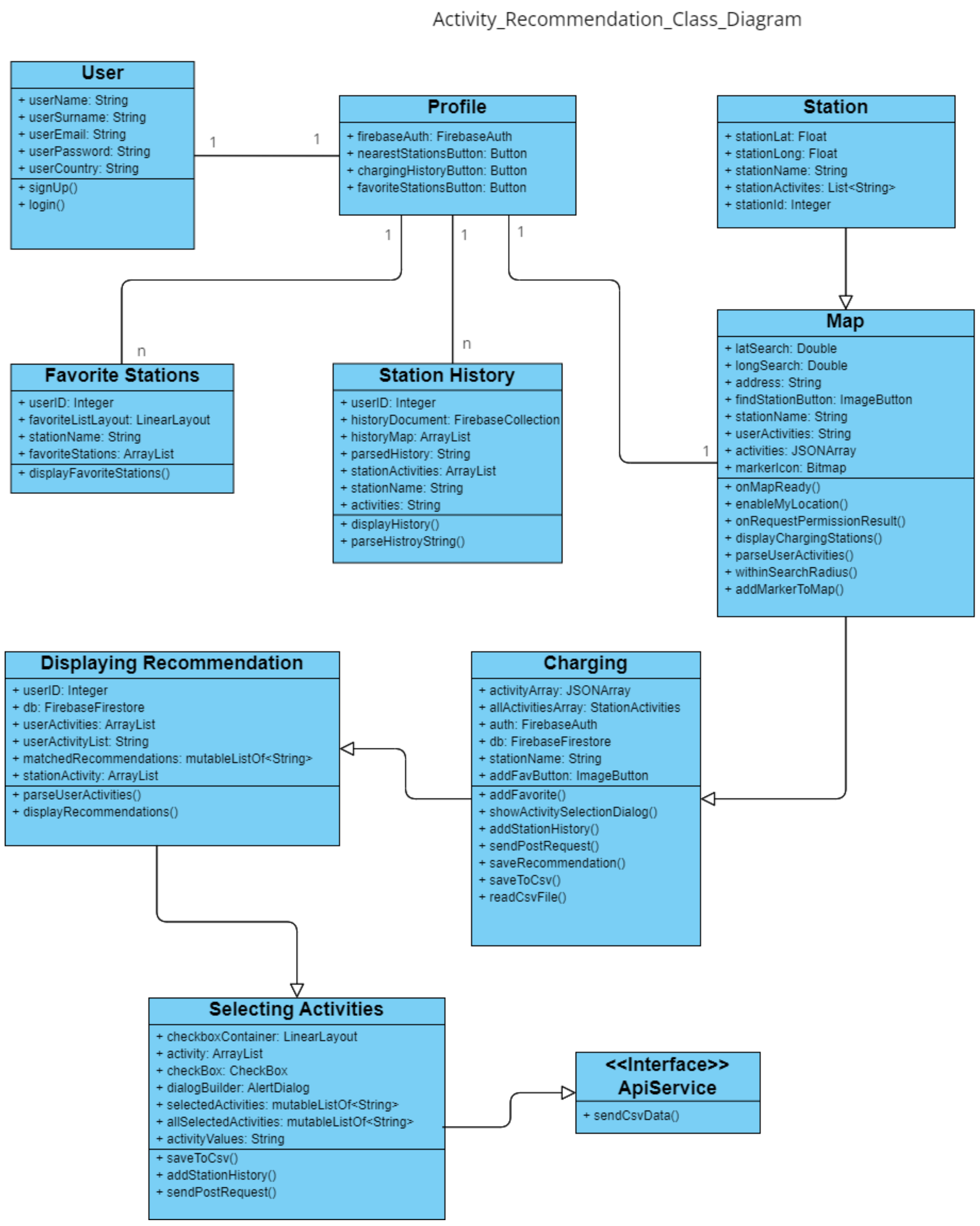


Figure 3.1. Class Diagram of User Side

mechanism works to ensure the format. After that, the user data is sent to the back end for storage securely. In the login part, the user is validated if she/he entered the true values and then these entered data are sent to the back end for authentication.

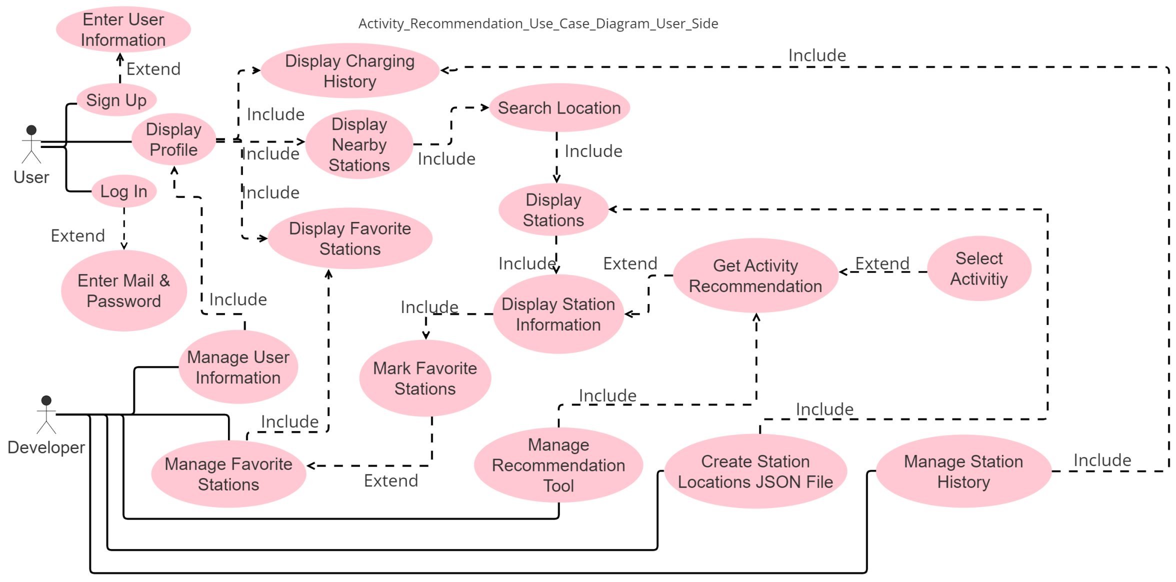


Figure 3.2. Use Case Diagram of User Side

In the activity selection part, EV user selects the activities that she/he has done during charging. These activities are sent to the back end to make recommendations in the future. And the station history belongs to that user is sent to the back end side and it is stored. The data store mechanism is handled like in the Figure 3.3.

3.2. Collaborative Filtering in Recommendation Systems

Customized and non-customized recommendation systems are the two varieties. Customized systems work better because they are made to meet the unique needs of each user, even when non-personalized systems could be simpler to use. Collaborative filtering is the one of the popular methods for developing personalized recommendation systems. It helps us to get the information from users who are similar to you and filter it to get the similarities between these users. It predicts user preferences.

User-to-user and item-to-item are the types of collaborative filtering algorithm. In the user-to-user collaborative filtering, it is considered that the user who likes an item tends to like a item that is liked by a similar user to that user. For example, if user A likes coffee, tea

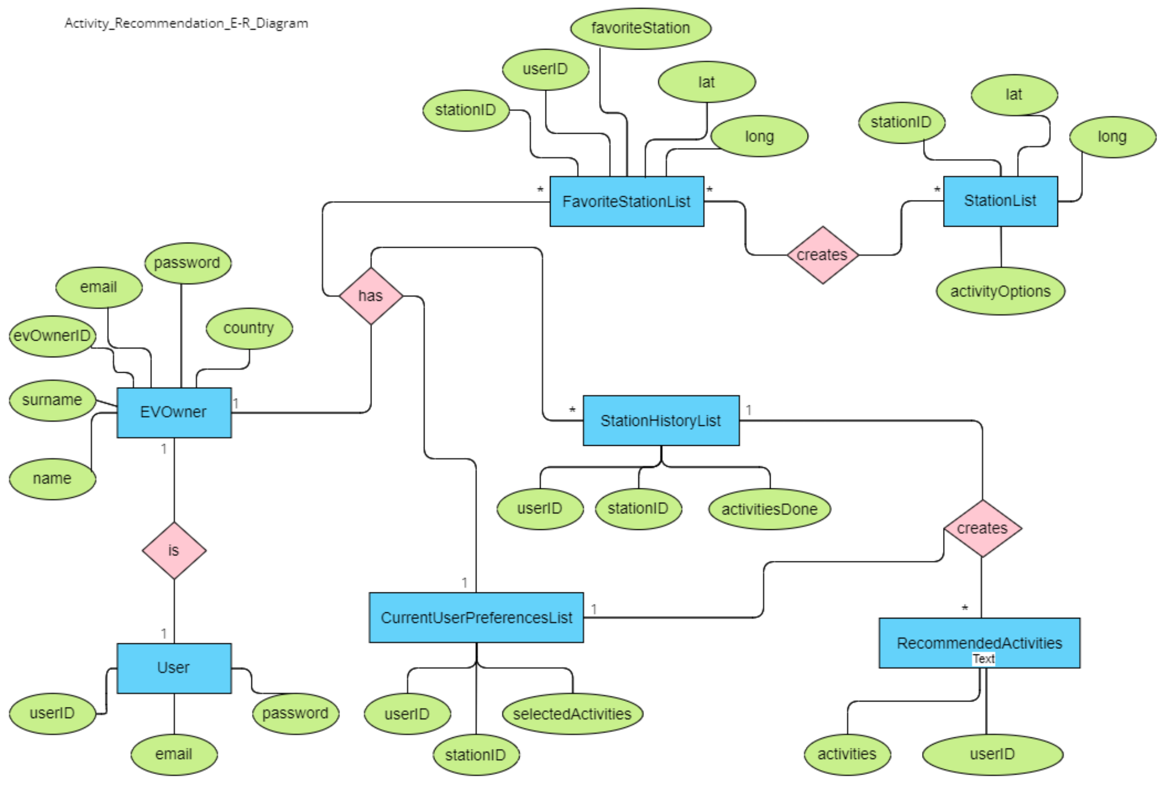


Figure 3.3. E-R Model for Database

and orange juice; the other user B also likes coffee, tea, orange juice and in addition to all of them user B likes lemonade. Then in the future, the lemonade option is recommended to the user A since the user A and B are the similar users and they tend to like similar items. On the other hand, in item-to-item collaborative filtering, the recommendations are generated based on the similar items, not similar users. In my project, the user-to-user collaborative filtering algorithm is more suitable when the aim of my application is considered. Because my project aims to get the user data and analyze that data if the users and their preferences are similar each other. As seen in the Figure 3.5 which describes the work flow of the collaborative filtering algorithm, the first step is to get the input data from user and turn the item-rating data into a utility matrix.

The rows of the matrix represent the users and the columns represent the item list:

$$utilityMatrix = \begin{pmatrix} userID & coffee & \cdots & restaurant \\ p7wmXiMfUFV8dygGL1keDhGoHxi2 & 1 & \cdots & null \\ \vdots & \vdots & \ddots & \vdots \\ 671XhngnqpU9fKkGKGSaSySRsG02 & 1 & \cdots & 1 \end{pmatrix} \tag{3.1}$$

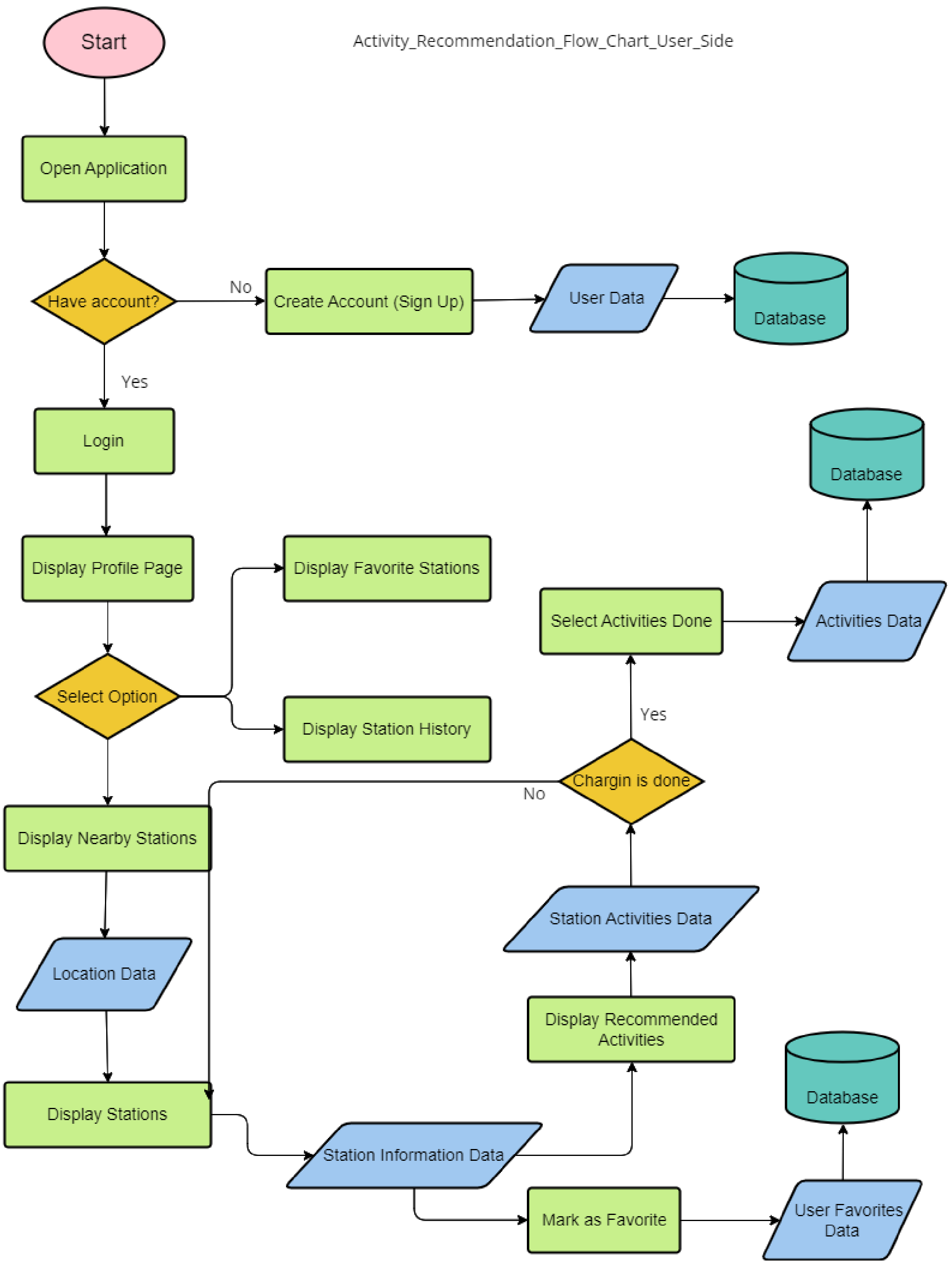


Figure 3.4. Flow Chart of User Side

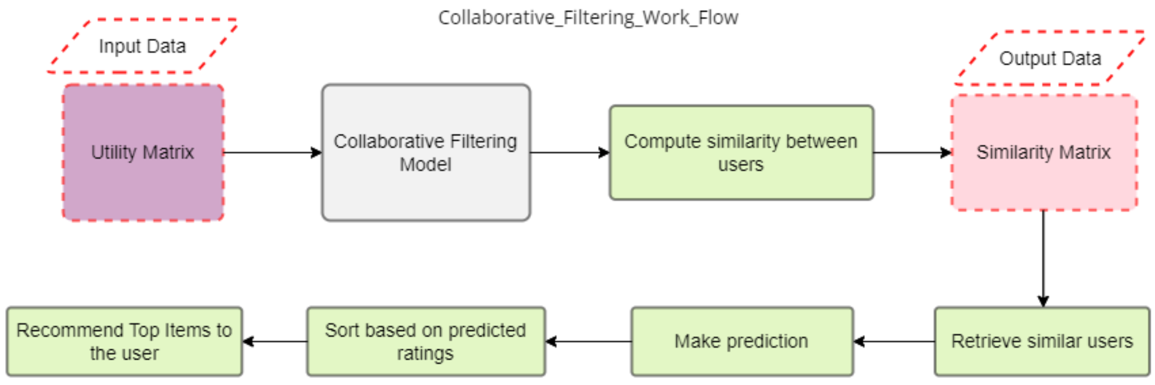

Collaborative_Filtering_Work_Flow
Input Data
Utility Matrix
Collaborative Filtering Model
Compute similarity between users
Output Data
Similarity Matrix
Retrieve similar users
Make prediction
Sort based on predicted ratings
Recommend Top Items to the user

Figure 3.5. Collaborative Filtering Work Flow

The '1's represents that the related activity is preferred by the user. On the other hand, 'null' value represents that the activity is not chosen by the user. Before moving to the next steps, the null values in the utility matrix are converted to '0' to make calculations more accurately. And then, this utility matrix is sent to Collaborative Filtering Model to generate the calculations related to the similarities which is calculated by a similarity function and to get the result. In similarity matrix, a matrix consists of user id, items and ratings is created:

$$similarityMatrix = \begin{pmatrix} userID & coffee & \cdots & restaurant \\ p7wmXiMfUFV8dygGL1keDhGoHxi2 & 1.9852 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 671XhngnqpU9fKkGKGSaSySRsG02 & 1.7329 & \cdots & 1.2754 \end{pmatrix} \tag{3.2}$$

Unlike the classic collaborative filtering algorithm which uses user's ratings for an item to make predictions, my algorithm uses binary data. Such that, if a user prefers an activity the section belongs to that user and activity selection is set as '1'. And if it is not selected, then it is set as 'null' value which will be converted into '0' to indicate that the related activity is not chosen by that user.

After the calculations are completed and we come up with a similarity matrix, the similar users are retrieved to make prediction. Then, we get rid of zero values, which mean that the relevant activity is definitely not recommended. After that, the highest activity ratings and their names are sorted, when the current userID 'p7wmXiMfUFV8dygGL1keDhGoHxi2' makes activity selections :

$$\left|\begin{array}{lcc} & activityName & rating \\ 1 & coffee & 1.7329 \\ 2 & restaurant & 1.2754 \\ 3 & fitness & 1.2135 \\ 4 & playground & 1.1945 \\ 5 & walking & 1.0234 \end{array}\right| \tag{3.3}$$

Finally, the top items are recommended to the user. For example, we should recommend coffee, restaurant, fitness, playground and walking activity options to the userID: ‘p7wmXiMfUFV8dygGL1keDhGoHxi2’ in the given order.

As a result, using the information that we get from similar users or items, we can develop recommendation systems and make predictions for the future needs thanks to collaborative filtering. Since the recommendations are so personalized, it can be a good effect on user experience and can increase the usage of these kind of applications.

3.3. Back End Side of The Application

The personalized recommendations that we provide to users are powered by the back end side of the EV charging recommendation system. The system includes several components and events as displayed in the Figure 3.7 which specifies the order of the events belong to the components. One of these important components is the gathering and keeping track of user information, charging history and favorite stations. Based on these data, the real-time information about recommendations are obtained through external APIs.

The data obtained from the user behaviour should be cleaned to remove inconsistencies, null values, duplicates or missing values in the dataset in the order shown in the Figure 3.6. Because missing values could distort similarity calculations, so it is very important. While standardization may be required, binary data is mostly in an appropriate format for calculating similarity. On the other hand, standardizing the data—that is, dividing by each column’s standard deviation and subtracting the mean—would guarantee an equitable comparison if the data were not binary.

One of the important points is the item similarity calculation which calculates the cosine similarity between activities:

$$similarity(A, B) = (A \cdot B)/(||A||||B||) \tag{3.4}$$

where A and B are the vectors.

After providing the cosine similarity equation, we have create a Data Frame where rows and columns represent activities and the similarity values. Then, we have to retrieve the similarity values for the given activities. And the CSV data that we got from the user side, should be converted into Data Frame to update the item similarity values.

```
Global Variables itemSimilarityDf, activityRatings, currentDataList
Function home
$selections \Leftarrow$ read CSV file
$new\ selections \Leftarrow fill\ the\ missing\ values\ with\ 0$
$item\ similarity \Leftarrow calculate\ cosine\ similarity\ of\ selections$
$item\ similarityDf \Leftarrow create\ Data\ Frame\ from\ itemSimilarity$

if $current\ data\ list\ is\ not\ empty$ then
  $similar\ activities \Leftarrow create\ a\ Data\ Frame$

  for $Each\ current\ data\ list$ do
    $similar\ rows \Leftarrow get\ similar\ activities\ based\ on\ activity\ preferences$
    $similar\ activities \Leftarrow add\ similarity\ row\ and\ transpose\ similar\ activities$
    $result \Leftarrow sum\ columns\ of\ similar\ activities\ sort\ in\ descending\ order$
    $top\ activities \Leftarrow get\ activity\ names\ from\ result$
  end for

  for $Each\ current\ data\ list$ do
    if $score > 0$ then
      $activity\ ratings \Leftarrow add\ pairs\ as\ activityName\ and\ score$
    end if
  end for
end if
$activity\ ratings \Leftarrow save\ CSV$
```

Figure 3.6. Machine Learning Collaborative Filtering Algorithm.

In the data processing and providing response part, the related values are calculated to get the recommendations based on the current data preferences using the updated similarity matrix. After that, the activity names and the recommendation rates should be send as response to the front end side as it is expressed in the Figure 3.8.

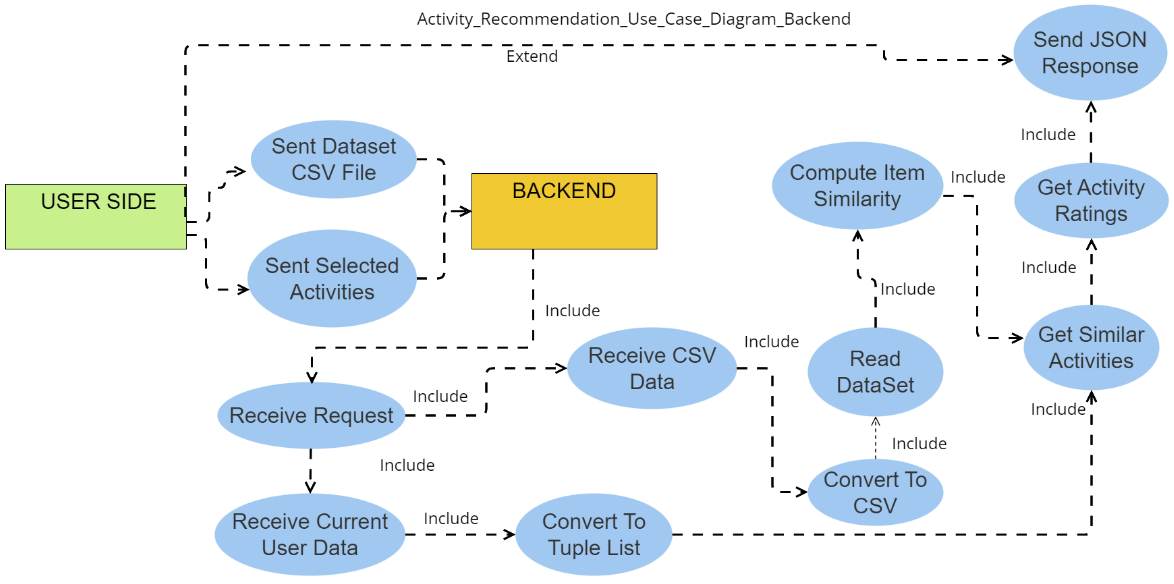


Figure 3.7. Use Case Diagram of Back End Side

3.4. Communication Between Front and Back End

The communication between front end and back end side is illustrated in the Figure 3.9. Firstly, the EV owner launches the application and create a profile whose data will be stored at the database. Then he/she is authenticated by their existing credentials. This authentication data is sent to the back end which checks the database if that user is valid. After that the user is shown a map to make the user select a charging station and then, the list of activities is displayed for the user to choose the activities they want to do. The selected activities are sent to the back end and stored at the database.

A POST request that consists of the newest version of CSV data, the user's current preferences and the user id is sent to the back end by the front end. The user's preferences are converted into a list of tuples in the back end side and CSV file is updated with the new one.

When the CSV data and user preferences are received, the recommendation process starts and follow the required steps. This process includes determining item similarities using the related methods and algorithms. And then, the recommendation engine searches for the similar activities preferred by the users. When the ratings belong to the similar activities are

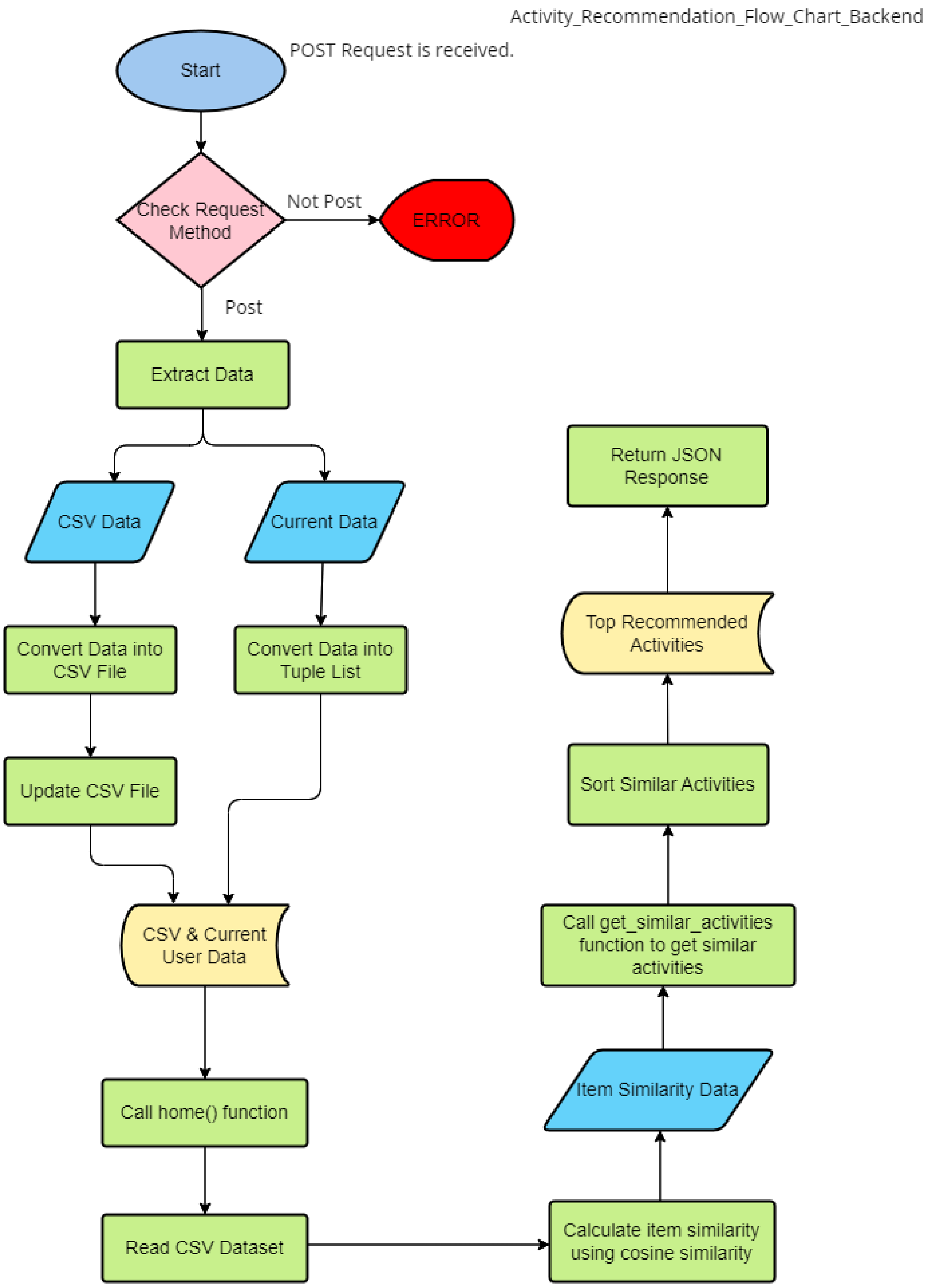

Activity_Recommendation_Flow_Chart_Backend
Start
POST Request is received.
Check Request Method
Not Post
ERROR
Post
Extract Data
CSV Data
Current Data
Convert Data into CSV File
Convert Data into Tuple List
Update CSV File
CSV & Current User Data
Call home() function
Read CSV Dataset
Calculate item similarity using cosine similarity
Item Similarity Data
Call get_similar_activities function to get similar activities
Sort Similar Activities
Top Recommended Activities
Return JSON Response

Figure 3.8. Flow Chart of Back End Side

calculated, the list of the recommended activities are sent to the front end side. In addition, the user can also mark the favorite stations in the front end side. These stations are sent to the back end and stored at the database. The front end side can request to get the favorite stations later and the user can display the preferred station history.

Back End Side Implement Recommendation Logic

- Use collaborative filtering to get the similar users and activities and make recommendations.

Back End Side Receiving Data

- Define URL patterns to match API endpoints and create a function to handle POST requests to an endpoint.
- Extract CSV data and current user data from the request comes from the user side and convert extracted data into suitable formats.
- Update the database with new csv data and current activity data.
- Call the recommendation engine to generate recommendations.
- Return the recommendations as a response.

User Side Make API Requests from Front End Project and Get Response

- Use related libraries to send POST requests to the API endpoints from the mobile application.
- Send necessary data (user preferences, CSV data) with the request body.
- When the response is received, parse it and display the recommendations to the user.

Figure 3.9. Algorithm For Communication Between Front and Back End.

3.5. Nearby Stations

For the EV owner, not showing all stations in the searched location is an important issue for increasing the user experience. In order to fulfill this requirement, we need to use a rule-based decision support mechanism as explained in 3.10. In this rule-based mechanism, the latitude and longitude information of the region searched by the user is retrieved. Then, a specific search radius is determined. Then, the charging stations within this radius range are shown to the user. In addition, the stations which include the recommended activity options, should be shown. In order to fulfill this requirement, the stations that are both close to the searched location and have certain activity options are filtered. So that, the user is saved from going to a station that is far away and prevented from going to a station that has activities that do not meet the user demands.

Handle User Location Request and Retrieve Searched Location

- Check if location permissions are approved.
- **IF** it is approved **THEN** enable the ”My Location” feature on the map.
- **IF** it is not approved **THEN** request location permissions.
- Get the address entered by the user and parse the address to obtain latitude and longitude.
- Add a marker at the Geo coded location and move the camera to the marker and zoom in.

Check Search Radius

- Calculate the distance between the search location and the station’s location.
- Return true if the distance is within the specified search radius, false otherwise.

Display Charging Stations

- Get the user’s ID and fetch the user’s preferred activities from the database.
- Load charging station data from a JSON file (or database).
- **FOR** each charging station
  - Get the station’s name, location (latitude, longitude), and activities.
  - Check if the station includes any of the user’s preferred activities and if the station is within the search radius of the user’s location.
  - **IF** both conditions are met **THEN** add a marker for the station to the map.

Figure 3.10. Algorithm For Displaying Nearby Stations.

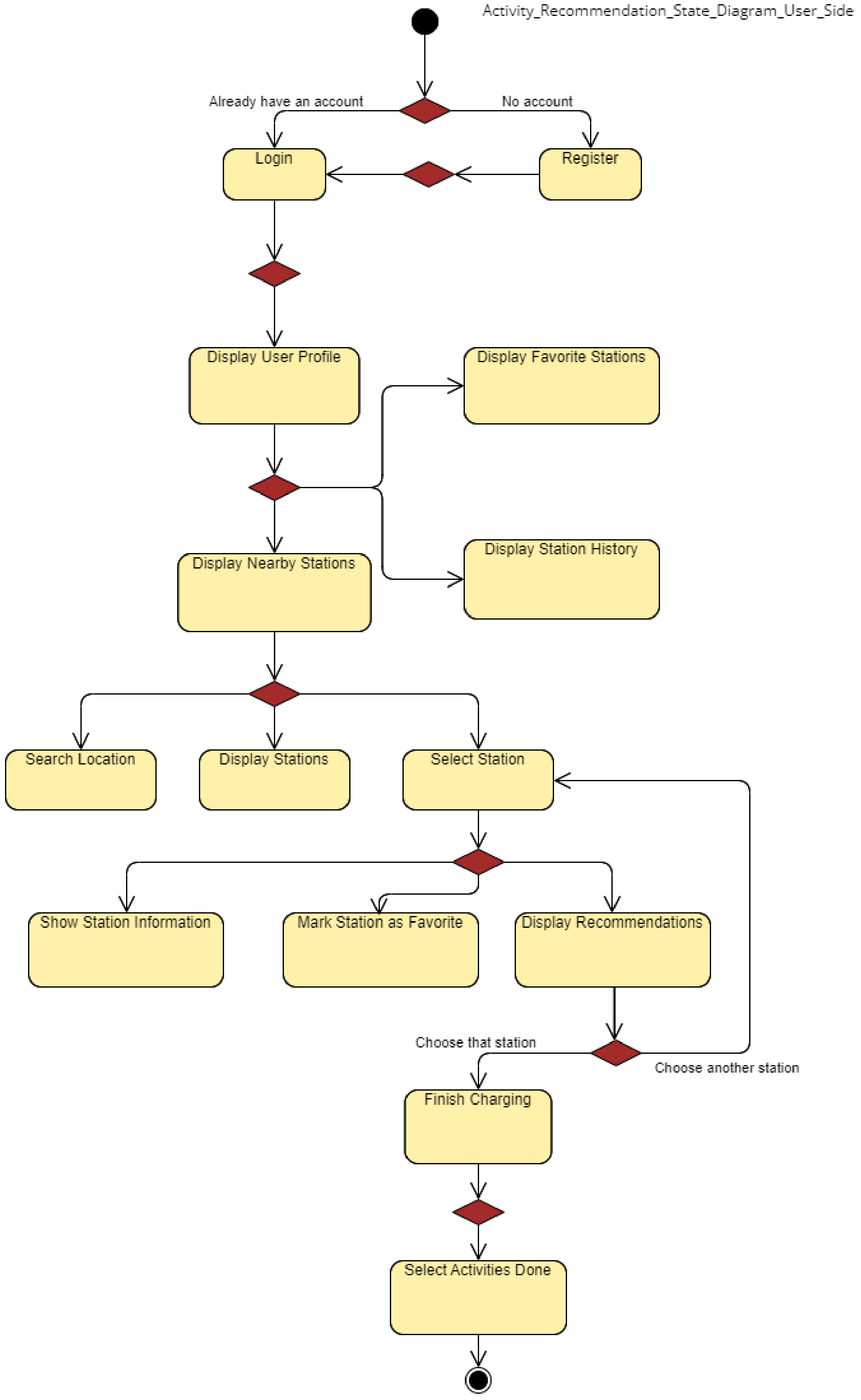
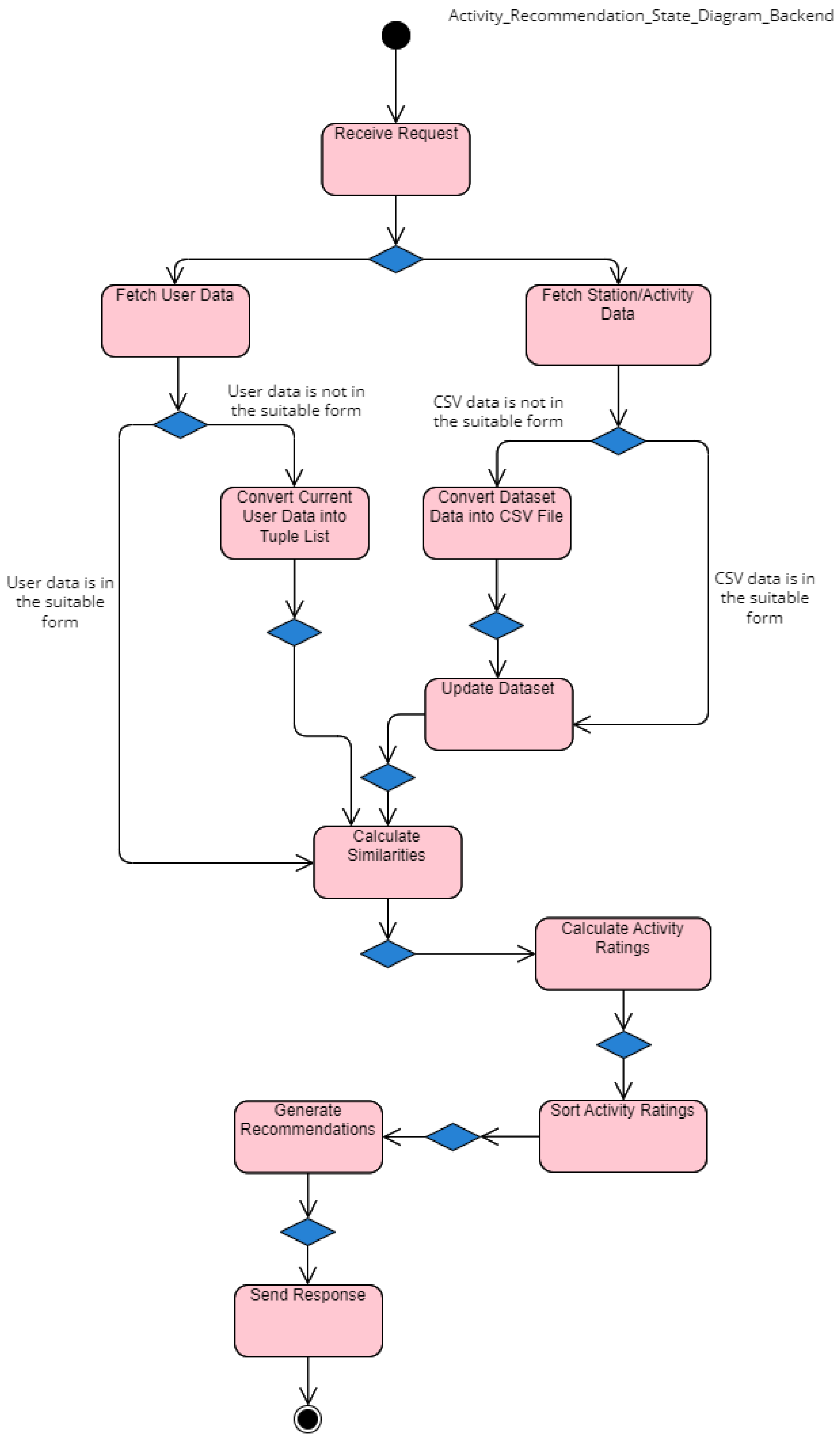


Figure 3.11. State Diagram of User Side



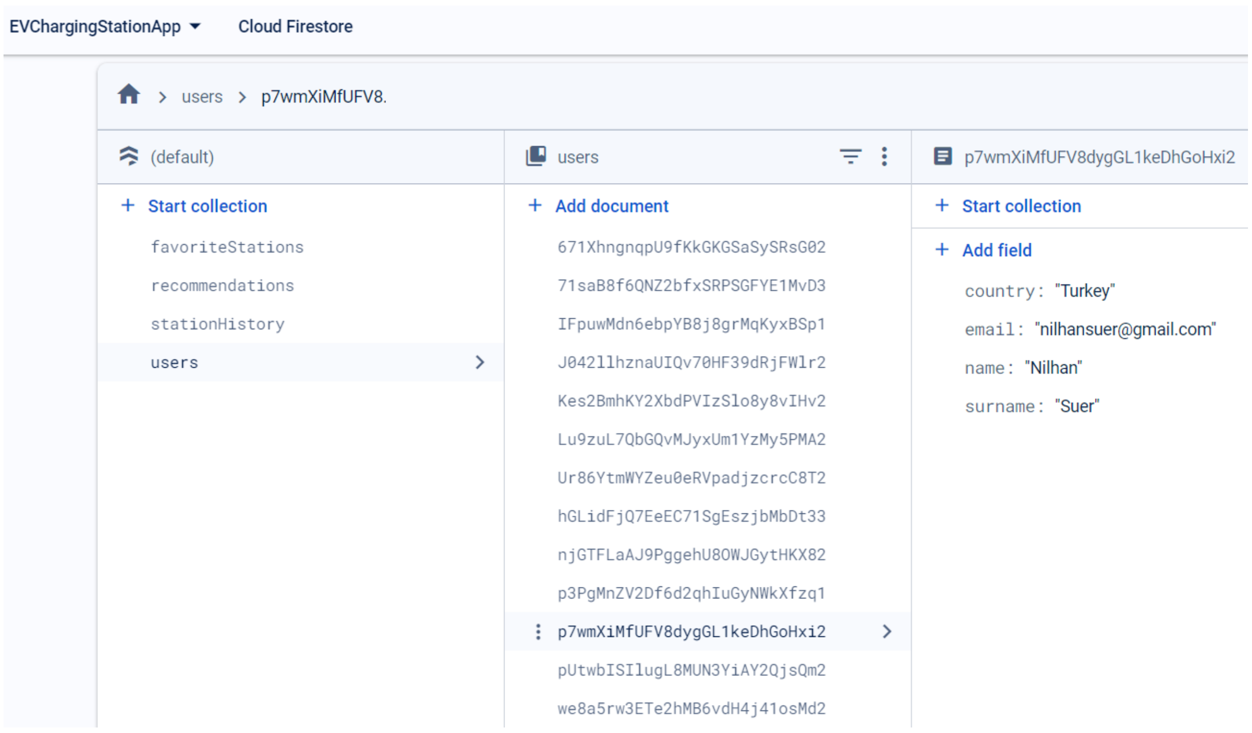
Activity_Recommendation_State_Diagram_Backend
Receive Request
Fetch User Data
Fetch Station/Activity Data
User data is not in the suitable form
CSV data is not in the suitable form
Convert Current User Data into Tuple List
Convert Dataset Data into CSV File
User data is in the suitable form
CSV data is in the suitable form
Update Dataset
Calculate Similarities
Calculate Activity Ratings
Sort Activity Ratings
Generate Recommendations
Send Response

Figure 3.12. State Diagram of Back End Side

4. DESIGN AND IMPLEMENTATION

The front end implementation is written in Kotlin for developing mobile applications for Android devices using Android Studio Development Environment. On the other hand, the back end side is written in Python using Visual Studio Code because of its ease of analysis and advanced libraries in areas such as artificial intelligence (AI) and machine learning. Kotlin was chosen for its improved features that is suitable for application development for Android devices, such as user interface and map integration. The order of components belong to the both front end and back end is displayed in the Figure 4.7.

As the external services I use Google Maps API to visualize map view. And I also get help from Firebase Authentication to provide the secure user authentication mechanism and I use Firebase Database to store user and station information.



```
favoriteStations
recommendations
stationHistory
users
```

```
671XhngnqpU9fKkGKGSaSySRsG02
71saB8f6QNZ2bfxSRPSGFYE1MvD3
IFpuwMdn6ebpYB8j8grMqKyxBSp1
J042llhznaUIQv70HF39dRjFWlr2
Kes2BmhKY2XbdPVIzSlo8y8vIHv2
Lu9zuL7QbGQvMJyxUm1YzMy5PMA2
Ur86YtmWYZeu0eRVpadjzcrcC8T2
hGLidFjQ7EeEC71SgEszjbMbDt33
njGTFLaAJ9PggehU8OWJGytHKX82
p3PgMnZV2Df6d2qhIuGyNWkXfzq1
p7wmXiMfUFV8dygGL1keDhGoHxi2
pUtwbISIlugL8MUN3YiAY2QjsQm2
we8a5rw3ETe2hMB6vdH4j41osMd2
```

```
country: "Turkey"
email: "nilhansuer@gmail.com"
name: "Nilhan"
surname: "Suer"
```

Figure 4.1. Firebase Database

In the front end, I use Android UI components such as Android Constraint or Relative Layouts. Firstly, I create the XML files under the layout folder. In the XML files, I use the related UI components like buttons, lists, cards, text views, edit texts and image buttons. After creating the XML files, I create the Kotlin Activities belong to that XML files. There are 8 Activity classes, 1 interface, 1 data class and 1 object class. The first activity called Login Activity class is created to handle the authentication process of user. If the user does not have an account, then the user should initially create an account and it is handled in the Sign up Activity class. The information entered in this class is stored at the Firebase Database in the ’users’ collection. As seen in the Figure 4.5, the user is directed to the profile page which has 5 options, after authentication.

When the user press ’Nearby Stations’, the user should be directed to the related activity class that has the map view. The user can type the desired location to the search area. And then when the ’Recommended Stations’ button is clicked,a rule-based algorithm works to decide if the station is in the specific search radius and includes the recommended activities. As soon as the user is directed to the Charging Activity class, the list of activity recommendations is displayed as a pop up as seen in the Figure 4.2. After that, the station activities which are stored at the JSON file, are listed on the station page. This JSON file includes station IDs, names, latitude, longitude and activity options. If the user decides to charge his/her EV car on that station, then it is necessary for him/her to press the ’Charging is Done’ button when the charging process finishes. After the EV owner completes the charging process, she/he have to

```
Function showPopup(activityArray)
    (i) Display the dialog box with the ”OK” button.
   (ii) Read the ’Stations.json’ file to retrieve the activity options belong to that station.
  (iii) Consult Firebase for user recommendations.
   (iv) Align suggested activities with station offerings.
    (v) The popup should show recommendations that match.
```

Figure 4.2. Activity Pop Up Algorithm.

press the ’Charging is done’ button. Then a dialog window meets the user and it is expected for the user to select the activities done. The activity selection process is achieved by the ’showActivitySelectionDialog()’ function as seen in the Figure 4.3. The selected activities are also added to the Firebase Database by ’addStationHistory()’ function to save the station history and related activities that have been done. After the adding data to the database part, the current data such as user activity preferences are saved to the ’preferencesDatabase.csv’ file and it is updated. Then the updated CSV file and the current data selections are sent to the back end side as post request and also they are saved to the ’recommendations’ collection of Firebase Database.

```
Function showActivitySelectionDialog()
    (i) Make a dialog box with checkboxes for every task that needs to be done at the
        station and by clicking the "OK" button:
            • Obtain the chosen activities, store them in a CSV file (preference-
              Dataset.csv).
            • Update the rows with the new selections.
            • Use a POST request to deliver current and updated CSV data to the back-
              end.
Function addStationHistory() and saveRecommendation()
    (i) Obtain the station name and current user ID and save the current activity selec-
        tions to the 'recommendations' collection of the Firebase Database.
    (ii) Add the new station and activities to the user's history in database.
Function sendPostRequest(csvData, currentData)
    (i) Send the back end a POST request that includes the updated CSV Data and the
        current activity selections of user.
```

Figure 4.3. Activity Selection Algorithm.

The application has also 'Charging History', 'Favorite Stations', 'About' and 'Settings' options on the Profile Page. It is as seen in the Figure 4.6. In the 'Charging History' page, the user's charging history is listed with respect to the previous preferences of her/him. The data is stored at the Firebase Database collection. And when the user wants to see the history, the data is retrieved from the database and displayed at the front end side. The EV owner can also mark a charging station as favorite. These favorite stations are stored at the Firebase Database in the 'Favorite Stations' collection with respect to the user uid.

4.1. API Calls

To send the post request and get the response, I use Kotlin Third-Party Libraries such as Retrofit and Okhttp for API calls. I create a client using OkHttpClient() method that belongs to the Kotlin's own library, and then send the data like CSV file and current user preferences through that client connection. And also, the responses are received by the client's connection to the specific back end URL which is created by Django.

In the back end side the post requests that comes from Kotlin project is handled by Django Rest Framework as seen in the Figure 4.4. Firstly, the API Rest Request comes from a client, the Kotlin project. The request types can be any of the HTTP methods such as GET or POST methods. The Kotlin project sends its request by using the static URLs and Routers which handles the API routes that is specified in urls.py file. So that, the Kotlin project can communicate with the correct view method in the Python side by the URL pattern and HTTP method. The URL dispatcher which is included in the urls.py file gets the request sent by Kotlin project and forwards the request to the API part whose infrastructure such as serializes is automatically provided by Django Rest Framework. API components sends the data to the Model through views.py file, which handles the request. In my project design, the model that includes the data structure for the recommendation logic is specified in the view.py file. The request data such as current user preferences and updated CSV file are retrieved to make it interact with the database. The expected response is prepared in view file and it returns a API response in JSON format which includes the personalized recommendation data.

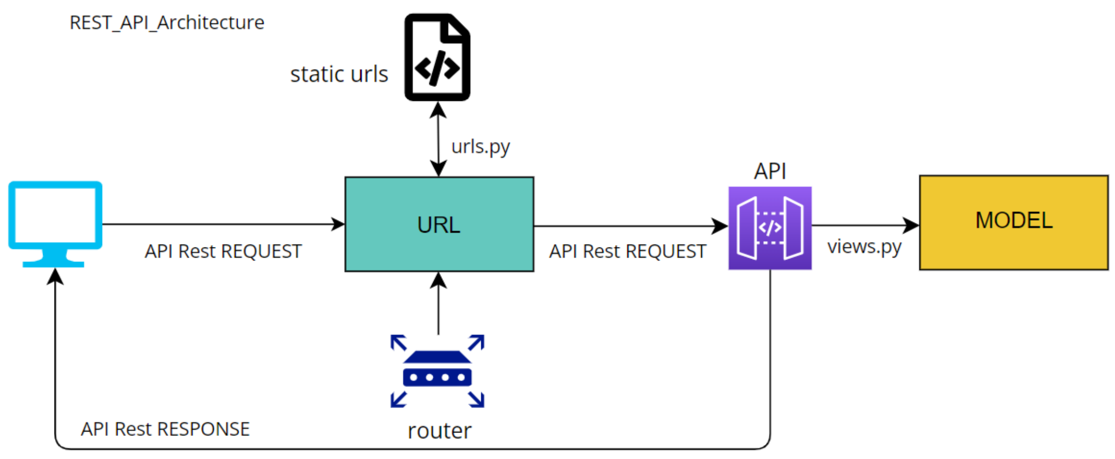


Figure 4.4. REST API Architecture

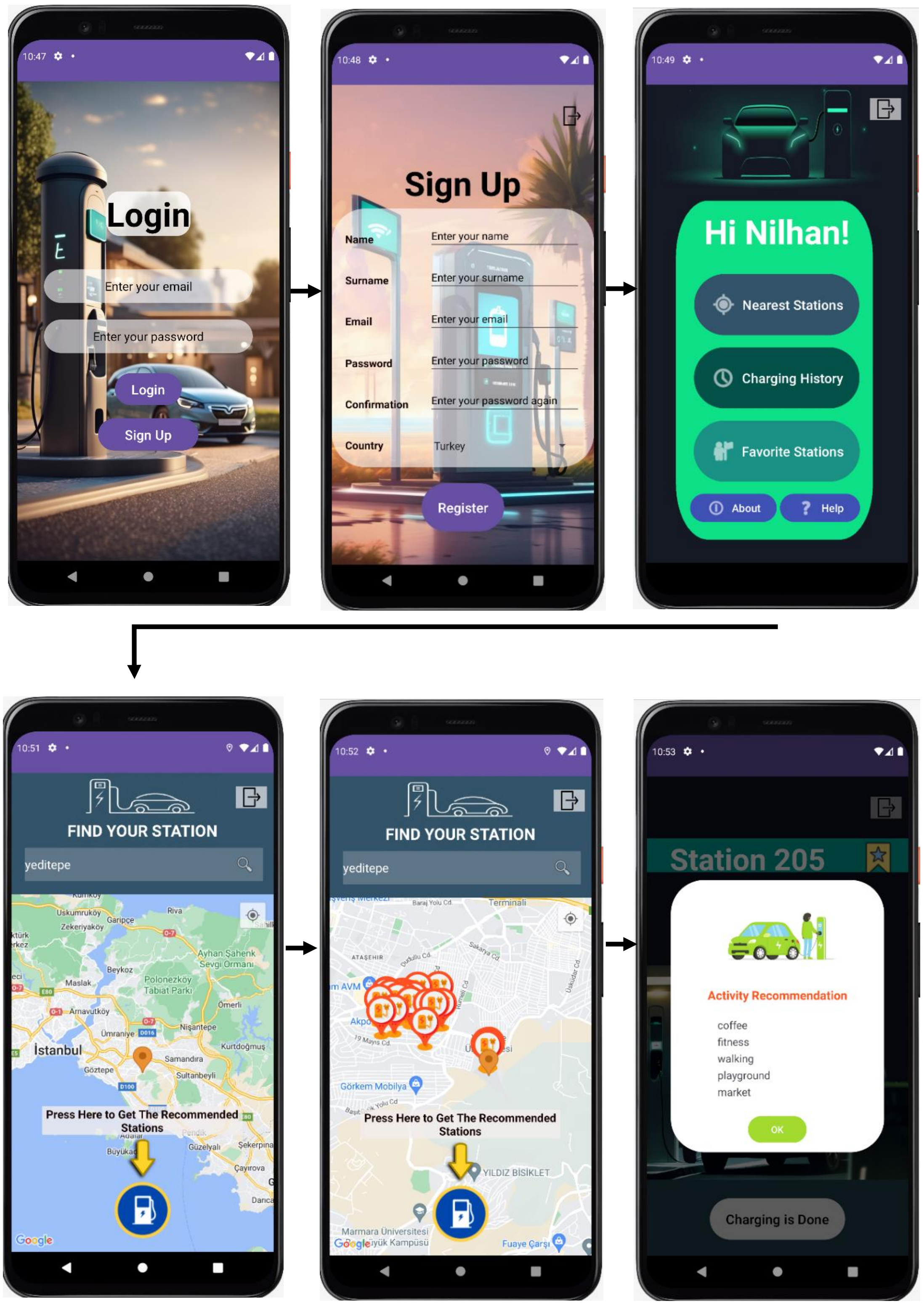
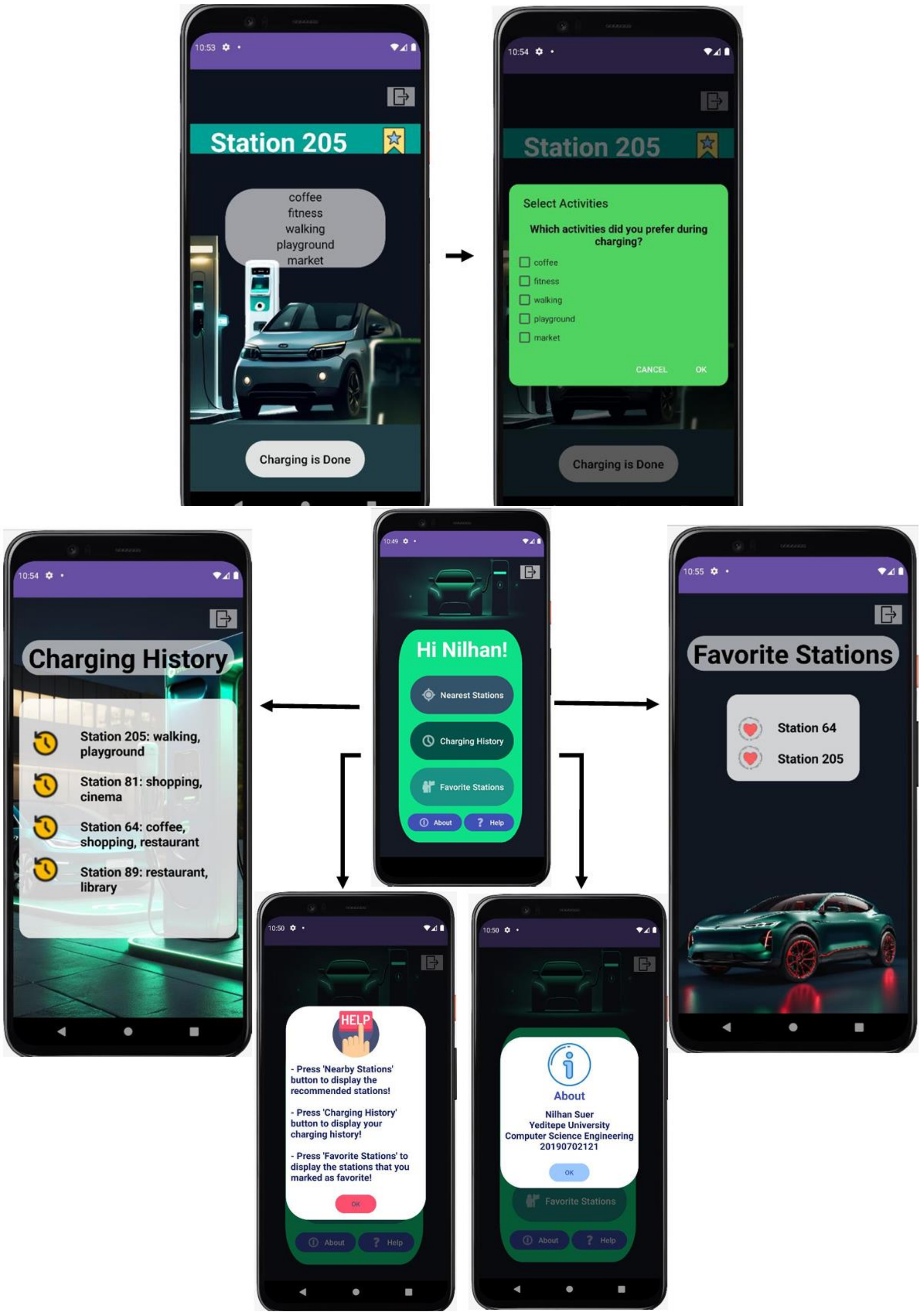


Figure 4.5. User Interface1



Select Activities

Which activities did you prefer during charging?

- [ ] coffee
- [ ] fitness
- [ ] walking
- [ ] playground
- [ ] market

CANCEL OK

Charging History

Station 205: walking, playground

Station 81: shopping, cinema

Station 64: coffee, shopping, restaurant

Station 89: restaurant, library

- Press 'Nearby Stations' button to display the recommended stations!

- Press 'Charging History' button to display your charging history!

- Press 'Favorite Stations' to display the stations that you marked as favorite!

About

Nilhan Suer
Yeditepe University
Computer Science Engineering
20190702121

OK

Figure 4.6. User Interface2

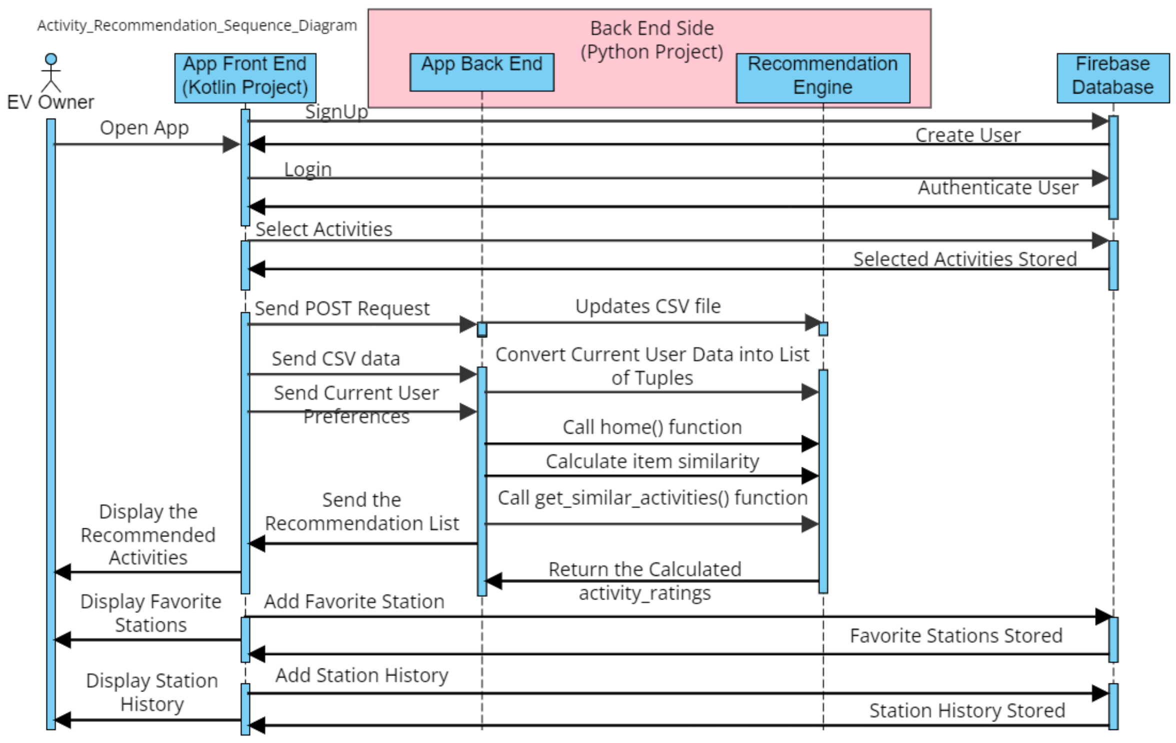

Activity_Recommendation_Sequence_Diagram
EV Owner
App Front End (Kotlin Project)
Back End Side (Python Project)
App Back End
Recommendation Engine
Firebase Database
Open App
SignUp
Create User
Login
Authenticate User
Select Activities
Selected Activities Stored
Send POST Request
Updates CSV file
Send CSV data
Convert Current User Data into List of Tuples
Send Current User Preferences
Call home() function
Calculate item similarity
Call get_similar_activities() function
Send the Recommendation List
Display the Recommended Activities
Return the Calculated activity_ratings
Display Favorite Stations
Add Favorite Station
Favorite Stations Stored
Display Station History
Add Station History
Station History Stored

Figure 4.7. Sequence Diagram

4.2. Generating Recommendations

In the home() function which exists in the Python project's view.py file, there are some variables. The path to the dataset that contains all the preferences of users, is specified by the variable (CSV File Path). To store a Data Frame of pairwise cosine similarities between activities, create a global variable. A global variable (Activity Ratings) holds the ratings of suggested activities an the user's currently selected activities are stored in the global variable (Current Data List). The one of the most important points is the home() function that is capable of reading the dataset, initialize the item similarity and computes the item similarity matrix by getting help from cosine similarity algorithm.

To generate the personalized activity recommendations, I prefer to use collaborative filtering algorithm since it is worked with a logic in the Figure 4.8. It is implemented in the function for getting similar activities. It accepts an activity name as input and outputs a Pandas Series which is a Python package that provides a efficient data structure management, using the activity similarity scores. Recommendations based on the user's chosen activities are generated using this function. While getting the recommendations, the cosine similarity algorithm which compares the similar vectors, is used and in addition to that, I use Scikit-learn

Python package for machine learning algorithms. In my project, the activity preferences of different users represents the cosine similarity vectors. User UIDs are the rows and activity names are the columns. If the user selects that activity, the related cell which is the intersection of user UID and the activity name, is set as value of 1. And if it is not, it is set as value of 0. Cosine similarity determines the angle between two of these vectors' cosines. If the angle is small, it means that the cosine similarity is high and these users' activity selections are similar to each other.

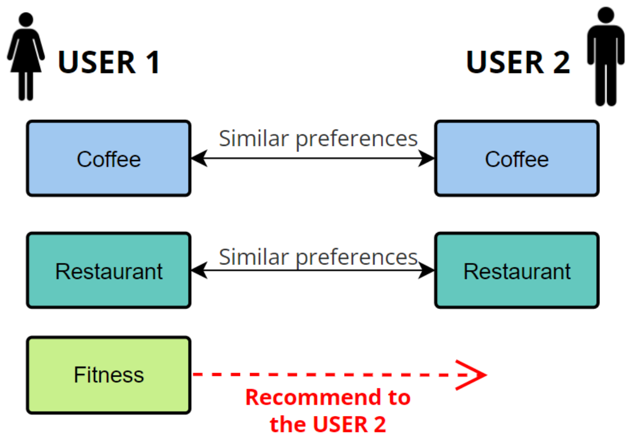


Figure 4.8. Recommendation Logic

5. TEST AND RESULTS

I apply usability, reliability, performance and security tests to decide if the application components work correctly with a secure environment and high performance.

Following test results are acquired from user-activity dataset results. Since the actual dataset includes a lot of activity options, I use an example dataset with less activity options to show that the algorithm works correctly. Firstly, the dataset is obtained by the EV users' activity selections. And every time a new user makes a selection, the dataset is updated. The example dataset in the following table consists of 4 users and 13 activities but normally, we have 29 activity options. The last row represents the current user's preferences.

Table 5.1. Example Dataset.

	Co.	Fi.	Wa.	Pl.	Sh.	Ci.	Li.	Re.	Ar.	Con.	Ma.	Sp.	Ka.
671X...	1	0	0	0	1	0	0	1	0	1	0	1	0
p7wm...	1	0	1	1	1	1	0	0	0	0	0	1	1
pUtw...	1	0	0	0	1	0	1	1	1	1	0	1	0
J042...	1	1	1	0	0	0	0	1	0	0	1	0	0

1: denotes selected activities
0: denotes non-selected activities
Co.: Coffee
Fi.: Fitness
Wa.: Walking
Pl.: Playground
Sh.: Shopping
Ci.: Cinema
Li.: Library
Re.: Restaurant
Ar.: Art Gallery
Co.: Concert
Ma.: Market
Ka.: Karaoke

When the item similarities based on the user preferences, are calculated; we come up with a result as seen in the Figure 5.1. As we see, the most preferable items by the users get the higher rates which means that the user who likes the item1 tends to like the item2 since the similar user likes the item2. And then, the similarity calculation becomes higher for item1 and item2. In our case, User ’J042...’ prefers *’Coffee’*, *’Fitness’*, *’Walking’*, *’Restaurant’* and *’Market’* activity options during EV charging. When we come to the recommendation part; the first 4 activity options should belong to the actual user preferences and the rest of them should decided by similar user activities. The other 3 users are similar to the fourth user because they have the similar selections such as *’Coffee’*, *’Restaurant’* or *’Walking’*. Since all of these 3 users prefer *’Shopping’* and *’Spa’* options commonly, the higher recommendation rates should belong to these options. The *’Concert’* option should have less recommendation rate because it is only preferred by 2 similar users and the current user is less similar to these users. *’Playground’*, *’Cinema’* and *’Karaoke’* options are also preferred by only 1 user. So, it also has to have less recommendation rate. User ’pUtw...’ has the least similarity with the current User ’J042...’. So that, the preferences of that user such as *’Art Gallery’* and *’Library’*, should be recommended in the last place. In summary, the order of the recommended activities should be like in this order:

’Fitness’, ’Market’, ’Coffee’, ’Walking’, ’Restaurant’, ’Shopping’, ’Spa’, ’Concert’, ’Playground’, ’Cinema’, ’Karaoke’, ’Library’, ’Art Gallery’

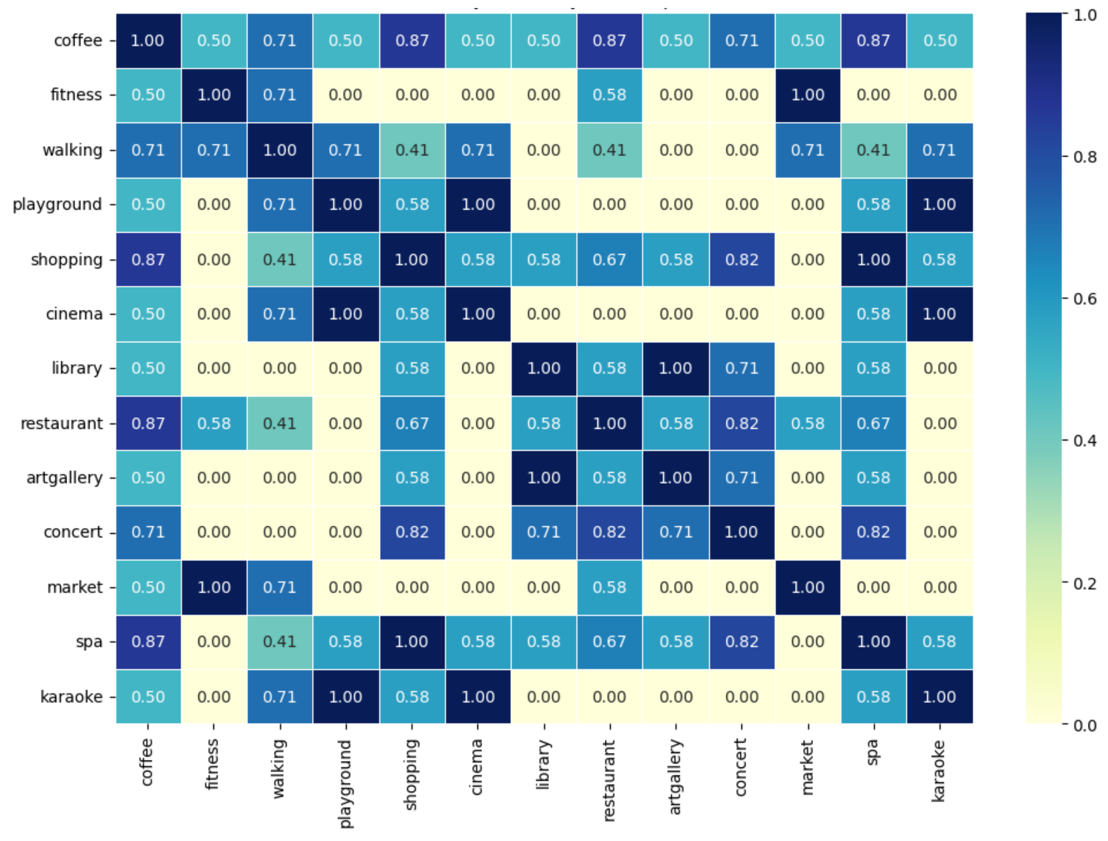


Figure 5.1. Similarity Heat Map

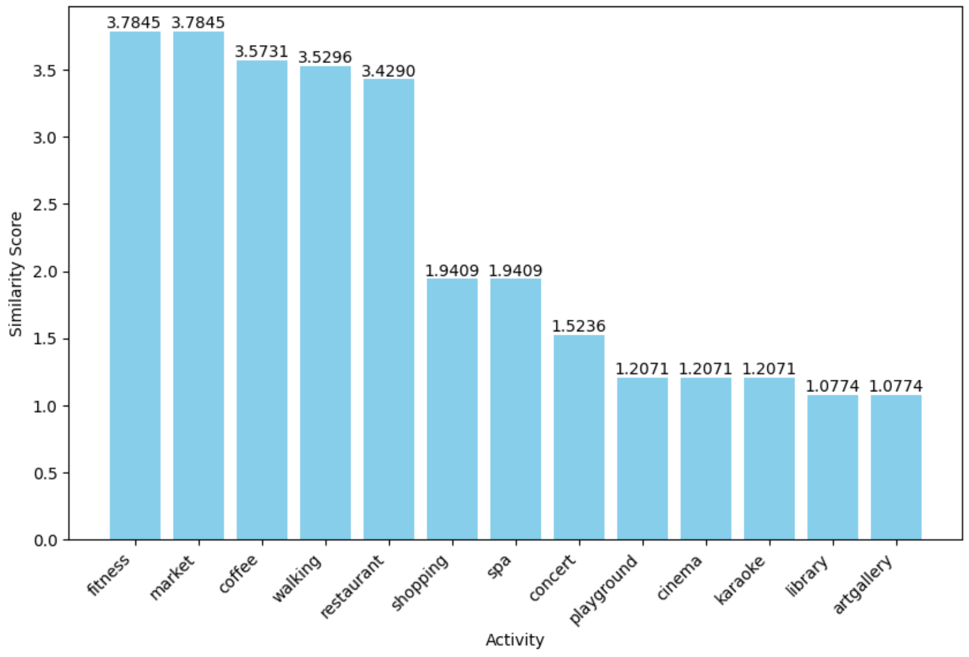
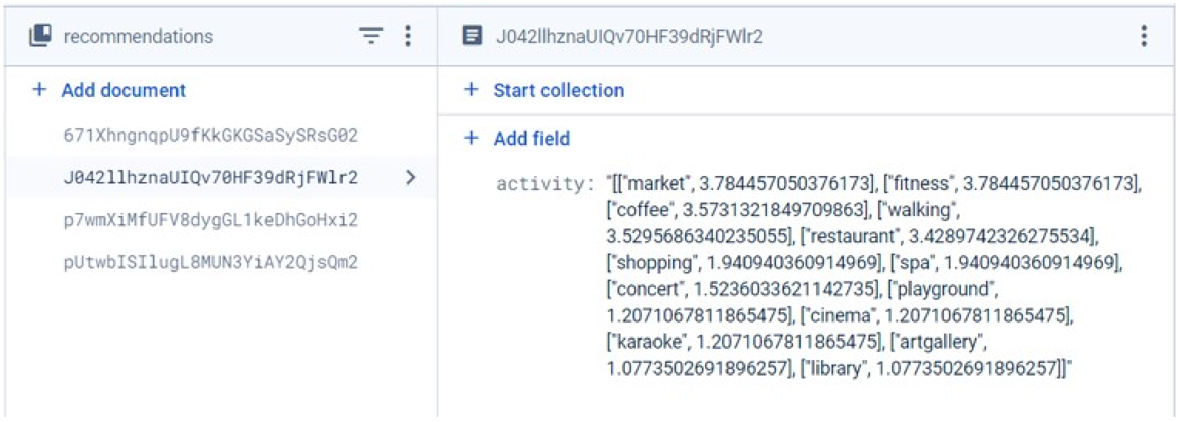


Figure 5.2. Top Recommended Activities for The Current Preferences of User

As we see in the Figure 5.2, the results confirmed that the algorithm works correctly and gives the accurate personalized recommendations for the EV user. The output is in the expected order and reliability principle is ensured. The activity preferences of similar users are recommended to the current user based on his/her selections. When we also check the database if the expected recommendations are stored, they are stored at the Firebase Database in the *'recommendations'* collection correctly to be used in the future as seen in the Figure 5.3.



```
671XhngnqpU9fKkGKGSaSySRsG02
J042llhznaUIQv70HF39dRjFWlr2
p7wmXiMfUFV8dygGL1keDhGoHxi2
pUtwbISIlugL8MUN3YiAY2QjsQm2

activity: "[["market", 3.784457050376173], ["fitness", 3.784457050376173],
          ["coffee", 3.5731321849709863], ["walking",
          3.5295686340235055], ["restaurant", 3.4289742326275534],
          ["shopping", 1.940940360914969], ["spa", 1.940940360914969],
          ["concert", 1.5236033621142735], ["playground",
          1.2071067811865475], ["cinema", 1.2071067811865475],
          ["karaoke", 1.2071067811865475], ["artgallery",
          1.0773502691896257], ["library", 1.0773502691896257]]"
```

Figure 5.3. Recommendation Output

I apply a speed performance test to understand if the algorithms work efficiently. I perform 100 tests to get an average value. Finally, the results from speed tests for machine learning algorithm confirmed that the algorithm works fast with the average speed value of 0.574 as seen in the Figure 5.4.

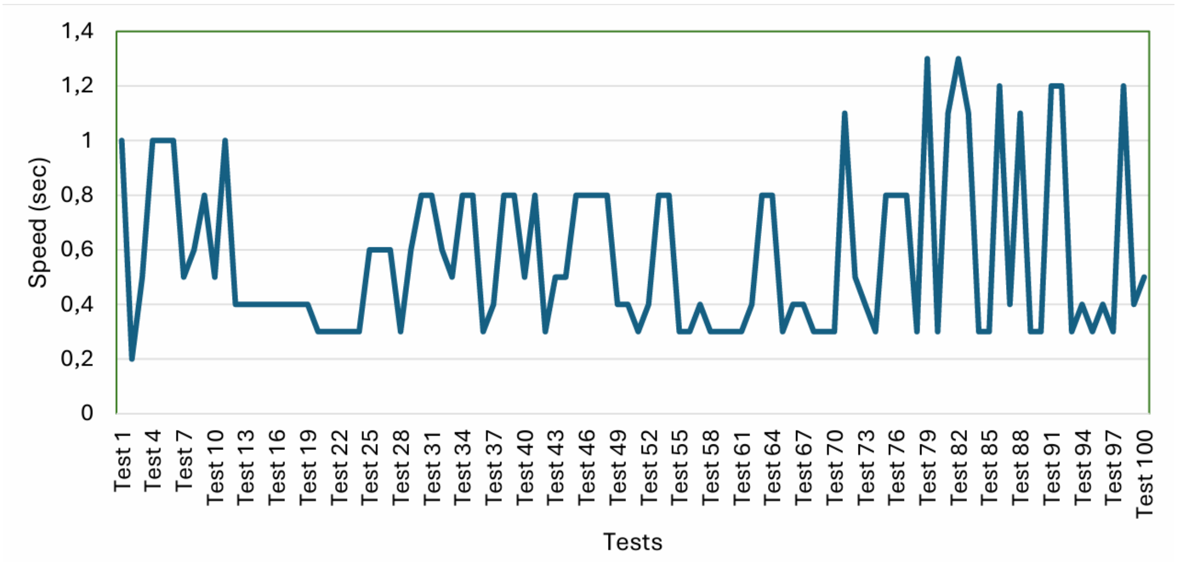


Figure 5.4. Speed Test for Machine Learning Algorithm

In the usability test, 10 users tested the mobile application and made some choices for the EV charging station activities. The results are in the following graphs 5.5 and 5.6.

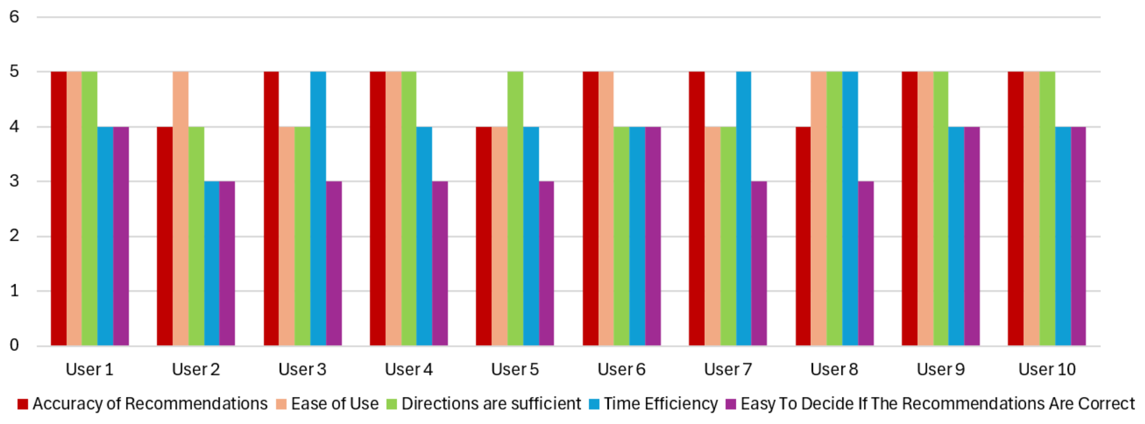


Figure 5.5. Usability Test

I also test the response time between front end and back end. When an EV user sends a

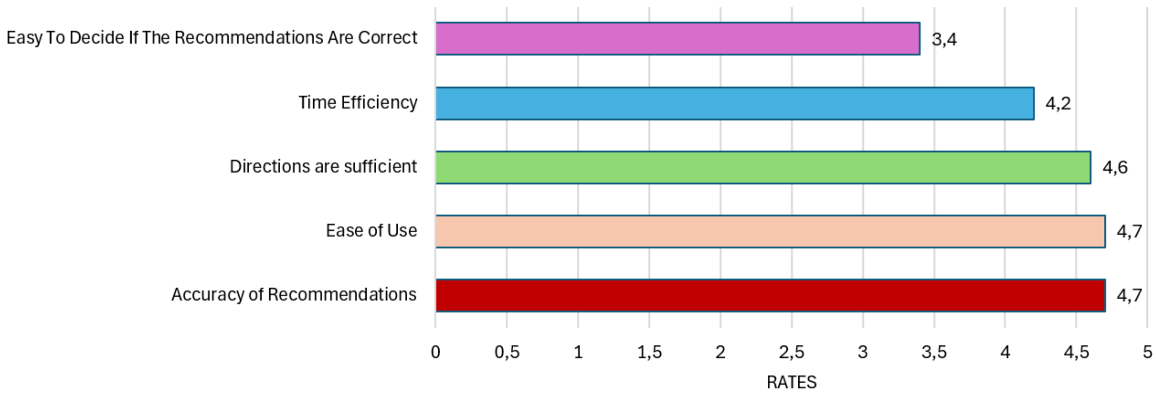


Figure 5.6. Average Result of Usability Test

request to the Python project, it takes an average of 0.405 seconds for the Python project to prepare the response and send the Kotlin project as seen in the Figure 5.7.

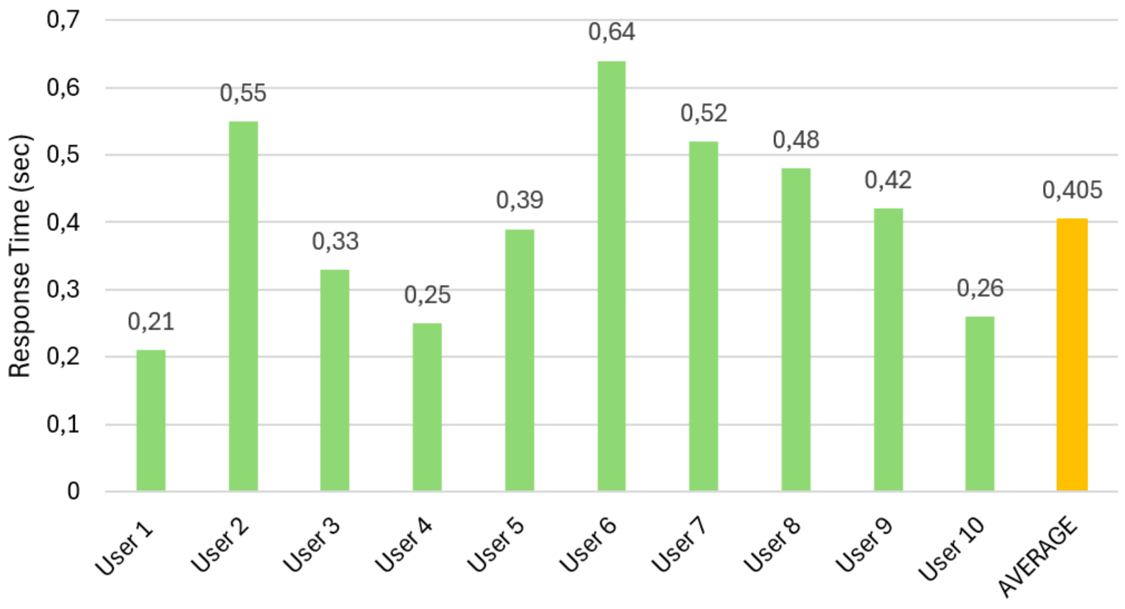


Figure 5.7. Response Time Between Front End and Back End

In addition to the previous performance test, CPU utilization of the Android Studio that I use to develop my Kotlin project and the VSCode that I use to develop Python Project, are like in the following Figure 5.8.

Lastly, I use authentication mechanism provided by Firebase to provide the secure login

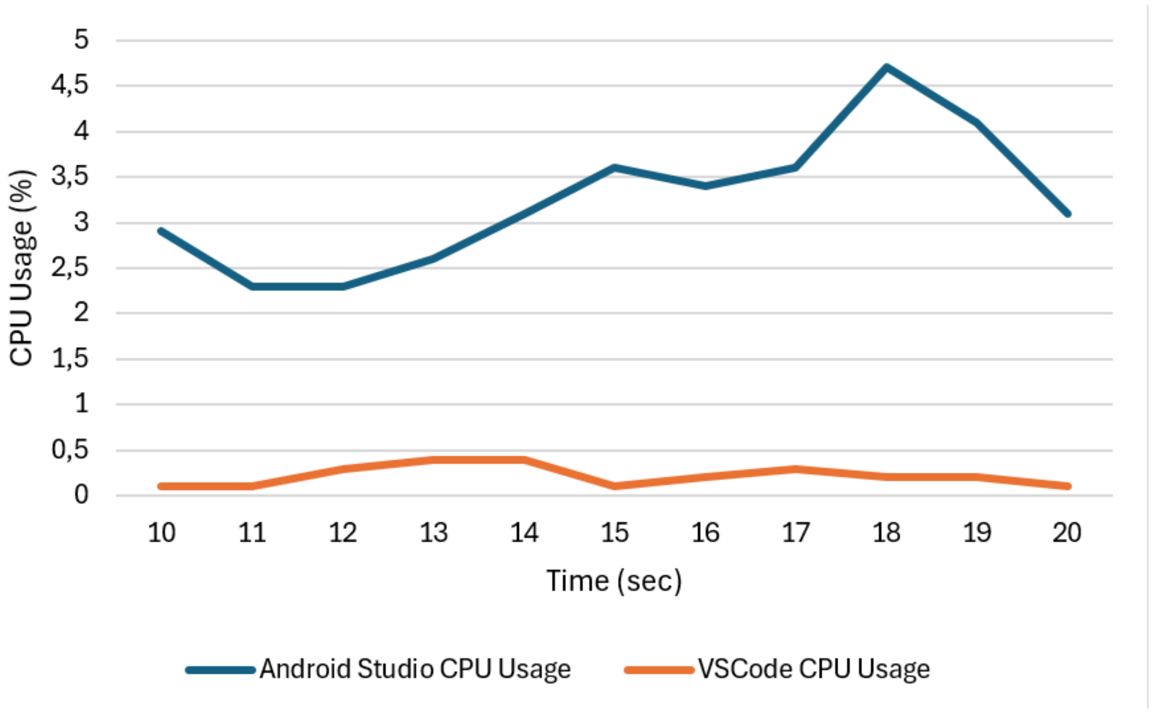


Figure 5.8. CPU Utilization

process. My project also provides the confidentiality principle, so that the data belong to a user is only accessible by that user. In addition, the user can access his/her data whenever he/she wants because all the data of the user is stored at the database and kept appropriately thanks to specific **collections** in the Firebase. Therefore, the availability and integrity principles are provided by my project.

6. CONCLUSION

Implemented machine learning algorithm which has an efficient speed while accomplishing the required process, has good success on creating accurate recommendations based on the EV users' preferences. In addition, the usability of the application is easy and user-friendly and the user experiences are improved for EV owners to save time.

6.1. Future Work

To improve the algorithm, hybrid recommendation system algorithms which consist of both collaborative and content based filtering, can be used to get more accurate, personalized and balanced recommendations. And also, the user interface can be more detailed such as displaying charging duration to achieve the better user experience. In the future, the data obtained by the EV Users' activity preferences can be used for specifying the location of the charging stations.

Bibliography

[1] IEA, "Global ev outlook 2023," 2023.

[2] Z. Lanyun, C. Rebeccaa, and R. Tracy, "Designing the user experience for new modes of electric vehicle charging: A shared vision, potential user issues and user attitudes," pp. 1–23, 2019.

[3] O. Frendo, J. Graf, N. Gaertner, and H. Stuckenschmidt, "Data-driven smart charging for heterogeneous electric vehicle fleets," vol. 1, 2020.

[4] S. Pareek, A. Sujil, S. Ratra, and R. Kumar, "Electric vehicle charging station challenges and opportunities: A future perspective," 2020.

[5] C. Li, S. Zhang, W. Ling, L. Zhao, and Y. Pan, "Enhancing user experience in electric vehicle charging applications (evca): A comprehensive analysis in the chinese context," pp. 1–23, 2024.

[6] A. ALBATAYNEH, M. N. ASSAF, D. ALTERMAN, and M. JARADAT, "Comparison of the overall energy efficiency for internal combustion engine vehicles and electric vehicles," vol. 24, pp. 1–12, 2020.

[7] T. H. W. Rebecca S. Levinson, "Impact of public electric vehicle charging infrastructure," 2020.

[8] P. Fabianek, "Multi-criteria assessment of the user experience at e-vehicle charging stations in germany," 2023.

[9] S. S. Gregory J. Carlton, "Electric vehicle charging station accessibility and land use clustering: A case study of the chicago region," vol. 9, pp. 5–10, 2021.

[10] W. Wang, X. Peng, J. Jia, J. Zhao, W. Xiao, and S. Su, "Optimal user oriented multi-level experience planning strategy for electric automobile charging path," 2020.

[11] E. P. R. I. (EPRI), "Interoperability of public electric vehicle charging infrastructure," 2019.

[12] C. Liu, K. K. Chai, X. Zhang, and Y. Che, "Enhanced proof-of-benefit: A secure blockchain-enabled ev charging system," 2019.

[13] F. Alanazi, "Electric vehicles: Benefits, challenges, and potential solutions for widespread adaptation," pp. 1–23, 2023.

[14] A. Ibrahim, E.-S. M. El-Kenawy, M. M. Eid, and A. A. Abdelhamid, "A recommendation system for electric vehicles users based on restricted boltzmann machine and waterwheel plant algorithms," pp. 1–26, 2023.

[15] R. M. S. Bhaskaran, “Enhanced personalized recommendation system for machine learning public datasets: Generalized modeling, simulation, significant results and analysis,” vol. 6, pp. 1583–1595, 2023.

[16] J. Ferreira, P. Pereira, P. P. Filipe, and J. Afonso, “Recommender system for drivers of electric vehicles,” vol. 5, 2011.

[17] P. J. Pannee Suanpang, “Optimizing electric vehicle charging recommendation in smart cities: A multi-agent reinforcement learning approach,” pp. 1–34, 2024.

[18] S. Liu, X. Xia, Y. Cao, Q. Ni, X. Zhang, and L. Xu, “Reservation-based ev charging recommendation concerning charging urgency policy,” vol. 74, 2021.