BSc (Hons) in Computing

Level – 6

Final Year Project Documentation

Module Title: Final Year Project

Prepared by:

Shifny Ahmed CB007753

[HF2161SEENG]

Date of Submission: 11th February 2022

Instructor: Prof. Priyantha Kumarawadu

Word Count: 10313

Acknowledgement

I would like to thank Professor Priyantha for providing all the advice and teaching required to do my FYP and I would like to thank my colleagues for helping me on any mistakes I make regarding my FYP. And finally, I would like to thank Staffordshire University for providing access for most of the online resources required for my research need.

Table of Contents

Table of figures

01. Introduction

In this modern world, people often get used to being served at their desired place, which is mostly at home. These people expect service providers such as doctors, nurses, barbers, plumbers, electricians, carpenters, car washers, cleaners, etc. to visit their home and offer their services, and most of those people have already been hiring these types of service providers to come to their home. This hiring is being done in a manual way by asking for suggestions of well-known service providers to people for the work they want done, and not every individual service provider in the above-mentioned category will be available all the time and the manual process of hiring will cost more time and effort.

In order to automate this process, we will be building a system that will allow people to register through the application as customers and hire their desired service provider for the service they require. This system will also recommend service providers to the customer based on the city customer lives and based on the work of any selected service provider, the first release of the system will be on website which will have several functionalities for the customers in a user-friendly manner.

01.1. Project Aim, Objective, and Scope

The main goal of this project is to connect customers with local professional service providers. This can be done by creating a recommendation system that will recommend the best service providers in their city as well as service providers based on their preferences.

We will implement a web application that will allow customers all over Sri Lanka to register and hire the best service provider in their city. Service providers in each category will be recommended to the customer based on the city and the customer's interests. The project's final product will be the first version of a web application that allows customers to register and hire only plumbers in their city, and several plumbers will be recommended based on their selection.

In order to start the project few areas have to be researched to understand the basics such as:

- **User Profiling** - Analyzing the characteristics of users' interest is a crucial part in developing a recommendation system for home services.
- **Recommendation system using machine learning -** As machine learning has shown great success in predicting information based on the past information provided, this can be used to recommend a service provider to the customer based on service provider rating.
- **Collaborative Filtering** - This is a filtering strategy used to recommend a user's preference to another similar user.
- **Content-Based Filtering** - This is the normal strategy used in most of the recommendation system which recommends the user based on the past interaction and preference of the user.
- **Python and Machine Learning** - Python is among the first programming languages to gain machine learning support in the form of libraries and tools. This can be researched for further details to learn the language.
- **Similar Web based Service providing System** - This can be researched to learn what functionalities are most popular among them and how much time they take to get implemented.

After the research and efficient information has been collected, we will be moving onto the implementation where we will be applying the dos and don'ts which we had identified. Dos and don'ts are listed below:

- The payment gate way won't be implemented in the first version of this application.
- The system builder will need to implement the application in a way where only plumber needs to be available in the service providing section in the first version of the application.
- The system builder will need to implement the application in a way where customer will be able to see the information such as name, work, city, occupation, qualification, availability, and customer reviews of the service provider.
- The system will not have any implementation where the customer can call the hired service provider through application.
- The system will not have any implementation where the customer can track the live location of the hired service provider.

- This system will only allow have customer's functionalities in the first version of this application.

02. Literature Review

02.1. Introduction

People in today's environment are accustomed to being served at their preferred location, which is usually their house. These people expect service providers such as doctors, nurses, barbers, plumbers, electricians, carpenters, car washers, cleaners, and other professionals to come to their homes and offer their services, and the majority of them have already done so. This hiring is done manually by asking known people for service providers to be suggested for the work they need done, and not every service provider in the above-mentioned category will be available at all times. In this instance, the customer may have to wait for their turn to be served. To solve this, we came up with a system which is built using machine learning to offer a Service Provider Recommendation System which will list down the available service providers and will recommend the best suiting service providers according to each user's interest.

To improve the efficiency of this home service recommendation system, it is critical to manage data generated on a daily basis in the home service application correctly, and to do so, systems must execute data collection, processing, analysis, and distribution (Garcia, et al., 2020)**.** Proper data management can boost productivity by allowing you to hire more service providers and earn more trust in the system. Over the year the amount of data generated by global sources of data such as Web sites, social media, smartphone apps, news networks, weather, political institutes, society, and the economy has been steadily increasing. Without appropriate preparation and processing, no matter how large the data is, it may be useless (Ribeiro, et al., 2015). In the home service application, best service providers over the years can be recommended to all the customers whenever the application is accessed, by collecting various customer rating on each service provider to arrange each service provider by ascending order in terms of overall rating and recommend them to the customer or the service providers may be recommended based on individual customer preference which will be collected based on the customer's past activity. To make such recommendation, Machine learning is often used.

02.2. Machine Learning and Recommendation Techniques

Machine Learning is a set of statistical methods for creating mathematical models that can draw conclusions from data samples (known as a training set) (Ribeiro, et al., 2015). It is among the most rapidly developing areas of computer science (Alpaydin, 2020). For scientific modeling, consumer behavior, energy consumption predictions, related article recommendation, and user trends, a variety of machine learning algorithms have been applied to extract useful knowledge from data (Ribeiro, et al., 2015).

There are many machine learning algorithms such as Association Rules, Unsupervised learning, Supervised Learning, and Reinforcement learning. Association rules is often used in supermarket basket analysis, this is the process of defining associations between things purchased by customers. For example: Consumers who buy milk are likely to buy Choco powder as well, and buyers who buy milk, but not Choco powder are potential Choco powder customers (Alpaydin, 2020). Unsupervised learning is a basic idea in which we don't have a supervisor to offer accurate values and only have input data to detect regularities in (Alpaydin, 2020), this is mostly used for unlabeled datasets. Supervised learning is vice versa explanation of unsupervised learning, we will be doing the machine learning with datasets which are labeled, few examples of the algorithms used are: Classification, Linear Regression, Naïve bayes and etc. Reinforcement learning is where the output of the system will be sequence of action which will be used to predict the future actions. One place where this can be used is gaming. Chess, for example, has a modest number of rules but is extremely complex due to the enormous number of possible moves at each phase and the overall number of moves throughout the game and good algorithm can be used which can learn to play games well by learning all the possible sequence of action (Alpaydin, 2020).

Techniques Used to build a recommendation system are many, 3 of those will be discussed below:

02.2.1. Content based Filtering

Based on the user's prior activities or input, content-based filtering recommends things that are similar to what they like. Some user-related characteristics may be offered expressly by the user. Other aspects, such as the programs they've already installed, may be implied.

This is one of the most effective recommendation techniques for new projects with little or no user data.

- **Implementation Process**

  There are two methods that may be applied in this situation. To begin, customers can be provided a list of attributes from which they can select the ones that they most identify with. Second, the algorithm may remember which goods the user has previously selected and add those characteristics to the user's data (Shoval, et al., 2008).

- **Advantage**

  Due to the little quantity of data, this approach is easily scalable. Furthermore, as it’s not required to compare data with other users, unlike other models, it may provide specialized findings tailored to the present user (Shoval, et al., 2008).

- **Disadvantage**

  This model demands a significant level of domain expertise on the part of those tasked with attributing features to products. As a result, its accuracy is highly dependent on the accuracy of that knowledge. Furthermore, content-based filtering is heavily reliant on previously identified user preferences. As a result, it is restricted in that it cannot expand on previously identified user interests (Shoval, et al., 2008).

02.2.2. Collaborative Filtering

Collaborative filtering is a method to extract items that a user might like based on the actions of other users. It works by scanning through a huge group of users to discover a smaller group of users with similar preferences to a specific user. It analyzes their favorite products and creates a ranked list of recommendations (Ekstrand, et al., 2010)**.**

- **Implementation Process**

  The initial step is to look for users or objects that are comparable. The second phase is to forecast user ratings for things that have not yet been rated.

02.2.3. Demographic Filtering

Users are classified into demographic groups based on their personal characteristics. The input data for the recommendation process comes from these classes. The goal of this technique is to identify groups of people who enjoy a particular product rather than a single brand or service (Ryngksai & Chameikho, 2014).

- **Implementation Process**

  If people in class B enjoy product A and there is a person b in class B who has not yet seen product A, this product can be recommended to person b (Ryngksai & Chameikho, 2014).

Any recommendation system's main idea is to recommend items to the potential customer by predicting their utility based on previous user actions or similar user actions. It's possible that the user profile will be utilized to filter the user's preferred content. Basically, Content-Based-Filtering predicts and recommends contents, including unseen ones, to new incoming users based on the preferences of the current user. These items may be comparable to previous rated items or contain keywords that are similar (Paireekreng, 2013). In the Collaborative Filtering technique, on the other hand, the preferences of other users must be considered for the user who is now using the recommendation system (Xu, et al., 2010). Similar users can be identified in collaborative filtering using their frequent preferences and in future, preference of one user can be recommended to other similar users. But Meteren and Someren suggested to construct a recommendation system based on content-based-filtering and collaborative filtering jointly to make the system more effective. They also discovered that the precision ratios for prediction for different topics in the same document could differ (Meteren & Someren, 2000).

02.3. User Profiling in Recommendation System

User profile is an important part of the recommendation system since it allows users to get personalized content and be recommended items that are relevant to them (Paireekreng, 2013). Wagner was the first to develop the concept of a user profile (Wagner, et al., 2002). The study proposed a framework for sophisticated mobile service personalization based on a profiling strategy that makes use of the semantic enrichment service. User modeling can assist in the creation of a user profile based on demographic factors in order to determine a user's characteristics. This can give a group of people with content that is relevant for their requirements (Paireekreng, 2013). The early-stage difficulty in the recommendation system can be addressed by the user profile, which tackles the issue of a user's lack of information.

02.4. Text Vectorization

Text Vectorization is a process that involves converting text into numerical form. There are several ways to perform text vectorization.

02.4.1. TF-IDF

The TF-IDF algorithm compares the relative frequency of words in a single document to the inverse proportion of that word across the entire document corpus. This determines the importance of a provided word in a given document (Salton & Buckley, 1988).

The TF-IDF is calculated as follows: Given a document collection *D*, a word *w*, and a single document $d \in D$, we calculate the TF-IDF values.

$$w_d = f_{w,\,d} * \log\left(|D|/f_{w,\,D}\right) \quad (2),$$

Figure 1: TF-IDF-Formulae

where $f_{w,\,d}$ is the number of times w appears in *d*, $|D|$ is the corpus size, and f_w, *D* is the number of documents where w appears in *D* (Salton & Buckley, 1988).

02.4.2. Count Vectorizer

The scikit-learn Python library provides Count Vectorizer. It is used to convert a text into a vector based on the frequency of each word that appears throughout the text. The performance of Count Vectorizer increases when used on short words which is mostly the case in our project (Geeksforgeeks, 2020).

We're looking for similarities between service providers based on the "work" and "city" columns, which contain sentences of no more than 2-4 words. The TF-IDF algorithm tends to downplay the significance of words that are more common throughout the corpus. Because our work and city columns have a limited number of words, each word is equally important. As a result, using the "Count Vectorizer" for this project would be appropriate.

02.5. Similarity Measuring

We may need to determine how similar a product is to one that the customer likes or is interested in before we can recommend it to them. This is where similarity metrics come into play. Anything can be used to determine how similar two products are. For example: the color, taste, or size of a fruit, can be used to determine its similarity.

In machine learning, similarity scores range from 0 to 1, with 1 indicating the most similar product (100 percent similar) and 0 indicating the least similar product (0 percent similar).

02.5.1. Cosine Similarity

Cosine similarity is a metric for determining how similar products are regardless of size. It calculates the cosine of the angle formed by two vectors projected in three dimensions. This is also one of the most widely used approaches in machine learning for determining similarity. (Ye, 2011).

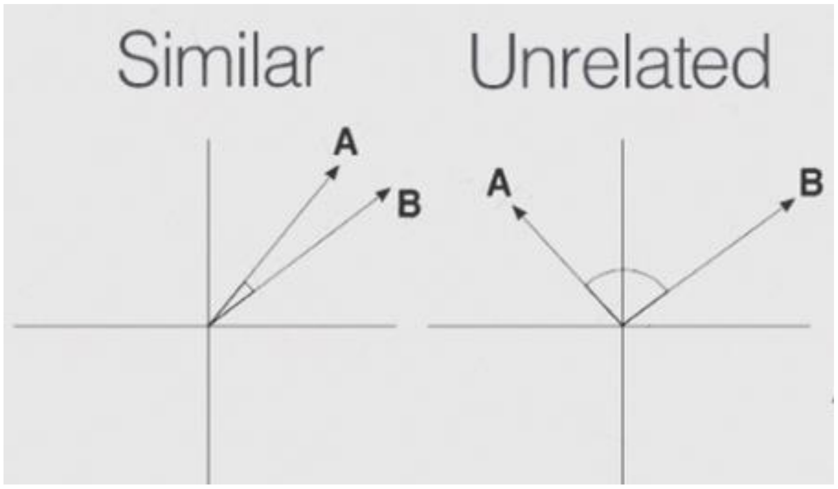


Figure 2: Cosine Similarity - Angle Definition

The cosine similarity measure is illustrated in the diagram above, where the similarity value is 0 when the angle between two vectors is 90 degrees; this indicates that the two vectors are unrelated. The similarity measure gets closer to 1 as the angle between the two vectors gets closer to each other; this means that both vectors are more similar to each other (Ye, 2011).

02.6. Conclusion

As machine learning has been found to be the only approach to building a recommendation system, different machine learning algorithms and filtering techniques have been found.

In this review, a home service recommendation system was proposed with all the relevant information required to implement and test it. The problem of the home service recommendation system in the early stages of recommendation can be considered based on content-based-filtering. And when the time processes, both collaborative and content-based filtering can be used jointly to make the system more effective, as suggested by Meteren and Someren (Meteren & Someren, 2000).

Content-based filtering can be accomplished in the first version by carrying out a few steps of the machine learning approach to any recommendation system. The training dataset needs to be preprocessed to remove any non-ASCII words and stop words, convert the words into lower case and attach them to the dataset as a new column. The words in the cleaned column can be converted into a vector based on the frequency of each word that appears throughout the column using the Count Vectorizer. Finally, to measure the similarity between the words, cosine similarity, which is one of the most widely used similarity measures in machine learning, can be used to find the similarity scores of each word in the column to each other. And to recommend any service provider to the customer, we can implement a function that takes into account the factor on which the recommendation needs to be made and recommend similar service providers to the customers.

The recommendation system can also utilize the benefit of user profiling, which can provide the user with personalized content and recommend relevant items to them. There is indeed a shift away from recommendations based on other users' ratings and toward personalized recommendations based on user preferences when the user shares more information related to the home services. Users can also manage their own preferences and profiles in order to receive customized content based on their user profile and how they manage their content choices.

03. Requirement Analysis

03.1. User Requirements

- **Customers**: These are public who will be able to register through our application to hire home service providers whenever required after the subscription has been made.

03.2. System Functionalities

- The system should be able to keep track of all the services hired by the customer in a database with all the relevant data including id of hired service provider and the customer.
- The system must be able to keep track of current hiring and hiring history of the logged in user.
- System must be able to provide recommendation based on the city user lives.
- System must be able to provide recommendation based on most frequently hired service in the user’s hiring history.
- System must recommend plumbers based on the work of any plumber the user selects.
- System must be able to send notification on important information of the user.
- System must be able to display the logged in user’s task bar after the user logs in.

03.3. Functional Requirements

Functional requirement is a software requirement which describes a functionality that a software component or system must be capable of accomplishing (Chung, et al., 2000). This section will be addressing all the functions that the system can perform.

- User must be able to register themselves in the application.
- A user authorization method must be included in the system, in which users must identify themselves using a username and password. Only users with this level of authorization have access to the system's data.
- The user must be able to log out of the application whenever wanted.
- User must be able to view their profile details
- User must be able to edit their profile details.
- User must be able to view currently hired plumber
- User must be able to view currently hired plumber details
- User must be able to update currently hired plumber as arrived
- User must be able to update hiring as work completed
- User must be able to view hired history of user
- User must be able to view details of individual hired history of user
- User must be able to add review on each hiring
- User must be able to view their review on each hiring
- User must be able to view all the customer review on all plumbers
- User should be able to hire a service provider by clicking the “Hire Now” button.

03.4. Non-Functional Requirements

In software system engineering, non-functional requirements are software requirements that specify how the software will accomplish something rather than explaining what it will do (Chung, et al., 2000).

➢ **Performance**

- The application will need to load the login page to the user within 3-5 sec of opening
- The application will need to load the home page to the user within 1-3 sec of logging in
- Each request made should be processed within 2-4 seconds.

➢ **Usability**

- Customer must be able to view the general details of the service providers.
- The Application UI needs to be engaging.
- The system home page needs to be manipulative to attract new customers

- Functionalities of the user must be handled and processed in a user-friendly manner.

➢ **Security**

- User password must be encrypted using strong hashing.
- Logged out user must not be able to make use of logged in user's functionality before logging in again

04. Methodology

04.1. Research and Application of Methods

As our system is about recommending a service provider to the customers, this application can be named as a recommendation system which basically is built using machine learning algorithms.

Here, we have decided to use a manually made plumber dataset which consist details of various Sri Lankan plumbers. We have decided to apply content-based filtering technique to overcome the starter problem.

04.2. Technology Selection

- Python and flask will be used to build this Home Service Recommender System using machine learning algorithms such as content-based filtering.
- As this is a web development project, no mobile development frameworks or libraries will be used.
- Content based Filtering will be used in this system to recommend service providers to the customers.

04.3. Development Platform

The final outcome of this project will be on a web application, so a development language, framework, and IDE will be needed to develop this application.

Development Language

Python is used as the development language as it plays important role in almost all the development such as gaming application, desktop applications, web applications, mobile applications and etc. It is also considered one of the easiest and high demand development language by most of the software developers.

Version: Python 3.9

For the front end of the application, HTML5 with Bootstrap has been used. To make the front end presentable few CSS and java scripts has also been used. To run python code within html, Jinja2 has been used.

Framework

Python is one of the easiest and most powerful development languages, and there are a variety of frameworks available for it.

Django and Flask are the two most popular web frameworks in Python, and we chose Flask as the web framework for this application because it is more understandable and easier to use.

IDE

JetBrains' PyCharm is one of the most widely used Python IDEs. PyCharm is the tool that Python developers need to get the job done quickly. PyCharm allows developers to write clean codes.

05. Solution Concept

The fundamental idea behind this application is for the system to recommend a service provider to customers depending on their interests. This might be accomplished by enabling the system to have functionalities for user: customers, user can register/login and use the functionalities made accessible to them. The application's solution idea is illustrated in the High-Level Component Diagram below, which shows the system's six primary components: database, customer, service provider, security, service, and recommender. To begin, the customer will require access control, which will be given by the security component. The customer components will rely on the Home Service Database component to complete this procedure. In order for the customer component to look for and engage with the service component, the service component will require the service provider component to provide their service. Finally, the recommender component will handle the fundamental concept of recommending a service provider, and it will rely on the home service database component to give the necessary data. The recommender component will process this data and provide recommendations to the customer component.

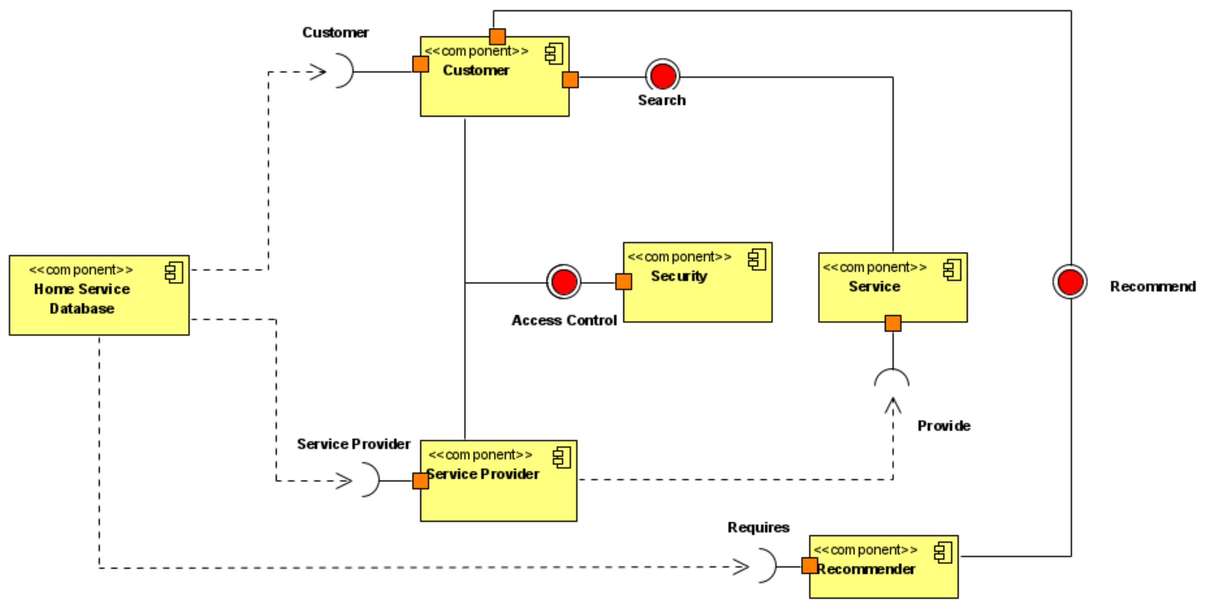


Figure 3: High Level Component Diagram

06. System Design

06.1. High-level use case

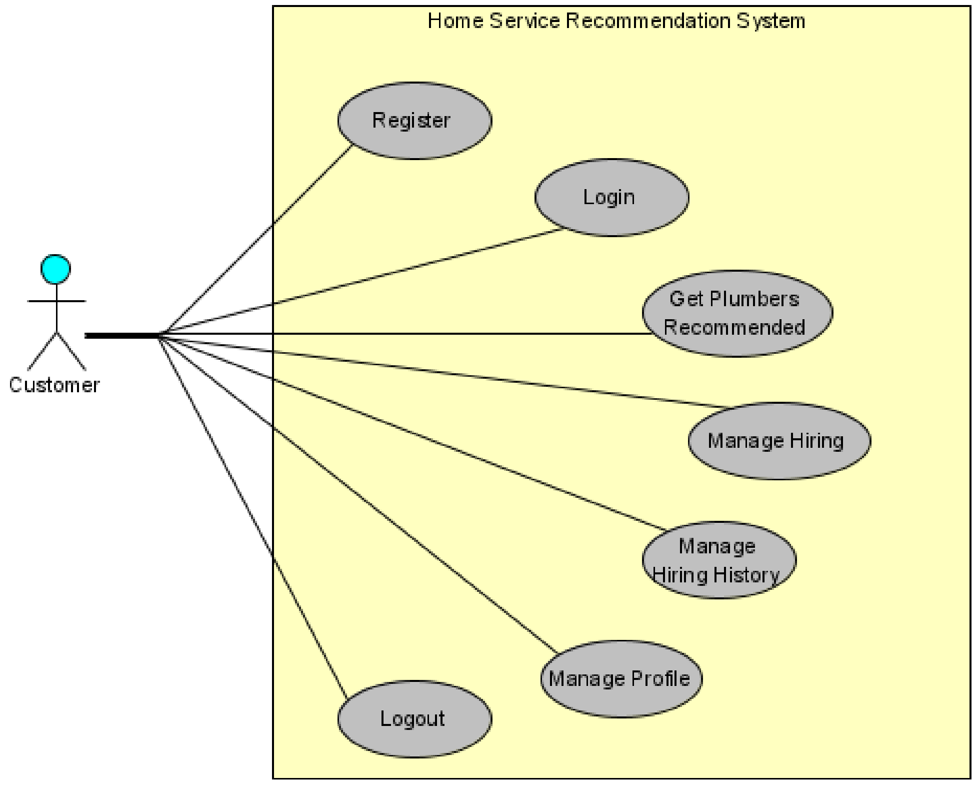


Figure 4: High-Level Use Case

06.2. Entity Relationship Diagram

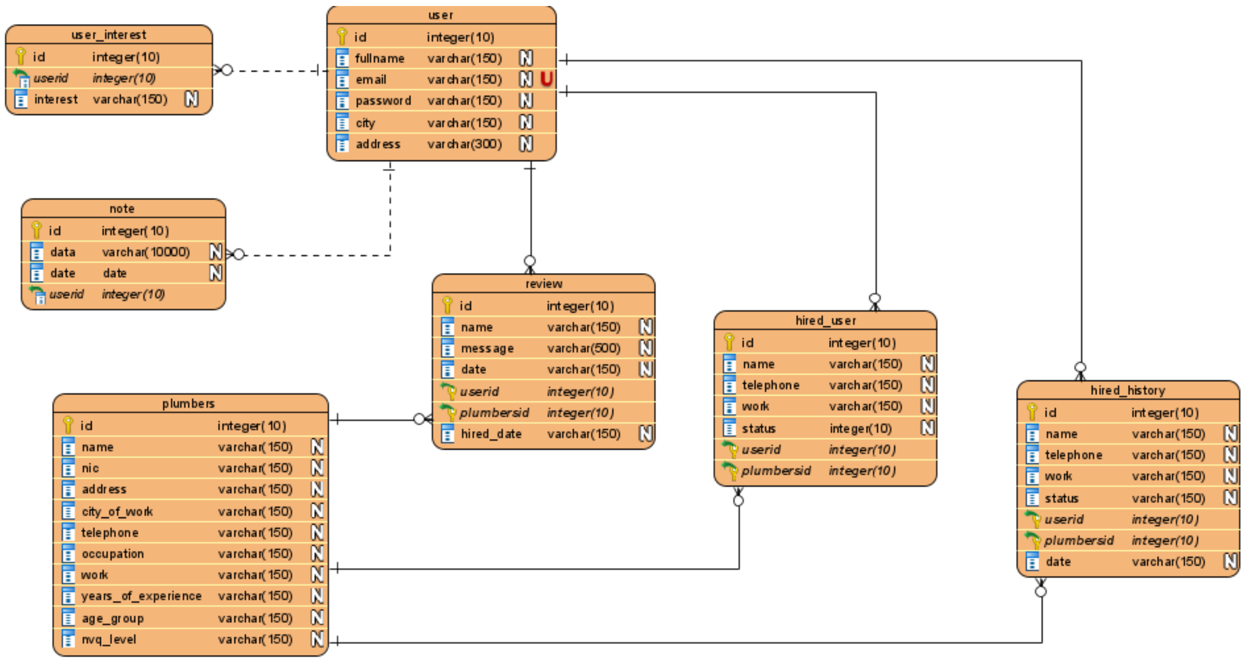


Figure 5: ER Diagram

06.3. Use Case Scenarios

USE CASE NAME	REGISTER
DESCRIPTION	Allows the user to register into the system
PRIMARY ACTOR	Customer
PRE-CONDITIONS	• System should be up and running
BASIC FLOW	1. System displays the login page 2. User clicks signup 3. System redirects user to their signup page 4. User enters the required information 5. User clicks the “signup” button. 6. System validates the information and saves in database 7. Display success message
EXTENSIONS	5A) If any of the required information is empty 1. System prompts the user to fill up the required information. 6A) If the entered password and confirm password does not match 1. The system displays error message 6B) If the entered email already exists 1.System displays an error message 6C) If full name field has value less than 3 character 1.System displays an error message 6D) If address is less than 10 characters 1.System displays an error message
POST CONDITION	The user can login to the system.
OPTIONAL INFORMATION	The password can be in between 7 to 10 characters.

USE CASE NAME	LOGIN
DESCRIPTION	Allows the user to login to the system.
PRIMARY ACTOR	Customer and Service Provider
PRE-CONDITIONS	• System must be up and running • User must be registered
BASIC FLOW	1. System displays the login page 2. User enters the username and password 3. User clicks "Login" button. 4. The system checks whether user exist 5. The system checks whether the entered password matched existing username. 6. System direct the user to their home page.
EXTENSIONS	3A) If the Username or Password is Empty 1. System prompts the user to fill the username or password fields 4A) If the user entered username does not exist 1. System display error message 5A) If the user entered password does not match existing username 1. System display error message
POST CONDITION	Hire a plumber

USE CASE NAME	RECOMMEND SERVICE PROVIDER BASED ON USER INTEREST
DESCRIPTION	Allows the user to get recommendation from the system based on the interest of the currently logged in user.
PRIMARY ACTOR	Customer
PRE-CONDITIONS	• System must be up and running • Customer must be logged in
BASIC FLOW	1. System displays customer homepage 2. Customer clicks “hire” 3.Display Plumber’s page 4. Customer clicks “Recommend based on interest” 5. System checks whether user has any past hiring interest 6. System recommends few plumbers based on the frequently hired plumbering work of the currently logged in user. 7. Display Plumbers page 8. Display recommended plumbers
EXTENSIONS	5A) If user has no past hiring interest 1. Display error message
POST CONDITION	Hire plumber

USE CASE NAME	ADD REVIEW
DESCRIPTION	Customer can add review about the service customer has hired
PRIMARY ACTOR	Customer
PRE-CONDITIONS	• System must be up and running • User must be logged in • User must have hired a plumber • Plumber must have completed his/her work
BASIC FLOW	1. User clicks the hiring history in navigation bar 2. User clicks “view details” of any plumber 3. System displays the past details of hiring to the user 4. User enters a review 5. User clicks add review 6. System adds the review to the database with customer id, plumber id, review message and date 7. Display success message 8. Display the added review for the selected service.
EXTENSIONS	5A) If review is empty 1.Prompt user to fill up the review field

USE CASE NAME	UPDATE PLUMBER AS ARRIVED
DESCRIPTION	This function will be called by the user when he/she clicks “Arrived” button in current hiring page.
PRIMARY ACTOR	Customer
PRE-CONDITIONS	• System must be up and running • Customer must be logged in • User must have hired a plumber
BASIC FLOW	1. User clicks current hiring 2. System displays the currently hired plumber to the user 3. Customer clicks “Arrived” 4. System updates the status of the current hiring as “Arrived’ 5. System sends a notification to the user 6. System displays success message
EXTENSIONS	-
POST CONDITION	Update as work completed

USE CASE NAME	UPDATE PLUMBER AS WORK COMPLETED
DESCRIPTION	This function will be called by the user when he/she clicks “Work Completed” button in current hiring page.
PRIMARY ACTOR	Customer
PRE-CONDITIONS	• System must be up and running • Customer must be logged in • User must have hired a plumber • User must have updated the current hiring as “Arrived”
BASIC FLOW	1. User clicks current hiring 2. System displays the currently hired plumber to the user 3. Customer clicks “Work Completed” 4. System saves the hiring details with status as “Work Completed” to hired history table. 5. System deletes the current hiring 6. System sends a notification to the user 7. System displays success message 8. System redirects the user to hired history page
EXTENSIONS	-
POST CONDITION	Add review

USE CASE NAME	EDIT PROFILE
DESCRIPTION	Allows the user to edit their profile
PRIMARY ACTOR	Customer and Service Provider
PRE-CONDITIONS	• System must be up and running • Customer must be logged in
BASIC FLOW	1. User clicks view profile in navigation bar 2. System redirects the user to profile page 3. User clicks edit profile 4. System redirects the user to edit profile page 5. User makes changes to his/her profile details 6. User clicks “update” 7. System validates updated details 8. System saves the updated information in database 9. Display Profile page 10. Display success message
EXTENSIONS	7A) If the entered email already exists and it’s not currently logged in user’s email 1.System displays an error message 7B) If full name field has value less than 3 character 1.System displays an error message 7C) If address is less than 10 characters 1.System displays an error message
POST CONDITION	View profile

06.4. Activity Diagram

Register

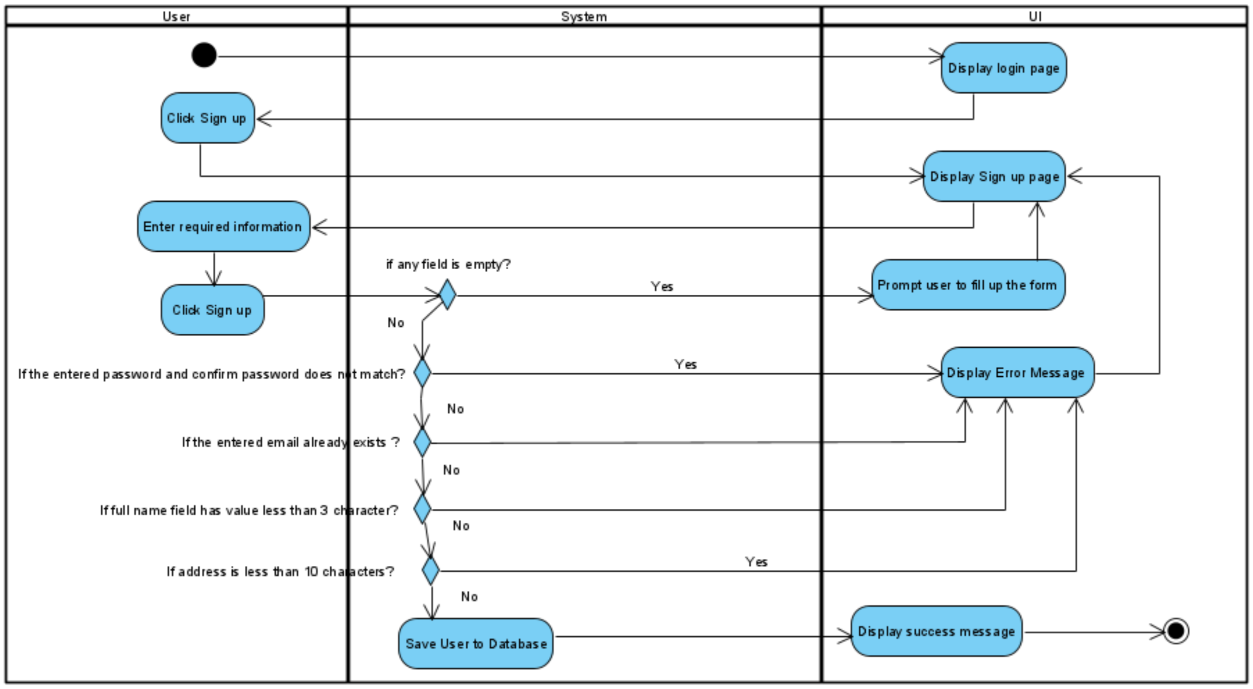

User
System
UI
Display login page
Click Sign up
Display Sign up page
Enter required information
if any field is empty?
Yes
Prompt user to fill up the form
Click Sign up
No
If the entered password and confirm password does not match?
Yes
Display Error Message
No
If the entered email already exists ?
No
If full name field has value less than 3 character?
No
If address is less than 10 characters?
Yes
No
Save User to Database
Display success message

Figure 6: Activity diagram - Register

Login

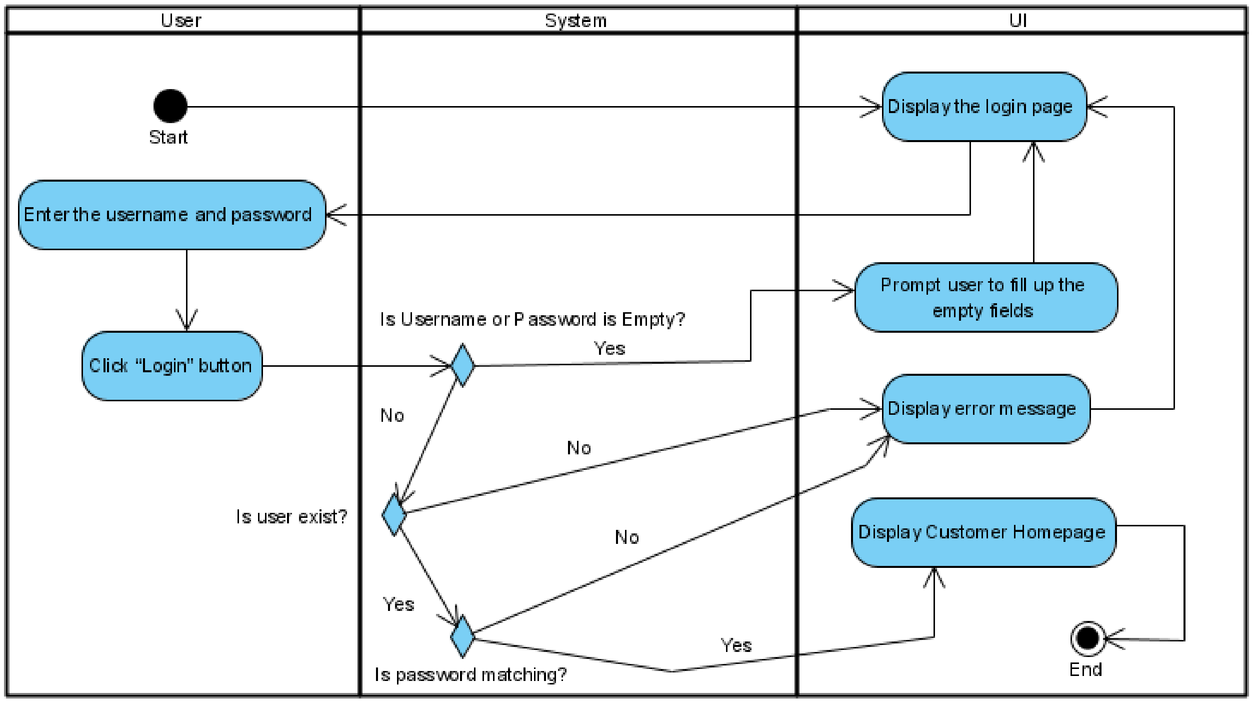


Figure 7:Activity Diagram - Login

Recommend Service Provider Based on user interest

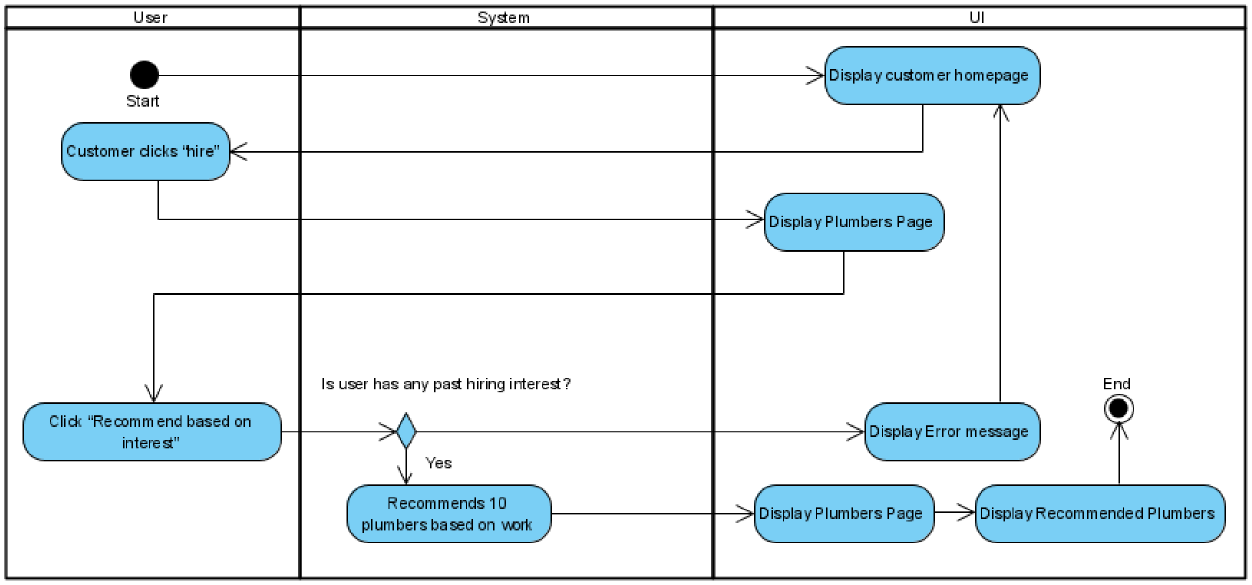

User
System
UI
Start
Display customer homepage
Customer clicks "hire"
Display Plumbers Page
Is user has any past hiring interest?
End
Click "Recommend based on interest"
Display Error message
Yes
Recommends 10 plumbers based on work
Display Plumbers Page
Display Recommended Plumbers

Figure 8:Activity Diagram - Recommend based on user interest

Add Review

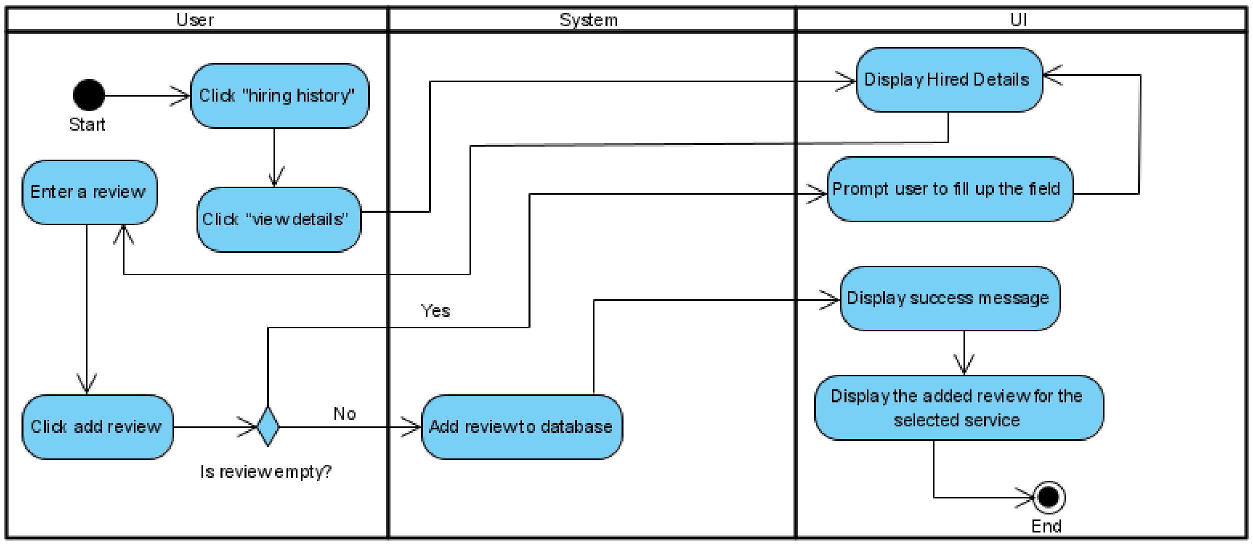


Figure 9:Activity Diagram - Add Review

Update plumber as arrived

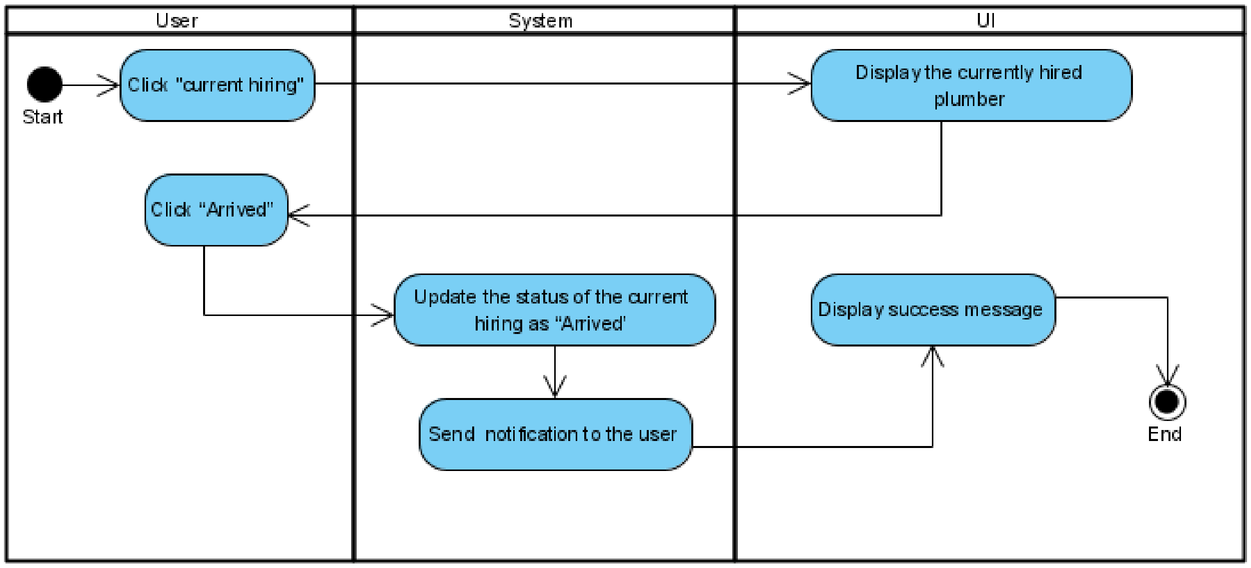

User
System
UI
Start
Click "current hiring"
Display the currently hired plumber
Click "Arrived"
Update the status of the current hiring as "Arrived'
Send notification to the user
Display success message
End

Figure 10: Activity Diagram - UPDATE PLUMBER AS ARRIVED

Update plumber as Work completed

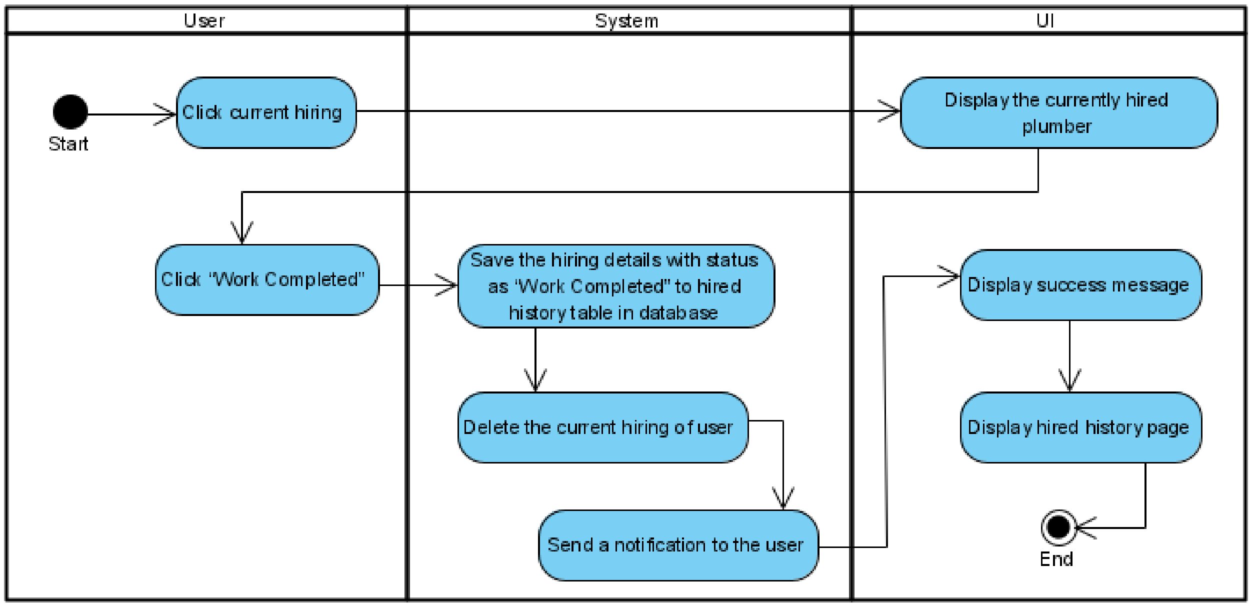

User
System
UI
Start
Click current hiring
Display the currently hired plumber
Click "Work Completed"
Save the hiring details with status as 'Work Completed" to hired history table in database
Display success message
Delete the current hiring of user
Display hired history page
Send a notification to the user
End

Figure 11: Activity Diagram - UPDATE PLUMBER AS WORK COMPLETED

Edit Profile

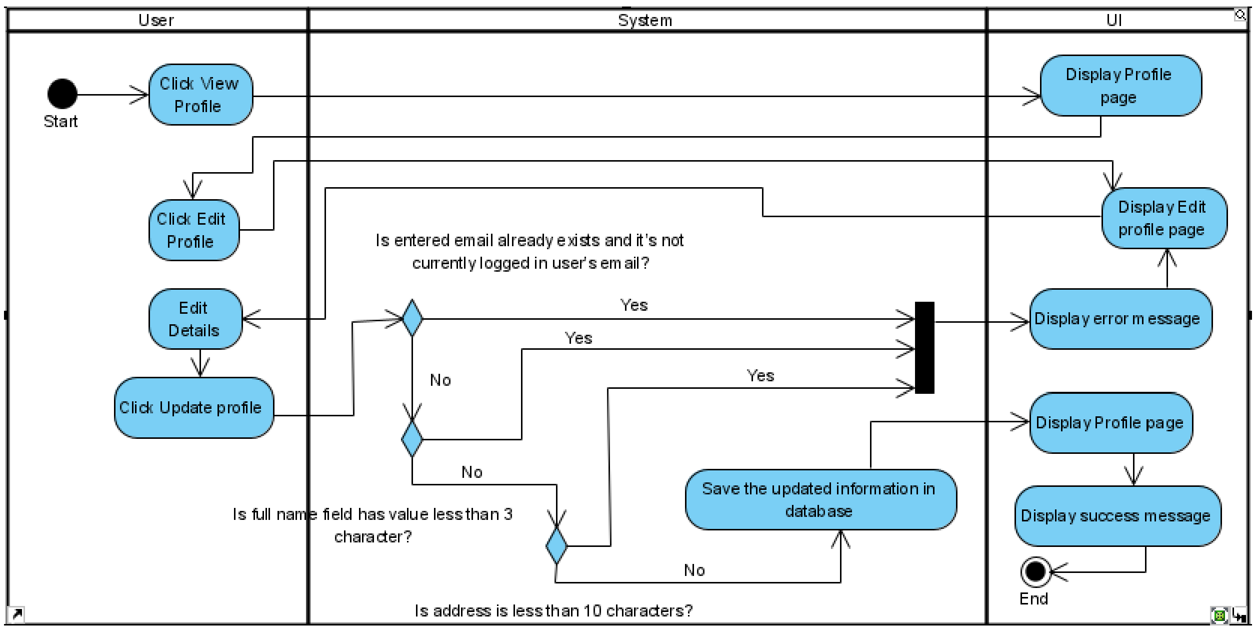

User
System
UI
Start
Click View Profile
Display Profile page
Click Edit Profile
Display Edit profile page
Is entered email already exists and it's not currently logged in user's email?
Edit Details
Yes
Yes
No
Yes
Display error message
Click Update profile
Display Profile page
No
Save the updated information in database
Is full name field has value less than 3 character?
Display success message
No
End
Is address is less than 10 characters?

Figure 12: Activity Diagram - EDIT PROFILE

06.5. Sequence Diagram

Register

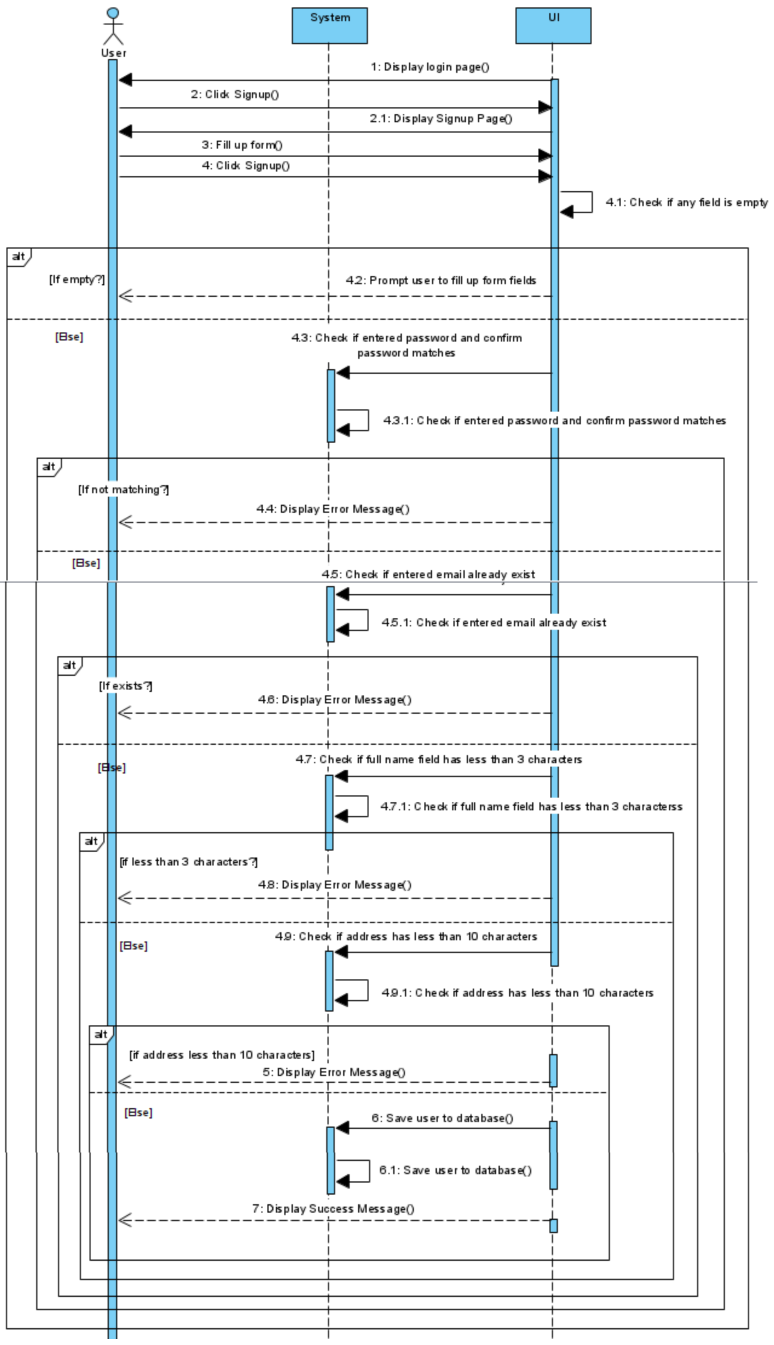

User
System
UI
1: Display login page()
2: Click Signup()
2.1: Display Signup Page()
3: Fill up form()
4: Click Signup()
4.1: Check if any field is empty
alt
[If empty?]
4.2: Prompt user to fill up form fields
[Else]
4.3: Check if entered password and confirm password matches
4.3.1: Check if entered password and confirm password matches
alt
[If not matching?]
4.4: Display Error Message()
[Else]
4.5: Check if entered email already exist
4.5.1: Check if entered email already exist
alt
[If exists?]
4.6: Display Error Message()
[Else]
4.7: Check if full name field has less than 3 characters
4.7.1: Check if full name field has less than 3 characterss
alt
[if less than 3 characters?]
4.8: Display Error Message()
[Else]
4.9: Check if address has less than 10 characters
4.9.1: Check if address has less than 10 characters
alt
[if address less than 10 characters]
5: Display Error Message()
[Else]
6: Save user to database()
6.1: Save user to database()
7: Display Success Message()

Figure 13: Sequence Diagram - Register

Login

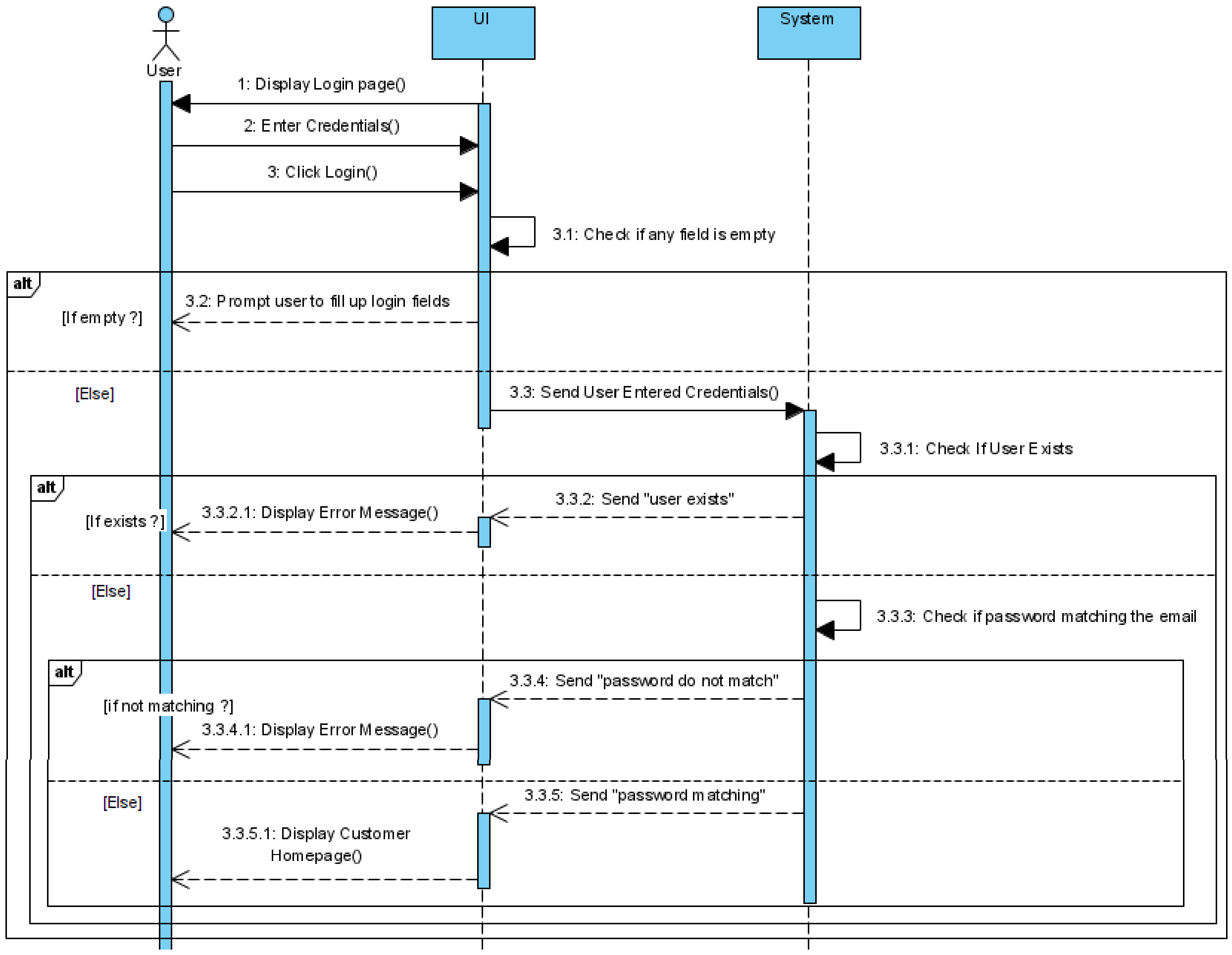

User
UI
System
1: Display Login page()
2: Enter Credentials()
3: Click Login()
3.1: Check if any field is empty
alt
[If empty ?]
3.2: Prompt user to fill up login fields
[Else]
3.3: Send User Entered Credentials()
3.3.1: Check If User Exists
alt
[If exists ?]
3.3.2: Send "user exists"
3.3.2.1: Display Error Message()
[Else]
3.3.3: Check if password matching the email
alt
[if not matching ?]
3.3.4: Send "password do not match"
3.3.4.1: Display Error Message()
[Else]
3.3.5: Send "password matching"
3.3.5.1: Display Customer Homepage()

Figure 14: Sequence Diagram - Login

Recommend Service Provider Based on user interest

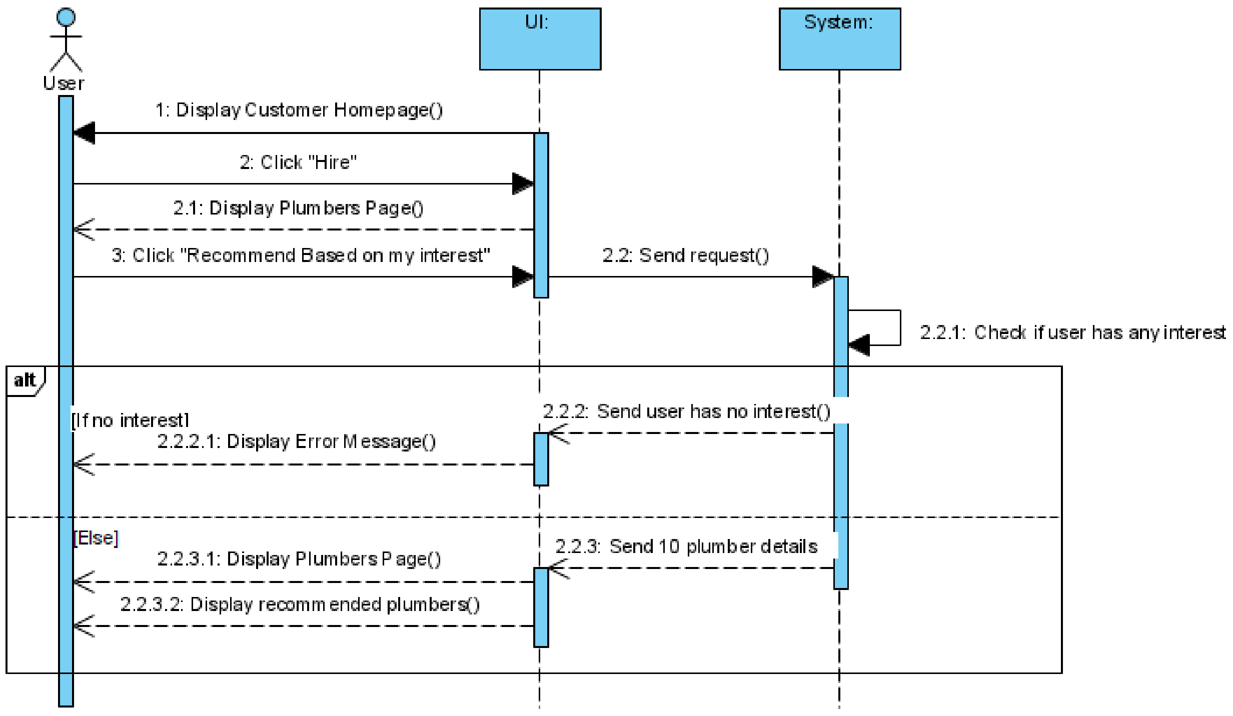

User
UI:
System:
1: Display Customer Homepage()
2: Click "Hire"
2.1: Display Plumbers Page()
3: Click "Recommend Based on my interest"
2.2: Send request()
2.2.1: Check if user has any interest
alt
[If no interest]
2.2.2: Send user has no interest()
2.2.2.1: Display Error Message()
[Else]
2.2.3: Send 10 plumber details
2.2.3.1: Display Plumbers Page()
2.2.3.2: Display recommended plumbers()

Figure 15: Sequence Diagram - Recommend Service Provider Based on user interest

Add Review

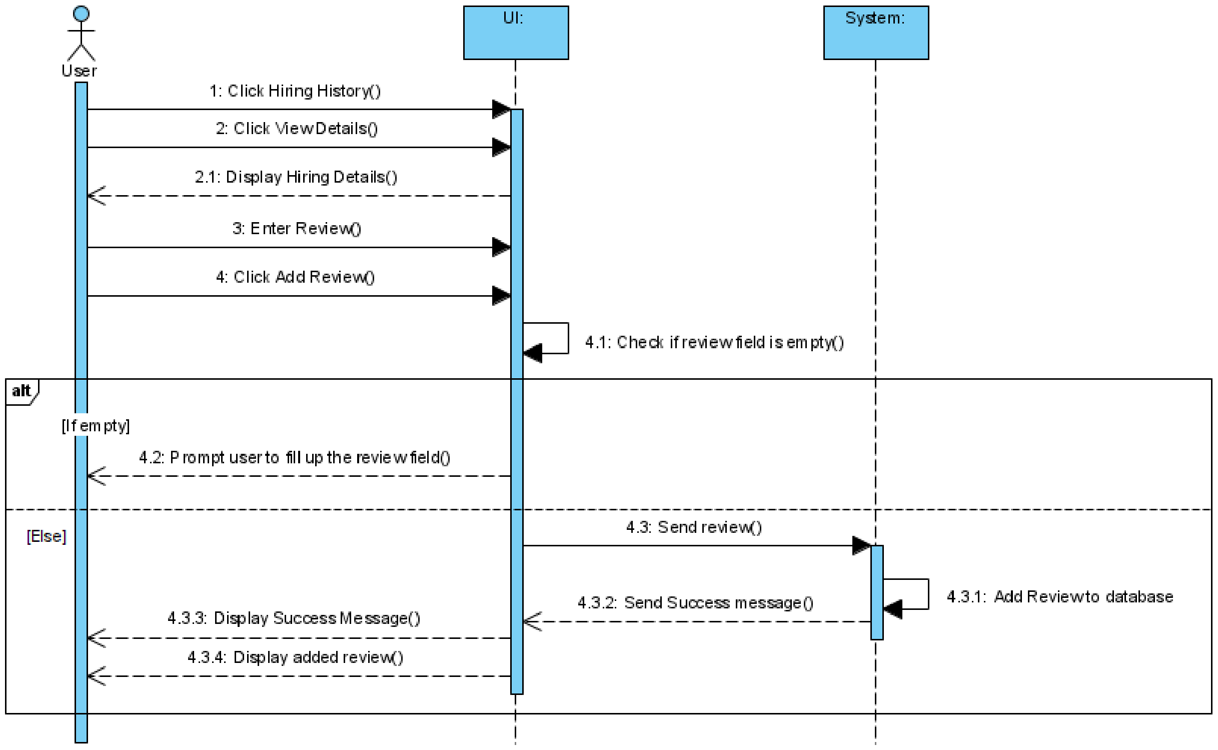

User
UI:
System:
1: Click Hiring History()
2: Click ViewDetails()
2.1: Display Hiring Details()
3: Enter Review()
4: Click Add Review()
4.1: Check if review field is empty()
alt
[If empty]
4.2: Prompt user to fill up the review field()
[Else]
4.3: Send review()
4.3.1: Add Review to database
4.3.2: Send Success message()
4.3.3: Display Success Message()
4.3.4: Display added review()

Figure 16: Sequence Diagram - Add review

Update plumber as arrived

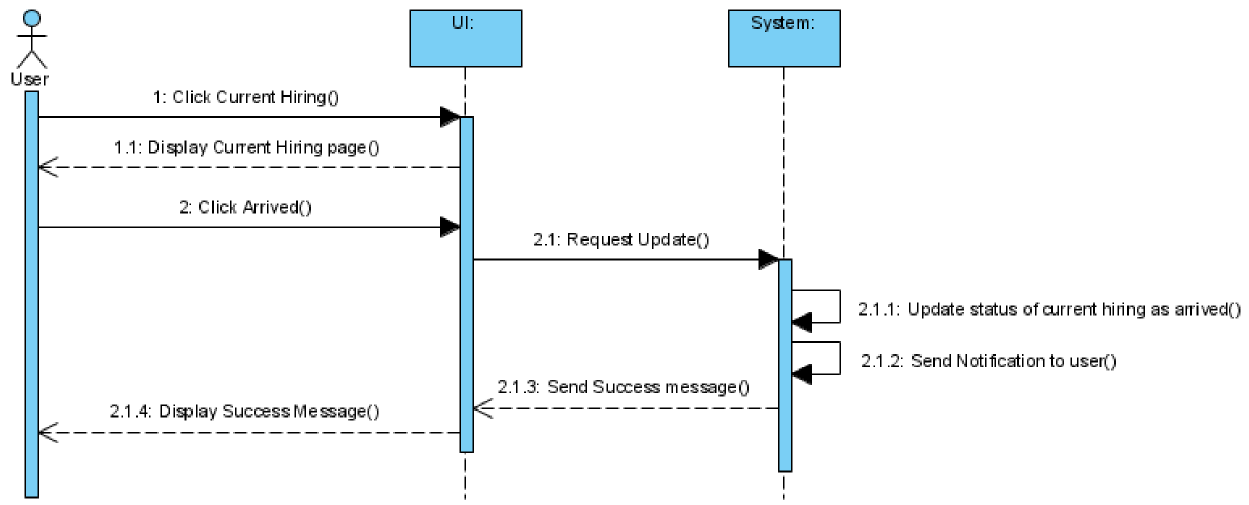

User
UI:
System:
1: Click Current Hiring()
1.1: Display Current Hiring page()
2: Click Arrived()
2.1: Request Update()
2.1.1: Update status of current hiring as arrived()
2.1.2: Send Notification to user()
2.1.3: Send Success message()
2.1.4: Display Success Message()

Figure 17: Sequence Diagram - Update Plumber as arrived

Update plumber as work completed

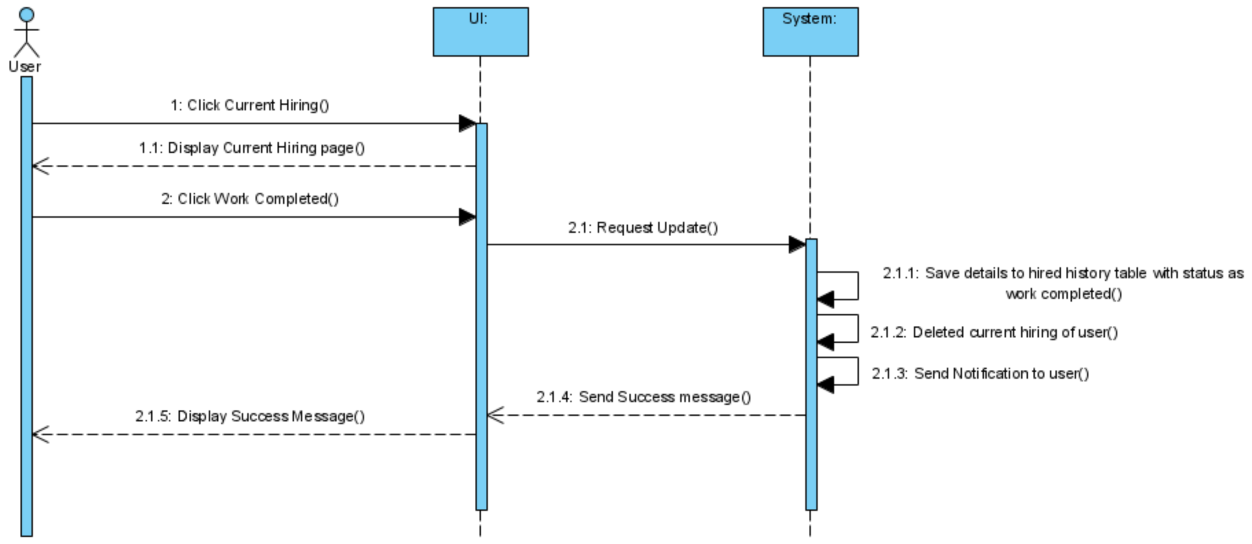

User
UI:
System:
1: Click Current Hiring()
1.1: Display Current Hiring page()
2: Click Work Completed()
2.1: Request Update()
2.1.1: Save details to hired history table with status as work completed()
2.1.2: Deleted current hiring of user()
2.1.3: Send Notification to user()
2.1.4: Send Success message()
2.1.5: Display Success Message()

Figure 18: Sequence Diagram - Update plumber as work completed

Edit Profile

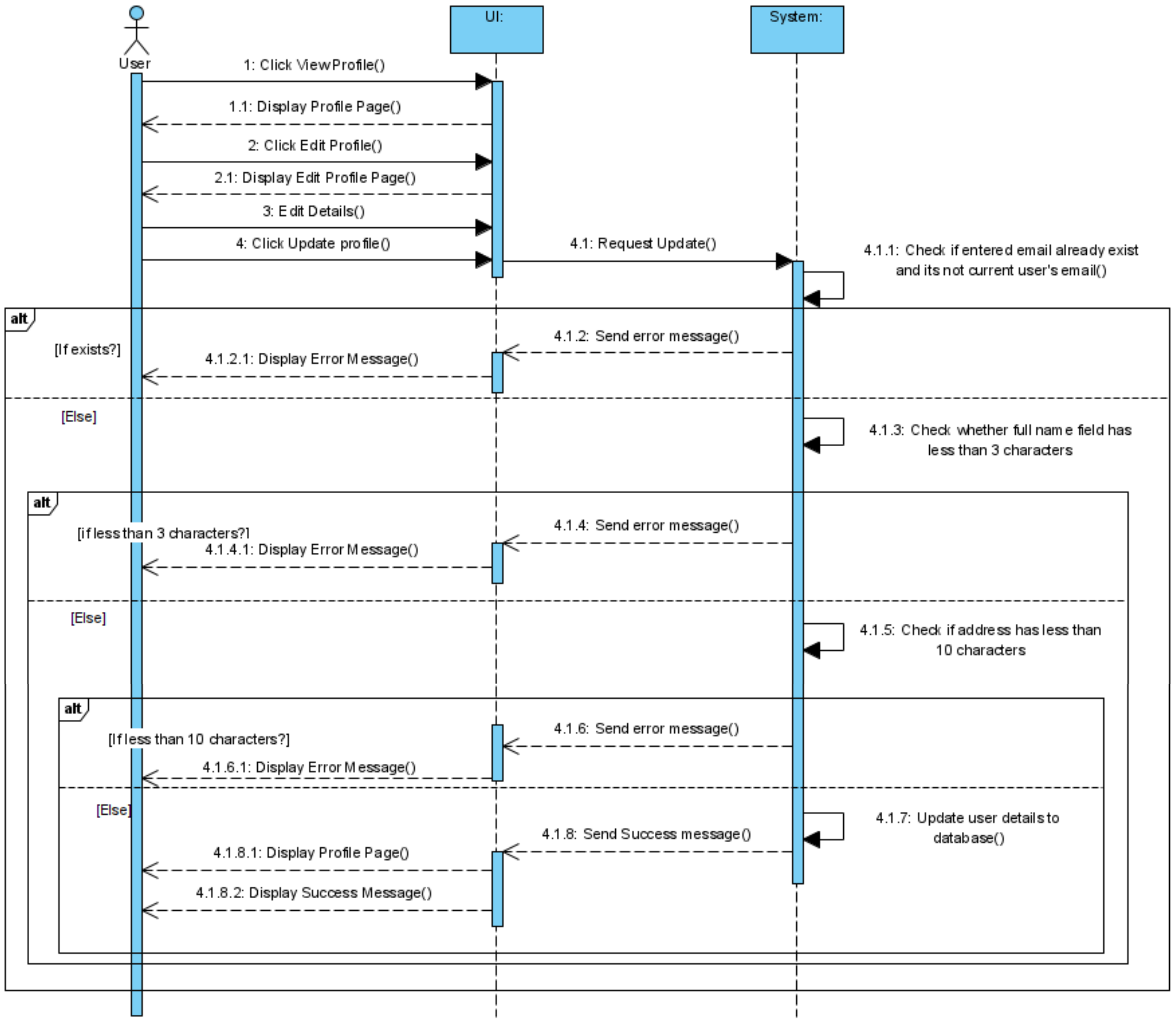

User
UI:
System:
1: Click ViewProfile()
1.1: Display Profile Page()
2: Click Edit Profile()
2.1: Display Edit Profile Page()
3: Edit Details()
4: Click Update profile()
4.1: Request Update()
4.1.1: Check if entered email already exist and its not current user's email()
alt
[If exists?]
4.1.2: Send error message()
4.1.2.1: Display Error Message()
[Else]
4.1.3: Check whether full name field has less than 3 characters
alt
[if less than 3 characters?]
4.1.4: Send error message()
4.1.4.1: Display Error Message()
[Else]
4.1.5: Check if address has less than 10 characters
alt
[If less than 10 characters?]
4.1.6: Send error message()
4.1.6.1: Display Error Message()
[Else]
4.1.7: Update user details to database()
4.1.8: Send Success message()
4.1.8.1: Display Profile Page()
4.1.8.2: Display Success Message()

Figure 19: Sequence Diagram - Edit Profile

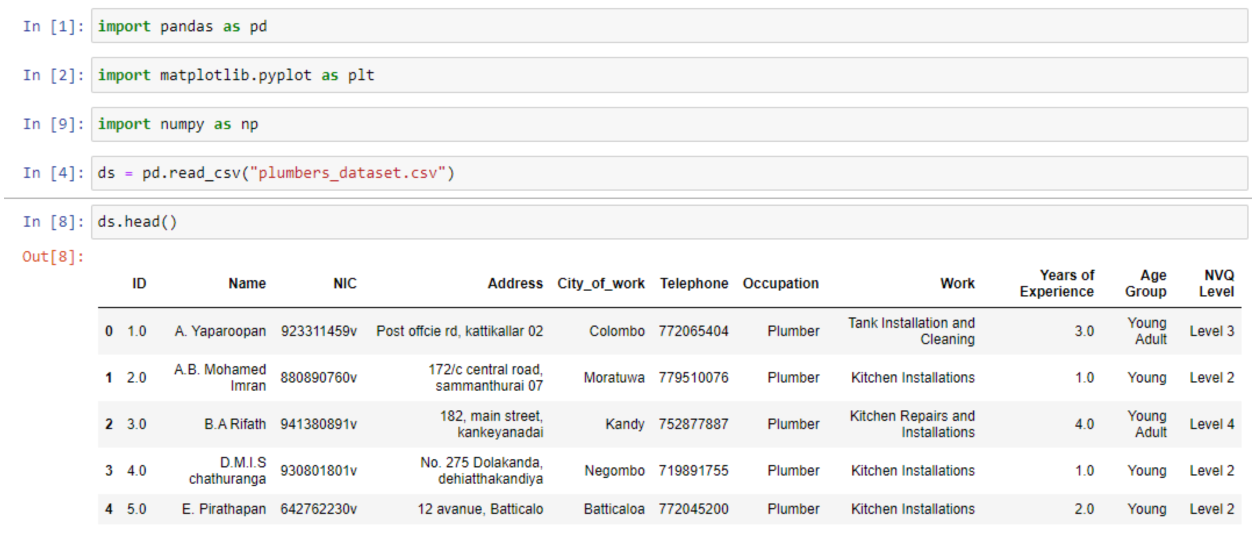
07. Implementation

This Web Application's back end will be based on the Python programming language and the Flask Framework, as mentioned in the Methodology. This web application's front end will be built using HTML5, Booostrap5, CSS, and JavaScript.

07.1. Recommendation Engine

The recommendation engine was developed and tested using Jupyter notebook. Below is the recommendation engine developed and tested in Jupyter notebook for recommending plumbers based on work.

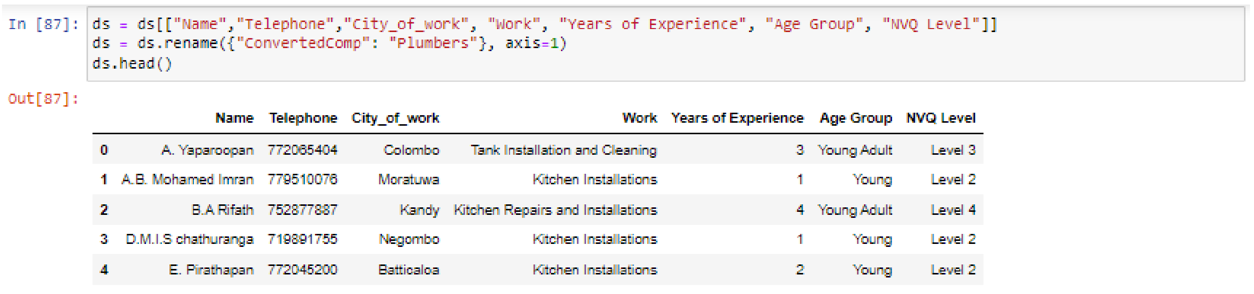
Loading dataset



```
In [1]: import pandas as pd

In [2]: import matplotlib.pyplot as plt

In [9]: import numpy as np

In [4]: ds = pd.read_csv("plumbers_dataset.csv")

In [8]: ds.head()
Out[8]:
```

	ID	Name	NIC	Address	City_of_work	Telephone	Occupation	Work	Years of Experience	Age Group	NVQ Level
0	1.0	A. Yaparoopan	923311459v	Post offcie rd, kattikallar 02	Colombo	772065404	Plumber	Tank Installation and Cleaning	3.0	Young Adult	Level 3
1	2.0	A.B. Mohamed Imran	880890760v	172/c central road, sammanthurai 07	Moratuwa	779510076	Plumber	Kitchen Installations	1.0	Young	Level 2
2	3.0	B.A Rifath	941380891v	182, main street, kankeyanadai	Kandy	752877887	Plumber	Kitchen Repairs and Installations	4.0	Young Adult	Level 4
3	4.0	D.M.I.S chathuranga	930801801v	No. 275 Dolakanda, dehiatthakandiya	Negombo	719891755	Plumber	Kitchen Installations	1.0	Young	Level 2
4	5.0	E. Pirathapan	642762230v	12 avanue, Batticalo	Batticaloa	772045200	Plumber	Kitchen Installations	2.0	Young	Level 2

Dropping unwanted columns for training



```
In [87]: ds = ds[["Name","Telephone","City_of_work", "Work", "Years of Experience", "Age Group", "NVQ Level"]]
         ds = ds.rename({"ConvertedComp": "Plumbers"}, axis=1)
         ds.head()
Out[87]:
```

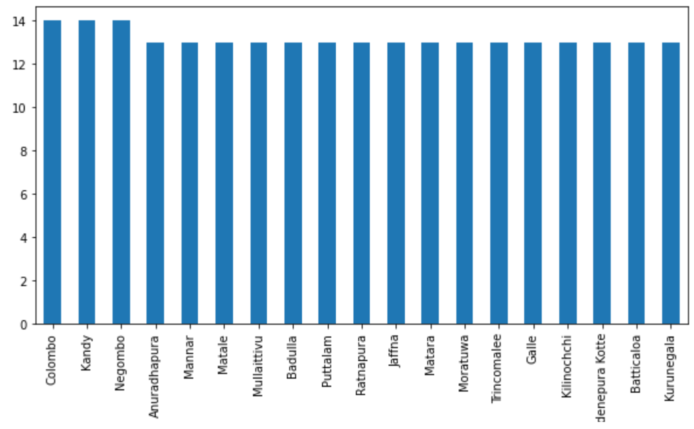
	Name	Telephone	City_of_work	Work	Years of Experience	Age Group	NVQ Level
0	A. Yaparoopan	772065404	Colombo	Tank Installation and Cleaning	3	Young Adult	Level 3
1	A.B. Mohamed Imran	779510076	Moratuwa	Kitchen Installations	1	Young	Level 2
2	B.A Rifath	752877887	Kandy	Kitchen Repairs and Installations	4	Young Adult	Level 4
3	D.M.I.S chathuranga	719891755	Negombo	Kitchen Installations	1	Young	Level 2
4	E. Pirathapan	772045200	Batticaloa	Kitchen Installations	2	Young	Level 2

Checking for null values and dropping it

```
In [48]: ds.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 250 entries, 0 to 249
         Data columns (total 5 columns):
          #   Column               Non-Null Count  Dtype
         ---  ------               --------------  -----
          0   City_of_work         250 non-null    object
          1   Work                 250 non-null    object
          2   Years of Experience  250 non-null    int64
          3   Age Group            250 non-null    object
          4   NVQ Level            250 non-null    object
         dtypes: int64(1), object(4)
         memory usage: 9.9+ KB

In [49]: ds = ds.dropna()
         ds.isnull().sum()

Out[49]: City_of_work           0
         Work                   0
         Years of Experience    0
         Age Group              0
         NVQ Level              0
         dtype: int64
```

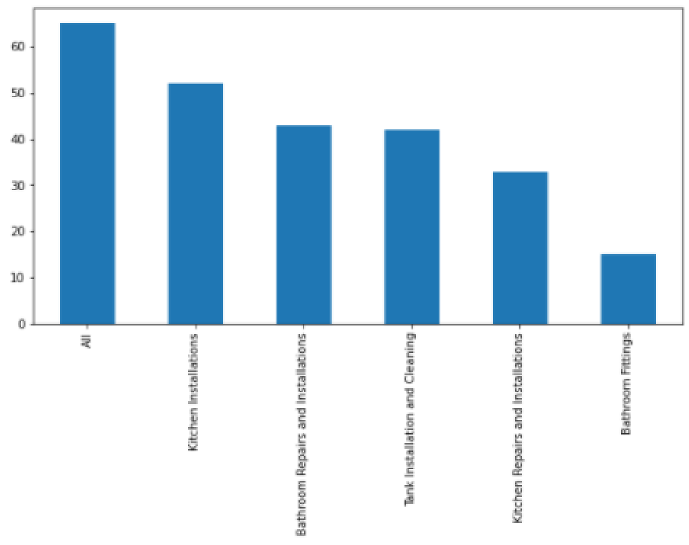
Visual representation of “City_of_work” column with count

```
In [58]: ds['City_of_work'].value_counts().plot(x = 'City_of_work', y ='count', kind = 'bar', figsize = (10,5)  )

Out[58]: <AxesSubplot:>
```



Visual representation of “work” column with count

```
In [59]: ds['Work'].value_counts().plot(x = 'Work', y ='count', kind = 'bar', figsize = (10,5)  )
Out[59]: <AxesSubplot:>
```

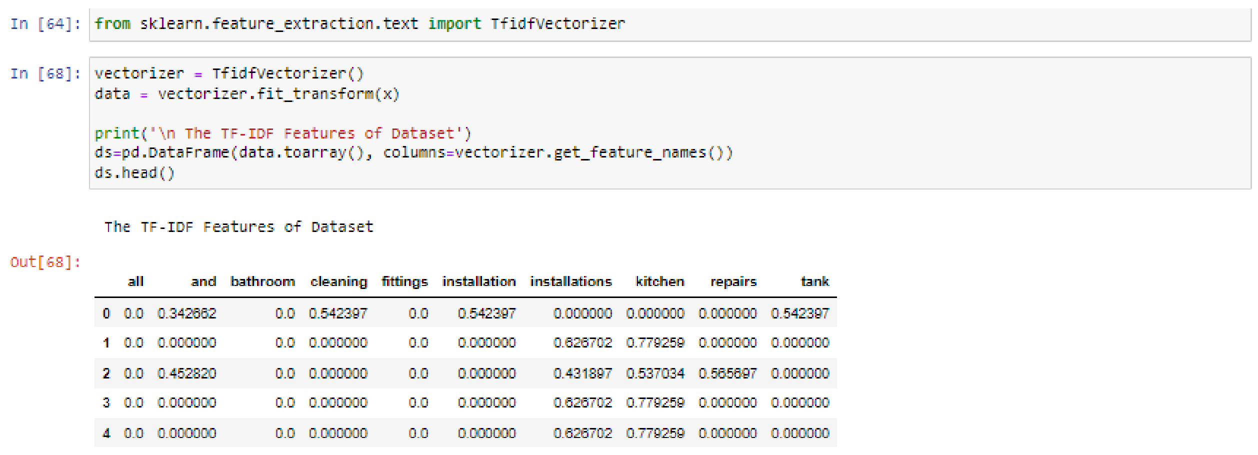


Text preprocessing to remove non-ASCII characters and stop words from the work column and convert it into lower case to avoid duplication while finding similarities

```
#Data preprocessing

# Function for removing NonAscii characters
def _removeNonAscii(s):
    return "".join(i for i in s if  ord(i)<128)

# Function for converting into lower case
def make_lower_case(text):
    return text.lower()

# Function for removing stop words(eg:'is','the', etc..)
def remove_stop_words(text):
    text = text.split()
    stops = set(stopwords.words("english"))
    text = [w for w in text if not w in stops]
    text = " ".join(text)
    return text

# Applying all the functions in description and storing as a cleaned_desc
ds['cleaned_work'] = ds['Work'].apply(_removeNonAscii)
ds['cleaned_work'] = ds.cleaned_work.apply(func = make_lower_case)
ds['cleaned_work'] = ds.cleaned_work.apply(func = remove_stop_words)
```

TF-IDF vectorizer is used to convert words in the “work” column into vectors

Below code will display the data frame of each word used in the “work” column and their TF-IDF values will be displayed



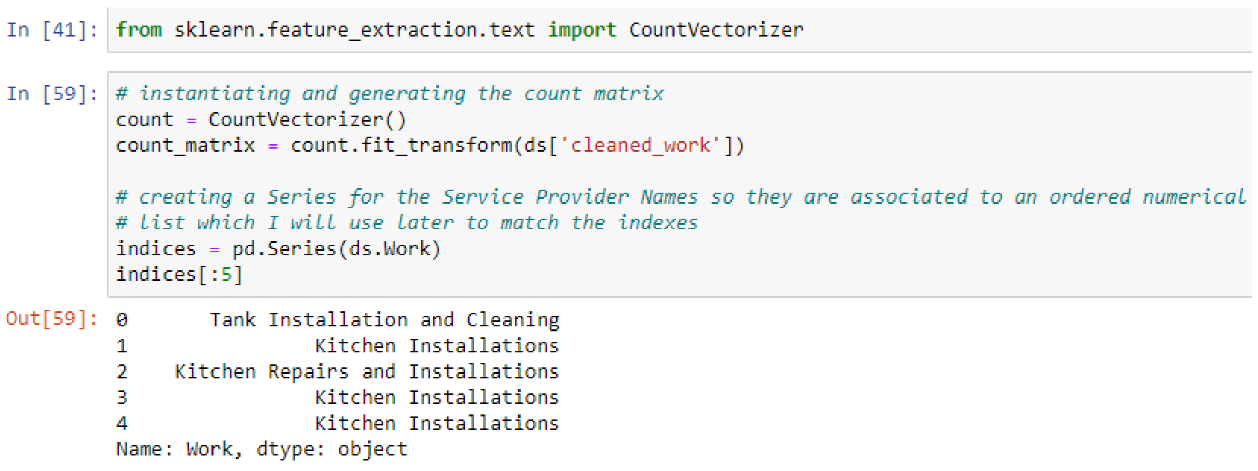
```
In [64]: from sklearn.feature_extraction.text import TfidfVectorizer
```

```
In [68]: vectorizer = TfidfVectorizer()
         data = vectorizer.fit_transform(x)

         print('\n The TF-IDF Features of Dataset')
         ds=pd.DataFrame(data.toarray(), columns=vectorizer.get_feature_names())
         ds.head()
```

```
          The TF-IDF Features of Dataset
Out[68]:
```

	all	and	bathroom	cleaning	fittings	installation	installations	kitchen	repairs	tank
0	0.0	0.342682	0.0	0.542397	0.0	0.542397	0.000000	0.000000	0.000000	0.542397
1	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.626702	0.779259	0.000000	0.000000
2	0.0	0.452820	0.0	0.000000	0.0	0.000000	0.431897	0.537034	0.565697	0.000000
3	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.626702	0.779259	0.000000	0.000000
4	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.626702	0.779259	0.000000	0.000000

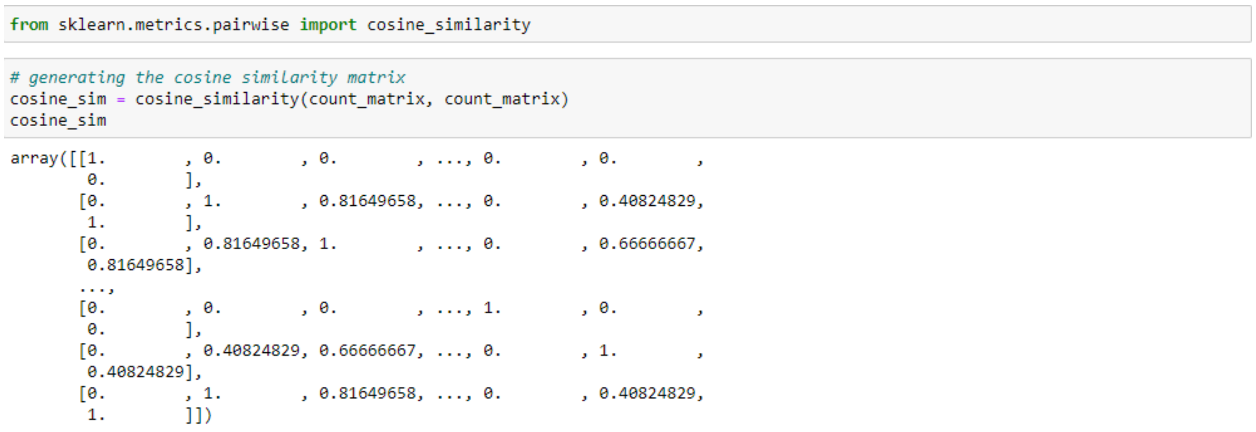
We are trying to find similarities amongst service providers based on “work” column. We’ll need a basic frequency counter for each word in my “cleaned_work” column (text preprocessed column) to accomplish this. TF-IDF tends to downplay the importance of words that are more prevalent across the corpus. Because we have a limited number of words in our work column, each word is equally essential. As a result, we choose to employ the "Count Vectorizer."



```
In [41]: from sklearn.feature_extraction.text import CountVectorizer
```

```
In [59]: # instantiating and generating the count matrix
         count = CountVectorizer()
         count_matrix = count.fit_transform(ds['cleaned_work'])

         # creating a Series for the Service Provider Names so they are associated to an ordered numerical
         # list which I will use later to match the indexes
         indices = pd.Series(ds.Work)
         indices[:5]

Out[59]: 0       Tank Installation and Cleaning
         1                 Kitchen Installations
         2    Kitchen Repairs and Installations
         3                 Kitchen Installations
         4                 Kitchen Installations
         Name: Work, dtype: object
```

We will be generating the similarity of the “cleaned_work” column across the dataset. For this, we will be using Cosine Similarity algorithm.

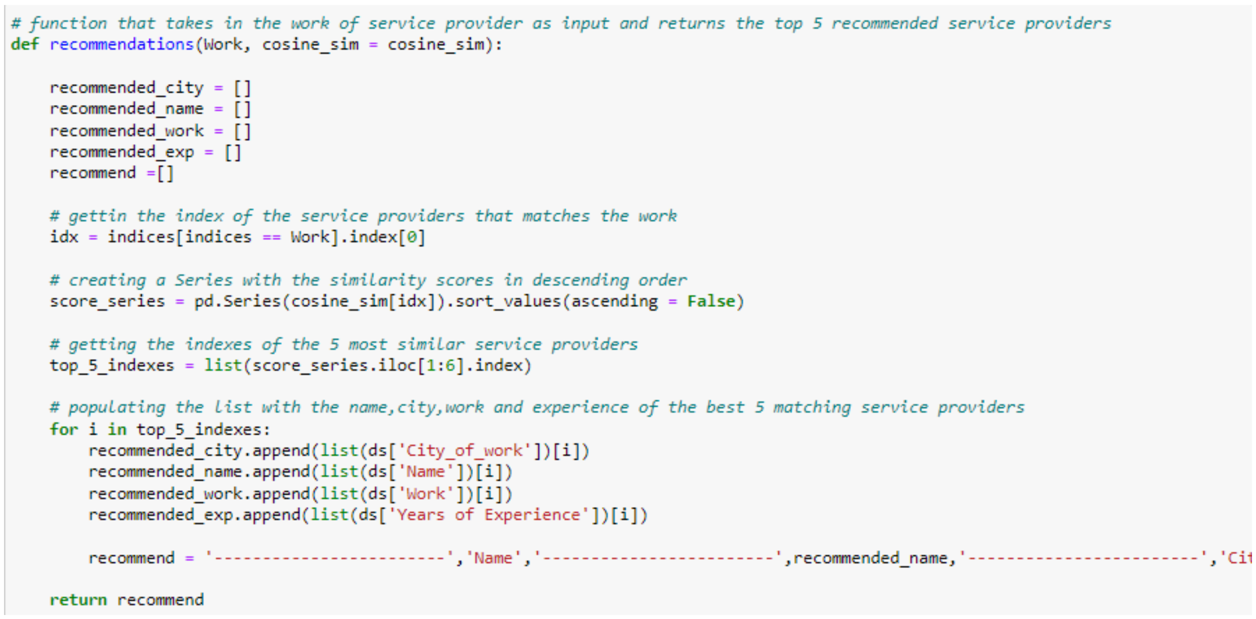


```
from sklearn.metrics.pairwise import cosine_similarity
```

```
# generating the cosine similarity matrix
cosine_sim = cosine_similarity(count_matrix, count_matrix)
cosine_sim
```

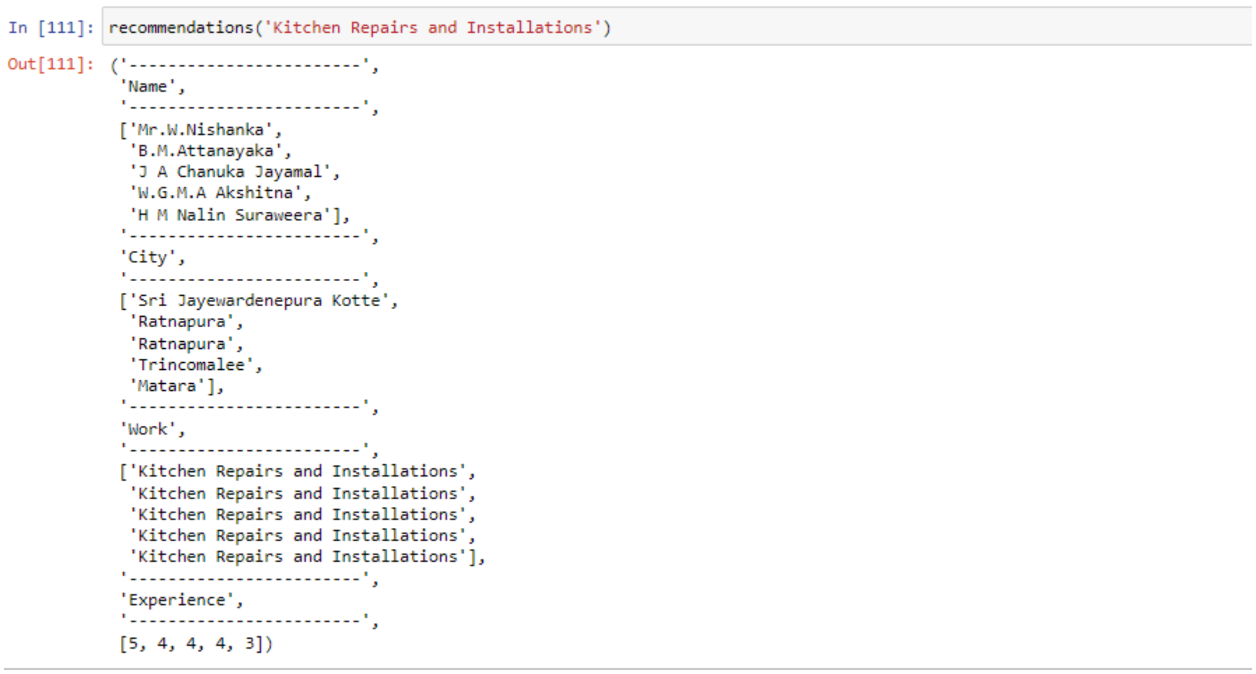
```
array([[1.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 1.        , 0.81649658, ..., 0.        , 0.40824829,
        1.        ],
       [0.        , 0.81649658, 1.        , ..., 0.        , 0.66666667,
        0.81649658],
       ...,
       [0.        , 0.        , 0.        , ..., 1.        , 0.        ,
        0.        ],
       [0.        , 0.40824829, 0.66666667, ..., 0.        , 1.        ,
        0.40824829],
       [0.        , 1.        , 0.81649658, ..., 0.        , 0.40824829,
        1.        ]])
```

Now we will be implementing a function to recommend top 5 service providers by taking work as input and name, city, work, and experience as output.



```
# function that takes in the work of service provider as input and returns the top 5 recommended service providers
def recommendations(Work, cosine_sim = cosine_sim):

    recommended_city = []
    recommended_name = []
    recommended_work = []
    recommended_exp = []
    recommend =[]

    # gettin the index of the service providers that matches the work
    idx = indices[indices == Work].index[0]

    # creating a Series with the similarity scores in descending order
    score_series = pd.Series(cosine_sim[idx]).sort_values(ascending = False)

    # getting the indexes of the 5 most similar service providers
    top_5_indexes = list(score_series.iloc[1:6].index)

    # populating the list with the name,city,work and experience of the best 5 matching service providers
    for i in top_5_indexes:
        recommended_city.append(list(ds['City_of_work'])[i])
        recommended_name.append(list(ds['Name'])[i])
        recommended_work.append(list(ds['Work'])[i])
        recommended_exp.append(list(ds['Years of Experience'])[i])

        recommend = '-----------------------','Name','-----------------------',recommended_name,'-----------------------','Cit

    return recommend
```

Now for testing purpose we will be giving an input manually



```
In [111]: recommendations('Kitchen Repairs and Installations')

Out[111]: ('------------------------',
           'Name',
           '------------------------',
           ['Mr.W.Nishanka',
            'B.M.Attanayaka',
            'J A Chanuka Jayamal',
            'W.G.M.A Akshitna',
            'H M Nalin Suraweera'],
           '------------------------',
           'City',
           '------------------------',
           ['Sri Jayewardenepura Kotte',
            'Ratnapura',
            'Ratnapura',
            'Trincomalee',
            'Matara'],
           '------------------------',
           'Work',
           '------------------------',
           ['Kitchen Repairs and Installations',
            'Kitchen Repairs and Installations',
            'Kitchen Repairs and Installations',
            'Kitchen Repairs and Installations',
            'Kitchen Repairs and Installations'],
           '------------------------',
           'Experience',
           '------------------------',
           [5, 4, 4, 4, 3])
```

07.2. Web Application

For the web application, we have included several functions along with the core functionality of this project.

The web application is developed in the manually created python package "website," which includes a static folder for storing static files like stylesheets and JavaScript, a template folder for storing HTML codes for the front end, a database file for storing application data, and four python files for init, auth, models, and views.

07.2.1. Auth file

This file handles the functionalities such as registering, logging in, and logging out.

07.2.2. Models

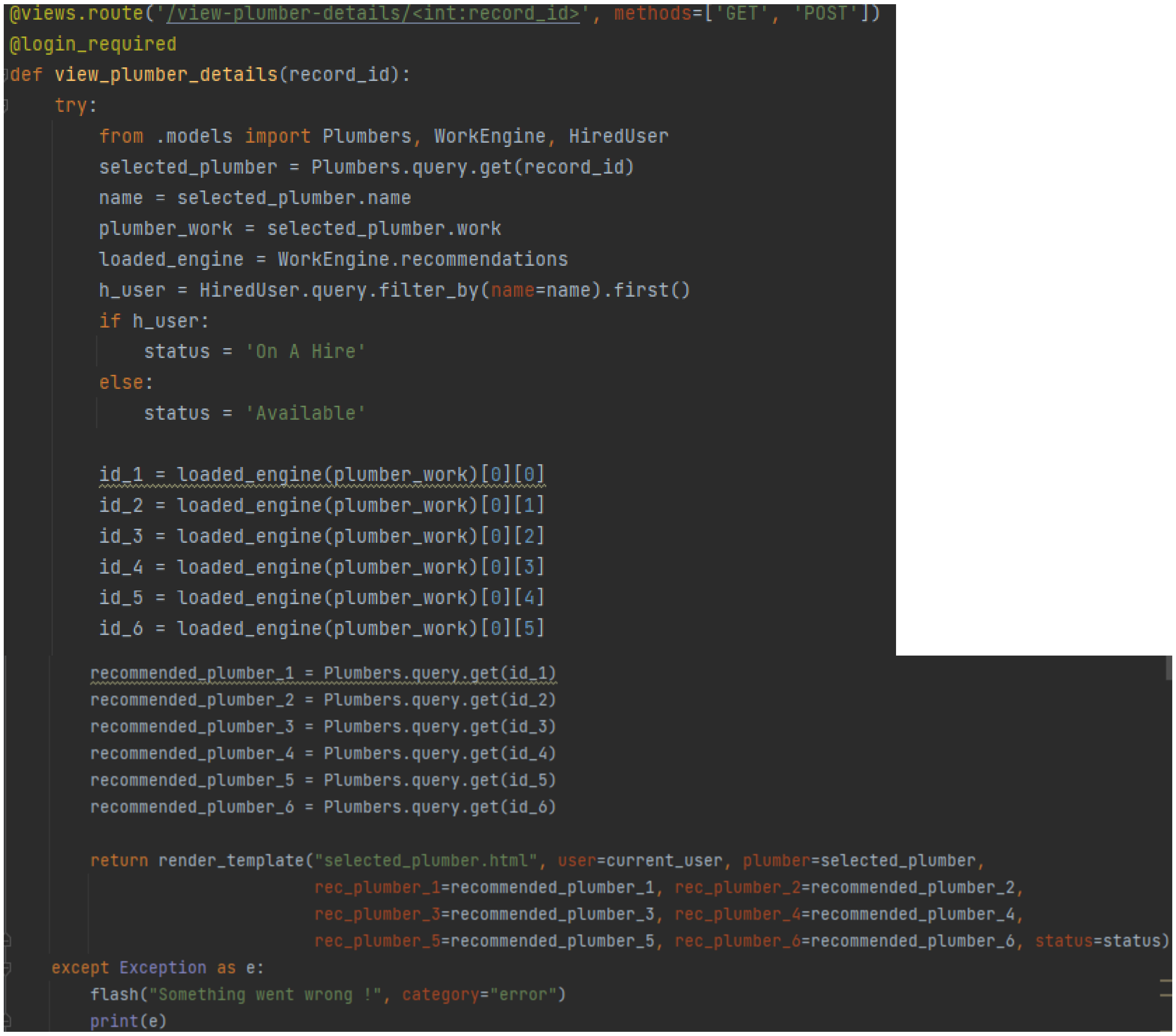
This file contains the code for Python to understand the structure of the database tables, as well as the code for the application's two recommendation engines: one that recommends based on the plumber's city, and the other that recommends based on the plumber's work.

07.2.3. Views

This file handles all the functionalities of the logged-in user except logout.

07.3. Recommendation function

07.3.1. Recommendation based on selected plumber's work

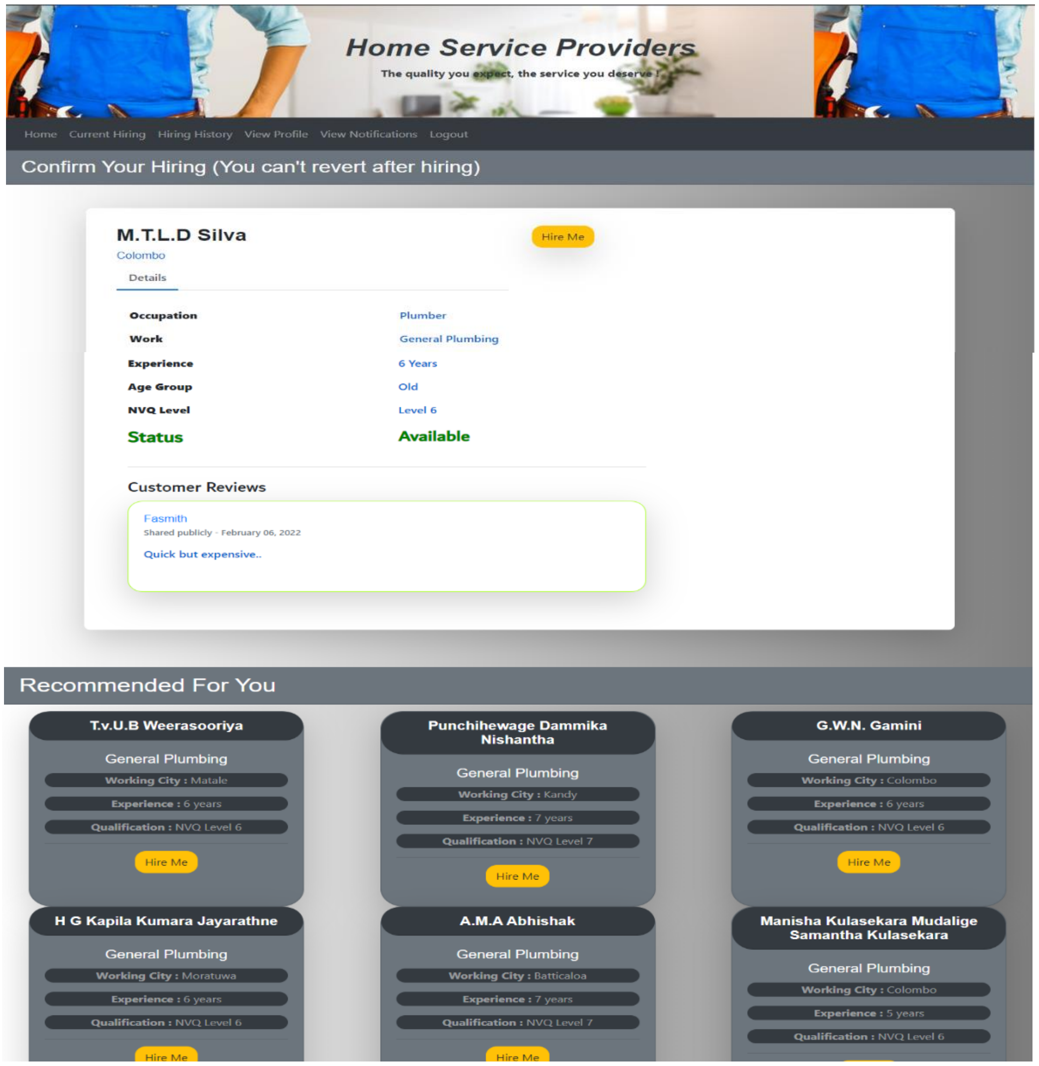


```python
@views.route('/view-plumber-details/<int:record_id>', methods=['GET', 'POST'])
@login_required
def view_plumber_details(record_id):
    try:
        from .models import Plumbers, WorkEngine, HiredUser
        selected_plumber = Plumbers.query.get(record_id)
        name = selected_plumber.name
        plumber_work = selected_plumber.work
        loaded_engine = WorkEngine.recommendations
        h_user = HiredUser.query.filter_by(name=name).first()
        if h_user:
            status = 'On A Hire'
        else:
            status = 'Available'

        id_1 = loaded_engine(plumber_work)[0][0]
        id_2 = loaded_engine(plumber_work)[0][1]
        id_3 = loaded_engine(plumber_work)[0][2]
        id_4 = loaded_engine(plumber_work)[0][3]
        id_5 = loaded_engine(plumber_work)[0][4]
        id_6 = loaded_engine(plumber_work)[0][5]

        recommended_plumber_1 = Plumbers.query.get(id_1)
        recommended_plumber_2 = Plumbers.query.get(id_2)
        recommended_plumber_3 = Plumbers.query.get(id_3)
        recommended_plumber_4 = Plumbers.query.get(id_4)
        recommended_plumber_5 = Plumbers.query.get(id_5)
        recommended_plumber_6 = Plumbers.query.get(id_6)

        return render_template("selected_plumber.html", user=current_user, plumber=selected_plumber,
                               rec_plumber_1=recommended_plumber_1, rec_plumber_2=recommended_plumber_2,
                               rec_plumber_3=recommended_plumber_3, rec_plumber_4=recommended_plumber_4,
                               rec_plumber_5=recommended_plumber_5, rec_plumber_6=recommended_plumber_6, status=status)
    except Exception as e:
        flash("Something went wrong !", category="error")
        print(e)
```

Figure 20: Recommendation based on selected plumber's work

This function is responsible for viewing the details of the selected plumber as well as checking the plumber's availability. This function will also check the work of the selected plumber and pass it to the recommendation function of the work engine class, which was imported from the model file, to find 6 best-matched plumbers who do work that is similar to the work of the selected plumber.

07.3.1.1. Output



Confirm Your Hiring (You can't revert after hiring)

M.T.L.D Silva
Colombo

Occupation	Plumber
Work	General Plumbing
Experience	6 Years
Age Group	Old
NVQ Level	Level 6
Status	Available

Customer Reviews

Fasmith
Shared publicly - February 06, 2022
Quick but expensive..

Recommended For You

T.v.U.B Weerasooriya
General Plumbing
Working City : Matale
Experience : 6 years
Qualification : NVQ Level 6

Punchihewage Dammika Nishantha
General Plumbing
Working City : Kandy
Experience : 7 years
Qualification : NVQ Level 7

G.W.N. Gamini
General Plumbing
Working City : Colombo
Experience : 6 years
Qualification : NVQ Level 6

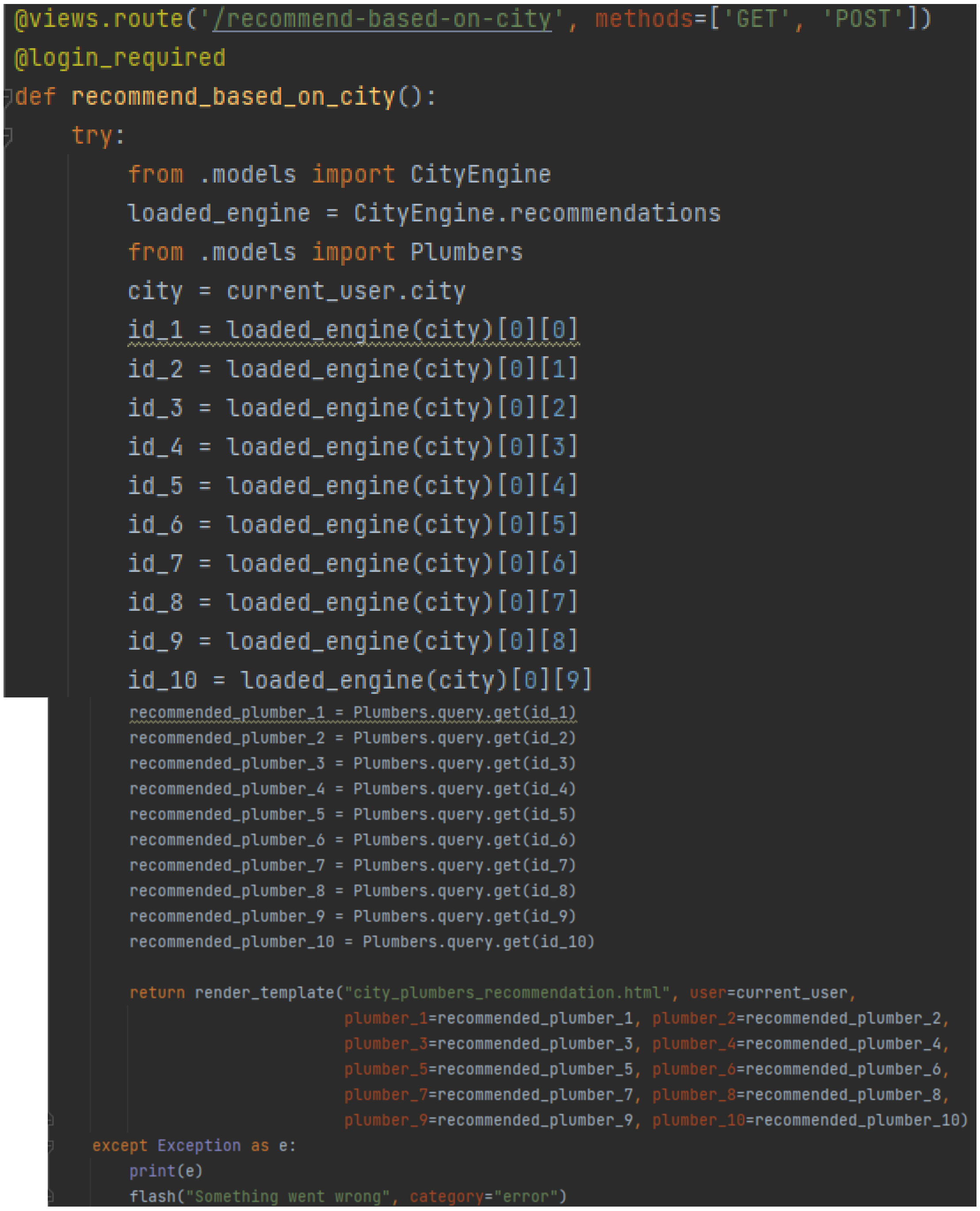
H G Kapila Kumara Jayarathne
General Plumbing
Working City : Moratuwa
Experience : 6 years
Qualification : NVQ Level 6

A.M.A Abhishak
General Plumbing
Working City : Batticaloa
Experience : 7 years
Qualification : NVQ Level 7

Manisha Kulasekara Mudalige Samantha Kulasekara
General Plumbing
Working City : Colombo
Experience : 5 years
Qualification : NVQ Level 6

Figure 21: Output of "recommendation based on selected plumber's work"

07.3.2. **Recommendation based on city where user lives.**

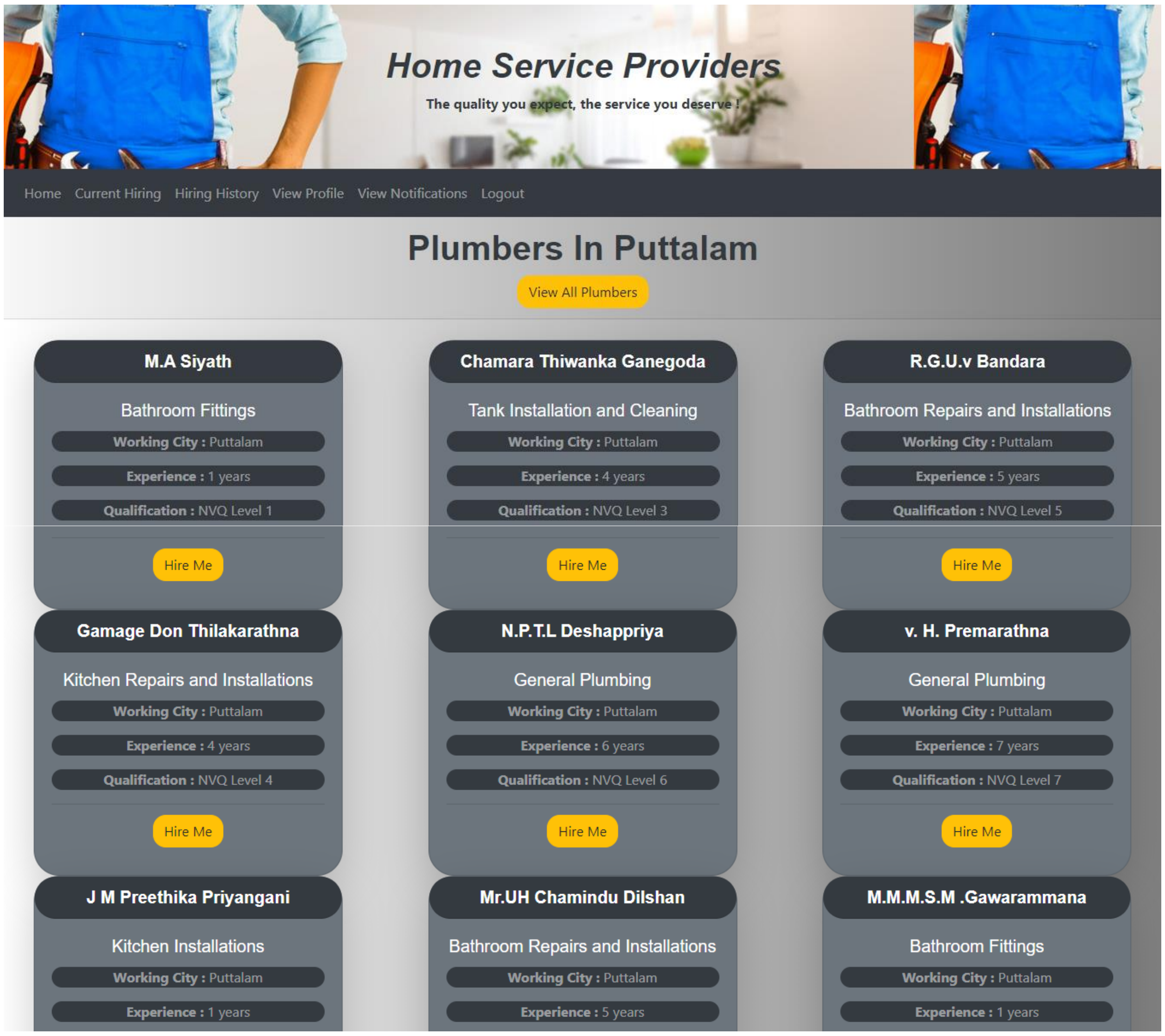


```
@views.route('/recommend-based-on-city', methods=['GET', 'POST'])
@login_required
def recommend_based_on_city():
    try:
        from .models import CityEngine
        loaded_engine = CityEngine.recommendations
        from .models import Plumbers
        city = current_user.city
        id_1 = loaded_engine(city)[0][0]
        id_2 = loaded_engine(city)[0][1]
        id_3 = loaded_engine(city)[0][2]
        id_4 = loaded_engine(city)[0][3]
        id_5 = loaded_engine(city)[0][4]
        id_6 = loaded_engine(city)[0][5]
        id_7 = loaded_engine(city)[0][6]
        id_8 = loaded_engine(city)[0][7]
        id_9 = loaded_engine(city)[0][8]
        id_10 = loaded_engine(city)[0][9]
        recommended_plumber_1 = Plumbers.query.get(id_1)
        recommended_plumber_2 = Plumbers.query.get(id_2)
        recommended_plumber_3 = Plumbers.query.get(id_3)
        recommended_plumber_4 = Plumbers.query.get(id_4)
        recommended_plumber_5 = Plumbers.query.get(id_5)
        recommended_plumber_6 = Plumbers.query.get(id_6)
        recommended_plumber_7 = Plumbers.query.get(id_7)
        recommended_plumber_8 = Plumbers.query.get(id_8)
        recommended_plumber_9 = Plumbers.query.get(id_9)
        recommended_plumber_10 = Plumbers.query.get(id_10)

        return render_template("city_plumbers_recommendation.html", user=current_user,
                               plumber_1=recommended_plumber_1, plumber_2=recommended_plumber_2,
                               plumber_3=recommended_plumber_3, plumber_4=recommended_plumber_4,
                               plumber_5=recommended_plumber_5, plumber_6=recommended_plumber_6,
                               plumber_7=recommended_plumber_7, plumber_8=recommended_plumber_8,
                               plumber_9=recommended_plumber_9, plumber_10=recommended_plumber_10)
    except Exception as e:
        print(e)
        flash("Something went wrong", category="error")
```

Figure 22: Recommendation based on city where user lives

This function is in control of displaying all of the plumbers in the logged-in user's city. This is accomplished by obtaining the user's city and passing it to the recommendation function of the city engine class, which will be imported from the model file, in terms of finding ten best-matched plumbers who work in the same city as the logged in user.

07.3.2.1. **Output**



Plumbers In Puttalam

M.A Siyath
Bathroom Fittings
Working City : Puttalam
Experience : 1 years
Qualification : NVQ Level 1

Chamara Thiwanka Ganegoda
Tank Installation and Cleaning
Working City : Puttalam
Experience : 4 years
Qualification : NVQ Level 3

R.G.U.v Bandara
Bathroom Repairs and Installations
Working City : Puttalam
Experience : 5 years
Qualification : NVQ Level 5

Gamage Don Thilakarathna
Kitchen Repairs and Installations
Working City : Puttalam
Experience : 4 years
Qualification : NVQ Level 4

N.P.T.L Deshappriya
General Plumbing
Working City : Puttalam
Experience : 6 years
Qualification : NVQ Level 6

v. H. Premarathna
General Plumbing
Working City : Puttalam
Experience : 7 years
Qualification : NVQ Level 7

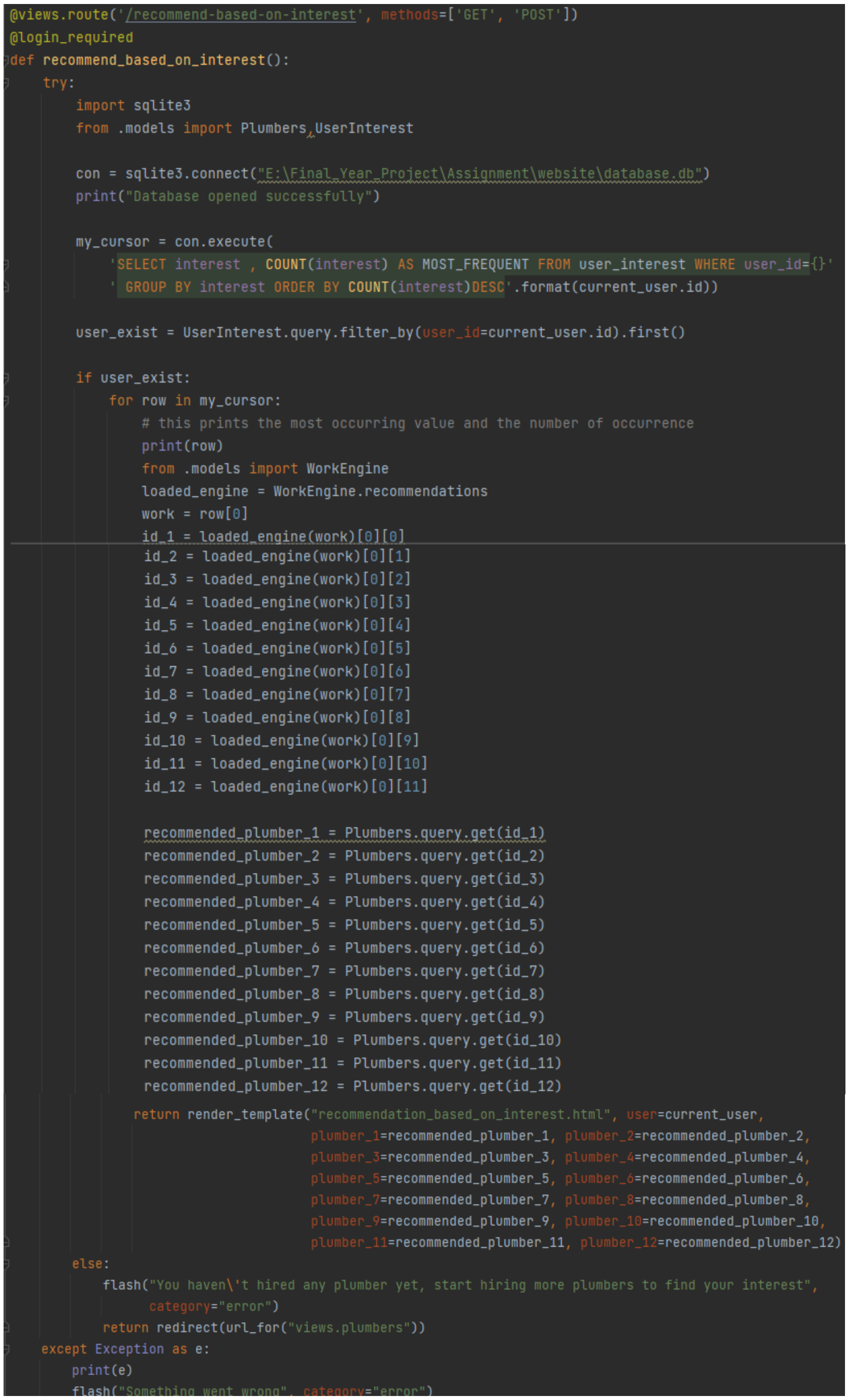
J M Preethika Priyangani
Kitchen Installations
Working City : Puttalam
Experience : 1 years

Mr.UH Chamindu Dilshan
Bathroom Repairs and Installations
Working City : Puttalam
Experience : 5 years

M.M.M.S.M .Gawarammana
Bathroom Fittings
Working City : Puttalam
Experience : 1 years

Figure 23: Output of "recommendation based on city where user lives"

07.3.3. Recommendation based on the user interest

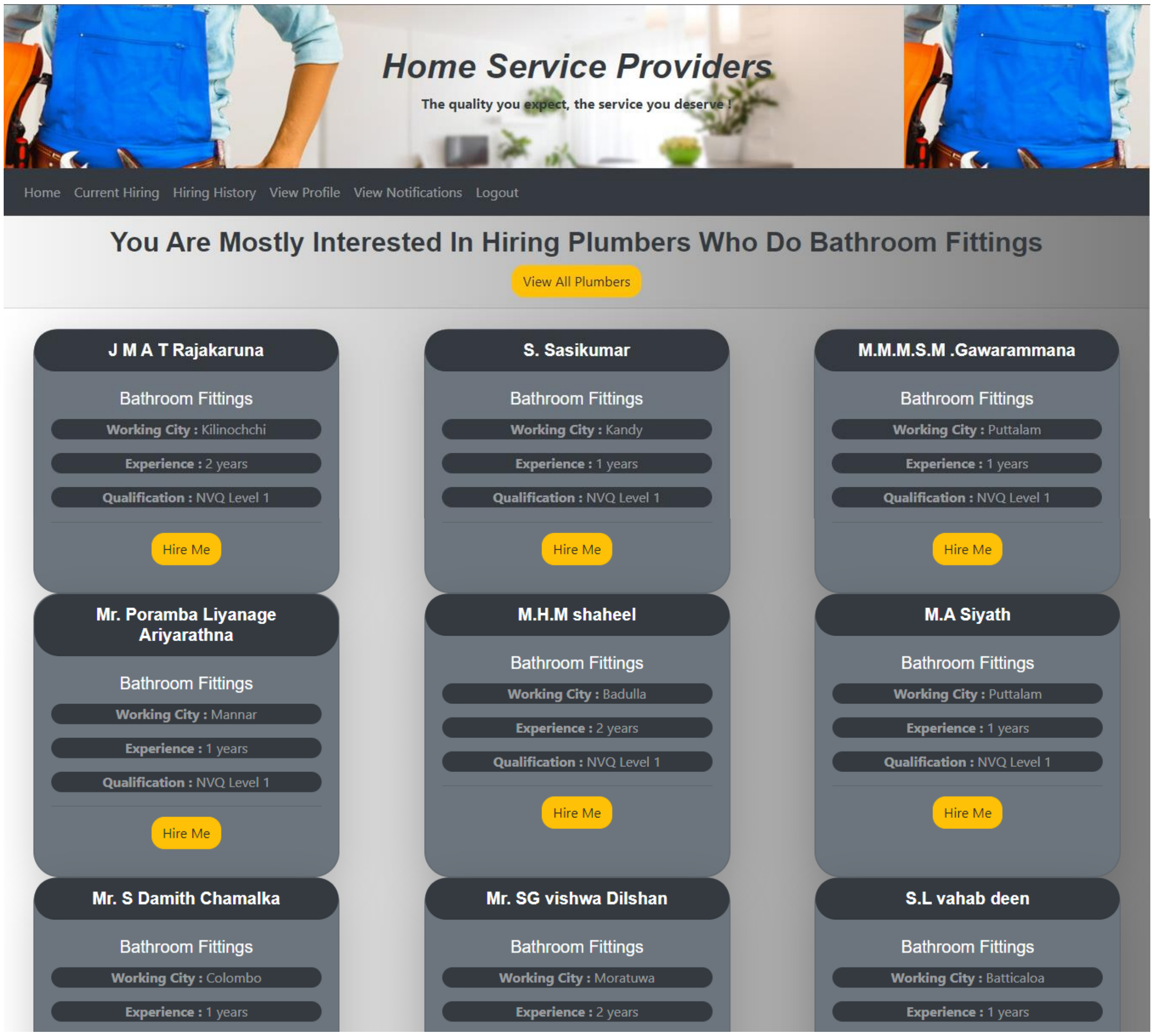


```python
@views.route('/recommend-based-on-interest', methods=['GET', 'POST'])
@login_required
def recommend_based_on_interest():
    try:
        import sqlite3
        from .models import Plumbers,UserInterest

        con = sqlite3.connect("E:\Final_Year_Project\Assignment\website\database.db")
        print("Database opened successfully")

        my_cursor = con.execute(
            'SELECT interest , COUNT(interest) AS MOST_FREQUENT FROM user_interest WHERE user_id={}'
            ' GROUP BY interest ORDER BY COUNT(interest)DESC'.format(current_user.id))

        user_exist = UserInterest.query.filter_by(user_id=current_user.id).first()

        if user_exist:
            for row in my_cursor:
                # this prints the most occurring value and the number of occurrence
                print(row)
                from .models import WorkEngine
                loaded_engine = WorkEngine.recommendations
                work = row[0]
                id_1 = loaded_engine(work)[0][0]
                id_2 = loaded_engine(work)[0][1]
                id_3 = loaded_engine(work)[0][2]
                id_4 = loaded_engine(work)[0][3]
                id_5 = loaded_engine(work)[0][4]
                id_6 = loaded_engine(work)[0][5]
                id_7 = loaded_engine(work)[0][6]
                id_8 = loaded_engine(work)[0][7]
                id_9 = loaded_engine(work)[0][8]
                id_10 = loaded_engine(work)[0][9]
                id_11 = loaded_engine(work)[0][10]
                id_12 = loaded_engine(work)[0][11]

                recommended_plumber_1 = Plumbers.query.get(id_1)
                recommended_plumber_2 = Plumbers.query.get(id_2)
                recommended_plumber_3 = Plumbers.query.get(id_3)
                recommended_plumber_4 = Plumbers.query.get(id_4)
                recommended_plumber_5 = Plumbers.query.get(id_5)
                recommended_plumber_6 = Plumbers.query.get(id_6)
                recommended_plumber_7 = Plumbers.query.get(id_7)
                recommended_plumber_8 = Plumbers.query.get(id_8)
                recommended_plumber_9 = Plumbers.query.get(id_9)
                recommended_plumber_10 = Plumbers.query.get(id_10)
                recommended_plumber_11 = Plumbers.query.get(id_11)
                recommended_plumber_12 = Plumbers.query.get(id_12)
                return render_template("recommendation_based_on_interest.html", user=current_user,
                                       plumber_1=recommended_plumber_1, plumber_2=recommended_plumber_2,
                                       plumber_3=recommended_plumber_3, plumber_4=recommended_plumber_4,
                                       plumber_5=recommended_plumber_5, plumber_6=recommended_plumber_6,
                                       plumber_7=recommended_plumber_7, plumber_8=recommended_plumber_8,
                                       plumber_9=recommended_plumber_9, plumber_10=recommended_plumber_10,
                                       plumber_11=recommended_plumber_11, plumber_12=recommended_plumber_12)
        else:
            flash("You haven\'t hired any plumber yet, start hiring more plumbers to find your interest",
                  category="error")
            return redirect(url_for("views.plumbers"))
    except Exception as e:
        print(e)
        flash("Something went wrong", category="error")
```

Figure 24: Recommendation based on the user interest

This function will search the user interest table for the most frequently hired plumbering jobs in order to determine which plumbering jobs the currently logged-in user is most interested in hiring. The SQL query will retrieve the logged in user's frequently hired plumbering work and store it in a variable. The recommendation function of the work engine class, which was imported from the model file, will use that variable to find 12 best-matched plumbers who do work that is similar to the work that the currently logged in user is interested in.

07.3.3.1. Output



You Are Mostly Interested In Hiring Plumbers Who Do Bathroom Fittings

J M A T Rajakaruna
Bathroom Fittings
Working City : Kilinochchi
Experience : 2 years
Qualification : NVQ Level 1

S. Sasikumar
Bathroom Fittings
Working City : Kandy
Experience : 1 years
Qualification : NVQ Level 1

M.M.M.S.M .Gawarammana
Bathroom Fittings
Working City : Puttalam
Experience : 1 years
Qualification : NVQ Level 1

Mr. Poramba Liyanage Ariyarathna
Bathroom Fittings
Working City : Mannar
Experience : 1 years
Qualification : NVQ Level 1

M.H.M shaheel
Bathroom Fittings
Working City : Badulla
Experience : 2 years
Qualification : NVQ Level 1

M.A Siyath
Bathroom Fittings
Working City : Puttalam
Experience : 1 years
Qualification : NVQ Level 1

Mr. S Damith Chamalka
Bathroom Fittings
Working City : Colombo
Experience : 1 years

Mr. SG vishwa Dilshan
Bathroom Fittings
Working City : Moratuwa
Experience : 2 years

S.L vahab deen
Bathroom Fittings
Working City : Batticaloa
Experience : 1 years

Figure 25: Output of "recommendation based on the user interest"

08. Testing

08.1. Research on Test Plan

Test scope, test objectives, test strategy, test environment, test deliverables, methods, techniques, and tools to be used are described in a test plan.

08.1.1. Strategies and Techniques

Software testing is the process of evaluating software with the goal of identifying errors and bugs. Software testing is also used to check the system's reliability, usability, code integrity, security, efficiency, maintainability, and other quality factors. Testing assists the developer in identifying errors and bugs that may have been missed during development and also helps in delivering a high-quality end product to the user (Sawant, et al., 2012).

08.1.1.1. Manual Testing

Manual testing is where testing is done statically. It is carried out in the early stages of the development cycle and is a time-consuming process. In other words, static testing is a type of manual testing that is usually conducted by developers, test teams, and analysts. This testing does not necessitate the execution of the software application's code (Sawant, et al., 2012).

Techniques of manual testing are conducted in 4 ways:

- **Informal Review**:
  The document's creator presents the contents to the audience, who then gives their feedback, allowing faults and flaws to be identified early on (GeeksforGeeks, 2019).
- **Walkthrough:**
  This method is used by experts to check for defects so that problems do not arise during the development or testing phases (GeeksforGeeks, 2019).
- **Inspection:**
  This technique involves verifying documents with a greater authority, such as software requirement specifications (GeeksforGeeks, 2019).

- **Technical Review:**
  This technique entails comparing documents in order to identify and correct errors. It is primarily carried out in a group of coworkers (GeeksforGeeks, 2019).

Since the dataset of this application is merged by collecting different information of Sri Lankan plumbers, the quality of the dataset may reduce. Manual testing is carried out to test the final dataset in this application.

08.1.1.2. Automated testing

The purpose of this testing is to look at the code's dynamic behavior. It also involves software testing for the input and output values to be analyzed (GeeksforGeeks, 2019). This is also called as automated testing

There are 4 types of automated testing:

- **Correctness testing**
  This way of testing involves the user to run the system to find errors.
- **Reliability testing**
  This method of testing aids in the early detection of design flaws and, as a result, provides assurance that the system meets its reliability requirements (Sawant, et al., 2012).
- **Security testing**
  This method of testing ensures that only authorized personnel can use the software and access the features that are appropriate for their security level (Sawant, et al., 2012).
- **Performance testing**
  This method of testing allows us to measure the performance characteristics of the application (Sawant, et al., 2012).

08.1.1.3.1. Correctness testing

This way of testing has 3 techniques:

- White box testing
- Black box testing
- Grey box testing

White box testing

White box testing is a powerful tool for detecting and resolving issues. White box testing is the process of providing a system with an input and observing how it processes that input to produce the desired output. This method enables testers to examine and verify a software system's internal workings (Sawant, et al., 2012). This is also considered as functional testing by developers and experts.

- This testing technique mainly focuses on:
    - Output that is expected to be returned
    - Identify any codes that are broken or badly written.
    - Identify the application's security flaws and vulnerabilities.

Black box testing

This testing strategy is critical because it will be tested from the perspective of the user without requiring any prior knowledge of the implementation. This is primarily used to ensure that the application as a whole is functioning properly. This ensures that the input is correctly accepted and that the output is properly generated (Sawant, et al., 2012).

08.2. Test Plan

Tests on each of the web application's functions, as well as the core recommendation functions, will be carried out. This will be done using the white and black box testing technique, which involves providing inputs and expecting the application to return an expected output in order to confirm that the app is ready for release. The application's final success percentage is expected to be at least above 90%, which will be calculated based on the number of test cases that passed and the number of test cases that failed the testing.

This application will be subjected to performance testing in order to assess its overall quality. To ensure that the application is release ready, the time it takes to run each function, the time it takes to load each page, the time it takes to get the user's recommendation, and much more will be tested together.

Finally, we'll perform security testing to ensure that the application doesn't crash even if an error occurs, that the customer page and functionalities are only accessible if the user is logged in, and that the customer password is properly encrypted.

08.3. Test Cases

Test ID	**Test Details**	**Inputs**	**Event Flow**	**Expected Results**	**Actual Results**	**Status**
TC001	Login with existing customer credentials	Email: shifny@gmail.com Password: shifny123	1.Open browser 2.Enter url: http://127.0.0.1:5000/ 3.Enter email 4.Enter password 5.Click Login	Login and display customer homepage	Logs in the member and displays customer homepage	**PASS**
TC002	Login with existing user email and wrong password	Email: shifny@gmail.com Password: shifny321	1.Open browser 2. Enter url: http://127.0.0.1:5000/ 3.Enter email 4.Enter password 5.Click Login	Display "Incorrect password, try again!"	Displays "Incorrect password, try again!"	**PASS**
TC003	Login with non-existing user email and password	Email: jason@gmail.com Password: jason123	1.Open browser 2. Enter url: http://127.0.0.1:5000/ 3.Enter email 4.Enter password 5.Click Login	Display "User does not exist."	Displays "User does not exist."	**PASS**
TC004	Login with empty values	-	1.Open browser 2. Enter url: http://127.0.0.1:5000/ 3. Click Login	Display "Please fill out this field	Displays "Please fill out this field	**PASS**
TC005	Register user with empty field or fields	-	1.Open browser 2. Enter url: http://127.0.0.1:5000/ 3. Click Sign up in Nav Bar 4. Click Register	Display "Please fill out this field	Displays "Please fill out this field	**PASS**
TC006	Register a user with already existing user's email	Full name: Peter Email: shifny@gmail.com Password: peter123 City: Puttalam	1.Open browser 2.Enter url: http://127.0.0.1:5000/ 3. Click Sign up in Nav Bar 4. Fill Form 5. Click Sign up	Display "Email is already taken, try another email."	Displays "Email is already taken, try another email."	**PASS**

		Address: 24/27, lily avenue				
TC007	Register a user with full name with characters less than 3	Full name: Pe Email: peter@gmail.com Password: peter123 City: Puttalam Address: 24/27, lily avenue	1.Open browser 2.Enter url: http://127.0.0.1:5000/ 3. Click Sign up in Nav Bar 4. Fill Form 5. Click Sign up	Display " Full name must be greater than 2 characters"	Displays " Full name must be greater than 2 characters"	**PASS**
TC008	Register a user with Address with characters less than 10	Full name: Josh Email: josh@gmail.com Password: josh1234 City: Puttalam Address: lily rd.	1.Open browser 2.Enter url: http://127.0.0.1:5000/ 3. Click Sign up in Nav Bar 4. Fill Form 5. Click Sign up	Display " Address is too short!"	Displays " Address is too short"	**PASS**
TC009	Register user with not matching password 1 and password 2	Full name: Peter Email: peter@gmail.com Password: peter123 Confirm Password: peter321 City: Puttalam Address: 24/27, lily avenue	1.Open browser 2.Enter url: http://127.0.0.1:5000/ 3. Click Sign up in Nav Bar 4. Fill Form 5. Click Sign up	Display "Passwords don't match"	Displays "Passwords don't match"	**PASS**
TC010	Register a user with password less than 7 characters	Full name: Jack Email: jack@gmail.com Password: jack12 City: Puttalam Address: 24/27, lily avenue	1.Open browser 2.Enter url: http://127.0.0.1:5000/ 3. Click Sign up in Nav Bar 4. Fill Form 5. Click Sign up	Display "Password must be at least 7 characters"	Displays "Password must be at least 7 characters"	**PASS**

TC011	Register a user with email less than 6 characters	Full name: Jack Email: j@g.c Password: jack123 City: Puttalam Address: 24/27, lily avenue	1.Open browser 2.Enter url: http://127.0.0.1:5000/ 3. Click Sign up in Nav Bar 4. Fill Form 5. Click Sign up	Display "Email must be greater than 5 characters"	Displays "Email must be greater than 5 characters"	**PASS**
TC012	Register with all correct details	Full name: Test Profile Email: test1234@gmail.com Password: test1234 City: Puttalam Address: 24/27, lily avenue	1.Open browser 2.Enter url: http://127.0.0.1:5000/ 3. Click Sign up in Nav Bar 4. Fill Form 5. Click Sign up	Display "Successfully signed up" and add the user to database with encrypted password	Displays "Successfully signed up" and adds the user to database with encrypted password	**PASS**
TC013	Logout from the application	-	1. Click Logout in Nav Bar	Log out the user and display "You have been logged out from our application"	Logs out the user and displays "You have been logged out from our application"	**PASS**
TC014	Access customer page without logging-in	-	1.Open browser 2.Enter url: http://127.0.0.1:5000/current-hiring	Display "Please log in to access this page"	Displays "Please log in to access this page"	**PASS**

TC015	View Plumber’s page	-	1. Login 2. Click “Hire a Plumber”	Display plumbers page with all the plumbers	Displays plumbers page with all the plumbers	**PASS**
TC016	View plumber detail	-	1. Login 2. Click “Hire a Plumber” 3. Click “Hire me”	Display selected plumber details and review of the plumber. Display recommended plumbers.	Displays selected plumber details and review of the plumber. Displays recommended plumbers.	**PASS**
TC017	Hire a plumber who works in a city where currently logged in user doesn’t live	Logged in user city: Puttalam Hiring Plumber city: Colombo	1. Login 2. Click “Hire a Plumber” 3. Click “Hire me” 4. Click “Hire me”	Display “<plumber name> is currently working <plumber city>, you can only hire plumbers from <user city>”	Displays A. Yaparoopan is currently working Colombo, you can only hire plumbers from Puttalam	**PASS**
TC018	Get recommendation based on logged in user’s city	Logged in user city: Puttalam	1. Login 2. Click “Hire a Plumber” 3. Click “Recommend based on my city”	Display recommended plumbers based on currently logged in user’s city	Displays recommended plumbers based on currently logged in user’s city	**PASS**
TC019	Get recommendation based on logged in user’s interest	Frequently hired work of plumber by logged in user: Bathroom Fittings	1. Login 2. Click “Hire a Plumber” 3. Click “Recommend based on my interest”	Display recommended plumbers based on currently logged in user’s interest	Displays recommended plumbers based on currently logged in user’s interest	**PASS**

TC020	Getting recommendation based on user interest without having any past hirings	-	1. Login 2. Click “Hire a Plumber” 3. Click “Recommend based on my interest”	Display “You haven't hired any plumber yet, start hiring more plumbers to find your interest”	Displays “You haven't hired any plumber yet, start hiring more plumbers to find your interest”	**PASS**
TC021	Get recommendation based on user selected plumber’s work	-	1. Login 2. Click “Hire a Plumber” 3. Click “Hire me”	Display selected plumber details and review of the plumber. Display recommended plumbers based on the selected plumber’s work.	Displays selected plumber details and review of the plumber. Displays recommended plumbers based on the selected plumber’s work.	**PASS**
TC022	Hire a plumber who works in the city where currently logged in user lives	Logged in user city: Puttalam	1. Login 2. Click “Hire a Plumber” 3. Click “Recommend based on my city” 4. Click “Hire me” 5. Click “Hire me”	Display success message and redirect user to current hiring page and display the currently hired plumber Send a notification to the user	Display “Chamara Thiwanka Ganegoda is hired successfully” and redirect user to current hiring page and display the currently hired plumber Send a notification to the user	**PASS**
TC023	Hire another plumber while having a plumber	-	1. Login 2. Click “Hire a Plumber” 3. Click “Hire me” 4. Click “Hire me”	Display error message to the user and redirect the user to current hiring page	Display “A plumber has already been hired by you. Once the current hire is completed, you can hire more” and	**PASS**

	already hired				redirect user to current hiring page.	
TC024	Another user trying to hire a plumber who is already on a hire currently	-	1. Login 2. Click “Hire a Plumber” 3. Click “Hire me” 4. Click “Hire me”	Display error message to the user and redirect the user to plumber’s page	Displays “Chamara Thiwanka Ganegoda is already on hire” and redirect user to plumber page	**PASS**
TC025	Trying to hire the same plumber again who was hired by the current user currently	-	1. Login 2. Click “Hire a Plumber” 3. Click “Hire me” 4. Click “Hire me”	Display error message to the user and redirect the user to plumber’s page	Displays “Chamara Thiwanka Ganegoda is already on hire” and redirect user to plumber page	**PASS**
TC026	View details of hired plumber	-	1. Click Current Hiring 2. Click View details	Display hired plumber details and status as “On The Way”	Displays hired plumber details and status as “On The Way”	**PASS**
TC027	Update hired plumber as arrived	-	1. Click Current Hiring 2. Click Arrived	Update the hired plumber status as “Arrived” and display success message	Update the hired plumber status as “Arrived” and displays “Chamara Thiwanka Ganegoda has arrived	**PASS**

				Send notification to the user.	at your location" "Make sure to click work completed button once the work is done" Sends notification to the user.	
TC028	View Details of hired plumber after arrival	-	1. Click Current Hiring 2. Click View details	Display hired plumber details and status as "Arrived"	Displays hired plumber details and status as "Arrived"	**PASS**
TC029	Update hired plumber as work completed	-	1. Click Current Hiring 2. Click Work Completed	Save the hiring details in hired history database table with status as "Completed" and delete the current hiring of the user. Redirect user to hired history page and display completed hirings of user and display success message and send notification to the user.	Saves the hiring details in hired history database table with status as "Completed" and deletes the current hiring of the user. Display "Chamara Thiwanka Ganegoda has completed his service" "Please pay the service amount requested by the plumber" and send notification to the user.	**PASS**
TC030	View Hiring details of past hired plumber	-	1. Click Hiring History 2. Click View details	Display details of hiring including date of hiring completion and status	Displays details of hiring including date of hiring completion and status	**PASS**

TC031	Add a review to service which was completed hiring	Review: Great service	1. Click Hiring History 2. Click View details 3. Enter review 4. Click add review	Add the review to the database with key information such as review, user id, plumber id and the hiring completed date And display the added review to the user	Adds the review to the database with key information such as review, user id, plumber id and the hiring completed date And displays the added review to the user	**PASS**
TC032	Add a review with characters less than 5	Review: gre	1. Click Hiring History 2. Click View details 3. Enter review 4. Click add review	Display “Please lengthen this text to 5 characters or more”	Displays “Please lengthen this text to 5 characters or more”	**PASS**
TC033	View profile	-	1. Click View Profile	Display currently logged in user’s profile	Displays currently logged in user’s profile	**PASS**
TC034	Get recommendation based on user interest while having only hired 1 plumbering service	-	1. Login 2. Click “Hire a Plumber” 3. Click “Recommend based on my interest”	Display recommended plumbers based on currently logged in user’s only hired plumber’s work	Displays recommended plumbers based on currently logged in user’s only hired plumber’s work	**PASS**

TC035	Get recommendation based on user interest while having only hired 2 plumbering service	-	1. Login 2. Click “Hire a Plumber” 3. Click “Recommend based on my interest”	Display recommended plumbers based on currently logged in user’s last hired plumber’s work	Displays recommended plumbers based on currently logged in user’s last hired plumber’s work	**PASS**
TC036	Get recommendation based on user interest while having hired 3 or more plumbering service but each of the plumbering service are unique and doesn’t repeat(no frequent hirings)	-	1. Login 2. Click “Hire a Plumber” 3. Click “Recommend based on my interest”	Display recommended plumbers based on currently logged in user’s last hired plumber’s work	Displays recommended plumbers based on currently logged in user’s last hired plumber’s work	**PASS**

TC037	View Edit Profile page	-	1. Click View Profile 2. Click Edit Profile	Display “edit profile” page with user details in editable fields.	Displays “edit profile” page with user details in editable fields.	**PASS**
TC038	Edit Profile with existing user’s email which is not currently logged in user’s email	Email: shifny@gmail.com	1. Click View Profile 2. Click Edit Profile 3. Update details/detail 4. Click update profile	Display “Email is already taken, try another email.”	Displays “Email is already taken, try another email.”	**PASS**
TC039	Edit Profile with full name less than 3 characters	Full name: JD	1. Click View Profile 2. Click Edit Profile 3. Update details/detail 4. Click update profile	Display " Full name must be greater than 2 characters"	Displays “Full name must be greater than 2 characters”	**PASS**
TC040	Edit Profile with address less than 10 characters	Address: Lilly rd.	1. Click View Profile 2. Click Edit Profile 3. Update details/detail 4. Click update profile	Display “Address is too short!”	Displays “Address is too short”	**PASS**
TC041	Edit profile full name	Current Full name: Raslan Updated full name: M. Raslan	1. Click View Profile 2. Click Edit Profile 3. Update details/detail 4. Click update profile	Update user’s full name and display success message in view profile page	Updates user’s full name and displays “Profile Updated Successfully” in view profile page	**PASS**

TC042	Edit Profile email	Current email: raslan@gmail.com Updated email: raslan.m@gmail.com	1. Click View Profile 2. Click Edit Profile 3. Update details/detail 4. Click update profile	Update user's email and display success message in view profile page	Updates user's email and displays "Profile Updated Successfully" in view profile page	**PASS**
TC043	Edit Profile City	Current city: Puttalam Updated city: Colombo	1. Click View Profile 2. Click Edit Profile 3. Update details/detail 4. Click update profile	Update user's city and display success message in view profile page	Updates user's city and displays "Profile Updated Successfully" in view profile page	**PASS**
TC044	Edit Profile address	Current address: 22/28, first lane, school road, Thilladiya Updated address: 12/38, 2nd lane, church road	1. Click View Profile 2. Click Edit Profile 3. Update details/detail 4. Click update profile	Update user's address and display success message in view profile page	Updates user's address and displays "Profile Updated Successfully" in view profile page	**PASS**
TC045	Edit all profile details	Current Full name: Ilham Ilyas Current Email: ilham@gmail.com Current City: Puttalam Current Address: 12th, joseph street, Puttalam Updated Full name: Ilham Updated Email: ilham.its@gmail.com Updated City: Kandy	1. Click View Profile 2. Click Edit Profile 3. Update details/detail 4. Click update profile	Update all user's details and display success message in view profile page	Updates all user's details and displays "Profile Updated Successfully" in view profile page	**PASS**

		Updated Address: 13th street, Puttalam				
TC046	View notification s	-	1. Click Notification	Display all notifications of currently logged in user. Most recent notifications must be displayed on top	Displays all notifications of currently logged in user. Most recent notifications are displayed on top	**PASS**
TC047	Delete a notification	-	1. Click Notification 2. Click cross symbol	Delete the selected notification from the database and display success message to the user	Deletes the selected notification from the database and displays “Notification Deleted” to the user	**PASS**

08.4. Test Conclusion

This application was tested using black box and white box testing, which required the program to be tested both from the customer's perspective and from the tester's perspective. This application was put through its tests with 47 test scenarios, all of which returned positive results as expected. This application's black box and white box testing results are 100 percent, which is higher than the expected level of 90 percent. This system was also put through its tests in terms of performance and security, and the application passed both tests with good scores. As a result, we can conclude that the program is release-ready and can be progressed for future enhancements because there is no need to focus on resolving faults, and we found none during our testing.

09. Critical Evaluation

The Home Service Recommendation System is a python-based recommendation system and website that allows users to register, login, and hire plumbers based on their preferences, choices, and location. The main goal of this application is to connect customers with service providers all around Sri Lanka. As a result, we can operate as a middleman between the buyer and the seller.

People adapted to life in a world where everything, including people's manual work, was automated. As a result, even though there are applications that satisfy similar principles as ours, we chose to automate the manual process of hiring a service provider, which has been done for years manually. And we decided to function as a mediator without becoming involved in the price charge for the service because, based on our study, similar systems exist that calculate the charge themselves without allowing the service provider to set the charge amount. Based on our study, customers are mostly unhappy with the service charge generated by similar systems. As a result, we decided involving in price would be bad move for the first release of this application, so we decided to consider about it as a future enhancement.

Throughout the development of this application, we researched a variety of topics and discussed the challenges we encountered with our mentor. Several decisions were taken that altered the idea of the project that we had throughout the early stages of development.

08.5. Challenges faced

We decided to recommend plumbers and electricians based on the user's interest and selection at the start of this application's development phase. And it was decided that this application would be developed for mobile devices, using the flutter framework and dart code. We later realized that our level of flutter knowledge was insufficient to develop such a large system with a visually appealing user interface as time went on.

Finding datasets for service providers working in Sri Lanka was also a challenge. Finally, after consulting with our mentor, we decided to combine several pieces of data and create a dataset dedicated entirely to plumbers. As a result of the change in plans, we were only able to provide plumber recommendations.

08.6. Knowledge Gained

I was limited in what I could accomplish and had to rely primarily on lecture notes and lecture recordings to study and investigate. Because I had to do more self-learning and research on this project due to the entirely new concept of software development in my learning career, it helped me break out of the usual development cycle, which I had done on all previous projects, and it helped me improve my self-learning and research skills.

Below are few topics which I came across while researching and learning:

08.6.1. **Machine Learning**

Machine learning is a sort of artificial intelligence in which we teach machines how to solve problems without directly programming them. We utilize existing data to train a function that can make a prediction given fresh data in machine learning.

08.6.2. **Flask Framework**

Flask is a web framework that prioritizes ease of use, simplicity, and fine-grained control. It just implements the minimum necessary, giving the developer unlimited control over which to import and use.

08.6.3. **Data Pre-processing**

Data preprocessing, also known as data cleaning, is the process of removing non-ASCII characters from a dataset, removing terms like "the," "and" and "is," and converting texts to lower case using a function. Data preprocessing is a critical function that must be performed during the machine learning process to ensure that the data being given is easily convertible to computer language.

08.6.4. **Similarity measuring**

This is basically used to measure the similarity of two objects passed. Before we can propose a product to a consumer, we may need to see how similar it is to one that they like or are interested in. Similarity measurements are useful in this situation. To determine how similar two products are, any method can be utilized. A fruit's color, flavor, or size, for example, can be used to determine its similarity.

08.6.5. **Recommendation System**

Recommendation systems are a type of machine learning that focuses on selecting the best product for a consumer based on their interests, preferences, location, other factors, and sometimes based on similar user’s preference. These are widely used in today's most popular applications to recommend users' needs without requiring them to do much work.

10. Future Enhancement

We decided that the program may be developed further by implementing new functionalities and logic because it has a 100% positive rate in black and white box testing. Few of the future enhancement which can be made to improve the application are listed below.

08.7. Recommendation based on both city and work

Since the existing application was only designed to recommend based on one value, it falls short of satisfying customers by failing to recommend a service provider who does the work that the client prefers and also works in the same city as the customer.

08.8. Allow customer to add rating to hired service provider

In the recommendation system, customer ratings are crucial. Because the current program is built on a content-based filtering mechanism, it may be enhanced to include both content-based and collaborative-based recommendation methods if user ratings are enabled.

08.9. Having Multiple User Functionalities

Because the current application is designed with only customer access, it may be difficult for it to be release-ready because service providers cannot register and offer their services, and the admin cannot control the application with high-level authority. By implementing this in the future, we will be able to conclude that the application is release-ready, and we will be able to begin deploying.

08.10. Allowing the customer to chat with the service provider

Customers may need to communicate with service providers during the route before the service provider arrives at the customer's home to confirm that the service provider is on the right track and is not being scammed. We can eliminate the possibility that a consumer will save the service provider's mobile number and use it to manually hire the service provider in the future by implementing this functionality. This issue could limit the amount of people who can use our app. To avoid this problem, the customer's contact information may not be shared with the service provider, and the service provider's information may not be shared with the customer. Both can communicate with each other through the application's built-in chat box.

11. Reference

Alpaydin, E., 2020. *Introduction to Machine Learning.* 4th ed. Cambridge: Massachisetts Institute of Technology.

Amaratunga, D., Fernando, N., Haigh, R. & Jayasinghe, N., 2020. Progress in Disaster. *The COVID-19 outbreak in Sri Lanka: A synoptic analysis focusing on trends, impacts, risks and science-policy interaction processes,* 8(100133).

Amaratunga, D., Fernando, N., Haigh, R. & Jayasinghe, N., 2020. Progress in Disaster Science. *The COVID-19 outbreak in Sri Lanka: A synoptic analysis focusing on trends, impacts, risks and science-policy interaction processes,* 8(100133).

Chung, L., Brian, N. A., Yu, E. & Mylopoulos, J., 2000. *Non-Functional Requirenments in Software Engineering.* 1st ed. New York: Kluwer Academic Publishers.

De Silva, N. & Ranasinghe, M., 2010. Maintainability risks of condominiums in Sri Lanka. *Journal of Financial Management of Property and Construction,* 15(1), pp. 41-60.

Ekstrand, D. M., Riedl, T. J. & Konstan, A. J., 2010. *Collaborative filtering recommender systems.* 2nd ed. Netherland: Foundation and Trends.

Garcia, R. et al., 2020. A systematic literature review on the use of machine learning in precision livestock farming. *Computers and Electronics in Agriculture,* 179(105829).

GeeksforGeeks, 2019. *Software Testing | Dynamic Testing.* [Online]
Available at: https://www.geeksforgeeks.org/software-testing-dynamic-testing/
[Accessed 15 01 2022].

GeeksforGeeks, 2019. *Software Testing | Static Testing.* [Online]
Available at: https://www.geeksforgeeks.org/software-testing-static-testing/#:~:text=Static%20Testing%20is%20a%20type,executed%20to%20detect%20the%20defects.
[Accessed 15 01 2022].

Geeksforgeeks, 2020. *Using CountVectorizer to Extracting Features from Text.* [Online]
Available at: https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/
[Accessed 23 12 2021].

International Labour Organization, 2020. *Working from Home: Estimating the worldwide potential,* Switzerland: International Labour Organization.

Meteren, R. v. & Someren, M. v., 2000. *Using Content-Based Filtering for Recommendation,* Netherlands: NetlinQ Group.

Paireekreng, W., 2013. *Mobile content recommendation system for re-visiting user using content-based filtering and client-side user profile,* Tianjin: IEEE.

Ribeiro, M., Grolinger, K. & Capretz, A. M., 2015. *MLaaS: Machine Learning as a Service,* Miami: IEEE.

Ryngksai, I. & Chameikho, L., 2014. ecommender Systems: Types of Filtering. *International Journal of Engineering Research & Technology,* 3(11), pp. 251-254.

Salton, G. & Buckley, C., 1988. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management,* 5(24), pp. 513-523.

Sawant, A. . A., Bari, P. H. & Chawan, P. M., 2012. Software Testing Techniques and Strategies. *International Journal of Engineering Research and Applications,* 2(3), pp. 980-986.

Shoval, P., Maidel, V. & Shapira, B., 2008. An Ontology- Content-based Filtering Method. *Information Theories & Applications,* 15(4), pp. 303-314.

Wagner, M., Balke, W. T., Hirschfeld, R. & Kellerer, W., 2002. *A Roadmap to Advanced Personalization of Mobile Services,* Augsberg: ODBASE.

Xu, G., Zhang, Y. & Li, L., 2010. *Web Mining and Social Networking.* 1st ed. New York: Springer.

Ye, J., 2011. Cosine similarity measures for intuitionistic fuzzy sets and their applications. *Mathematical and Computer Modelling,* 53(2), pp. 91-97.

12. Appendix

12.1. Project Planning

ID	Tasks	Duration	Aug				Sep				Oct				Nov				Dec				Jan and Feb			
1	PPF	5days	🟧																							
2	PSR	3weeks		🟧	🟧	🟧																				
3	Project Research	5Weeks				🟧	🟧	🟧	🟧	🟧																
4	MPR	2Weeks											🟧	🟧												
5	Additional Research	5Weeks												🟧	🟧	🟧	🟧	🟧								
6	Developing Frontend Prototype	3Week											🟧	🟧	🟧											
7	Prototype Testing	5days														🟧										
8	Developing Frontend	5Weeks														🟧	🟧	🟧	🟧	🟧						
9	Developing Backend	8Weeks													🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧				
10	Connect Frontend and Backend	3Weeks																					🟧	🟧		
11	Complete Documentation	2Weeks																						🟧	🟧	
12	System Testing	2Week																							🟧	
13	Complete the project	3Weeks																							🟧	🟧
14	Final Submission Checking	3Weeks																							🟧	🟧