Recommendation System

Final Year Project

Session 2018-2021

A project submitted in partial fulfilment of the

COMSATS University Degree

of

BS in Computer Science / Software Engineering (CUI)

Department of Computer Science

COMSATS University Islamabad, Lahore Campus

01 January 2022

Project Detail

Type (Nature of project)	[✓] **D**evelopment [ ] **R**esearch [ ] **R&D**
Area of specialization	Mobile App Development, Web App Development, Machine Learning, Natural Language Processing

Project Group Members

Sr.#	Reg. #	Student Name	Email ID	*Signature
(i)	SP18-BCS-159	Ibad Ahmad	ibad23ahmad@gmail.com	[illegible]
(ii)	SP18-BCS-047	Haseeb Yaseen	haseeb.yaseen08@gmail.com	[illegible]
(iii)	SP18-BCS-007	Wahaj Hafeez	wahajhafeez63@gmail.com	[illegible]

*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to work of others

Plagiarism Free Certificate

This is to certify that, I am Ibad Ahmad S/D/o Naseem Anwar, group leader of FYP under registration no CIIT/SP18-BCS-159/LHR at Computer Science Department, COMSATS University Islamabad, Lahore Campus. I declare that my FYP proposal is checked by my supervisor and the similarity index is 17% that is less than 20%, an acceptable limit by HEC. Report is attached herewith as Appendix A.

Date: 26-12-2021 Name of Group Leader: Ibad Ahmad Signature: [illegible]

Name of Supervisor: Kanza Hamid Co-Supervisor (if any):____________________

Designation: Lecturer Designation: ____________________

Signature: [illegible] Signature: ____________________

Abstract

Our project is a generic recommendation system, which will be varying and adopting in the environment with the passage of time. In this project, our aim is to provide an online recommendation system service, which people can use to make their experience a lot better on contrary to their experience with some other service. To find a correct, low budget orientated and location friendly service is very important and required for users but it’s not a cake walk today as there are plenty of sources which are not among those which can be reliable.

As of now, lot of recommendation systems are not able to suggest users an appropriate place that map their needs. Information mismatch have a great negative impact on such recommendation system predictions. To make a customised recommended application for imparting beneficial and powerful on line services, we want mass reviews and updated data from online databases.

Acknowledgement

We have worked hard on this project and we are very grateful and appreciative for the guidance from COMSATS University Lahore and the continued supervision of Ma`am Kanza Hamid. We thank Ma`am Kanza for choosing us for this project. She provided us with ongoing help, support, and guidance. Without your valuable support and assistance, it would have been impossible to start this project.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1 Introduction

1.1 Introduction

In different fields, including e-commerce, social media applications, and video platforms, recommendation systems [1] are one of the most important aspects. They help people to suggest what might be interesting, useful, and relatable for them.

Models are built to recommend the content which is appropriate according to user requirement or pattern by depending on different factors like user information, interest of user, targeted age, reviews and ratings. This helps to keep user engage with the application and it becomes easy to suggest something to users according to his/her needs by using provided data.

Starting with the basic idea, Recommendation system is sort of data filtering method, in which on the basis of certain parameters you show certain data to user from database. The parameters can be rating, prices/rates, accessing comfort, nearby one etc. In simple words, Recommendation systems provides only that information which will be user hoping against the content he has provided to system.

Now our goal was to create a recommendation system which will suggest its users about multiple things like hospitals, restaurants, hotels etc. depending on the different parameters like ratings, remarks/feedback and the list goes on. On the other hand, system should be able to be adaptive with new parameter’s value which can help the system to remain up to date with the future recommendations.

Now if we talk about the basic parameters, these are as following

- Rating
- Reviews
- Distance

1.2 Objectives

- To make it easy and effective for users to get best recommendations nearest to their location and easily accessible.

- To develop such application that will automate the function of finding the appropriate recommendation according to user requirement, thus bring easiness, save time and efforts needed to find a best suitable suggestion.
- To overcome the drawbacks of traditional approaches for finding a thing by developing such recommendation system that will uses natural language processing and machine learning techniques to produce a recommendation list.
- To develop such intelligent recommendation system that performs accurately and efficiently and help users to know about the authentic, best, affordable, nearest hospitals, restaurants, hotels etc. whom which they can trust or prefer.
- To create a model, that will vary time to time in order to acquire excellence in further recommendations.

1.3 Problem statement

In the modern world, a number of applications are there that suggest what is most appropriate for the user. Recommendation systems are widely used to suggest the appropriate, relevant, according to mood and requirement. Tech giants like Facebook, YouTube, and Netflix uses RS to show what is according to the pattern of users. On the other hand, applications like Yelp, Food Panda, etc. use the RS to show the results which are according to the reviews and ratings of other users that have previously experienced that place. Each of these platforms uses a different type of RS for generating different types of recommendations. But most of the applications are handling a single domain, like food, cars, entertainment, etc.

Our project was to come up with a generic recommendation system that would be able to recommend any field of the second type explained i.e. Recommendation on the basis of feedback. The generic model would analyse textual data (reviews) and after that, we will generate the final rating on the basis of the combination of user ratings and reviews generated by the model on the basis of reviews. Moreover, we will make this process dynamic i.e. as more reviews come, ratings will be changed according to reviews.

1.4 Assumptions & constraints

1.4.1 Assumptions

The project was supposed to be a user-friendly web and mobile application, which would be able to provide the recommendation of places in your surroundings on the basis of the feedback provided by the other users. The services users will get are:

- Users will get recommendations on the basis of the experience of other users.
- Recommendations given to users will be dynamic, which will be changed over the period of time.
- Users will be able to see the place's profile and read comments to get a better idea about any place.
- Users will be able to share his/her experience in terms of feedback.

1.4.2 Constraints

There are many positive aspects of the project in different ways, but there is some limitation of the project too, both technical and non-technical. As for the technical side, our recommendation system depends on the reviews and we don't have any control over the language used in reviews, which may lead to the problem of discarding of reviews. Another problem is that no system has a perfect database that covers all areas, so if someone wants to get a response about a location that does not exist in our database, in that case we would depend upon Yelp’s API, data will be feed to model and then show ratings. As far as non-technical constraints are concerned, in the initial stage, we will only show recommendations and there will be no such system like appointment or booking of places. Our System won’t be able to provide product reviews from several businesses like shopping malls etc.

1.5 Project motivation and scope

Since we are living in the 3rd word country and are among a developing one in Asia we don’t have recourses that can cover up the mass requirements. There are hundreds of thousands of cases weekly that are unable to get even first aid timely especially those who are lying in or near to rural areas. Pakistan’s population which belongs to this section is reported to be 63.09% in 2019 which is a shocking stat itself. [2]

One of the many examples we can site here is of *Muniba Mazari* [3]also known as the Iron Lady of Pakistan. She is a Pakistani activist, anchor artist, model, singer and motivational speaker. On 27 February 2008, Muniba and her husband were travelling from Quetta to Rahim Yar Khan. Their car met with an accident, in which she sustained several major injuries, including broken bones in her arm, rib-cage, shoulder blade, collarbone and spine. Her lungs and liver were also deeply cut. Moreover, her entire lower body was left paralyzed. She was taken to a nearby hospital, which was ill-equipped to deal with such a severe case. She was then taken to her native hospital but result didn't change as there were no doctors or equipment's to handle the patient either. She was then moved to a hospital in Rahim Yar Khan, and eventually, she was admitted to the Agha Khan Hospital, Karachi. Post-surgery, she was left bed-ridden for two years.

The idea to cite the above example is there are several cases of these context where the population is unable to find the emergency service just because of the unawareness and no helping hands that can lead them to desired output. Our aim is same, to provide that sort of application which will be able to guide with best possible source of help that user can imagine or desire at runtime.

Natural Language processing is a vast field that acts as an umbrella for many different fields. Getting Recommendations by using reviews are also one of the things which come under this. This application will cover most of the modules that we counter in normal life ranging from Hotels, Malls, and Hospitals etc. Furthermore, user can apply filters as per their use. In future, more modules can be added to this system as user requirements increase and so do filters. We have develop such a system which will help in the following:

- Generate recommendations on the basis of feedback i.e. ratings and reviews.
- Make a dynamic system, so that ratings will be changed over the period of time according to new feedbacks

.

Chapter 2

Requirement Analysis

2 Requirements Analysis

2.1 Literature review / Existing system study

There is a lot of work in the field of Recommendation systems has been done. Recommendations systems are used to recommend different things while considering multiple parameters. From the literature and existing systems, some related applications, working platforms and researches have been discussed below, apart from working principles of RS.

2.1.1 Research-Based Related Work

1) Recommendation systems: Principles, methods and evaluation

This is a research-based article [4], which explains the characteristics, methods, and potential of different types of recommendation systems that are used to show the data which is according to the interest of the user.

The authors explained multiple types and phases of RS and explained the working of each phase. One of the most important phases is the collection of information for different types of recommendations. Types are explained in detail, which are:

- Explicit Feedback
- Implicit feedback
- Hybrid feedback

Phases of systems are divided as follow which shows the relationship between each phase:

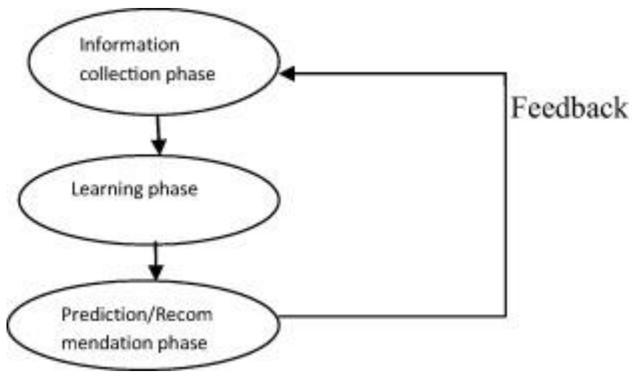

Information collection phase
Learning phase
Prediction/Recom mendation phase
Feedback

Figure 1: Phases of Recommendation System

Once the phases of recommendations are described, filtering techniques were discussed that was used to give recommendations. The authors have described each filtering method, its pros & cons, sub-types, and examples of each filtering approach in the paper. Figure 2 is showing the filtering approaches for RS.

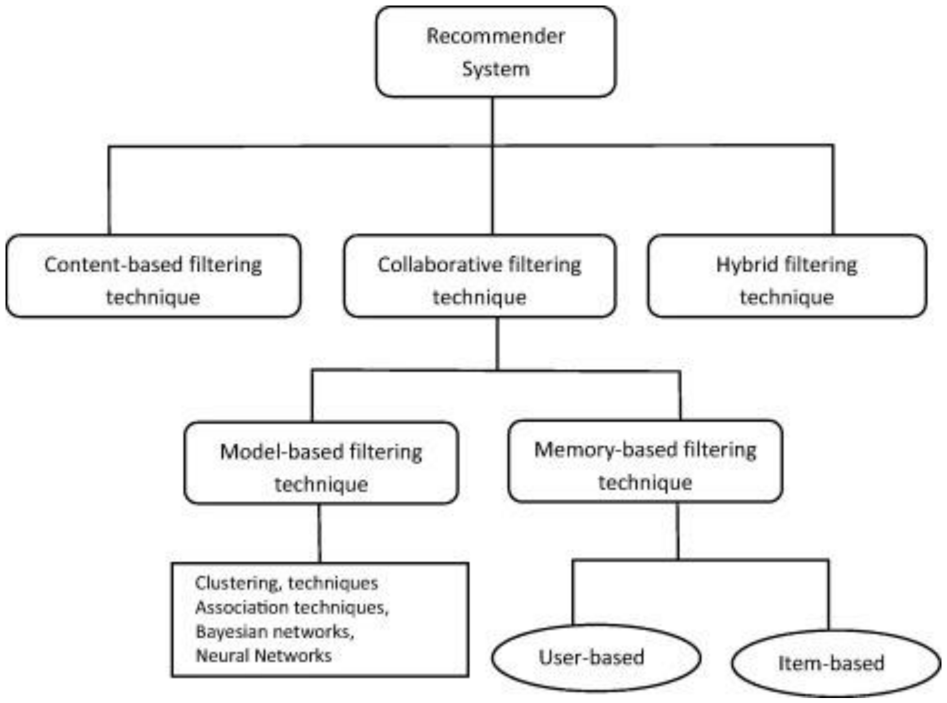
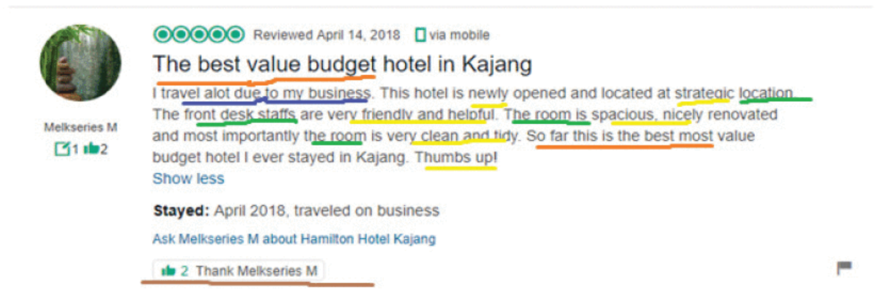

Recommender System
Content-based filtering technique
Collaborative filtering technique
Hybrid filtering technique
Model-based filtering technique
Memory-based filtering technique
Clustering, techniques
Association techniques,
Bayesian networks,
Neural Networks
User-based
Item-based

Figure 2: Filtering approaches in Recommendation System

2) Multi-Criteria Review-Based Recommender System–The State of the Art

In this research paper [5], there are multiple parameters that have been discussed, on the basis of which RS can generate ratings or in other words can show better results. The authors explained the importance of reviews and the advantages of taking "reviews" into consideration. Moreover, problems have been discussed which are there, when we work with "reviews". Figure 3 shows the information which we can extract from reviews.



Reviewed April 14, 2018 via mobile

The best value budget hotel in Kajang

I travel alot due to my business. This hotel is newly opened and located at strategic location. The front desk staffs are very friendly and helpful. The room is spacious, nicely renovated and most importantly the room is very clean and tidy. So far this is the best most value budget hotel I ever stayed in Kajang. Thumbs up!

Show less

Stayed: April 2018, traveled on business

Ask Melkseries M about Hamilton Hotel Kajang

2 Thank Melkseries M

1. **Total Review Polarity Score:** yellow underline marks the sentiment words for calculating the total score.
2. **Term / Features:** green lines show either a term (most frequent words) or a feature (concept)
3. **Context:** blue lines
4. **Comparative words:** red lines
5. **Helpful:** brown lines.

Figure 3: Information in reviews

Another section of this research paper is Multi-Criteria RS which explains that how we can generate better recommendations by considering multiple factors. This is what we are trying to implement in our FYP i.e. consider ratings and reviews of the place and generate ratings based on both. Figure 4 depicts different steps of Multi-Criteria Review based RS.

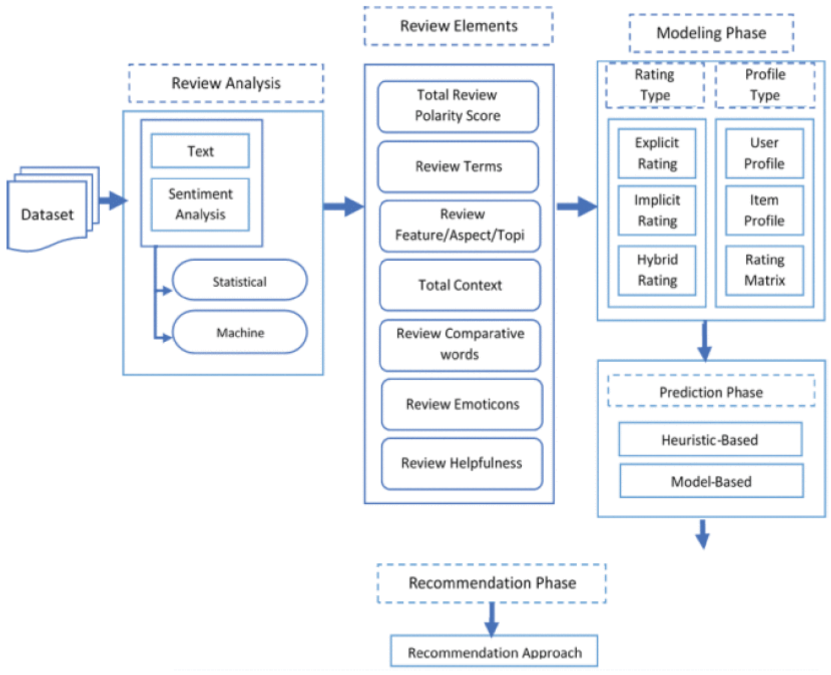


Figure 4: Multi-Criteria Review based RS

3) Recommender System Based on Consumer Product Reviews

This research paper [6] proposed one of the solutions to create a recommendation system based on reviews. This solution considers mining different parameters from the text, and on the basis of that, we evaluate multiple factors. Moreover, prioritization and ontology have also been created to recommend the best on the basis of reviews. In the paper, RS for cameras is developed. The following Figure 5 shows the general flow of the proposed solution.

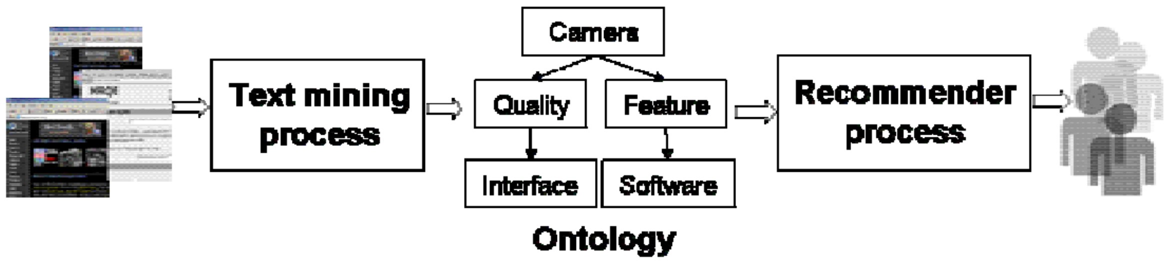


Figure 5: Architecture of the proposed solution

2.1.2 Application-Based Related Work

1) Yelp

Yelp [7]is one of the biggest platforms in the RS category. Yelp recommends multiple places related to different fields base on the reviews and ratings of previous users. For instance, if you select the restaurants and search for "steaks", depending on which city you are in, it will show you the highest-rated restaurants that serve steak. You can further filter the recommendations on the basis of different factors like nearby, price-efficient, etc. But Yelp's services are not available in our Region.

2) TripAdvisor

TripAdvisor [8] is another app that recommends restaurants, hotels, vacation rentals in multiple countries and this depends on the ratings and reviews given by previous users. One can see the profile of the place which includes photos, reviews, and ratings about the place. You can also get customized recommendations depending on the price range and nearby to someplace. But recommendations are always from best high to low ratings which are calculated by considering the feedback of users.

3) Urban Spoon

Urban Spoon [9] is another RS-based app that is like Yelp but only handles a single module i.e. finding places to eat. By using Urban Spoon, you can not only search places to eat but customize searches like the internet availability, category of food, etc. On Urban Spoon not only business owners and users provide reviews but also local food critics and chefs also give their reviews.

4) Pinterest

Pinterest has one of the most widely used recommender networks (Pixie), with over 10 billion suggestions served per day. They've installed personalization engines that can serve the good recommendation to the right customer at the right time, selecting from a pool of over 100 billion artefacts in real time—by combining the data they've accumulated over the time with human curation. [10]

5) Netflix

When you use Netflix, their recommendations system tries to make it as easy as possible for you to find a show or movie to watch. They calculate the probability of users for watching a specific movie in Netflix based library on a variety of factors, including:

- your experiences using their service (such as your viewing history and how your ratings),
- other participants with common preferences and values

These bits of information are used as inputs to Netflix algorithms [11].

2.2 Stakeholders list (Actors)

In table 1 below, List of all the stakeholders are mentioned.

Table 1: List of stakeholders (Actors)

Stakeholder	Area/Skill set	Type of Stakeholder	Expectations	Influence	Impact
Software Engineer	Software	Internal	applies software engineering principles to the development, maintenance, testing, and evaluation of applications	High	High
Database Developer	Database	Internal	Database components of the application must be created or maintained.	Low	Low
Developer	Development	Internal	builds and creates applications, writes, debugs, and executes application source code	High	High
Natural Language Processing NLP) Engineer	Programming, Statistical Analysis, Text Representation	Internal	Transforms natural language data representation into features that classification algorithms may understand.	Low	High
Machine Learning Engineer	Programming, Statistics, Data Modeling and Evaluation	Internal	creates algorithms and builds model that enables application to take action automatically	High	High
Application Tester	Analytical, Critical, Logical	Internal	Creates algorithms and models that allow an application to take action on its own.	Low	High
SQA Engineer	Quality Assurance	Internal	In order to ensure consistency, track, evaluate, and test the application during production.	High	Low

End User	Customer	External	After the software has been completely produced, use it and provide input.	Low	High
Project Team	Database, Machine Learning, Development	Internal	Manage Databases , models retraining and development stuff	High	High
Project Supervisor	Machine Learning	Internal	Supervision of the project	High	High

2.3 Requirements elicitation

2.3.1 Functional requirements

In Software Engineering, functional requirements define the scope and working of system components in a way that is feasible for both client and the team developing it. It differs software to software.

2.3.1.1 FR – 01 User Sign Up:

In Table 2 below, the functional requirements of the sign up page are mentioned.

Table 2: Functional Requirement-01: Sign Up

FR 01 – 01	**User enters his/her Full name**
FR 01 – 02	User enters his/her username
FR 01 – 03	User enters password (must contain alpha-numeric and special characters and should be at least six characters long)

FR 01 – 04	User confirms his/her password
FR 01 – 05	User accepts Terms and conditions checkbox
FR 01 – 06	User clicks on the signup button
FR 01 – 07	Error is displayed in-case of any failed step
FR 01 – 08	User can also signup with their Google account

2.3.1.2 FR – 02 User Log in:

In Table 3 below, the functional requirements of the login page are mentioned.

Table 3: Functional Requirement-02: Login

FR 02 – 01	**User enters his/her Full name**
FR 02 – 02	User confirms his/her password
FR 02 – 03	User accepts Terms and conditions checkbox
FR 02 – 04	User clicks on the log in button
FR 02 – 05	Error is displayed in-case of any failed step
FR 02 – 06	User can also sign in with their Google account

2.3.1.3 FR – 03 Forget Password:

In Table 4 below, the functional requirements of the forget password page are mentioned.

Table 4: Functional Requirement-03: Forget Password

FR 03 – 01	**User enters his/her Email**
FR 03 – 02	User gives the verification code he’d received through email
FR 03 – 03	User enters new password

FR 03 – 04	User confirms the new password
FR 03 – 05	User clicks the reset password button
FR 03 – 06	Error is displayed in-case of any failed step

2.3.1.4 FR – 04 User Log Out:

In Table 5 below, the functional requirements of the login page are mentioned.

Table 5: Functional Requirement-04: Logout

FR 04 – 01	**User clicks on sign out button**
FR 04 – 02	He/ She is redirected to login page

2.3.1.5 FR – 05 User Profile:

In Table 6 below, the functional requirements of the user profile/dashboard are mentioned.

Table 6: Functional Requirement-05: User Profile

FR 05 – 01	**User goes to profile page**
FR 05 – 02	He/ She can update his personal information
FR 05 – 03	User can delete his/her profile permanently (warning will be generated and extra verification step will be given)
FR 05 – 04	In-case of account deletion, all of his data from database will be deleted and will be redirected to log in page afterwards.

2.3.1.6 FR – 06 Searching / Get Recommendations:

In Table 7 below, the functional requirements to get recommendations are mentioned.

Table 7: Functional Requirement-06: Searching

FR 06 – 01	**User gives the requirements.**
FR 06 – 02	User applies the filter
FR 06 – 03	User clicks the search button

2.3.1.7 FR – 07 View Places Profiles:

In Table 8 below, the functional requirements to view place profile are mentioned.

Table 8: Functional Requirement-07: View Place Profile

FR 07 – 01	**User enters place he/she wants to see profile of**
FR 07 -02	User clicks the view profile button
FR 07 – 03	Place Profile will be displayed like location, rating and top comments etc.
FR 07 – 04	User can gives his feedback by clicking on the give feedback button
FR 07 – 05	User will give a place its rating ,comment and the required information
FR 07 – 06	User feedback will be added.
FR 07 – 07	Error is displayed in-case of any failed step

2.3.2 Non-functional requirements

On the other hand, non-functional requirements describe how system should behave. In short context, it represents the quality or attributes of the system. These requirements cover up all the left-overs from the functional requirements.

2.3.2.1 NFR – 01 Transportability / Compatibility:

In Table 9 below, the non-functional requirements related to compatibility are mentioned.

Table 9: Non-Functional Requirement-01: Compatibility

NFR 01 – 01	**Systems android and web app should be compatible with all the desktops and native devices respectively.**
NFR 01 – 02	System shouldn't undergo any major changes when sifting to any other Operating System
NFR 01 - 03	Both apps should be consistent and generalize throughout

2.3.2.2 NFR – 02 Performance:

In Table 10 below, the non-functional requirements related to performance of the application are mentioned.

Table 10: Non-Functional Requirement-02: Performance

NFR 02 – 01	**Performance of the system must be good**
NFR 02 – 02	System's quick response will outline its performance irresponsive of the mass interacting with him.

2.3.2.3 NFR – 03 Secure:

In Table 11 below, the non-functional requirements related to secure environment of platform are mentioned.

Table 11: Non-Functional Requirement-03: Compatibility

NFR 03 – 01	**The system must be secure so do the database which has primary role in system functioning properly**
NFR 03 – 02	System Database should be on a secure cloud platform.
NFR 03 - 03	System should ensure both data integrity and user privacy

NFR 03 - 04	The system should maintain correct as well as consistent information which will help to generate good recommendations.

2.3.2.4 NFR – 04 Reliability:

In Table 12 below, the non-functional requirements related to reliability are mentioned.

Table 12: Non-Functional Requirement-04: Reliability

NFR 04 – 01	**System should only respond to correct and valid feedbacks and ignore the fake ones.**
NFR 04 – 02	System can also block the fake user permanently

2.3.2.5 NFR – 05 Usability / User Friendly:

In Table 13 below, the non-functional requirements related to user friendly environment of the application are mentioned.

Table 13: Non-Functional Requirement-05: User-Friendly

NFR 05 – 01	**System must present a formal documentation tutorial in the form of pop-ups for newbies.**
NFR 05 – 02	The system should only focused on what is primary / important for the user and don't focus on what would put the user in any type of mess or confusion
NFR 05 - 03	Web Application should be information orientated while native app should be task orientated.

2.3.2.6 NFR – 06 Maintainability:

In Table 14 below, the non-functional requirements related to maintenance are mentioned.

Table 14: Non-Functional Requirement-06: Maintainability

NFR 06 – 01	**The system should be maintained properly. Whenever there is a problem, it must be fixed as soon as possible.**
NFR 06 – 02	Moreover, the system must be timely updated according to the needs of the user

2.3.3 Requirements traceability matric

In Table 15 below, Requirement traceability matrix has been displayed.

Table 15: Requirement Traceability Matric

Test Case ID	**FR_ID**	**Description of Requirement**	**Objective**	**Priority**
01	FR-01	The system will provide users with 2 types of accounts i.e. Admin and User.	Register a new user.	High
01	FR-02	The system will facilitate the user with a login system.	Logging user account.	High
02	FR-06	The system will facilitate user with the recommendation of different places based upon different parameters	Make Recommendation of Places	High
03	FR-07	The system will shows a complete Place profile to the user which he has just searched with options like place reviews, rating or location etc.	Display Place Profile to the user	High
04	FR-07	User can add his respective review of that place	User Feedback about Place	High

2.4 Use case descriptions

2.4.1 Use case-02: Sign In

The use case description for the login process is shown in Table 16.

Table 16: Use case description-01: Login

ID:	**1**
Name of Use Case:	Login
Actors:	Admin, User, Database
Description:	The admin/user can log in to the application.
Pre-Condition:	Admin/User should've their account registered beforehand.
Post-Condition:	Admin/User has been signed into the application, successfully.
Events:	1. Admin/user opens the app. 2. Admin/user clicks on the login / sign in button. 3. Admin/user fills the required fields (username and password) 4. After the authentication has been done, user will be redirected to dashboard
Alternatives Flow:	In case of forget password, user will be provided with the reset link by the app.
Exceptions:	None

2.4.2 Use case-02: Sign Up

The use case description for the sign up process is shown in Table 17.

Table 17: Use case description-02: Sign Up

ID:	**2**

Name of Use Case:	Sign Up
Actors:	User, Database
Description:	User can create their respective account for application.
Pre-Condition:	No account of that credentials already exist.
Post-Condition:	User account has been created and they are logged in
Events:	1. User opens the app. 2. User clicks on the sign up button. 3. User fills the required fields of the form 4. After the sign up procedure has been done, user will be redirected to dashboard
Alternatives Flow:	User already has an account.
Exceptions:	None

2.4.3 Use case-03: Logout

The description of the use case of the logout process is shown in Table 18.

Table 18: Use case description-03: Logout

ID:	**3**
Name of Use Case:	Logout
Actors:	Admin, User
Description:	The user/admin can logout from the application.
Pre-Condition:	The user/admin should be signed in his/her account.
Post-Condition:	User/admin has been successfully logged out from the application.
Events:	1. The user presses the logout button. 2. The user logs out and the login page appears.
Alternatives Flow:	The user has already logged out.
Exceptions:	None

2.4.4 Use case-05: User Portal

The use case description for the User Portal is shown in Table 19.

Table 19: Use case description-05: User Portal

ID:	**5**
Name of Use Case:	User Portal
Actors:	User, Database, Yelp's API, Trained Model
Description:	User can manage his profile , view profiles of the places , add places to his/her favorites section, give feedback in the form of reviews and comments, get recommendations
Pre-Condition:	User has been logged in.
Post-Condition:	User has carried out its desire goal
Events:	1. User opens the app. 2. User logs in 3. User chooses either of the task he wants to perform 4. User performs the task by following the valid set of operations
Alternatives Flow:	In case of forget password, user will be provided with the reset link by the app.
Exceptions:	User Portal has been deleted due to some reason by admin or by himself accidently.

2.4.5 Use case-06: Admin Portal

The use case description for the Admin Portal is shown in Table 20.

Table 20: Use case description-6: Admin Portal

ID:	**6**

Name of Use Case:	Admin Portal
Actors:	Admin, Database
Description:	Admin can view the users, their details and can check recommendations and other functions of app. Moreover, can manage app users either by deleting or updating.
Pre-Condition:	Admin has logged in or admin is a valid admin
Post-Condition:	Admin has carried out its desire goal
Events:	1. Admin opens the app. 2. Admin logs in 3. Admin chooses either of the task he wants to perform 4. Admin performs the task by following the valid set of operations
Alternatives Flow:	None
Exceptions:	Admin is not a valid admin

2.4.6 Use case-07: Recommendations

The use case description for the Admin Portal is shown in Table 21.

Table 21: Use case description-7: Recommendations

ID:	**7**
Name of Use Case:	Recommendations
Actors:	User, Database, Trained Model, Yelp's API
Description:	Application shall get location given by user and will show recommendation according to need.
Pre-Condition:	Location must be valid.
Post-Condition:	Recommendations have been shown to user.
Events:	1. User enters location and choose the recommendations which he/she needs. 2. From database data will be retrieved against the location.

	3. Retrieved recommendations will be shown to user according to system ratings of the places available at that location.
Alternatives Flow:	Entered location's data is not available in database. In that case following flow will be considered. 1. User enters location and choose the recommendations which he/she needs. 2. If data is not in database, it will be retrieved from Yelp's API. 3. Retrieved recommendations will be shown to user according to Yelp's provided data of the places available at that location. 4. After that data of that location will be feed to trained model and data will be maintained of that location, according to Model's generated ratings.
Exceptions:	None

2.5 Use case design

2.5.1 Use case-01: User and Admin Login

The use case for the login process for the user and admin is shown in Figure 6. The users and admin will enter their credentials, the System will match them with the database and the appropriate function will run. While there remains the option of "forgot password" for users.

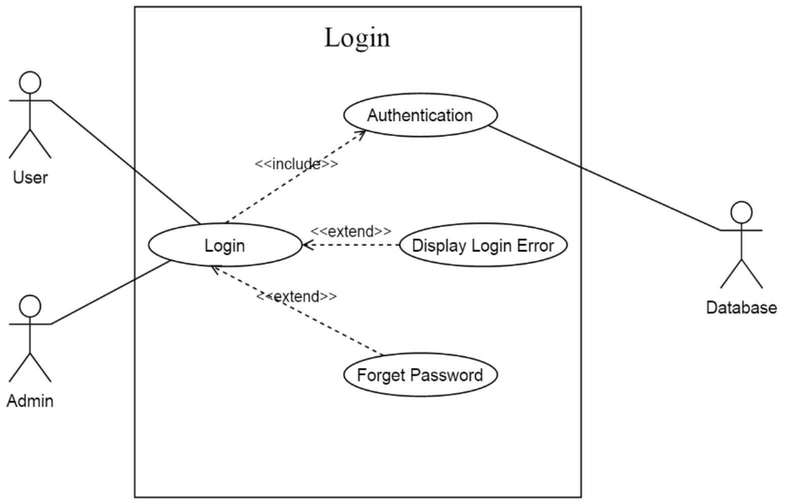


Figure 6: Use case-01: Login

2.5.2 Use case-02: User Sign Up

The use case for the signup process for the user is shown in Figure 7.

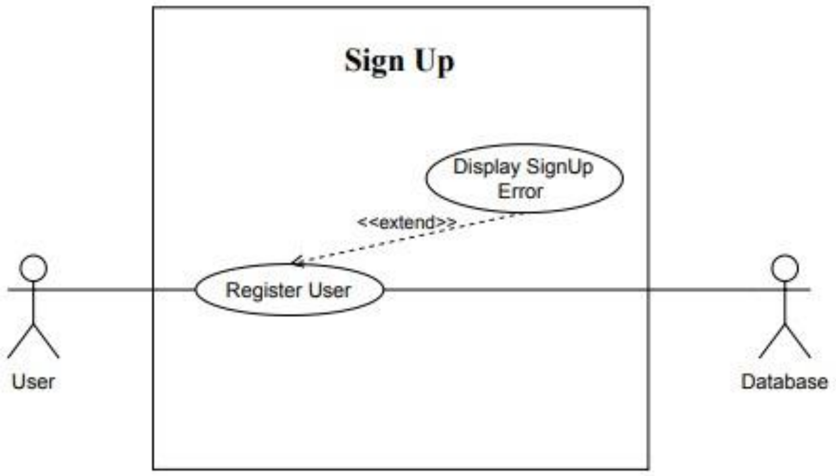


Figure 7: Use case-02: Sign Up

2.5.3 Use case-03: User and Admin Logout

The use case for the logout process for the user and admin is shown in Figure 8.

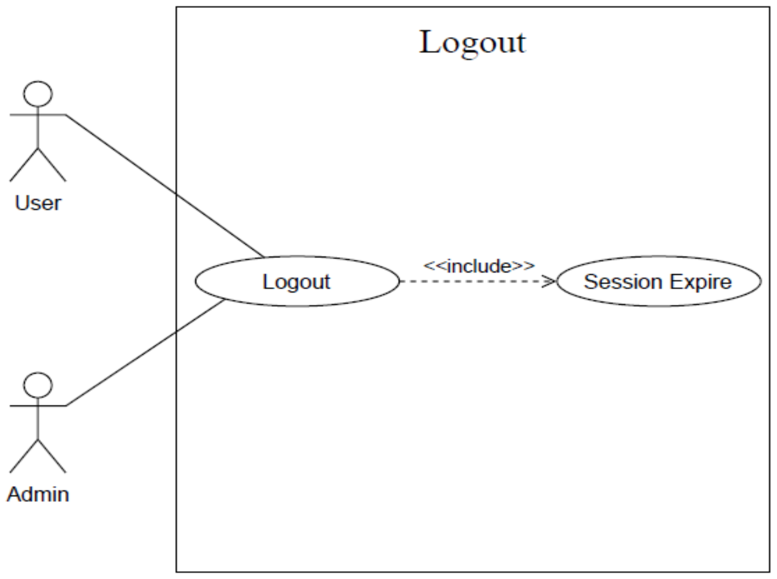


Figure 8: Use case-03: Logout

2.5.4 Use case-04: User Portal

The use case for the user portal shown in Figure 9. Users can interact with applications in multiple ways. First of all, they will register themselves. To get the recommendations, the user will enter its location and the whole process of giving recommendations will be performed. Users can view places and also give feedbacks against places, while the other options are related to account management.

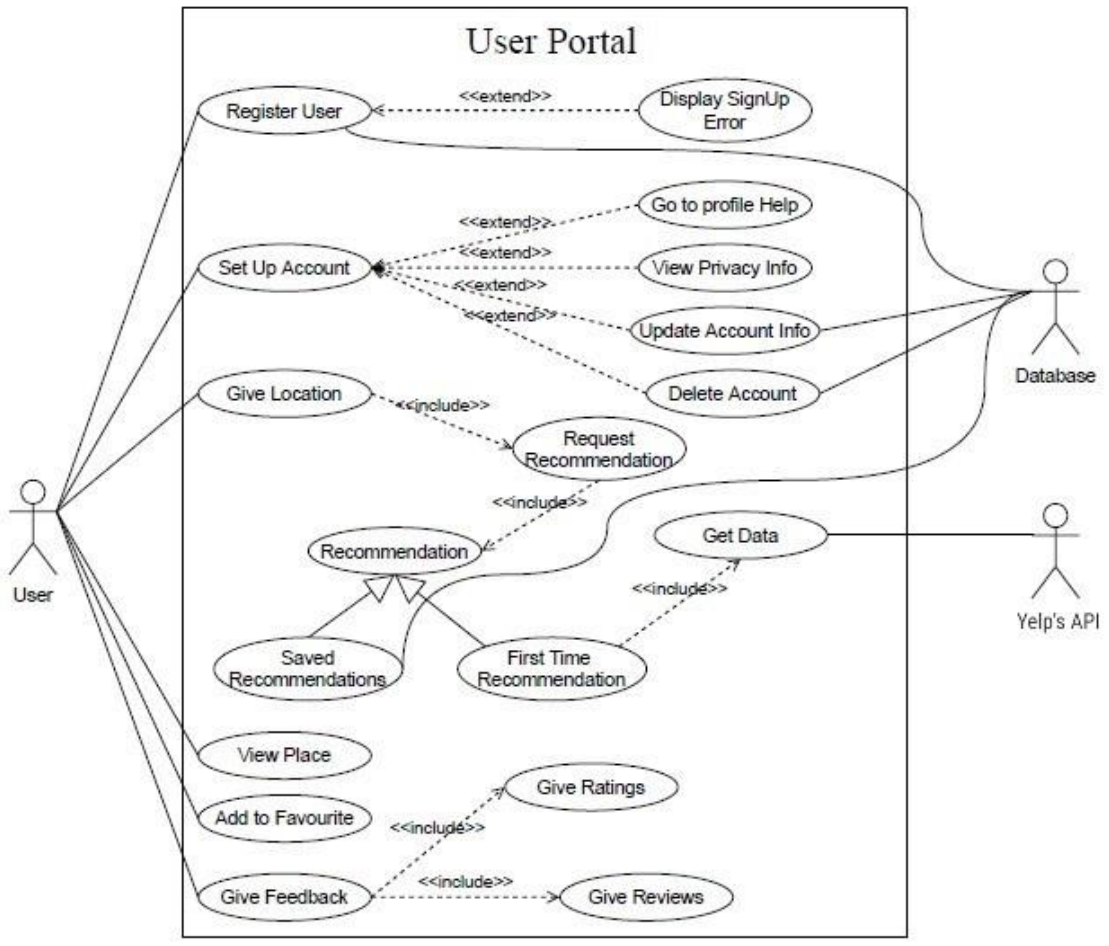


Figure 9: Use case-04: User Dashboard

2.5.5 Use case-05: Admin Portal

The use case for the admin portal is shown in Figure 10. Admin has authority to view users, update the status of any user or block any user from the application.

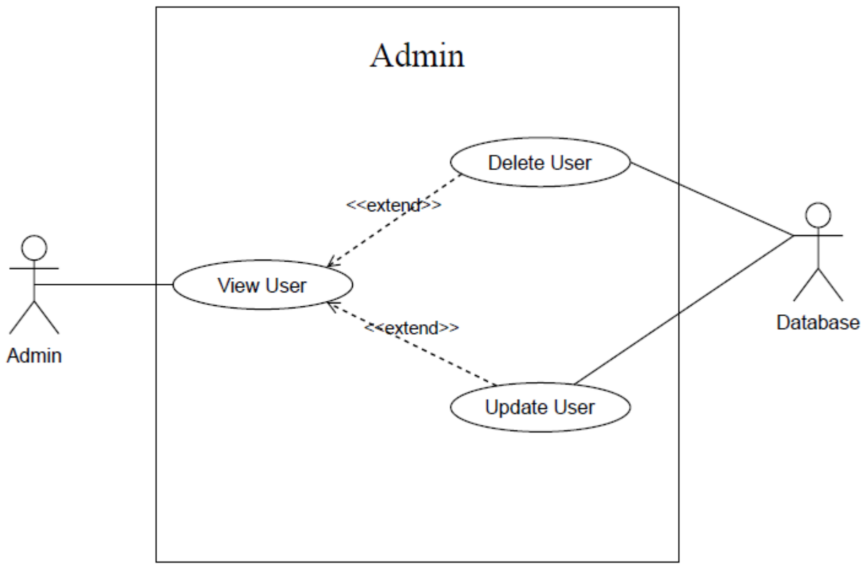


Figure 10: Use case-05: Admin Portal

2.5.6 Use case-06: System Recommendations

The use case for system recommendations place is shown in Figure 11. To give recommendations to the user, the system will obtain its location. Obtained location will be checked in the database, if data of that location is available, recommendations will be given according to system ratings of the places. But if the location is not there then we will fetch data from Yelp’s API. Afterward, the fetched data will be feed to the trained model, and ratings of places will be shown to user as well as stored in the database for latter recommendations.

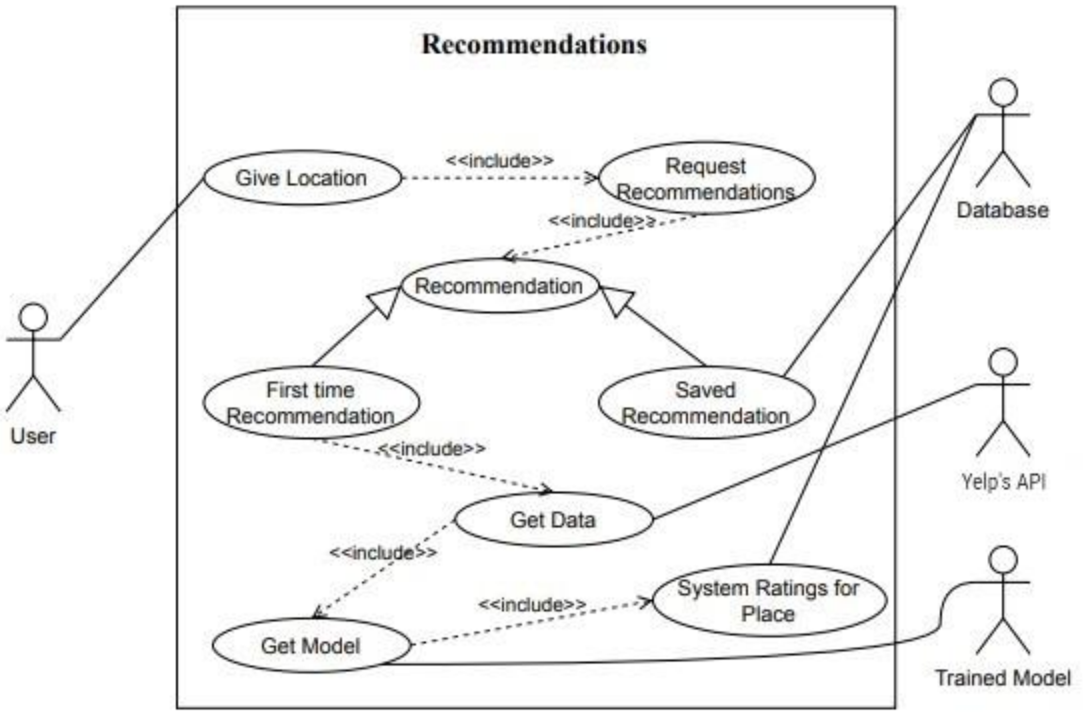


Figure 11: Use case-06: Recommendations

2.5.7 Use case: Recommendation System (Complete System)

The use case in figure 12 depicts the overview of the complete recommendation system by combining all modules of the system in one diagram for a better understanding of the viewer.

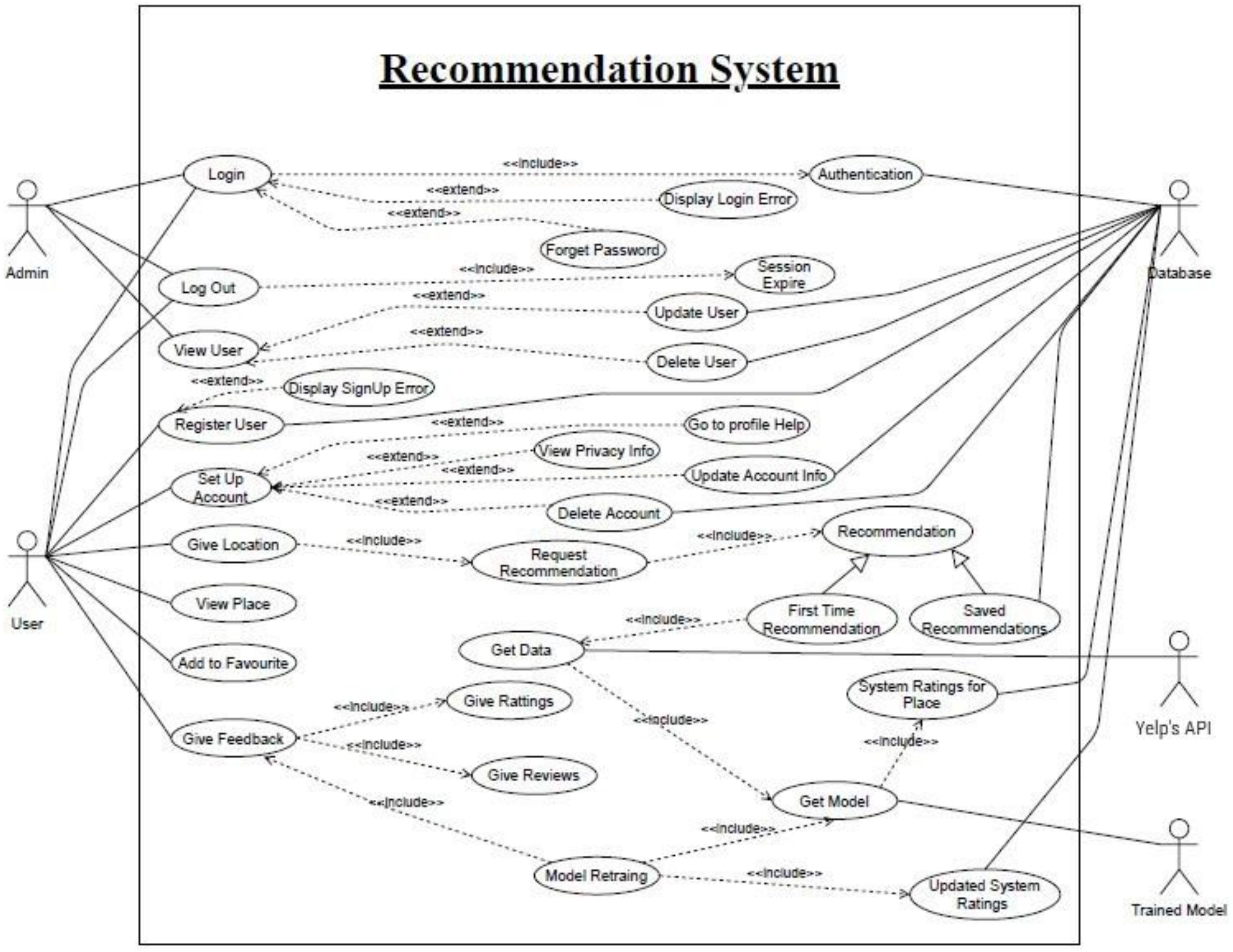


Figure 12: Use case: Recommendation System

2.6 Software development life cycle model

One of the essential thoughts of the product advancement measure is SDLC models which represents Software Development Life Cycle models. There are numerous development life cycle models that have been created to accomplish diverse required targets. The models determine the different phases of the interaction and the request where they are done. The most utilized, mainstream and significant SDLC models are given underneath:

- Waterfall model
- V model
- Incremental model
- RAD model
- Agile model
- Iterative model
- Spiral model
- Prototype model

2.6.1 Model Used in our project:

The incremental form model is a strategy for programming advancement where the model is planned, carried out and tried gradually (somewhat more is added each time) until the item is done. It includes both turn of events and support. The item is characterized as completed when it fulfils the entirety of its prerequisites. Every cycle goes through the prerequisites, plan, coding and testing stages. What's more, each ensuing arrival of the framework adds capacity to the past functionalities until all planned practically has been executed. This model joins the components of the waterfall model with the iterative way of thinking of prototyping.

2.6.2 Why?

- This model will be possible for us if there is any shift needed all through the task. Thus, it would be more sensible
- It is a lot simpler to recognize the dangers and handle them independently in the emphases
- Our venture would require amendments until we get our last undertaking.
- Testing and troubleshooting during the more modest cycles would be a superior alternative. Every build is an achieved handily in milestones.

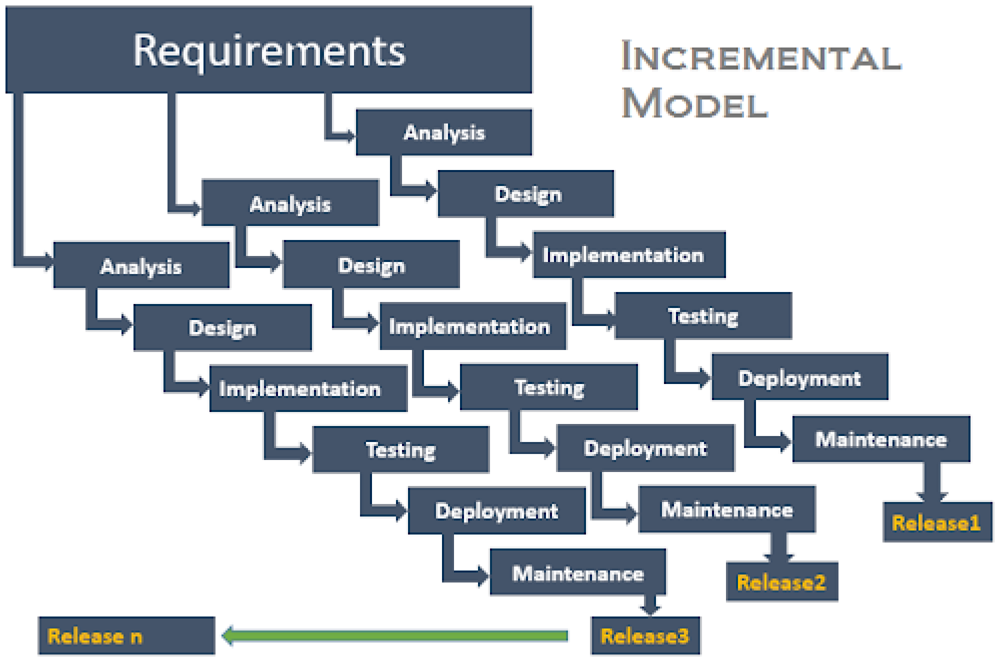


Figure 13: Incremental Model for software development life cycle

Figure 13 shows how incremental model works in different iterations. Each iteration is a bit more progressive at the beginning which makes the work increment in small builds and provides flexibility and efficiency to the model for risk/revisions analysis.

Chapter 3

System Design

3 System Design

3.1 Work breakdown structure (WBS)

Work breakdown structure of whole project has been shown in Figure 14. WBS describes different phases like model training, project management etc. so on till the deployment phase of project.

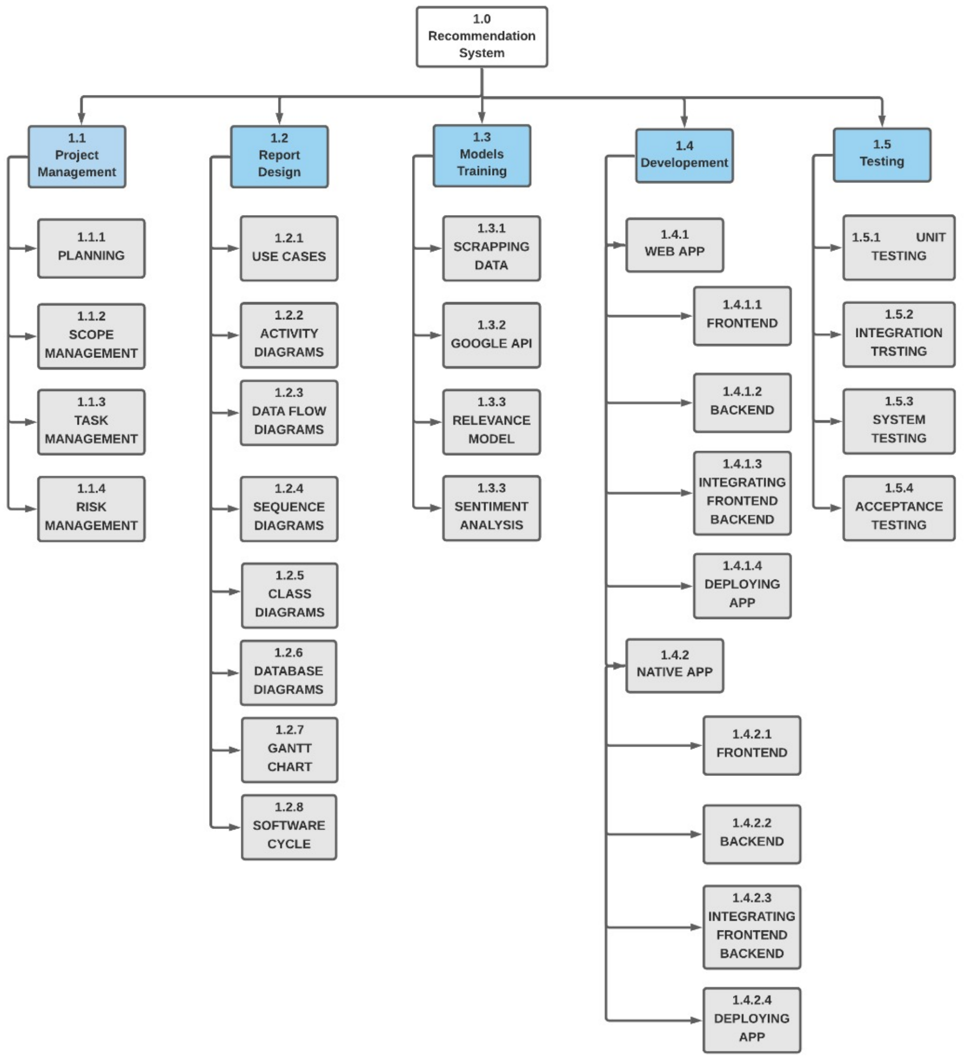


Figure 14: WBS

3.2 Activity diagram

3.2.1 Login

The flow of activities of login show in Figure 15. The users and admin will enter their credentials, the System will match them with the database and the appropriate function will run.

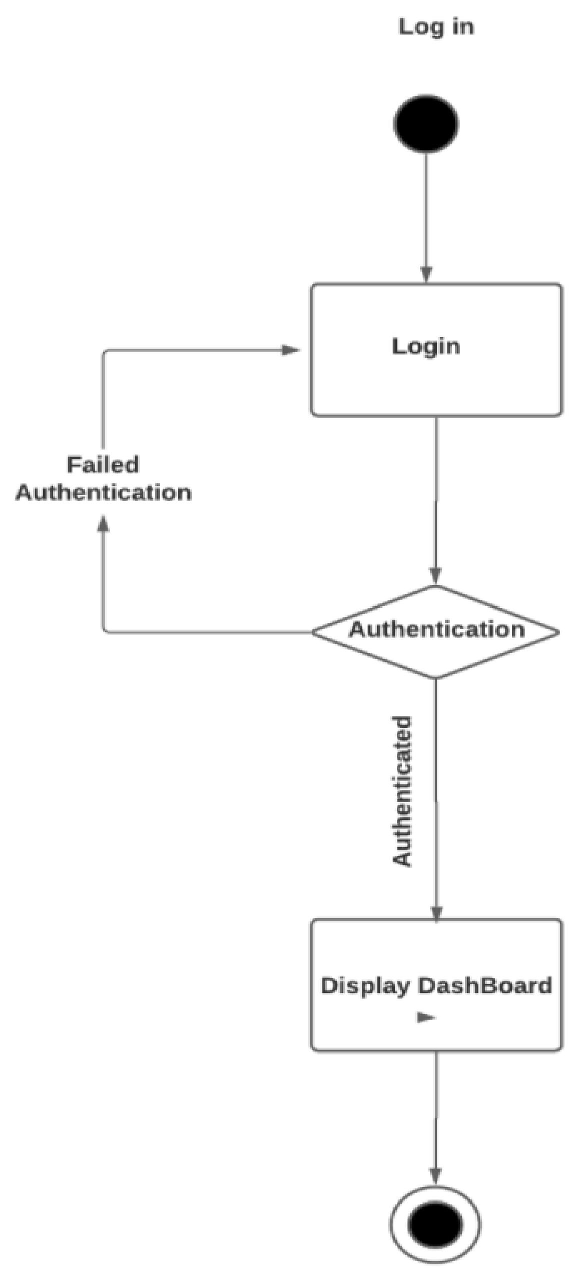


Figure 15: Activity Diagram 1: Login

3.2.2 Register User

The flow of activities of Register user show in Figure 16.

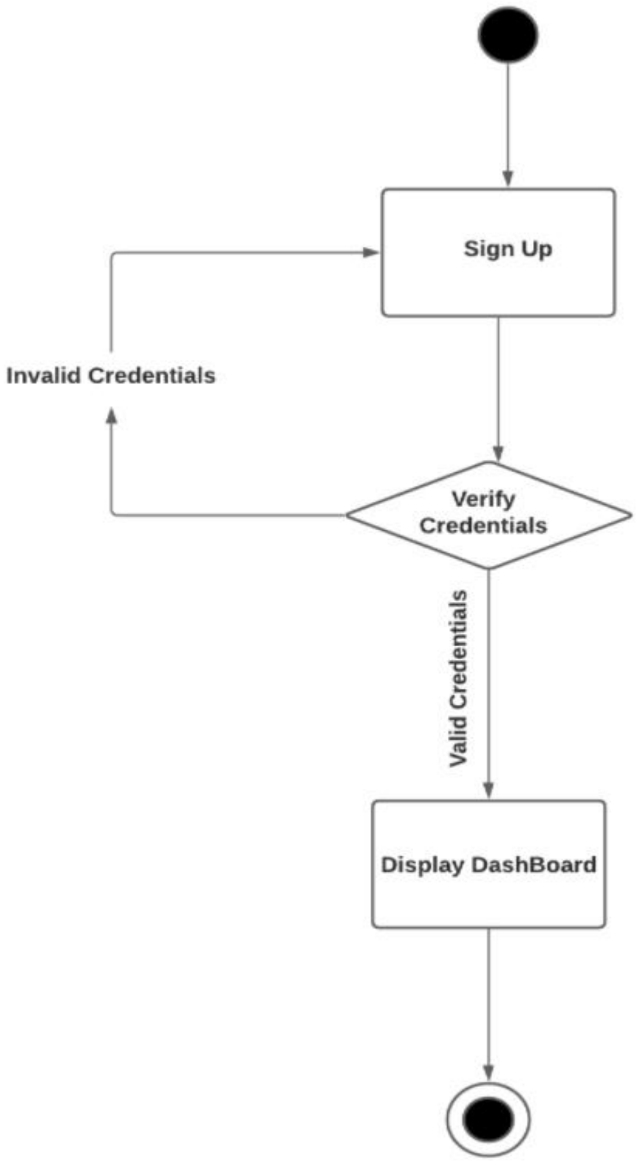


Figure 16: Activity Diagram 2: Register user

3.2.3 System Recommendations

The activity diagram of the flow of activities of user search result is shown in Figure 17. To give recommendations to the user the system will obtain its location. If the location there in database, places with system ratings will recommended otherwise Yelp's API data is used to recommendation.

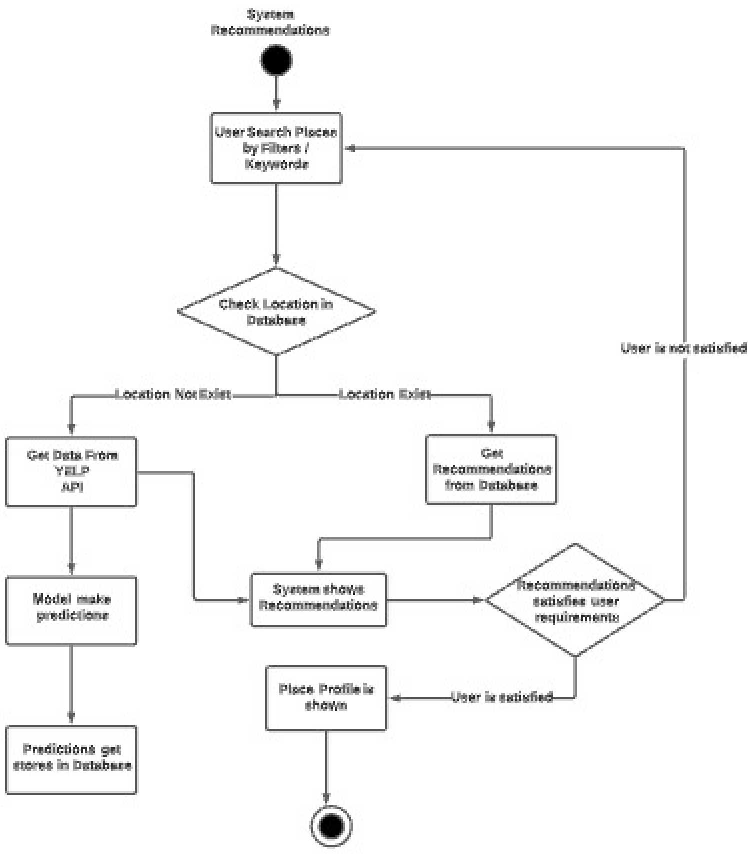

System Recommendations
User Search Places by Filters / Keywords
Check Location in Database
Location Not Exist
Location Exist
User is not satisfied
Get Data From YELP API
Get Recommendations from Database
Model make predictions
System shows Recommendations
Recommendations satisfies user requirements
Place Profile is shown
User is satisfied
Predictions get stores in Database

Figure 17: Activity Diagram 3: Recommendation

3.3 Sequence diagram

3.3.1 Login Sequence

Figure 18 shows sequence diagram of login, the figure describes each module's participation in sequence for login.

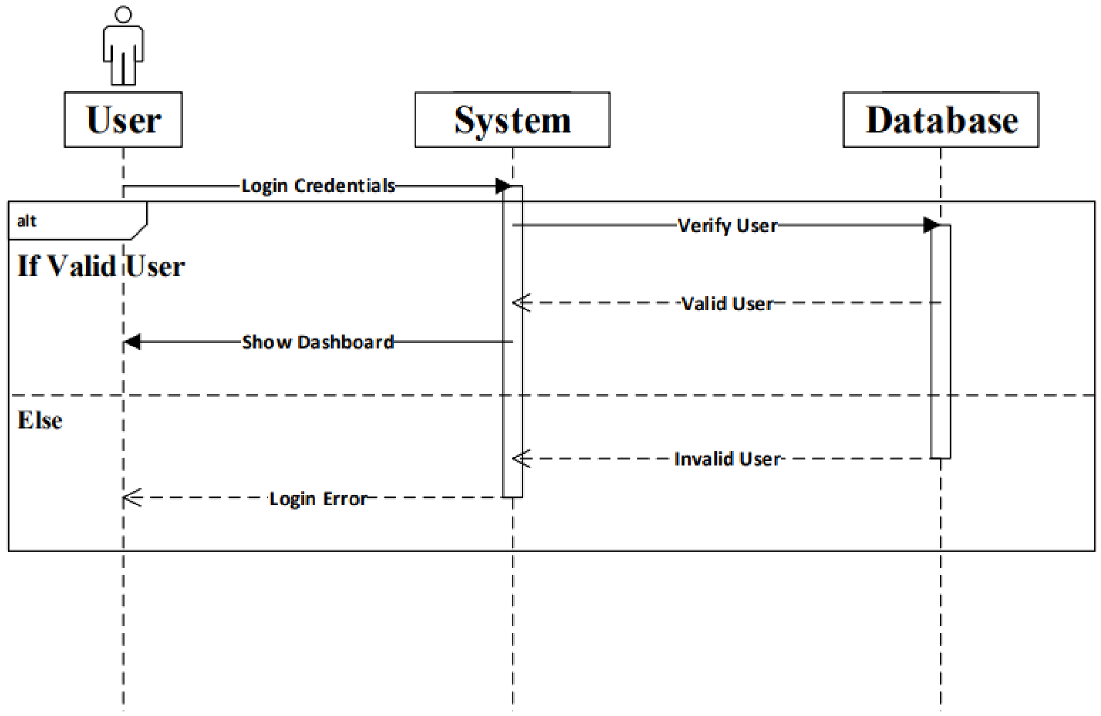


Figure 18: Sequence Diagram: Login

3.3.2 System Recommendations

Figure 19 shows the whole interaction of user with the application to perform different task. The figure contains the sequence and the activation of each module against the task.

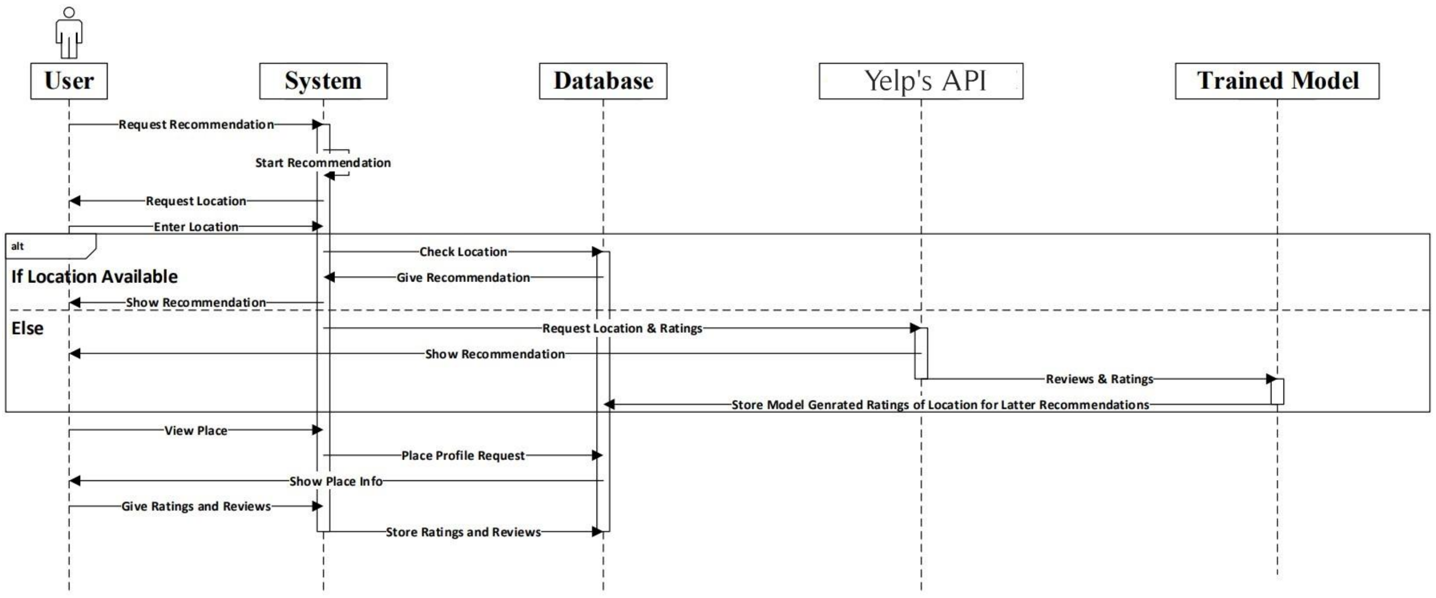


Figure 19: Sequence Diagram: System Recommendations

3.4 Software architecture

The user will log in to the application, to get the recommendations, he will enter his location, and then data will be fetched from the database and shown to the user. The user will give feedback against the recommended places, model will feed with new feedbacks and system ratings against places will update accordingly. In Figure 20 explained each of the phases.

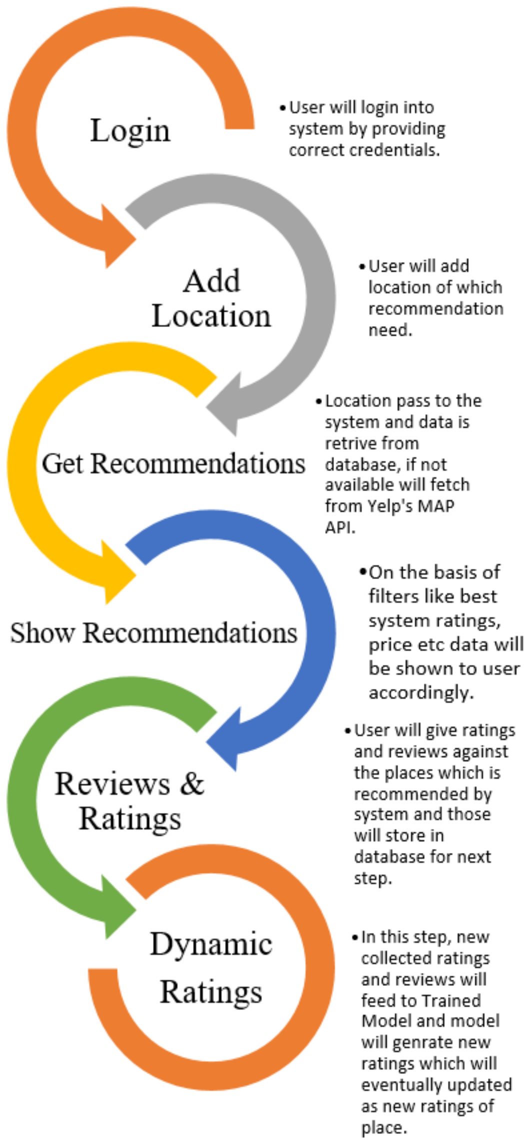

Login
• User will login into system by providing correct credentials.
Add Location
• User will add location of which recommendation need.
Get Recommendations
• Location pass to the system and data is retrive from database, if not available will fetch from Yelp's MAP API.
Show Recommendations
• On the basis of filters like best system ratings, price etc data will be shown to user accordingly.
Reviews & Ratings
• User will give ratings and reviews against the places which is recommended by system and those will store in database for next step.
Dynamic Ratings
• In this step, new collected ratings and reviews will feed to Trained Model and model will genrate new ratings which will eventually updated as new ratings of place.

Figure 20: Software Architecture

3.5 Class diagram

Figure 21 shows the class diagram, depicts system's frontend that how each entity is in relation with the other.

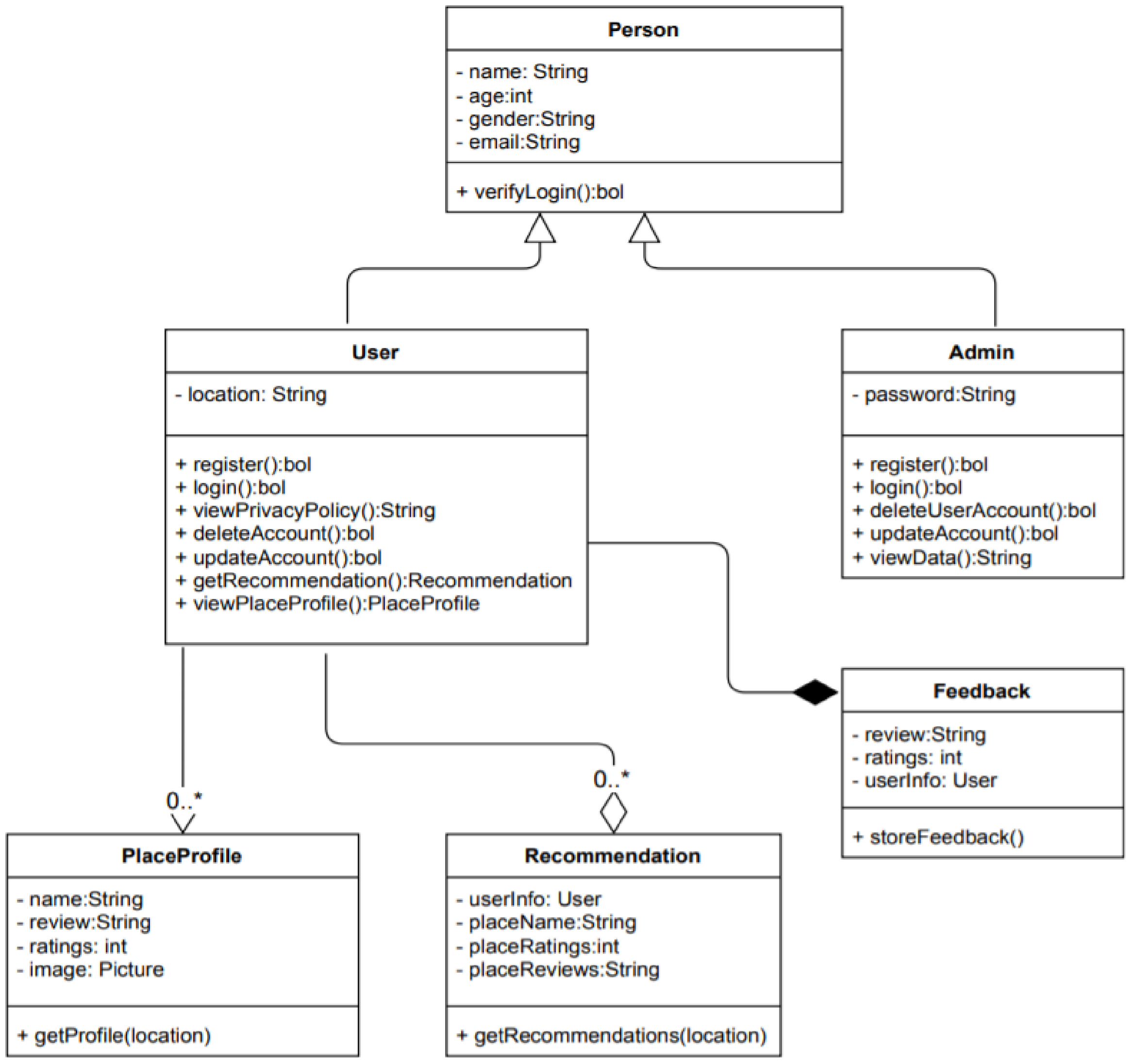


Figure 21: Class Diagram

3.6 Database diagram

Figure 22 shows the database diagram, depicts system's backend that how each table is in relation with the other.

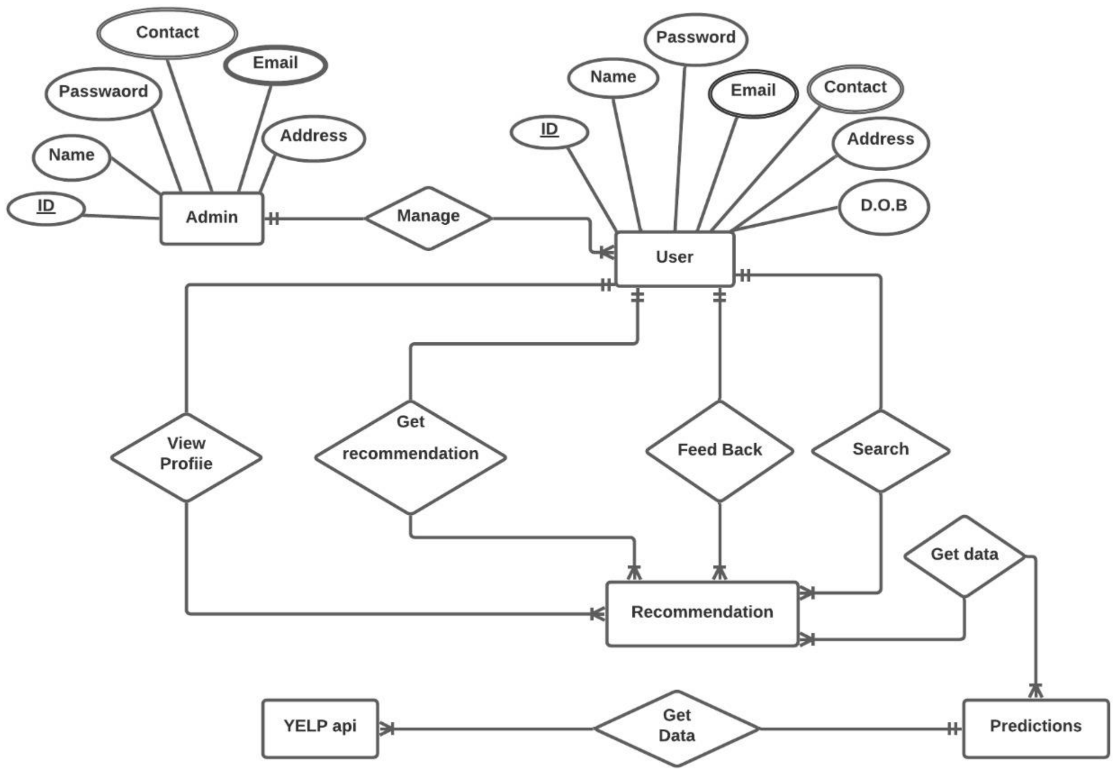
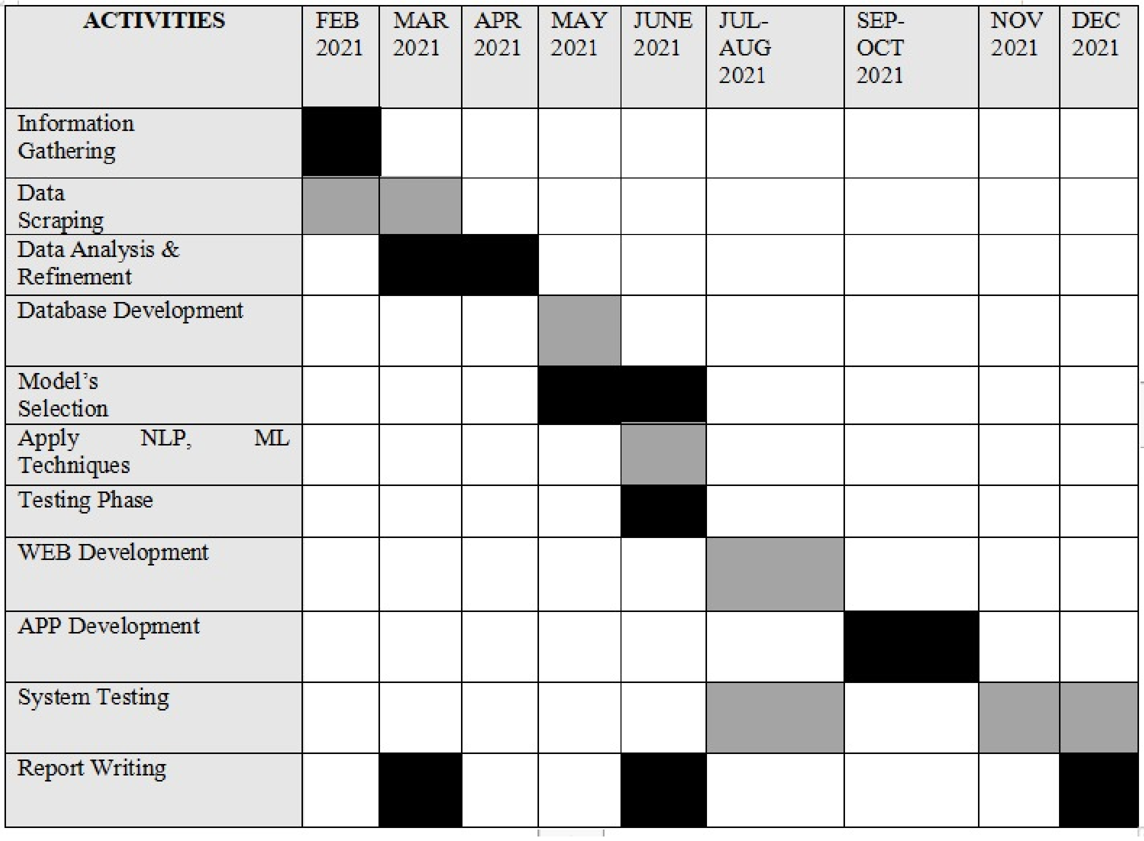


Figure 22: Database Diagram

3.7 Network diagram (Gantt chart)

Figure 23 graphically shows each task of the project and the duration in which that will be done. In parallel, tasks are mentioned, and against each of the shaded areas shows the duration.

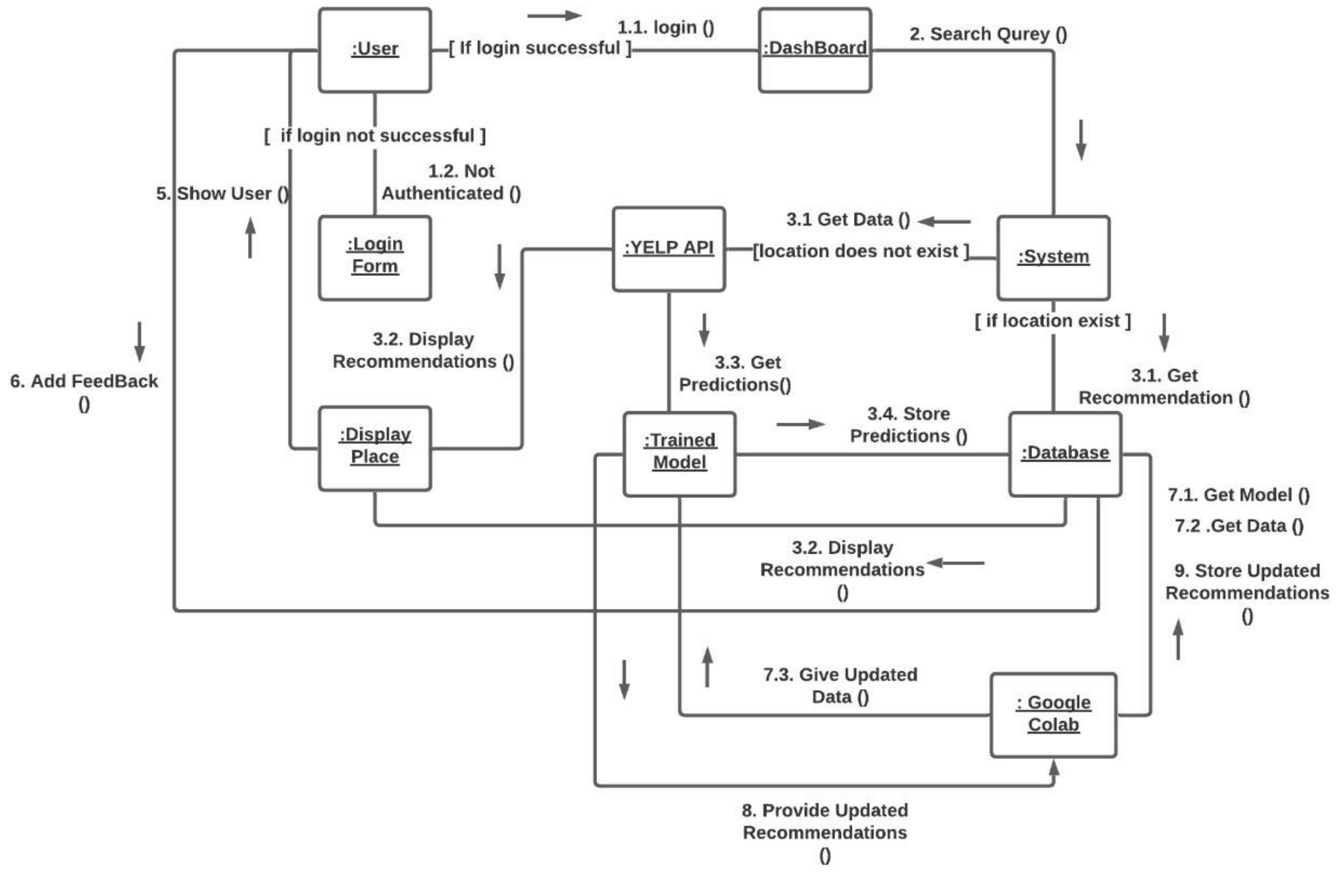


ACTIVITIES	FEB 2021	MAR 2021	APR 2021	MAY 2021	JUNE 2021	JUL-AUG 2021	SEP-OCT 2021	NOV 2021	DEC 2021
Information Gathering	■								
Data Scraping	▨	▨							
Data Analysis & Refinement		■	■						
Database Development				▨					
Model's Selection				■	■				
Apply NLP, ML Techniques					▨				
Testing Phase					■				
WEB Development						▨			
APP Development							■		
System Testing						▨		▨	▨
Report Writing		■			■				■

Figure 23: Network Diagram

3.8 Collaboration diagram

Figure 24 shows the collaboration diagram for Cleverus. It explains the flow from the user to the backend processes of the application.



:User
1.1. login ()
[If login successful]
:DashBoard
2. Search Qurey ()
[if login not successful]
1.2. Not Authenticated ()
5. Show User ()
:Login Form
:YELP API
3.1 Get Data ()
[location does not exist]
:System
3.2. Display Recommendations ()
[if location exist]
3.3. Get Predictions()
6. Add FeedBack ()
3.1. Get Recommendation ()
:Display Place
:Trained Model
3.4. Store Predictions ()
:Database
7.1. Get Model ()
7.2 .Get Data ()
3.2. Display Recommendations ()
9. Store Updated Recommendations ()
7.3. Give Updated Data ()
: Google Colab
8. Provide Updated Recommendations ()

Figure 24: Collaboration Diagram

Chapter 4

System Testing

4 System Testing

4.1 Unit Testing

In unit testing, every one of the segments of the application are exclusively tested. It is the first and most key piece of testing. In this progression, we checked the individual components of our project that execute unit errands and are segments of the task's entire work process.

4.1.1 Test Case 01: Login

The test case in table 23 shows the test case for the Functional Requirement 02: Login. The table shows that what users can possibly do and what might go wrong if not tested. Aim of the test is to log in with the correct login credentials.

Table 22: Test case-01

Test case Name	TC-01
Application Name	Recommendation System
Use Case	Login
Input Summary	User will enter its credentials and click the login button.
Output Summary	SUCCESS: User has been logged in successfully. FAILURE (Expected): The access has been denied and an error message will be displayed i.e. “Either username or password is incorrect.”
Pre-Conditions	User must enter correct login credentials
Post-Conditions	User logged in and navigate to its user portal.

4.1.2 Test Case 02: Recommendation

The test case in table 24 shows the test case for the Functional Requirement 06: Login. Aim of the test is to show recommendations to user, if the given location is correct.

Table 23: Test case-02

Test case Name	TC-02
Application Name	Recommendation System
Use Case	Recommendation
Input Summary	User will select the category and enter its location or give access to system's location.
Output Summary	SUCCESS: Recommendations will be shown to user. FAILURE (Expected): There are multiple errors which may occur. • Invalid location • Server's access in area etc. In each case error or problem message related to it will be displayed.
Pre-Conditions	User must enter correct location.
Post-Conditions	Appropriate recommendations will be shown to user.

4.1.3 Test Case 03: Feedback

The test case in table 25 shows the test case for giving feedback against place. Aim of the test is to show save valid feedback of user, if both reviews and ratings are provided.

Table 24: Test case-03

Test case Name	TC-03
Application Name	Recommendation System
Use Case	Feedback
Input Summary	User will click on the “Give feedback” button and after that user will enter feedback.
Output Summary	SUCCESS: User’s feedback will store in system. FAILURE (Expected): There can be 2 types of error • Either reviews or ratings are missing. In this case appropriate error message will be shown to user.
Pre-Conditions	User must enter all and correct parameters to give feedback.
Post-Conditions	User’s feedback will be stored and success message will be displayed.

4.1.4 Test Case 04: View Place Profile

The test case in table 26 shows the test case for the Functional Requirement 07: View Place Profile. Aim of the test is to show place pictures, reviews and ratings.

Table 25: Test case-04

Test case Name	TC-04
Application Name	Recommendation System
Use Case	View Place Profile
Input Summary	User will select the category and search the business in the search bar and then click on the business after getting recommendation.
Output Summary	SUCCESS: Business profile will be shown to user. FAILURE (Expected): Error reasons might be “Business not exist”. In that case appropriate message will be shown to user.
Pre-Conditions	User must search for valid place/business name.
Post-Conditions	Business profile will be shown to user.

4.1.5 Test Case 05: Model Retraining

The test case in table 27 shows the test case for the retraining of model. Aim of the test is to retrain model over new feedbacks and update the ratings of the place.

Table 26: Test case-05

Test case Name	TC-05
Application Name	Recommendation System
Use Case	Model Retraining
Input Summary	New user feedback and trained model will be accessed on Google Colab by Admin. Then trained model will be feed in the updated user feedbacks.
Output Summary	SUCCESS: Model has been trained successfully on updated recommendations and then these recommendations will be stored in database. FAILURE (Expected): There were problems with either user feedbacks or with the train model.
Pre-Conditions	Updated feedbacks must be available for retraining.
Post-Conditions	Model has been trained successfully on updated user feedbacks and updated recommendations are stored in database.

4.2 Integration Testing

Integration testing is the second step of the product testing strategy. At this degree of testing, the framework is tried subsequent to joining the different units into bunches. It is to test the deficiencies and blunders in the collaboration between the interacted units.

We have integrated following modules and test them:

- Connecting the BERT with Flask to utilize BERT.
- Connecting of Flask with MongoDB.
- Connectivity of Flask APIs with web application and mobile application.
- Connectivity of Firebase with mobile and web application for authentication purposes.

After detailed testing of each integrated par, we have successfully completed the integration testing.

4.3 Acceptance Testing

Acceptance testing is characterized as the last advance step for the product testing system. It is to administer whether the necessary details of the framework are met. At this progression, we gauge whether the framework under test is finished with the nuts and bolts and necessities for conclusive handling. We have done this progression toward the finish of the relative multitude of cycles to furnish a framework with appropriate details.

We have tested our project against following scenarios:

- Only authorized user will able to login and reviews and ratings are provided by only those users who have the account.
- Generating ratings on the basis of reviews and ratings.
- Search via location name or by map drag working properly.
- Data correctly stored and retrieved from database.
- Correct responses from APIs.
- Web and Mobile application’s worked in the same flow as it is expected.

After a detailed development process in which on each module we have worked in the best of the ways we could, now our system fulfil the above-mentioned things.

Chapter 5

Application End

5 Application End

In this section screenshots of our recommendation system's web and mobile applications are attached, showing the UI of our system.

5.1 Web Application User Interface

User interface of web application is made using React JS.

5.1.1 Login Screen

Figure 25 shows the login page of web application. Here user can either login by giving correct credentials or can create new account.

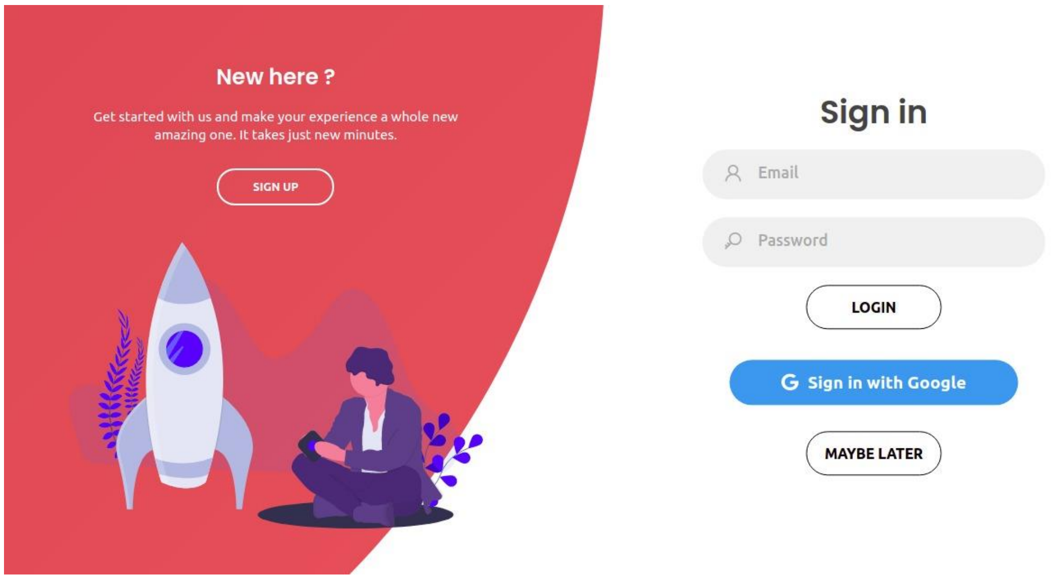
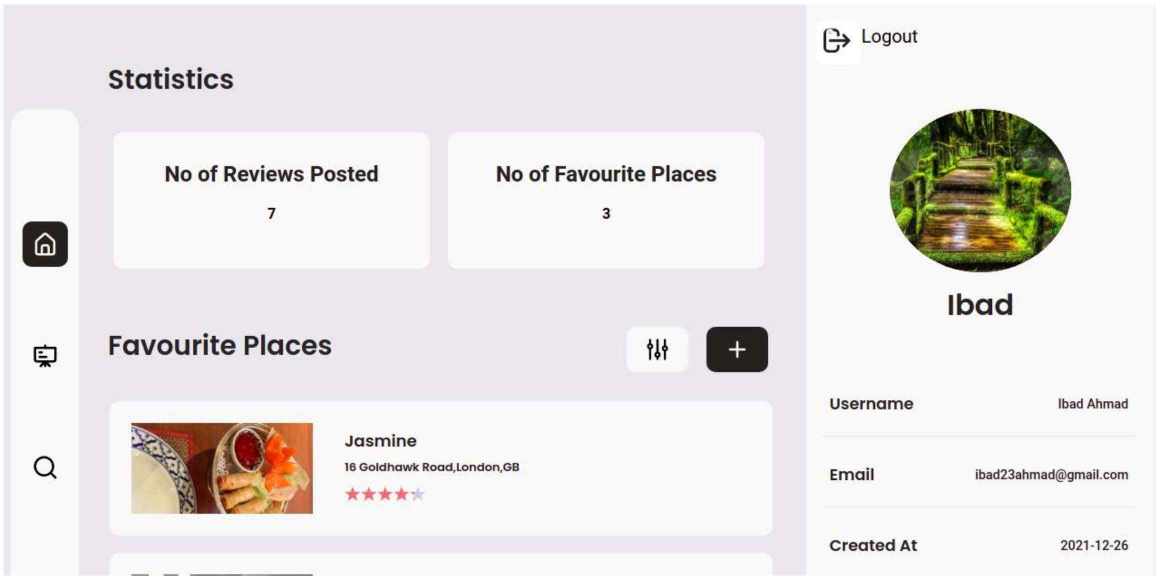

New here ?
Get started with us and make your experience a whole new amazing one. It takes just new minutes.
SIGN UP
Sign in
Email
Password
LOGIN
Sign in with Google
MAYBE LATER

Figure 25: Web Application's Login Page

5.1.2 User's Dashboard

Figure 26 shows the user's dashboard. Where user can change profile picture, check his/her liked places and can do some other stuff.


Statistics

No of Reviews Posted
7

No of Favourite Places
3

Favourite Places

Jasmine
16 Goldhawk Road,London,GB

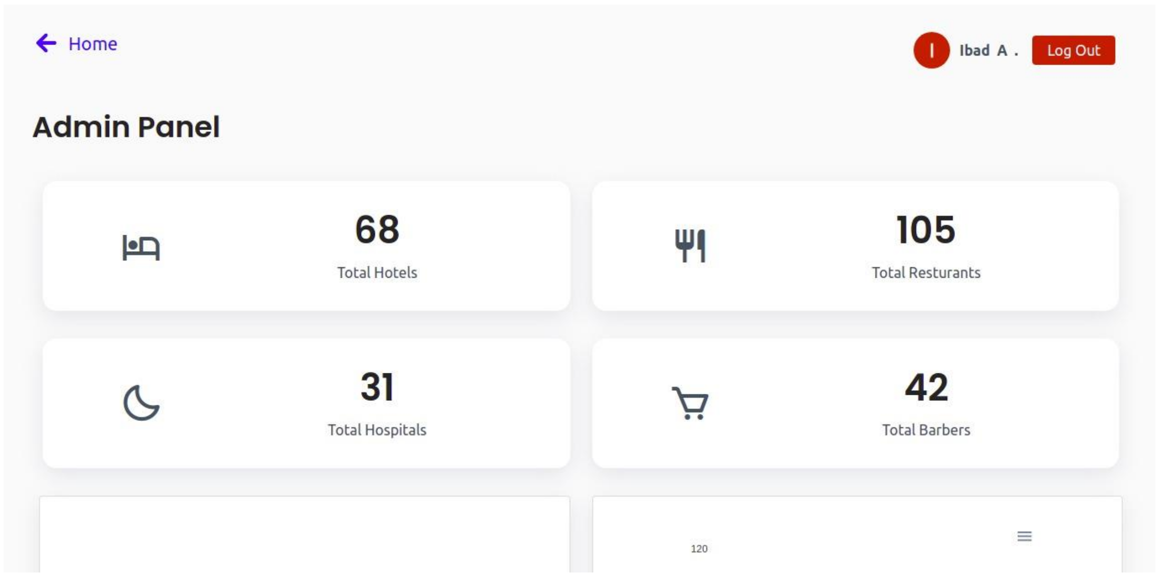
Ibad

Username	Ibad Ahmad
Email	ibad23ahmad@gmail.com
Created At	2021-12-26

Figure 26: User's Dashboard

5.1.3 Admin's Dashboard

Figure 27 and 28 shows the admin panel of application, where admin can see stats related to application and can perform some admin level tasks.


Admin Panel

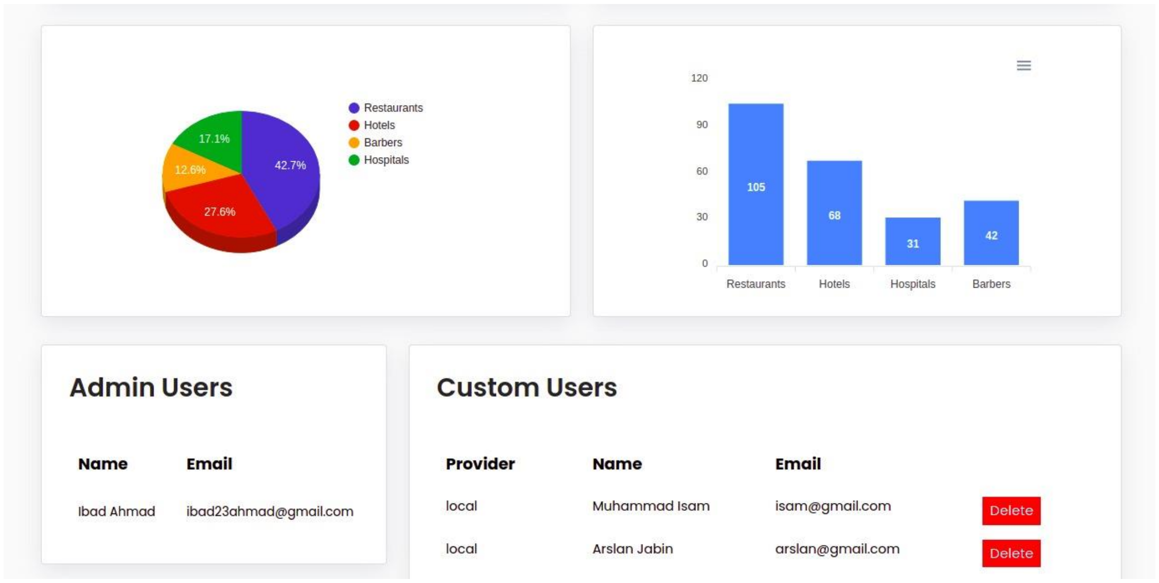
68
Total Hotels

105
Total Resturants

31
Total Hospitals

42
Total Barbers

Figure 27: Admin's Dashboard



Admin Users

Name	Email
Ibad Ahmad	ibad23ahmad@gmail.com

Custom Users

Provider	Name	Email	
local	Muhammad Isam	isam@gmail.com	Delete
local	Arslan Jabin	arslan@gmail.com	Delete

Figure 28: Admin Panel (cont.)

5.1.4 Home Page

Figure 29 and 30 shows the home page of application where user will engage mostly with the main and essential features of our app.

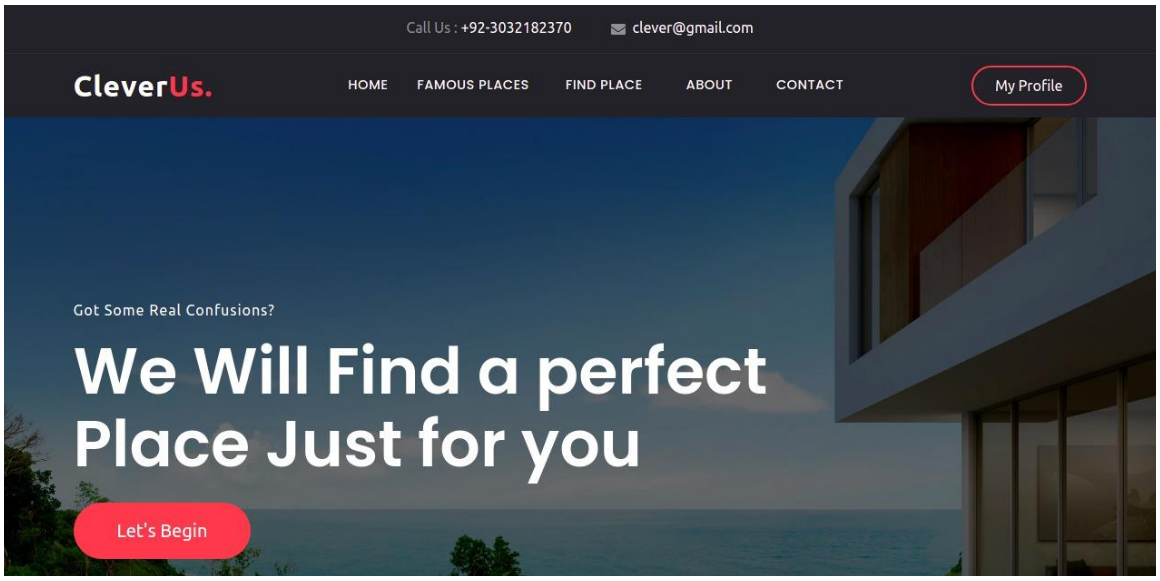
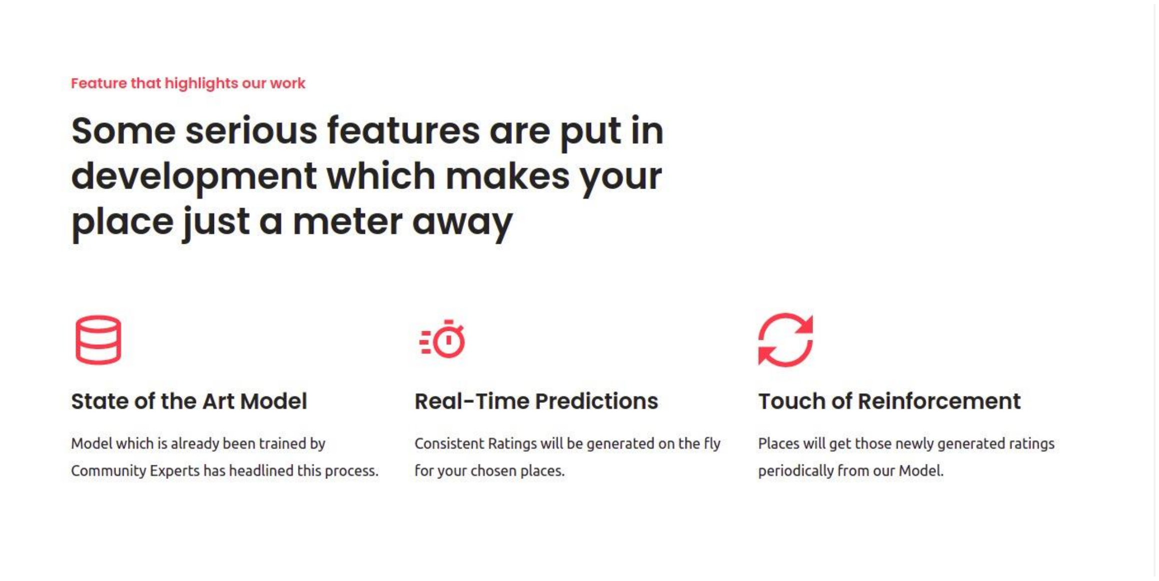


Figure 29: Web Application's Home Page



Feature that highlights our work

Some serious features are put in development which makes your place just a meter away

State of the Art Model

Model which is already been trained by Community Experts has headlined this process.

Real-Time Predictions

Consistent Ratings will be generated on the fly for your chosen places.

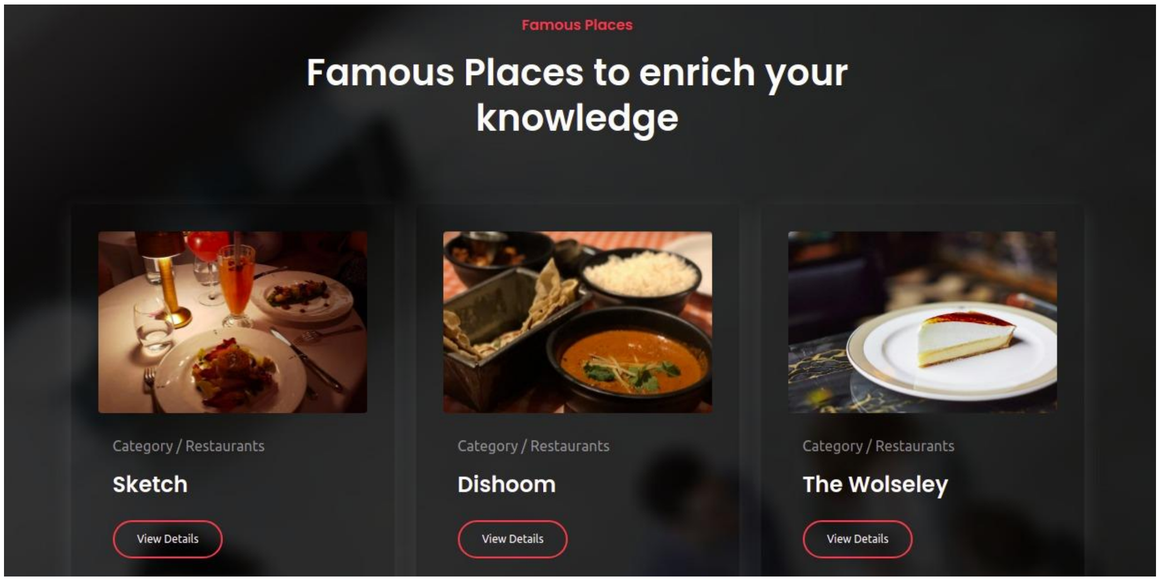
Touch of Reinforcement

Places will get those newly generated ratings periodically from our Model.

Figure 30: Web Application's Home Page (cont.)

5.1.5 Popular Places

Figure 31 shows the popular places page, where the places against most comments has been shown to users.



Famous Places

Famous Places to enrich your knowledge

Category / Restaurants

Sketch

View Details

Category / Restaurants

Dishoom

View Details

Category / Restaurants

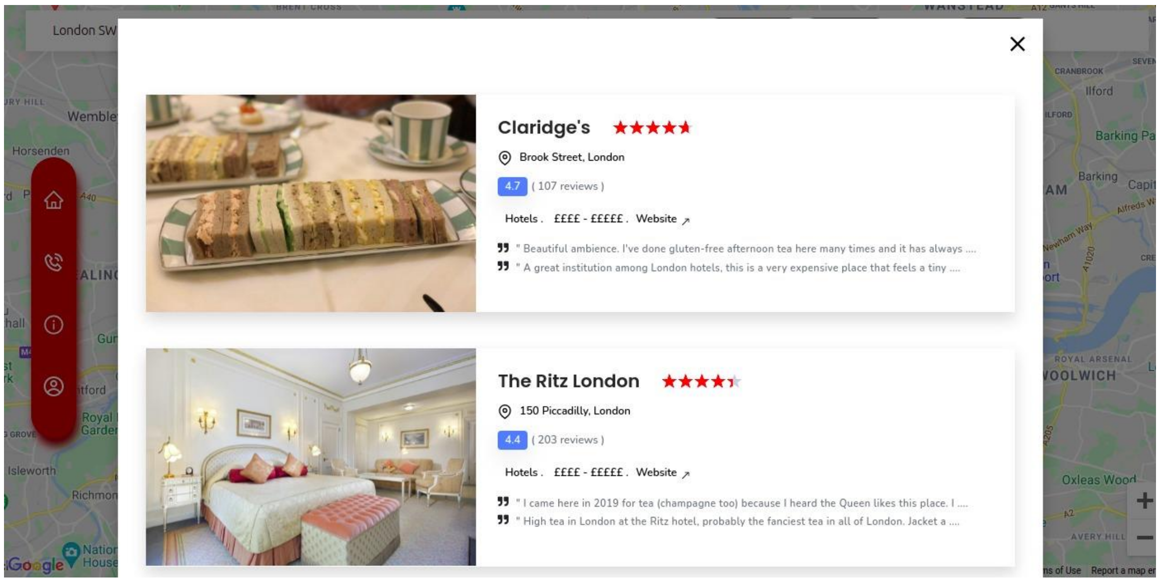
The Wolseley

View Details

Figure 31: Popular Places Page

5.1.6 Recommendation Page

Figure 32 shows the most important page of application. Where user can select the location and module (of which he/she wants the recommendation) and basis on that recommendations will be displayed to user.


Claridge's

Brook Street, London

4.7 (107 reviews)

Hotels . ££££ - £££££ . Website

" Beautiful ambience. I've done gluten-free afternoon tea here many times and it has always ....

" A great institution among London hotels, this is a very expensive place that feels a tiny ....

The Ritz London

150 Piccadilly, London

4.4 (203 reviews)

Hotels . ££££ - £££££ . Website

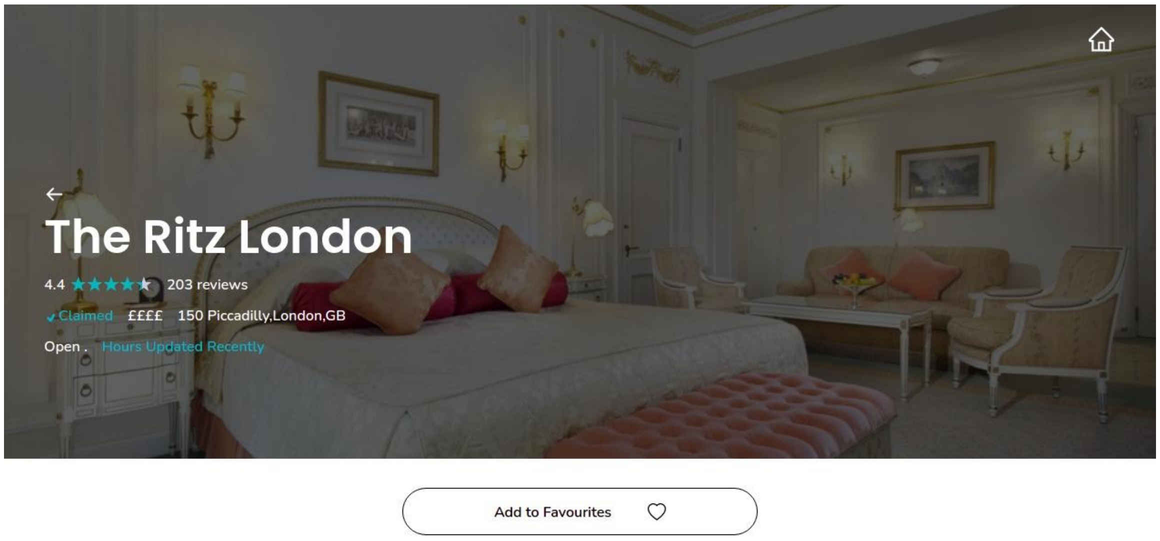
" I came here in 2019 for tea (champagne too) because I heard the Queen likes this place. I ....

" High tea in London at the Ritz hotel, probably the fanciest tea in all of London. Jacket a ....

Figure 32: Recommendation Page

5.1.7 Place's Detail Page

Figure 33 and 34 shows the place's detail page, in which user can check the comments of other user, can add place to its fav bucket or can add new comment. Moreover, other information related to place is also provided to user.

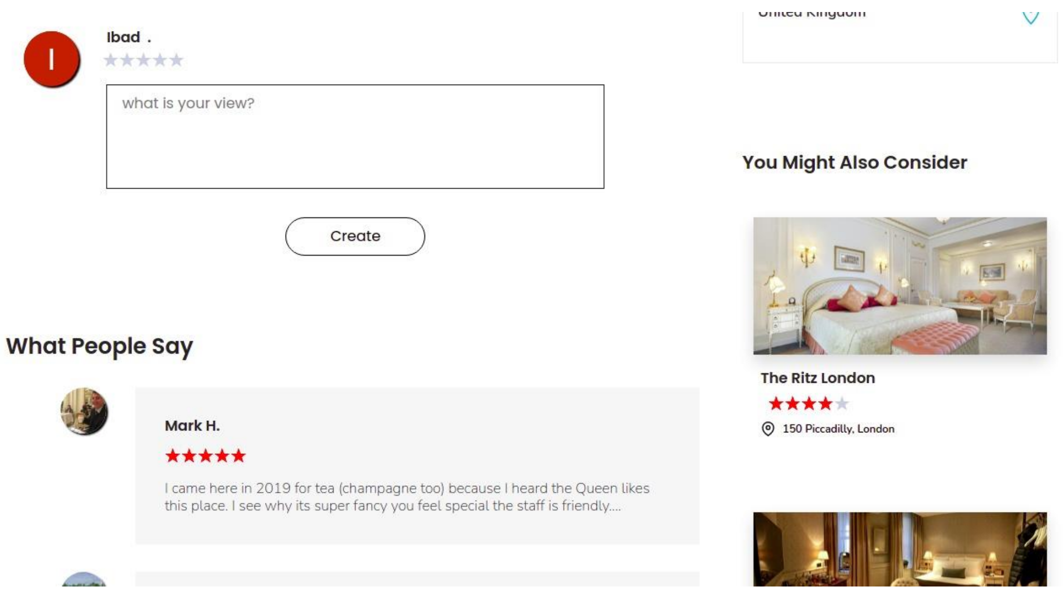

The Ritz London

4.4 203 reviews

Claimed ££££ 150 Piccadilly,London,GB

Open . Hours Updated Recently

Figure 33: Place's Detail Page



I came here in 2019 for tea (champagne too) because I heard the Queen likes this place. I see why its super fancy you feel special the staff is friendly....

Figure 34: Place's Detail Page (cont.)

5.2 Mobile Application User Interface

User interface of mobile application is made using React Native.

5.2.1 Start Page

Figure 35 shows the start page of application, which will open when user opens the application.

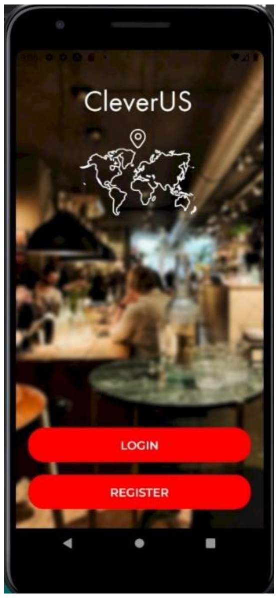


Figure 35: Start Page of Mobile Application

5.2.2 Sign Up Page

Figure 36 shows the sign up page of application, where user can creates his/her account.

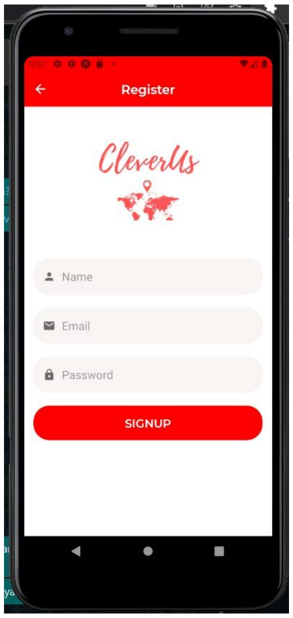


Figure 36: Mobile Application Sign Up Page

5.2.3 Login Page

Figure 37 shows login page of application, if user already has account, he/she can simply login to account by giving correct credentials.

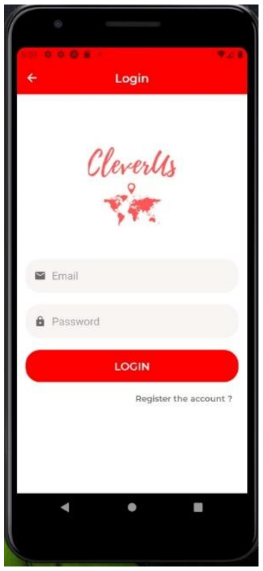
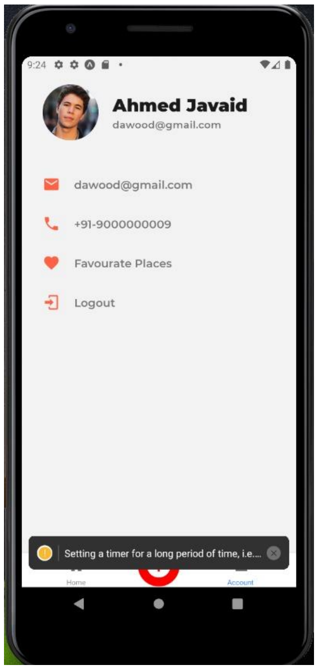


Figure 37: Mobile Application Login Page

5.2.4 User's Information Page

Figure 38 shows the basic user information page, where user can change his/her profile, check fav places and can logout from application.



Setting a timer for a long period of time, i.e....

Figure 38: Mobile Application User Info Page

5.2.5 Recommendation Page

Figure 39 and 40 shows the recommendation page of application, where user selects module and location and on the basis of that recommendations have given to user.

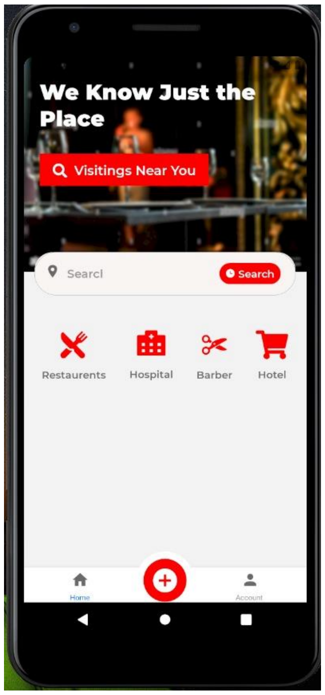


Figure 39: Mobile Application Recommendation Page

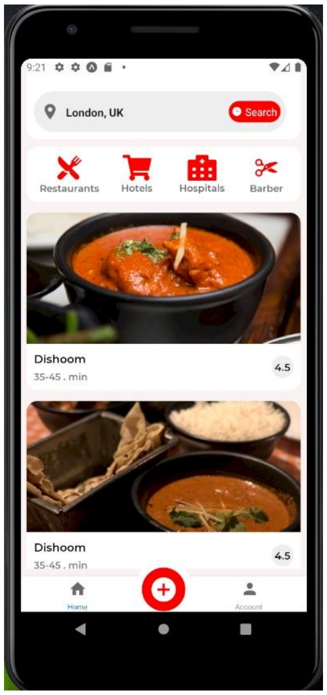
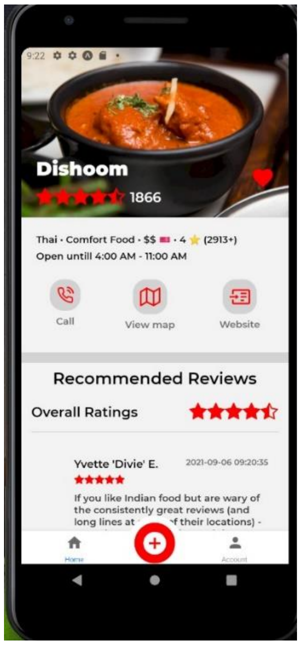


Figure 40: Mobile Application Recommendation Page (cont.)

5.2.6 Place’s Detail Page

Figure 41 and 42 shows the place’s detail page, in which user can check the comments of other user, can add place to its fav bucket or can add new comment. Moreover, other information related to place is also provided to user.

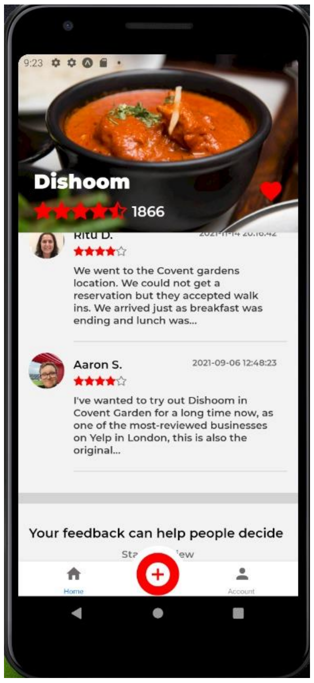


Thai · Comfort Food · $$ · 4 (2913+)
Open untill 4:00 AM - 11:00 AM

Recommended Reviews
Overall Ratings

Yvette 'Divie' E. 2021-09-06 09:20:35
If you like Indian food but are wary of the consistently great reviews (and long lines at of their locations) -

Figure 41: *Mobile Application Place’s Detail Page*



Ritu D. 2021-11-14 20:16:42
We went to the Covent gardens location. We could not get a reservation but they accepted walk ins. We arrived just as breakfast was ending and lunch was...

Aaron S. 2021-09-06 12:48:23
I've wanted to try out Dishoom in Covent Garden for a long time now, as one of the most-reviewed businesses on Yelp in London, this is also the original...

Your feedback can help people decide
Sta ew

Figure 42: Mobile Application Place's Detail Page (cont.)

Chapter 6

Project Code

6 Project Code

Our project's backend code is written in Python, and for UI/UX we have used React JS and React Native. But the backbone of the project is Python's code, which can be divided into following parts:

- BERT Model
- Generate Ratings
- Flask's APIs

6.1 BERT Model

BERT Model is used to predict the sentiment score of reviews, which help to create the intelligent ratings of the place. Reviews have been given to BERT model and in the result BERT returns the sentiment score against each review.

First review has been cleaned and after that we initialize BERT and set its parameters. After that BERT Classifier class is created which will use afterwards. Then BERT prediction method is created, which take the safe model as parameter and predict the sentiment analysis. Now by using that method we will predict the sentiment scores.

```
def text_preprocessing(text):
    # Remove '@name'
    text = re.sub(r'(@.*?)[\s]', ' ', text)

    # Replace '&amp;' with '&'
    text = re.sub(r'&amp;', '&', text)

    # Remove trailing whitespace
    text = re.sub(r'\s+', ' ', text).strip()


    return text

def initialize_model(epochs=4):
    """Initialize the Bert Classifier, the optimizer and the learnin
g rate scheduler.
    """
    # Instantiate Bert Classifier
    bert_classifier = BertClassifier(freeze_bert=False)

    # Tell PyTorch to run the model on GPU
    bert_classifier.to(device)
    # Create the optimizer
    optimizer = AdamW(bert_classifier.parameters(),
                      lr=5e-5,    # Default learning rate
                      eps=1e-8    # Default epsilon value
                      )

    # Total number of training steps
    total_steps = len(train_dataloader) * epochs

    # Set up the learning rate scheduler
    scheduler = get_linear_schedule_with_warmup(optimizer,
                                                num_warmup_steps=0,
# Default value
                                                num_training_steps=t
otal_steps)
    return bert_classifier, optimizer, scheduler

# Create a function to tokenize a set of texts
def preprocessing_for_bert(data):

    # Create empty lists to store outputs
    input_ids = []
    attention_masks = []

    # For every sentence...
    for sent in data:
        encoded_sent = tokenizer.encode_plus(
            text=text_preprocessing(sent),  # Preprocess sentence
            add_special_tokens=True,        # Add `[CLS]` and `[SEP]
`
            max_length=MAX_LEN,
            pad_to_max_length=True,
            return_attention_mask=True      # Return attention mask
            )

        # Add the outputs to the lists
        input_ids.append(encoded_sent.get('input_ids'))
        attention_masks.append(encoded_sent.get('attention_mask'))

    # Convert lists to tensors
    input_ids = torch.tensor(input_ids)
    attention_masks = torch.tensor(attention_masks)

    return input_ids, attention_masks
```

```
# Create the BertClassfier class
class BertClassifier(nn.Module)
    def __init__(self, freeze_bert=False):

        super(BertClassifier, self).__init__()
        # Specify hidden size of BERT, hidden size of our classifier
, and number of labels
        D_in, H, D_out = 768, 50, 2

        # Instantiate BERT model
        self.bert = BertModel.from_pretrained('bert-base-uncased')

        # Instantiate an one-layer feed-forward classifier
        self.classifier = nn.Sequential(
            nn.Linear(D_in, H),
            nn.ReLU(),
            #nn.Dropout(0.5),
            nn.Linear(H, D_out)
        )

        # Freeze the BERT model
        if freeze_bert:
            for param in self.bert.parameters():
                param.requires_grad = False

    def forward(self, input_ids, attention_mask):
                # Feed input to BERT
        outputs = self.bert(input_ids=input_ids,
                            attention_mask=attention_mask)

        # Extract the last hidden state of the token `[CLS]` for cla
ssification task
        last_hidden_state_cls = outputs[0][:, 0, :]

        # Feed input to classifier to compute logits
        logits = self.classifier(last_hidden_state_cls)

        return logits


def bert_predict(model, test_dataloader):
    # Put the model into the evaluation mode. The dropout layers are
 disabled during
    # the test time.
    model.eval()
    all_logits = []

    # For each batch in our test set...
    for batch in test_dataloader:
        # Load batch to GPU
        b_input_ids, b_attn_mask = tuple(t.to(device) for t in batch
)[:2]

        # Compute logits
        with torch.no_grad():
            logits = model(b_input_ids, b_attn_mask)
        all_logits.append(logits)

    # Concatenate logits from each batch
    all_logits = torch.cat(all_logits, dim=0)

    # Apply softmax to calculate probabilities
    probs = F.softmax(all_logits, dim=1).cpu().numpy()

    return probs
```

6.2 Generate Ratings

In this section we will convert the JSON response into data frame and give it to model getPredictions function, after getting sentiment score, we use the formula to generate system intelligent ratings.

```
#Function that creates dataframe of json file
def jsonToDataframe(df1):

  df2=[]

  #Simple dataframe which will be like above one
  for business in df1['data']:
    obj={'name':'','google_ratings':0,'reviews':[]}
    obj['name']=business['name']
    obj['google_ratings']=business['rating']
    for review in business['reviews']:
      obj['reviews'].append(review['text'])
      df2.append({'name': obj['name'],'google_ratings':obj['google_r
atings'],'reviews':obj['reviews']})

  dataframe = pd.DataFrame.from_dict(df2)
  #Empty dataframe and initialing its columns
  df=pd.DataFrame()
  df["name"]=""
  df["reviews"]=""
  df["google_rating"]=0
  df["sentiment_score"]=0
  df["system_rating"]=0
  #Get each object of json 1 by 1
  for index,val in dataframe.iterrows():
    #iterate over reviews
    for v in val["reviews"]:
      df=df.append({"name":val["name"],"reviews":v,"google_rating":v
al['google_ratings']},ignore_index=True)
  return df

def getPredictions(df):
  # Run `preprocessing_for_bert` on the prediction set
  pre_inputs, pre_masks = preprocessing_for_bert(df['reviews'])

  # Create the DataLoader for our prediction set
  pre_dataset = TensorDataset(pre_inputs, pre_masks)
  pre_sampler = SequentialSampler(pre_dataset)
  pre_dataloader = DataLoader(pre_dataset, sampler=pre_sampler, batc
h_size=10)

  #Compute predicted probabilities on the test set
  probs = bert_predict(model, pre_dataloader)

  # Get predictions from the probabilities
  threshold = 0.5

  return np.where(probs[:, 1] > threshold, 2, 0)
def getNeutral(df):
  for index,row in enumerate(df['reviews']):
    sentiment=use_TextBlob(row)
    if sentiment==1:
      df['sentiment_score'][index]=1
  return df
#this function takes dataframe with prediction column and return the
 system genrated score column data frame
#dataframe given by this function is converted into json and post to
 JS code .
def systemRating(df):
  final=pd.DataFrame()
  final["name"]=""
  final["google_rating"]=0
  final["system_rating"]=0
  for val in df["name"].unique():
    df2=df[df["name"]==val]
    count=sum=predictSum=0
    for index, row in df2.iterrows():
      count=count+1
      sum=sum+row["google_rating"]
      predictSum=predictSum+row["sentiment_score"]
    ratings=(predictSum/count)+sum/count
    final=final.append({"name":val,"google_rating":df2["google_ratin
g"].iloc[0],"system_rating":ratings},ignore_index=True)
  return final
def scale(ratings):

  scaler = MinMaxScaler(feature_range=(1, 5))
  orignal_system_rating=ratings+[0,7]
  system_ratings = np.array(orignal_system_rating).reshape(-1, 1)
  scaler = scaler.fit(np.array(system_ratings).reshape(-1, 1))
  scaled_ratings = scaler.transform(system_ratings)
  # system_ratings=[round(num, 1) for num in sclaed_ratings]

  # transferring back to list
  system_ratings = scaled_ratings.tolist()
  system_ratings=[round(num, 1) for num in list(chain.from_iterable(
system_ratings))]

  return system_ratings[:-2]
```

6.3 Flask's APIs

This section is used with front end to perform CRUD Operations. We have used MongoDB as database. Here are different APIs which are used for different purposes. All APIs are not mentioned, just important ones are there.

```
def getLocations(lat,lng,category,actualLat,actualLng):
  global response,response1
  api_key='y3s9ZkkmoGS-H-
lv4qMC8gZCWAPun7GEn8tiEzL8EiTR9_3EYYQuOWmzKTjbiQsDnfmvHOZL-MX1lj-
kkvnK0oqLWdIWKYXftJSyFC_bVDIpT3ZkxvzOoRmtsmluYXYx'
  headers = {'Authorization': 'Bearer {}'.format(api_key)}
  search_api_url = 'https://api.yelp.com/v3/businesses/search?catego
ries=%s&limit=7&latitude=%f&longitude=%f&radius=1000'%(category,lat,
lng)
```

```
  if (lng and lat):
    response = requests.get(search_api_url, headers=headers)
    businesses=response.json()['businesses']
    for business in businesses:
      search_api_url = 'https://api.yelp.com/v3/businesses/%s/review
s'%(business['id'])
      response1 = requests.get(search_api_url, headers=headers)
      business['reviews']=response1.json()['reviews'] or []
      business['type']=category
      business['lat']=actualLat
      business['lng']=actualLng
    places={'data':businesses}

  return places


@app.route('/api/predictions',methods=[ 'POST'])
def basic():

  try:
    client = MongoClient("Connection String")
    db=client.get_database("Cleverus")
    records=db.businesses
  except:
    return jsonify({"error": "Connection Error",}), 502

  p_lat,p_lng,n_lat,n_lng=add_blur(request.json['latituide'],request
.json['longitude'])
  data=records.find_one({"latitude":{"$gte":n_lat,"$lt":p_lat},"long
itude":{"$gte":n_lng,"$lt":p_lng}})

  if (data==None):
    response=getLocations(request.json['latituide'],request.json['lo
ngitude'], request.json['category'],request.json['latituide'],reques
t.json['longitude'])
    if not response['data']:
        return jsonify({"error": "No Data",}), 500
    df1=response
    df=jsonToDataframe(response)
    df['sentiment_score']=getPredictions(df)
    df=getNeutral(df)
    df=systemRating(df)
    df['system_rating']=scale(df['system_rating'].tolist())
    df1=addSystemRtaing(df1,df)
    return jsonify(df1)

  else:
    if (data[request.json['category']]!=[]):
     return jsonify({'data':data[request.json['category']]}),201
    else:
      response=getLocations(request.json['latituide'],request.json['
longitude'], request.json['category'],data['latitude'],data['longitu
de'])
      if not response['data']:
        return jsonify({"error": "No Data",}), 500
      df1=response
      df=jsonToDataframe(response)
      df['sentiment_score']=getPredictions(df)
      df=getNeutral(df)
      df=systemRating(df)
      df['system_rating']=scale(df['system_rating'].tolist())
      df1=addSystemRtaing(df1,df)
      return jsonify(df1)


@app.route('/api/store',methods=['POST'])
def dataStore():
  try:
    client = MongoClient("Connection String")
    db=client.get_database("Cleverus")
    records=db.businesses
  except:
    return jsonify({"error": "Connection Error",}), 502

  content=request.get_json(force=True)
  businessType=content["businessType"]
  content=content["data"]
  #print(businessType)
  if(content==None or businessType==None):
    return jsonify({"error": "Bad Request",}), 400

  lat=content["latitude"]
  lng=content["longitude"]
  p_lat,p_lng,n_lat,n_lng=add_blur(lat,lng)
  data=records.find_one({"latitude":{"$gte":n_lat,"$lt":p_lat},"long
itude":{"$gte":n_lng,"$lt":p_lng}})
  if(data==None):
    for index,val in enumerate(content[businessType]):
      content[businessType][index]['lat']=content["latitude"]
      content[businessType][index]['lng']=content["longitude"]
    records.insert_one(content)
    return json.dumps({'success':True,'lat':content["latitude"],'lng
':content["longitude"]}), 200, {'ContentType':'application/json'}
  else:
    for val in content[businessType]:
      val['lat']=data["latitude"]
      val['lng']=data["longitude"]
      data[businessType].append(val)
    records.update_one({'longitude':data["longitude"],'latitude':dat
a["latitude"]},{"$set": { f"{businessType}":data[businessType] } })
    return json.dumps({'success':True,'lat':data["latitude"],'lng':d
ata["longitude"]}), 200, {'ContentType':'application/json'}


@app.route('/api/fetch',methods=['GET'])
def datafetch():
  content=request.args
  lat=float(content['lat'])
  lng=float(content['lng'])
  business=content['business']
  id=content['id']
  if (lat==None or lng==None or business==None or id==None):
    return jsonify({"error": "Bad Request",}), 400
  try:
    client = MongoClient("Connection String")
    db=client.get_database("Cleverus")
    records=db.businesses
  except:
    return jsonify({"error": "Connection Error",}), 502
  finally:
    getData=records.find_one({"longitude":lng,"latitude":lat})
    if(getData==None):
      return jsonify({"error": "No Content",}), 204
    business_Data=getData[business]
    if(business_Data==None):
      return jsonify({"error": "No Content",}), 204
    for val in business_Data:
      if(val["id"]==id):
        return jsonify(val),200
    return jsonify({"error": "No Place Found",}), 204

@app.route('/api/similarBuisness',methods=['GET'])
def getRandomfromBusiness():
  content=request.args
  lat=float(content['lat'])
  lng=float(content['lng'])
  business=content['business']
  if (lat==None or lng==None or business==None):
    return jsonify({"error": "Bad Request",}), 400
  try:
    client = MongoClient("Connection String")
    db=client.get_database("Cleverus")
    records=db.businesses
  except:
    return jsonify({"error": "Connection Error",}), 502
  finally:
    getData=records.find_one({"longitude":lng,"latitude":lat})
    if(getData==None):
      return jsonify({"error": "No Content",}), 204
    business_Data=getData[business]
    if(business_Data==None):
      return jsonify({"error": "No Content",}), 204
    else:
      business_Data.sort(key=lambda x: x['review_count'], reverse=Tr
ue)
      if(len(business_Data)>=3):
        return jsonify(business_Data[1:4]),200
      else:
        return jsonify(business_Data),200


@app.route('/api/deleteUser',methods=['DELETE'])
def deleteUser():
  content=request.args
  id=content['userID']
  if (id==None):
    return jsonify({"error": "Bad Request",}), 400
  try:
    client = MongoClient("Connection String")
    db=client.get_database("Cleverus")
    records=db.users
  except:
    return jsonify({"error": "Connection Error",}), 502
  finally:
    getData=records.delete_one({"userID":id})
    return 'Deleted Successfully'


@app.route('/api/addReview',methods=['POST'])
```

```
def addReview():
  lat=float(request.json['latitude'])
  lng=float(request.json['longitude'])
  category=request.json['category']
  id=request.json['id']
  userID=request.json['userID']
  review=request.json['review']

  if (lat==None or lng==None or category==None or id==None or review
==None or userID==None):
    return jsonify({"error": "Bad Request",}), 400
  try:
    client = MongoClient("Connection String")
    db=client.get_database("Cleverus")
    records=db.businesses
  except:
    return jsonify({"error": "Connection Error",}), 502
  finally:
    getData=records.find_one({"longitude":lng,"latitude":lat})
    if(getData==None):
      return jsonify({"error": "No Content",}), 204
    business_Data=getData[category]
    if(business_Data==None):
      return jsonify({"error": "No Content for this Category",}), 20
4
    else:
      for i,val in enumerate(business_Data):
        if (val['id']==id):
          reviews=val['reviews']
          reviews.append(review)
          business_Data[i]['reviews']= reviews
          db.businesses.update_one({'longitude':lng,'latitude':lat},
{"$set": { f"{category}":business_Data } })
          user=db.users.find_one({"userID":userID})
          if (user==None):
            user={"userID":userID, "favPlaces":[],"noOfReviews":1}
            db.users.insert_one(user)
          else:
            db.users.update_one({'userID':userID},{"$inc":{'noOfRevi
ews':1}})
          return 'Done',200
    return jsonify({"error": "No Place Found",}), 204


@app.route('/api/addFavPlace',methods=['POST'])
```

```
def addFavPlace():
  content=request.get_json(force=True)
  userID=content["userID"]
  favPlace=content["favPlaces"]
  if(userID==None or favPlace==None):
    return jsonify({"error": "No Content",}), 204
  try:
    client = MongoClient("Connection String")
    db=client.get_database("Cleverus")
    records=db.users
  except:
    return jsonify({"error": "Connection Error",}), 502

  user=records.find_one({"userID":userID})
  if(user==None):
    user={
        "userID":userID,
        "favPlaces":favPlace,
        "noOfReviews":0
    }
    records.insert_one(user)
    return jsonify({}), 200
  else:
    for val in favPlace:
      if [fav for fav in user["favPlaces"] if fav['id'] == val['id']
]:
        return jsonify({}), 201
    user["favPlaces"].append(val)
    records.update_one({"userID":userID},{"$set":{"favPlaces":user["
favPlaces"]}})
    return jsonify({}), 200

@app.route('/api/favPlaceExist',methods=['POST'])
def checkFavPlace():
  content=request.get_json(force=True)
  userID=content["userID"]
  favPlace=content["favPlaces"]
  if(userID==None or favPlace==None):
    return jsonify({"error": "No Content",}), 204
  try:
    client = MongoClient("Connection String")
    db=client.get_database("Cleverus")
    records=db.users
  except:
    return jsonify({"error": "Connection Error",}), 502
  user=records.find_one({"userID":userID})
  if(user==None):
    return jsonify({}), 200
  else:
    for val in favPlace:
      if [fav for fav in user["favPlaces"] if fav['id'] == val['id']
]:
        return jsonify({}), 201
    return jsonify({}), 200


@app.route('/api/getFavPlace',methods=['GET'])
def getFavPlace():
  content=request.args
  userID=content['userID']
  if(userID==None):
    return jsonify({"error": "No Content",}), 204
  try:
    client = MongoClient("Connection String ")
    db=client.get_database("Cleverus")
    records=db.users
  except:
    return jsonify({"error": "Connection Error",}), 502

  user=records.find_one({"userID":userID})
  if(user==None):
    return jsonify([]),200
  else:
    return jsonify(user["favPlaces"]),200

updated_ratings_df=[];

@app.route('/api/updateRating',methods=['POST'])
def makeRatingDynamic():
  global updated_ratings_df;
  content=request.get_json(force=True)
  lat=float(content['lat'])
  lng=float(content['lng'])
  business=content['business']
  id=content['id']
  if (lat==None or lng==None or business==None or id==None):
    return jsonify({"error": "Bad Request",}), 400
  try:
    client = MongoClient("Connection String")
    db=client.get_database("Cleverus")
    records=db.businesses
  except:
    return jsonify({"error": "Connection Error",}), 502
  finally:
    getData=records.find_one({"longitude":lng,"latitude":lat})
    if(getData==None):
      return jsonify({"error": "No Content",}), 204
    business_Data=getData[business]
    if(business_Data==None):
      return jsonify({"error": "No Content",}), 204
    for i,val in enumerate(business_Data):
      if(val["id"]==id):
        df1={'data': [val]}
        df=jsonToDataframe(df1)
        df['sentiment_score']=getPredictions(df)
        updated_ratings_df=df
        df=getNeutral(df)
        df=systemRating(df)
        df['system_rating']=scale(df['system_rating'].tolist())
        df1=addSystemRtaing(df1,df)
        business_Data[i]['system_rating']=df1['data'][0]['system_rat
ing']
        db.businesses.update_one({'longitude':lng,'latitude':lat},{"
$set": { f"{business}":business_Data } })
        return jsonify(business_Data[i]),200
```

Chapter 7

Conclusion

7 Conclusion

7.1 Problems faced and lessons learned

7.1.1 Dataset Availability:

Initial problem with starting this project was to find out some reliable, fair quantity and good quality dataset with various end to end business embedded in between. Struggle was big as there is no such dataset available on the internet and for the data crawling part, major websites were not willing to allow.

7.1.2 Google Colab Limitations:

Provided RAM and storage by the Colab servers are not enough to train NLP problems with sample of the dataset as huge. Even loading data from JSON file to convert to comma separated format was hard and needs to be divided into batches. Only way to get around was to use partitioning of the data and perform the task into rounds.

7.1.3 Sentiment Analysis State of the Art Models:

Dataset we had was from Most Famous European Review Community YELP. Dataset was un-annotated which was notifying about another major problem round the corner. Now the race for reliable state of the art sentiment analysis models started which didn't end that easily. At least 5-7 models were tried on small sampled dataset including Text-Blob, Flair to predict the partiality of reviews. At the end BERT does solve this problem but after training two complete datasets of Twitter and IBDB.

7.1.4 Team-Work and Task Management

As the time started to pass and project deadline started to plunder, we realize how much of a task our group was assigned and the roadmap we three had to follow to reach some milestone. To follow that roadmap, we had to ensure task management was in order and whole team was putting their full strength in

7.1.5 Finding Alternatives

It's must that you will face problems and cons with every step you take towards complex projects. Similar was the case here too as we needed several tweaking's and alternatives at every inch but that’s how you supposed to become a better debugger and problem solver. Every struggle we had has provided us some knowledge and that how this field has supposed to be, so no worries

7.2 Project summary

We have built a generic and dynamic model which can be able to suggest anything, while the suggestions will be depending on the feedback of the previous users.

The proposed solution is somehow similar to yelp; in which we will generate system ratings of the place on the basis of reviews given by users with the combination of users' ratings. Multiple things would be considered when it comes to the process of generating ratings like polarity of reviews, a balance between ratings and reviews. On other hand to make a dynamic system that will not depend on a 1-time rating, rather change over a period of time will depend on the new reviews. To train the model we have used the "IMDB" dataset [12]. Currently we are working on 4 modules i.e. hospitals, restaurants, hotels, and barbers. One of the plus points of the proposed solution is that we will be able to use the same model for multiple modules, but the constraint will remain, related to the availability of data for that module.

7.3 Future work

Some of the future goals are mentioned below which will help to make recommendation applications more user-friendly, robust and flexible while keeping the quality maintained.

- Integrate more modules over the time according to their need.
- To perform better, Introduce Reinforcement Learning.
- We will create the module using which one will make online bookings; this will save the time of the user as well as will help the service providers to interact with our app in a better way.
- Adding the appointment section with the domain experts like doctors, property dealers etc.
- Make blog within application where ranked pros publish their experience and reviews to aware the people about the latest insights

Chapter 8

References

References

[1] B. Rocca, “Introduction to recommender systems,” *Towards Data Science*, 3 June 2019.

[2] “UNESCO,” [Online]. Available: http://uis.unesco.org/en/country/pk?theme=education-and-literacy. [Accessed 5 June 2021].

[3] Dontgiveupworld, “Inspiring Story Of Muniba Mazari,” *Medium.com,* 4 January 2018.

[4] F. Isinkaye, “Recommendation systems: Principles, methods and evaluation,” *Egyptian Informatics Journal,* vol. 16, no. 3, pp. 261-273, November 2015.

[5] S. M. Al-Ghuribi, “Multi-Criteria Review-Based Recommender System – The State of the Ar,” *IEEE,* November 2019.

[6] S. Aciar, “Recommender System Based on Consumer Product Reviews,” December 2006.

[7] “Yelp,” [Online]. Available: https://www.yelp.com/. [Accessed 5 June 2021].

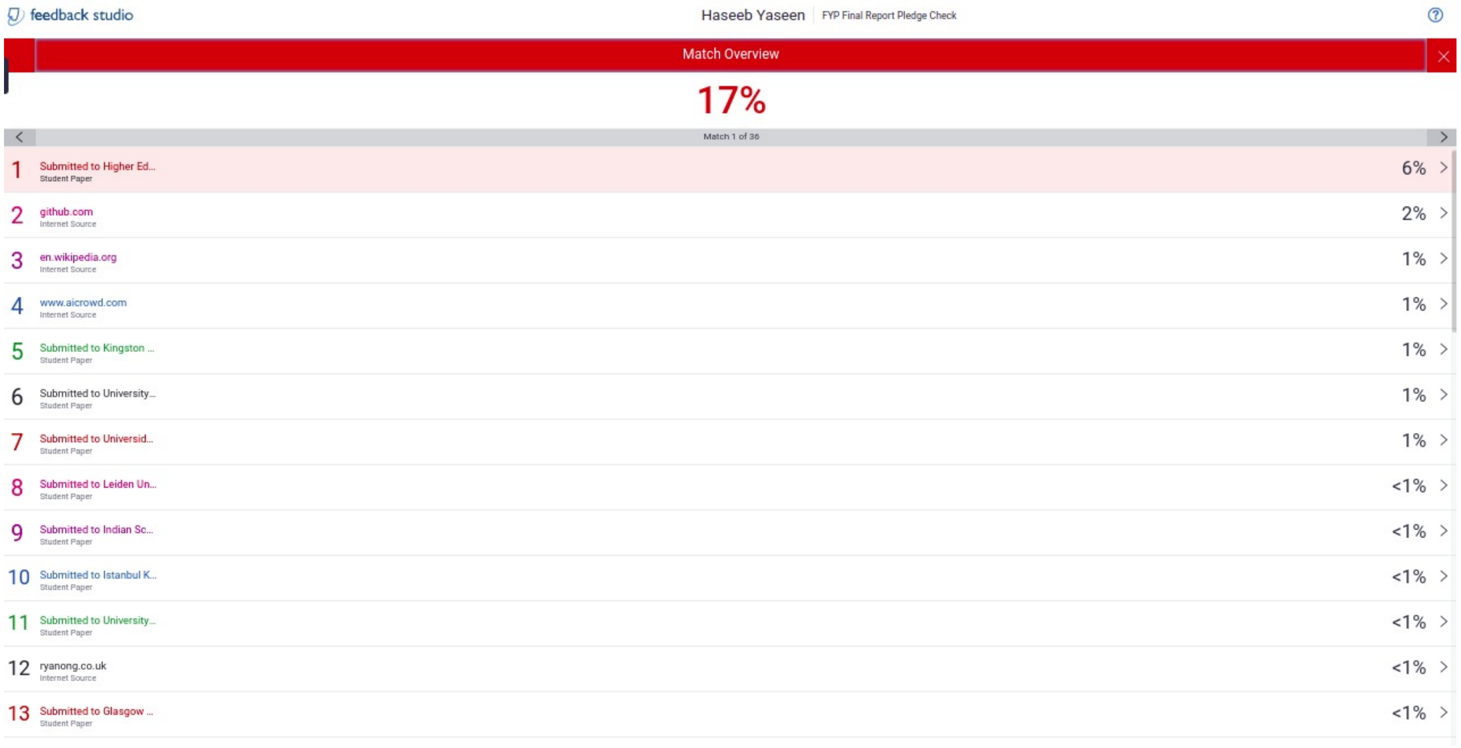
[8] “Trip Advisor,” [Online]. Available: https://www.tripadvisor.com/. [Accessed 5 June 2021].

[9] “Urban Spoon,” [Online]. Available: https://play.google.com/store/apps/details?id=com.urbanspoon&hl=en&gl=US. [Accessed 5 June 2021].

[10] P. Eksombatchai, “An update on Pixie, Pinterest’s recommendation system,” *Medium.com,* 30 November 2018.

[11] D. Chong, “Deep Dive into Netflix’s Recommender System,” *Towards Data Science,* 30 April 2020.

[12] “Dataset, IMDB,” 16 February 2021. [Online]. Available: https://www.imdb.com/interfaces/. [Accessed 5 June 2021].

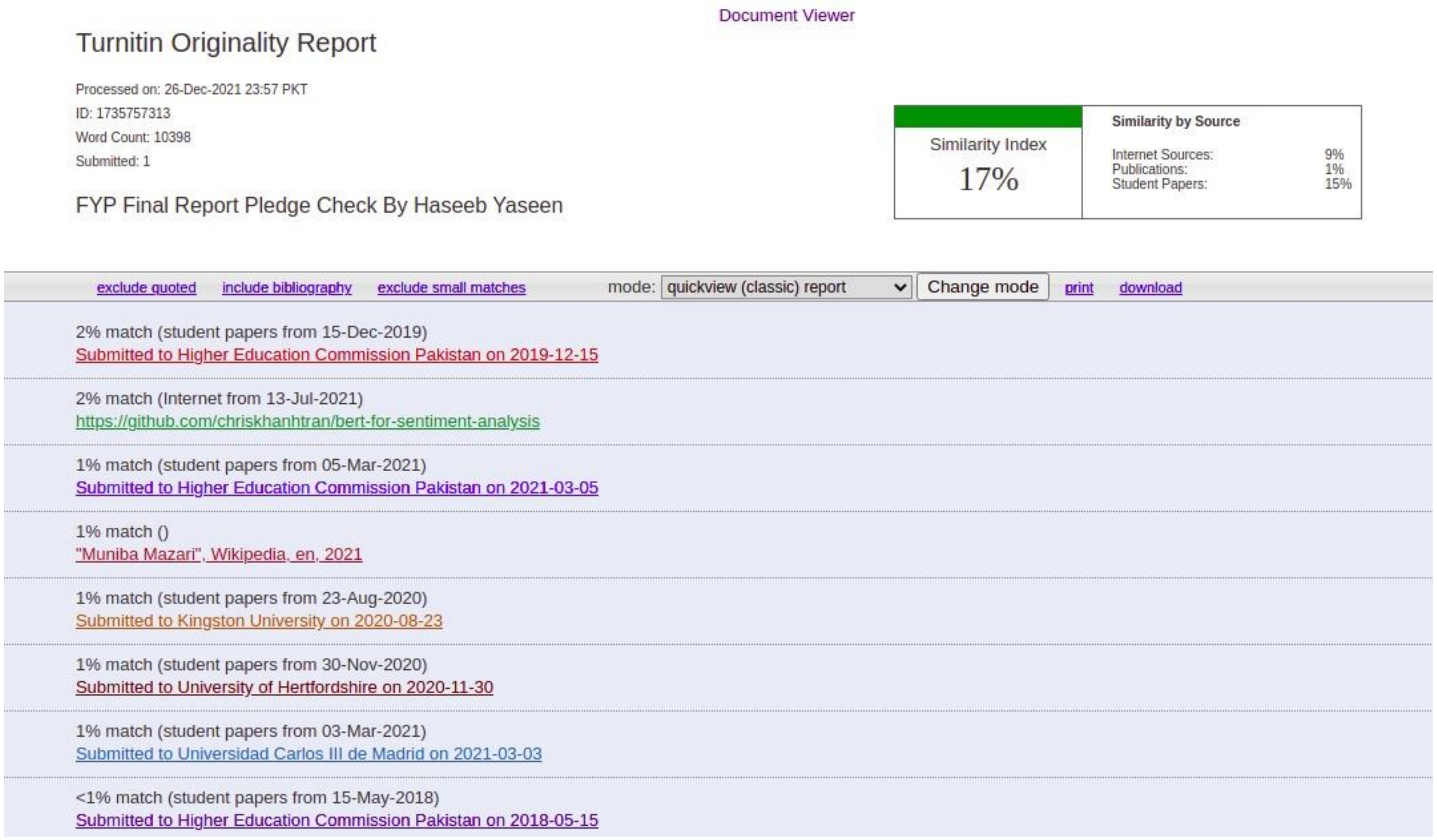

feedback studio | Haseeb Yaseen | FYP Final Report Pledge Check

Match Overview

17%

Match 1 of 36

1	Submitted to Higher Ed... Student Paper	6%
2	github.com Internet Source	2%
3	en.wikipedia.org Internet Source	1%
4	www.aicrowd.com Internet Source	1%
5	Submitted to Kingston ... Student Paper	1%
6	Submitted to University... Student Paper	1%
7	Submitted to Universid... Student Paper	1%
8	Submitted to Leiden Un... Student Paper	<1%
9	Submitted to Indian Sc... Student Paper	<1%
10	Submitted to Istanbul K... Student Paper	<1%
11	Submitted to University... Student Paper	<1%
12	ryanong.co.uk Internet Source	<1%
13	Submitted to Glasgow ... Student Paper	<1%


Document Viewer

Turnitin Originality Report

Processed on: 26-Dec-2021 23:57 PKT
ID: 1735757313
Word Count: 10398
Submitted: 1

FYP Final Report Pledge Check By Haseeb Yaseen

Similarity Index
17%

Similarity by Source

Internet Sources:	9%
Publications:	1%
Student Papers:	15%

exclude quoted | include bibliography | exclude small matches | mode: quickview (classic) report | Change mode | print | download

2% match (student papers from 15-Dec-2019)
Submitted to Higher Education Commission Pakistan on 2019-12-15

2% match (Internet from 13-Jul-2021)
https://github.com/chriskhanhtran/bert-for-sentiment-analysis

1% match (student papers from 05-Mar-2021)
Submitted to Higher Education Commission Pakistan on 2021-03-05

1% match ()
"Muniba Mazari", Wikipedia, en, 2021

1% match (student papers from 23-Aug-2020)
Submitted to Kingston University on 2020-08-23

1% match (student papers from 30-Nov-2020)
Submitted to University of Hertfordshire on 2020-11-30

1% match (student papers from 03-Mar-2021)
Submitted to Universidad Carlos III de Madrid on 2021-03-03

<1% match (student papers from 15-May-2018)
Submitted to Higher Education Commission Pakistan on 2018-05-15