

# **Counsel – A MERN Stack Recommending System for Educational Institutes**

Project Team

Muhammad Abdul Nafay	19P-0117
Sheheryar Ali	19P-0120
Saad Javed	19P-0111

Session 2014-2018

Supervised by

Shoaib Muhammad Khan



**Department of Computer Science**

**National University of Computer and Emerging Sciences  
Peshawar, Pakistan**

**May-June, 2023**

---

## Student's Declaration

We declare that this project titled “*Counsel – A MERN Stack Recommending System for Educational Institutess*”, submitted as requirement for the award of degree of Bachelors in Computer Science, does not contain any material previously submitted for a degree in any university; and that to the best of our knowledge, it does not contain any materials previously published or written by another person except where due reference is made in the text.

We understand that the management of Department of Computer Science, National University of Computer and Emerging Sciences, has a zero tolerance policy towards plagiarism. Therefore, We, as authors of the above-mentioned thesis, solemnly declare that no portion of our thesis has been plagiarized and any material used in the thesis from other sources is properly referenced.

We further understand that if we are found guilty of any form of plagiarism in the thesis work even after graduation, the University reserves the right to revoke our BS degree.

Muhammad Abdul Nafay

Signature: \_\_\_\_\_

Sheheryar Ali

Signature: \_\_\_\_\_

Saad Javed

Signature: \_\_\_\_\_

---

Verified by Plagiarism Cell Officer

Dated:

# Certificate of Approval



The Department of Computer Science, National University of Computer and Emerging Sciences, accepts this thesis titled *Counsel – A MERN Stack Recommending System for Educational Institutess*, submitted by Muhammad Abdul Nafay (19P-0117), Sheheryar Ali (19P-0120), and Saad Javed (19P-0111), in its current form, and it is satisfying the dissertation requirements for the award of Bachelors Degree in Computer Science.

## Supervisor

Shoaib Muhammad Khan

Signature: \_\_\_\_\_

---

Zeshan Khan Alvi

FYP Coordinator

National University of Computer and Emerging Sciences, Peshawar

---

Dr. Hafeez Ur Rehman

HoD of Department of Computer Science

National University of Computer and Emerging Sciences

## **Acknowledgements**

One of the biggest challenges for every young adult is to find the right educational institutes according to their preference and interest. Cases of students dropping out at the beginning of the semester are also quite significant. The main cause of failure is the selection of the right institutes that fits the user's preferences. The traditional method of getting recommendation is to either search global universities ranking on the internet or asking other peers for their suggestions. This way is inefficient, unreliable and it also doesn't keep the user's interests in mind. If an overseas student is thinking over further education in Pakistan, then he/she will have a hard time of which university to choose on. This project aims to solve all of these problems by having a platform which requires all the relevant input from the user (such as a user's transcript, interests, budget, location etc.), and it applies different machine learning algorithms to give an output in the form of a list of different institutes that are ranked according to the user's input along with the features of each institute which tells that why it is being recommended.

Muhammad Abdul Nafay

Sheheryar Ali

Saad Javed

## **Abstract**

One of the biggest challenges for every young adult is to find the right educational institutes according to their preference and interest. Cases of students dropping out at the beginning of the semester are also quite significant. The main cause of failure is the selection of the right institutes that fits the user's preferences. The traditional method of getting recommendation is to either search global universities ranking on the internet or asking other peers for their suggestions. This way is inefficient, unreliable and it also doesn't keep the user's interests in mind. If an overseas student is thinking over further education in Pakistan, then he/she will have a hard time of which university to choose on. This project aims to solve all of these problems by having a platform which requires all the relevant input from the user (such as a user's transcript, interests, budget, location etc.), and it applies different machine learning algorithms to give an output in the form of a list of different institutes that are ranked according to the user's input along with the features of each institute which tells that why it is being recommended.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	History of Recommendation System . . . . .	1
1.2	Types of Recommendation System . . . . .	3
1.2.1	Content-based recommendation techniques . . . . .	3
1.2.2	Collaborative Based Filtering . . . . .	3
1.2.3	Hybrid Recommendation . . . . .	4
<b>2</b>	<b>Literature Review</b>	<b>5</b>
2.1	University Recommending System for Graduate Studies in USA [11] . . .	5
2.1.1	Basic Idea . . . . .	5
2.1.2	Methodologies . . . . .	5
2.1.2.1	Data Scrapping . . . . .	5
2.1.2.2	Data Cleansing and Transformation . . . . .	6
2.1.3	Machine Learning Models Used . . . . .	6
2.1.4	Result . . . . .	7
2.1.5	Limitation . . . . .	7
2.2	Fuzzy-Based Recommendation System for University Major Selection [12]	7
2.2.1	Basic Idea . . . . .	7
2.2.2	Methodologies . . . . .	8
2.2.2.1	Data Collection . . . . .	8
2.2.2.2	Overall Process . . . . .	8
2.2.3	Results . . . . .	8
2.2.4	Limitations . . . . .	8
2.3	Analysis and Design of Personalized Recommendation System for Uni- versity Physical Education . . . . .	9

2.3.1	Basic Idea . . . . .	9
2.3.2	Methodologies . . . . .	9
2.3.2.1	Input Module . . . . .	9
2.3.2.2	Personalized processing module . . . . .	9
2.3.2.3	Output module . . . . .	10
2.3.2.4	User management module . . . . .	10
2.3.3	Results . . . . .	11
2.3.4	Limitations . . . . .	11
2.4	An Approach to a University Recommendation by Multi-criteria Collaborative Filtering and Dimensionality Reduction Techniques . . . . .	11
2.4.1	Basic Idea . . . . .	11
2.4.2	Methodologies . . . . .	12
2.4.3	Results . . . . .	12
2.4.4	Limitations . . . . .	14
<b>3</b>	<b>Project Vision</b>	<b>15</b>
3.1	Problem Statement . . . . .	15
3.2	Proposed Statement . . . . .	15
3.3	Motivation . . . . .	16
3.4	Objective . . . . .	16
3.5	Project Scope . . . . .	16
3.6	Constraints . . . . .	17
<b>4</b>	<b>Software Requirement Specifications</b>	<b>19</b>
4.1	Implementation . . . . .	19
4.2	Software Architecture Diagram . . . . .	19
4.3	Use Case Diagram . . . . .	19
4.4	Swimlane Diagram . . . . .	20
4.5	Web Application . . . . .	20
4.5.1	Backend . . . . .	20
4.5.1.1	MVC Framework . . . . .	20
4.5.1.2	Controller . . . . .	20



4.5.1.3	Model . . . . .	21
4.5.1.4	View . . . . .	21
4.5.2	Frontend . . . . .	22
4.5.2.1	UI Screens . . . . .	22
4.6	Functional Requirement . . . . .	22
4.7	Non-Functional Requirement . . . . .	23
<b>5</b>	<b>Iterations 1</b>	<b>31</b>
5.1	Methodologies . . . . .	31
5.1.1	Council Bot . . . . .	31
5.1.1.1	Tokenization . . . . .	31
5.1.1.2	Lemmatization . . . . .	32
5.1.2	Plant Bot . . . . .	32
5.1.3	Machine Learning to generate recommendations . . . . .	33
5.1.3.1	Decision Tree Classification . . . . .	33
5.1.3.2	RandomForest Classifier (RFC) . . . . .	33
<b>6</b>	<b>Iterations 2</b>	<b>35</b>
6.1	Use Case Diagram . . . . .	35
6.2	Activity Diagram . . . . .	35
6.3	Component Diagram . . . . .	35
6.4	Updated UI Screens . . . . .	35
6.5	Data Collection . . . . .	36
6.6	Machine Learning Models . . . . .	41
6.6.1	Decision Trees Classifier (DTC) . . . . .	41
6.6.2	Introduction to Ensemble Models . . . . .	41
6.6.2.1	Bagging Classifier . . . . .	42
6.6.2.2	Random Forest Classification (RFC) . . . . .	43
6.6.2.3	Adaptive Boosting (ADA Boost) . . . . .	44
6.6.2.4	XGBoosting . . . . .	45
6.6.3	Stacking Ensemble Model . . . . .	45
6.7	Algorithm Design . . . . .	46

6.8	Evaluation and Testing . . . . .	46
<b>7</b>	<b>Iterations 3</b>	<b>49</b>
7.1	Use Case Diagram . . . . .	49
7.2	Activity Diagram . . . . .	49
7.3	Component Diagram . . . . .	49
7.4	Updated UI Screens . . . . .	49
7.5	Data Collection . . . . .	50
7.5.1	Google Forms . . . . .	50
7.5.2	Dummy Data Set . . . . .	55
7.6	Chatbot . . . . .	55
7.6.1	Training data . . . . .	55
7.6.2	Tokenization . . . . .	55
7.6.3	Stemming . . . . .	56
7.6.4	Feedforward Network . . . . .	57
7.6.5	Workflow . . . . .	58
7.7	Machine Learning Integration with Web Application . . . . .	58
7.7.1	Server Side Scripts . . . . .	58
7.8	Initial Recommendation . . . . .	59
7.9	N-List Recommendation . . . . .	61
	<b>References</b>	<b>64</b>

# List of Figures

2.1	System Application Architecture Diagram . . . . .	10
2.2	Precision . . . . .	13
2.3	F1 Metric . . . . .	13
4.1	Abstract Implementation Diagram . . . . .	24
4.2	Software Architecture Diagram . . . . .	24
4.3	Use Case Diagram . . . . .	25
4.4	Swimlane Diagram . . . . .	26
4.5	MVC Framework . . . . .	27
4.6	Logging In . . . . .	28
4.7	Signing In/Registration Page . . . . .	28
4.8	Home Tab/Dashboard . . . . .	29
4.9	Search/Explore Tab . . . . .	29
4.10	Counsel Tab . . . . .	30
6.1	Use Case Diagram . . . . .	36
6.2	Activity Diagram . . . . .	37
6.3	Component-Diagram . . . . .	38
6.4	Logging In . . . . .	39
6.5	Signing In/Registration Page . . . . .	39
6.6	Home Tab/Dashboard . . . . .	40
6.7	Search/Explore Tab . . . . .	40
6.8	Council Tab . . . . .	41
6.9	Decision Tree Example . . . . .	42

6.10	Given a Dataset, bootstrapped subsamples are pulled. A Decision Tree is formed on each bootstrapped sample. The results of each tree are aggregated to yield the strongest, most accurate predictor. . . . .	43
6.11	A random forest takes a random subset of features from the data, and creates n random trees from each subset. Trees are aggregated at the end. .	44
6.12	Stacking-Ensemble-Model . . . . .	46
6.13	Stacking Ensemble model Cross Validation Accuracy . . . . .	47
7.1	Use Case Diagram . . . . .	50
7.2	Activity Diagram . . . . .	51
7.3	Component-Diagram . . . . .	52
7.4	Home Page . . . . .	53
7.5	About Page . . . . .	53
7.6	Council Tab . . . . .	54
7.7	Recommendation Tab . . . . .	54
7.8	Spawning python script from nodejs file . . . . .	60

# List of Tables

2.1	Statistics of the Features . . . . .	6
2.2	Results . . . . .	7
4.1	UsersModel . . . . .	21
4.2	UniversityModel . . . . .	21
4.3	UserDataModel . . . . .	22
6.1	CleanData.csv . . . . .	38

# Chapter 1

## Introduction

Recommender systems can be defined as programs which attempt to recommend the most suitable items (products or services) to particular users (individuals or businesses) by predicting a user's interest in an item based on related information about the items, the users and the interactions between items and users [4]. The aim of developing recommender systems is to reduce information overload by retrieving the most relevant information and services from a huge amount of data, thereby providing personalized services. The most important feature of a recommender system is its ability to “guess” a user's preferences and interests by analyzing the behavior of this user and/or the behavior of other users to generate personalized recommendations [8].

### 1.1 History of Recommendation System

Recommender Systems (RSs) is an emerging research field that has grown fast and become popular. The increase of interest in this research topic has also been driven by great improvements in internet technology and e-commerce. E-commerce has been one of the major reasons why this became so popular. It helped buyers with no experience in online shopping, by cross-selling the products and by improving customer loyalty. The peak explosion of research in RSs occurred when Amazon launched their Collaborative Filtering (CF) method at the end of the 1990s, successfully increasing their sales. The

successful Amazon became popular and other online businesses started to implement RSs on their website. Amazon has patented its CF method as a United States Patent. Due to the fact that the main goal of an RS is to find the preferred information and eliminate information which is not liked by a user, the RS field can be considered as a subset of information filtering. The process of exploring a user's preferences from their historical data is followed by processing it using machine learning algorithms to build a ranked list of recommended items, as preferred by the user [14]. The idea of exploiting computers to recommend the best item for the user has been around since the beginning of computing. The first implementation of the RS concept appeared in 1979, in a system called Grundy, a computer-based librarian that provided suggestions to the user on what books to read. This followed in the early 1990s with the launch of Tapestry, the first commercial RS. Another RS implementation for helping people find their preferred articles was launched in the early 1990s by GroupLens, a research lab at the University of Minnesota, USA. They named the system after the group, GroupLens Recommender System. This system claims to have a similar spirit to that of Tapestry, Ringo, BellCore and Jester. Further development of RSs in the late 1990s was the implementation of Amazon Collaborative Filtering, one of the most widely known RS technologies. Since this era, RSs based on Collaborative Filtering has become very popular and has been implemented by many e-commerce and online systems. Many toolboxes for RSs have also been developed. The success story of Amazon also gave rise to the development of many RS algorithms known as hybrid approaches, which combines collaborative filtering with content-based filtering [14]. Following the successful era at the end of the 1990s, the industry offered generous funding to implement RSs research. The most popular competition in RSs was held by Netflix. They launched the Netflix Prize in 2006 and give a million US Dollars to the winner of the competition who provided the best RS movie recommendation. In 2010, YouTube also implemented an RS on its website and it tries to predict what a user would like to see next based on what they usually like to watch, based on their own preferences and interests [1].

## 1.2 Types of Recommendation System

### 1.2.1 Content-based recommendation techniques

Content-based (CB) recommendation techniques recommend articles or commodities that are similar to items previously preferred by a specific user [10]. The basic principles of CB recommender systems are: 1) To analyze the description of the items preferred by a particular user to determine the principal common attributes (preferences) that can be used to distinguish these items. These preferences are stored in a user profile. 2) To compare each item's attributes with the user profile so that only items that have a high degree of similarity with the user profile will be recommended [10]. In CB recommender systems, two techniques have been used to generate recommendations. One technique generates recommendations heuristically using traditional information retrieval methods, such as cosine similarity measure. The other technique generates recommendations using statistical learning and machine learning methods, largely building models that are capable of learning users' interests from the historical data (training data) of users.

### 1.2.2 Collaborative Based Filtering

Collaborative filtering (CF)-based recommendation techniques help people to make choices based on the opinions of other people who share similar interests [7]. The CF technique can be divided into user-based and item-based CF approaches [2]. In the user-based CF approach, a user will receive recommendations of items liked by similar users. In the item-based CF approach, a user will receive recommendations of items that are similar to those they have loved in the past. The similarity between users or items can be calculated by Pearson correlation-based similarity [9], constrained Pearson correlation (CPC)-based similarity, cosine-based similarity, or adjusted cosine-based measures. When calculating the similarity between items using the above measures, only users who have rated both items are considered. This can influence the similarity accuracy when items which have received a very small number of ratings express a high level of similarity with other items. To improve similarity accuracy, an enhanced item-based CF approach was presented by



combining the adjusted cosine approach with Jaccard metric as a weighting scheme. To compute the similarity between users, the Jaccard metric was used as a weighting scheme with the CPC to obtain a weighted CPC measure [13]. To deal with the disadvantage of the single-rating based approach, multi-criteria collaborative filtering was developed [6].

### 1.2.3 Hybrid Recommendation

To achieve higher performance and overcome the drawbacks of traditional recommendation techniques, a hybrid recommendation technique that combines the best features of two or more recommendation techniques into one hybrid technique has been proposed. According to Robin Burke [3], there are seven basic hybridization mechanisms of combinations used in recommender systems to build hybrids: weighted, mixed, switching, feature combination, feature augmentation, cascade and meta-level. The most common practice in the existing hybrid recommendation techniques is to combine the CF recommendation techniques with the other recommendation techniques in an attempt to avoid cold-start, sparseness and/or scalability problems.

# Chapter 2

## Literature Review

### 2.1 University Recommending System for Graduate Studies in USA [11]

#### 2.1.1 Basic Idea

Often, the students wonder whether their profile is good enough for a certain university. This paper solves the problem by taking scrapping the required data from [www.edulix.com](http://www.edulix.com), and a data-set containing profiles of students with admits/rejects to 45 different universities in USA was built. Based on this data set, various models were trained and a list of 10 best universities are suggested such that it maximizes the chances of a student getting an admit from that university list.

#### 2.1.2 Methodologies

##### 2.1.2.1 Data Scrapping

They Initially narrowed the list for 45 different universities. Universities with skewed data were dropped down. Then a crawler was built to get the list of students and the links to their profiles on Edulix. Once the unique set of students was identified, the data was

Table 2.1: Statistics of the Features

	Research Exp.	Industry Exp.	Intern Exp.	GRE Verbal	GRE AWE	Journal Publications	Conference Publications	CGPA
Mean	0.29	3.46	0.39	148.31	5.29	0.03	0.04	0.75
Std. Deviation	2.42	11.11	2.26	15.39	1.48	0.25	0.32	0.36
Min	0.00	0.00	0.00	0.00	1.50	0.00	0.00	0.00
Max	53.00	132.00	96.00	170.00	6.00	12.00	8.00	0.98
25%	0.00	0.00	0.00	145.00	3.00	0.00	0.00	0.70
50%	0.00	0.00	0.00	150.00	3.50	0.00	0.00	0.77
75%	0.00	0.00	0.00	154.00	4.00	0.00	0.00	0.84

scraped from each profile using ‘Beautiful Soup’.

### 2.1.2.2 Data Cleansing and Transformation

About 45000 samples of raw data was obtained by as a result of scraping. Cleansing the data of had to be done, since this field was just a text box and not a selected field. A total of 1435 distinct undergraduate universities and 53 distinct majors were obtained after filtering and each of these were used as binary features.

All the statistical values of each features is given in Table [2.1](#)

### 2.1.3 Machine Learning Models Used

Three different models, Support Vector Machine, K-Nearest Neighbors and Random Forest, were built using a combination of all the features mentioned above, to classify a student profile to the best university that they must apply to, among the available 45 universities. Once the best university was found for the student, the 9 most similar universities in terms of the selected features was found by computing Euclidean distances to give a total of 10 universities, that the student must apply to. They split the dataset into 80:20 ratio. They used cross validation technique to avoid data overfitting as it reduces the variance by averaging over k different partitions, so the performance estimate is less sensitive to the partitioning of the data.

Table 2.2: Results

Baseline	K Nearest Neighbour	Random Forest	SVM
22.2%	50.6%	50.5%	53.4%

### 2.1.4 Result

In this work, K-Nearest Neighbor, Random Forest and Support Vector Machine were considered for recommending the 10 best universities for aspiring graduate students and their performances are summarized in Table 2.2

### 2.1.5 Limitation

- Accuracy of the models predicting the data is not that great.
- This project recommends universities by taking the data of Admit/Rejects of different students from multiple universities into account only.
- Takes only 45 universities from 1047 total universities in USA.

## 2.2 Fuzzy-Based Recommendation System for University Major Selection [12]

### 2.2.1 Basic Idea

This paper focuses on the decision of choosing a university major. To make a suitable decision, a student needs an expert opinion, time, and effort. Therefore, a decision-making system should be developed in order to help prospective students to increase their educational outcome and productivity.

## **2.2.2 Methodologies**

### **2.2.2.1 Data Collection**

The first survey targeted high school students (specifically 239 prospective participants). The second survey targeted university students (specifically 392 university participants) to give their insights after spending a year in a specific major.

### **2.2.2.2 Overall Process**

- Dataset features passed into Fuzzy Expert System (FES)
- Inference Mechanism is used
- Defuzzification is used
- Cluster Based preference is applied which puts all majors in groups, where majors in the same group are more similar than the majors in different groups

## **2.2.3 Results**

Results showed that 66 Percent were strongly pleased with the system and 54 Percent were pleased with suggestions provided by the system.

## **2.2.4 Limitations**

- Interest or preference of students is not taken into account, Dataset generated by taking the eligibility criteria for each of the major groups
- This is the first software development phase (pre-alpha version). Actual application not been implemented yet
- Focuses on university in Taif, Saudi Arabia.

## **2.3 Analysis and Design of Personalized Recommendation System for University Physical Education**

### **2.3.1 Basic Idea**

This paper starts with the idea of college students' demands for P.E classes, and recommendation of an exercise is given according to the user's physicality/Fitness.

### **2.3.2 Methodologies**

The system function can be divided into four parts: input module, personalized processing module, output module, and user management module, in which the personalized processing module is the core of the whole system.

#### **2.3.2.1 Input Module**

Mainly responsible for the collection and update of users' information along with the personal preference of students in sports, including the degree of endurance, preference of skill, space and number of participants etc.

#### **2.3.2.2 Personalized processing module**

This is the critical part of the system, which directly determines the performance of the system, it includes the following functional parts : confirm the identity of students in order to provide different students with different recommendations; collect students' personal preference about sports and their physical information; after the students pass the certification, the system obtains their records about personal preference and physical test; generate recommended program of physical education course according to the preference information combined with the physical conditions and the main training objectives of various courses.

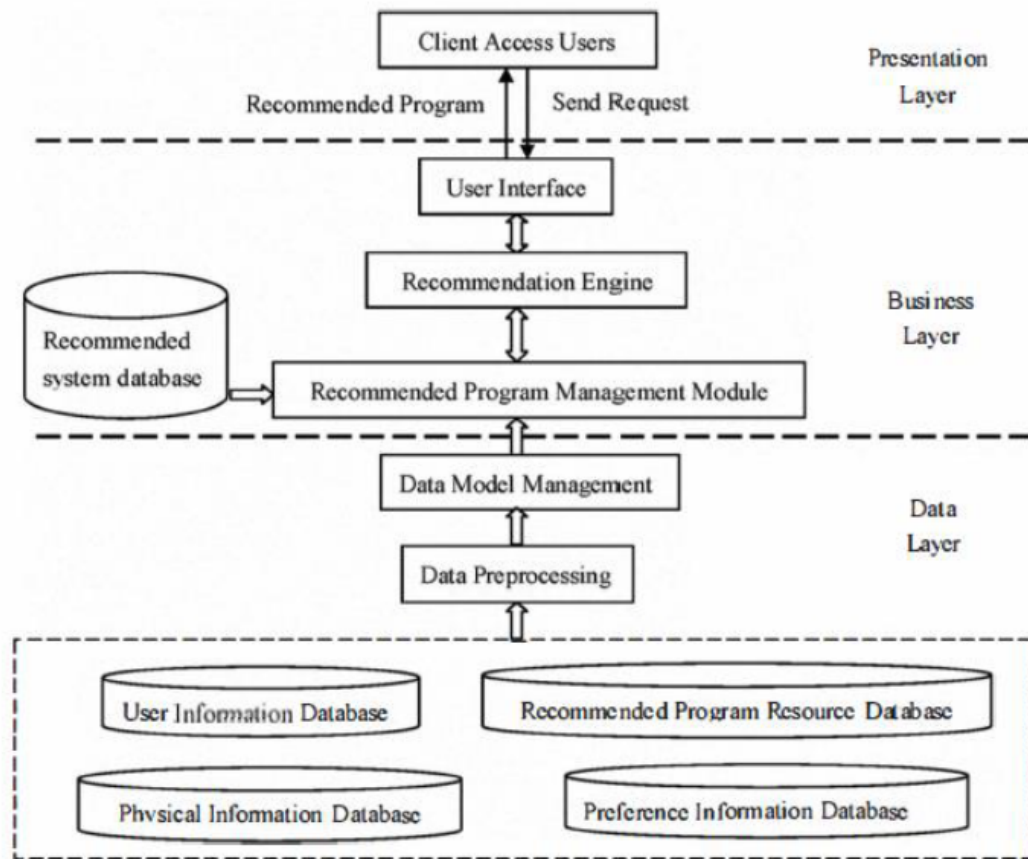


Figure 2.1: System Application Architecture Diagram

### 2.3.2.3 Output module

This should display various types of course programs recommended to the users. The programs include recommended courses, recommended reasons, the relevant venues information, other similar courses, other sports proposals.

### 2.3.2.4 User management module

This is to manage the basic information for students and teachers' permissions.

The whole design of the system is shown in Figure 2.1

### **2.3.3 Results**

The recommended process in the system designed in this paper is unidirectional. Results of the analysis will be recommended to student's; students can view the recommendation of the different results by changing the personal information input. Evaluations on the accuracy of the recommending system itself hasn't been implemented in this paper.

### **2.3.4 Limitations**

- No two-way Interaction with the system through students' giving feedback and system's modification.
- Just a concept, implementation hasn't been done yet.
- Uses Just Featured Based Collaborative Filtering (FBCF) with Pearson correlation algorithm

## **2.4 An Approach to a University Recommendation by Multi-criteria Collaborative Filtering and Dimensionality Reduction Techniques**

### **2.4.1 Basic Idea**

This paper proposes a University Recommendation System (URS), which provides the University or Engineering College recommendations to students, where should they apply for admission. URS makes use of Multi-Criteria Item-Based Collaborative Filtering and Dimensionality Reduction techniques to generate high quality recommendations.



### 2.4.2 Methodologies

- Step 1: Generate the 3-order (student-college-criteria) tensor the from the interaction record i.e., multi-criteria ratings submitted by students for colleges.
- Step 2: HOSVD with PCA mean is applied on the 3-order tensor for dimensionality reduction to get best approximation of ratings.
- Step 3: After tensor decomposition and tensor approximation, the lower dimensional approximated data is used for similarity evaluation using cosine-similarity measure.
- Step 4: Selecting the active student and active colleges and predicting the individual criteria rating using the neighborhood formation for predicting the unknown overall rating.
- Step 5: After overall rating prediction, proposed algorithms make the predictions and list of Top-N College recommendations for the students.

### 2.4.3 Results

From Figure 2.2, it can be seen that both the curve increases as the dataset size increases. However, the precision value of the proposed algorithm is larger than MC-IB-CF method. Which means more accurate results can be obtained using proposed method.

From the Figure 2.3 it can be seen that both the curve increases as the dataset size increases. However, the F1 value of the proposed algorithm is larger than MC-IB-CF method. Which means more accurate results can be obtained using proposed method.

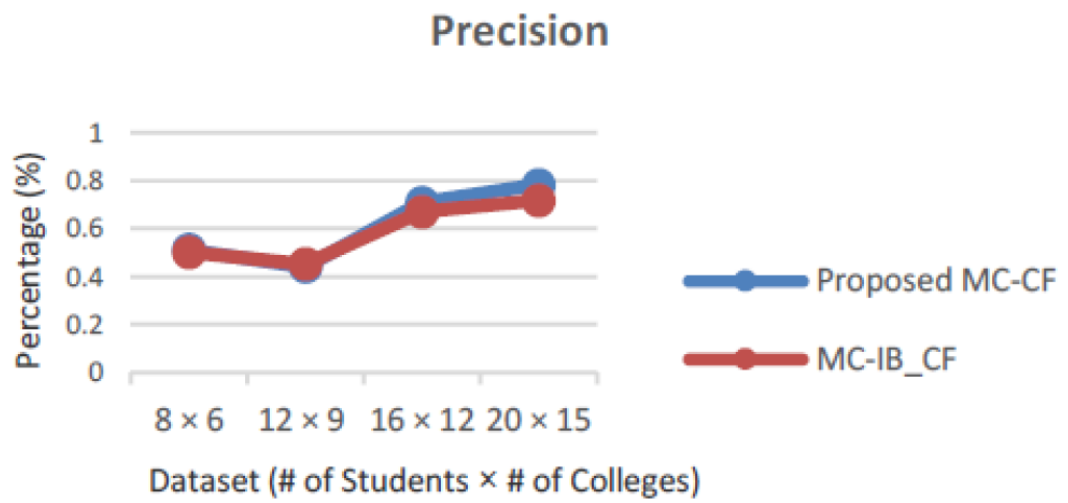


Figure 2.2: Precision

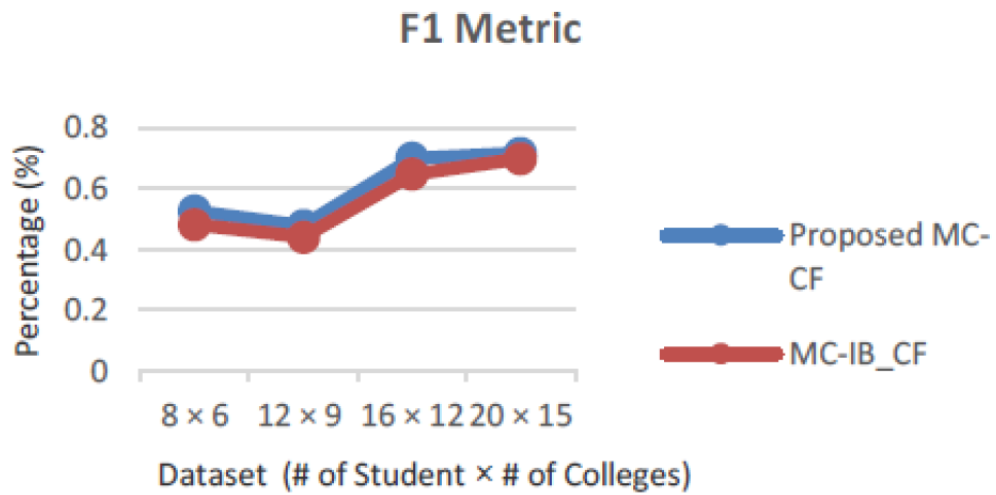


Figure 2.3: F1 Metric

#### **2.4.4 Limitations**

- Affordability and the location of the institute itself isn't taken into consideration.
- Due to sparsity in rating i.e., few Students have rated the same University/College, which results slightly increase in the Execution Time.

# **Chapter 3**

## **Project Vision**

### **3.1 Problem Statement**

On the Internet today, an overabundance of information can be accessed, it becomes difficult for the users to process and evaluate options and make appropriate choices, especially in the case for choosing the right educational institute which is a very key factor for a student's future. The traditional method of getting recommendation is to either search global universities ranking on the internet or asking other peers for their suggestions. This is both ineffective and unreliable.

### **3.2 Proposed Statement**

This project addresses this problem by proposing a system that provides a list of educational institutes that best suits the user preferences/needs. Different algorithms will be used in the users input such that the list is ranked according to the user's features; the university's geographical location, student's academic record/transcript, interests, budget etc. We will use different algorithms that are well-known in the Machine Learning community, which as a result can be used to produce a list of university recommendations.

## 3.3 Motivation

The motivation behind this project is to help young adults decide which educational institute is best according to their interest, academic profile, geo-graphical location and most importantly budget. This is a crucial decision in a student's life, since a wrong decision will either make the student lose their interest in their designated field or perform poorly because of the institute environment. Nowadays, majority of the overseas students are applying for universities for further education in Pakistan. Unlike the students who have a Pakistani background, overseas students will have a hard time finding and then applying for the university. This platform will also assist those parents who have an uneducated background and doesn't know the technical details of finding the university which is best suited for their son/daughter.

## 3.4 Objective

Any user who are seeking for an Educational Institute are overloaded with information on the internet. Our main objective is to have a platform which allows the students to see every option which is relevant to them. The recommendation system will rank institutes based on how closely they match a user's preference. The user's preferences might includes its location, the amount of budget allocated, academic profile etc.

## 3.5 Project Scope

Our target audience primarily focuses on all types of students and parents. Student's might include pupils who are looking for secondary schools or below, or undergraduates/postgraduates students seeking for universities for their bachelors/masters/PHD's. Our platform is also designed for parents so that they could finds whats best suited for their son/daughter and will make sure that every user could get their recommendation based on their preference and needs. This could be expanded Internationally since the only thing needed is the data.

## 3.6 Constraints

Our project is fully dependent on how much data we have, and this depends on time which we are limited with. For now, we will focus more on undergrads students. But, we will definitely expand our project and will collect more data so that any type of student (anywhere in the world) would be eligible.



# Chapter 4

## Software Requirement Specifications

### 4.1 Implementation

Figure 4.1 shows a very abstract view of the interaction between the user and the system. This process starts with the user interacting with the chatbot on the REACT Application, once the features of the users are extracted by the chatbot, it will be compared with the features of already enrolled students which are stored in MongoDB. Python script is run by using `child-process.spawn()` from NodeJS and the data is retrieved by using `py-mongo` from the database. Once the data is retrieved, different machine learning algorithm can be applied and the recommendations will be sent back to the database, so that the data is retrieved using Nodejs and Express and is displayed in React to the end user.

### 4.2 Software Architecture Diagram

Figure 4.2 shows the Software Architecture diagram

### 4.3 Use Case Diagram

Figure 4.3 shows the use case diagram



## 4.4 Swimlane Diagram

Figure 4.4 shows the swimlane diagram

## 4.5 Web Application

This section focuses on the frontend and backend of our MERN stack application

### 4.5.1 Backend

The backend (or “server-side”) is the portion of the website you don’t see. It’s responsible for storing and organizing data, and ensuring everything on the client-side actually works. We used NodeJS and Express for the backend and MongoDB to store the data.

#### 4.5.1.1 MVC Framework

The Model-View-Controller (MVC) framework is an architectural/design pattern that separates an application into three main logical components Model, View, and Controller. Figure 4.5 shows this architecture.

#### 4.5.1.2 Controller

The controller is the component that enables the interconnection between the views and the model so it acts as an intermediary. The controller doesn’t have to worry about handling data logic, it just tells the model what to do. It process all the business logic and incoming requests, manipulate data using the Model component and interact with the View to render the final output. The controller used in our projects are explained below

**UserController:** Responsible for logging/signing in the user. For the registration of user (signing in). We took all the relevant data from the user along with their email and password. A salt (random characters) is generated and is hashed with the password which is

Table 4.1: UsersModel

Name	Type
email	String
password	String
fullname	String
CNIC	String
Gender	Boolean
DOB	Date
Nationality	String
HomeAddress	String
ContactNo	String
FatherName	String
FatherCNIC	String
FatherContactNo	String
FatherOccupation	String
MotherOccupation	String

Table 4.2: UniversityModel

Name	Type
university-name	String
campus-location	String
private/public	String
HEC-recognized	Boolean
campus-size	String
acceptance-rate	String
Approx. Expense	Float
hostel-accommodation	Boolean
loan/scholarship	Boolean

stored in the database. When logging in that same hash is generated and is compared with the hashed that is stored in the database. If it matches than the user is navigated to the dashboard.

**UserDashboardController:** Responsible for displaying user's profile by **getting** the data from database as well as getting data of different universities for **search tab**. It is also responsible for storing user's preferences in the **counsil tab**.

#### 4.5.1.3 Model

The Model component corresponds to all the data-related logic that the user works with. The models used are given in Table [4.1](#), [4.2](#), [4.3](#)

#### 4.5.1.4 View

The View component is used for all the UI logic of the application. It generates a user interface for the user. Views are created by the data which is collected by the model component but these data aren't taken directly but through the controller. It only interacts with the controller. This is discussed in Section [4.5.2](#)

Table 4.3: UserDataModel

Name	Type
CNIC	String
budget	Number
Hobbies-Interest	String
study-group	String
Preferred-Location	String
transcript	File

## 4.5.2 Frontend

The front-end application, commonly known as the interface of an application, is the layer or element that the user has the ability to use, see, and interact with through buttons, images, interactive elements, navigational menus, and text. Our frontend application is made using REACT.

### 4.5.2.1 UI Screens

Screenshots of UI is given in Figure [4.6](#), [4.7](#), [4.8](#), [4.9](#), [4.10](#). These are all subject to change.

## 4.6 Functional Requirement

- The user has the ability to sign-in or register themselves by relevant information.
- The user has the ability to log-in to their personal dashboard.
- The user has a personalized dashboard from which they could navigate using sidebar.
- The user could see all the personal information in the home tab.
- The user could explore different universities and their corresponding features in the search tab.
- The user could interact with chatbot to upload his/her preferences.

- The user could manually enter their data (budget, study-group, Image of transcript etc.) on counsel tab.
- The user get a list of recommendations along with features of those recommendations on why its being recommended.
- The user has the ability to log out.

## **4.7 Non-Functional Requirement**

- The user can only sign in by giving data that is legitimate. For Example, A valid email address, a strong password etc.
- The password is hashed and is stored in hash form in the database.
- The website is secure and no-one shouldn't be able to access ones data.
- The interaction between chatbot is both confidential and acceptable

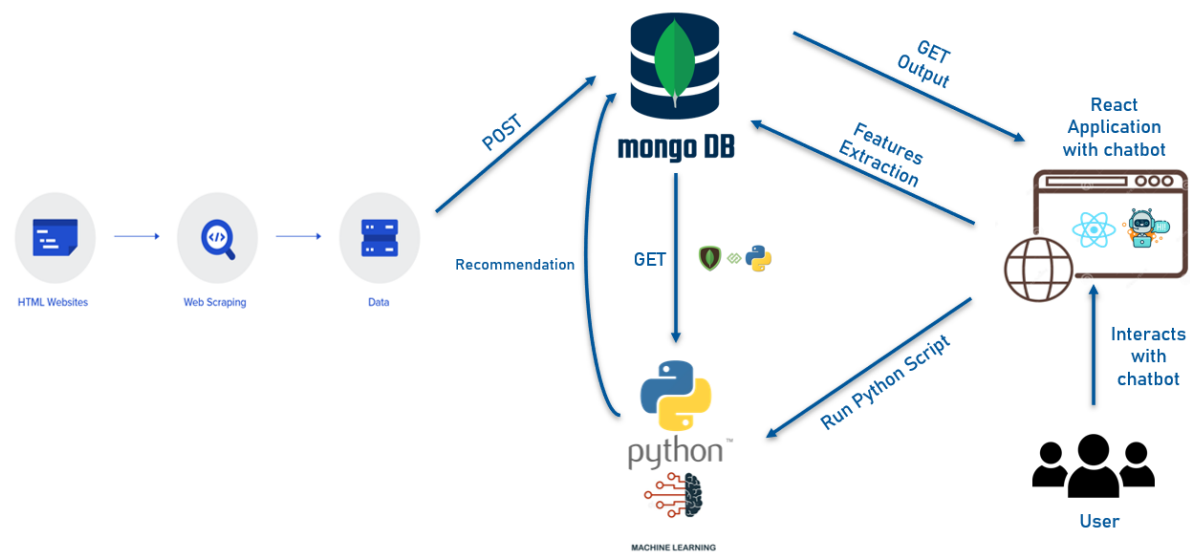


Figure 4.1: Abstract Implementation Diagram

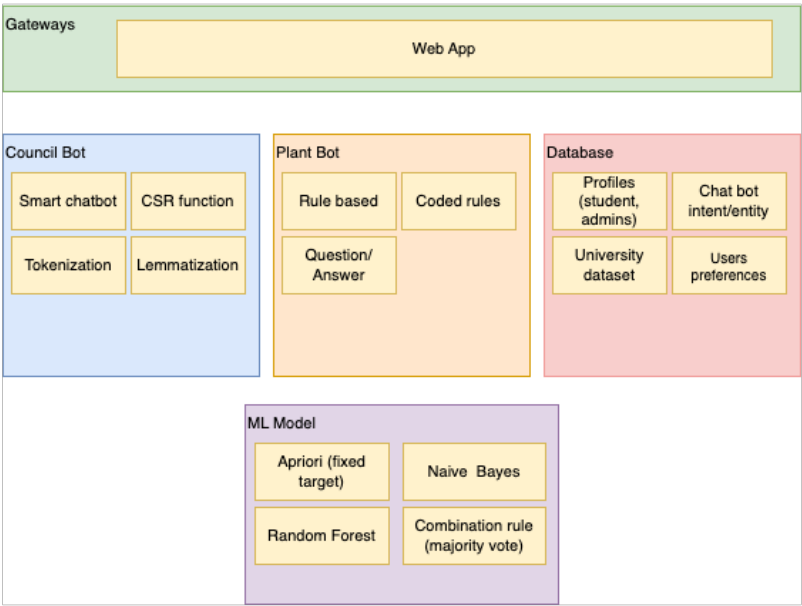


Figure 4.2: Software Architecture Diagram

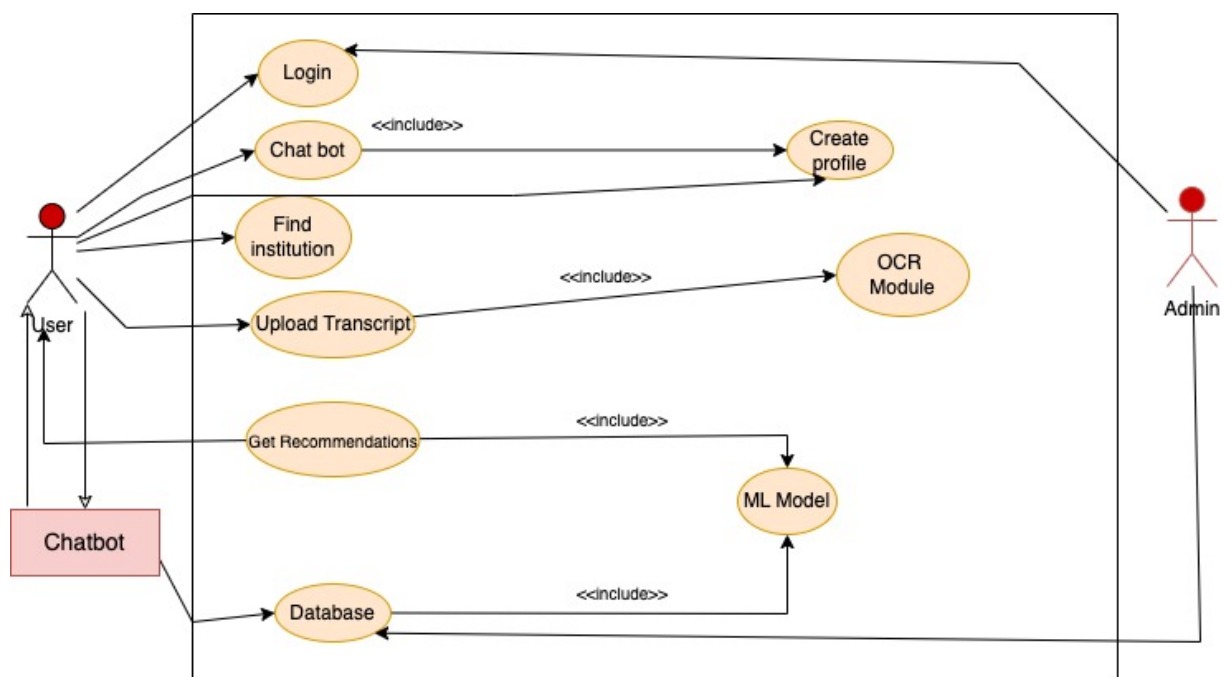


Figure 4.3: Use Case Diagram

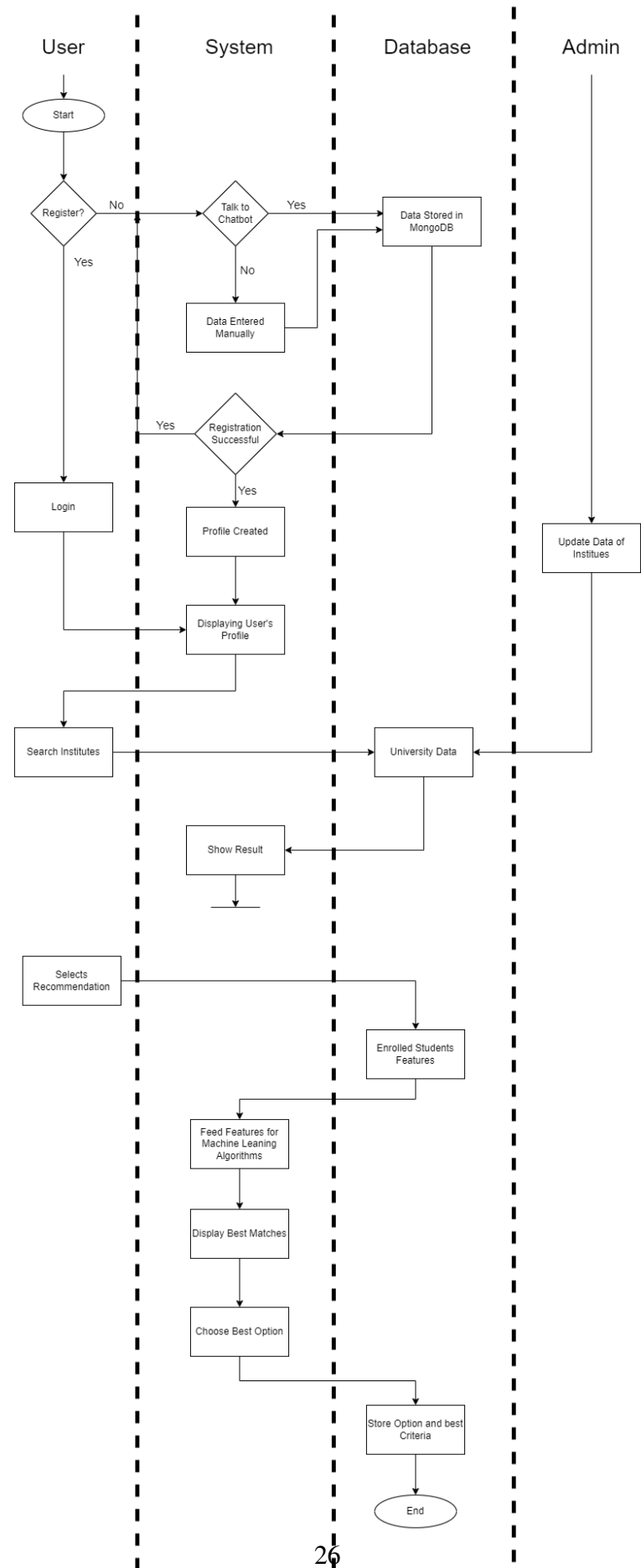


Figure 4.4: Swimlane Diagram

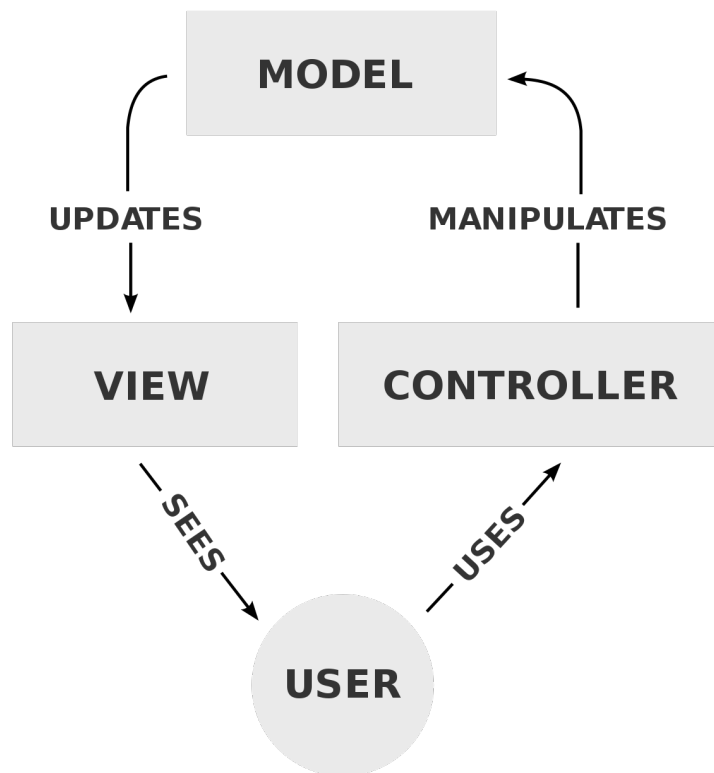


Figure 4.5: MVC Framework



The screenshot shows a web application interface for a 'Counsellor'. At the top, there is a header bar with the text 'Counsellor' on the left and 'Login Signup' on the right. The main content area is light gray and contains a central white box with a subtle drop shadow. Inside this box, the title 'Log In' is centered at the top. Below the title, there are two input fields: 'Email address:' followed by a text box, and 'Password:' followed by a text box. At the bottom of the box is a 'Log in' button.

Figure 4.6: Logging In

The screenshot shows a web application interface for a 'Counsellor'. At the top, there is a header bar with the text 'Counsellor' on the left and 'Login Signup' on the right. The main content area is light gray and contains a central white box with a subtle drop shadow. Inside this box, the title 'Registration' is centered at the top. Below the title, there are several input fields arranged in two columns. The left column contains: 'Full Name' (text box), 'Date of Birth' (calendar icon and text box with 'dd-yyy' format), 'Contact No.' (text box), 'Father's CNIC' (text box), 'Home Address' (text box), and 'Enter Your New Email' (text box). The right column contains: 'CNIC' (text box), 'Nationality' (text box), 'Father's Name' (text box), 'Father's Contact No.' (text box), and 'Gender' (dropdown menu with '--Choose--' selected). At the bottom of the box is a 'Sign up' button.

Figure 4.7: Signing In/Registration Page

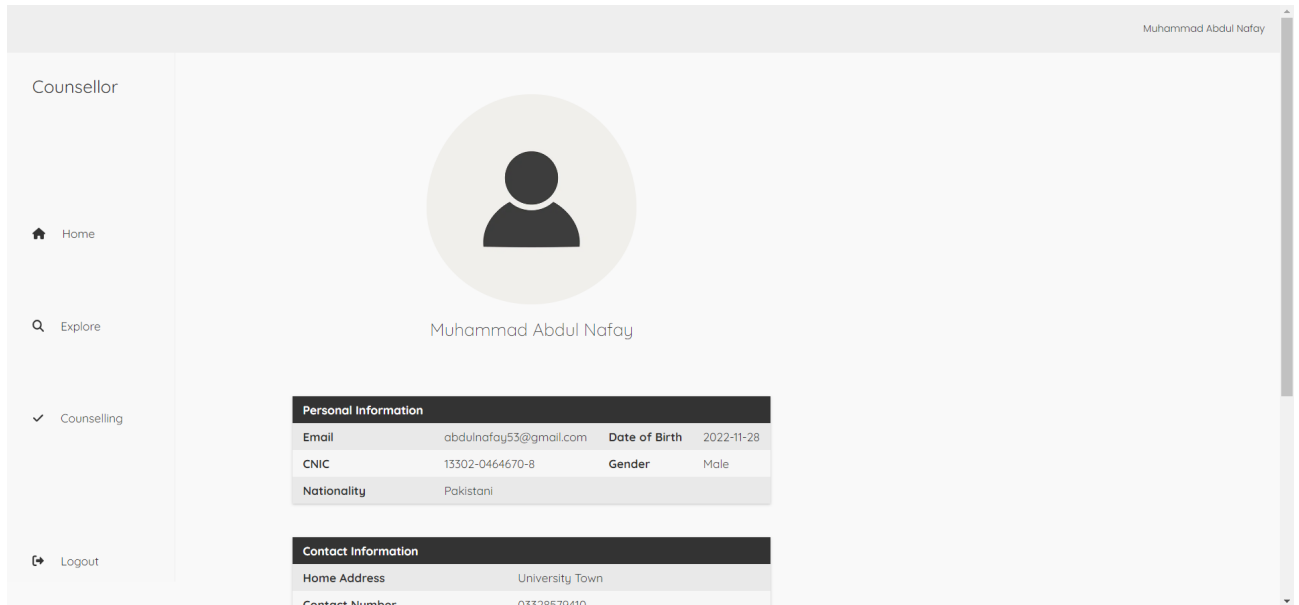


Figure 4.8: Home Tab/Dashboard

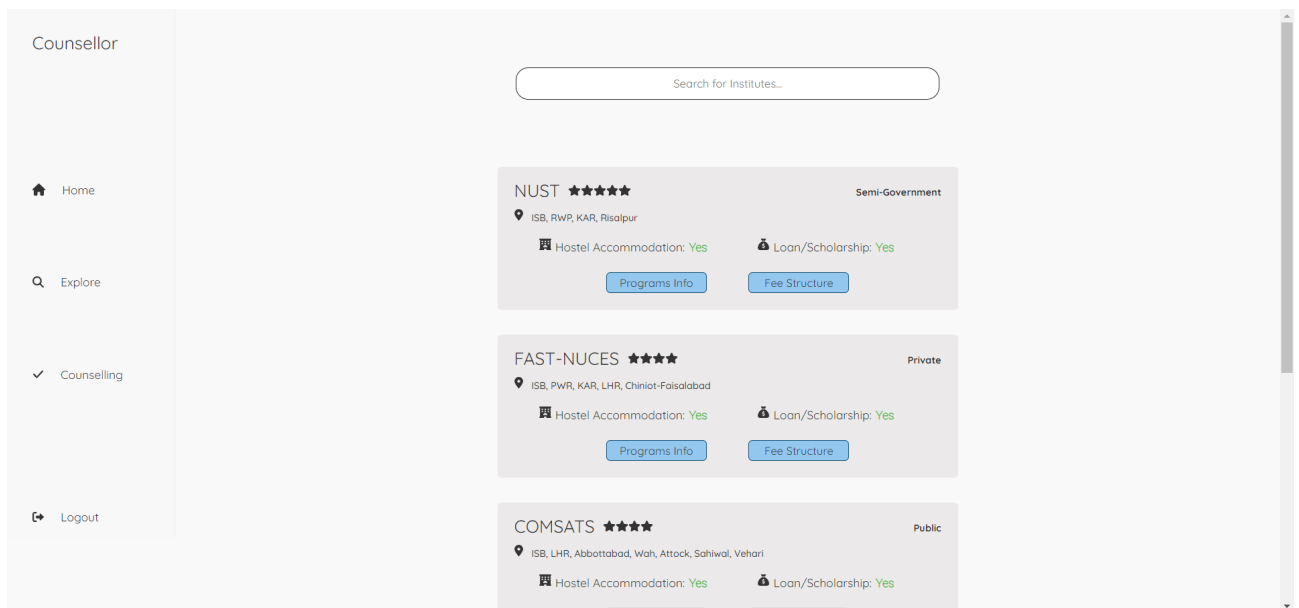


Figure 4.9: Search/Explore Tab

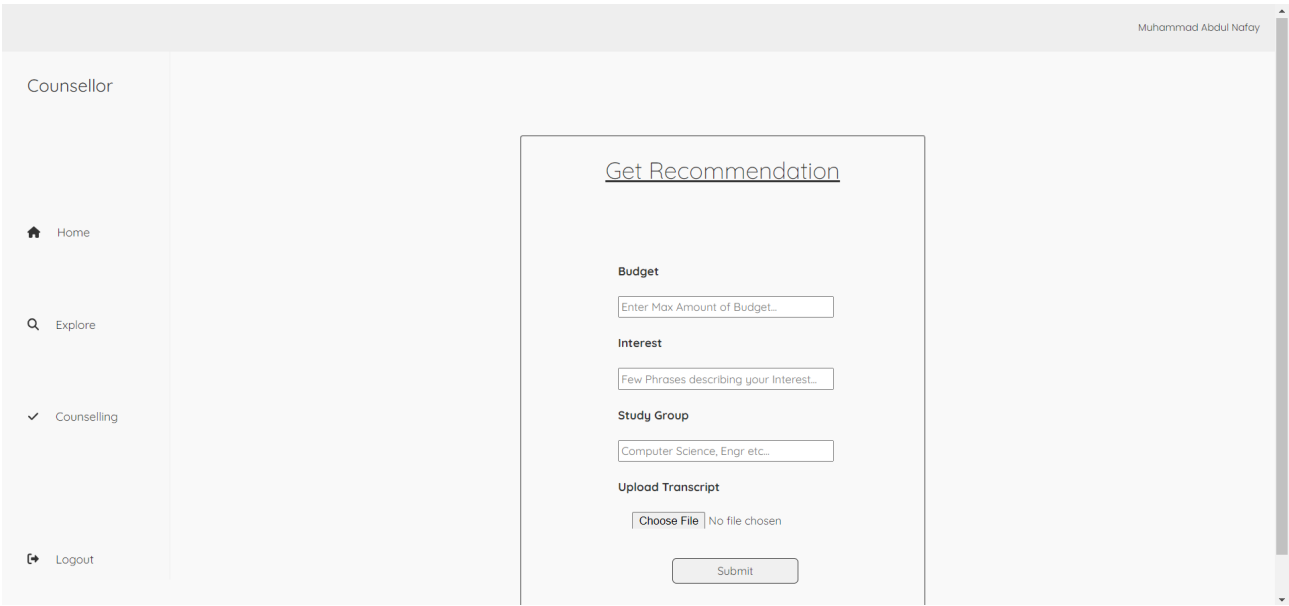


Figure 4.10: Counsel Tab

# Chapter 5

## Iterations 1

### 5.1 Methodologies

#### 5.1.1 Council Bot

CouncilBot is a Self-learning, smart chatbot that is meant to handle all of Councils customer interactions. Users would be able to converse with CouncilBot and receive responses that are easy to understand and relevant to their problems. CouncilBot operates on NLP techniques such as Tokenization and Lemmatization to completely understand a user's statement and generate a valid/acceptable response.

##### 5.1.1.1 Tokenization

Tokenization is the breaking down of sentences into their smallest units called tokens. Punctuation, words and other marks would be considered tokens by grouping them together. These tokens can later be used in different models, or can be grouped into types and tagged for intent files [5].

### 5.1.1.2 Lemmatization

Lemmatization is a method of finding a simpler but related word. It uses lexical knowledge to get base forms of words. NLTK provides WordNetLemmatizer class which is a thin wrapper around the wordnet corpus. This class uses morphy() function to the WordNet CorpusReader class to find a lemma [5].

The purpose of CouncilBot is to deal with all new users coming to the application for the first time, it can help build a user's profile by taking in their personal information (if they feel so inclined, otherwise they can manually enter it themselves) by using multiple customer service techniques used in NLP. CouncilBot will also provide recommendations generated from the Machine Learning Module.

### 5.1.2 Plant Bot

PlantBot is a rule based chat bot that chooses response based on predefined set of responses set by the programmer Rule-based model chatbots are the type of architecture which most of the first chatbots have been built with, like numerous online chatbots. The knowledge used in the chatbot is humanly hand-coded and is organized and presented with conversational patterns. A more comprehensive rule database allows the chatbot to reply to more types of user input. However, this type of model is not robust to spelling and grammatical mistakes in user input. Most existing research on rule-based chatbots studies response selection for single-turn conversation, which only considers the last input message. In more human-like chatbots, multiturn response selection takes into consideration previous parts of the conversation to select a response relevant to the whole conversation context.

Sentiment analysis (or opinion mining) is a technique used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in student feedback, and understand student needs.

The purpose of PlantBot is purely for data collection from the students already enrolled in universities in Pakistan. Since we did not find any efficient method of gathering data

on these online and we require a dataset for our Machine learning module. Since this is a cold start problem, it will take some time to generate the entire dataset by first putting plantbot out there to enrolled students to interact with and provide their information, through a simple web app. This would then be stored in our database.

### **5.1.3 Machine Learning to generate recommendations**

The machine learning module consists of 3 classifiers working in conjunction with each other to provide the best result. Currenrtly these classifiers are NaiveBayes, RandomForest, and Apriori algorithms. They will all classify the dataset on the class attribute. Results will then be put through a Majority vote to decide the best recommendation.

#### **5.1.3.1 Decision Tree Classification**

Decision Trees is a Supervised Machine Learning Algorithm that uses a set of rules to make decisions, similarly to how humans make decisions. DecisionTreeClassifier is a class capable of performing multi-class classification on a dataset. In case that there are multiple classes with the same and highest probability, the classifier will predict the class with the lowest index amongst those classes. The intuition behind Decision Trees is that you use the dataset features to create yes/no questions and continually split the dataset until you isolate all data points belonging to each class.

#### **5.1.3.2 RandomForest Classifier (RFC)**

Random forest is an ensemble machine learning algorithm. It is perhaps the most popular and widely used machine learning algorithm given its good or excellent performance across a wide range of classification and regression predictive modeling problems.



# Chapter 6

## Iterations 2

### 6.1 Use Case Diagram

Figure [6.1](#) shows the use case diagram.

### 6.2 Activity Diagram

Figure [6.2](#) shows the use case diagram.

### 6.3 Component Diagram

Figure [6.3](#) shows the Components diagram.

### 6.4 Updated UI Screens

Screenshots of Updated UI is given in Figure [6.4](#), [6.5](#), [6.6](#), [6.7](#), [7.6](#). These are all subject to change.



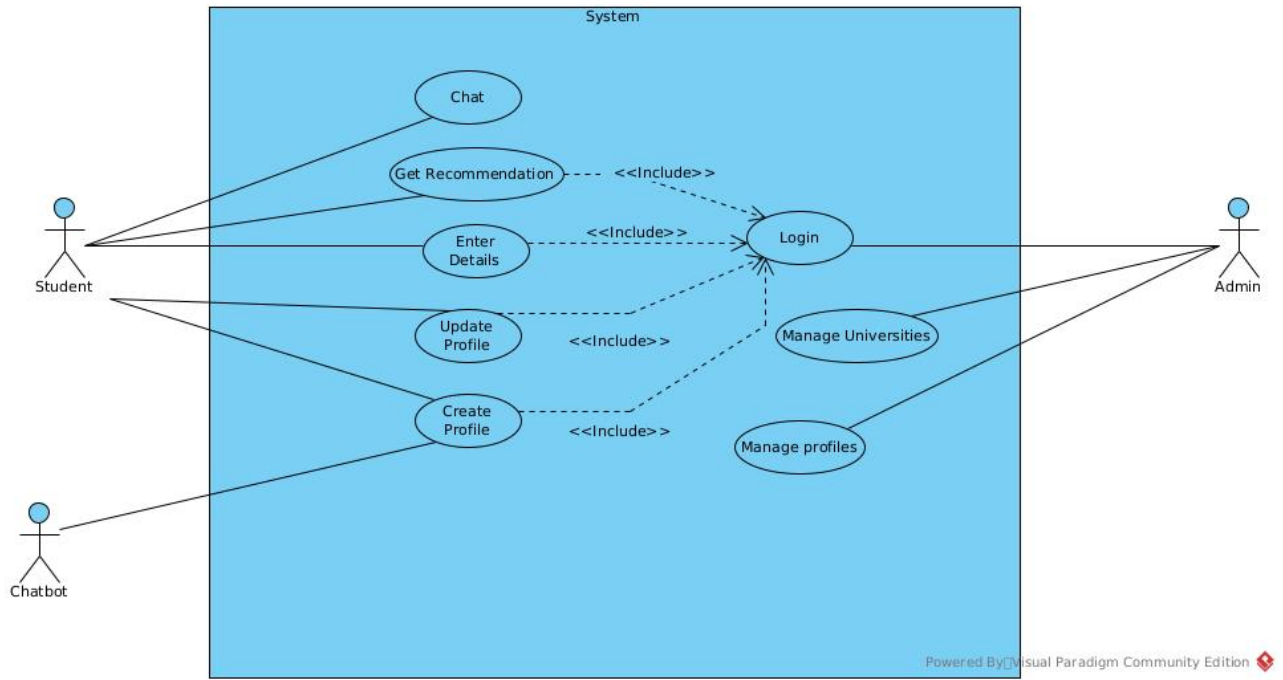


Figure 6.1: Use Case Diagram

## 6.5 Data Collection

Our main purpose of this whole project is to recommend user's universities based on their preferences. Every user has different requirements of why he/she choose to get enrolled in a specific university. Now to train our machine learning model which we will discuss later on. We need to train a data set containing preferences of different enrolled users in multiple universities. To do that, we need to collect the data first, which needs a lot of time. In this moment of time, we want to train our models on dummy data set just to see the accuracy and the overall result. The Table 6.1 shows the column headings and their corresponding data type which we will use in our project. This data set contains demographics of the user, the total budget (approx.) which was required in their degree program, parents information, and the user's academic profile and finally the university which they got enrolled in. Assuming we have this data, we can apply a supervised machine learning algorithm with the user's preferences as input and the corresponding university will be our class attribute which will be the output of our model.

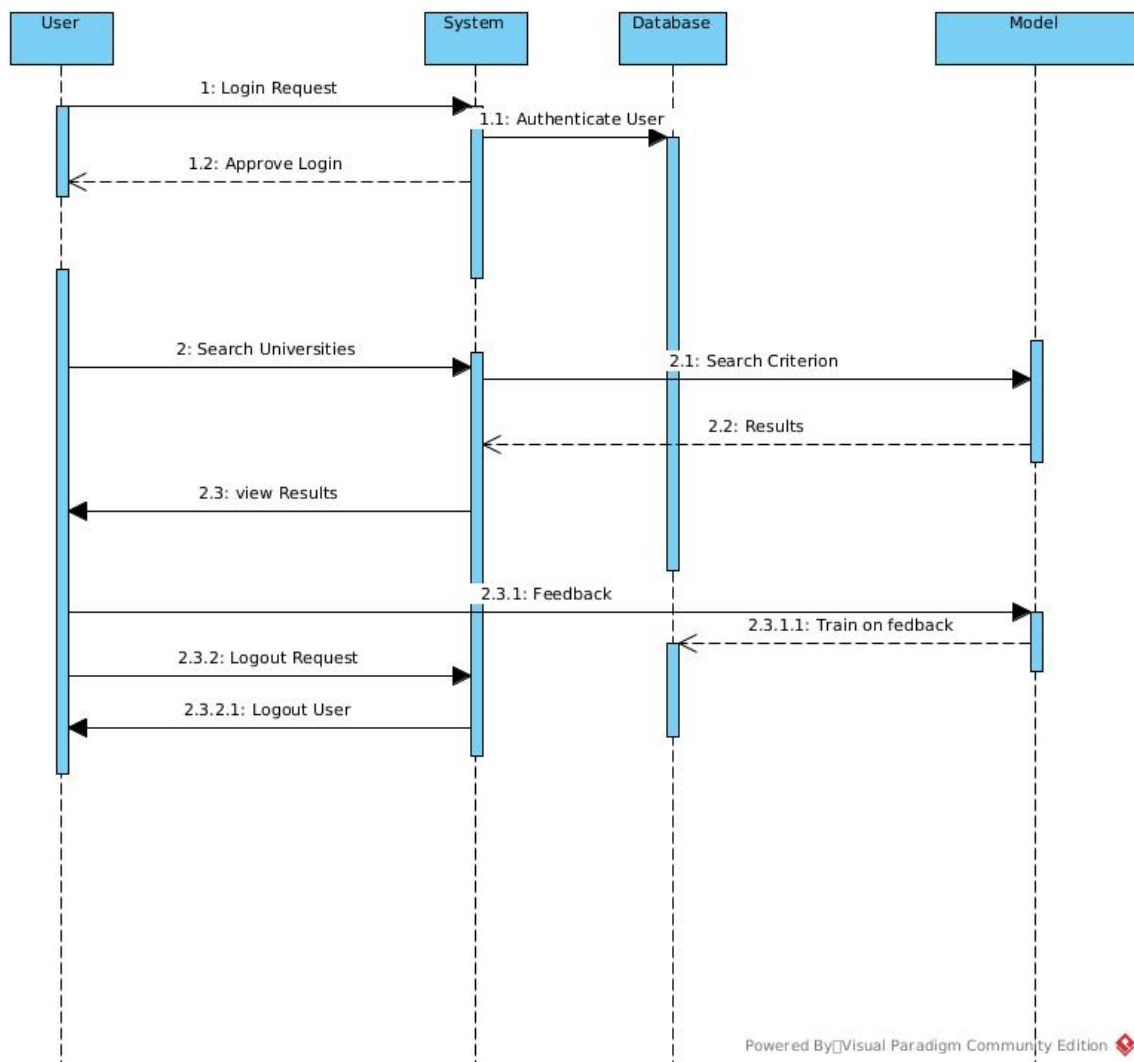


Figure 6.2: Activity Diagram

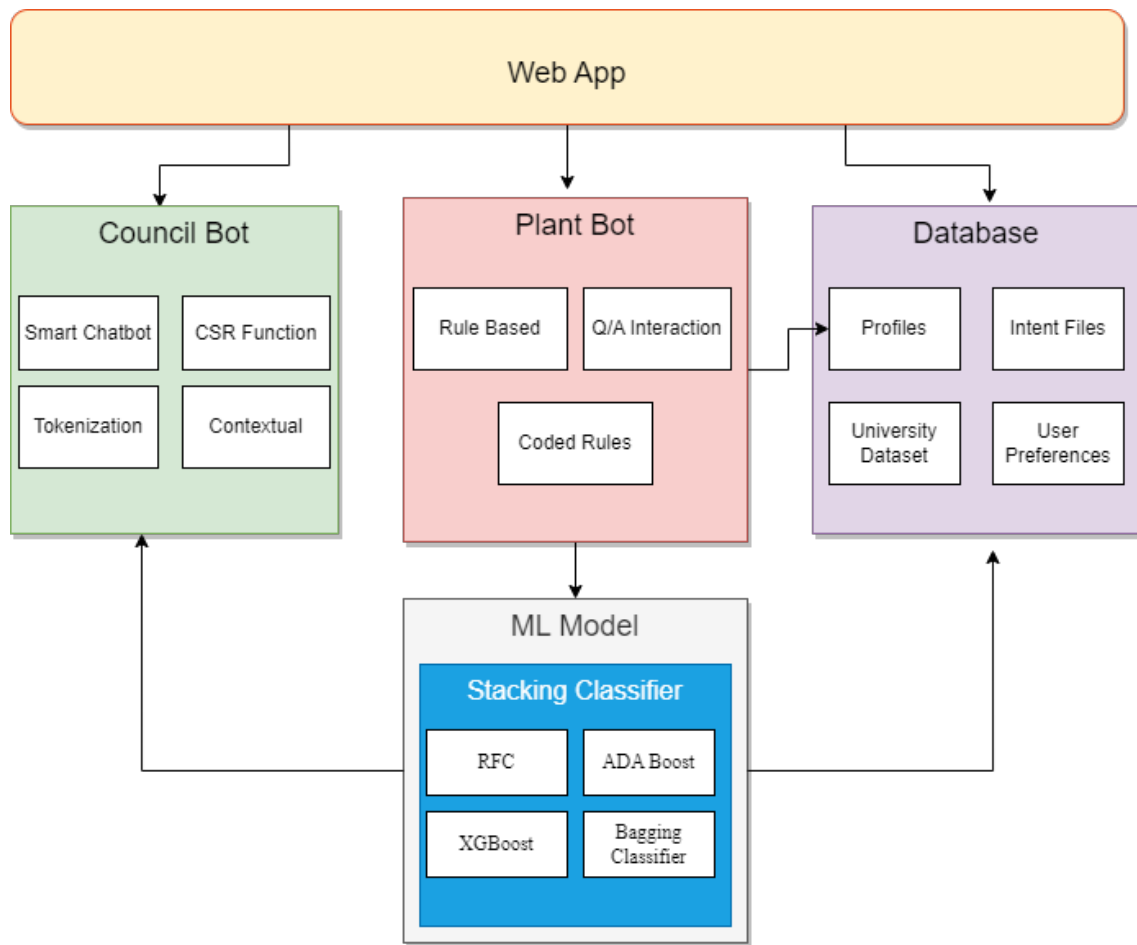


Figure 6.3: Component-Diagram

Table 6.1: CleanData.csv

Name	Type
Gender	String
Race/Ethnicity	String
Home-City	String
Preferred-Language	String
Hobby-Interest	String
Budget (in millions)	Number
FatherOccupation	String
MotherOccupation	String
Study-Group	String
Degree-Program	String
Matric-Marks	Number
Inter-Marks	Number
Year of Admission	String
Campus	String
University	String

The screenshot displays the 'Council' website's login interface. At the top, a navigation bar includes the 'Council' logo, a 'Home' link, an 'About Us' link, and buttons for 'Login' and 'Signup'. The main content area features a 'Sign in' form with a purple lock icon and the text 'Sign in'. The form contains two input fields: 'Email Address \*' and 'Password \*'. Below the password field is a checkbox labeled 'Show Password'. A blue 'LOG IN' button is positioned below the inputs. A link that reads 'Don't have an account? Sign Up' is located at the bottom of the form. The footer consists of a dark bar with the 'Council' logo, a descriptive sentence: 'A project to help young students make the perfect decision in finding a university that best fits their preferences and requirements.', navigation links for 'Home', 'About us', 'Log In', and 'Sign In', and social media icons for Facebook, Twitter, YouTube, Instagram, and LinkedIn.

Figure 6.4: Logging In

The screenshot shows the 'Registration' form on the 'Council' website. The form is titled 'Registration' and includes a note: 'Please fill in the form below. Required fields are marked with an Asterisk (\*)'. The form is organized into several sections. The first section contains 'Full Name \*', 'DOB' (with a date picker), and 'Nationality \*'. The second section includes 'Gender' (a dropdown menu), 'Preferred Language' (a dropdown menu), and 'Race\_ethnicity' (a dropdown menu). Below these is a label 'Please select your first language' and a 'Date of Birth' field with a calendar icon. The third section contains 'Enter Your Home Address' (with a location pin icon), 'Contact no \*', and 'Home City' (a dropdown menu). The fourth section includes 'Father's Name \*', 'Father's DOB' (with a date picker), and 'Father's Contact Number \*'. The fifth section contains 'Father's Occupation' (a dropdown menu) and 'Mother's Occupation' (a dropdown menu). The final section has 'Enter your new Email \*' and 'Enter your new Password \*'. A green 'REGISTER' button is located at the bottom of the form. The website's navigation bar and footer are consistent with the previous screenshot.

Figure 6.5: Signing In/Registration Page

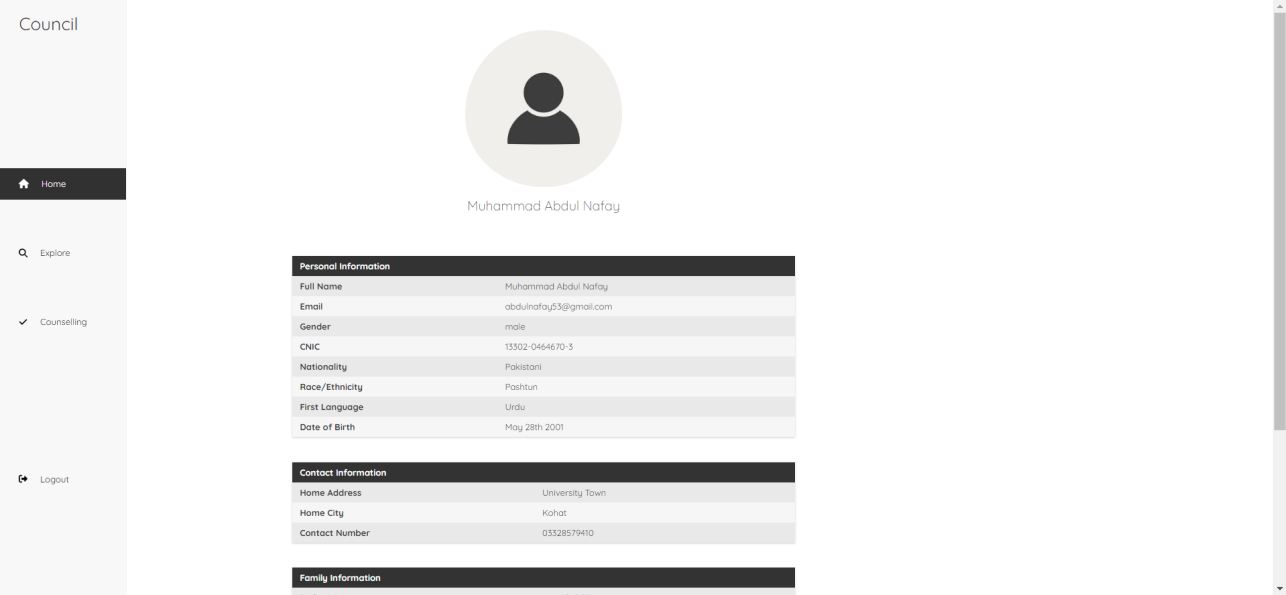


Figure 6.6: Home Tab/Dashboard

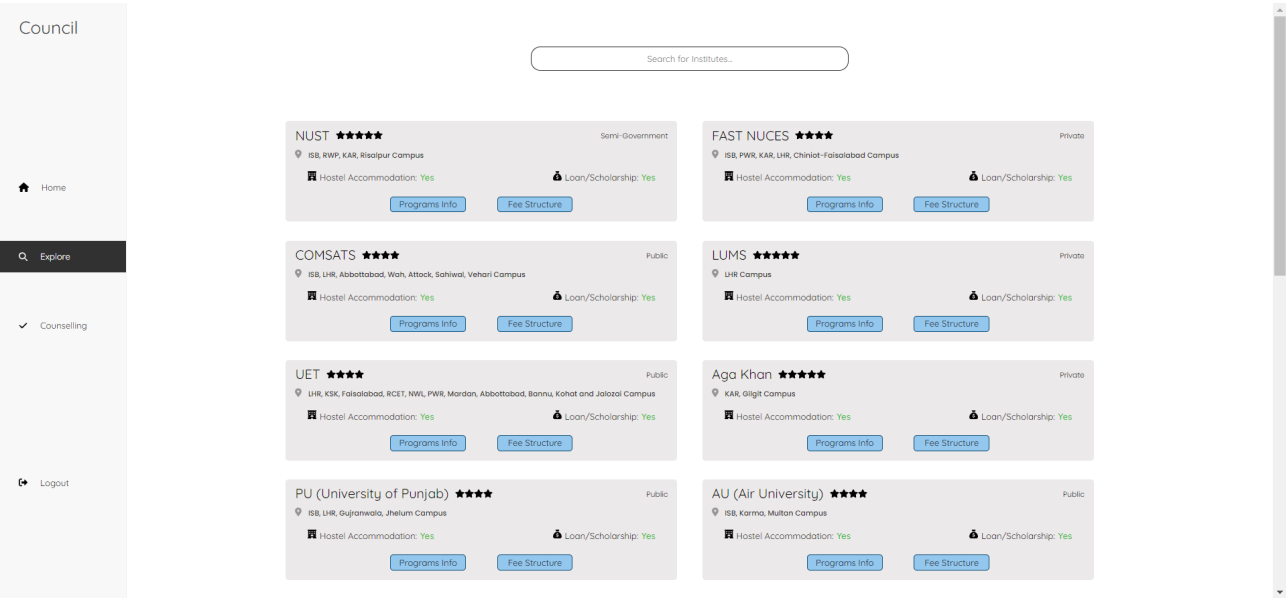
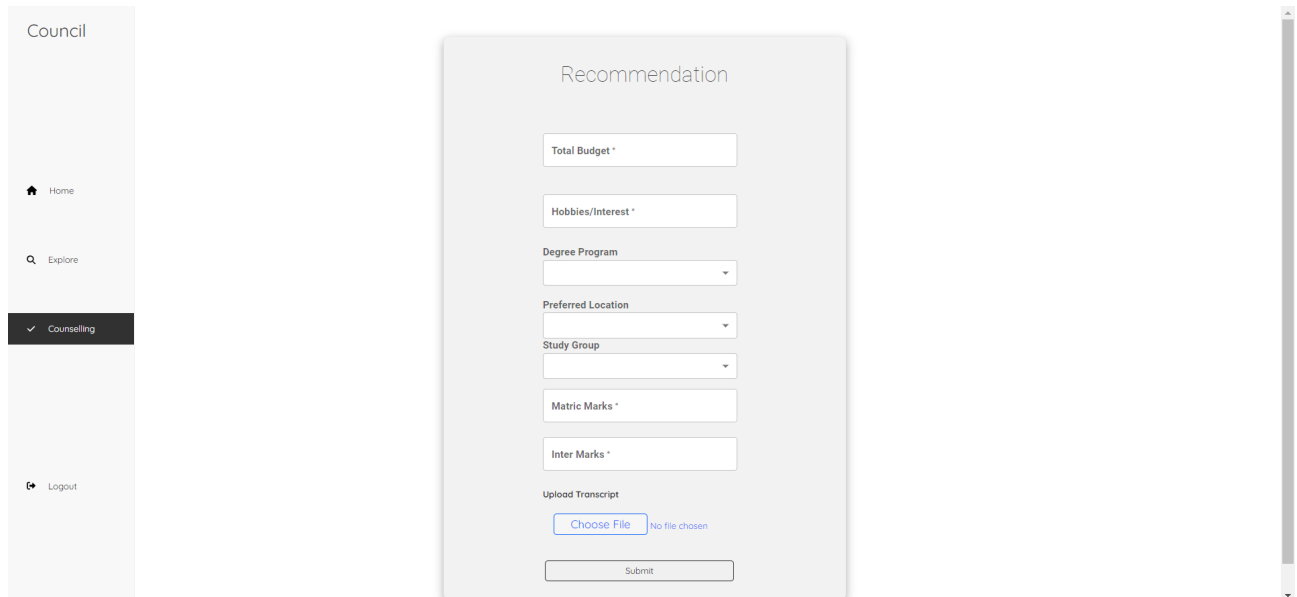


Figure 6.7: Search/Explore Tab



The screenshot displays a web application interface. On the left is a vertical sidebar titled 'Council' containing navigation links: 'Home' (with a house icon), 'Explore' (with a magnifying glass icon), 'Counselling' (with a checkmark icon and highlighted in dark grey), and 'Logout' (with a door icon). The main content area is titled 'Recommendation' and contains a form with the following fields: 'Total Budget \*' (text input), 'Hobbies/Interest \*' (text input), 'Degree Program' (dropdown menu), 'Preferred Location' (dropdown menu), 'Study Group' (dropdown menu), 'Matric Marks \*' (text input), and 'Inter Marks \*' (text input). Below these is an 'Upload Transcript' section with a 'Choose File' button (labeled 'No file chosen') and a 'Submit' button at the bottom.

Figure 6.8: Council Tab

## 6.6 Machine Learning Models

### 6.6.1 Decision Trees Classifier (DTC)

A decision tree is a flowchart-like structure in which each internal node represents a test on a feature (e.g. whether a coin flip comes up heads or tails) , each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules. Figure 6.9 illustrate the basic flow of decision tree for decision making with labels (Rain(Yes), No Rain(No)).

### 6.6.2 Introduction to Ensemble Models

Ensemble Methods, what are they? Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model. An example of a base models in Decision Trees Classifier. When making Decision Trees, there are several factors we must take into consideration: On what features do we make our decisions on? What is the threshold for classifying each question into a yes or no

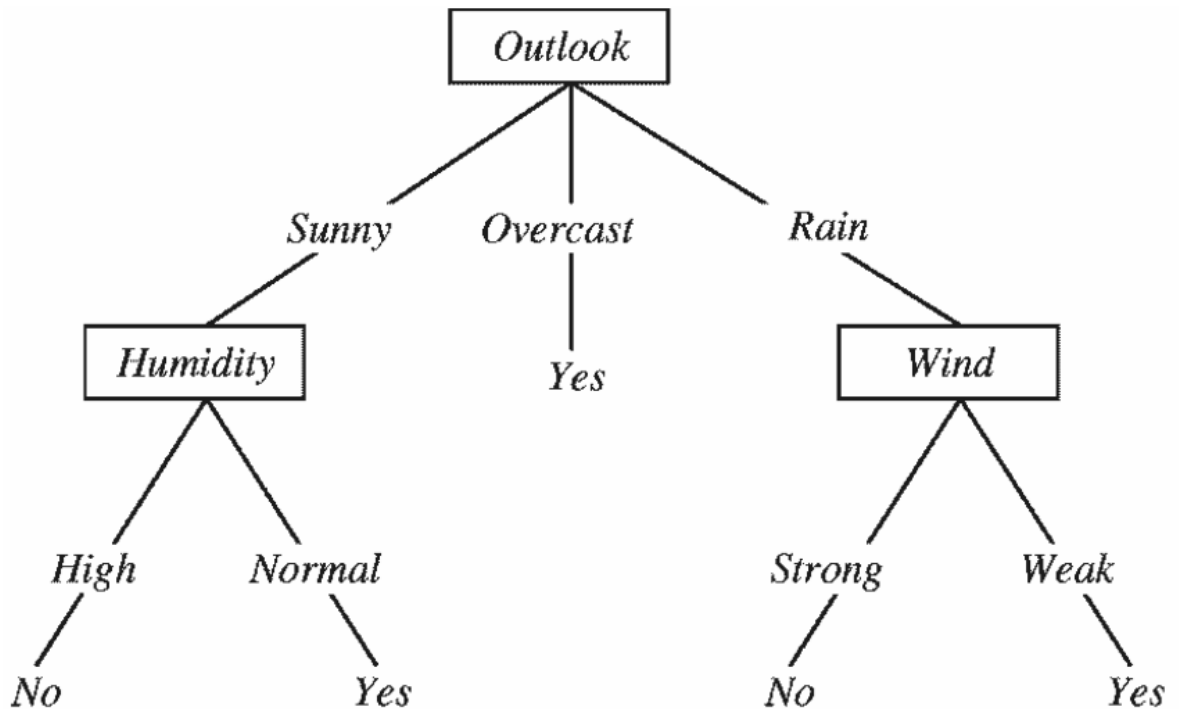


Figure 6.9: Decision Tree Example

answer? In the first Decision Tree, what if we wanted to ask ourselves if we had friends to play with or not. If we have friends, we will play every time. If not, we might continue to ask ourselves questions about the weather. By adding an additional question, we hope to greater define the Yes and No classes. This is where Ensemble Methods come in handy! Rather than just relying on one Decision Tree and hoping we made the right decision at each split, Ensemble Methods allow us to take a sample of Decision Trees into account, calculate which features to use or questions to ask at each split, and make a final predictor based on the aggregated results of the sampled Decision Trees. The ensemble models that we used are discussed below.

#### 6.6.2.1 Bagging Classifier

Bagging, or Bootstrap Aggregating. Bagging gets its name because it combines Bootstrapping and Aggregation to form one ensemble model. Given a sample of data, multiple bootstrapped subsamples are pulled. A Decision Tree is formed on each of the bootstrapped subsamples. After each subsample Decision Tree has been formed, an algorithm

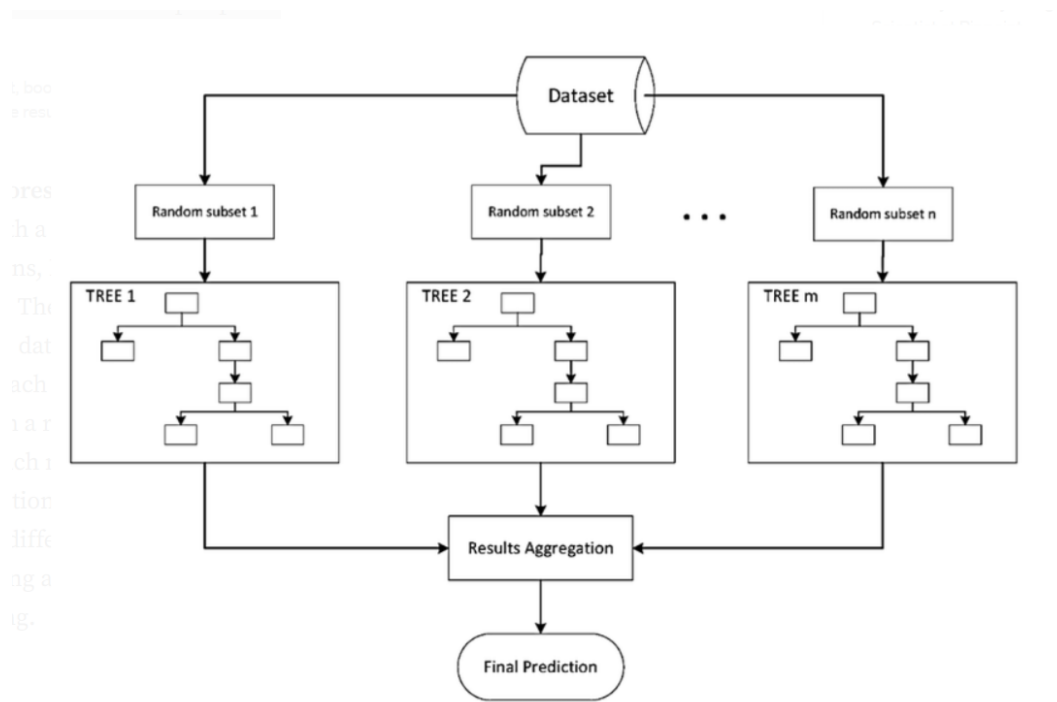


Figure 6.10: Given a Dataset, bootstrapped subsamples are pulled. A Decision Tree is formed on each bootstrapped sample. The results of each tree are aggregated to yield the strongest, most accurate predictor.

is used to aggregate over the Decision Trees to form the most efficient predictor. Figure 6.10 will illustrate this.

### 6.6.2.2 Random Forest Classification (RFC)

Random Forest Models can be thought of as Bagging, with a slight tweak. When deciding where to split and how to make decisions, Bagged Decision Trees have the full disposal of features to choose from. Therefore, although the bootstrapped samples may be slightly different, the data is largely going to break off at the same features throughout each model. In contrary, Random Forest models decide where to split based on a random selection of features. Rather than splitting at similar features at each node throughout, Random Forest models implement a level of differentiation because each tree will split based on different features. This level of differentiation provides a greater ensemble to aggregate over, and producing a more accurate predictor. Refer to Figure 6.11 for a better understanding.



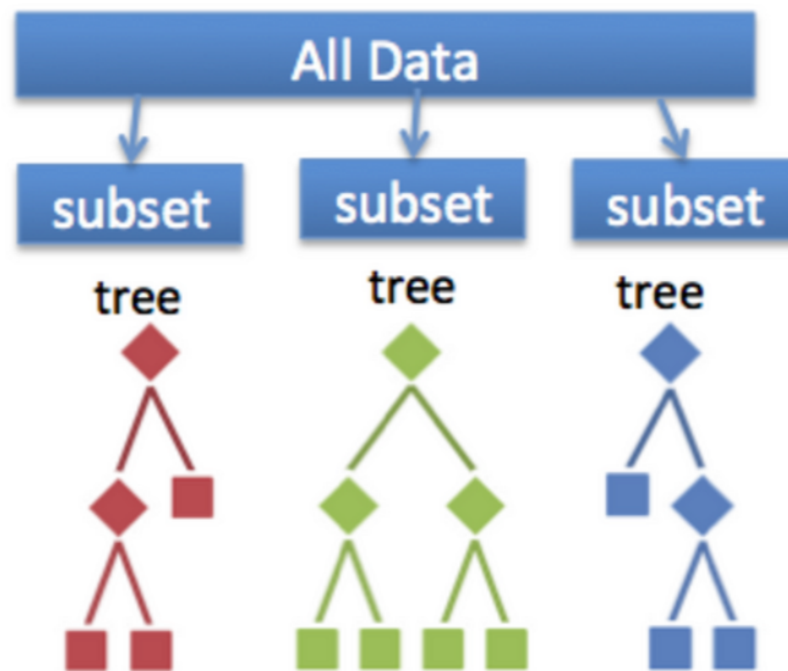


Figure 6.11: A random forest takes a random subset of features from the data, and creates  $n$  random trees from each subset. Trees are aggregated at the end.

### 6.6.2.3 Adaptive Boosting (ADA Boost)

Boosting methods work in the same spirit as bagging methods: we build a family of models that are aggregated to obtain a strong learner that performs better. However, unlike bagging that mainly aims at reducing variance, boosting is a technique that consists in fitting sequentially multiple weak learners in a very adaptative way: each model in the sequence is fitted giving more importance to observations in the dataset that were badly handled by the previous models in the sequence. AdaBoost is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances. Boosting is used to reduce bias as well as variance for supervised learning. It works on the principle of learners growing sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones.

#### 6.6.2.4 XGBoosting

XGBoost stands for eXtreme Gradient Boosting. This is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for “Extreme Gradient Boosting” and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression. One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

#### 6.6.3 Stacking Ensemble Model

Introducing Stacking, an ensemble machine learning algorithm that learns how to best combine each of the models in an ensemble to come up with the best performance.

- An ordinary machine learning model only tries to map input towards output by generating a relationship function.
- Stacking acts on one level above the ordinary by learning the relationship between the prediction result of each of the ensembled models on out-of-sample predictions and the actual value.

The general framework of a stacked ensemble consists of two or more base models (level-0 models) and a higher level meta model (level-1 model) with their functions as below:

- **Base-Models (Level-0 Models):** Models that fit the training data and predict out-of-sample data.
- **Meta-Model (Level-1 Model):** Model that fits on the prediction from base-models and learns how to best combine the predictions.

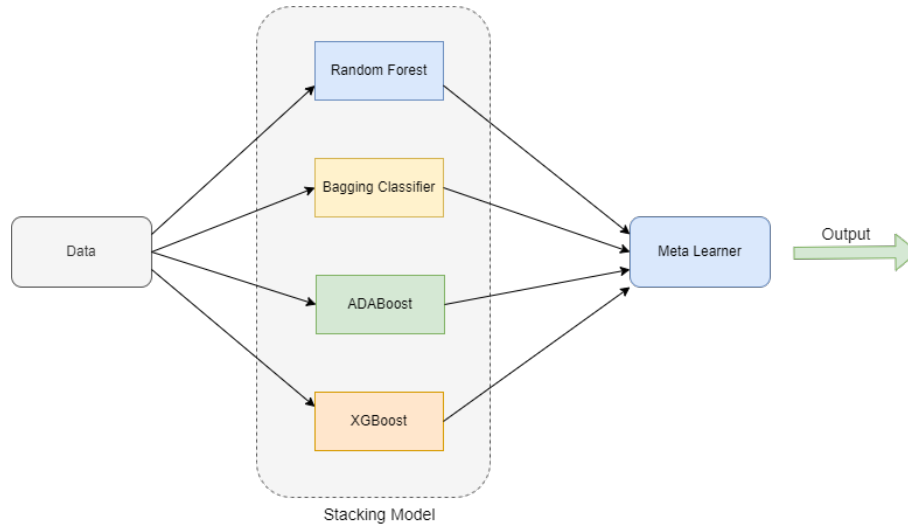


Figure 6.12: Stacking-Ensemble-Model

## 6.7 Algorithm Design

The algorithm design is illustrated in Figure [6.12](#)

## 6.8 Evaluation and Testing

Evaluation on machine learning models are done to confirm that they are performing as expected and that they are good enough for the task they were created for. The evaluation stage is performed after model training is finished. Different techniques are used depending on the type of problem and type of algorithm. There are two methods of evaluating models in, **Hold-Out** and **Cross-Validation**. To avoid over-fitting, both methods use a test set (not seen by the model) to evaluate model performance. We have used cross validation to test our stacking ensemble model. We used the `score(X, y, sample-weight=None)[source]` method which returns the mean accuracy on the given test data and labels. In multi-label classification, this is the subset accuracy which is a harsh metric since you require for each sample that each label set be correctly predicted. Once this is calculated, we used 10 fold cross validation to calculate the models final accuracy. Our model accuracy is shown in Figure [6.13](#)

```
_id: ObjectId('6421f2d5c89f1a4fa0c8eb6b')  
Stacking_Model: BinData(0, 'gASV1hIBAAAAACMGnNrbGVhcm4uZW5zZW1ibGUuX3N0YWNraw5nIiwSU3RhY2tpbmdDbGFzc2lmaWVylJOUKYGUfZQojApIc3Rp...')  
name: "Stacking_Model"  
created_time: 1679948091.414443  
accuracy: 92.71844660194175
```

Figure 6.13: Stacking Ensemble model Cross Validation Accuracy



# Chapter 7

## Iterations 3

### 7.1 Use Case Diagram

Figure 7.1 shows the use case diagram.

### 7.2 Activity Diagram

Figure 7.2 shows the use case diagram.

### 7.3 Component Diagram

Figure 7.3 shows the Components diagram.

### 7.4 Updated UI Screens

Screenshots of Updated UI is given in Figure 6.4, 6.5, 6.6, 6.7, 7.6. These are all subject to change.

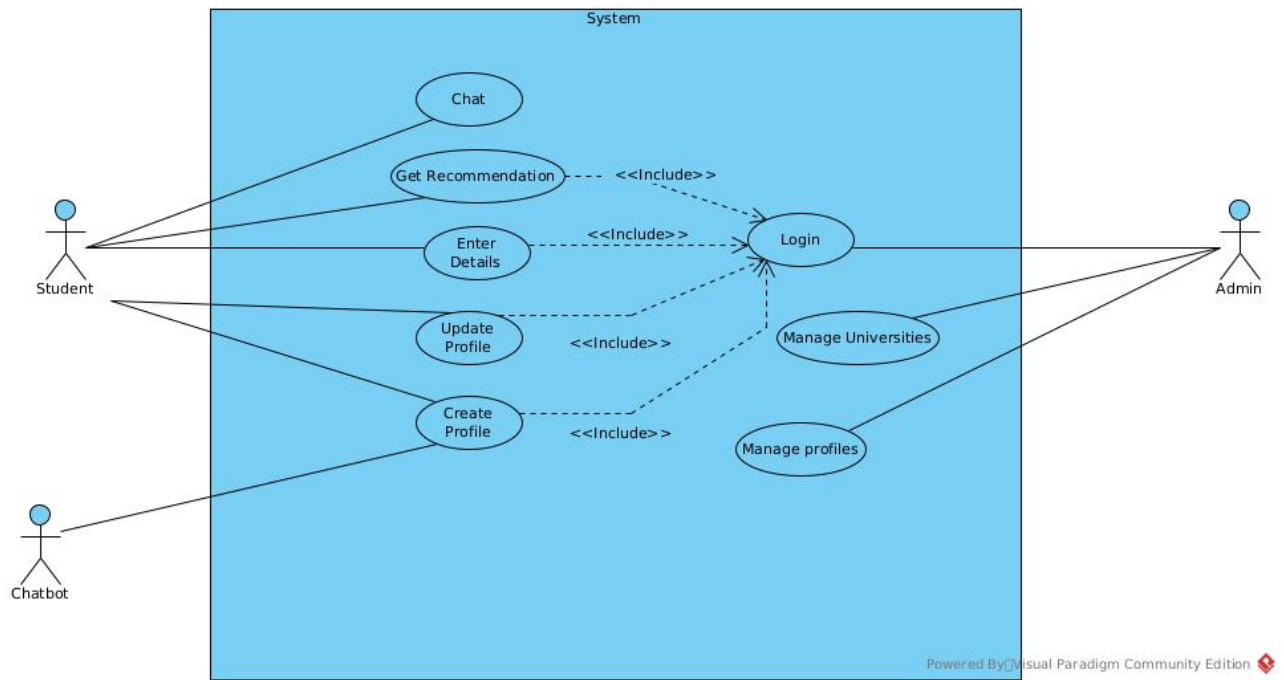


Figure 7.1: Use Case Diagram

## 7.5 Data Collection

In Iteration 2, we used dummy data set for our machine learning algorithms. Now, our main focus is on the data. There were 3 main sources of our data:

### 7.5.1 Google Forms

With Google Forms, you can create and analyze surveys right away. You get instant results as they come in. And, you can summarize survey results at a glance with charts and graphs. The questions that were asked in the google forms were directly reflected from Figure 6.1, but the answers that came were unstructured or raw. So, they had to be cleaned, records such as the university, where one entry is "NUST" and the other as "National University of Science and Technology" must be one. As of right now, we got almost 150 responses from google forms.

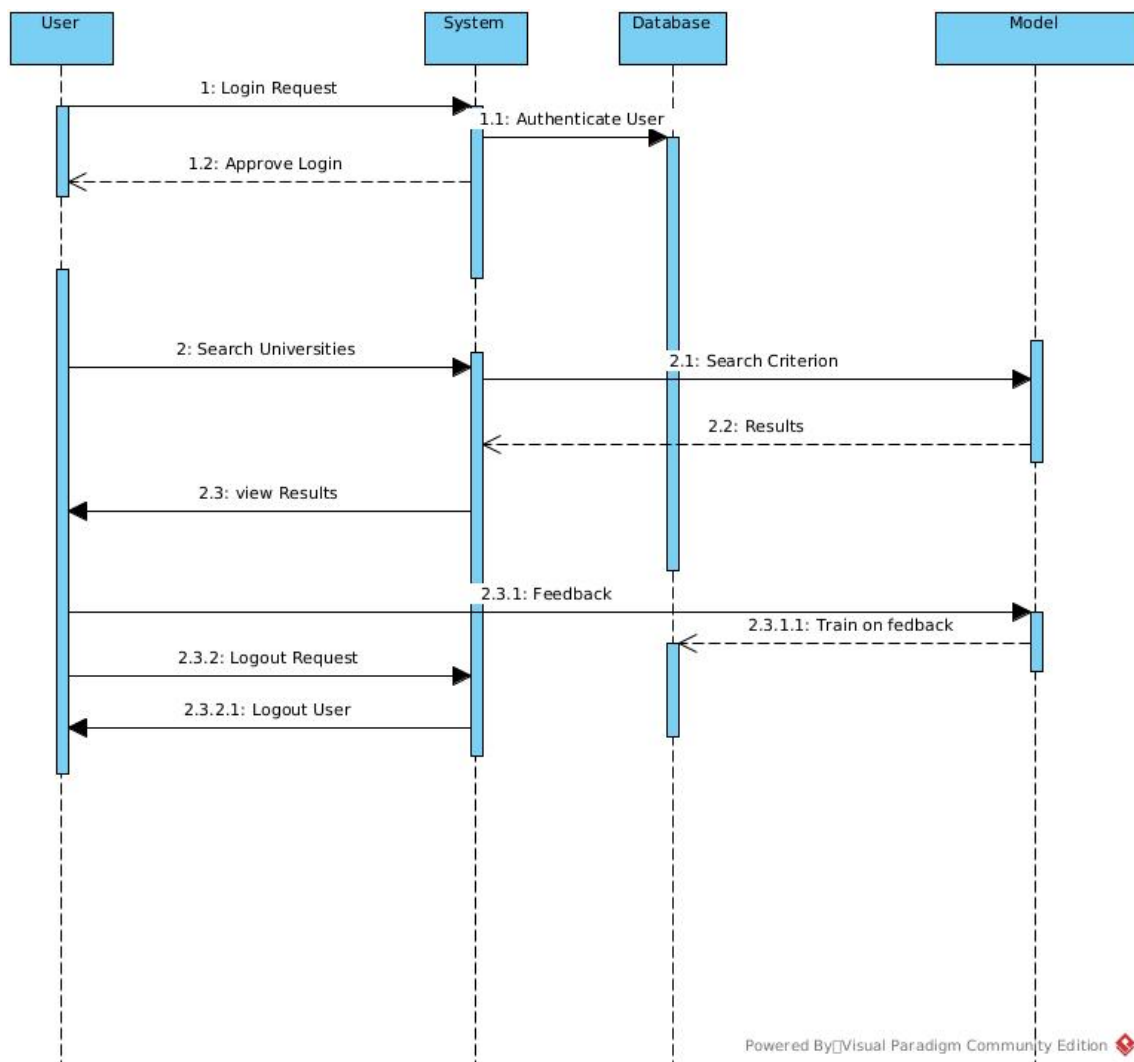


Figure 7.2: Activity Diagram



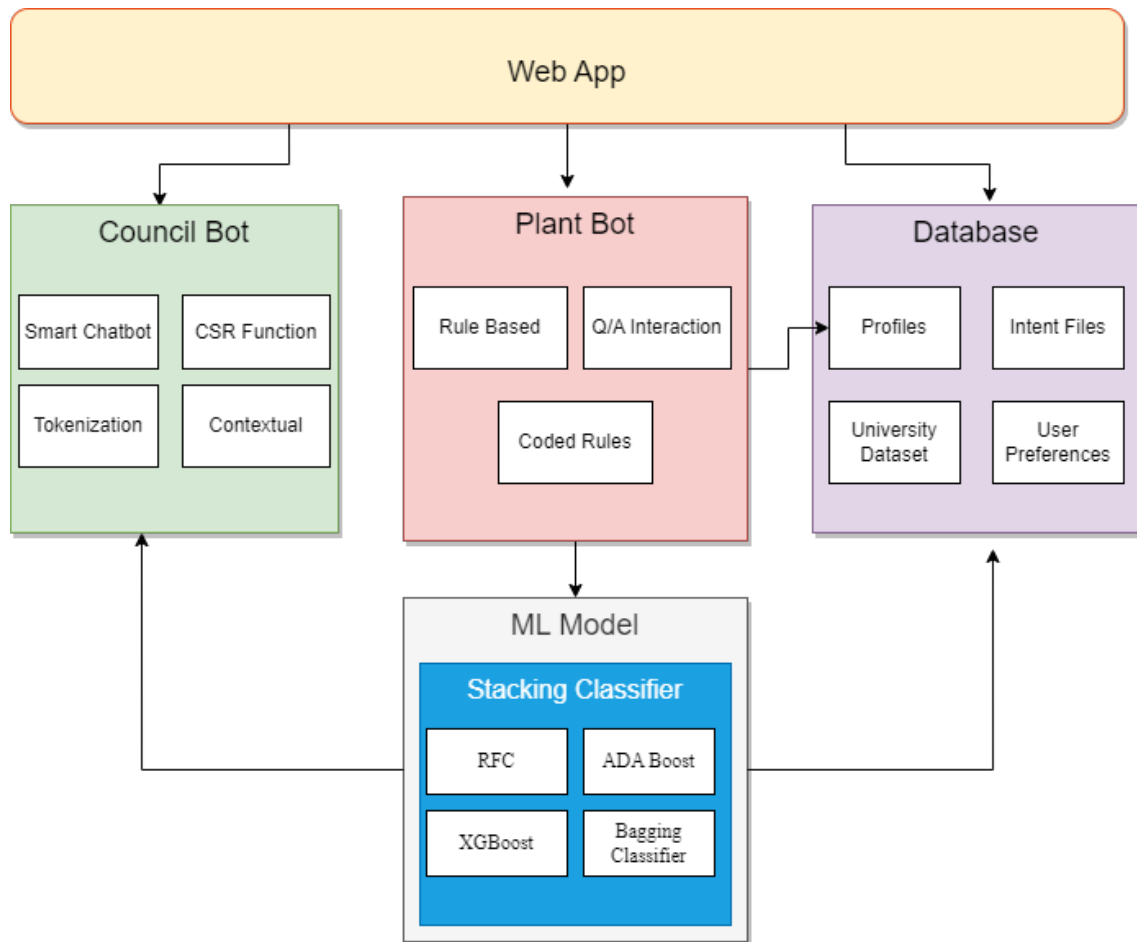


Figure 7.3: Component-Diagram

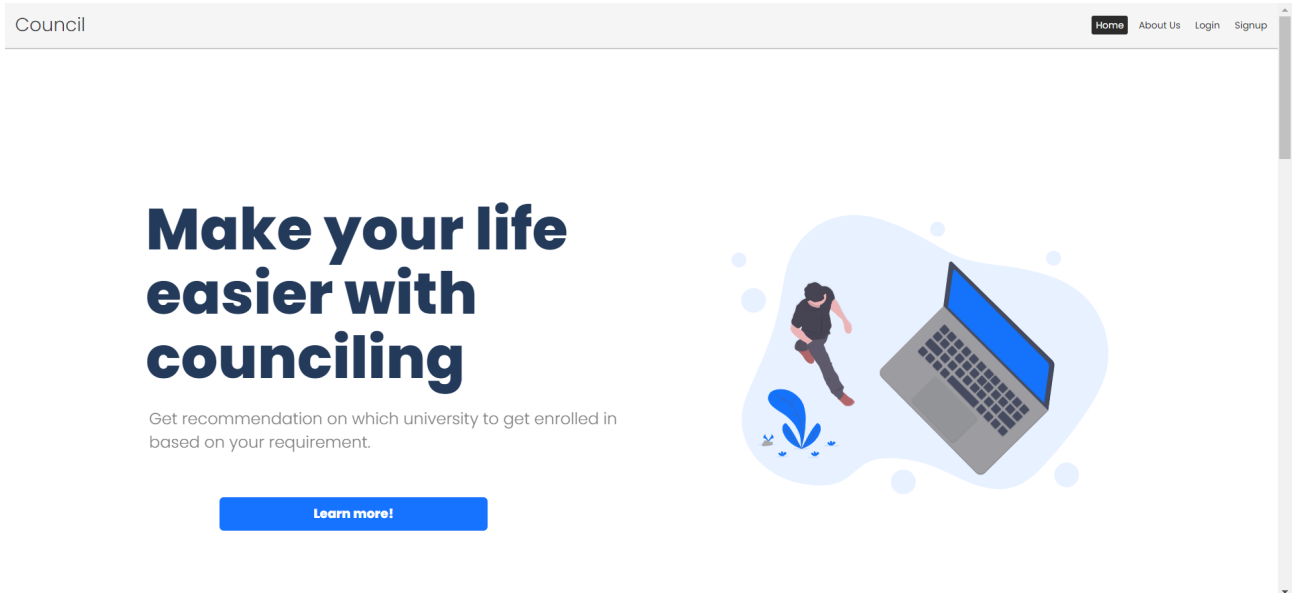


Figure 7.4: Home Page

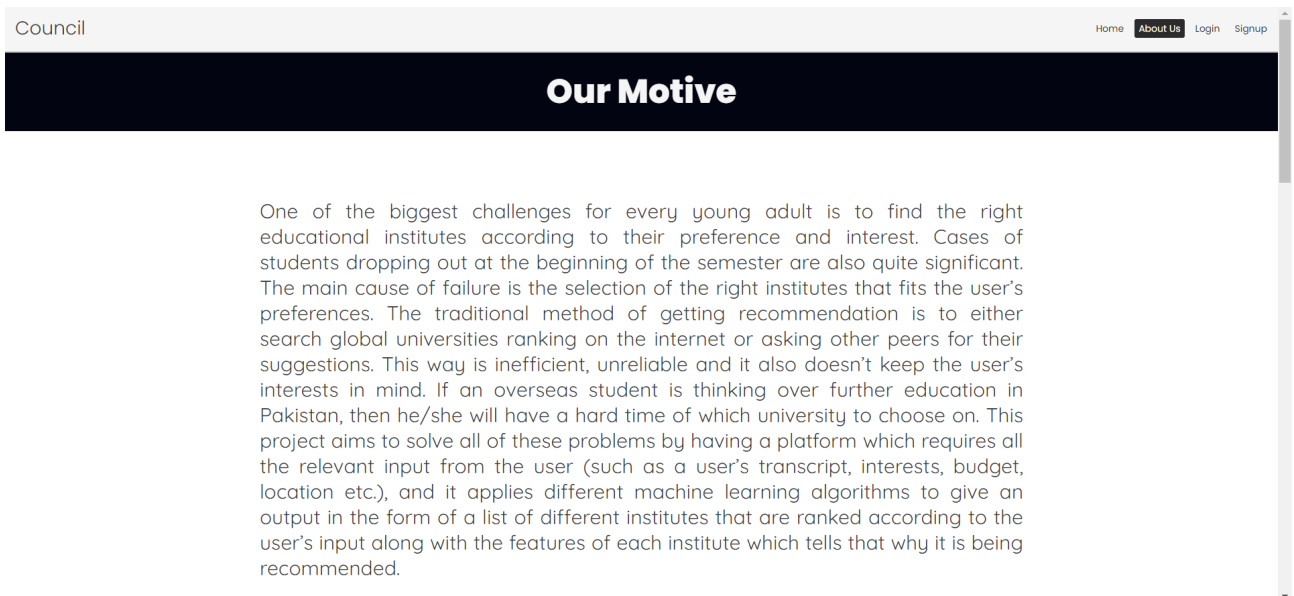


Figure 7.5: About Page

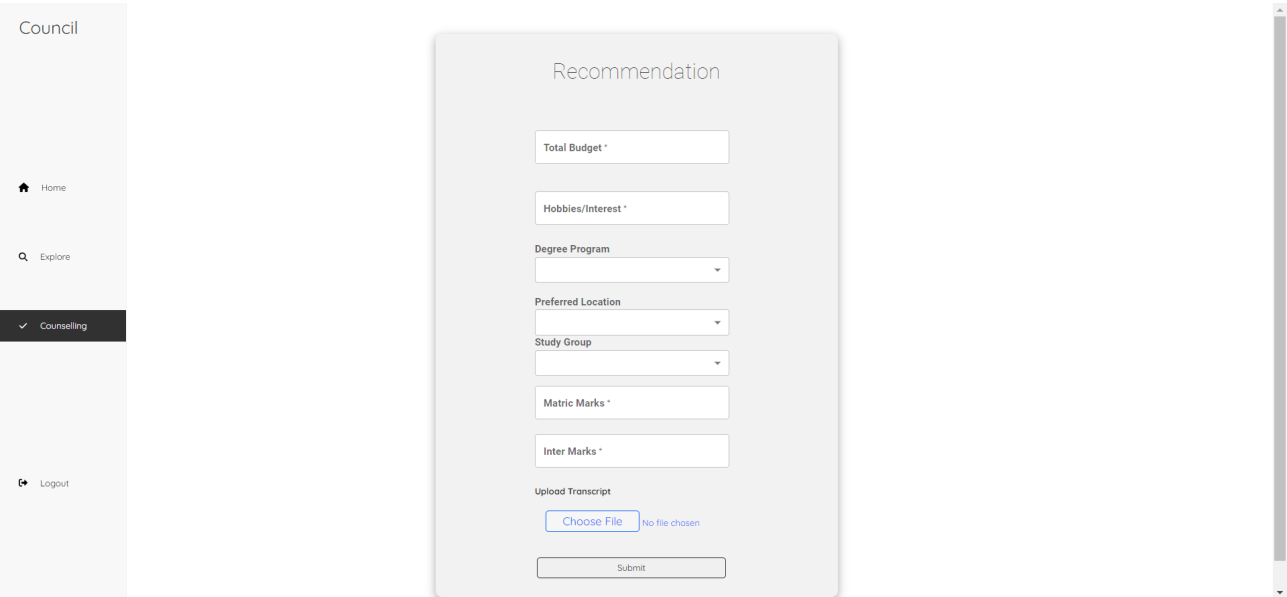


Figure 7.6: Council Tab

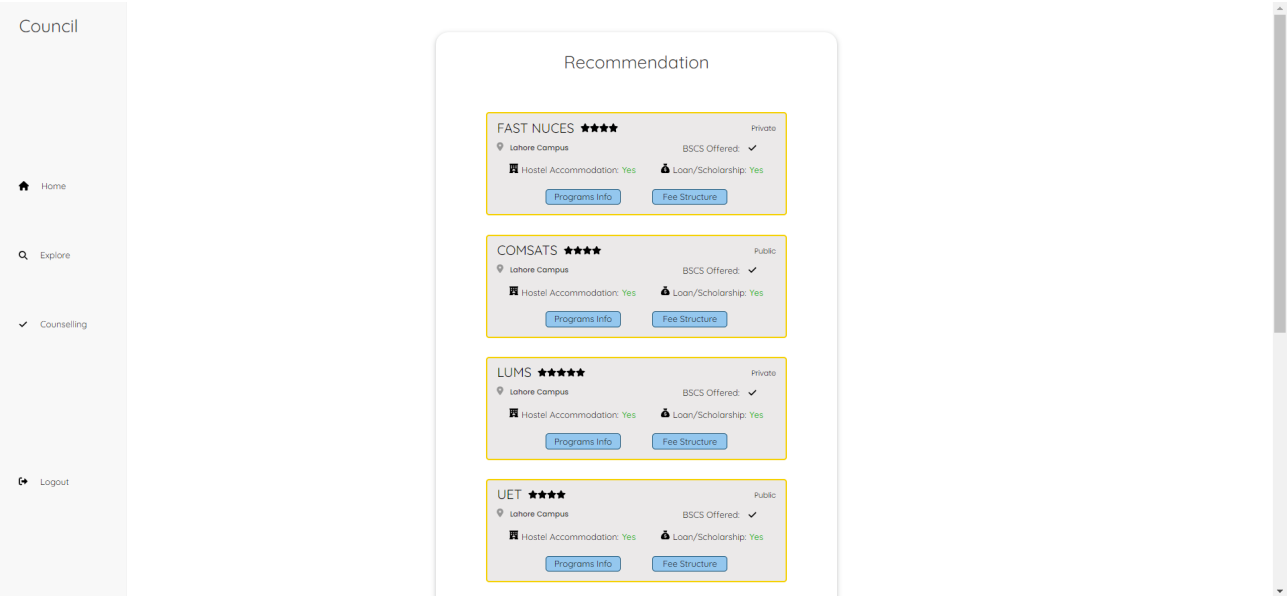


Figure 7.7: Recommendation Tab

### 7.5.2 Dummy Data Set

Since machine learning requires well over 2000+ records, we are using dummy data set along with the real data which we got from google form. Our main goal is to increase the amount of real data coming from the real users which we are currently limited with. Once our platform is at production level and is available for everyone, users will have the ability to select one of the university which they most preferred from the N list recommendation coming from council, which will be added directly to our training data.

## 7.6 Chatbot

This approach uses pytorch to create an interactive chatbot that will be used to interface with the majority of users.

### 7.6.1 Training data

Training data to recognize patterns uses bag of words that uses an array that splits all the words into all single words. There are two arrays being used, an “all array” that contains all the words and then for each pattern (found in the intents file) all words will be compared with this array. If the position is found then a 1 will be placed in the new array, otherwise a 0 will be placed. This is done for all the different patterns, this creates the X training data. The labels for all X training data are put into Y which is basically the weights assigned after running through the neural network. X and Y are used to train a feed forward network that will generate the most relevant response by calculating the probability of the patterns fed into it

### 7.6.2 Tokenization

Tokenization is a process in natural language processing (NLP) that involves breaking down a text into smaller units called tokens. These tokens can be words, phrases, or even individual characters, depending on the task at hand. The resulting tokens are used as

input for further NLP processing, such as text classification, sentiment analysis, or machine translation. There are different ways to tokenize text, but one of the most common methods is to split the text based on whitespace characters such as spaces, tabs, and line breaks. For example, consider the following sentence: "The quick brown fox jumps over the lazy dog." The whitespace-based tokenization of this sentence would produce the following tokens: ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog."] Note that the final token includes the period at the end of the sentence. This may or may not be desirable, depending on the application. Another common approach is to use more advanced tokenization techniques such as regular expressions, which can split the text based on patterns such as punctuation marks, hyphens, or apostrophes. This can be useful in cases where the text contains complex structures that cannot be easily tokenized using whitespace. Tokenization is an important preprocessing step in many NLP tasks, as it provides a structured representation of the text that can be easily processed by machine learning algorithms. It is also a fundamental building block for more advanced NLP techniques such as part-of-speech tagging, named entity recognition, and dependency parsing.

### **7.6.3 Stemming**

Stemming is a process in natural language processing (NLP) that involves reducing a word to its base or root form, which is often called a stem. This is done by removing any affixes, such as prefixes or suffixes, that are attached to the word. The goal of stemming is to reduce variations of words to their common base form, which can help improve the accuracy of text analysis and information retrieval systems. For example, consider the words "jumping", "jumps", and "jumped". These words all have the same base form, "jump", which represents the core meaning of the word. By applying a stemming algorithm, we can convert these variations to their base form, which can simplify subsequent processing steps. There are different stemming algorithms available, but one of the most popular is the Porter stemming algorithm. This algorithm applies a set of rules to remove common English suffixes and prefixes, such as "s", "ed", "ing", and "ly", among others. For example, applying the Porter algorithm to the word "jumps" would result in the stem "jump". Stemming can be useful in various NLP tasks, such as information retrieval, text

classification, and sentiment analysis. For example, in a search engine, stemming can help match queries with relevant documents that contain related words in their stemmed form. Similarly, in a sentiment analysis system, stemming can help identify the sentiment of a text by mapping related words to their common stem. However, stemming also has some limitations. For example, it may not always produce accurate results, especially when dealing with words that have multiple meanings or complex structures. Moreover, some variations of words may have different stems depending on the context, which can lead to errors in text analysis.

### 7.6.4 Feedforward Network

A feedforward neural network is a type of artificial neural network in which the information flows only in one direction, from the input layer to the output layer, without any feedback connections. The network consists of multiple layers of neurons, each layer processing the output of the previous layer to produce a final output. The input layer receives the input data and passes it on to the next layer, which performs a mathematical operation on the input data and generates a new set of values. This process is repeated for all subsequent layers until the output layer is reached, which produces the final output. The feedforward network can have multiple hidden layers, which are layers between the input and output layers that perform complex computations on the input data to extract relevant features. The number of hidden layers and the number of neurons in each layer can vary depending on the complexity of the problem being solved. The mathematical operation performed by each neuron in the feedforward network is typically a weighted sum of the inputs followed by a nonlinear activation function. Weights are learned during training using a technique called backpropagation, which involves adjusting the weights based on the difference between the predicted output and the actual output. Some advantages of feedforward networks include their ability to handle high-dimensional data, their ability to learn complex nonlinear mappings, and their ease of implementation and training. However, they may not be suitable for problems in which the input-output mapping is unknown, or the data has a high degree of variability or noise. In such cases, more advanced techniques such as recurrent neural networks or convolutional neural networks

may be more appropriate.

### **7.6.5 Workflow**

The chatbots workflow is generalized into a few different steps. The input from user is tokenized, it is put into lowercase and stemmed. All the pronunciation and other special characters are cleaned out, and the final array is put through the bag of words function. This generates the X vector for our training data.

## **7.7 Machine Learning Integration with Web Application**

Considering data set and the machine learning models are ready to use. How will we integrate this into our MERN stack application? Ideally we want the user to input their requirements into the web app and get their recommendation with the help of our ML models. To understand this, we need to look at server side script.

### **7.7.1 Server Side Scripts**

A typical backend file in Javascript must contains 4 things:

- Import of packages along with declaration of constant `app = express()`
- Middleware functions, that have access to the request object ( `req` ), the response object ( `res` ), and the next function in the application's request-response cycle.
- Routing, which are invoked by `app.use('routing folder path')`. In routing folder, multiple routes are defined along with the method (POST, GET, DELETE, PUT etc.) which have access to the controller functions.
- Connection to the database along with the `app.listen(PORT)` statement which runs the whole server only when the connection to the database is established and everything else before had a positive response (status code 200)

Now, in our case, we need to take input from the user and somehow invoke the python script from our nodejs file. To do that, we first need the user input which we can get by the `app.use(express.json())` middleware. Destructing **req.user** property will give us access to the user's input that they have typed in the frontend of the application. Only thing left now is to somehow spawn the python script which coincidentally can be done through **const child-process = require("child-process")** package, and then destructing **spawn** from it. The `spawn()` function creates a child process that inherits specific attributes from the parent. Here the parent function is the nodejs file, and the child is the python script that contains the machine learning algorithms. We will take input from the frontend (by destructing `req.user`) and pass them as parameters to the child process (in this case the `recommendation.py` file). Figure 7.8 shows a part of server side code which explains this. Exceptions are also defined in case if any error is occurred in the python script such that it gives the output on the browser instead of being stuck at the loading screen.

## 7.8 Initial Recommendation

The python script (in this case `recommendation.py` file) first takes all the input of the user by using `sys.argv[]` property. Using this input, the data set of the enrolled users (Figure 6.1) is filtered, such that the university that the corresponding user got enrolled in matches the criteria by the input given. For example, the data set is filtered by location of Peshawar only and where the degree program of BBA is being offered. The resulting data set will contain only those students who are enrolled in university where BBA is being offered and are located in Peshawar only. This data set will be concatenated with the user's data and will go through pre-processing stage where renaming and reordering of columns are done. Numeric data is categorized and is finally encoded. ADABOOST, XGBOOST, BaggingClassifier and RFC are trained under Stacking Model. The user's data or the last row is dropped from the data set and corresponding class attribute is then predicted by using `stackingModel.predict(user's data)` method. The resulting value is then decoded using `le.inverse-transform(value)` property, which will give us the name of the university. This recommended university has gone through both content based filtering (Filtered list



```
const test = async () => {
  let data1;
  const pythonOne = spawn("python", [
    "recommendation_v5.py",
    CNIC,
    budget,
    interest,
    degree_program,
    preferred_location,
    study_group,
    matric_marks,
    inter_marks,
    gender,
    Preferred_language,
    Race_ethnicity,
    HomeCity,
  ]);
  pythonOne.stdout.on("data", function (data) {
    data1 = data.toString();
  });

  pythonOne.on("close", (code) => {
    if (data1 === undefined || data1 === null || data1 === 0) {
      data1 = [];
      res.status(200).json(data1);
    } else res.status(200).json(data1);
  });
};
test();
});
```

Figure 7.8: Spawning python script from nodejs file

of university which matches the criteria/condition of the user) and collaborative filtering (Recommended university is given where the user's preferences are matching/similar the most.)

## **7.9 N-List Recommendation**

After initial recommendation is given. We will use university data set (Figure 4.2) to find the cosine similarity of the initial recommendation which gives us a list of multiple university most similar to least similar.



# Bibliography

- [1] Shubham Kumar Agrawal. Recommendation system -understanding the basic concepts. 2021.
- [2] Jose Konstan John Riedl Badrul Sarwar, George Karypis. Item-based collaborative filtering recommendation algorithms. 2001.
- [3] Robin Burke. Hybrid web recommender systems. 2007.
- [4] A. Hernando A. Gutiérrez J. Bobadilla, F. Ortega. Recommender systems survey. 2013.
- [5] Kerem Kargin. Nlp: Tokenization, stemming, lemmatization and part of speech tagging. 2021.
- [6] Norafida Ithnin Mehrbakhsh Nilashi, Othman Bin Ibrahim. Multi-criteria collaborative filtering with high accuracy using higher order singular value decomposition and neuro-fuzzy system. 2014.
- [7] George Karypis Mukund Deshpande. Item-based top-n recommendation algorithms. 2004.
- [8] H.R. Varian P. Resnick. Recommender systems | communications of the acm. 1997.
- [9] Mitesh Suchak Peter Bergstrom John Riedl Paul Resnick, Neophytos Iacovou. GroupLens: An open architecture for collaborative filtering of netnews. 1994.
- [10] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. 2007.

- [11] Aditya Suresh Kumar Swetha Krishnakumar Ramkishore Swaminathan, Joe Manley. University recommender system for graduate studies in usa.
- [12] Nada Alzhrani Shaima Alghamdi and Haneen Algethami. Fuzzy-based recommendation system for university major selection. 2019.
- [13] Qusai Shambour. A hybrid trust-enhanced collaborative filtering recommendation approach for personalized government-to-business e-services. 2011.
- [14] Vatsal. Explaining and implementing content based, collaborative filtering and hybrid recommendation systems in python. 2021.