# UNIVERSITY ADMIT ELIGIBLITY PREDICTOR

## NALAIYA THIRAN PROJECT BASED LEARNING ON PROFESSIONAL READLINESS FOR INNOVATION, EMPLOYMENT AND ENTERPRENEURSHIP

**A PROJECT REPORT**

**Submitted By**

Kishor S [2019506043]
Raghavasimhan T V [2019506068]
Tejeshwini R S [2019506103]
Srivatsan K P [2019506097]
Abhinash S [2019506004]

**INDEX:**

1. **INTRODUCTION**

a. Project Overview

It is challenging for candidates to narrow down their potential universities for study in the current global environment. There are a number of academic prerequisites to be admitted to these universities. Students frequently wait until the very last minute to decide whether or not their applications will be accepted because there isn't any actual documentation that outlines the requirements due to the overwhelming number of universities at various levels.

b. Purpose

It assists students in picking the appropriate college. In which students can register with their personal and academic information to forecast college acceptance, and administrators can assign seats to students. It assists students in picking the appropriate college. In which students can register with their personal and academic information to forecast college acceptance, and administrators can assign seats to students.

2. **LITERATURE SURVEY**

a. Existing problem

i. No clear standards defined for each university
ii. Vague requirements
iii. Student dont get a single point of search and identification
iv. Poor dataset

b. References

Waters et al., establishes a model which provides accurate prediction with minimal hardware and software dependencies.[1] Çano et al., suggested a cutting-edge plan for a hybrid recommender system for college admission. It entails the cooperation of two cascading hybrid recommenders with the aid of a college predictor.[2]

M S Acharya et al., compared various models and found that linear regression had the edge owing to the linear dependency among the features in the data.[3] They did not, however, consider correlations among features such as GPA, GRE scores etc., which are not negligible.

Aljasmi et al., trained a multilayer perceptron that surpasses prior work that used multiple linear regression classifiers, random forests and kNN classifiers by showing significant reduction in mean absolute error. [4]

Omaer et al., proposed a graduate admission prediction system using Deep Neural Network which has performed more accurately and efficiently compared to the existing methods but performed little preprocessing.[5] Sridhar et al., used a stacked ensemble learning model with MLPs to achieve a very high accuracy but fails to consider subjective factors. [6]

Sivasangari et al., provided the analysis of scores versus chance of prediction based on historical data so that students can understand whether their profile is suitable or not. This model uses linear regression and random forest algorithms but cat boost algorithm produced highest accuracy of 95 percent.[7]

Neda and Gago-Masague developed various models in an attempt to establish feasibility of integrating ML even from the side of the university, in order to evaluate applications. However, their models were trained and tuned for a specific university (University of California at Irvine) and failed to perform well on subjective factors.[8]

Joshi et al., built decision tree regressor and random forest regressor which was compared with linear regression. Sequentially it was proved that linear regression produced highest accuracy of 82 percent among them.[9]


1. A. Waters and R. Miikkulainen, “GRADE: Machine Learning Support for Graduate Admissions”, AIMag, vol. 35, no. 1, p. 64, Mar. 2014.
2. E. Çano and M. Morisio, “Hybrid recommender systems: A systematic literature review,” Intelligent Data Analysis, vol. 21, no. 6, pp. 1487–1524, Nov. 2017, doi: 10.3233/ida-163209.
3. M. S. Acharya, A. Armaan and A. S. Antony, "A Comparison of Regression Models for Prediction of Graduate Admissions," 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), 2019, pp. 1-5, doi: 10.1109/ICCIDS.2019.8862140.
4. Aljasmi, Sara & Nassif, Ali & Shahin, Ismail & Elnagar, Ashraf. (2020). Graduate Admission Prediction Using Machine Learning. 14. 10.46300/91013.2020.14.13.
5. M. Omaer Faruq Goni, A. Matin, T. Hasan, M. Abu Ismail Siddique, O. Jyoti and F. M. Sifnatul Hasnain, "Graduate Admission Chance Prediction Using Deep Neural Network," 2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE), 2020, pp.259-262,doi: 10.1109/WIECON-

ECE52138.2020.9397988.

6. S. Sridhar, S. Mootha and S. Kolagati, "A University Admission Prediction System using Stacked Ensemble Learning," 2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA), 2020, pp. 162-167, doi: 10.1109/ACCTHPA49271.2020.9213205.

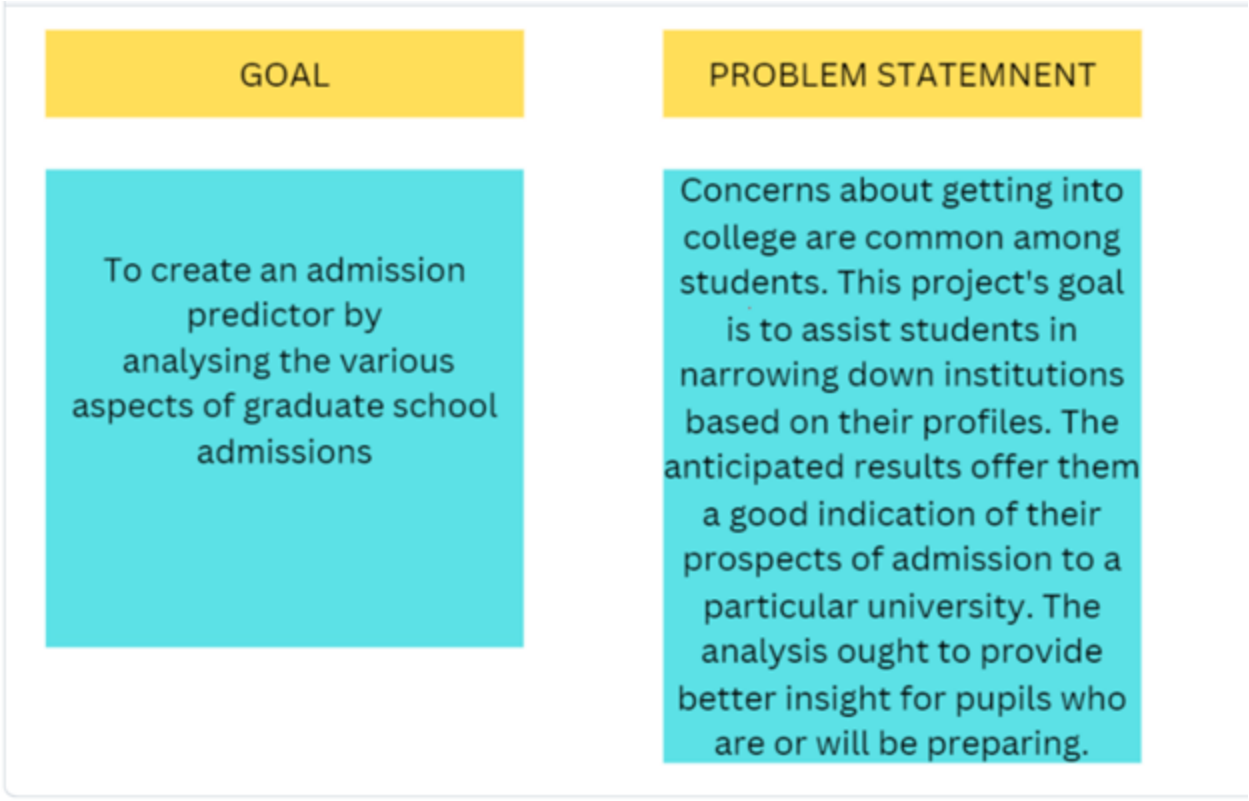
7. A. Sivasangari, V. Shivani, Y. Bindhu, D. Deepa and R. Vignesh, "Prediction Probability of Getting an Admission into a University using Machine Learning," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 1706-1709, doi: 10.1109/ICCMC51019.2021.9418279.

8. B. M. Neda and S. Gago-Masague, "Feasibility of Machine Learning Support for Holistic Review of Undergraduate Applications," 2022 International Conference on Applied Artificial Intelligence (ICAPAI), 2022, pp. 1-6, doi: 10.1109/ICAPAI55158.2022.9801571.

9. Joshi Padma. N, P. Chandana, G. Kavya, J. Sreekar, Y. Sowmith Reddy, “University Admission Prediction using Machine Learning”, YMER - Volume 21: Issue 5 (May) -2022. [Online]. Available: http://ymerdigital.com/uploads/YMER2105F1.pdf

10. N. Gupta, A. Sawhney and D. Roth, "Will I Get in? Modelling the Graduate Admission Process for American Universities," 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), 2016, pp. 631-638, doi:10.1109/ICDMW.2016.0095.

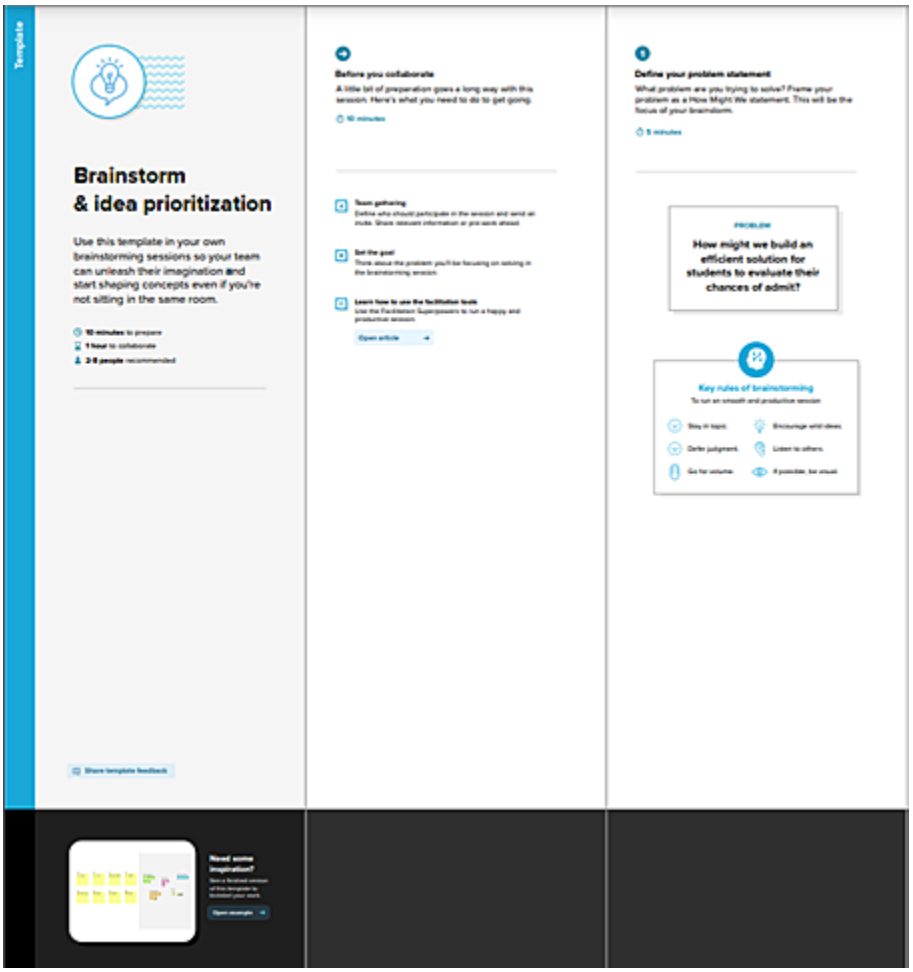
c. Problem Statement Definition

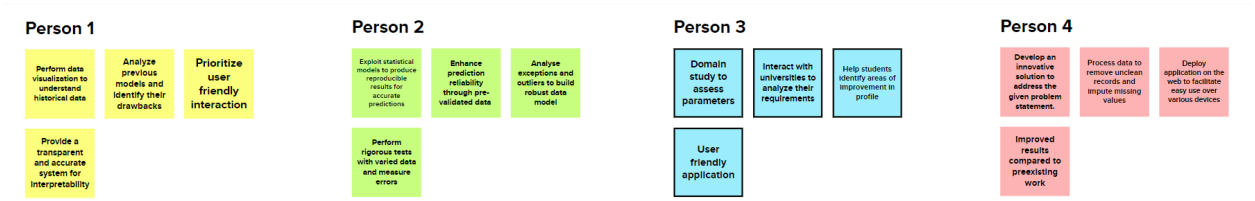


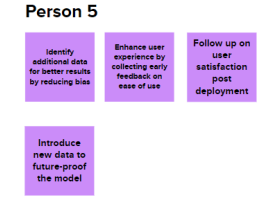
3. **IDEATION & PROPOSED SOLUTION**

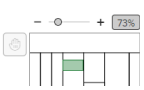
a. Empathy Map Canvas

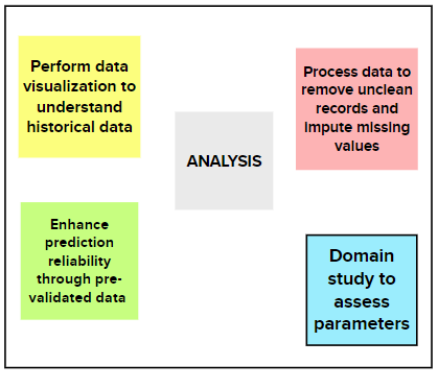
b. Ideation & Brainstorming

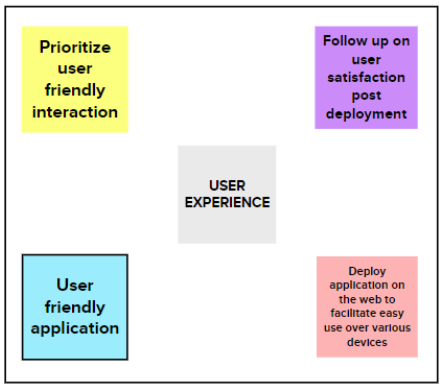


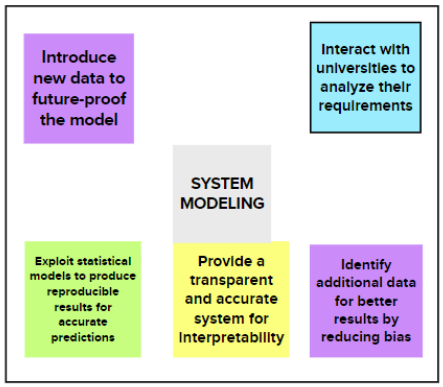



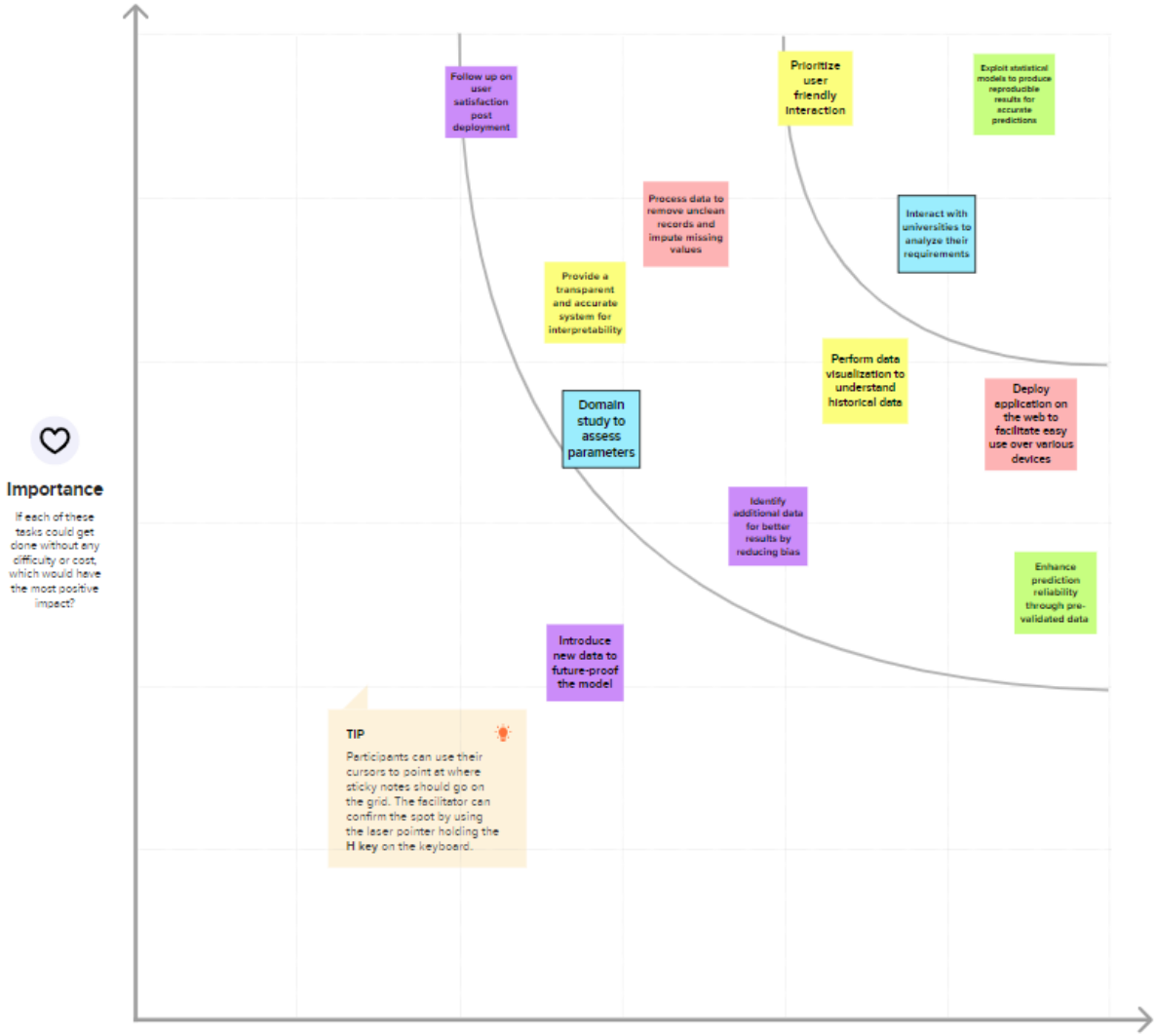






browse, organize, and categorize important ideas as themes within your mural.

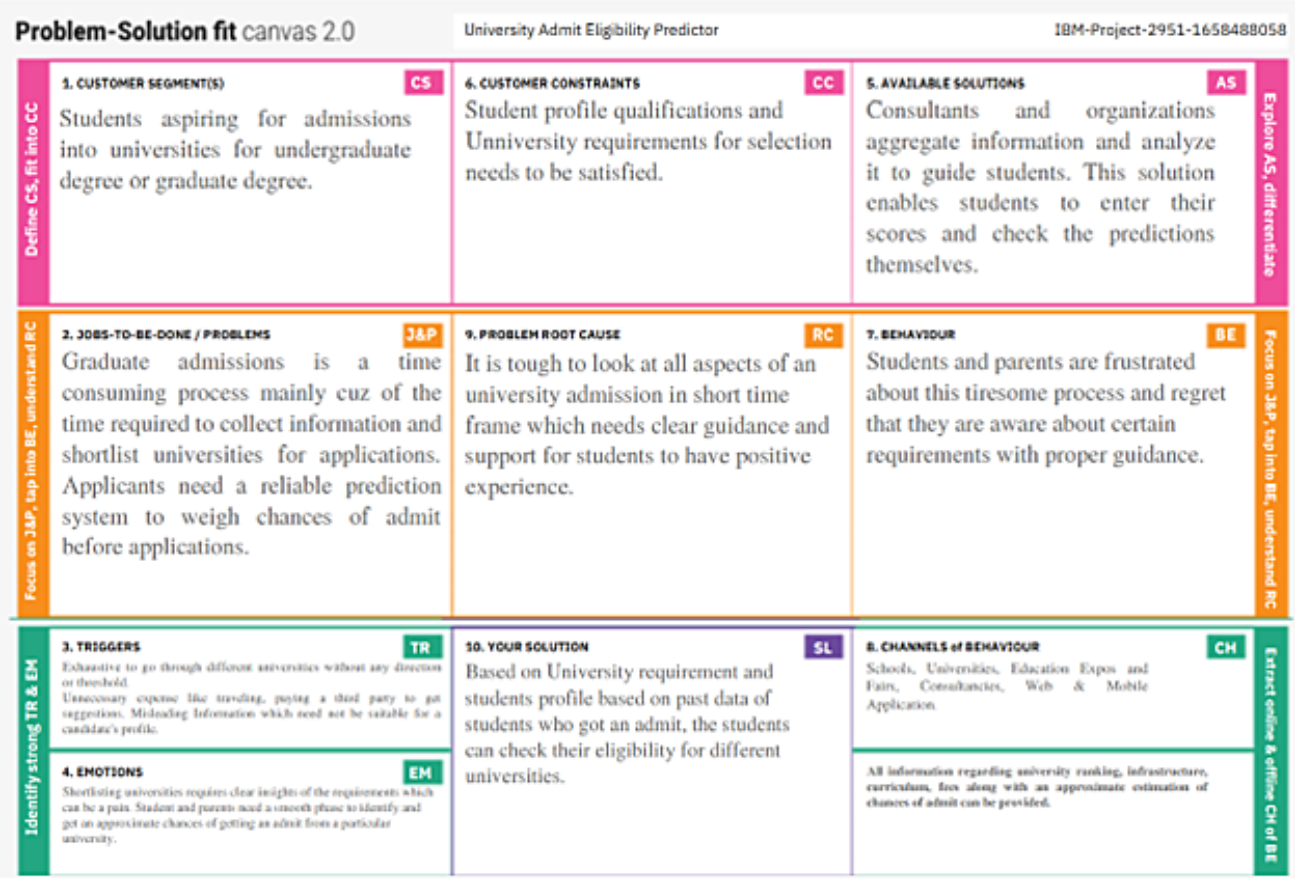



Follow up on user satisfaction post deployment
Prioritize user friendly interaction
Exploit statistical models to produce reproducible results for accurate predictions
Process data to remove unclean records and impute missing values
Interact with universities to analyze their requirements
Provide a transparent and accurate system for interpretability
Perform data visualization to understand historical data
Domain study to assess parameters
Deploy application on the web to facilitate easy use over various devices
Identify additional data for better results by reducing bias
Enhance prediction reliability through pre-validated data
Introduce new data to future-proof the model
TIP
Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

c. Proposed Solution

| S.No. | Parameter | Description |
|---|---|---|
| 1. | Problem Statement | Graduate admissions prove to be periods of extreme stress and anxiety for applicants around the world. At present, there is a demand for a reliable system to give an admit prediction. In this project, we plan to model a system that provides a reliable prediction for the students to get admitted to a university. |
| 2. | Idea/Solution | Analyze historical data to identify key patterns in an individual's profile to evaluate the chances of getting admission to a university. Interact with students' and universities' opinions to understand requirements that are expected. |
| 3. | Novelty | The model improves upon the existing work to produce the highest accuracy for the prediction of the chance to get into a particular university, given its rank and the profile of a student. |
| 4. | Social Impact/Customer Satisfaction | It provides a rough evaluation for the students to know in which category of universities they are likely to get admitted. |
| 5. | Business Model | Students and Consultancies can use this system as it predicts a user's chances of admission to the universities of their choice. The universities can use this solution to list their requirements and benchmarks. |
| 6. | Scalability | Currently, the model is designed to predict the universities where the student will be eligible to get admission, the model can be extended to predict which courses will be available for the students based on their educational ranks. |

d. Problem Solution fit



4. **REQUIREMENT ANALYSIS**

a. Functional requirement

Following are the functional requirements of the proposed solution.

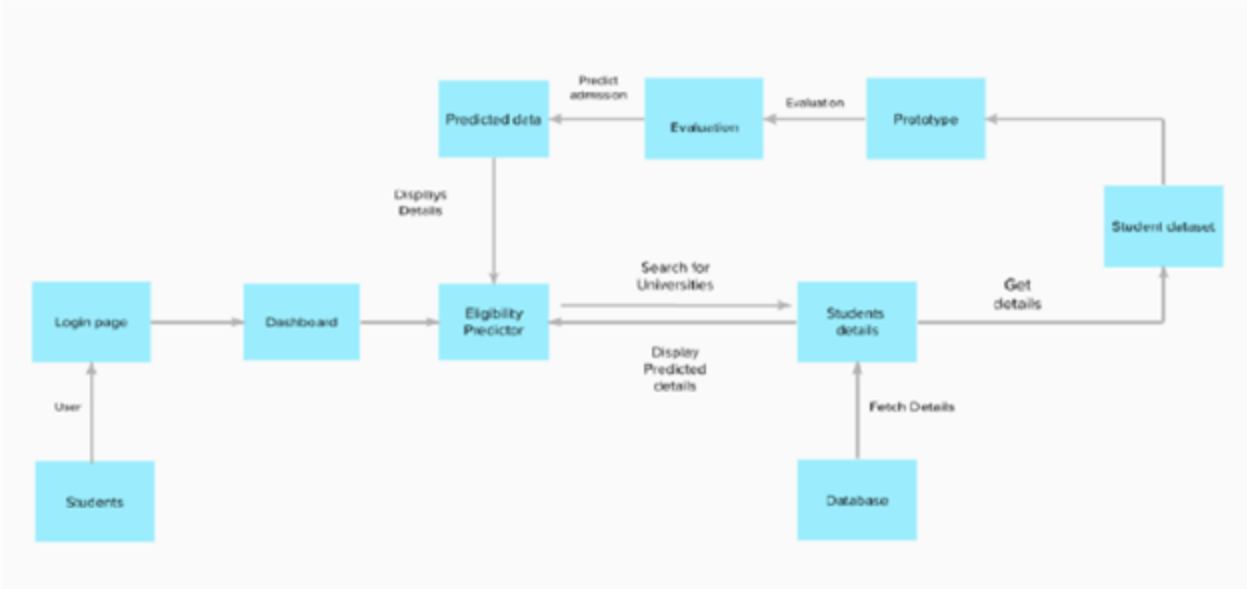
| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR - 1 | User Registration | Registration through Form<br>Registration through LinkedIN |
| FR - 2 | User Credentials | Store it in the database for verification |
| FR - 3 | User Details | Fetch details such as GRE score, LOR score, SOP score, etc from the user |
| FR - 4 | User Choice | Dropdown list for selecting university |
| FR - 5 | Suggestions | Ranking top 10 universities based on user details |

b. Non-Functional requirements

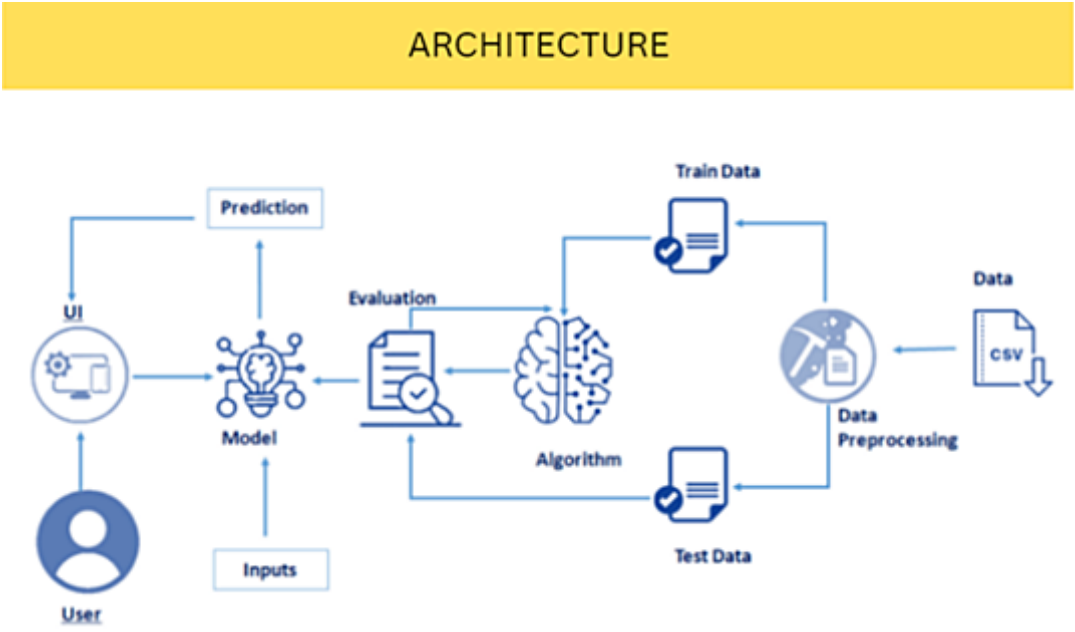
| NFR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | Usability | Any score must be accepted for prediction |
| NFR-2 | Performance | The performance and interface must be user friendly |
| NFR-3 | Availability | Anyone must be able to register and login |
| NFR-4 | Scalability | It must be able to handle increase in the number of users |

5. **PROJECT DESIGN**

a. Data Flow Diagrams




b. Solution & Technical Architecture



c. User Stories

| User Type | Functional Requirement | User Story Number | User Story Task | Acceptance criteria | Priority | Release |
|---|---|---|---|---|---|---|
| Customer (Web user) | Registration | USN-1 | As a user, can register for the application by entering my email password, and confirming my password. | I can access my account / dashboard | High | Sprint-I |
| | | USN-2 | As a user, I will receive confirmation email | I can receive confirmation email & click confirm | High | Sprint-I |
| | | USN-3 | As a user, can register for the app authentication through form | I can register & access the dashboard with form | Low | Sprint-2 |
| | | USN-4 | As a user, I can register tor the application through G mail | I can register and access the dashboard | Medium | Sprint-I |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | USN-5 | As a user I can log into the application by enterin email sword | I can access various | High | Sprint-I |
| | Dashboard | USN -6 | As a user, I can search through various universities | I can access several universities | High | Sprint-I |
| | Search | USN-7 | As a user , I can search for Universities with different field | I can receive information related to universities on various locations | High | Sprint-2 |
| | | USN-9 | As a user , I can view the University details | I will get the information on seat availability, el ibili criteria. | High | Sprint-2 |
| | Receive notification | USN-9 | As a user, I will receive notifications about the Suggested universities based on student marks | I will get frequent updates of the preferred universities | Low | Sprint-2 |
| | Chat with expert | USN-10 | As user, I can chat with the expert for Clarifications | I can clear my doubts th rough Chat with expert o ion | Medi um | Sprint-2 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Admin | Analysis | USN-11 | As an admin. I will analyse the given dataset | I can analyse the dataset | High | Sprint-2 |
| | Predict | USN-12 | AS an admin, I Will predict the admission | can predict eligibility for admission | High | Sprint-2 |

## 6. PROJECT PLANNING & SCHEDULING

### a. Sprint Planning & Estimation

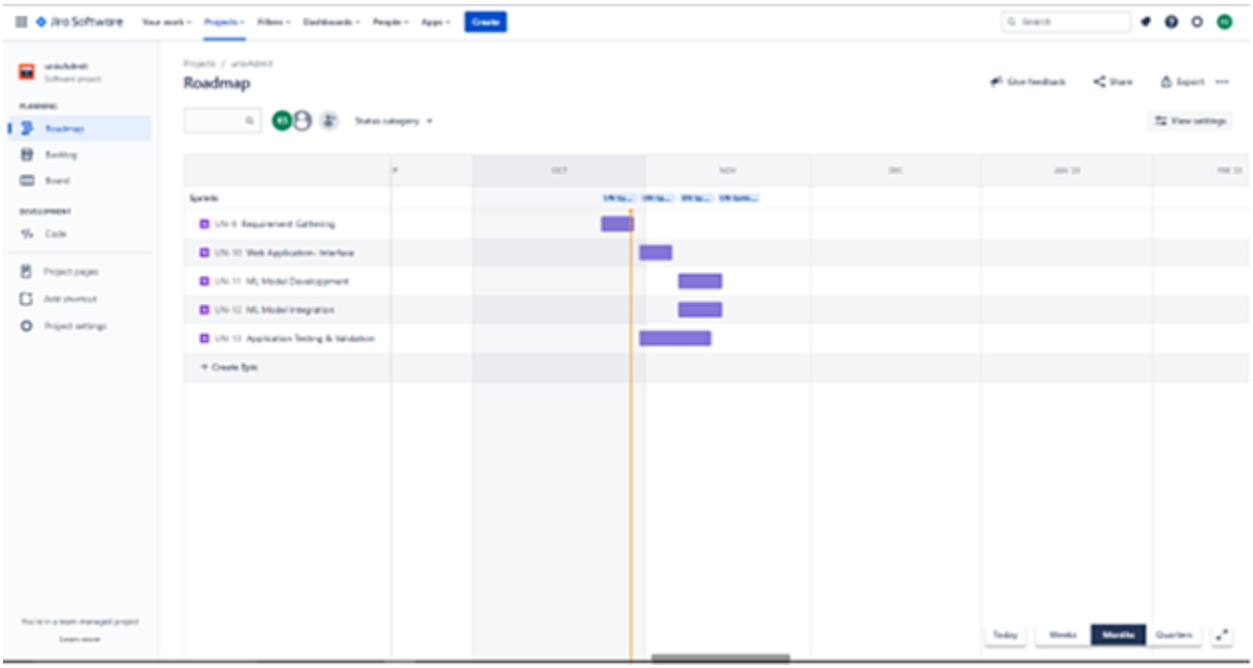
| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-1 | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 2 | High | Kishor S, Srivatsan K P |
| Sprint-1 | Registration | USN-2 | As a user, I will receive confirmation email once I have registered for the application | 1 | High | Tejeshwini R S |
| Sprint-2 | Login | USN-5 | As a user, I can log into the application by entering email & password | 1 | High | Raghavasimhan T V, Abhinash S |
| Sprint-2 | Dashboard | USN-6 | As a user, I can view and edit my details and access the services through the dashboard over the browser. | 1 | High | Srivatsan K P, Abhinash S |
| Sprint-3 | Model Building | USN-7 | The data that is available is modeled using machine learning techniques. | 2 | High | Kishor S, Raghavasimhan T V |
| Sprint-3 | Model Testing | USN-8 | The model is evaluated with various performance metrics and fine-tuned. | 2 | High | Tejeshwini R S |
| Sprint-4 | Integration | USN-9 | Integrate the frontend and the developed ML model using flask and deploy on cloud. | 2 | High | Kishor S, Raghavasimhan T V |
| Sprint-4 | System Testing & Validation | USN-10 | A thorough system testing is conducted and the process are validated, along with use case testing | 2 | Medium | Abhinash S |

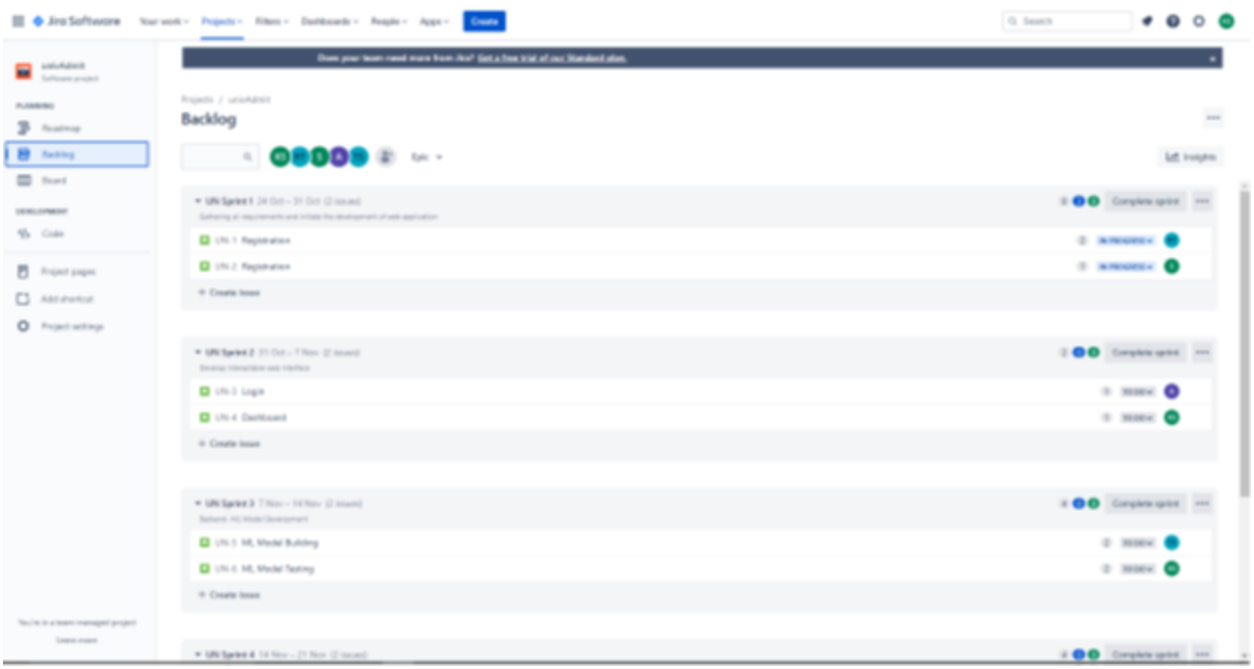
### b. Sprint Delivery Schedule

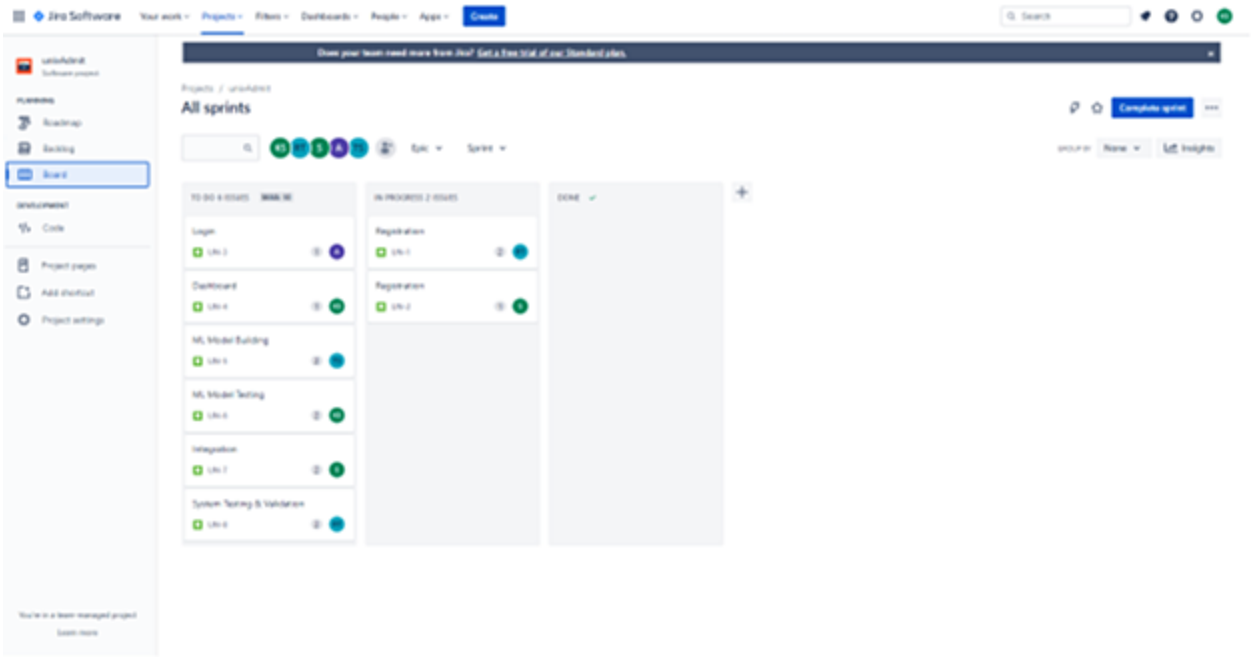
| Sprint | Total Story Points | Duration | Sprint Start Date | Sprint End Date (Planned) | Story Points Completed (as on Planned End Date) | Sprint Release Date (Actual) |
|---|---|---|---|---|---|---|
| Sprint1 | 20 | 6 Days | 24 Oct 2022 | 29 Oct 2022 | 20 | 29 Oct 2022 |
| Sprint2 | 20 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | 20 | 05 Nov 2022 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Sprint3 | 20 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 20 | 12 Nov 2022 |
| Sprint4 | 20 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | 20 | 19 Nov 2022 |

c. Reports from JIRA




Jira Software
Backlog
Complete sprint


Jira Software
All sprints
Complete sprint

## 7. **CODING & SOLUTIONING**

### a. Model Prediction

In [49]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os, types
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='WM1y1zxWyebRX9Y1BgAgC7oAr9JB3FL8eyBDaIfdhVo4',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'univadmit-donotdelete-pr-gimwwaicqw2pki'
object_key = 'Admission_Predict.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

admission = pd.read_csv(body)
admission.head()
```

Out[49]:

| | Serial No. | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Research | Chance of Admit |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 0.92 |
| 1 | 2 | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 0.76 |
| 2 | 3 | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 0.72 |
| 3 | 4 | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | 1 | 0.80 |
| 4 | 5 | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | 0 | 0.65 |

In [50]:
```
admission.head()
```

Out[50]:

| | Serial No. | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Research | Chance of Admit |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 0.92 |
| **1** | 2 | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 0.76 |
| **2** | 3 | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 0.72 |
| **3** | 4 | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | 1 | 0.80 |
| **4** | 5 | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | 0 | 0.65 |

In [51]:
```
admission.shape
```

Out[51]:
```
(400, 9)
```

In [52]:
```
admission.columns
```

Out[52]:
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

In [53]:
```
admission.describe()
```

Out[53]:

| | Serial No. | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Research | Chance of Admit |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 400.000000 | 400.000000 | 400.000000 | 400.000000 | 400.000000 | 400.000000 | 400.000000 | 400.000000 | 400.000000 |
| **mean** | 200.500000 | 316.807500 | 107.410000 | 3.087500 | 3.400000 | 3.452500 | 8.598925 | 0.547500 | 0.724350 |
| **std** | 115.614301 | 11.473646 | 6.069514 | 1.143728 | 1.006869 | 0.898478 | 0.596317 | 0.498362 | 0.142609 |
| **min** | 1.000000 | 290.000000 | 92.000000 | 1.000000 | 1.000000 | 1.000000 | 6.800000 | 0.000000 | 0.340000 |
| **25%** | 100.750000 | 308.000000 | 103.000000 | 2.000000 | 2.500000 | 3.000000 | 8.170000 | 0.000000 | 0.640000 |
| **50%** | 200.500000 | 317.000000 | 107.000000 | 3.000000 | 3.500000 | 3.500000 | 8.610000 | 1.000000 | 0.730000 |
| **75%** | 300.250000 | 325.000000 | 112.000000 | 4.000000 | 4.000000 | 4.000000 | 9.062500 | 1.000000 | 0.830000 |
| **max** | 400.000000 | 340.000000 | 120.000000 | 5.000000 | 5.000000 | 5.000000 | 9.920000 | 1.000000 | 0.970000 |

In [54]:
```
admission.isnull().sum()
```

Out[54]:
```
Serial No.          0
GRE Score           0
TOEFL Score         0
University Rating   0
SOP                 0
LOR                 0
CGPA                0
Research            0
Chance of Admit     0
dtype: int64
```

In [55]:
```
X=admission.drop(['Serial No.','Chance of Admit '],axis=1)
X.shape
```

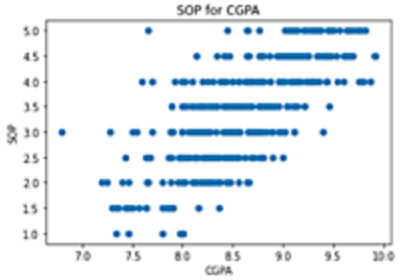
Out[55]:
```
(400, 7)
```

In [56]:
```
y=admission['Chance of Admit ']
y.shape
```

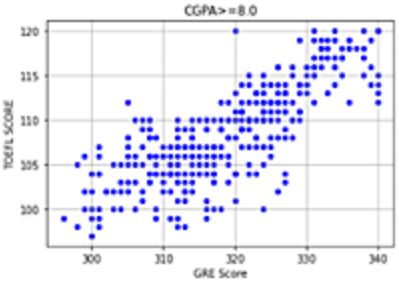
Out[56]:
```
(400,)
```

In [57]:
```
plt.scatter(admission['GRE Score'],admission["CGPA"])
plt.title('CGPA vs GRE Score')
plt.xlabel('GRE Score')
plt.ylabel('CGPA')
plt.show()
```


In [58]:
```
plt.scatter(admission['CGPA'],admission['SOP'])
plt.title('SOP for CGPA')
plt.xlabel('CGPA')
plt.ylabel('SOP')
plt.show()
```

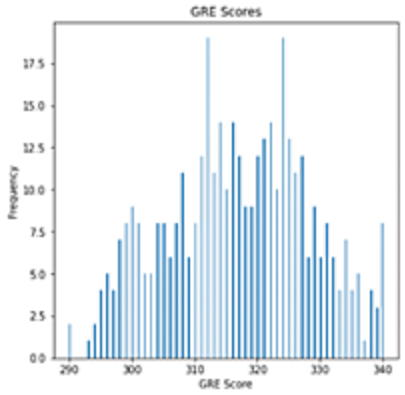



In [59]:
```
admission[admission.CGPA >= 8.0].plot(kind='scatter', x='GRE Score', y='TOEFL Score',color="BLUE")

plt.xlabel("GRE Score")
plt.ylabel("TOEFL SCORE")
plt.title("CGPA>=8.0")
plt.grid(True)

plt.show()
```



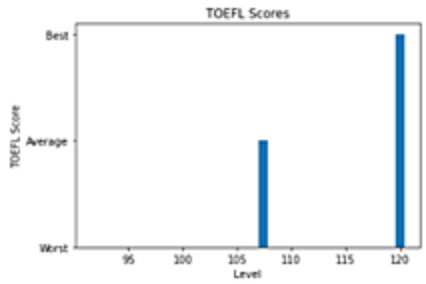
In [60]:

```
admission["GRE Score"].plot(kind = 'hist',bins = 200,figsize = (6,6))

plt.title("GRE Scores")
plt.xlabel("GRE Score")
plt.ylabel("Frequency")

plt.show()
```



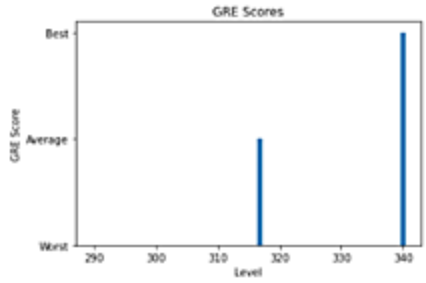

In [61]:

```
p = np.array([admission["TOEFL Score"].min(),admission["TOEFL Score"].mean(),admission["TOEFL Score"].max()])
r = ["Worst","Average","Best"]
plt.bar(p,r)

plt.title("TOEFL Scores")
plt.xlabel("Level")
plt.ylabel("TOEFL Score")

plt.show()
```



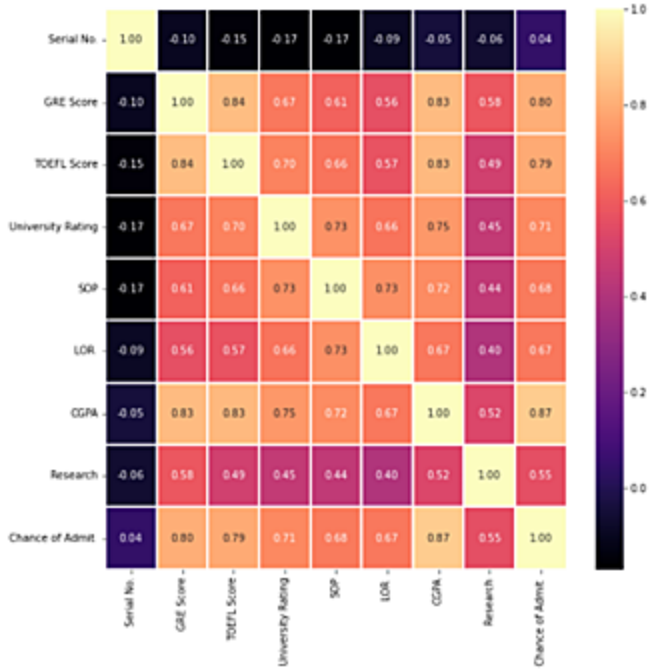

In [62]:

```
g = np.array([admission["GRE Score"].min(),admission["GRE Score"].mean(),admission["GRE Score"].max()])
h = ["Worst","Average","Best"]
plt.bar(g,h)

plt.title("GRE Scores")
plt.xlabel("Level")
plt.ylabel("GRE Score")

plt.show()
```



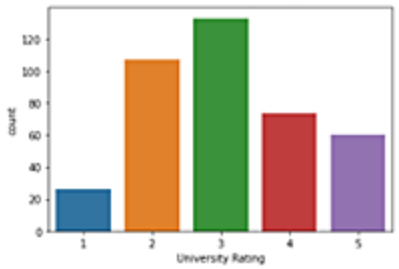
In [63]:

```
import seaborn as sns
plt.figure(figsize=(10, 10))
sns.heatmap(admission.corr(), annot=True, linewidths=0.05, fmt= '.2f',cmap="magma")
plt.show()
```



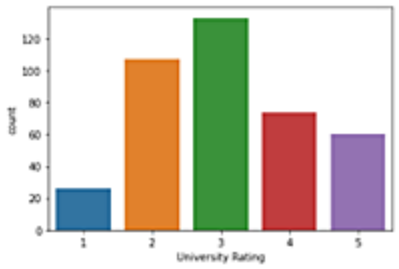
In [64]:
```
admission.Research.value_counts()

sns.countplot(x="University Rating",data=admission)
```

Out[64]:



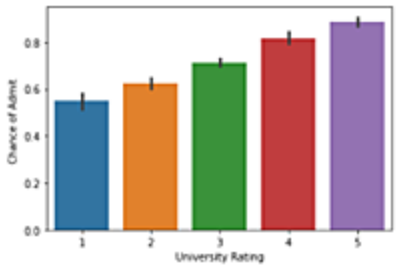

In [65]:
```
admission.Research.value_counts()

sns.countplot(x="University Rating",data=admission)
```

Out[65]:




In [66]:
```
sns.barplot(x="University Rating", y="Chance of Admit ", data=admission)
```

Out[66]:




In [67]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20)
```

In [68]:
```
X_train.shape
```

Out[68]: (320, 7)

In [69]:
```
X_test.shape
```

Out[69]: (80, 7)

In [70]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
X_train[X_train.columns] = scaler.fit_transform(X_train[X_train.columns])
X_test[X_test.columns] = scaler.transform(X_test[X_test.columns])
X_train.head()
```

Out[70]:

| | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Research |
|---|---|---|---|---|---|---|---|
| **310** | 0.60 | 0.428571 | 0.5 | 0.500 | 0.625 | 0.621795 | 1.0 |
| **260** | 0.74 | 0.571429 | 1.0 | 1.000 | 0.625 | 0.746795 | 1.0 |
| **330** | 0.74 | 0.750000 | 0.5 | 0.625 | 0.500 | 0.596154 | 1.0 |
| **132** | 0.38 | 0.464286 | 1.0 | 0.625 | 0.625 | 0.564103 | 0.0 |
| **155** | 0.44 | 0.607143 | 0.5 | 0.500 | 0.500 | 0.605769 | 0.0 |

In [71]:
```
from sklearn.ensemble import RandomForestRegressor
rgr=RandomForestRegressor()
rgr.fit(X_train,y_train)
```

Out[71]:
```
RandomForestRegressor()
```

In [72]:
```
rgr.score(X_test,y_test)
```

Out[72]:
```
0.7905367721307983
```

In [73]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.5.2)
Requirement already satisfied: numpy in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from xgboost) (1.20.3)
Requirement already satisfied: scipy in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from xgboost) (1.7.3)
Note: you may need to restart the kernel to use updated packages.
```

In [74]:
```
import xgboost as xgb
xg = xgb.XGBRegressor()
xg.fit(X_train,y_train)
```

Out[74]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=56,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

In [75]:
```
xg.score(X_test,y_test)
```

Out[75]:
```
0.73737345050915
```

In [76]:
```
y_predict=rgr.predict(X_test)
y_predict
#Y_test.shape
```

Out[76]:
```
array([0.7573, 0.8436, 0.6794, 0.8412, 0.9302, 0.9169, 0.7198, 0.5111,
       0.5273, 0.9429, 0.6098, 0.6066, 0.6224, 0.7344, 0.8156, 0.5989,
       0.6294, 0.6162, 0.9044, 0.4879, 0.6199, 0.7667, 0.5973, 0.6986,
       0.7053, 0.7453, 0.653 , 0.6756, 0.7956, 0.8181, 0.8423, 0.779 ,
       0.7554, 0.9338, 0.6089, 0.5187, 0.7301, 0.8035, 0.6736, 0.7962,
       0.7686, 0.5496, 0.9349, 0.7242, 0.8889, 0.8502, 0.8237, 0.711 ,
       0.6185, 0.7838, 0.9428, 0.7005, 0.6615, 0.93  , 0.9179, 0.738 ,
       0.699 , 0.7166, 0.6574, 0.8452, 0.4883, 0.9037, 0.4946, 0.8499,
       0.5354, 0.4776, 0.6463, 0.7462, 0.9589, 0.6731, 0.5533, 0.8749,
       0.7411, 0.664 , 0.8675, 0.9369, 0.7007, 0.6405, 0.6715, 0.422 ])
```

In [77]:
```
from sklearn.metrics import mean_squared_error, r2_score,mean_absolute_error
import numpy as np

print("Mean Absolute Error:", mean_absolute_error(y_test, y_predict))
print("Mean Squared Error:", mean_squared_error(y_test, y_predict))
print("Root Mean Squared Error:", np.sqrt(mean_squared_error(y_test, y_predict)))
```

```
Mean Absolute Error: 0.04643874999999999
Mean Squared Error: 0.004074108875000005
Root Mean Squared Error: 0.06382874646270287
```

In [78]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

In [79]:
```
lr_model=lr.fit(X_train,y_train)
pred_lr=lr_model.predict(X_test)
```

In [80]:
```
lr.score(X_test,y_test)
```

Out[80]: 0.8123922665035088

In [81]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [82]:
```
dr=DecisionTreeRegressor()
dr.fit(X_train,y_train)
dr.score(X_test,y_test)
```

Out[82]: 0.6596442041588508

In [84]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url" : "https://us-south.ml.cloud.ibm.com",
    "apikey" : "ea40K5XPtK0m7Bulo5Tryw3cns7e7SU4GdkXuH55Ck1o"
}
client = APIClient(wml_credentials)
```

In [85]:
```
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])
```

In [86]:
```
space_uid = guid_from_space_name(client, 'admitpredict')
print("sSpace UID = "+space_uid)
```

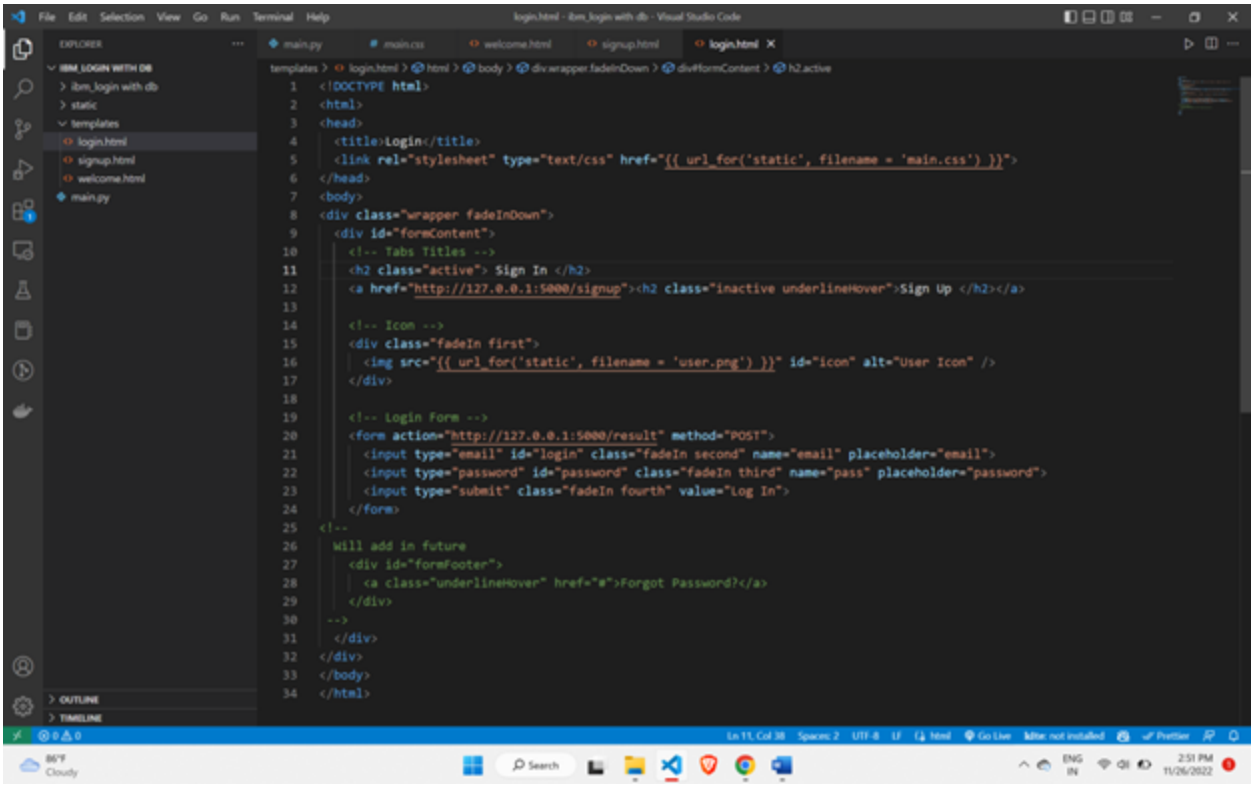
sSpace UID = da1959c6-5f6f-47dc-871b-d701ebd08177

In [87]:
```
client.set.default_space(space_uid)
```

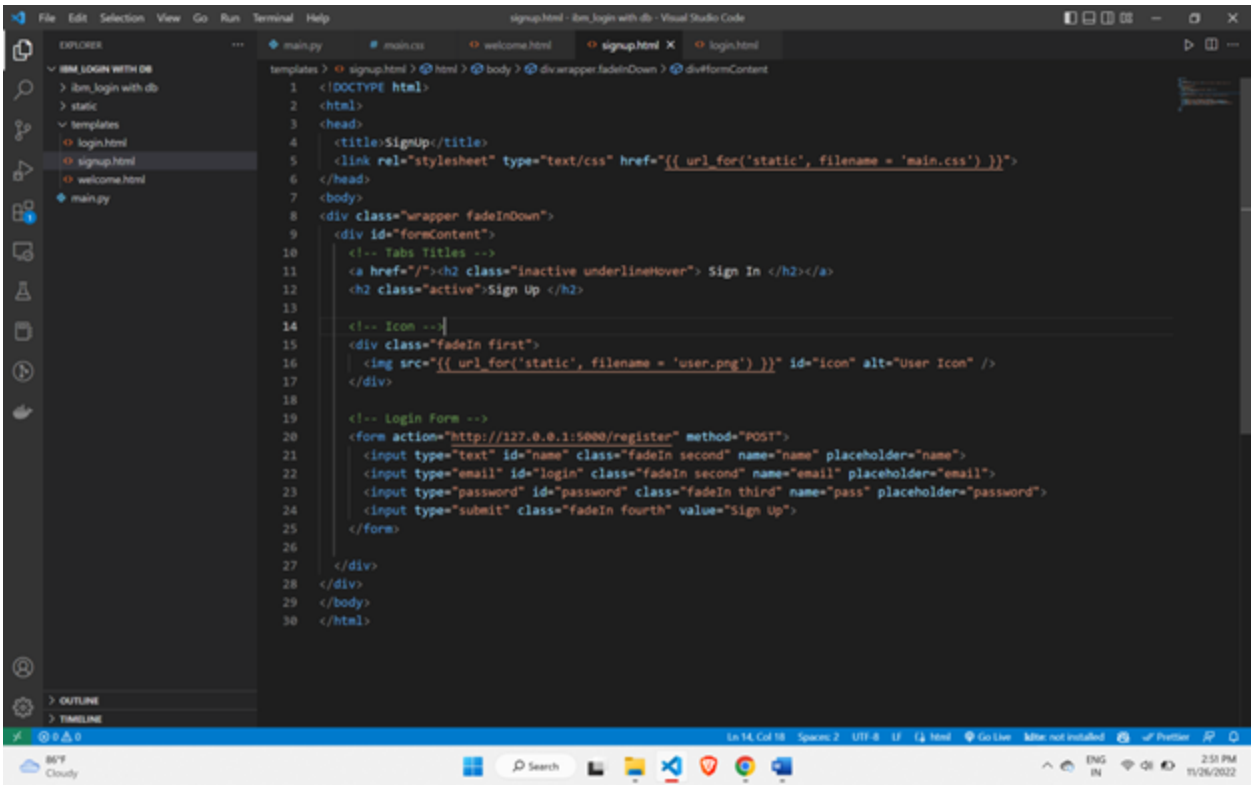
Out[87]: 'SUCCESS'

b. Web Application

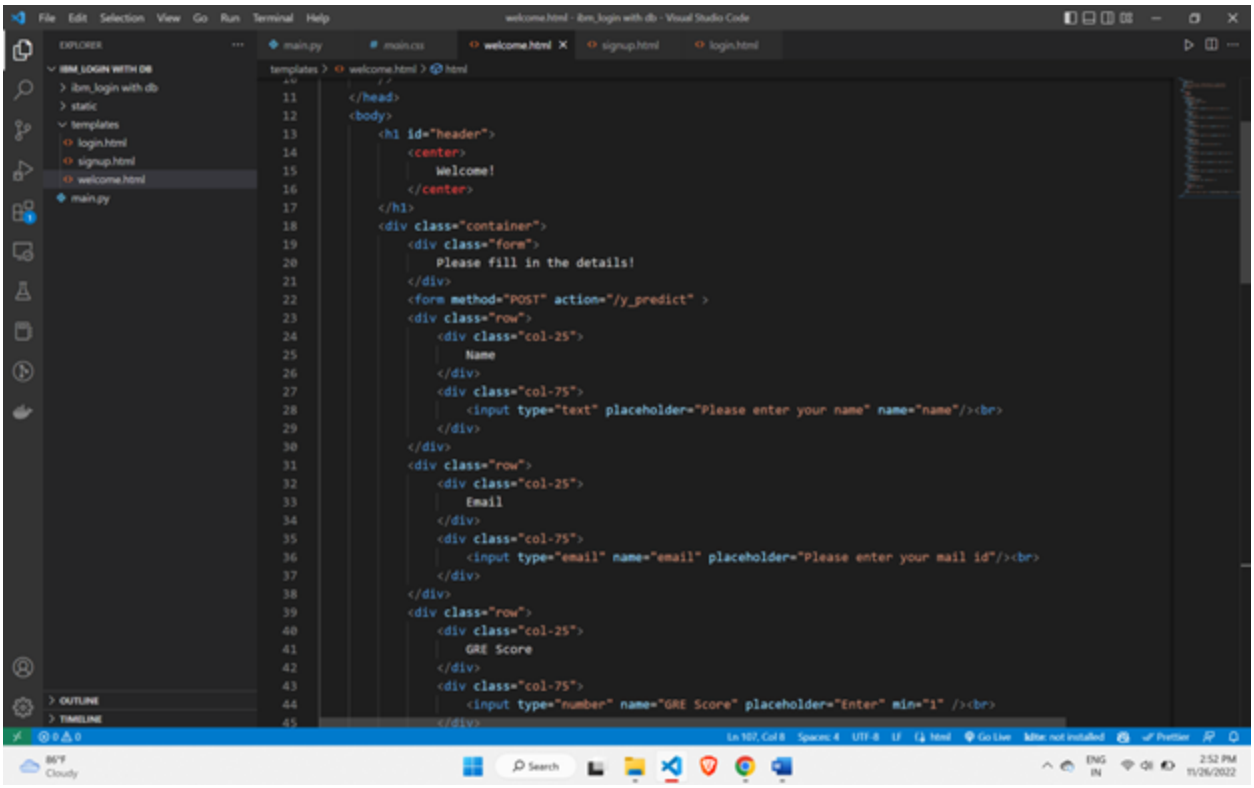
Login.html



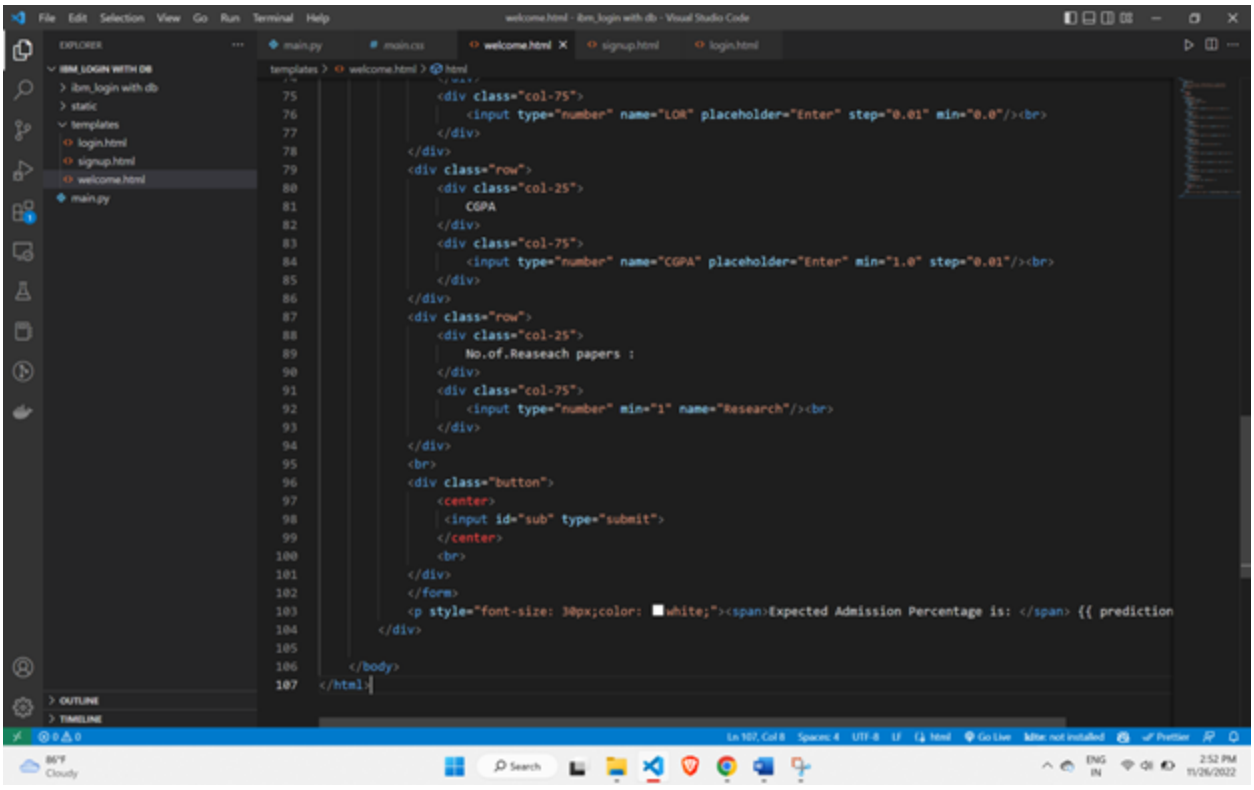
Signup.html


File Edit Selection View Go Run Terminal Help
signup.html - ibm_login with db - Visual Studio Code
EXPLORER
IBM_LOGIN WITH DB
ibm_login with db
static
templates
login.html
signup.html
welcome.html
main.py
main.py
main.css
welcome.html
signup.html
login.html
templates > signup.html > html > body > div.wrapper.fadeInDown > div#formContent
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>SignUp</title>
5 <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename = 'main.css') }}">
6 </head>
7 <body>
8 
9 
10 <!-- Tabs Titles -->
11 <a href="/"><h2 class="inactive underlineHover"> Sign In </h2></a>
12 <h2 class="active">Sign Up </h2>
13
14 <!-- Icon -->
15 
16 <img src="{{ url_for('static', filename = 'user.png') }}" id="icon" alt="User Icon" />
17 
18
19 <!-- Login Form -->
20 <form action="http://127.0.0.1:5000/register" method="POST">
21 <input type="text" id="name" class="fadeIn second" name="name" placeholder="name">
22 <input type="email" id="login" class="fadeIn second" name="email" placeholder="email">
23 <input type="password" id="password" class="fadeIn third" name="pass" placeholder="password">
24 <input type="submit" class="fadeIn fourth" value="Sign Up">
25 </form>
26
27 
28 
29 </body>
30 </html>
OUTLINE
TIMELINE
Ln 14, Col 18 Spaces: 2 UTF-8 LF html Go Live Mitte: not installed Prettier
86°F Cloudy
Search
ENG IN
2:51 PM 11/26/2022

Welcome.html

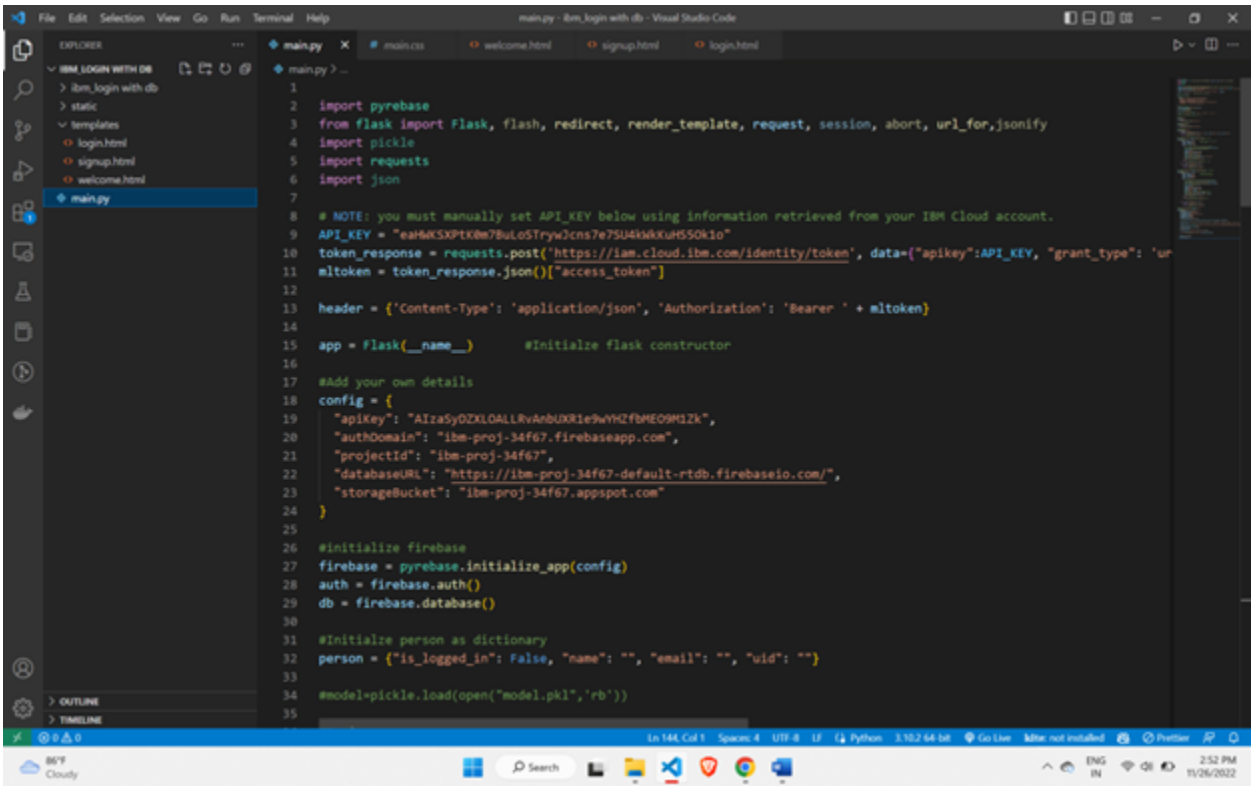


```
    </head>
    <body>
        <h1 id="header">
            <center>
                Welcome!
            </center>
        </h1>
        <div class="container">
            <div class="form">
                Please fill in the details!
            </div>
            <form method="POST" action="/y_predict" >
            <div class="row">
                <div class="col-25">
                    Name
                </div>
                <div class="col-75">
                    <input type="text" placeholder="Please enter your name" name="name"/><br>
                </div>
            </div>
            <div class="row">
                <div class="col-25">
                    Email
                </div>
                <div class="col-75">
                    <input type="email" name="email" placeholder="Please enter your mail id"/><br>
                </div>
            </div>
            <div class="row">
                <div class="col-25">
                    GRE Score
                </div>
                <div class="col-75">
                    <input type="number" name="GRE Score" placeholder="Enter" min="1" /><br>
```

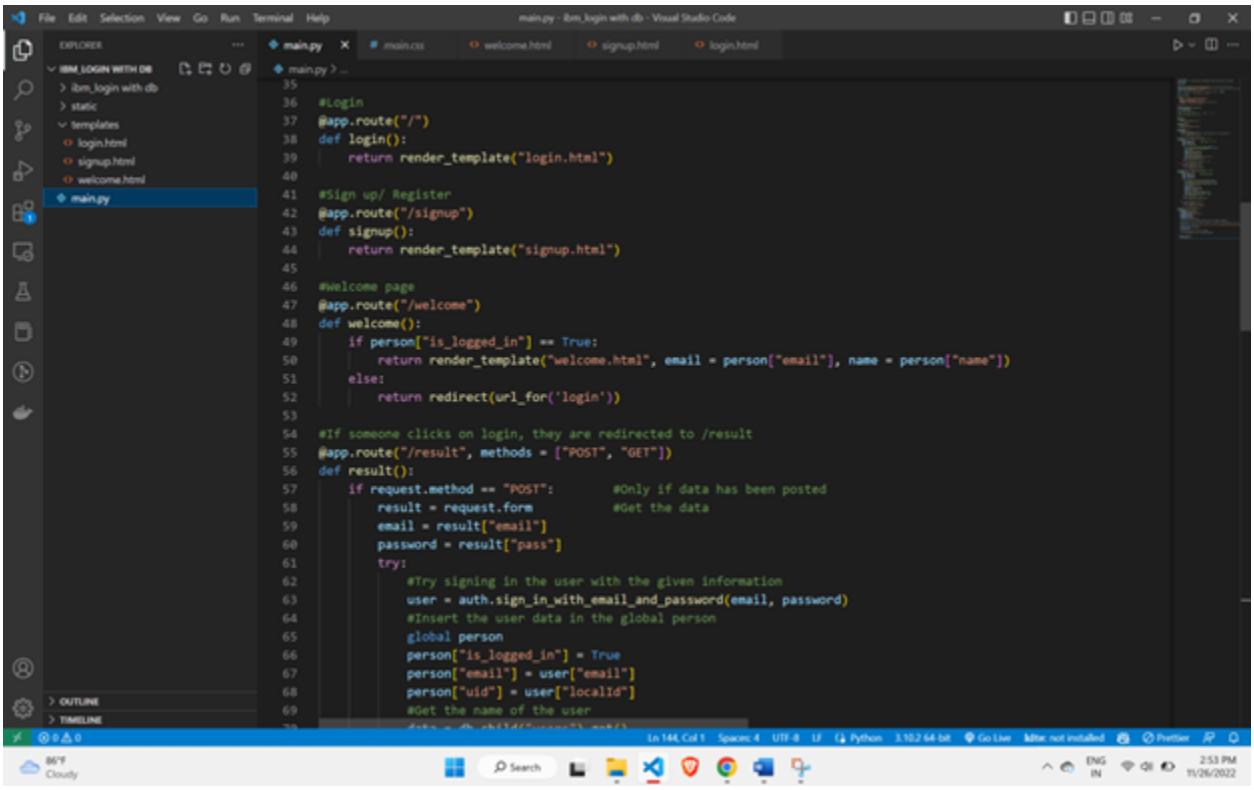


```
                <div class="col-75">
                    <input type="number" name="LOR" placeholder="Enter" step="0.01" min="0.0"/><br>
                </div>
            </div>
            <div class="row">
                <div class="col-25">
                    CGPA
                </div>
                <div class="col-75">
                    <input type="number" name="CGPA" placeholder="Enter" min="1.0" step="0.01"/><br>
                </div>
            </div>
            <div class="row">
                <div class="col-25">
                    No.of.Reaseach papers :
                </div>
                <div class="col-75">
                    <input type="number" min="1" name="Research"/><br>
                </div>
            </div>
            <br>
            <div class="button">
                <center>
                <input id="sub" type="submit">
                </center>
                <br>
            </div>
            </form>
            <p style="font-size: 30px;color: white;"><span>Expected Admission Percentage is: </span> {{ prediction
        </div>

    </body>
</html>
```

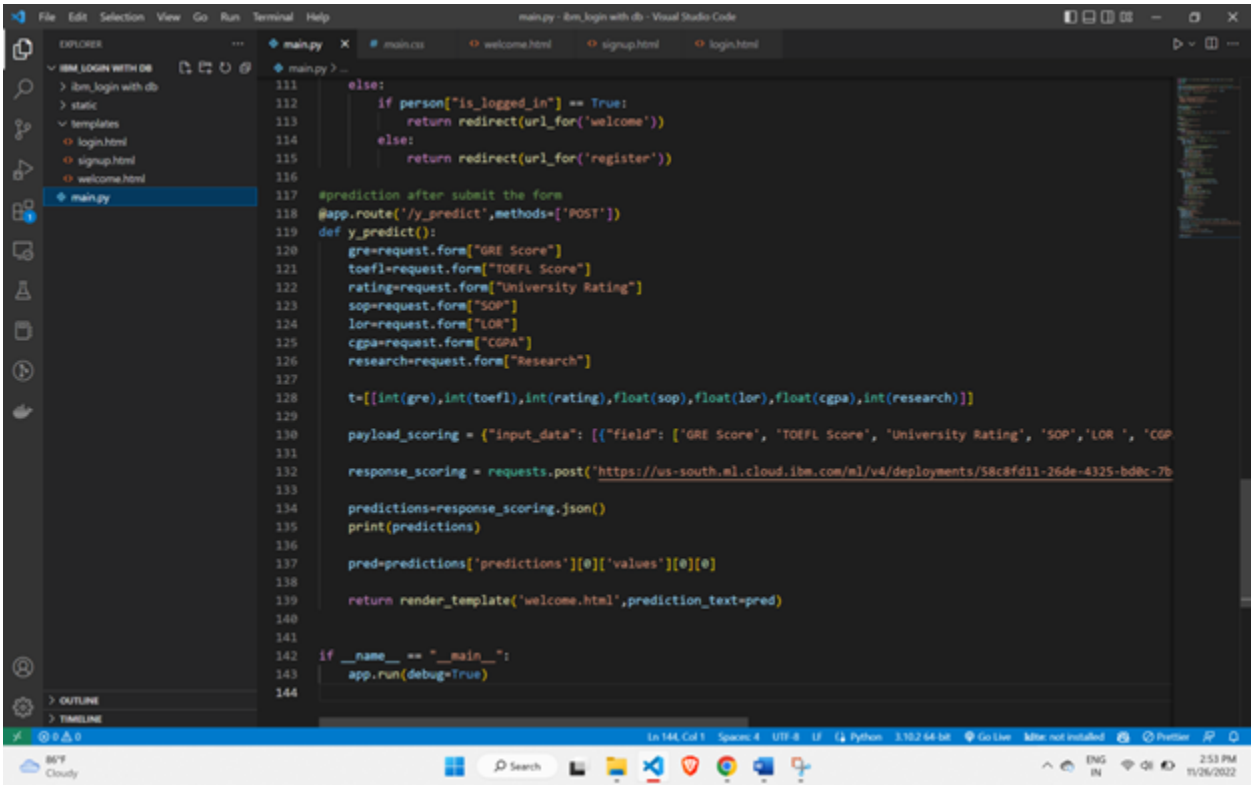
App.py



```python
import pyrebase
from flask import Flask, flash, redirect, render_template, request, session, abort, url_for,jsonify
import pickle
import requests
import json

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "eaHkWSXPtK0m7BuLoSTrywJcns7e7SU4kwkXuHS5Ok1o"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":API_KEY, "grant_type": 'ur
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

app = Flask(__name__)       #Initialze flask constructor

#Add your own details
config = {
  "apiKey": "AIzaSyDZXLOALLRvAnbUXRie9wYHZfbMEO9M1Zk",
  "authDomain": "ibm-proj-34f67.firebaseapp.com",
  "projectId": "ibm-proj-34f67",
  "databaseURL": "https://ibm-proj-34f67-default-rtdb.firebaseio.com/",
  "storageBucket": "ibm-proj-34f67.appspot.com"
}

#initialize firebase
firebase = pyrebase.initialize_app(config)
auth = firebase.auth()
db = firebase.database()

#Initialze person as dictionary
person = {"is_logged_in": False, "name": "", "email": "", "uid": ""}

#model=pickle.load(open("model.pkl",'rb'))
```



```python
#Login
@app.route("/")
def login():
    return render_template("login.html")

#Sign up/ Register
@app.route("/signup")
def signup():
    return render_template("signup.html")

#Welcome page
@app.route("/welcome")
def welcome():
    if person["is_logged_in"] == True:
        return render_template("welcome.html", email = person["email"], name = person["name"])
    else:
        return redirect(url_for('login'))

#If someone clicks on login, they are redirected to /result
@app.route("/result", methods = ["POST", "GET"])
def result():
    if request.method == "POST":        #Only if data has been posted
        result = request.form           #Get the data
        email = result["email"]
        password = result["pass"]
        try:
            #Try signing in the user with the given information
            user = auth.sign_in_with_email_and_password(email, password)
            #Insert the user data in the global person
            global person
            person["is_logged_in"] = True
            person["email"] = user["email"]
            person["uid"] = user["localId"]
            #Get the name of the user
```


File Edit Selection View Go Run Terminal Help
main.py - ibm_login with db - Visual Studio Code
EXPLORER
IBM_LOGIN WITH DB
ibm_login with db
static
templates
login.html
signup.html
welcome.html
main.py
main.py
main.css
welcome.html
signup.html
login.html
main.py > ...
111 else:
112 if person["is_logged_in"] == True:
113 return redirect(url_for('welcome'))
114 else:
115 return redirect(url_for('register'))
116
117 #prediction after submit the form
118 @app.route('/y_predict',methods=['POST'])
119 def y_predict():
120 gre=request.form["GRE Score"]
121 toefl=request.form["TOEFL Score"]
122 rating=request.form["University Rating"]
123 sop=request.form["SOP"]
124 lor=request.form["LOR"]
125 cgpa=request.form["CGPA"]
126 research=request.form["Research"]
127
128 t=[[int(gre),int(toefl),int(rating),float(sop),float(lor),float(cgpa),int(research)]]
129
130 payload_scoring = {"input_data": [{"field": ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP','LOR ', 'CGP
131
132 response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/58c8fd11-26de-4325-bd0c-7b
133
134 predictions=response_scoring.json()
135 print(predictions)
136
137 pred=predictions['predictions'][0]['values'][0][0]
138
139 return render_template('welcome.html',prediction_text=pred)
140
141
142 if __name__ == "__main__":
143 app.run(debug=True)
144
OUTLINE
TIMELINE
Ln 144, Col 1 Spaces: 4 UTF-8 LF Python 3.10.2 64-bit Go Live Ide: not installed Prettier
86°F
Cloudy
Search
ENG
IN
2:53 PM
11/26/2022

## 8. **TESTING**

### a. Test Cases

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Date | 3-Nov-22 | | | | |
| | | | | Team ID | PNT2022TMID35985 | | | | |
| | | | | Project Name | University Admit Eligibility Predictor | | | | |
| | | | | Maximum Marks | 4 marks | | | | |
| **Test case ID** | **Feature Type** | **Component** | **Test Scenario** | **Pre-Requisite** | **Steps To Execute** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| LoginPage_TC_OO1 | Functional | Home Page | Verify user is able to see the Login/register popup when user clicked on My account button | | 1.Enter URL and click go<br>2.Click on My Account dropdown button<br>3.Verify login/register popup displayed or not | Username: abc@gmail.com<br>password: 123456789 | Login/register popup should display | Working as expected | Pass |
| LoginPage_TC_OO2 | UI | Home Page | Verify the UI elements in Login/Signup popup | | 1.Enter URL and click go<br>2.Verify login/Signup popup with below UI elements:<br>a.email text box<br>b.password text box<br>c.Login button<br>d.Don't have an account? Sign Up | https://shopenzer.com/ | Application should show below UI elements:<br>a.email text box<br>b.password text box<br>c.Login button<br>d.Don't have an account? Sign Up | Working as expected | Pass |
| LoginPage_TC_OO3 | Functional | Home page | Verify user is able to log into application with Valid credentials | | 1.Enter URL(https://shopenzer.com/) and click go<br>2.Enter Valid username/email in Email text box<br>3.Enter valid password in password text box<br>4.Click on login button | Username: abc@gmail.com<br>password: 123456789 | User should navigate to form for collecting user details | Working as expected | Pass |
| LoginPage_TC_OO4 | Functional | Login page | Verify user is unable to log into application with invalid credentials | | 1.Enter URL(https://shopenzer.com/) and click go<br>2.Enter invalid username/email in Email text box<br>3.Enter valid password in password text box<br>4.Click on login button | Username: abc@gmail<br>password: 123456789 | Application should show 'invalid email' validation message. | Working as expected | Pass |
| LoginPage_TC_OO4 | Functional | Login page | Verify user is unable to log into application with invalid credentials | | 1.Enter URL(https://shopenzer.com/) and click go<br>2.Enter invalid username/email in Email text box<br>3.Enter valid password in password text box<br>4.Click on login button | Username: abc<br>password: 123456789 | Application should show 'Missing @ in email' validation message. | Working as expected | Pass |
| LoginPage_TC_OO5 | Functional | Login page | Verify user is unable to log into application with invalid credentials | | 1.Enter URL(https://shopenzer.com/) and click go<br>2.Enter valid username/email in Email text box<br>3.Enter invalid password in password text box<br>4.Click on login button | Username: abc@gmail.com<br>password: 1234567890 | Application should show 'Incorrect email or password ' validation message. | Working as expected | Pass |
| SignUpPage_TC_OO6 | Functional | Sign up page | Verify user is able to sign up with valid email | | 1.Enter URL(https://shopenzer.com/) and click go<br>2.Select sign up option<br>3.Enter valid username/email in Email text box<br>4.Click on login button | Email: abc@gmail.com | Application should acknowledge successful sign up | Working as expected | Pass |
| DashboardForm_TC_OO7 | Functional | Dashboard Form Validation | Verify values entered for marks are non-negative | | 1.Enter URL(https://shopenzer.com/) and click go<br>2.Login with valid credentials<br>3.Enter valid values for numerical fields | | Application should allow submission of form | Working as expected | Pass |

### b. User Acceptance Testing

#### i. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved.

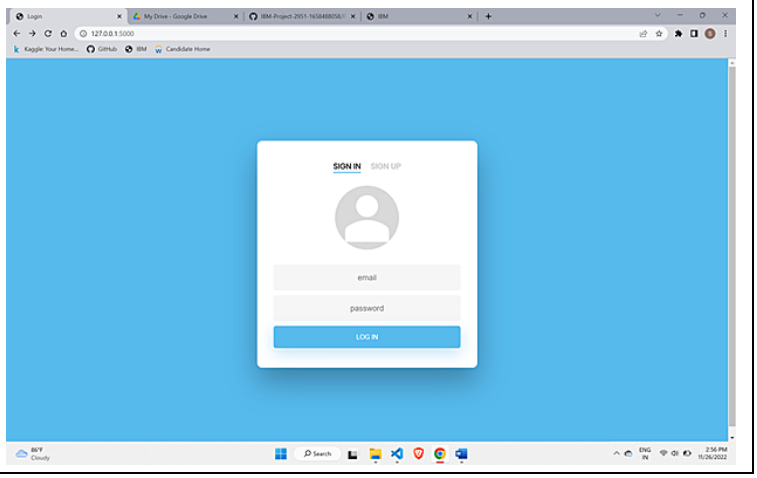
| Resolution | Severity 1 | Severity 2 | Severity 3 | Severity 4 | Subtotal |
|---|---|---|---|---|---|
| By Design | 0 | 4 | 2 | 3 | 9 |
| Duplicate | 1 | 0 | 3 | 0 | 4 |
| External | 2 | 3 | 0 | 1 | 6 |
| Fixed | 11 | 2 | 4 | 4 | 21 |
| Not Reproduced | 0 | 0 | 1 | 0 | 1 |
| Skipped | 0 | 0 | 1 | 1 | 2 |
| Won't Fix | 0 | 5 | 2 | 1 | 8 |
| Totals | 14 | 14 | 13 | 10 | 51 |

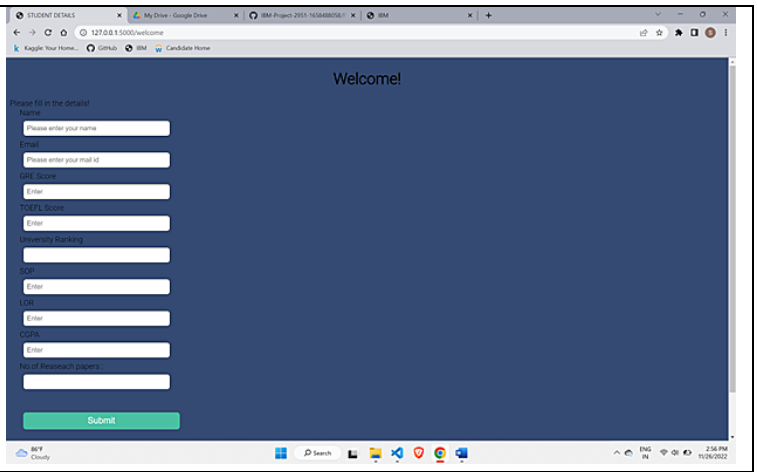
ii. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

| Section | Total Cases | Not Tested | Fail | Pass |
|---|---|---|---|---|
| Print Engine | 7 | 0 | 0 | 7 |
| Client Application | 3 | 0 | 0 | 3 |
| Security | 2 | 0 | 0 | 2 |
| Outsource Shipping | 3 | 0 | 0 | 3 |
| Exception Reporting | 9 | 0 | 0 | 9 |
| Final Report Output | 4 | 0 | 0 | 4 |
| Version Control | 2 | 0 | 0 | 2 |

## 9. RESULTS (Performance Metrics)

| S.No. | Parameter | Screenshot / Values |
|---|---|---|
| 1. | Dashboard design | No of Visualizations / Graphs -   |

| | | |
|---|---|---|
| | |  |
| 2. | Metrics | `print('Mean Absolute Error:', mean_absolute_error(y_test,pred_lr))`<br>`print('Mean Squared Error:', mean_squared_error(y_test,pred_lr))`<br>`print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test,pred_lr)))`<br><br>Mean Absolute Error: 0.047854809740532175<br>Mean Squared Error: 0.004565099424278329<br>Root Mean Squared Error: 0.06756551949240329 |

## 10. **ADVANTAGES & DISADVANTAGES**

### a. Advantages

i. Reaching pupils who are dispersed geographically
ii. Reducing time in activities.
iii. handling of data centrally.
iv. Admission without paperwork and with fewer staff.
v. Operative effectiveness.

### b. Disadvantages

i. Users have to input all the values of the parameters .
ii. The ratings being subjective will not yield perfect result.
iii. There is no standard benchmark, causing in variance.

## 11. **CONCLUSION**

This project has been successfully developed a web application that will predict university admit eligibilty for a candidate. The front end takes in different parameters from scores to university ranking a candidate is looking for. These values are taken to the back end

where a linear regression model is built to compute the chances of admit.

## 12. **FUTURE SCOPE**

This project can be further supported with real time requirements from the universities available and conduct analysis on the current sets of candidates to see how each individual stand in their batch.

## 13. **APPENDIX**

Source Code: https://github.com/IBM-EPBL/IBM-Project-2951-1658488058

Github Project Link: https://github.com/IBM-EPBL/IBM-Project-2951-1658488058/tree/main/PROJECT%20DEVELOPMENT%20PHASE/Sprint%204

Video Demo: https://drive.google.com/file/d/12mSRFACwSvGR6P5snfq1zZZC7cf-pNY_/view?usp=share_link