

Project Report

IBM-Project-55486-1669096804

University Admit Eligibility Predictor

1. INTRODUCTION

1.1 Project Overview

The problem statement is to design a college prediction/ prediction system and to provide a probabilistic insight into college administration for overall rating, cut-offs of the colleges, admission intake and preferences of students.

It has always been a troublesome process for students in finding the perfect university and course for their further studies.

At times they do know which stream they want to get into, but it is not easy for them to find colleges based on their academic marks and other performances.

We aim to develop and provide a place which would give a probabilistic output of how likely it is to get into a university given their details.

1.2 Purpose

- Students are often worried about their chances of admission to University.
- The aim of this project is to help students in shortlisting universities with their profiles.
- The predicted output gives them a fair idea about their admission chances to a particular university.
- This analysis should also help students who are currently preparing or will be preparing to get a better idea.

2. LITERATURE SURVEY

2.1 Existing problem

- We have so many websites with problems in inaccuracy and not getting the right thing out.
- Our project, which is based on University Admit Eligibility Predictor, with a great accuracy mark, gives the output more effectively and efficiently.

2.2 References

- P.KaviPriya, “A Review on Predicting Students’ Academic Performance Earlier, Using Data Mining Techniques”, International Jour-nal of Advanced Research in Computer Science and Software Engineering
- Ali Daud, Naif Radi Aljohani, “Predicting Student Performance using Advanced Learning Analytics”, 2017 International World Wide Web Conference Committee (IW3C2).
- Marium-E-Jannat,SaymaSultana,Munira Akther, “A Probabilistic Machine Learning Approach for Eligible Candidate Selection”, Inter-national Journal of Computer Applications (0975 – 8887)Volume 144 – No.10, June 2016
- Sudheep Elayidom, Dr. Sumam Mary Idikkula, “Applying Data mining using Statistical Techniques for Career Selection”, International Journal of Recent Trends in Engineering, Vol. 1, No. 1, May 2009.
- Dr. Mahendra Tiwari ,Manmohan Mishra, “Accuracy Estimation of Classification Algorithms with DEMP Model”, International Jour-nal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 11, November 2013.
- Ms. Roshani Ade,Dr. P. R. Deshmukh, “An incremental ensemble of classifiers as a technique for prediction of student’s career choice”, 2014 First International Conference on Networks & Soft Compu-ting
- Nikita Gorad ,Ishani Zalte, “Career Counselling Using Data Mining”, International Journal of Innovative Research in Computer and Communication Engineering.
- Bo Guo , Rui Zhang, “Predicting Students Performance in Educa-tional Data Mining”,2015 International Symposium on Educational Technology
- Ali Daud , Naif Radi Aljohani , “Predicting Student Performance using Advanced Learning Analytics”
- Rutvija Pandya Jayati Pandya , “C5.0 Algorithm to Improved KNN with Feature Selection and Reduced Error Pruning“, Inter-national Journal of Computer Applications (0975 – 8887) Volume 117 – No. 16, May 2015.

- Comparative Analysis of KNN Algorithms: ID3, C4.5 and KNN Shiju Sathyadevan and Remya R. Nair
- Yu Lou, Ran Ren, "A Machine Learning Approach for Future Career Planning"
- Gareth James, Daniela Witten, Trevor Hastie, "An Introduction to Statistical Learning with Applications in R"
- Anuj Karpatne, Gowtham Atluri, "Theory- Guided Data Science: A New Paradigm for Scientific Discovery from Data", IEEE transactions on knowledge and data engineering, vol.29, no. 10, october 2017.

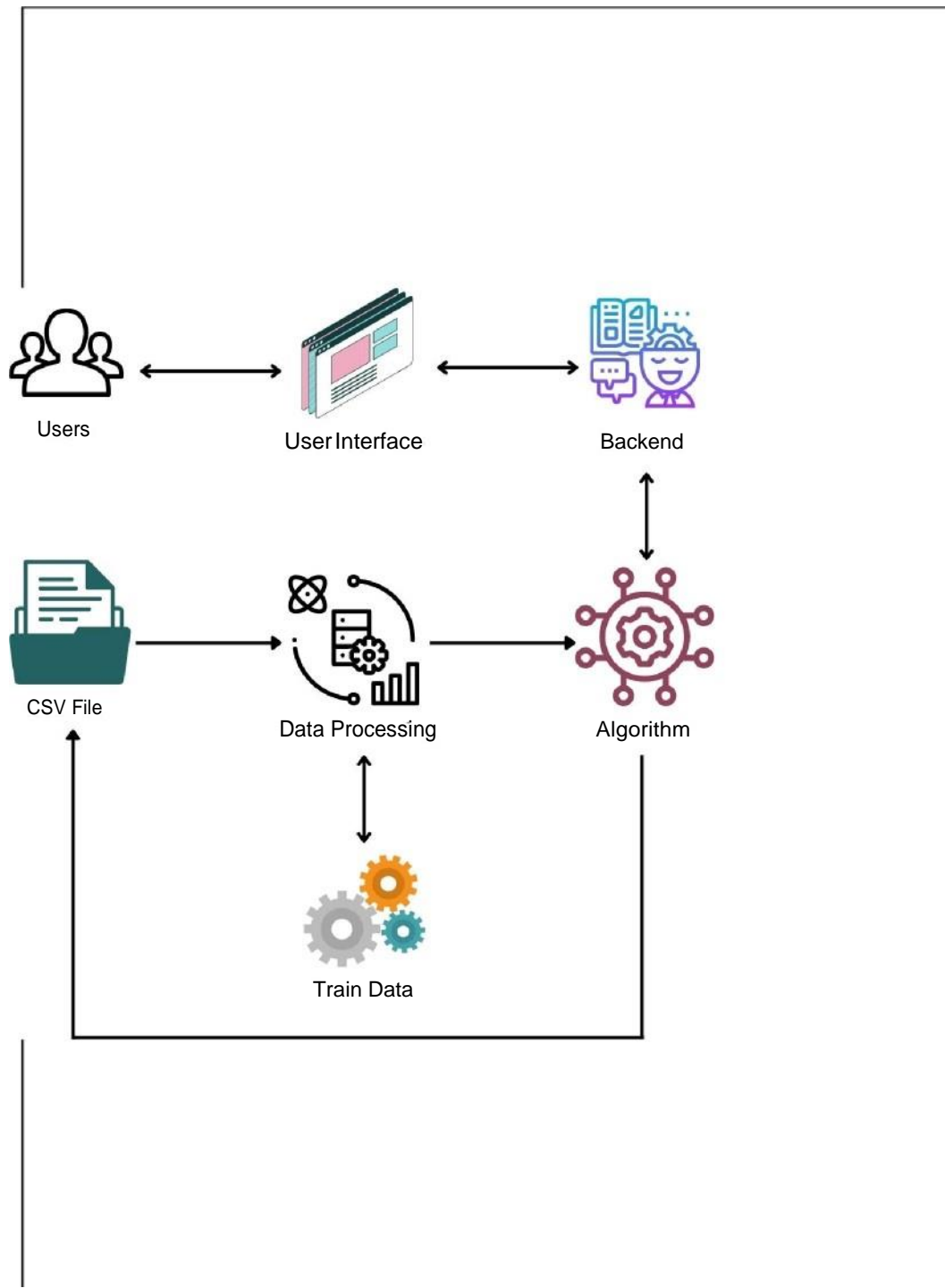
2.3 Problem Statement Definition

Problem Statement

- The problem statement is to design a college prediction/ prediction system and to provide a probabilistic insight into college administration for overall rating, cut-offs of the colleges, admission intake and preferences of students.
- It has always been a troublesome process for students in finding the perfect university and course for their further studies.
- At times they do know which stream they want to get into, but it is not easy for them to find colleges based on their academic marks and other performances.
- We aim to develop and provide a place which would give a probabilistic output of how likely it is to get into a university given their details.

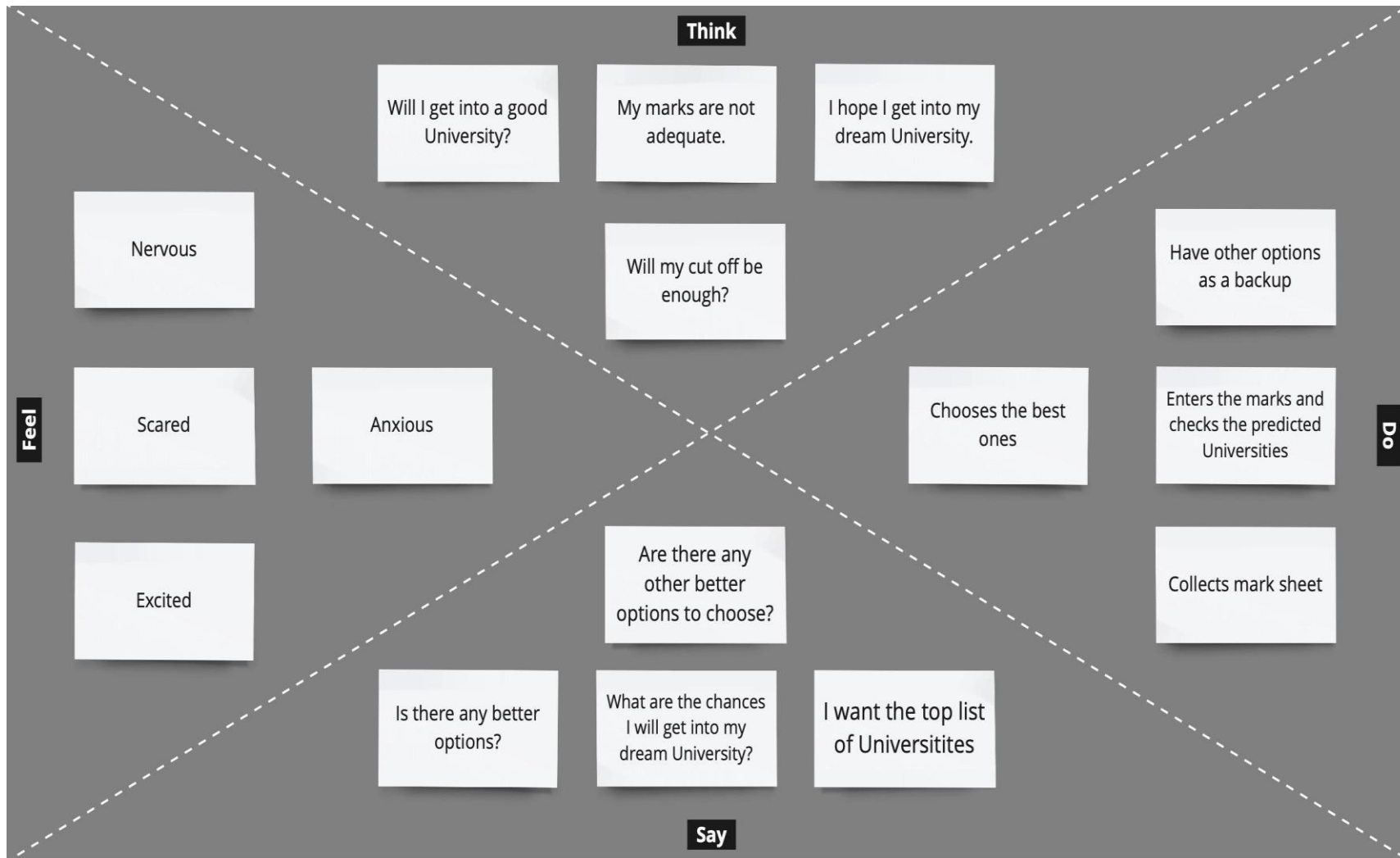
Abstract:

- Students are often worried about their chances of admission to University.
- The aim of this project is to help students in shortlisting universities with their profiles.
- The predicted output gives them a fair idea about their admission chances to a particular university.
- This analysis should also help students who are currently preparing or will be preparing to get a better idea.



3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



PAIN

- Worried if eligible for good Universities
- Anxious about not having enough marks
- Nervous to see the eligibility criteria
- Less awareness about the number of good colleges

GAIN

- Gets the list of good Universities
- Strong about the marks gained
- Fine with the correct eligibility criteria
- More awareness about the number of good colleges

3.2 Ideation & Brainstorming

BRAINSTORMING

1. A beautiful interface will be created with UX Research in mind to give users the best possible User Interface and Experience.

2. The user will enter the marks of their Grade 12 board exam

3. This mark will be forwarded to the backend

4. The algorithm for this particular program will take the inputs and process it

5. The algorithm will fetch the data from the predefined CSV file which contains the list of Universities

6. Now this data will be processed by using Applied Data Science method

7. This ADS method will also train the data by using a specified model for better predictions

8. Now the data that is fetched by the Algorithm is now transmitted from Backend to Frontend User Interface

3.3 Proposed Solution

Proposed Solution :

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be	Students do not have much idea about the procedures, availability and e universities where they want to join, so they seek help from various consultancies to help them secure admission in the universities based on , for which the students are supposed to pay a hefty amount as consultancy fee.
2.	Idea / Solution description	Providing an accurate prediction for the student's admission into the f their choice based on various parameters like IELTS, GRE, Academic Performance, etc.
3.	Novelty / Uniqueness	It seems there are no web applications for predicting the eligibility criteria for getting in to their dream university and also requirement of d insights on specific areas where they can improve their talents and skills.
4.	Social Impact / Customer Satisfaction	It helps student the right decision for choosing the dream universities. Its st of consultancy services by creating a direct connection between the d their dream universities.

5.	Business Model (Revenue Model)	Universities are under the immense pressure to admit more students and ent for the success. To overcome this pressure, they can make use of odel in which it helps them to ease the intake process of students and improve efficiency.
6.	Scalability of the Solution	Further to reduce the immense pressure faced by the students to get a university, the model can also be involved to consider university for examinations and to maintain the latest eligibility criteria for students.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Landing page	There is no registration from the user end. The users can access the website with ease, without worrying about any security issues.
FR-2	Entering Marks	The users will enter their respective marks that are required. Based on the live data we get from the user, we provide them the list of Universities they are eligible to attend.
FR-3	List Display	The list of Universities will be displayed based on the marks given.

4.2 Non-Functional requirements

Following are the non-functional requirements of the proposed solution.

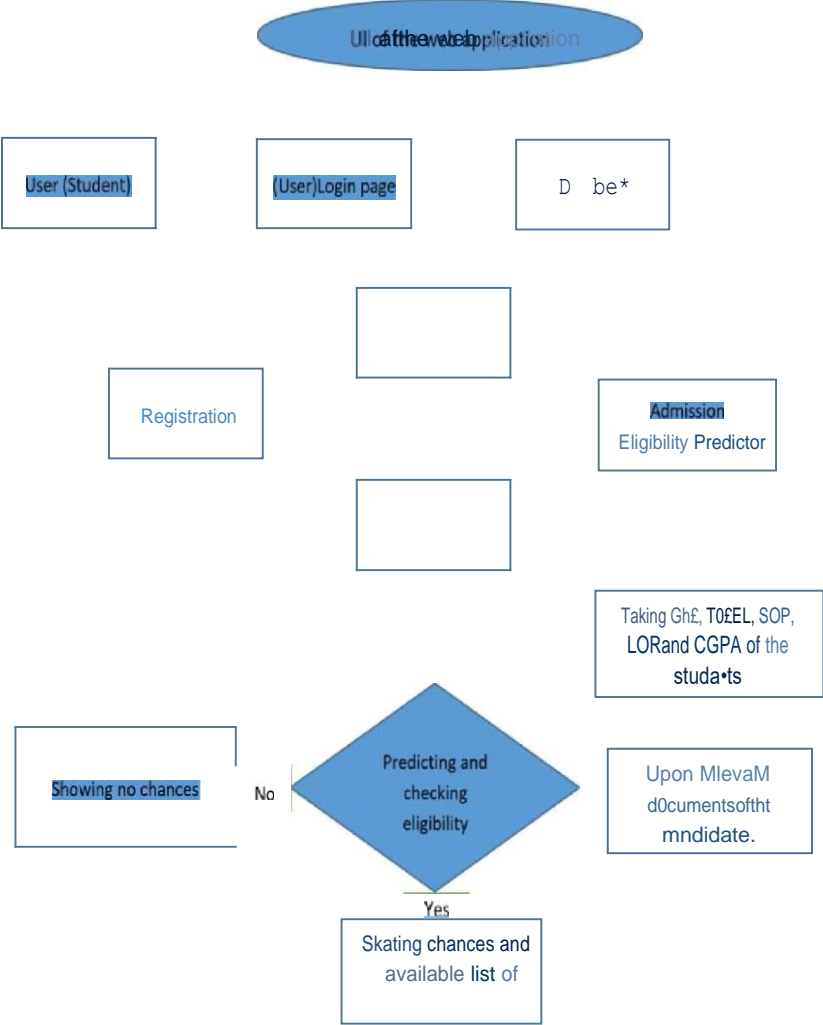
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	User friendly, with direct instructions and UX principles considered.

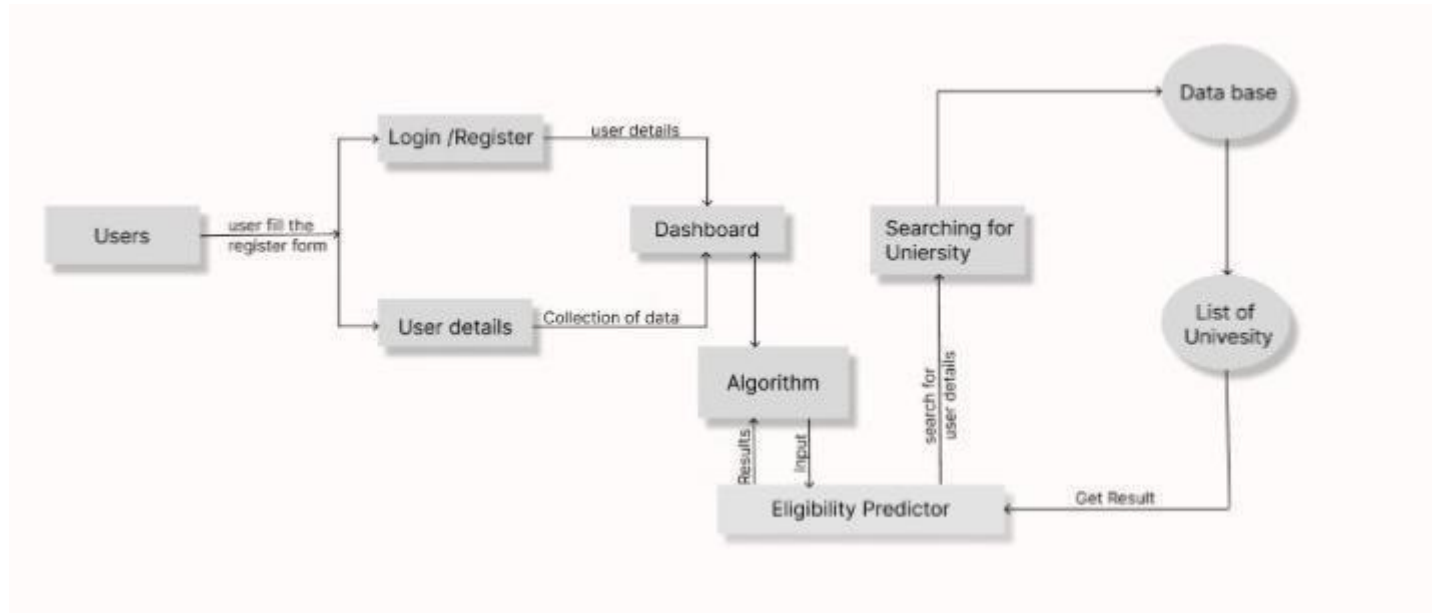
NFR-2	Security	As we don't get the personal data from the user, their data is protected and there won't be any leakage. The system gets trained by passing only the data of marks to the cloud.
NFR-3	Reliability	The website is reliable in terms of immediate information regarding the university decisions.
NFR-4	Performance	It is a light application, with a flask in the backend.
NFR-5	Availability	It is free of cost and available to anyone who is looking to find the Universities that fit them and their needs.
NFR-6	Scalability	It can be further extended to Higher Education and abroad studies.

5. PROJECT DESIGN

5.1 Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.





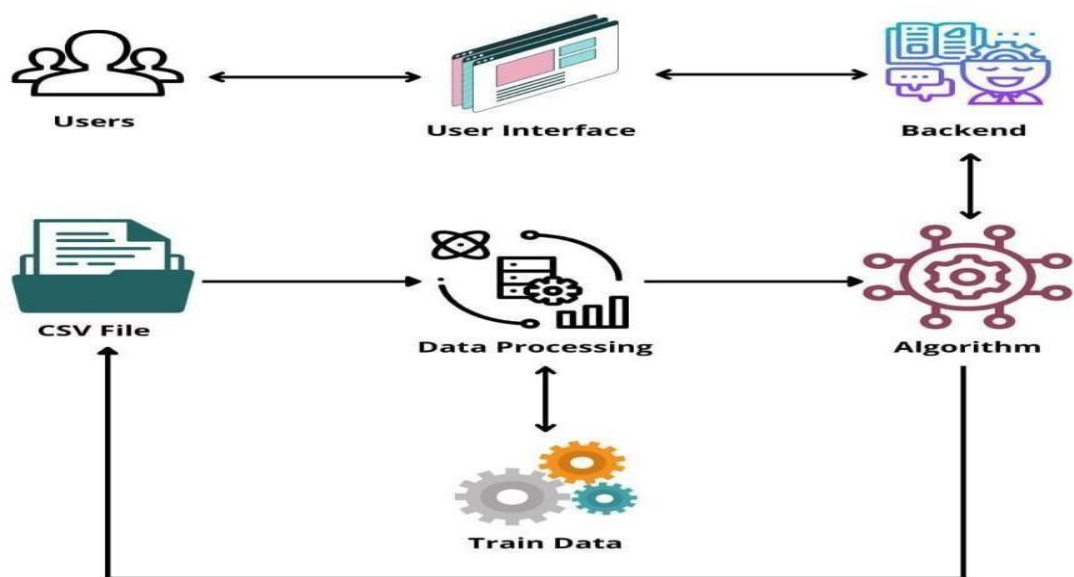
User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer 1 (Web user)	Landing Page	USN-1	It is pretty clear about why we use this website and how it should be used, just by looking at this page.	Understandable	High	Sprint-1
Customer 2 (Web User)	Landing Page	USN-2	The concept of the application is clear with all the how to do instructions and everything.	Very clear	High	Sprint-1
Customer 3 (Web User)	Marks page	USN-3	Its pretty clear what kind of data should be given in.	I can give my marks details.	Low	Sprint-2

Customer 4 (Web User)	Results Page	USN-4	I can see the right and correct results based on the marks given to the system	Got the details	High	Sprint-1
--------------------------	--------------	-------	--------------------------------------------------------------------------------------	-----------------	------	----------

5.2 Solution & Technical Architecture

Architecture Diagram:



Solution Architecture :

Students are often worried about their chances of admission to University.

The aim of this project is to help students in shortlisting universities with their profiles.

The predicted output gives them a fair idea about their admission chances to a particular university.

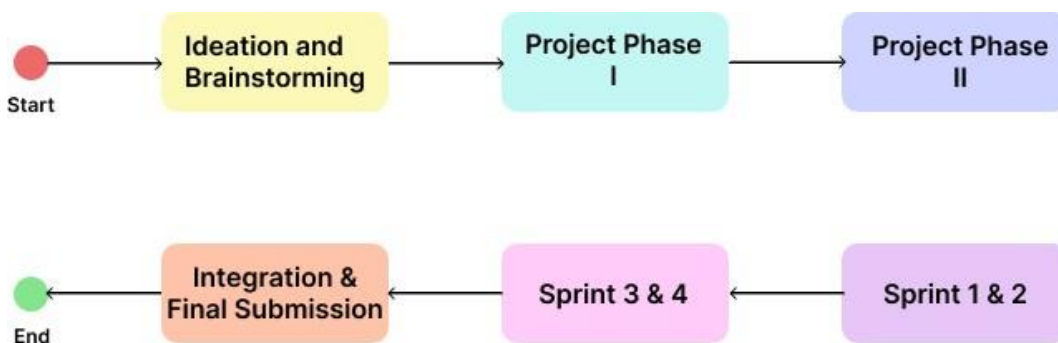
This analysis should also help students who are currently preparing or will be preparing to get a better idea.

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer 1 (Web user)	Landing Page	USN-1	It is pretty clear about why we use this website and how it should be used, just by looking at this page.	Understandable	High	Sprint-1
Customer 2 (Web User)	Landing Page	USN-2	The concept of the application is clear with all the how to do instructions and everything.	Very clear	High	Sprint-1
Customer 3 (Web User)	Marks page	USN-3	Its pretty clear what kind of data should be given in.	I can give my marks details.	Low	Sprint-2
Customer 4 (Web User)	Results Page	USN-4	I can see the right and correct results based on the marks given to the system	Got the details	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation



- The sprints started right after our Training session by IBM.
- Though it started then, we started brainstorming our project since the beginning of this semester's calendar.

- We completed the “Ideation and Brainstorming” phase first, moving on to each other phases one by one.
- Each took a task (exactly what we estimated) to complete.
- First Sprint contains the HTML Code, which acts as a building block for our application.
- Second Sprint is the CSS Code, which enhances the look of the website.
- Third Sprint is the initialization of the flask language and a little bit of backend code. This is where we learned all the important topics needed to complete this project.
- Then the last Sprint, the fourth one, contains only the backend python-flask code, that performs various Data Manipulation and trains the model

6.2 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Frontend - HTML	USN-1	I'm able to see the tables or columns where I can inject my marks into.	2	Medium	Vino S
Sprint-2	Frontend - CSS	USN-2	Now the application looks more appealing and nice to the eyes.	1	Low	Mayakanan L
Sprint-3	Flask	USN-3	I can see that my data is being processed.	2	Medium	Shruthi PG
Sprint-4	Python	USN-4	I can get the results from the inputs I have given to the system.	2	High	Sindhuja

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	17.11.22
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	17.11.22
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	17.11.22
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	17.11.22

7. CODING & SOLUTIONING

7.1 Feature 1

HTML CODE :

```
<!DOCTYPE html>
```

```
<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Login</title>

    <link rel="stylesheet" href="login-style.css">

    <link
href="https://fonts.googleapis.com/css?family=Raleway:700,500,1000&display=sw
ap" rel="stylesheet">

<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js">
</script>


<script>

    $(function() {

        var people = [];

        $.getJSON('
https://api.thingspeak.com/channels/1013258/feeds.json?results=1',
function(data) {

            $.each(data.feeds, function(i, f) {

                var tblRow = "<tr>" + "<td>" + f.created_at + "</td>" + "<td>" +
f.entry_id + "</td>" + "<td>" + f.field8 + "</td>" + "</tr>"

                $(tblRow).appendTo("#userdata tbody");

            });

        });

    });

}
```

```
});

});

</script>

</head>

<body>

    <main>

        <div class="background">

            <div class="text">

                <center style="color:blue">

<h1>PREDICTING COLLEGES</h1>

                </center><br>

            <center>

<form action="https://api.thingspeak.com/update?api_key=QNRW0798ZZV2OEIL&"
method="post" target="_blank">

GRE Score          :<input type="text" name="field1"><br><br>

TOEFL Score        :<input type="text" name="field2"><br><br>

University Rating  :<input type="text" name="field3"><br><br>

SOP                :<input type="text" name="field4"><br><br>

LOR                :<input type="text" name="field5"><br><br>
```

CGPA :<input type="text" name="field6">

Research :<input type="text" name="field7">

<button type="submit">SUBMIT</button>

</form>

<table id= "userdata" border="2">

<thead>

<th>Date</th>

<th>S.no</th>

<th>COLLAGES</th>

</thead>

<tbody>

</tbody>

</table>

```
<iframe id='track' frameborder="0" scrolling="no" width="1" height="1">

</script>

</center>

</div>

</div>

</main>

</body>

</html>
```

CSS CODE:

```
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  text-decoration: none;
}

body{
  font-family: 'Raleway', sans-serif;
  background: #000;
}

.background{
  background: url(background.jpeg) no-repeat;
  background-position: center top;

  background-size: contain;
height:2200px;
position:relative
  display: flex;
}

.text, .box{
  margin-top:0vh;
  flex: 1;
}

.text{
  margin-left: 0%;
  font-weight: 200px;
color:white;
```

```
}

.box{
    margin-right: 25%;
}

.text h1{
    font-size: 70px;
    color: #fff;
    font-weight: 500;
}

.text h2{
    font-size: 70px;
    color: #fff;
    font-weight: 500;
}

.text p{
    font-size: 20px;
    color: #fff;
    font-weight: 300;
}

.text p a{
    color: #fff;
    font-weight: 700;
}

.form{
    background: transparent;
    color: #fff;
    box-sizing: border-box;
    display: flex;
    flex-direction: column;
    width: 250px;
}

input{
    margin: 20px 0;
    padding: 10px;
    background: transparent;
    border: none;
    outline: none;
    color: #fff;
    font-family: 'Raleway', sans-serif;
}

.username, .password{
    border-bottom: 1px solid #fff;
```

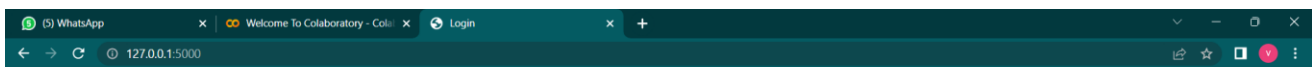
```

}

.button{
    background: transparent;
    border: 1px solid #fff;
    color: #fff;
    font-size: 18px;
}

.button:hover{
    background: #000;
    color: #fff;
}

```



PREDICTING COLLEGES

GRE Score :

TOEFL Score :

University Rating :

SOP :

LOR :

CGPA :

Research :

Date	S.no	COLLAGES
2022-11-19T17:17:12Z	110	null

7.2 Feature 2

college.ipynb

```
import numpy as np
```

```
import pandas as pd
```

```
from sklearn import metrics
```

```
from sklearn.model_selection import train_test_split
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns

import pickle

data = pd.read_csv('data.csv')

data.head()

data.shape

X = data.iloc[:, :-1]

X.head()

y = data.iloc[:, -1]

y.head()

print(X)

print(y)

data['Chance of Admit '].value_counts()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

sns.countplot(x='Chance of Admit ', data=data)

plt.show()

X_train.shape

X_train.head()

y_test.shape

y_test.head()

from sklearn.metrics import accuracy_score

max_accuracy = 0


from sklearn.neighbors import KNeighborsClassifier


for x in range(1, 100):

    model = KNeighborsClassifier(n_neighbors=x)

    model.fit(X_train, y_train)
```



```
y_pred = model.predict(X_test)

current_accuracy = round(accuracy_score(y_pred,y_test)*100,2)

if(current_accuracy>max_accuracy):

    max_accuracy = current_accuracy

    best_x = x
```

```
#print(max_accuracy)
```

```
print(best_x)
```

```
model = KNeighborsClassifier(n_neighbors=best_x)
```

```
model.fit(X_train,y_train)
```

```
y_pred = model.predict(X_test)
```

```
filename = 'knn.sav'
```

```
pickle.dump(model, open(filename, 'wb'))
```

```
acc=(metrics.accuracy_score(y_pred,y_test)*100)
```

```
print("Accuracy is:",acc)
```

```
cm1 = metrics.confusion_matrix(y_pred,y_test)
```

```
total1=sum(sum(cm1))
```

```
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
```

```
print('Sensitivity : ', sensitivity1 )
```

```
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
```

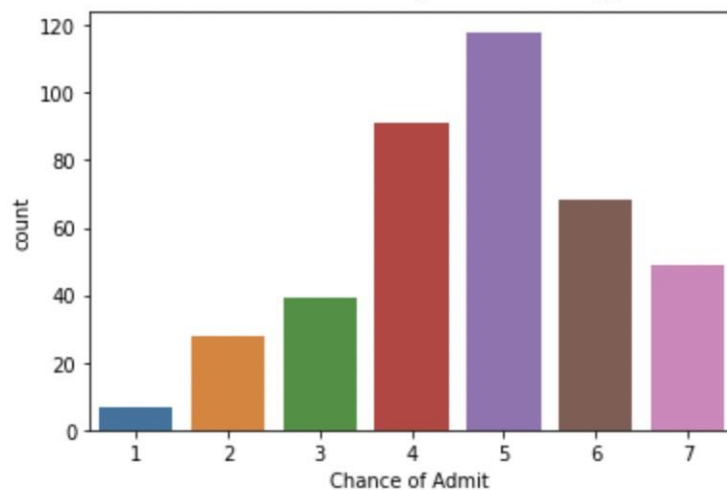
```
print('Specificity : ', specificity1)
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	337	118	4	4.5	4.5	9.65	1
1	324	107	4	4.0	4.5	8.87	1
2	316	104	3	3.0	3.5	8.00	1
3	322	110	3	3.5	2.5	8.67	1
4	314	103	2	2.0	3.0	8.21	0
..
395	324	110	3	3.5	3.5	9.04	1
396	325	107	3	3.0	3.5	9.11	1
397	330	116	4	5.0	4.5	9.45	1
398	312	103	3	3.5	4.0	8.78	0
399	333	117	4	5.0	4.0	9.66	1

[400 rows x 7 columns]

```
0      7
1      5
2      5
3      5
4      4
..
395    6
396    6
397    7
398    4
399    7
```

Name: Chance of Admit , Length: 400, dtype: int64



14

Accuracy is: 53.333333333333336

Sensitivity : nan

Specificity : 1.0

```
import pickle
import urllib.request
import json
from time import sleep
while True:
    conn =
    urllib.request.urlopen("https://api.thingspeak.com/channels/1013258/feeds.json?results=1")
    response = conn.read()
    print ("http status code=%s" % (conn.getcode()))
```

```

data=json.loads(response)
x=int(data['feeds'][0]['entry_id'])
y=x
conn.close()
while x==y:
    conn =
urllib.request.urlopen("https://api.thingspeak.com/channels/1013258/feeds.json?results=1")
    response = conn.read()
    #print ("http status code=%s" % (conn.getcode()))
    data=json.loads(response)
    y=int(data['feeds'][0]['entry_id'])
    conn.close()

```

```

conn =
urllib.request.urlopen("https://api.thingspeak.com/channels/1013258/feeds.json?results=1")
    response = conn.read()
    print ("http status code=%s" % (conn.getcode()))
    data=json.loads(response)
    a=float(data['feeds'][0]['field1'])
    b=float(data['feeds'][0]['field2'])
    c=float(data['feeds'][0]['field3'])
    d=float(data['feeds'][0]['field4'])
    e=float(data['feeds'][0]['field5'])
    f=float(data['feeds'][0]['field6'])
    g=float(data['feeds'][0]['field7'])

```

```

conn.close()
filename = 'knn.sav'
loaded_model = pickle.load(open(filename, 'rb'))
person_reports = [[a,b,c,d,e,f,g]]
predicted = loaded_model.predict(person_reports)
print("ANALYSING... ")
print(predicted[0])
sleep(15)
if predicted[0]==1:
    conn =
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&field8=1-VIT
2-JPR
3-AGNI")
    elif predicted[0]==2:
        conn =
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&field8=1-SREC
2-KEC
3-KPR")
        elif predicted[0]==3:
            conn =
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&field8=1-KONGU
2-KCT
3-HIT")

```

```

elif predicted[0]==4:
    conn =
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&
field8=1-SASTHRA
2-SKCET
3-BIT")
elif predicted[0]==5:
    conn =
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&
field8=1-SRM
2-THIAGARAJAR
3-NIIT")
elif predicted[0]==6:
    conn =
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&
field8=1-PSG
2-CIT
3-GCT")
elif predicted[0]==7:
    conn =
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&
field8=1-IIT
2-MIT
3-ANNA_UNIVERSITY-CHE")

```

```

http status code=200
http status code=200
ANALYSING....
5

```

Integration:

```

from flask import Flask, render_template
app = Flask(__name__)
import os
import subprocess
@app.route('/')
def index():
    return render_template('login.html')

```

```
if __name__ == '__main__':
    app.run(debug=True)
```

8. TESTING

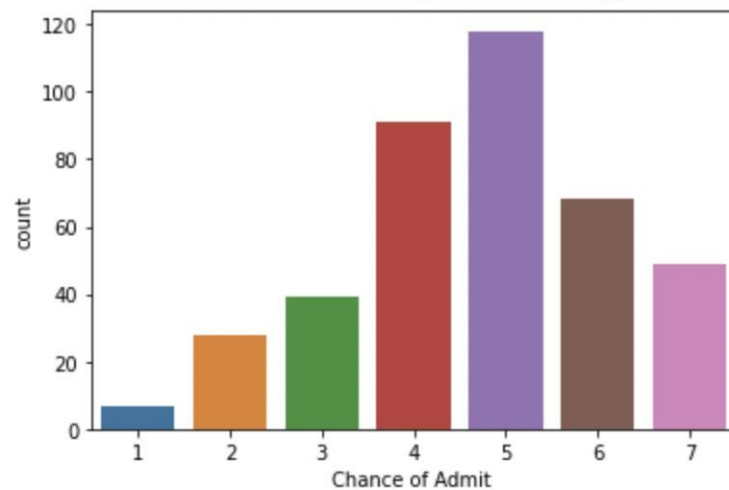
8.1 Test Cases

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	337	118	4	4.5	4.5	9.65	1
1	324	107	4	4.0	4.5	8.87	1
2	316	104	3	3.0	3.5	8.00	1
3	322	110	3	3.5	2.5	8.67	1
4	314	103	2	2.0	3.0	8.21	0
..
395	324	110	3	3.5	3.5	9.04	1
396	325	107	3	3.0	3.5	9.11	1
397	330	116	4	5.0	4.5	9.45	1
398	312	103	3	3.5	4.0	8.78	0
399	333	117	4	5.0	4.0	9.66	1

[400 rows x 7 columns]

```
0      7
1      5
2      5
3      5
4      4
..
395    6
396    6
397    7
398    4
399    7
```

Name: Chance of Admit , Length: 400, dtype: int64



14

Accuracy is: 53.333333333333336

Sensitivity : nan

Specificity : 1.0

```
http status code=200
http status code=200
ANALYSING....
5
```

- All the test cases got passed and the expected output is received as the result.

8.2 User Acceptance Testing

User 1

I can see that the application is quite easy to access and use to see the right Universities for myself

I love how there is no registration or login process, which protects my personal data

The results I got from this software is very close to what I'm eligible for in terms of Universities

User 2

I immediately hoped on to the results after giving my GRE Scores and other related scores

I see that the process is easy and simple and also straight to the point

I got my expected results and I can see that it is quite accurate

9. RESULTS

9.1 Performance Metrics

The data is trained and tested with all three algorithms and out of all KNN gave more accuracy with 90.3 percent and then the KNN with 88.33 percent accuracy. As KNN gave the highest accuracy, all further data predictions are chosen to be followed with KNN. So, finally a web application is made to give the input parameters of the student and the final

prediction is generated and displayed. The background algorithm being used is KNN and the new prediction are keep on adding to the dataset for further more accuracy.

11. CONCLUSION

It has been concluded that the software system that we built is successfully executing our aim. The students that are willing to get into a great college, use this website to get more awareness. All this features given to the users at ease, without collecting any of their personal data except the marks, which we use to train the system, to produce more better and accurate result as we go.

12. FUTURE SCOPE

A powerful web application can be developed where inputs are not given directly instead student parameters are taken by evaluating students through various evaluations and examinations. Technical, analytical, logical, memory based, psychometry and general awareness, interests and skill based tests can be designed and parameters are collected through them so that results will be certainly accurate and the system will be more reliable to use.

Also KNNs have few limitations like overfitting, no pruning, lack of capability to deal with null and missing values and few algorithms have problem with huge number of values. All these can be taken into consideration and even more reliable and more accurate algorithms can be used. Then the project will be more powerful to depend upon and even more efficient to depend upon.

13. APPENDIX

Source Code:

Login.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```
<title>Login</title>

<link rel="stylesheet" href="login-style.css">

<link
href="https://fonts.googleapis.com/css?family=Raleway:700,500,1000&display=sw
ap" rel="stylesheet">

<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js">
</script>


<script>

    $(function() {

var people = [];

    $.getJSON('
https://api.thingspeak.com/channels/1013258/feeds.json?results=1',
function(data) {

    $.each(data.feeds, function(i, f) {

        var tblRow = "<tr>" + "<td>" + f.created_at + "</td>" + "<td>" +
f.entry_id + "</td>" + "<td>" + f.field8 + "</td>" + "</tr>"

        $(tblRow).appendTo("#userdata tbody");

    });

});

});

});

</script>
```



```
</head>

<body>

    <main>

        <div class="background">

            <div class="text">

                <center style="color:blue">

<h1>PREDICTING COLLEGES</h1>


                </center><br>

            <center>

<form action="https://api.thingspeak.com/update?api_key=QNRW0798ZZV2OEIL&"
method="post" target="_blank">

GRE Score          :<input type="text" name="field1"><br><br>


TOEFL Score        :<input type="text" name="field2"><br><br>


University Rating  :<input type="text" name="field3"><br><br>


SOP                :<input type="text" name="field4"><br><br>


LOR                :<input type="text" name="field5"><br><br>


CGPA               :<input type="text" name="field6"><br><br>


Research           :<input type="text" name="field7"><br><br>


<button type="submit">SUBMIT</button>
```

```
</form>
```

```
<br>
```

```
<br>
```

```
<br>
```

```
<br>
```

```
<br>
```

```
<br>
```

```
<br>
```

```
<br>
```

```
<br>
```

```
<table id= "userdata" border="2">
```

```
  <thead>
```

```
    <th>Date</th>
```

```
    <th>S.no</th>
```

```
    <th>COLLAGES</th>
```

```
  </thead>
```

```
  <tbody>
```

```
  </tbody>
```

```
</table>
```

```
<iframe id='track' frameborder="0" scrolling="no" width="1" height="1">
```

```
</script>
```

```
</center>
```

```
</div>
```

```
        </div>

    </main>

</body>

</html>
```

Login-style.css:

```
*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    text-decoration: none;

}

body{

    font-family: 'Raleway', sans-serif;

    background: #000;

}

.background{

    background: url(background.jpeg) no-repeat;

    background-position: center top;

    background-size: contain;

height:2200px;

position:relative

    display: flex;

}
```

```
.text, .box{  
    margin-top:0vh;  
    flex: 1;  
}
```

```
.text{  
    margin-left: 0%;  
    font-weight: 200px;  
color:white;  
}
```

```
.box{  
    margin-right: 25%;  
}
```

```
.text h1{  
    font-size: 70px;  
    color: #fff;  
    font-weight: 500;  
}
```

```
.text h2{  
    font-size: 70px;  
    color: #fff;  
    font-weight: 500;  
}
```

```
.text p{  
    font-size: 20px;  
    color: #fff;  
    font-weight: 300;  
}
```

```
.text p a{  
    color: #fff;  
    font-weight: 700;  
}
```

```
.form{  
    background: transparent;  
    color: #fff;  
    box-sizing: border-box;  
    display: flex;  
    flex-direction: column;  
    width: 250px;  
}
```

```
input{  
    margin: 20px 0;  
    padding: 10px;  
    background: transparent;  
    border: none;
```

```
    outline: none;

    color: #fff;

    font-family: 'Raleway', sans-serif;
}
```

```
.username, .password{

    border-bottom: 1px solid #fff;
}
```

```
.button{

    background: transparent;

    border: 1px solid #fff;

    color: #fff;

    font-size: 18px;
}
```

```
.button:hover{

    background: #000;

    color: #fff;
}
```

Server.py

```
from flask import Flask, render_template

app = Flask(__name__)

import os

import subprocess

@app.route(r'/')

def index():
```

```
return render_template('login.html')
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True)
```

College.ipynb

(i)

```
import numpy as np
```

```
import pandas as pd
```

```
from sklearn import metrics
```

```
from sklearn.model_selection import train_test_split
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
import pickle
```

```
data = pd.read_csv('data.csv')
```

```
data.head()
```

```
data.shape
```

```
X = data.iloc[:, :-1]
```

```
X.head()
```

```
y = data.iloc[:, -1]
```

```
y.head()
```

```
print(X)
```

```
print(y)
```

```
data['Chance of Admit '].value_counts()
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

```
sns.countplot(x='Chance of Admit ', data=data)
```

```
plt.show()
```

```
X_train.shape
```

```
X_train.head()
```

```
y_test.shape
```

```
y_test.head()
```

```
from sklearn.metrics import accuracy_score
```

```
max_accuracy = 0
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
for x in range(1,100):
```

```
    model = KNeighborsClassifier(n_neighbors=x)
```

```
    model.fit(X_train,y_train)
```

```
    y_pred = model.predict(X_test)
```

```
    current_accuracy = round(accuracy_score(y_pred,y_test)*100,2)
```

```
    if(current_accuracy>max_accuracy):
```

```
        max_accuracy = current_accuracy
```

```
        best_x = x
```

```
#print(max_accuracy)
```

```
print(best_x)
```

```
model = KNeighborsClassifier(n_neighbors=best_x)
```

```
model.fit(X_train,y_train)
```

```
y_pred = model.predict(X_test)
```

```
filename = 'knn.sav'
```

```
pickle.dump(model, open(filename, 'wb'))
```

```
acc=(metrics.accuracy_score(y_pred,y_test)*100)
```

```
print("Accuracy is:",acc)
```



```
cm1 = metrics.confusion_matrix(y_pred,y_test)
```

```
total1=sum(sum(cm1))
```

```
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
```

```
print('Sensitivity : ', sensitivity1 )
```

```
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
```

```
print('Specificity : ', specificity1)
```

(ii)

```
import pickle
```

```
import urllib.request
```

```
import json
```

```
from time import sleep
```

```
while True:
```

```
    conn = urllib.request.urlopen("https://api.thingspeak.com/channels/1013258/feeds.json?results=1")
```

```
    response = conn.read()
```

```
    print ("http status code=%s" % (conn.getcode()))
```

```
    data=json.loads(response)
```

```
    x=int(data['feeds'][0]['entry_id'])
```

```
    y=x
```

```
    conn.close()
```

```
    while x==y:
```

```
        conn = urllib.request.urlopen("https://api.thingspeak.com/channels/1013258/feeds.json?results=1")
```

```
        response = conn.read()
```

```
        #print ("http status code=%s" % (conn.getcode()))
```

```
        data=json.loads(response)
```

```

y=int(data['feeds'][0]['entry_id'])

conn.close()


conn = urllib.request.urlopen("https://api.thingspeak.com/channels/1013258/feeds.json?results=1")

response = conn.read()

print ("http status code=%s" % (conn.getcode()))

data=json.loads(response)

a=float(data['feeds'][0]['field1'])

b=float(data['feeds'][0]['field2'])

c=float(data['feeds'][0]['field3'])

d=float(data['feeds'][0]['field4'])

e=float(data['feeds'][0]['field5'])

f=float(data['feeds'][0]['field6'])

g=float(data['feeds'][0]['field7'])


conn.close()

filename = 'knn.sav'

loaded_model = pickle.load(open(filename, 'rb'))

person_reports = [[a,b,c,d,e,f,g]]

predicted = loaded_model.predict(person_reports)

print("ANALYSING .. ")

print(predicted[0])

sleep(15)

if predicted[0]==1:

    conn =
    urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&field8=1-VIT

2-JPR

3-AGNI")

elif predicted[0]==2:

```

```
conn =  
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&field8=1-SREC
```

```
2-KEC
```

```
3-KPR")
```

```
elif predicted[0]==3:
```

```
conn =  
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&field8=1-KONGU
```

```
2-KCT
```

```
3-HIT")
```

```
elif predicted[0]==4:
```

```
conn =  
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&field8=1-SASTHR  
A
```

```
2-SKCET
```

```
3-BIT")
```

```
elif predicted[0]==5:
```

```
conn =  
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&field8=1-SRM
```

```
2-THIAGARAJAR
```

```
3-NIIT")
```

```
elif predicted[0]==6:
```

```
conn =  
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&field8=1-PSG
```

```
2-CIT
```

```
3-GCT")
```

```
elif predicted[0]==7:
```

```
conn =  
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=QNRWO798ZZV2OEIL&field8=1-IIT
```

```
2-MIT
```

```
3-ANNA_UNIVERSITY-CHE")
```

GitHub & Project Demo Link

Github: <https://github.com/IBM-EPBL/IBM-Project-9130-1658982501>

Project Demo Link: https://youtu.be/J_mIJmjV6cY