UNIVERSITY ADMIT ELIGIBILITY PREDICTOR

TEAM ID: PNT2022TMID50761

1.INTRODUCTION

1.1 Project Overview

University Admit Eligibility Predictor System is a web based application in which students can register with their personal as well as marks details for prediction the admission in colleges and the administrator can allot the seats for the students. Administrator can add the college details and the batch details. The predicted output gives them a fair idea about their admission chances in a particular university. This analysis should also help students who are currently preparing or will be preparing to get a better idea.

1.2 Purpose

A person's education plays a vital role in their life. While playing for education students often have several questions regarding the courses, universities, job opportunities, expenses involved, etc., Securing admission in their dream university is one of their main concerns. It is seen that often students prefer to pursue their education from universities which have global recognition.

2. LITERATURE SURVEY

A good literature review can ensure that a proper research question has been asked and a proper theoretical framework and research methodology have been chosen. To be precise, a literature review serves to situate the current study within the body of the relevant literature and to provide context of the reader. In such case, the review usually precedes the work.

Sl.No	Tittle	Author &Publication	Year	Description
1	College Admission Prediction using Ensemb Machine Learning Model	Vandit Manish Jain, Rih Satia & *Internatic Research Journal Engineering Technology(IRJET).*	2021	This paper aims to build a mode that can help students to pick th right Universities based on their profiles by using Machine Learning, Linear Regression algorithm, Decision Trees algorithm, Random Forests algorithm

2	Prediction for University Admission using Machin Learning	Chithra Apoorva D A, Malepati ChanduNath, Pe Rohith, Bindu Shree.S, Swaroop.S & *Internation Journal of Recent Technology and Engineering(IJRTE).* Prediction for University Admission using Machin Learning	2020	For this work, several machine learning algorithms have been used, K- Nearest Neighbor and Linear Regression, Random Fo are used. Students can use the model to assess their chances of getting admission into a particul university with an average accuracy of 79 percentage.
3	College Admission Predictor	Annam Mallikharjuna Rc Nagineni Dharani, A. Sat Raghava, J. Buvanambig K. Sathish & *Journal of Network Communication and Emerging Technolo (JNCET).*	2018	In this paper the overall objectiv in the development of database technology has been to treat dat an organizational resource and a integrated whole.
4	COLLEGE RECOMMENDATION SYSTEM FOR ADMISSION	Miss. Deokate monali, M Gholave Dhanashri, Miss Jarad Dipali, Miss. Khomane Tejaswini, Gui by: Prof. Nale R. K & *International Research Journal of Engineering a Technology (IRJET).*	2018	In this system it use different algorithms, in that we are going add semantic analysis algorithm which will capture the positive a negative comments. Naïve baye and adaboost algorithm rating parameter to stream college.

2.1 Existing Problem

Today in college's student details are entered manually. The student details in separate records are tedious task. Referring to all these records and updating is needed. There is a chance for more manual errors. When the student is visiting the college for admission, first he/she gets admission form from reception. Fill the form and submit it into office. Filled form is checked with documents like merit lists an details received from university and verified by an office person. If there is any mistake then it will be corrected. Then the admission number is assigned to the candidate by the institute. At the time of submission of the form, fees will be paid by the candidate. Candidates will get the receipt for the deposited fees.

2.2 References

[1] Vandit Manish Jain, Rihaan Satia, "College Admission Prediction using Ensemble Machine Learning Models", *International Research Journal of Engineering and Technology*, Vol. 08, Issue: 12 | dec2021, pp. 403-407.

[2] Chithra Apoorva D A, Malepati ChanduNath, Peta Rohith, Bindu Shree.S, Swaroop.S, "Prediction for University Admission using Machine Learning", *International Journal of Recent Technology and Engineering,* Vol. 8, Issue-6, March 2020, pp. 4922-4926.

[3] Annam Mallikharjuna Roa, Nagineni Dharani, A. Satya Raghava, J. Buvanambigai, K. Sathish, "College Admission Predictor", *Journal of Network Communications and Emerging Technologies(JNCET),* vol. 8, Issue 4, April(2018), pp. 142-147.

[4] Miss. Deokate monali, Miss. Gholave Dhanashri, Miss. Jarad Dipali, Miss. Khomane Tejaswini, Guided by: Prof. Nale R.K, "COLLEGE RECOMMENDATATION SYSTEM FOR ADMISSION", *International Research Journal of Engineering and Technology (IRJET),* vol. 05, Issue: 03 | March-2018, pp. 1269-1271.

2.3 Problem Statement Definition

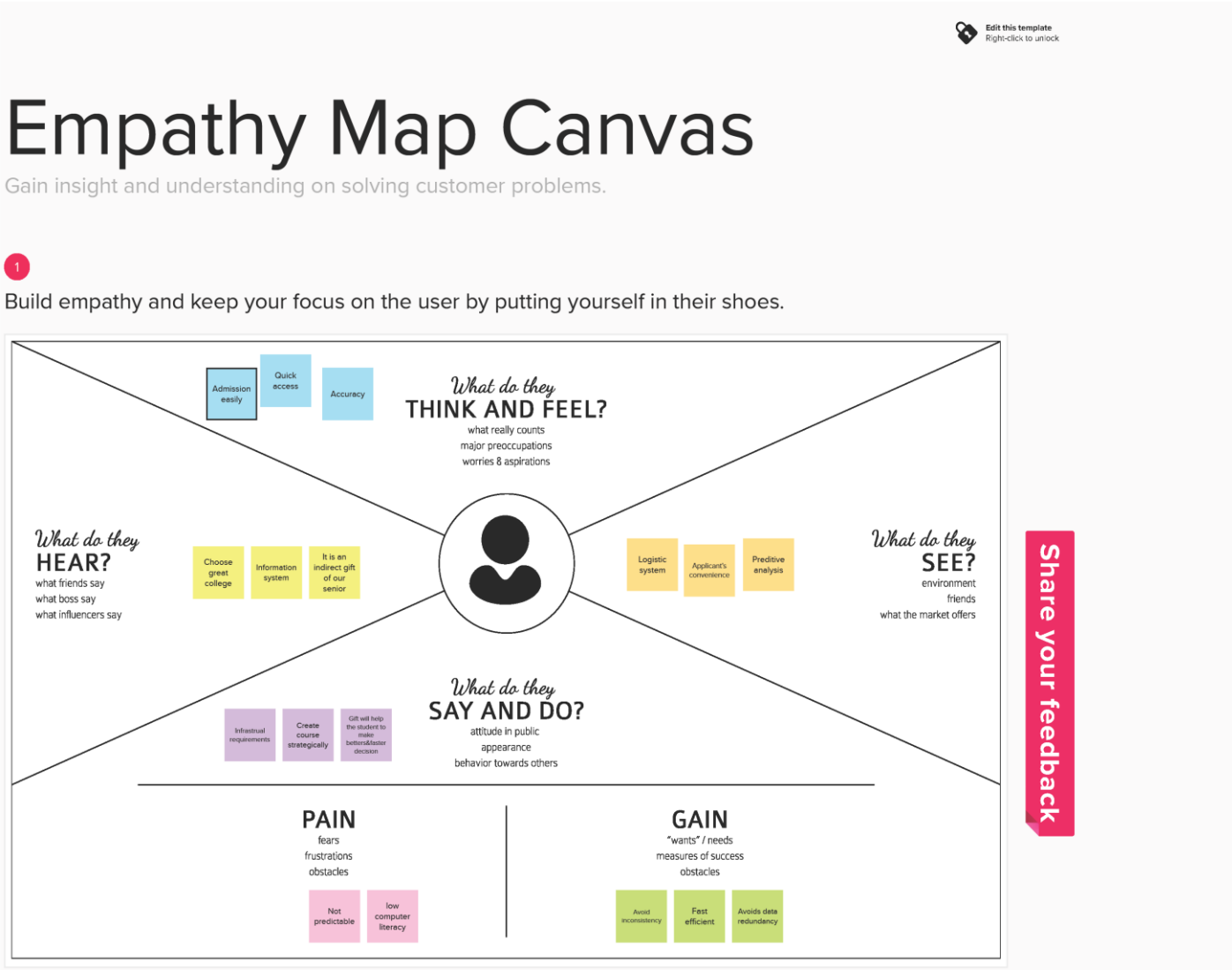
Students are often worried about their chances of admission to university which leads to frustration and anxiety. The aim of this project is to help students in shortlisting universities with their profiles

Build an application that predicts the university admission chances of a student powered by machine learning models. Train the model and host it on IBM cloud. The majority of international students studying in the USA are from India and China. In the past decade, India has seen a huge increase in the number of students opting to pursue their education from foreign universities in countries like The USA, Ireland, Australia, Germany, etc. Although there are significant universities and colleges in India, students are finding it difficult to get admission in the highly ranked colleges and also getting a job is a challenge as the ratio of number students to the number work opportunities available is quite high. India is one of the leading counties in the number of software engineers produced each year; it becomes tough for the students to find jobs in elite companies due to high competition. This motivates a good number of students to pursue post-graduation in their field. It is seen that the number of students pursuing Masters in Computer Science field from universities in the USA is quite high; the focus of this research will be on these students.

3. IDEATION & PROPOSED SOLUTION

The project aims to develop an application and the application also uses IBM cloud storage for storing objects. An application that predicts the university admission chances of a student powered by machine learning models. Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions.

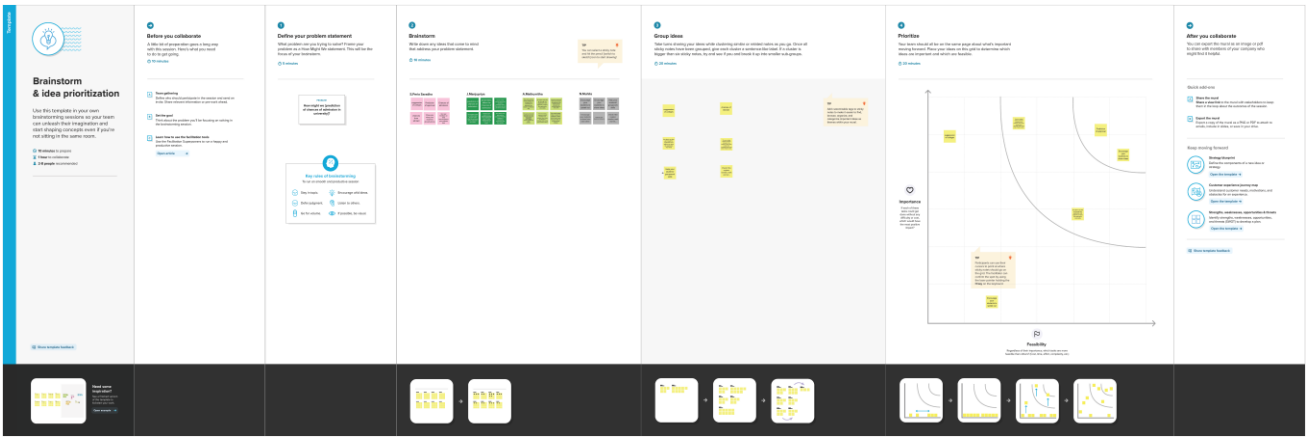
3.1 Empathy Map Canvas



3.2 Ideation &Brainstorming

This task of shortlisting the universities where the student has high chances of admission is difficult for mainly for the international students, so they end up with applying to many universities in hopes of getting admission in few of them thus investing an extra amount of money in the applications. There are several portals and websites which provide information and help to students in shortlisting the universities, but they are not reliable. Most of the students don't take the risk of evaluating the colleges by themselves, and they seek the help of the education consultancy firms to do it for them. Again for this students have to pay a huge amount of

fee to the education consultant.



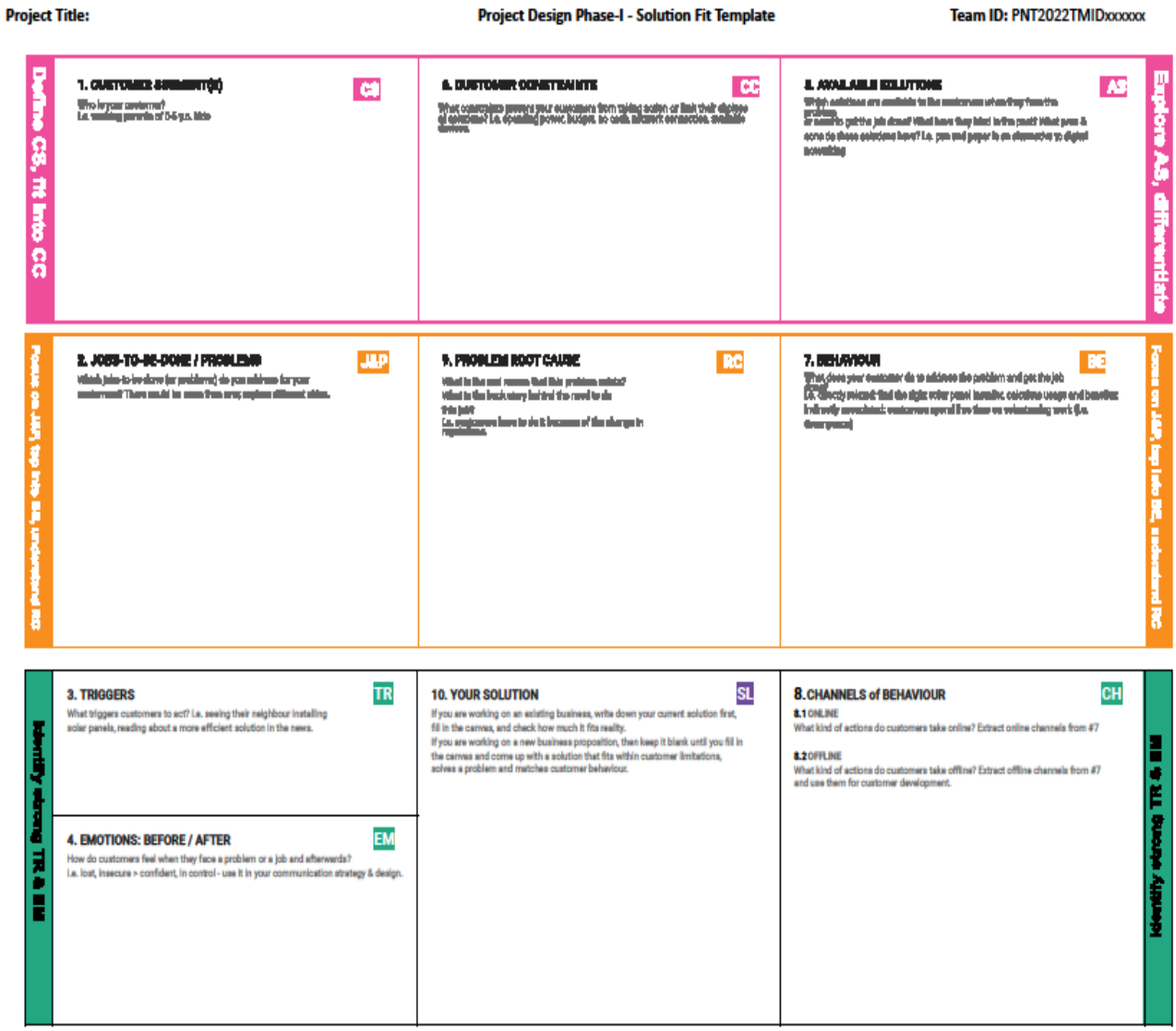
3.3 Proposed Solution

The main goals of the system is to automate the process carried out in the organization with improved performance and realize the vision of paperless admission.

Finally, Built HTML code and K Nearest Neighbours and Decision Tree algorithms were used as they were found to be the best fit for the system developed. Also, we will be creating a simple user interface which will help the users to input the data related to student profile and get the predicted result for the application based on the profile as output. This research will thus eventually help students saving the extra amount of time and money they have to spend at the education consultancy firms. And also it will help them to limit their number of application to a small number by proving them the suggestion of the universities where they have the best chance of securing admission thus saving more money on the application fees.

3.4 Problem Solution Fit

We will be developing a University Admit Eligibility Predictor system which will help the students to predict the chances of their application being selected for a particular university for which they wish to apply based on their profile.



4. REQUIREMENT ANALYSIS

Requirements analysis is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

4.1 Functional requirement

- Prediction
- Input form
- Percent of Chance

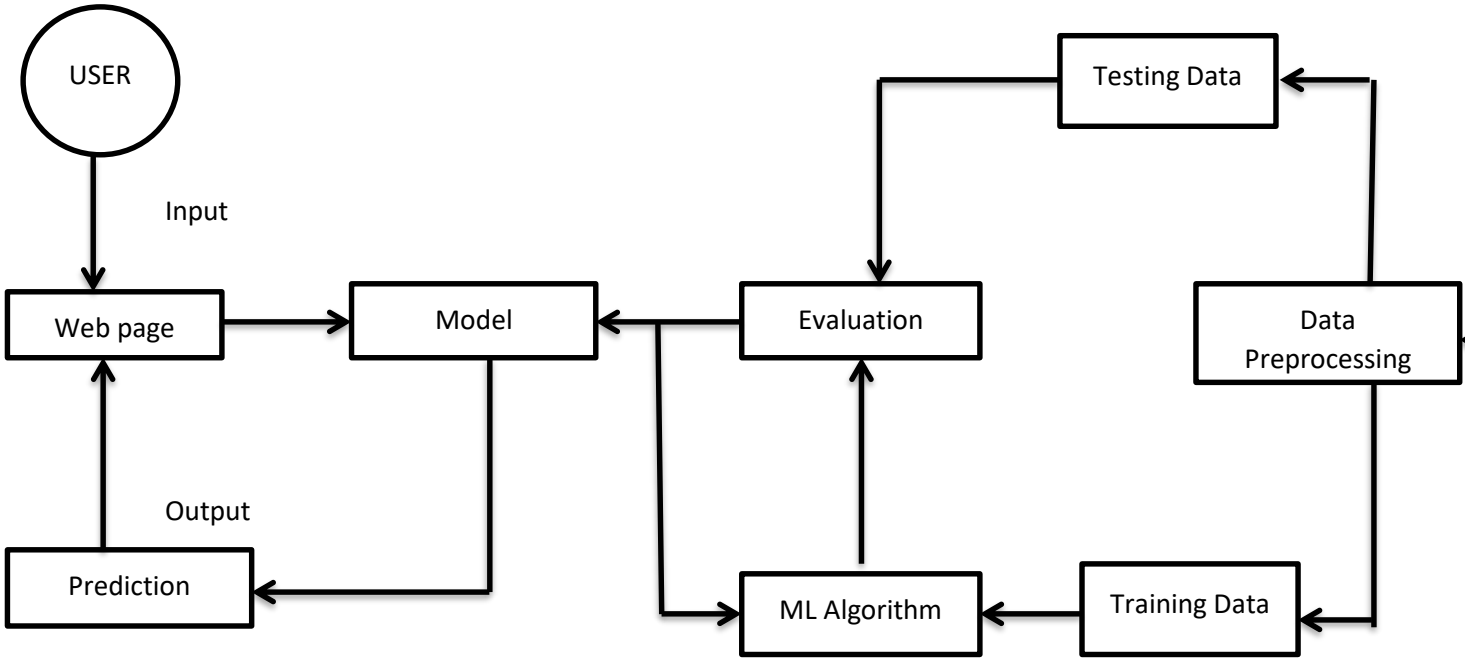
4.2 Non-Functional requirements

- Speed
- Security
- Portability
- Compatibility
- Capacity
- Reliability
- Environment
- Localization

5. Project Design

5.1 Data Flo **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.



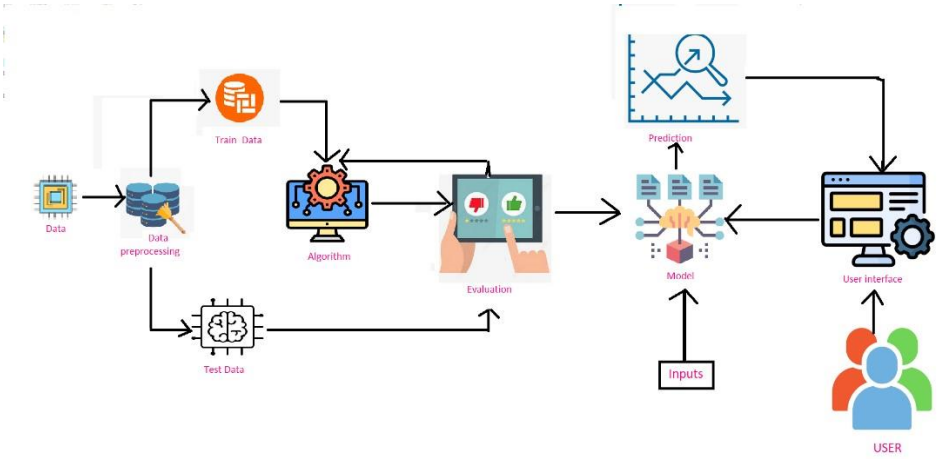
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a student, I can enter the name, email, mobile number ,personal details	I can access my account / dashboard	High	Sprint1
Customer(Web user)		USN-2	As a student, I can enter using name, email and mobile number	I can receive confirmation email & click confirm	High	Sprint1
	Evaluation Page	USN-3	Enter the GCPA score, SOP or LOR for validating. As an administrator, I can upgrade or update the application.	I can register & access the dashboard with email Login	Low	Sprint2
Administration		USN-4	As a user, I can register for the application through Gmail	To Find your eligible	Medium	Sprint1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can log into the application by entering email & password	High	Sprint1
	Prediction	USN-6	I can see the eligibility of the particular university which i like to join	with the help of dataset it shows the predicted resul	High	Sprint-3

5.2 Solution &Technical Architecture

Solution Architecture

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.



: Architecture and data flow of the voice patient diary sample application

5.3 User Stories

A user story is an informal, natural language description of features of a software system. They are written from the perspective of an end user or user of a system, and may be recorded on index cards, Post-it notes, or digitally in project management software. Depending on the project, user stories may be written by different stakeholders like client, user, manager, or development team

6. Project Planning & Estimation

6.1 Sprint & Planning Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Pre-process the data	USN-1	Collect &download the dataset.	2	High	S.Peria Saradha
Sprint-1		USN-2	Import required libraries.	1	High	N.Muhila
Sprint-1		USN-3	Read and clean dataset.	2	Low	Manjupriya
Sprint-2	Model building	USN-1	Split the data into independent and dependent variables.	2	Medium	Mathumitha
Sprint-2		USN-2	Build the regression model.	1	High	S.Peria Saradha, Muhila, Manjupriya
Sprint-3	Application Building	USN-1	Build the Python application.	2	Medium	Muhila, Mathumitha, Manjupriya
Sprint-3		USN-2	Test the Application model.	3	High	Manjupriya, Mathumitha, Peria Saradha

Sprint-4	Train the model	USN-1	Train the model	3	High	N.Muhila,

Sprint	Total Stor Points	Duration	Sprint End 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
Sprint 2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint 3	20	6 Days	7 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint 4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

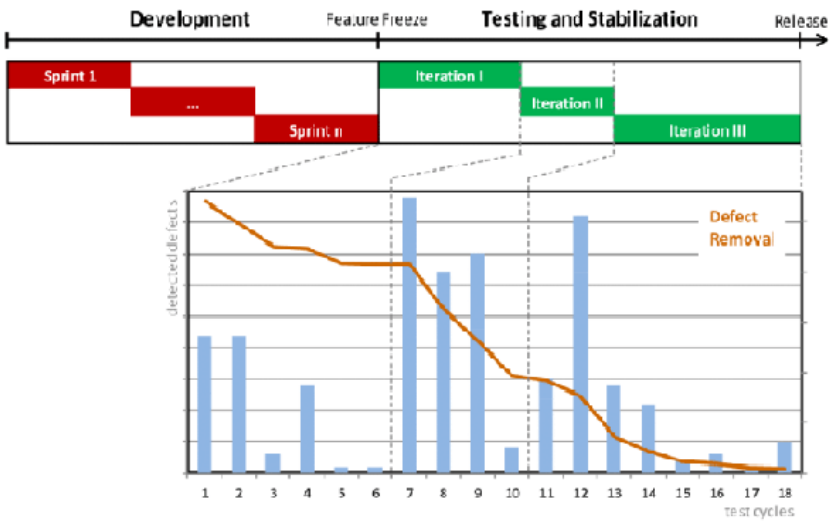
AV = Sprint duration/Velocity = 20/10 = 2

Sprint Delivery Schedule

TITLE	DESCRIPTION	DATE
Prepare Empathy map	Prepare Empathy Map to get knowledge about users emotions related to this problem. Prepare the list of problem faced by user.	27 August 2022
Ideation	List the ideas to address the problem and clusterize and priorities the similar ideas using brainstorming canvas.	12 October 2022
Literature survey & Information gathering	Prepare literature by gathering information about the project by referring the technical paper and research publications	5 November 2022
Proposed solution	Prepare the proposed solution document, which includes novelty, feasibility, scalability of solution.	4 November 2022

Problem solution fit	Prepare problem solution fit document to get idea about solution behaviour and available solution	6 November 2022
Solution Architecture	Prepare solution architecture diagram to get idea of the solution in design format.	4 November 2022

6.3 Reports From JIRA



7.CODING & SOLUTIONING

7.1 Feature 1 –HTML & FLASK APP

The following is the flask app code and working

```
<!DOCTYPE html>
<html>
<head>
  <SCRIPT language=Javascript>

    function check(e, value) {
      //Check Charater
      var unicode = e.charCode ? e.charCode : e.keyCode;
      if (value.indexOf(".") != -1)
        if (unicode == 46) return false;
      if (unicode != 8)
        if ((unicode < 48 || unicode > 57) && unicode != 46) return false;
    }

  </SCRIPT>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<style>
body {
  font-family: Arial, Helvetica, sans-serif;
}

* {
  box-sizing: border-box;
}

/* style the container */
.container {
  position: relative;
  border-radius: 5px;
  background-color: #C7E3E1;
  padding: 20px 0 30px 0;
}

/* style inputs and link buttons */
input,
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<style>
body {
  font-family: Arial, Helvetica, sans-serif;
}

* {
  box-sizing: border-box;
}

/* style the container */
.container {
  position: relative;
  border-radius: 5px;
  background-color: #C7E3E1;
  padding: 20px 0 30px 0;
}

/* style inputs and link buttons */
input,
.btn {
  width: 100%;
  padding: 12px;
  border: none;
  border-radius: 4px;
  margin: 5px 0;
  opacity: 0.85;
  display: inline-block;
  font-size: 17px;
  line-height: 20px;
  text-decoration: none; /* remove underline from anchors */
}

input:hover,
.btn:hover {
```

```
input:hover,
.btn:hover {
  opacity: 1;
}


/* style the submit button */
input[type=submit] {
  background-color: #65d4b9;
  color: rgb(240, 240, 240);
  cursor: pointer;
}

input[type=submit]:hover {
  background-color: hsl(120, 71%, 45%);
}

/* Two-column layout */
.col {
  float: left;
  width: 50%;
  margin: auto;
  padding: 0 50px;
  margin-top: 6px;
}

/* Clear floats after the columns */
.row:after {
  content: "";
  display: table;
  clear: both;
}

/* vertical line */
.vl {
```

```
/* vertical line */
.vl {
  position: absolute;
  left: 50%;
  transform: translate(-50%);
  border: 2px solid rgb(32, 31, 31);
  height: 490px;
}

/* text inside the vertical line
.vl-innertext {
  position: absolute;
  top: 50%;
  transform: translate(-50%, -50%);
  background-color: #f1f1f1;
  border: 1px solid #ccc;
  border-radius: 50%;
  padding: 8px 10px;
} */

/* hide some text on medium and large screens */
.hide-md-lg {
  display: none;
}

/* bottom container */
.bottom-container {
  text-align: center;
  background-color: #ffffff;
  border-radius: 0px 0px 4px 4px;
  border-radius: 15px;
  margin: 15px;
  padding: 10px;
}

/* Responsive layout - when the screen is less than 650px wide, make the two columns stack on top of each other instead of next to each other */
```

```
/* Responsive layout - when the screen is less than 650px wide, make the two columns stack on top of each other instead of next to each other */
@media screen and (max-width: 650px) {
  .col {
    width: 100%;
    margin-top: 0;
  }
  /* hide the vertical line */
  .vl {
    display: none;
  }
  /* show the hidden text on small screens */
  .hide-md-lg {
    display: block;
    text-align: center;
  }
}
</style>
</head>
<body>

<div class="container">
  <form action="{{url_for('predict')}}" method="post">
    <div class="row">
      <marquee class="bottom-container"><h2>University Admit Eligibility Predictor!</h2></marquee>
      <div class="vl">
        <span class="vl-innertext"></span>
      </div>

      <div class="col">
          <p><b>A Simple Web App to Predict The Chances of Getting Admission in Universities Based on Student's Profile</p></b>
          <h3>Input Guide</h3>
      <ul>
                        <li>GRE Score (out of 340)</li><br>
                        <li>TOEFL Score (out of 120)</li><br>
                        <li>University Rating (out of 5) - the category of the target university</li><br>
```

```
                    <li>TOEFL Score (out of 120)</li><br>
                    <li>University Rating (out of 5) - the category of the target university</li><br>
                    <li>Statment of Purpose {SOP} Strength (out of 5)</li><br>
                    <li>Letter of Recommendation {LOP} Strength (out of 5)</li><br>
                    <li>Undergraduate CGPA (out of 10)</li><br>
                    <li>Research Experience (0 for NONE and 1 for YES)</li>

        </ul>
    </div>

    <div class="col">
      <div class="hide-md-lg">
      </div>

                    <input type="number" name="GRE Score" placeholder="GRE Score" required="required" min="0" max="340"/>
                    <input type="number" name="TOEFL Score" placeholder="TOEFL Score" required="required" min="0" max="120"/>
                    <input type="number" name="University Rating" placeholder="University Rating" required="required" min="1" max="5"/>
                    <input type="number" name="SOP" placeholder="SOP" required="required" onkeypress="return check(event,value)" step="0.1" min="1" max="5"/>
                    <input type="number" name="LOR" placeholder="LOR" required="required" onkeypress="return check(event,value)" step="0.1" min="1" max="5"/>
                    <input type="number" name="CGPA" placeholder="CGPA" required="required" onkeypress="return check(event,value)" step="0.01" min="1" max="10"/>
                    <input type="number" name="Research" placeholder="Research" required="required" min="0" max="1"/>

                    <input type="submit" value="Predict"></input>

                    <h4 style="text-align: center;">{{prediction_text}}</h4>
    </div>

   </div>
  </form>

</div>


</body>
</html>
```

7.2 Feature 2

Develop the application

File Edit Format View Help

```
# Core Packages

import pandas as pd

import seaborn as sns

import streamlit as st

from PIL import Image

import requests

import matplotlib

matplotlib.use('Agg')



# EDA Packages

st.set_page_config(page_title='GAD Analysis', page_icon='images/logo.png',

                   layout='wide', initial_sidebar_state='auto')

sns.set(rc={'figure.figsize': (20, 15)})



DATA_URL = ('dataset/gad.csv')



st.markdown('# Graduate Admission Dataset')
```

File Edit Format View Help

```
img = Image.open('images/gad.png')

st.image(img, width=720, caption='Graduate Admission Dataset')



st.markdown('### **About the Dataset:**')

st.info('This dataset was built \

    with the purpose of helping students in \

        shortlisting universities with their profiles. \

            The predicted output gives them a fair \

                idea about their chances for a particular university. \

                    This dataset is inspired by the UCLA Graduate Dataset from Kaggle. \

                        The graduate studies dataset is a dataset which describes the probability of \

                            selections for Indian students dependent on the following parameters below.')



img = Image.open('images/univ.png')

st.image(img, width=720, caption="Top 5 Universities in the US")
```

*sprint 4 - Notepad

File Edit Format View Help

```
st.subheader('Raw data')

    st.write(df)



if st.sidebar.checkbox('Dataset Quick Look'):

    st.subheader('Dataset Quick Look:')

    st.write(df.head())



if st.sidebar.checkbox("Show Columns"):

    st.subheader('Show Columns List')

    all_columns = df.columns.to_list()

    st.write(all_columns)



if st.sidebar.checkbox('Statistical Description'):

    st.subheader('Statistical Data Descripition')

    st.write(df.describe())
```

Ln 141, Col 3 | 100% | Macintosh (CR) | UTF-8

Type here to search | 25°C Haze | ENG | 06:30 PM 19-11-2022

File Edit Format View Help

```
def load_data(nrows):

    df = pd.read_csv(DATA_URL, nrows=nrows)

    def lowercase(x): return str(x).lower()

    df.set_index('Serial No.', inplace=True)

    df.rename(lowercase, axis='columns', inplace=True)

    return df

st.title('Lets explore the Graduate Admission Dataset')

# Creating a text element and let the reader know the data is loading.
data_load_state = st.text('Loading graduate admissions dataset...')

# Loading 500 rows of data into the dataframe.

df = load_data(500)
# Notifying the reader that the data was successfully loaded.

data_load_state.text('Loading graduate admissions dataset...Completed!')

# Explore Dataset

st.header('Quick  Explore')

st.sidebar.subheader('Quick  Explore')

st.markdown("Tick the box on the side panel to explore the dataset.")

if st.sidebar.checkbox("Show Raw Data"):

    st.subheader('Raw data')
```

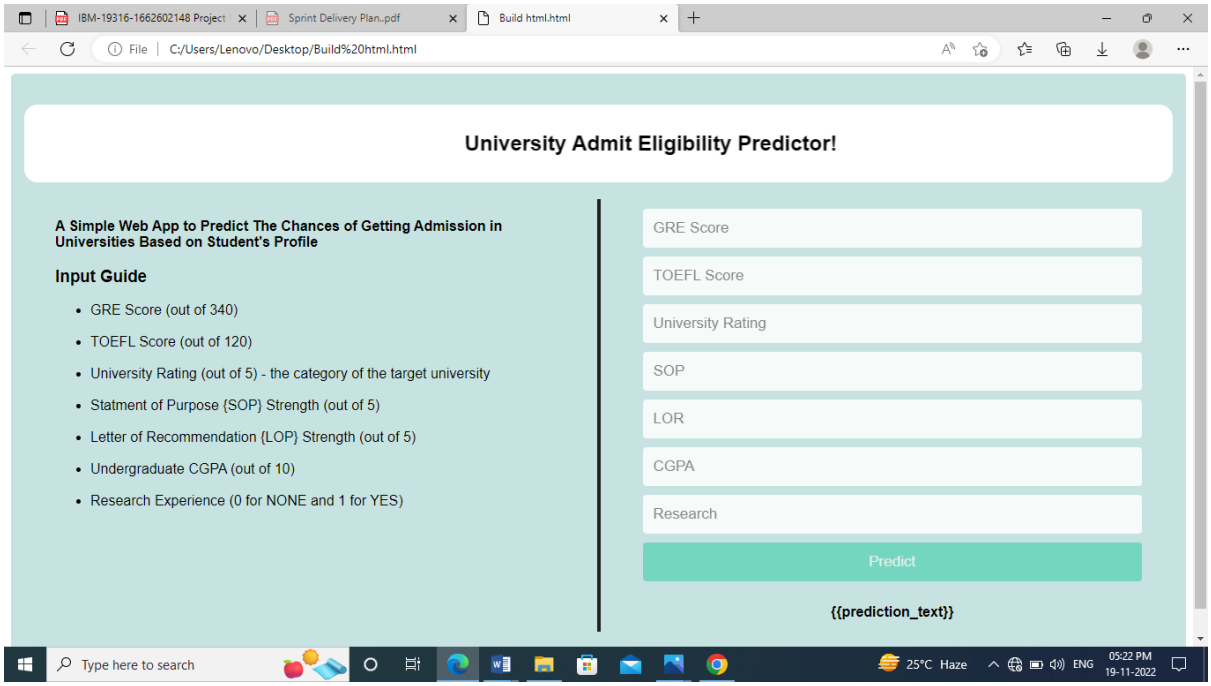
```
    st.write(df)
if st.sidebar.checkbox('Dataset Quick Look'):

    st.subheader('Dataset Quick Look:')

    st.write(df.head())
if st.sidebar.checkbox("Show Columns"):
    st.subheader('Show Columns List')
    all_columns = df.columns.to_list()
    st.write(all_columns)
if st.sidebar.checkbox('Statistical Description'):
    st.subheader('Statistical Data Descripition')
    st.write(df.describe())
if st.sidebar.checkbox('Missing Values?'):
    st.subheader('Missing values')
    st.write(df.isnull().sum())
st.header('Data Visualization')
st.markdown("Tick the box on the side panel to create your own Visualization.")
st.sidebar.subheader('Data Visualization')
if st.sidebar.checkbox('Count Plot'):
    st.subheader('Count Plot')
    st.info("If error, please adjust column name on side panel.")
    column_count_plot = st.sidebar.selectbox(
        "Choose a column to plot count.", df.columns[:5])
    fig = sns.countplot(x=column_count_plot, data=df)
    st.set_option('deprecation.showPyplotGlobalUse', False)
    st.pyplot()
if st.sidebar.checkbox('Distribution Plot'):
    st.subheader('Distribution Plot')
    st.info("If error, please adjust column name on side panel.")
column_dist_plot = st.sidebar.selectbox(
        'Choose a column to plot density.', df.columns[:5])
    fig = sns.distplot(df[column_dist_plot])
    st.set_option('deprecation.showPyplotGlobalUse', False)
    st.pyplot()
```

```
    def lowercase(x): return str(x).lower()

    df.set_index('Serial No.', inplace=True)

    df.rename(lowercase, axis='columns', inplace=True)

    return df

st.title('Lets explore the Graduate Admission Dataset')

# Creating a text element and let the reader know the data is loading.
data_load_state = st.text('Loading graduate admissions dataset...')

# Loading 500 rows of data into the dataframe.

df = load_data(500)
# Notifying the reader that the data was successfully loaded.

data_load_state.text('Loading graduate admissions dataset...Completed!')

# Explore Dataset

st.header('Quick  Explore')

st.sidebar.subheader('Quick  Explore')

st.markdown("Tick the box on the side panel to explore the dataset.")

if st.sidebar.checkbox("Show Raw Data"):

    st.subheader('Raw data')

    st.write(df)
if st.sidebar.checkbox('Dataset Quick Look'):
```

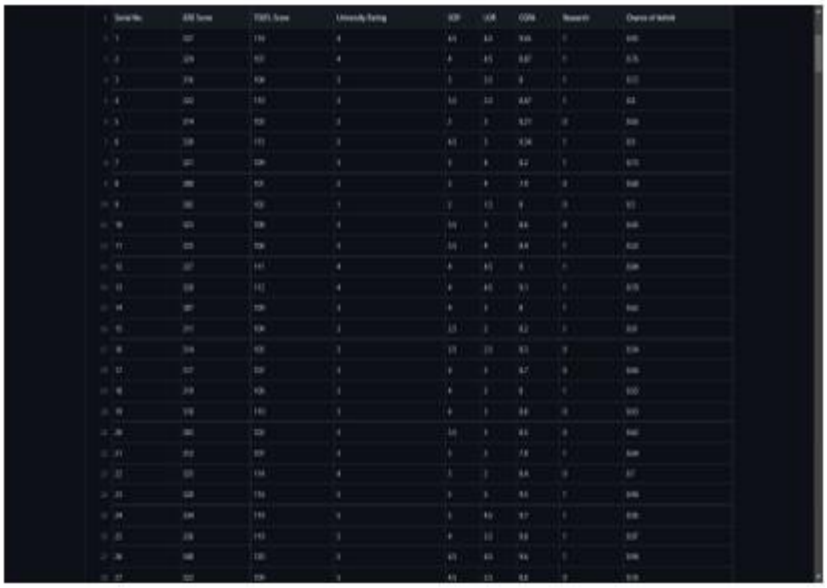
```
    st.subheader('Dataset Quick Look:')

    st.write(df.head())
if st.sidebar.checkbox("Show Columns"):
    st.subheader('Show Columns List')
    all_columns = df.columns.to_list()
    st.write(all_columns)
if st.sidebar.checkbox('Statistical Description'):
    st.subheader('Statistical Data Descripition')
    st.write(df.describe())
if st.sidebar.checkbox('Missing Values?'):
    st.subheader('Missing values')
    st.write(df.isnull().sum())
st.header('Data Visualization')
st.markdown("Tick the box on the side panel to create your own Visualization.")
st.sidebar.subheader('Data Visualization')
if st.sidebar.checkbox('Count Plot'):
    st.subheader('Count Plot')
    st.info("If error, please adjust column name on side panel.")
    column_count_plot = st.sidebar.selectbox(
        "Choose a column to plot count.", df.columns[:5])
    fig = sns.countplot(x=column_count_plot, data=df)
    st.set_option('deprecation.showPyplotGlobalUse', False)
    st.pyplot()
if st.sidebar.checkbox('Distribution Plot'):
    st.subheader('Distribution Plot')
    st.info("If error, please adjust column name on side panel.")
column_dist_plot = st.sidebar.selectbox(
        'Choose a column to plot density.', df.columns[:5])
    fig = sns.distplot(df[column_dist_plot])
    st.set_option('deprecation.showPyplotGlobalUse', False)
    st.pyplot()
# Showing the Prediction Model
st.header('Building Prediction Model')
st.sidebar.subheader('Prediction Model')
```

```
st.sidebar.subheader('Prediction Model')
st.markdown("Tick the box on the side panel to run Prediction Model.")
if st.sidebar.checkbox('View Prediction Model'):
    st.subheader('Prediction Model')
    # pickle_in = open('models/Multiple_Linear_Regression.pkl', 'rb')   # model = pickle.load(pickle_in)
    API_KEY = "<Your API Key>"
    token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
    API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
    mltoken = token_response.json()["access_token"]
    header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
    @st.cache()
    # defining the function to predict the output
    def convert_toefl_to_ielts(val):
        if val > 69 and val < 94:
            score = 6.5
        if val > 93 and val < 102:
            score = 7.0
        if val > 101 and val < 110:
            score = 7.5
        if val > 109 and val < 115:
            score = 8.0
        if val > 114 and val < 118:
            score = 8.5
        if val > 117 and val < 121:
            score = 9.0
        return score
    def pred(gre, toefl, sop, lor, cgpa, resc, univ_rank):
        # Preprocessing user input
        # ielts = convert_toefl_to_ielts(toefl)
        if resc == 'Yes':
            resc = 1
        else:
            resc = 0
        #Predicting the output
        # prediction = model.predict(
        #     [[gre, toefl, univ_rank, sop, lor, cgpa, resc]])
```

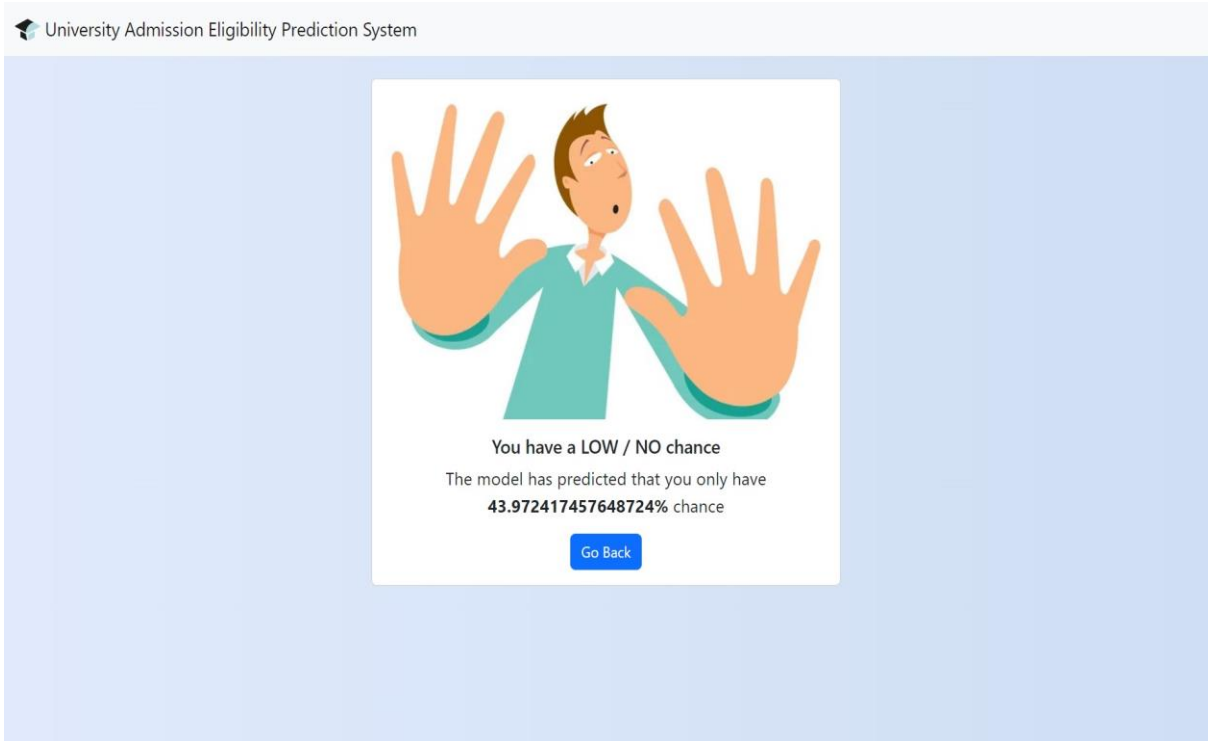
```
        # prediction = model.predict(
        #     [[gre, toefl, univ_rank, sop, lor, cgpa, resc]])
        payload_scoring = {"input_data": [{"field": [["GRE Score","TOEFL Score","University Rating","SOP","LOR ","CGPA", "Research"]],
        "values": [[gre, toefl, univ_rank, sop, lor, cgpa, resc]]}]}
       response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/uaep_deployment/predictions?version=2022-11-12', json=payload_scoring,
        headers={'Authorization': 'Bearer ' + mltoken})
        prediction = response_scoring.json()['predictions'][0]['values'][0][
        st.info("Chance of Admittance for University Rank " + str(univ_rank) + " = " + str(prediction[0]*100) +" %"
        if prediction[0] >= 0.6667:
                st.success(
               'Congratulations! You are eligible to apply for this university!')
            chance = Image.open('images/chance.png')
            st.image(chance, width=300, caption="High Chances !")
        else:
            st.caption('Better Luck Next Time :)')
            no_chance = Image.open('images/noChance.jpg')
            st.image(no_chance, width=300, caption="Low Chances :(")
    # Main function for the UI of the webpage
    def main():
        # Text boxes in which user can enter data required to make prediction
        gre = st.number_input('GRE Score (out of 340):', min_value=0, max_value=340, value=260, step=1)
        toefl = st.number_input('TOEFL Score (out of 120):', min_value=0, max_value=120, value = 80, step=1)
        sop = st.slider("SOP Score (out of 5):", value=0.0,
                        min_value=0.0, max_value=5.0, step=0.5)
        lor = st.slider("LOR Score (out to 5):", value=0.0,
                        min_value=0.0, max_value=5.0, step=0.5)
        resc = st.selectbox('Research Experience:', ("Yes", "No"))
        cgpa = st.number_input('Enter CGPA (out of 10):', min_value=0.0, max_value=10.0, value=5.0, step=0.1)
        univ_rank = st.slider("University Rank (1 to 5):", value=1,
                        min_value=1, max_value=5, step=1)
        # when 'Predict' is clicked, make the prediction and store it
        if st.button("Predict")
            result = pred(gre, toefl, sop, lor, cgpa, resc, univ_rank)
    if __name__ == '__main__':
        main()
```

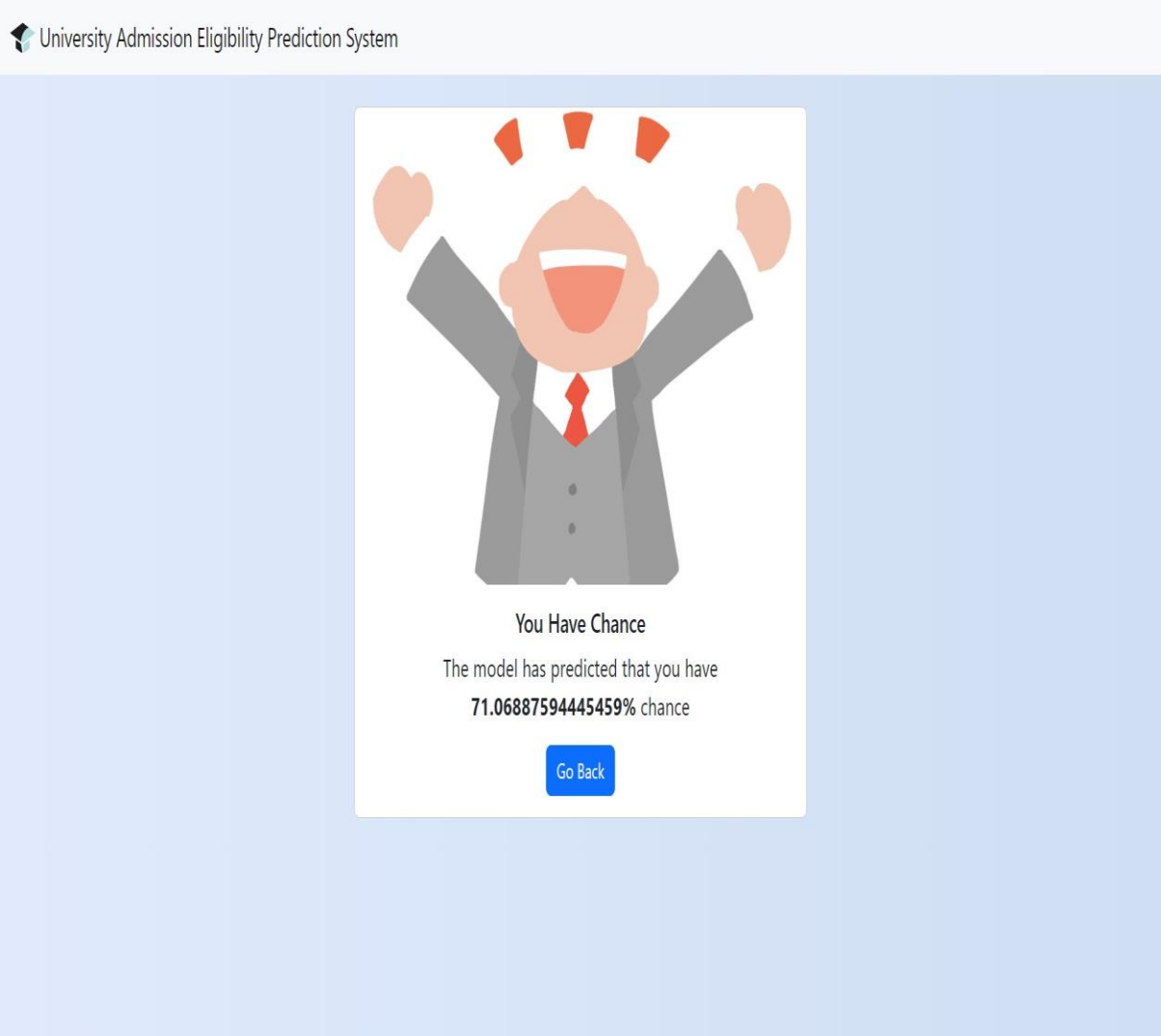

IBM-19316-1662602148 Project
Sprint Delivery Plan..pdf
Build html.html
File | C:/Users/Lenovo/Desktop/Build%20html.html
University Admit Eligibility Predictor!
A Simple Web App to Predict The Chances of Getting Admission in Universities Based on Student's Profile
Input Guide
GRE Score (out of 340)
TOEFL Score (out of 120)
University Rating (out of 5) - the category of the target university
Statment of Purpose {SOP} Strength (out of 5)
Letter of Recommendation {LOP} Strength (out of 5)
Undergraduate CGPA (out of 10)
Research Experience (0 for NONE and 1 for YES)
GRE Score
TOEFL Score
University Rating
SOP
LOR
CGPA
Research
Predict
{{prediction_text}}
Type here to search
25°C Haze
ENG
05:22 PM
19-11-2022

8.TESTING



9. RESULT


University Admission Eligibility Prediction System
You have a LOW / NO chance
The model has predicted that you only have
43.972417457648724% chance
Go Back



10. ADVANTAGES & DISADVANTAGES

❖ ADVANTAGES

- It helps student for making decision for choosing a right college.
- Here the chance of occurrence of error is less when compared with the existing system.
- It is fast, efficient and reliable.
- Avoids data redundancy and inconsistency.
- Very user-friendly.
- Easy accessibility of data

❖ DISADVANTAGES

- **Security Concerns**
- **Authenticity**

- **Infrastructural Requirements**

11.CONCLUSION

We have successfully developed an application using python flask, HTML, CSS.
By using the application, we can predict weather we can get admission in the desired University or not.

12. FUTURE SCOPE

In future we would like to enhance the existing model in such a way that consumer feels the same way when purchasing in store using Virtual reality and other upcoming technologies Reaserch to improve the accuracy of the system is under progress.

13. APPENDIX Source Code GitHub & Project Demo Link

http://ibmsmart.pythonanywhere.com/home