Initial Project Planning Report

Date	14 Dec 2024
Team ID	739961
Proje Name SmartLender - ct	Plant seedling classification using Deep learning
Maximum Marks	4 Marks
Maximum Marks	4 Marks

Product Backlog, Sprint Schedule, and Estimation

Use the below template to create a product backlog and sprint schedule

Spr int	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Team Members	Spr int Sta rt Dat e	Sprint End Date (Planne d
Sp r int - 1	Data Collection and Preprocessin g	SL-3	Understandin g & loading data	Low	sathwika	202 4/10 /20	2024/10/21
Spr int- 1	Data Collection and Preprocessin g	SL-4	Data cleaning	High	sathwika	202 4/10 /20	2024/10/21

Medium

2024/10/21

202

4/10

sathwika

SL-5

EDA

Data

Collection

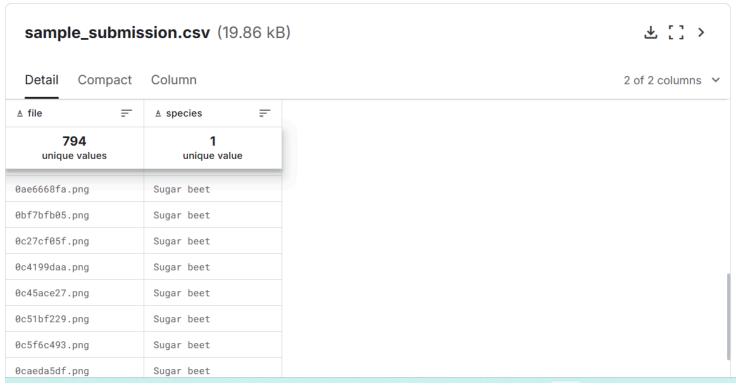
Sp

r int

1	and Preprocessin g					/20	
Sp r int - 4	Project Report	SL-20	Report	High	fareed	202 4/10 /21	2024/10/22
Sp r int - 2	Model Development	SL-8	Training the model	Medium	Rohan	202 4/10 /21	2024/10/22
Sp r int - 2	Model tuning and testing	SL-13	Evaluating the model	Medium	Rohan	202 4/10 /21	2024/10/22

Sprin t	Function a l Requirem ent (Epic)	User Story Number	User Story / Task	Priority	Team Members	Sprint Start Date	Sprin t End Date (Plan ned)
Sprin t-3	Web integrati on and Deploy ment	SL-16	Building Html templates	Low	varsha	2024/ 10/22	2024/10 /24

Sprin t-3	Web integrati on and Deploy ment	SL-17 Local deployment	Local deployment	Medium	varsha	2024/ 10/24	2024/10 /27
	1110110						



Screenshot:

```
JII IIASK IIIPOLL IIASK, LEHUEL_LEIIPIALE, LEYUESL
import openai
# Initialize Flask app
app = Flask( name )
# Set your OpenAI API key
openai.api_key = 'your-openai-api-key' # Replace with your actual API key
# Home route
app.route('/')
def home():
   return render template('index.html') # Render the home page
 Chat route
Dapp.route('/chat', methods=['POST'])
def chat():
   user_message = request.form['user_message'] # Get the user's message from the form
   # Send the user message to OpenAI API and get a response using the new API method
       response = openai.ChatCompletion.create(
           model="gpt-4", # Use GPT-4 or another model like gpt-3.5, etc.
           messages=[{"role": "user", "content": user_message}],
           max_tokens=150,
           temperature=0.9
       openai_response = response['choices'][0]['message']['content'].strip() # Extract the response text
   except Exception as e:
       openai response = f"Error: {e}"
   return render_template('index.html', user_message=user_message, bot_response=openai_response)
  __name__ == '__main__':
   app.run(debug=True)
```