Project Planning Report - Group 22 Vines

Virtual Garden GitHub Repository

Virtual Garden Milestone 1: Project Planning

Table of Content

1. Introduction

Virtual Garden is a web application designed to assist plant enthusiasts in identifying plant species, assessing their health, and receiving tailored care recommendations. By integrating AI-powered image recognition and real-time weather data, the app simplifies plant care for users of all experience levels. This milestone establishes a structured project plan, covering API selection, key features, prototype development, SDLC model selection, task breakdown, scheduling, risk assessment, and system architecture. With these elements in place, the project is set for efficient execution and development.

2. API Selection and Usage

Selected APIs

- **API 1: [Google Gemini]**
    - Purpose: [AI image analysis (for plants) and NLP]
    - Usage in Project:
        - Identifies plant species from user uploaded photos
        - Assesses plant health by detecting diseases, pests or deficiencies
        - Generates personalized care recommendations based on the plant current conditions
- **API 2: [OpenWeatherMap]**
    - Purpose: [Provides real time and forecasted weather data]
    - Usage in Project:
        - Fetches current weather conditions to inform the user about watering and care decisions
        - Provides weather forecasts to help users prepare for changing conditions
        - Adjusts care recommendations based on weather factors like temperature, humidity, and rainfall

3. Features and Implementation

3.1 Planned Features

Feature	Feature Name	API Used	Description	User Benefit
1	[Plant Species Identification]	[Google Gemini]	[Users can upload a photo of a plant, and the AI will identify its species based on the image]	[Helps users quickly and accurately determine what plants they have in case the user does not know the species]
2	[Plant Health Analysis]	[Google Gemini]	[The AI scans images for signs of disease, pests, or nutrient deficiencies and provides diagnosis and treatment solutions]	[Allows early detection of plant issues, helps prevent further damage]
3	[AI Care Recommendation]	[Google Gemini]	[Based on the identified species and the plant health status, the AI generates care tips about watering, light, humidity, temperature, fertilization, soil/potting]	[Simplifies plant care by providing a comprehensive guide to taking care of that specific plant]
4	[Display Current Weather and Forecast]	[OpenWeather Map]	[API will provide local weather data to be displayed to the user]	[The user will have an accurate and clear idea of current weather and forecast which is relevant to plant care]
5	[Care Tips Based on Current Weather]	[OpenWeather Map]	[API will provide current weather context to the AI so it can generate actionable advice for taking care of the plant]	[Helps users keep plants healthy by providing care adjustments based on temperature and humidity]

6	[Care Planning Based on Forecast]	[OpenWeather Map]	[API will provide weather forecast context to the AI so it can generate advice for each day in the forecast]	[Helps users plan ahead when taking care of their plant by providing advice for upcoming days]

3.2 Feature Revisions

- OpenWeatherMap: Redundant features were removed and replaced by ones that integrate with the UI/Google Gemini.

4. Prototype Development

Mid-Fidelity Prototype: Figma

The interactive prototype was created using Figma and demonstrates the user flow for the app's main analysis feature as well as sub-features like the About, FAQ, and Contact pages. The UI was designed to adhere to the 10 Usability Heuristics, including but not limited to Visibility of System Status, Consistency & Standards, and Aesthetic & Minimalist Design.

5. Software Development Lifecycle (SDLC) Model

Chosen SDLC Model

- Agile: Kanban - GitHub Projects
- Justification:

1. Small Team Collaboration:

Due to only having 4 people, in a small team, Agile is friendly to small teams and allows us to iterate quickly based on feedback.

2. Optimized Task Management:

With a Kanban, we can easily visualize the task in different stages, (Planning, To-do, In Progress, Review, Testing, and Done), and efficiently distribute tasks. This lets us have a smooth task distribution as well.

3. Flexibility In Development:

Kanban allows us to modify, add, or adjust requirements easily. To make our project easier to implement.

4. Workflow Efficiency:

Our project involves multiple components, kanban makes us able to work on tasks without waiting for a cycle to complete. This is also because of the timelines set, we can't be dependent on fixed development cycles.

5. Faster Feedback:

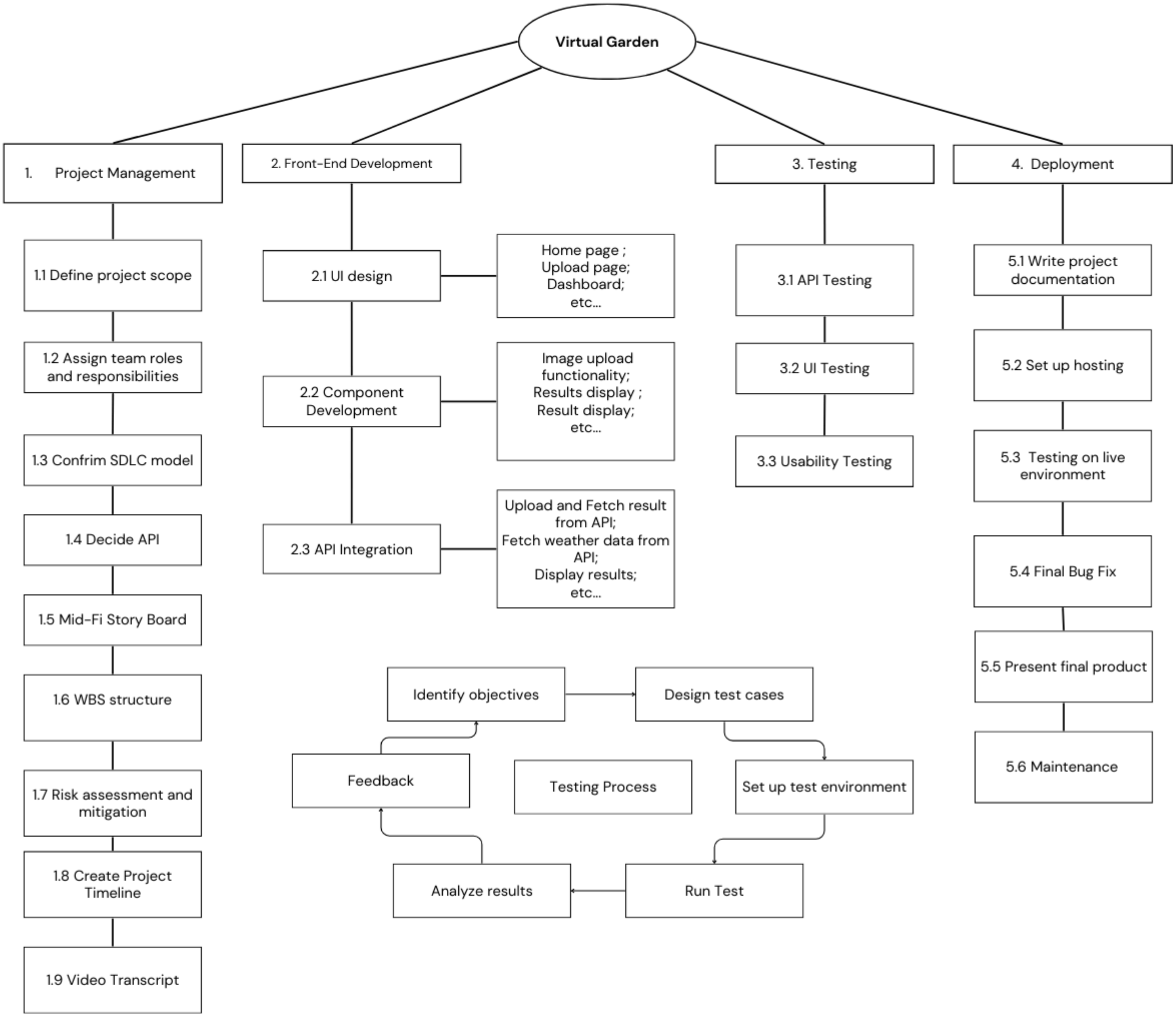
We can get quick feedback after every task is finished and improve or change the project based on what we have.

6. Better Collaboration

Each member can pick up tasks based on their skills. Ensuring group members can self-managed and efficient workflows.

6. Work Breakdown Structure (WBS)

- Hierarchical list of all project tasks
- Prioritization based on dependencies and importance
- Tasks converted into GitHub Issues



7. Project Schedule and Milestones

Timeline Overview

Milestone	Task	Start Date	Internal Deadline	Final Deadline	Buffer Deadline
M0: Proposal	Finalize project idea, APIs & features	Jan 20	Jan 29	Feb7	2 Days
	Write project proposal report	Jan 24	Feb 5	Feb 7	2 Days
	Submit project proposal	Feb 10	Feb 9	Feb 9 (deadline extended)	1 day
M1: Project Planning	Finalize API selections	Feb 8	Feb 14	Feb 16	2 days
	Define & refine features	Feb 9	Feb 15	Feb 17	2 days
	Create mid-fidelity prototype in Figma	Feb 10	Feb 18	Feb 20	2 days
	Select & justify SDLC model	Feb 12	Feb 16	Feb 20	2 days
	Develop Work Breakdown Structure (WBS)	Feb 14	Feb 18	Feb 22	2 days
	Write Risk Assessment & Mitigation Plan	Feb 15	Feb 21	Feb 24	3 days

Milestone	Task	Start Date	Internal Deadline	Final Deadline	Buffer Deadline
	Create Data Flow Diagrams (DFD Level 0 & 1)	Feb 16	Feb 22	Feb 15	3 days
	Design MVC Model Diagram	Feb 18	Feb 22	Feb 26	3 days
	Write Project Planning Report	Feb 18	Mar 3	Mar 5	2 days
	Create Video Presentation	Feb 25	Mar 5	Mar 7	2 days
	Submit M1 Report & Video	Feb 27	Mar 6	Mar 7	1 day
M1.5: Check-in	Finish API 1 features	Mar 8	Mar 15	Mat 17	2 days
	Confirm API 2 features	Mar 10	Mar 15	Mar 17	2 days
	Develop interactive UI with API 1	Mar 12	Mar 16	Mar 18	2 days
	Set up CI/CD Pipeline	Mar 13	Mar 17	Mar 19	2 days
	Set up automated tests for API 1	Mar 15	Mar 19	Mar 21	2 days
	Submit M1.5 Report & TA Meeting	Mar 10	Mar 16	Mar 17-21	1 - 7 days
M2: Feature Development & Testing	Complete API integrations (Google Gemini & OpenWeatherMap)	Mar 21	Mar 27	Mar 30	3 days

Milestone	Task	Start Date	Internal Deadline	Final Deadline	Buffer Deadline
	Implement core features (Plant Identification, Health Analysis)	Mar 23	Mar 31	April 3	3 days
	Implement advanced features (Weather-based Care Tips, Forecast Planning)	Mar 25	April 3	April 5	2 days
	Develop UI & refine user experience	Mar 27	April 5	April 7	2 days
	Conduct unit & integration testing	April 1	April 6	April 9	2 days
Final Submission & Deployment	Final debugging & testing	April 6	April 9	April 10	1 day
	Deploy website & ensure functionality	April 7	April 10	April 11	1 day
	Prepare final report & documentation	April 8	April 10	April 12	2 days
	Record project video presentation	April 8	April 10	April 12	2 days
	Submit M2 Final Report & Project Presentation	April 2	April 6	April 8	2 days

8. Risk Assessment & Mitigation Strategies

Risk Analysis Table:

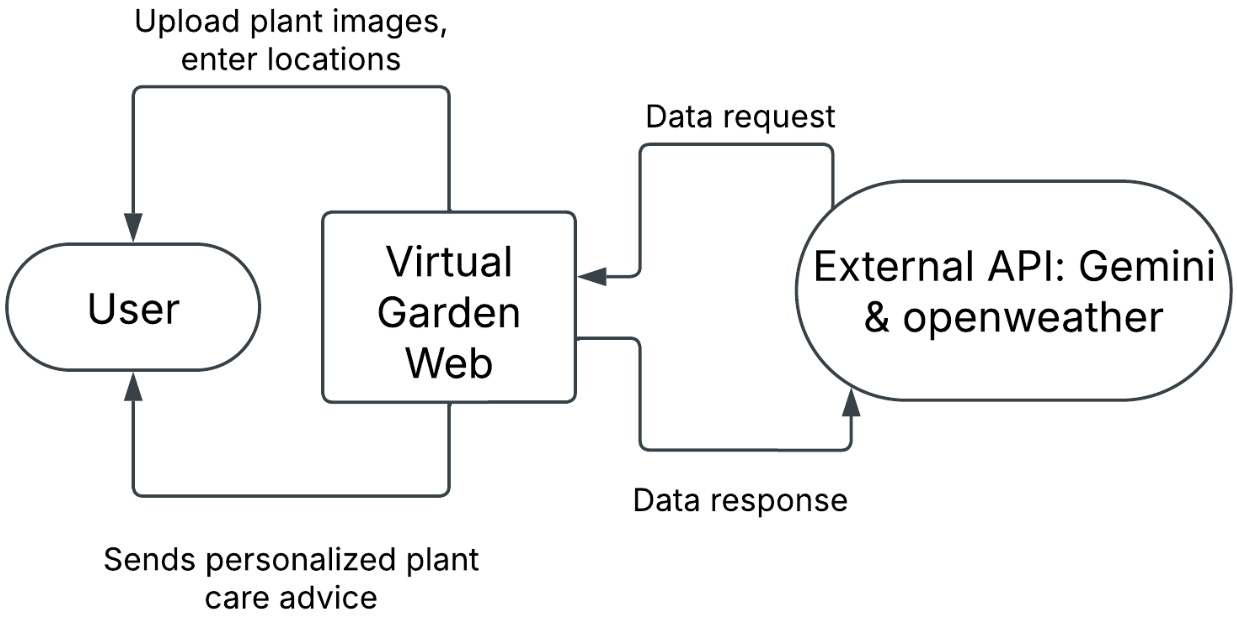
Risk Level	Issue	Description	Mitigation Strategy
Low	Spelling or grammar mistakes	Any minor textual errors in UI, documentation, or reports.	Use automated proofreading tools (grammarly), combined with constantly conducting manual proofreading.
Low	Slightly slow page load times	Any non-critical delays in page rendering due to the UI elements.	Optimize the front-end code to speed things up; like use lazy for loading images and reduce unnecessary API calls.
Low	UI inconsistencies	Any minor design inconsistencies that could affect user experience.	Conduct regular UI/UX testing and adhere to style guides.
Low	Minor API Rate Limit exceeded	If the temporary API call limit is exceeded, it could cause short-term data unavailability.	Implement caching mechanisms and optimize API calls to reduce unnecessary requests.
Low	Incomplete or Missing Documentation	Lack of proper project documentation could cause the development to slow down.	Constantly require team members to document work as part of their tasks and review.
Medium	Unclear user interface	Users could find the interface confusing, which could affect usability.	Redo UI design or add more guide
Medium	AI misidentifies plants	AI could incorrectly classify plant species, leading to incorrect care advice.	Add confidence rating and ability for users to manually correct misidentified plants.

Risk Level	Issue	Description	Mitigation Strategy
Medium	API response issues	The API may occasionally return slow responses or errors	Display error messages when API returns slow responses or errors.
Medium	Image upload errors	Users might upload unsupported file formats or large images.	Ensure image inputs are restricted to valid formats and resolutions, and provide clear error messages otherwise.
Medium	Unexpected UI/UX Bugs	Users may encounter minor layout or functionality issues.	Implement thorough testing using different devices and browsers and address issues based on severity.
High	Major API Downtime	API services (Google Gemini, OpenWeatherMap) may experience extended outages.	Integrate backup APIs, cache frequently accessed data, and implement "degradation" to maintain part of the functionality
High	Incorrect AI diagnosis	AI might provide misleading plant health diagnoses, leading to user frustration.	Include disclaimers, and offer users the option to read additional resources
High	Bug causing crashes	A major bug in the system might cause the application to crash.	Try rollback first, Set the error monitor and logging, maintain frequent backups, and fix the bug before deployment.
High	Critical Security Vulnerabilities	Potential security threats such as API key exposure, SQL injection, or data leaks.	Potential security threats such as API key exposure, SQL injection, or data leaks.
High	Data Loss or Corruption	Important data might be lost due to a system failure or user error.	Perform regular backups, implement version control, and introduce confirmation steps for critical actions.

9. Data Flow Diagrams

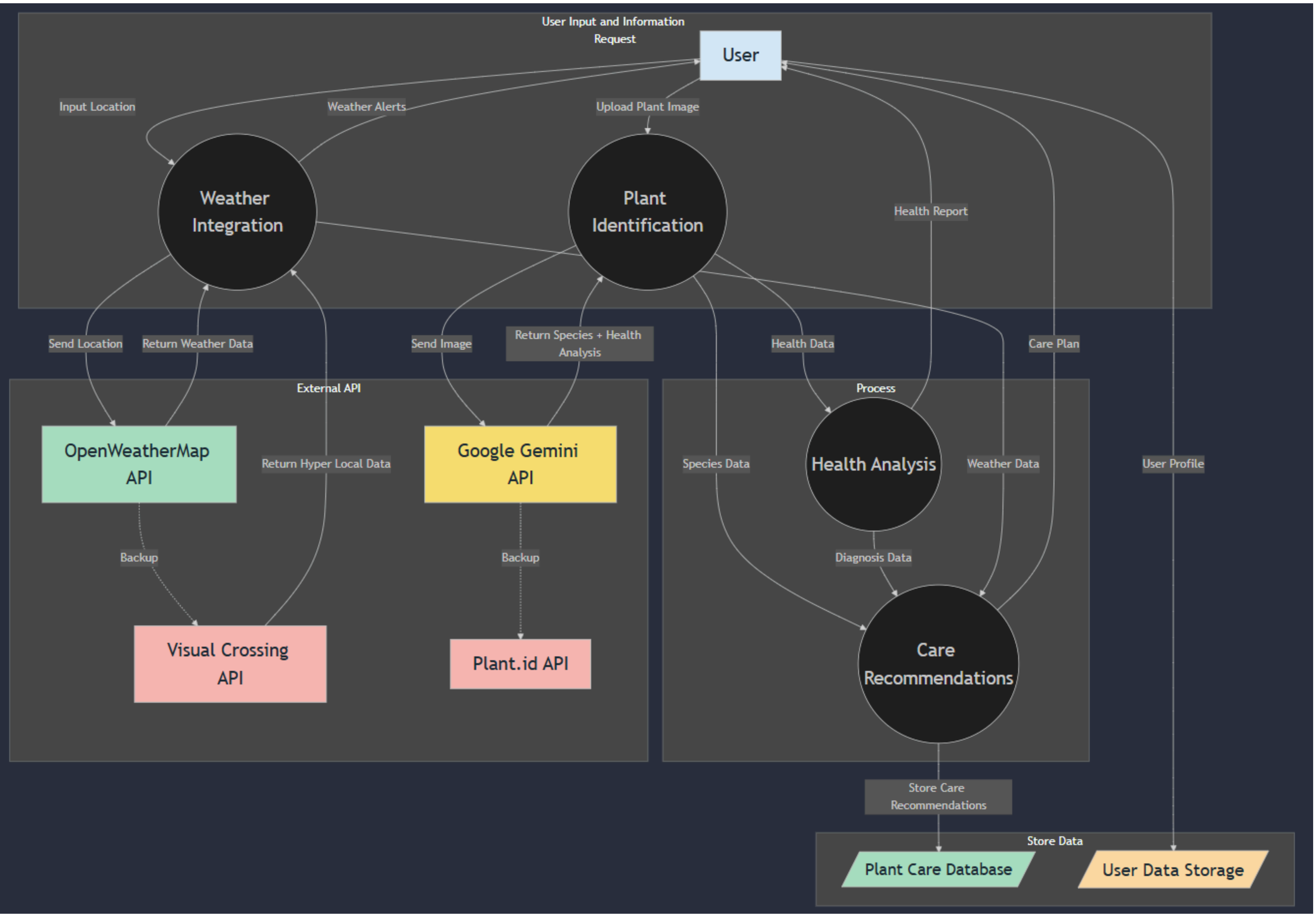
9.1 Level 0 Data Flow Diagram : Work Breakdown Structure

- Diagram illustrating data interaction

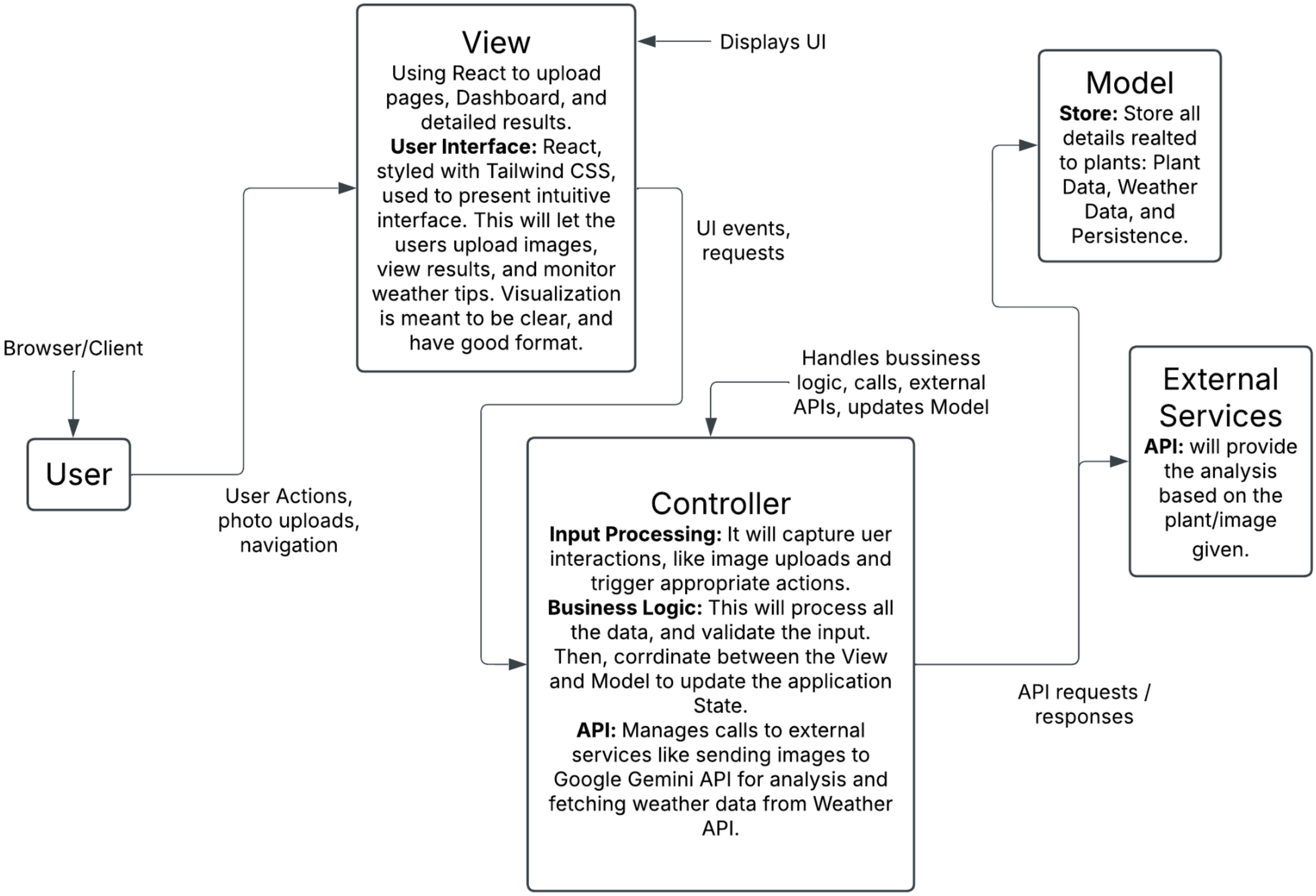

Upload plant images, enter locations
Data request
User
Virtual Garden Web
External API: Gemini & openweather
Data response
Sends personalized plant care advice

9.2 Level 1 Data Flow Diagram : Work Breakdown Structure

- Detailed breakdown of data flow within the application



10. MVC Model Diagram


Browser/Client
User
User Actions, photo uploads, navigation
Displays UI
UI events, requests
Handles bussiness logic, calls, external APIs, updates Model
API requests / responses

View

Using React to upload pages, Dashboard, and detailed results.

User Interface: React, styled with Tailwind CSS, used to present intuitive interface. This will let the users upload images, view results, and monitor weather tips. Visualization is meant to be clear, and have good format.

Model

Store: Store all details realted to plants: Plant Data, Weather Data, and Persistence.

External Services

API: will provide the analysis based on the plant/image given.

Controller

Input Processing: It will capture uer interactions, like image uploads and trigger appropriate actions.

Business Logic: This will process all the data, and validate the input. Then, corrdinate between the View and Model to update the application State.

API: Manages calls to external services like sending images to Google Gemini API for analysis and fetching weather data from Weather API.

11. Appendix

11.1 Group Member Contributions

Name	Role	Contribution
Sadhika Huria	Project Manager	● Managed all tasks. ● Organized tasks, and report. ● Created MDC Model ● Created Project Timeline ● Contributed creating Risk Assessment and Mitigation Strategies ● Contributed making WBS ● Edited the presentation video
Nathan Fassler	UI/UX	● Finished low and medium-fidelity storyboards.
Junhao Xu	Testing;Developer	● Work Breakdown Structure ● Software Development Lifecycle Model ● Added Subtitles to the video
Duong Ha Minh Khoa	API Developer	● Finished API selection/usage, Features and Implementation ● Worked on Data Flow Diagrams (idea for level 0 and finished the level) ● Contributed in some Risk Assessment & Mitigation Strategies

11.2 Changelog

Change #	Description	Date	Reason
1	Revised API Features	03/05/2025	OpenWeatherMap: Redundant features were removed and replaced by ones that integrate with the UI/Google Gemini.

End of Report