Virtual Garden — Final Project Report

Group 22 - Vines | CMPT 276 Spring 2025

Table of Contents

1. Links and Details

2. User Needs & Success Evaluation
    - 2.1 Problem Summary
    - 2.2 How the Application Solves it
    - 2.3 Feedback from Real Users

3. SDLC Model Analysis
    - 3.1 Chosen Model: Agile (Kanban)
    - 3.2 Changes Made to the Process

4. Feature Implementation by API
    - 4.1 Google Gemini API
    - 4.2 OpenWeatherMap API

5. CI/CD Pipeline Overview
    - 5.1 Pipeline Tools Used
    - 5.2 Workflow Description

6. Testing Strategy
    - 6.1 Testing Types

6.2 Tools & Frameworks

6.3 Peer Feedback Integration

7. System Architecture

7.1 Architecture Diagram

7.2 Level 1 Data Flow Diagram (Updated)

7.3 MVC Model Diagram (Updated)

8. Known Bugs and Issues

9. Future Work and Improvements

10. Lessons Learned

11. Appendix

11.1 Group Member Contributions

11.2 Changelog Table

11.3 Peer Testing Feedback

---

1. Links and Details

- **Project Title**: Virtual Garden
- **Team Name**: Group 22 — Vines
- **Team Members**:
    - Sadhika Huria – Project Manager
    - Nathan Fassler – UI/UX Designer
    - Junhao Xu – Testing, Developer
    - Duong Ha Minh Khoa – API Developer
- **Links**:
    - GitHub Repository: [Link]
    - Deployed Website: [Link]
    - Demo Video: [Link]
    - Video Presentation: [Link]

---

2. User Needs & Success Evaluation

2.1 Problem Summary

Plant care is a rewarding journey, however it can often be intimidating, especially for individuals with fewer knowledge and time. Many people face critical challenges in this journey, ranging from beginner houseplant enthusiasts to experienced gardeners. Some of the common challenges include difficulty identifying plant species, diagnosing health issues and determining the correct care routine for each plant. The problem is the individuals frequently find themselves overwhelmed by going online and often finding conflicting or generic advice; they might have issues diagnosing problems like yellowing leaves, or fungal growth, etc. Our app's goal is to help users based on their needs only. The care advice, the weather details, the diagnosis is customly tailored to their needs.

2.2 How the Application Solves It

Explain how each key feature meets those needs, referencing:

- Plant identification: Our plant identification features, lets you upload pictures, which would then be analyzed by Google Gemini API to identify the plant.

- Health assessment: Following plant identification, the website automatically provides the user with the health assessment of the plant. It lets the users know what condition their plant is in.

- Personalized care tips: Along with health assessment, personalized care tips are provided. This is customly tailored according to the health assessment, the plants needs, and the weather at the time.

- Weather integration: Weather and seasons play an important role in a plant's health, and care routine. The website provides real-time weather details that would help the user have an understanding of how to take care of their plant.

2.3 Feedback from Real Users

We had 5 testers in our peer-reviewing session:

- **Overall User Satisfaction:** 100% of the users rated the app a 5/5 for ease of use.
- **5-star rating:** 80% of the users gave 5/5 rating, with the average score being 4.8 stars
- **Design and layout:** According to users said the intuitive design score is 4.8/5
- **Confusion or errors:**
  User 1: "File type bug? Something went wrong when I uploaded the first image, might be an API problem."
  User 2: "There were proper error messages, however they were in a different language than the rest of the application
- **Positive comments:**
  "Very nicely designed application. Useful, would use myself."
  "The buttons work perfect and smooth... I can see myself using this website to get info about unknown plants in my mom's apartment."
- **Improvements suggestions:**
  Adding a favicon
  Reducing white space
  Improving error message with language consistency

---

3. SDLC Model Analysis

3.1 Chosen Model: Agile (Kanban)

- As a small team, instead of using a fixed linear process, we continuously adapted, prioritized, and delivered features. As we had other classes and other things to attend, it was very important that we are able to communicate how much work has been done, and at what step of the process each team member is on.
- Using github issues, multiple issues, including non-coding tasks were created, which were then put onto the kanban board, which allowed us to view our progress.
- As we were all also playing the role of developers, communication was very important to know who was working on which tasks. Assigning ourselves or others on each github issue, made sure the communication was proper. Each member could go to the repo, and see what tasks are still needed to be picked up, and thus would pick them up; or they could realize someone else has started a task they were thinking of picking up. The kanban board was very helpful throughout this process.
- To ensure team collaboration, and to make sure everybody is happy with results and the process, each review process was tried to be completed by a different team-member (different than the one who completed the task).
- Efforts were made so that, after each PR or feature implementation, manual testing was done by various team members to ensure a continuous cycle. This allowed us to catch many bugs in the early stages, instead of just finding everything at once.

3.2 Changes Made to the Process

- Originally the task progress is expected to follow a strict linear flow. However, as many tasks were small, many team members were allowed to skip directly from “In progress” to “Done”, or from “Ready to be picked up” to “in review”. This made it easier for team members to not give constant updates on just minor tasks, and made sure the board wasn’t overly congested.

---

4. Feature Implementation by API

4.1 Google Gemini API

Feature	Description	Implementation summary	Changes Since M1
Species Identification	Allow users to upload a photo and receive the identified plant species.	Implemented using Google Gemini API.	None- implemented as originally planned.
Health Assessment	Detects potential plant issues, such as pests or diseases.	The images are passed to the Gemini model, which would then identify signs of plant stress. The response is parsed and shown as a health overview with potential issues listed.	The idea was to have a separate page for all the results, as well. However it was implemented with the identification
AI Care Recommendations	Provide care instructions based on plant species and health conditions.	Generate personalized advice on watering, sunlight, soil, and maintenance.	The plan was to divide up the website, which would also have the advice section, but this was also implemented along with identification.

4.2 OpenWeatherMap API

Feature	Description	Implementation summary	Changes Since M1
Real-Time Weather	Shows current temperature, humidity, and weather conditions.	Used call to OpenWeatherMap API to fetch current weather based on geolocation.	One idea was to have a fixed section on at all times, to show weather details. Something like on a dashboard, or navbar. However it was only integrated with the identification.
Weather-Suitability	Evaluates temperature and humidity suitability for the plant	Temperature and humidity is passed to the AI, and returns evaluation of weather suitability	Implemented as planned.
Weather-Aware Insights	Suggests adjustments (e.g., water less on rainy days).	Weather data is passed to the AI prompt along with plant details. Which would result in generating weather specific tips.	Implemented as planned.

5. CI/CD Pipeline Overview

5.1 Pipeline Tools Used

- GitHub Actions (CI)
  We used GitHub Actions to implement Continuous Integration. This pipeline runs automatically on every push and pull request to the main branch. It makes sure all teammates run with the same dependencies and keeps the main branch production-ready at all times.
- Render/Vercel (CD)
  We deployed the application using Render now. Vercel was used for testing. It allowed us to redeploy whenever new changes are merged into the main branch easily. But based on the cost, we use a copy repository to deploy for free.

5.2 Workflow Description

- Branching strategy
  Main branch – Always contains the latest production-ready code.
  Feature branches – Used for developing individual features.
  Fix branches – Used for bug fixes.
  Developers created pull requests to merge changes into main, and the code was reviewed and tested before merging.

- Automatic testing setup
  Each pull request and push to the main branch triggers a GitHub Actions workflow that:
    Installs dependencies
    The test suite ensures that functions work correctly.
    If any test fails, the GitHub Actions workflow stops, and contributors are

notified directly in the pull request before the code can be merged.

- Auto-deployment flow
  Once changes are updated in our demo repository
  Render will automatically detect the update and build the latest version of the website.
  The website is then deployed to a production URL.
- Error handling or logging integration
  In our React application, the main error handling focuses on the Contact page and the analysis page.
  Contact page:
  We will ask users to input the correct email address and type in their name with a small pop-up window when sending a message. Also we show users the status of message sending. If the message length is more than the limit, or the message failed to be sent, we will return the error notice “Failed to send message.”. Otherwise, it will show “Message sent successfully!” .
  Analysis page:
  When the user attempts to upload an image without selecting a file, the wrong type of file, or the image is too big, we will send an error message to the user to make sure they choose the file that fits our requirements.
  Also if the picture does not contain plants, we will also ask the user to send a correct image including plants with an error message.

---

6. Testing Strategy

6.1 Testing Types

- Unit Testing (what was tested?)
  We performed unit testing on key frontend components to ensure that individual pieces of functionality behave correctly in isolation.
  PlantCard: Verified it renders plant data correctly (name, health status, image). Can define if it is a real plant or not.
  UploadImage: Confirmed proper file handling, preview generation, and correct file type.
  Weather: Always output the correct weather during every test.
  Email: Make sure every time email can be sent successfully and requires a valid email address.
  Page fit: Test the page fit and show normally under different page ratios.

- Integration Testing (coverage details)
  We tested the integration of the following modules to ensure data flow correctly between components and API endpoints.
  Image upload → Gemini API → Plant identification + care advice
  Weather data from OpenWeatherMap API → real-time recommendations

- Manual Peer Testing (real user interaction)
  We conducted manual peer testing with real users, including teammates and classmates. They were asked to:
    Upload plant images and analyze plants.
    Review the accuracy and clarity of health assessments and care advice.
    Try all of our functions they want and give us suggestions.
  After they finish these, we use their feedback to improve our website and find out the problem we have.

6.2 Tools & Frameworks

- Development Frameworks
    - React.JS
    - Tailwind CSS
    - Vite
    - EmailJS
- APIs & External Services
    - Gemini API
    - Open Weather Map API
    - EmailJS API
    - Geolocation API
    - Render
- Tooling & Project Management
    - Github Actions
    - Github Project
    - Figma

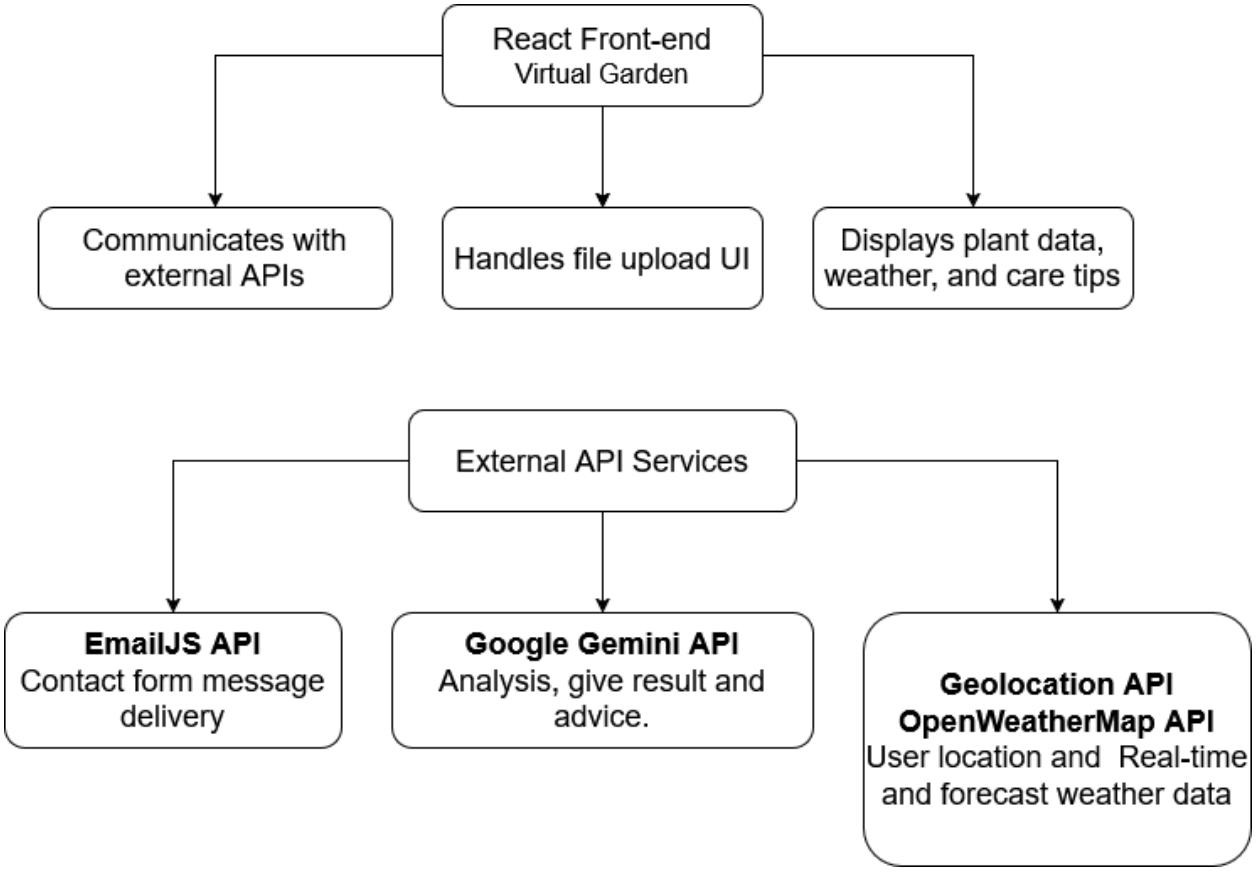
6.3 Peer Feedback Integration

- Summary of what was improved
    - Improved our prompt for the Gemini API.
    - Fixd our UI problem the website had.
    - Added icon and title.

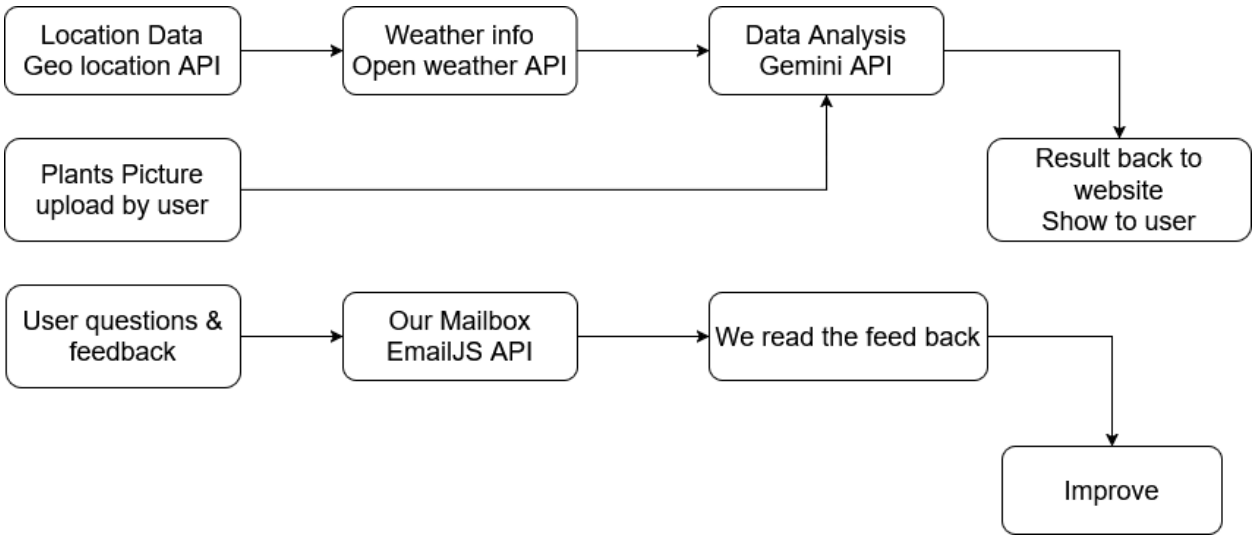
---

7. System Architecture

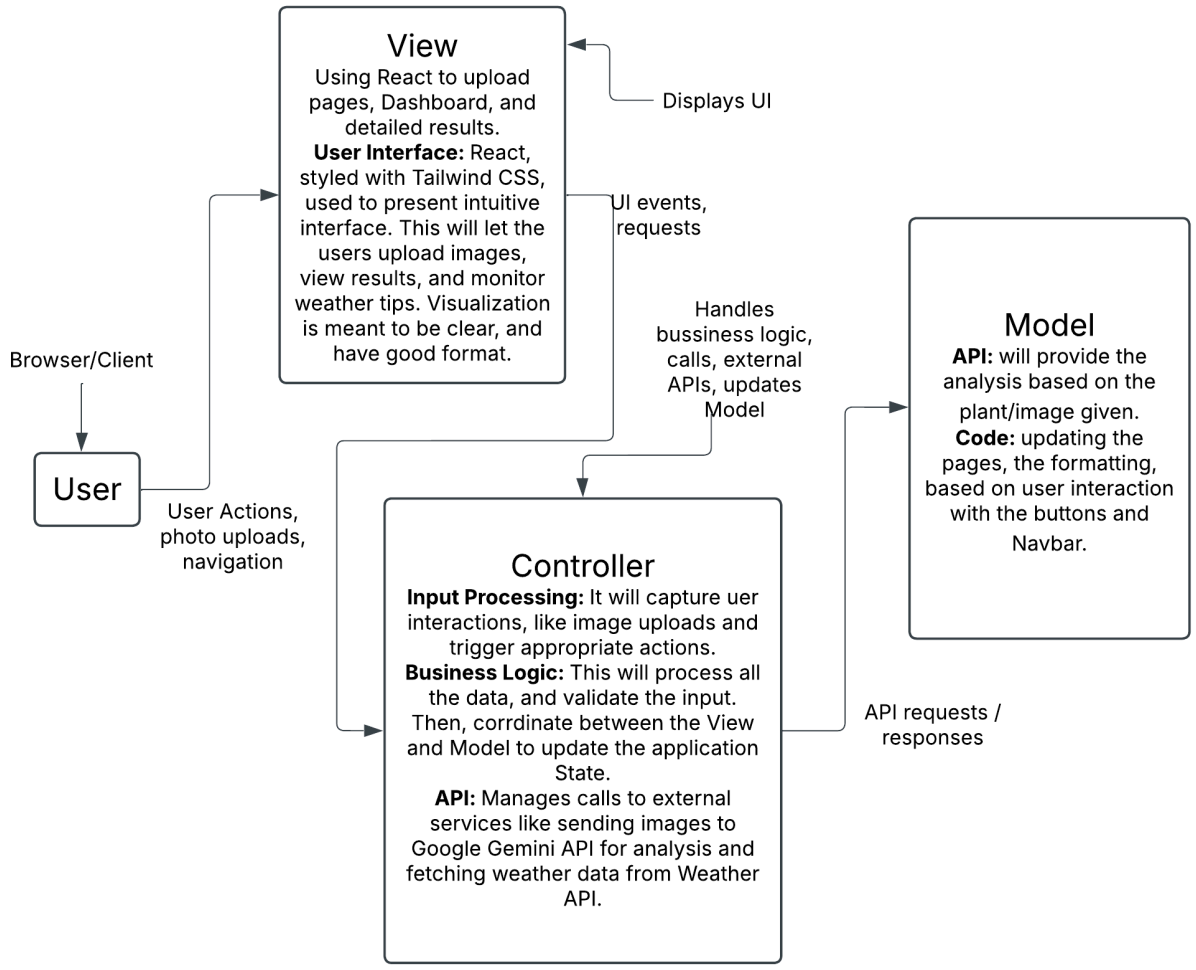
7.1 Architecture Diagram



7.2 Level 1 Data Flow Diagram (Updated)



7.3 MVC Model Diagram (Updated)



8. Known Bugs and Issues

Bug Description	Steps to Reproduce	Severity
Example: Plant health bar UI breaks	Upload large image > Analyze	Medium
Cannot access to location information (seems to be user location permission/setting)	Upload image > Analyze	Low
Font error (seems to be user browser error)	Upload image > Analyze	Low

---

9. Future Work and Improvements

- Features you couldn't implement yet
    - User account to store user's info and user's plant data.
    - Dashboard to let users track their plants info easily.
    - Ability to change selected photo before analyzing

- Ideas for making UX more intuitive
    - Animation when loading the page and elements.
    - Scrolling user reviews.
    - New User guide.
    - Dark mode.
    - Back to the top button.

- Possible backup API integrations
    - Visual Crossing API
    - Plant ID API

- Mobile responsiveness or mobile app version?
    - Separate mobile version.
    - To make the web page work properly on mobile phones, we currently only have a simple narrow ratio adaptation.

- Optional features like plant tracking, reminders, etc.
    - Calendar Integration
    - Care Reminders / Notifications
    - Multi-Language Support

---

10. Lessons Learned

Challenges and Takeaways:

- API Integration:

  Integrating two different APIs came with unexpected issues, such as inconsistent response formats, rate limits, and different types of errors one could encounter. We also found out that some weather details or error messages would differ depending on the browser, and device's language. Oftentimes, they would come up in different languages and formats.

  We tried testing in different computers and environments, as well as different browsers to nitpick each error and deal with it. Since one of our team members was facing this problem on their computer (with language), most of the error testing was done on their computer.

- Deployment:

  We had trouble figuring out where to deploy our website. A Lot of challenges were often faced in this process. Sometimes it was the difference between dynamic and static sites. Or sometimes the sites were paid. Sometimes, only individual project deployment was available, and not group projects.

  After exploring various places, we ended up doing a free trial for the deployment, which we just planned on renewing with a different account again to showcase as our final submission. After the submission we plan on using github pages for deployment.

11. Appendix

11.1 Group Member Contributions

Name	Role	Contributions
Sadhika Huria	Project Manager	Creating and organizing tasks, as well as assigning tasks. Ensuring work is completed in a timely manner. Organize the report sections, and videos.
Nathan Fassler	UI/UX	Development of UI elements, application prototypes.
Junhao Xu	Testing	Implementation, development and testing of websites.
Duong Ha Minh Khoa	APIs	Implement the main features of the application, making tests for the application.

11.2 Changelog Table

Change #	Description	Date	Reason
1	Adjusted plant care recommendation feature flow	April 3	UI clarity
2	Optimized CI testing trigger	April 5	Improved deployment reliability

---

11.3 Peer Testing Feedback

- Link to feedback form

- Text summary of key takeaways:
  Peer testing provided valuable insights to us. It showed us how users truly experience our app. While the unanimous 5/5 ratings for usability concluded that the interface is beginner and user-friendly; the other feedback also showed us the things we initially overlooked, such as excessive white space. This really affects the user impact.
  Through the peer-testing, we were reminded of what we learnt in class about how a good software is beyond just the functionality; the attention to details create a human-centered experience.
