Development Plan: Room8

Team 19
Mohammed Abed
Maged Armanios
Jinal Kasturiarachchi
Jane Klavir
Harshil Patel

April 10, 2025

Date	Version	Notes
2024-09-24	Team	Version 0.0
2024-11-14	Jinal	Version 0.1
2025-01-08	Maged	Version 1.0: Implemented TA feedback
2025-04-10	Harshil	Version 1.1: Implementing TA feedback and improving readability and formatting

The Development Plan outlines the workflows, processes, and resources required to successfully develop and deliver this project. It defines the timelines, roles, and responsibilities while ensuring alignment with the project's objectives and the team's success criteria.

1 Confidential Information

There is no confidential information associated to this project.

2 IP to Protect

There is no intellectual property to protect in this project.

3 Copyright License

MIT License: link

4 Team Meeting Plan

- The team will meet two times a week without the supervisor on Tuesday and Friday evenings
  - Meetings will be held virtually over Discord unless specified otherwise
- Meetings with the industry supervisor (R. Tharmarasa) have not been scheduled yet
- The project manager (Jinal) will begin every meeting with a discussion of existing action items, followed by a discussion of the coming deliverable, and finally end the meetings by assigning action items to the developers

5 Team Communication Plan

The platforms used for communicating will be iMessage, Discord, and GitHub. iMessage will be used for informal discussion and to schedule ad hoc meetings (either virtual or in-person). Discord will be used to host virtual meetings, and to share relevant links and resources to other developers. Finally, GitHub will be used to keep a formal track of all technical issues and meeting minutes.

6 Team Member Roles

Assigned roles are subject to change and do not exempt members from working on differing aspects of the project. These roles serve as expertise guidelines for members and stakeholders for more efficient communication.

Table 1: Team Roles

Name	**Roles**
Mohammed	General Developer, DevOps Developer
Maged	General Developer, Frontend Developer
Jinal	General Developer, Project Manager, Frontend Developer
Jane	General Developer, Team Liaison, ML Developer
Harshil	General Developer, Backend Developer

Table 2: Role Definitions

Role	**Responsibilities**
Project Manager	In charge of organizing project deadlines, creating meetings, allocating tasks accordingly, and ensuring project milestones are followed on time.
General Developer	In charge of contributing code to the repository, creating tests for their code, creating and managing issues as they arise, reviewing pull requests, and creating adequate documentation for their code.
Team Liaison	In charge of communicating with the supervisors and faculty.
Frontend Developer	In charge of building and styling the frontend UI elements of the application.
Backend Developer	In charge of handling server side functionality and network logic.
DevOps Developer	In charge of ensuring CI/CD pipeline and infrastructure operate smoothly and that automation is done efficiently
ML Developer	Responsible for researching and developing models tailored to meet the requirements of the project.

6.1 Changing Roles

Team members may request role changes by submitting a proposal via GitHub issue or team chat, where the message will include the following information:

- The desired new role.
- Reason for the change (e.g., workload adjustment, skill alignment).
- Impact on current responsibilities.

Approval requires:

- Majority vote (Greater than or equal to 50% of team) through a formal vote during a team meeting (recorded in meeting minutes).

Upon approval:

- The outgoing member must document pending tasks in the team backlog.
- Knowledge transfer through synchronization meetings.

6.2 Taking Over Roles

For temporary coverage (e.g., illness, exams):

- Members arrange handoff via GitHub issue tagged "Handoff" specifying:
  - Duration of coverage.
  - Key tasks to manage.
- Verbal agreement plus written confirmation in team chat.

For permanent role changes:

- Follow standard role change process (Section 3.1).
- Exception: If member is inactive for 2+ weeks, team may reassign role via majority vote.

7 Workflow Plan

This section goes over the plan used to track issues and how Github features such as projects and issues board will be utilized during development.

7.1 Issue Tracking

All tickets will be managed in GitHub Issues and linked to GitHub projects' Kanban board. Issues will be created using one of the following templates inside the ".github/issue_template" folder:

- bug_report.md
- feature.md
- research.md
- general.md

Issues will also use the following labels accordingly

- Code
- Management
- Bug
- Documentation
- Research
- Feature
- Maintenance
- Enhancement

Once an issue is created and filled with the appropriate information and labels, it will then be added to the backlog list inside the Kanban board.

The Kanban board will have a "Backlog" list for all the current tasks, a "To Do" list for the current sprint's tickets, "In Progress" for currently active tickets, "In Review" when the ticket with the associated pull request is being reviewed, "Dev" for tickets which are live on the development environment and are being tested, and "Done" for all the finished tickets and code which is in the production environment.

7.2 Git Workflow

All tasks must have an associated GitHub issue to track the progress and completeness of work. Tickets assigned the "Code" label require the approval of a pull request (PR) to be completed. Tickets without the "Code" label require the review of one other member before the task is marked as completed. The repository will maintain two deployed branches. A "dev" branch for development and testing and the "main" branch for production. Here are the steps for a code ticket:

- Move the ticket to "In Progress" inside of the Kanban board.
- Pull the latest "dev" branch
- Checkout a new branch following the naming convention "ticketnumber-short-description". For example, "23-integrate-new-model".
- Complete the corresponding changes
- Commit and push the branch to the remote repository
- Create a pull request to the "dev" branch with a short description of the changes
- Add any relevant screenshots
- Add the "Closes" tag. For example: "Closes #23"
- Assign a reviewer
- Move the ticket to "In Review"

Once the pull request is approved and all automated pipelines are successfully passed, the ticket will be merged to the "dev" environment and the CI/CD pipeline will deploy the new changes to the "dev" application. The ticket should now be moved to "Dev". Releases will be done aperiodically according to project deadlines. All associated tickets in "Dev" should be moved to "Done".

7.3 CI/CD Pipelines

GitHub Actions will be used to automate the testing and deployment of changes in the "dev" repository. The deployment of the "main" branch will be completed manually as releases will be completed on an as-needed basis.

Sample workflow:

- Trigger workflow on new commits to the “dev” branch
- Build and run applications and complete automated tests
- Build production application
- Update infrastructure and corresponding environment variables
- Deploy new code

8 Project Decomposition and Scheduling

GitHub Project Link: https://github.com/jinalkast/room8/blob/main/LICENSE

8.1 Tasks and Due Dates

Table 3: Tasks and Due Dates

Task To Be Completed	**Date Due**
Problem Statement, POC Plan, and Dev Plan	September 24
Requirements Doc Rev0	October 9
Hazard Analysis Rev0	October 23
VnV Plan Rev0	Novemeber 1
POC Demo	Novemeber 11-12 (Exact Date TBD)
. Design Doc Rev0 Demo	January 15
Rev0 Demo	February 3-14 (Exact Date TBD)
VnV Report Rev0	March 7
Final Demo (Rev1)	March 24-30
Final Documentation(Rev1)	April 2

9 Proof of Concept Demonstration Plan

Section entails the plan for the proof of concept demo and the risks that are involved.

9.1 Plan For Demonstration

The Proof of Concept (POC) demo will demonstrate a variety of features related to the Room8 application. The main purpose of the demonstration is to prove that the core functionalities of the system are feasible. It will be broken down

into two main components: the web application which houses a number of useful features, and the artificial intelligence (AI) algorithm for cleanliness detection. The features of the web application include a bill splitter, scheduler, and chatbot group message notifier. The bill splitter allows roommates to enter shared bills and split them with their fellow roommates. The scheduler displays a calendar view with the ability to add activities and assign them to members of the same house (i.e. chores, group events, etc.). The final component of the web app is the chatbot, which when activated, creates a group chat with all members of the house and sends relevant messages For the POC demo, the cleanliness detection algorithm will have some basic functionality. First, it will utilize a pre-trained model to detect objects in an image. Then, our own algorithm will output the difference of states between two consecutive frames ("before" and "after" picture of the shared space, i.e. kitchen).

9.2 Risks Involved

The following are the risks involved with the POC Demo:

- **Usability and User Experience Issues:** If the complexity of the components in the web app or their integration with each other is difficult for a user to navigate, it could cause them to be frustrated with their experience for example, creating a chore requires too much typing. The interactions should be intuitive.
- **Technical Limitations and Performance:** Since the POC build will not be built with scalability in mind (as it is treated as "throwaway" code), it may face issues handling multiple requests and slow down/crash when running the artificial intelligence backend code to detect changes in an image.
- **Relying on a Pre-Trained Algorithm for Object Detection:** The pre-trained model may not be robust enough for our specific use cases of common areas such as kitchens or living rooms. This may result in poor performance and could prompt the team to consider training our own model instead. The model will also not be refined to ignore common objects that will not move such as refrigerator.
- **Chatbot Sending Excessive Notifications:** Since the chatbot is configured to send notifications for new bills, chores, and other related events to the house, users may be spammed with messages and be dissatisfied with the feature.
- **Synchronization Between Features:** If data entered for one of the features (i.e. bill splitter, chore scheduler) does not update correctly, then users

will face the issue of seeing inaccurate and/or invalid data for other related features.

- **Limited Error Handling:** Since the POC is focused on delivering basic functionality and proving their feasibility, there isn't an emphasis on handling errors appropriately. This may result in users receiving inadequate feedback when they interact with the system in a way that it was not expecting. An example of this is like what was mentioned above with the application triggering a notification because the refrigerator was detected in the shared space.

9.3 Criteria for Success

If the following has been successfully demonstrated at the POC demo then the POC will be successful:

- Users are able to create an account and login on the web application
- Users can create bills, manage bills, and view past bills on the bill splitter page
- Users can see a schedule for the upcoming week and create activities which are assignable to other members of the house
- Users are able to activate the chatbot for their house which sends an introduction message to a group SMS created with all members of the house and the chatbot
- The algorithm is supplied with before and after images and then detects objects that were added, removed or moved and gives a cleanliness score depending on what was added or removed.

10 Expected Technology

Table 4: Expected Technologies

Technology	**Reasoning**
ReactJS, AngularJS, VueJS	Application will be built as a Progressive Web Application (PWA). This enables the team to build both a desktop and mobile version without requiring building separate frontends.
ExpressJS, NextJS, Django, FastAPI, Go (Gin)	Backend will be built using popular frameworks due to the abundance of documentation as well as available boilerplate. Django and FastAPI provide access to Python's many libraries such as TensorFlow and PyTorch which would ease the implementation of the AI model.
Firebase, Supabase, AWS Amplify, Appwrite	Backend-as-a-service to remove database, authentication, and account management implementation details.
PostgresSQL, MySQL, MongoDB	Further inquiry is required to decide upon document vs. relational databases. The use of a BaaS would limit the possible database options and flexibility.
PyTorch, TensorFlow	These technologies can be used to implement the AI model. This can be done directly in the backend or as separate microservices.
AWS, Google Cloud Platform, Azure	AI model can be built using existing technologies on cloud providers, providing us with more computing power and reduced load on our backend servers.
Docker, Terraform, Kubernetes, GitHub Actions	CI/CD pipeline and infrastructure as code if cloud technologies are used.
AWS, Google Cloud, Platform Azure, Vercel	Application hosting.
Jest, Cypress	Automated testing.
Git, GitHub	Version control and repository management.

11 Coding Standard

This section covers areas of coding standards that will be held during development of Room8 by going into detail about naming conventions and comments and documentation standards.

11.1 Naming Conventions

The naming conventions will be the standard for the respective programming language (ex: snake case for python). Boolean variables will always be named as a question (ex: isReady).

11.2 Comments and Documentation

Adhering to the points below will ensure readable and understandable code:

- Add comments for code sections that are not easily understood at first glance
- Clear and brief comments
- If longer descriptions are needed for functions and methods then should be appropriately documented in description document when applicable

Appendix — Reflection

Questions

1. Why is it important to create a development plan prior to starting the project?

2. In your opinion, what are the advantages and disadvantages of using CI/CD?

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?

Answers

1. Creating a development plan before starting the project has several benefits. Firstly, when creating and working on a project disagreements are inevitable. Creating the plan first can help deal with these disagreements before development starts so developers don't have different standards, goals, technologies, or programming practices. Secondly, the development plan can create a timeline/structure for the development team to follow and stay on track, in the event anyone forgets or is confused about how to proceed with the project, the development plan can always be referenced as a single source of truth. In summary, creating the development plan first is beneficial because it aligns the team on the project plan and can always be referred to when developing to realign team members.

2. The benefits of CI/CD in a software project can be grouped into three benefits. These benefits are saved time, reduced human error, and increased visibility to where breaking changes occur. CI/CD saves you time because it can run repetitive tasks in place of a developer after pushes, commits, or creating merge requests. Some tasks it can automate include running tests, pushing builds to test/production environments, and enforcing code styling. Automated testing after every commit or pull will inform developers if any new code has introduced a change that broke previous functionality (assuming the presence of well-written tests) and prevent it from being introduced into the production codebase. Finally, enforcing code styling using CI/CD by preventing merges that break the styling convention will force developers to adhere to the standards established and make reading other developers' code easier as its style is consistent with the entire project.

3. This deliverable did not produce any major disagreements as the team had many long discussions before this delivery about our project and plan of

action. In addition, our team is aware of each other's skills and often relies on the person with the most experience in a specific skill to make decisions on matters they're well-versed in, preventing disagreements.

Appendix - Team Charter

Project Information

Project Name: Room8
Project Description: Roommates often complain about cleanliness of communal spaces and have a difficult time holding each other accountable. Shared living spaces often cause friction in communicating expectations and frustrations. This application uses computer vision to detect the cleanliness of indoor environments, quantify who is creating/cleaning messes, and relay that information to occupants. It essentially serves as a mediator for student houses.
Project Supervisor: R. Tharmarasa
Project Repository: https://github.com/jinalkast/room8

Team Members

Table 5: Team Members

Name	**Specialization(s)**
Mohammed Abed	Devops Cloud Platforms
Maged Armanios	Mobile Development
Jinal Kasturiarachchi	Accessibility & Frontend Design
Jane Klavir	Artificial Intelligence & Image Detection
Harshil Patel	Security & Privacy

Team Values and Principles

Table 6: Team Values and Principles

ID	**Value or Principle**
1	We don't expect each other to know everything beforehand and we know there'll be a lot of learning throughout this project.
2	We will work as a team to complete our objectives.
3	We can always ask each other for support at any time.
4	We will work hard to ensure a good work-life balance for the team.

Communication & Meeting Guidelines

Table 7: Communication & Meeting Guidelines

ID	Guideline
1	When reaching out to teammates, be clear and concise in your communication, specifying the purpose of your message and any action items needed. Choose an appropriate medium, such as email or a group chat, and ensure your message is respectful and professional. It's also helpful to be mindful of your teammates' schedules and deadlines.
2	When responding to teammates, aim to reply promptly to avoid project delays. Acknowledge their messages within 24 hours and provide clear, constructive feedback or answers. If you need additional time to address a complex issue, let them know within the same 24-hour deadline that you need additional time to respond and set a reasonable deadline for your response.
3	For effective meetings, the person who initiates must chair it and come prepared with a clear agenda and actionable items. All participants should respect each other's opinions and avoid rude behavior such as arguing, interrupting others, or not paying attention while someone is speaking. Meetings should conclude with next steps or action items for members to execute/follow up on.
4	It is not expected for everyone to attend every meeting as the members can often have conflicting schedules or other priorities come up. It is recommended that each member attend as much as they can to prevent additional discussions to realign absent members. Members who are unable to make a meeting that requires their presence should propose an alternative meeting time instead of expecting the person to constantly reach out to avoid frustration amongst members.

Work Guidelines

Table 8: Work Guidelines

ID	Guideline
1	All work completed is to be proofread for typos and be formatted with the appropriate headers.
2	If necessary, the work should also be dated and include signature sections to show the group's collective.
3	Every individual on the team is forbidden from submitting a deliverable without every piece of work in the document being proofread by a majority of the group.
4	Guide 4 can be altered in circumstances where a member is absent to a majority of the members minus the absent members.
5	If you feel another team member is not fulfilling their responsibilities, remind them of their responsibilities early on.
6	If a team member is blocked on an action item for any reason, discuss as a team how to support them in their work so the deadlines can be met.
7	If there is a long-term blocker that would prevent the team member from working on the project for a considerable amount of time, the blocked team member is to address this to the professor or TA.