Development Plan

1 Confidential Information?

This project does not have confidential information from industry to protect. However, we all personally identifiable information (PII) will be kept confidential. Personal data will not be disclosed or made accessible to the public, and is securely stored within each user's account, accessible only to the account holder. We adhere to stringent security protocols to protect user information in compliance with applicable data protection regulations.

2 IP to Protect

There is no Intellectual Property (IP) to protect.

3 Copyright License

We are adopting the MIT license, which enables the free distribution of our work. The license can be found here: MIT License.

4 Team Meeting Plan

4.1 Meeting Schedules and Locations

- Every **Monday** (excluding breaks & holidays) from **2:30 PM to 4:20 PM** (2 hours) at the designated tutorial time. All team members are required to attend, be punctual, and come prepared with updates or questions.
- The meeting will be held in person in one of the following buildings: **MDCL, LRW, Gerald Hatch Center, or Mills Library**. Consult the team the weekend before the meeting to choose a suitable tutorial/meeting room and if necessary, reserve it for a 2 hour time period.
- **Contingency for in-person meetings**: If extenuating circumstances (e.g., severe weather) prevent meeting in person, the meeting will be held virtually or postponed to the next available date based on group consensus. Meetings may also be held virtually if discussed and agreed upon by the majority of the members. Virtual meetings will take place on our shared Discord server.
- When required, the team can host additional meetings throughout the week that align with each member's schedules to address certain roadblocks/issues encountered throughout the working week. The meeting will take place in the same buildings listed above or virtual if there is no seating available.

4.2 Meeting Structure

Each meeting will be scheduled in advance with an agenda prepared by the Meeting Secretary (more details on roles discussed in Section 4.3.). Attendance will be recorded at the start of each meeting, after which agenda items will be discussed. A list of action items will be noted at the bottom of the agenda, where each item will be assigned a specific team member for implementation.

The weekly Monday meetings will be conducted in an agile/scrum-like approach.

- The first **15-30 minutes** will be dedicated to a standup, where each person will be giving a brief update on their progress for the prior week, their current plan/task/actions items for the upcoming week, and any roadblocks or questions they may have that concerns other parts of the system.
- The remaining time meeting is allotted for additional development work, peer programming and/or tech debt. Time should be used effectively based on the upcoming deadlines for that week.

5 Team Communication Plan

- The main source of communication between the team will be via **Discord**. Meetings that need to be conducted virtually will be done through the dedicated voice channel (*General*) within the team server.
- All development-related issues will be hosted via the project's GitHub Issues tab.
  - This provides us a source of traceability for every task completion along with an interface to receive instructional feedback.
- GitHub Projects will be used as our primary project management tool. This board will be reviewed every week during Monday meetings and regularly updated throughout the week.

Member Name	**Discord (primary contact)**	**MacID**	**GitHub**
Payton Chan	@paytchan	chanp29	paytonchan
Eric Chen	@Er.$_{.}ic$	chene40	chene40
Fondson Lu	@chubbydukki	luh57	fondsonlu
Jason Tan	@jason_tan	tanj60	tan-jason
Angela Wang	@ziyuna	wanga91	angelaw7

Table 1: Team contact information

6 Team Member Roles

The roles listed below may be subject to change at any point throughout the term, as deemed necessary to best accommodate the needs of the project and the overall team objectives. They are currently unassigned; assignment of roles will be done every few weeks during Monday meetings to ensure that everyone has a chance at taking on certain responsibilities.

Additionally, these roles should not be seen as limiting the tasks or responsibilities one can take on. Team members are expected to step up and assume additional tasks when necessary, particularly when their expertise or skills align with the project's demands. It may also be necessary at certain times for members to "wear multiple hats" and assume more than one role.

Meeting Chair: The meeting chair's job is to ensure the group meetings are being conducted in an organized fashion and to facilitate the procedure for internal conflicts.

Meeting Secretary: The meeting secretary is responsible for preparing meeting agendas and taking notes during all group discussions. These notes are to be used for future reference and to keep track of thought processes that should be provided in the project documentation.

Milestone Coordinator: The milestone coordinator has the responsibility of facilitating the creation and completion of deliverables. They are also responsible for assuring that each deliverable meets the criteria provided and that the standard of excellence is being maintained throughout the duration of the project.

Documentation Coordinator: The documentation coordinator reviews documentation as the project progresses, ensuring that documents have a consistent level of quality and reflects the current state of the project. They work with team members to update, organize, and maintain a variety of documents, while also implementing best practices and managing version control.

QA Tester: The QA tester is responsible for evaluating software to identify bugs and ensure it meets quality standards. They design and execute test plans and collaborate with the development team to address issues, ensuring a high-quality user experience.

Systems Architect: The systems architect is responsible for designing the overall architecture of a software system, ensuring that it meets both technical and business requirements.

Frontend Specialist: The frontend specialist focuses on creating the user interface and user experience of a web application, using technologies like HTML, CSS, and JavaScript. They work to ensure that the application is visually appealing, responsive, and user-friendly, collaborating closely with designers and backend developers.

Backend Specialist: The backend specialist is responsible for server-side logic, database interactions, and application functionality. They build and maintain the core application infrastructure using programming languages and frameworks, ensuring that the frontend and backend communicate seamlessly to deliver a smooth and efficient user experience.

User Researcher: The user researcher conducts studies to understand user behaviors, needs, and motivations through methods like interviews, surveys, and usability testing. They analyze the findings to provide insights that enhance product design and development, ensuring that the final product aligns with user expectations

7 Workflow Plan

7.1 GitHub Issues

- GitHub Issues will be used to track and manage all development tasks.
  - An issue will be created for each task – each task should be small enough such that one developer can finish it within a time frame of one week. If the task is too large, it should be broken down into smaller tasks.
  - All issues must include the following fields:
    - **Assignee(s) -** Indicates who has/will be working on the issue.
    - **Labels -** Information regarding the category the ticket falls under (e.g., Feature, Bug, Tech Debt, Version Bump, etc.)
    - **Milestone -** The milestone/deliverable that the ticket is associated with.

7.2 GitHub Projects

- GitHub Projects will be used as the project management tool to manage the roadmap and plan the project.
  - All issues should be placed in the appropriate column (Backlog, In Progress, In Review, Complete) and updated as soon as changes are made.

7.3 Git Branches

- Git Branches will be used to make changes to the project.
  - The type of development work should be noted as a prefix to the branch
    - `feat` for feature work.
    - `bugfix` for bugfix.
    - `chore` for chore or tech debt work.
  - Branches should follow the naming convention: `<prefix>/<modId>/<desc>...-<desc>`. This is to avoid conflicting branch names and to make it clear on the topic of the change.
    - E.g., `feat/chene40/update-dependencies`.
  - Feature branches should always be branched off the updated main branch.
  - Rebase branches whenever possible for pulling new updates.
  - Once desired changes are complete on a branch, the author should open a pull request (PR) and follow the below PR guidelines.

7.4 Pull Request (PR) Guidelines

- All PRs must include a descriptive title and a small description of what the PR achieved/changed.
- Upon opening a PR, the author(s) should inform the team that it is ready for review. One member of the team will be auto-assigned as a reviewer.
- All PRs must have at least one approved review before merging.
- Reviewers should review a requested PR within a **24 hr** time frame (**48 hrs max**).
- PR should be listed as `Draft` if implementation/fixes have not been completed and converted to `Ready for Review` once a PR can be reviewed.
- PR must be linked to an issue using the appropriate `keywords` at the end of the descriptions section.
- All PRs are to be **squashed and merged**. This is important for ensuring the cleanliness of the commits on the main branch.
- On push to an open PR, a GitHub Action should run which deploys the branch to a CI/CD pipeline in which all automated tests should run to ensure at least 90% line coverage. If this condition isn't met, the PR will not be allowed to be merged.

7.5 GitHub Actions

- GitHub Actions will be utilized for CI/CD to continuously deploy the application and keep it running at all times.
  - Tests, linters, and formatters will be run on each push to a PR and the main branch.
  - If a PR causes a GitHub Action to fail, it is the responsibility of the PR implementer and reviewer to resolve the issue as soon as possible so that it does not bottleneck other developers' code integrations and implementations.

7.6 GitHub Tags

- GitHub Tags will be used to mark Revision 0 and Revision 1 versions of the project for each milestone. These should only be used for final versions of each revision and not draft work.

8 Project Decomposition and Scheduling

8.1 Project Management

We will be using GitHub projects as our project management tool. It will be used to organize and view Issues on a board organized by the status of the Issue (i.e., Backlog, Ready, In progress, In review, Done), as well as to keep track of the roadmap of the project. The project can be found here GitHub Project.

8.2 Project Decomposition

The project will be decomposed into seven milestones over the duration of the term. These milestones have provisional deadlines that may be subject to change depending on the project's progress.

Milestone 0 - User Elicitation and Requirements Specification

- **Deadline:** Friday, October 4, 2024 @ 11.59 PM
- **Estimated Time:** 2 weeks
- All team members are expected to actively participate in distributing surveys and conducting in-person interviews across campus. The insights gathered from these user engagements will be crucial in defining and refining our project requirements, ensuring they are aligned with real user needs and expectations.

Milestone 1 - System Design, HLDs, DFDs, and POCs of Ploutos

- **Deadline:** Friday, October 4, 2024 @ 11.59 PM
- **Estimated Time:** 2 weeks
- All team members must convene to collaboratively discuss and refine the system design. This meeting will encompass the development of high-level designs (HLDs), data flow diagrams (DFDs), and proof of concepts (POCs). The objective is to ensure a cohesive architectural framework and to establish a well-defined workflow that will guide subsequent phases of the project.

Milestone 2 - Project Initialization & Development Environment Setup

- **Deadline:** Friday, October 11, 2024 @ 11.59 PM (Note: Last Day Before Reading Week)
- **Estimated Time:** 1 week
- The work will be distributed evenly among team members, unless a consultation determines that another individual should take on additional responsibilities due to extenuating circumstances. The team member responsible for setting up the repository will also create the necessary onboarding documentation. Another individual will manage group-related accounts (e.g., AWS), while a designated member will oversee workflow management, including the creation of Git issue labels, milestones, and tasks. All team members are expected to follow the installation guide provided by the repository lead and ensure proper initialization of the development environment.

Milestone 3 - Create Preliminary Figma Designs for Mobile App Workflow

- **Deadline:** Friday, November 1, 2024 @ 11.59 PM (Note: Due 2 Weeks After Reading Week)
- **Estimated Time:** 2 weeks
- All team members are expected to contribute to a collaborative Figma file to develop the necessary mockups and wireframes. Individuals with more experience in Figma may assume leadership roles to facilitate the design process, while those with less familiarity are encouraged to enhance their skills through active participation. It is essential that every team member provides valuable input, ensuring that the designs reflect a comprehensive and cohesive approach to the project's objectives.

Milestone 4 - Translate and Implement Figma Wireframes Into Front End Code

- **Deadline:** Sunday, December 1, 2024 @ 11.59 PM (Estimated Time: 3 Weeks Excluding Exam Season and Winter Break)
- **Estimated Time:** 4 weeks
- All team members are expected to collaborate in translating the Figma wireframes into functional front-end code. Individuals with more experience in front-end development may take on leadership roles to guide the implementation process, while those with less familiarity are encouraged to deepen their understanding through active involvement. It is imperative that each team member contributes their insights and expertise, ensuring that the final product adheres to the design specifications and meets the project's quality standards.

Milestone 5 - Build Out Back End API Services and Connect with Front End

- **Deadline:** Friday, February 1, 2025 @ 11.59 PM (Note: Exam Season Starts; Estimated Time: 3 Weeks)
- **Estimated Time:** 4 weeks
- All team members are expected to collaborate in developing the necessary back-end API services and ensuring their integration with the front-end components. Those with more experience in back-end development may assume leadership roles to oversee the implementation process, while less experienced members are encouraged to enhance their skills through active participation. It is crucial that every team member contributes their knowledge and insights, ensuring that the final services are robust, efficient, and aligned with the overall project objectives.

Milestone 6 - E2E Test, Cleanup, Final Deploy and Showcase for Expo

- **Deadline:** Friday, February 15, 2025 @ 11.59 PM
- **Estimated Time:** 2 weeks
- All team members are expected to collaborate in performing end-to-end testing as a cohesive unit. This collaborative effort will provide an opportunity to identify and resolve any lingering bugs, refine necessary documentation, and collectively prepare for the Expo showcase. It is imperative that each member contributes their insights and expertise, ensuring that the final product is polished and ready for presentation.

Milestone 7 - Final Documentation

- **Deadline:** Wednesday, April 2, 2025 @ 11.59 PM
- **Estimated Time:** 2 weeks (Start March 16, 2025 or earlier)
- It is crucial that the project team engages in a collaborative effort to aggregate and meticulously refine all pertinent documents, including the user guide, requirements, and design documentation, approximately two weeks prior to the submission deadline. This coordinated approach will facilitate thorough review and revision, ensuring that all materials are cohesive, accurately reflect the project's objectives, and adhere to the highest standards of quality. Such diligence will be paramount for a successful final submission.

Milestone 3-6 - Integrate and Train Optical Character Recognition (OCR) Model Into Scanner (To Be Completed Concurrently Starting Milestone 3)

- **Training Period:** October 21, 2024 - March 21, 2025
- All team members are required to actively engage in the training of our OCR machine learning model. Each member must collect and retain receipts as inputs for the model whenever feasible, as these inputs are critical for facilitating continuous testing and refinement. These receipts should be uploaded to the shared Drive folder. This iterative training process is to be conducted concurrently with other development activities for the application, ensuring that the model not only evolves in response to real-world data but also integrates seamlessly with the overall project. By committing to this collaborative effort, the team will enhance the model's accuracy and reliability, ultimately contributing to the success of the application.

9 Proof of Concept Demonstration Plan

9.1 Plan for Proof of Concept Demonstration in November

Demonstrate a working mobile application on a mobile device that can perform the following functionalities:

1. When a button is pressed, the phone camera opens. The user can then take a picture of an itemized receipt.
2. Once the picture has been taken, the application runs it through the ML model, where it processes the image. The application should then list out the recognized items from the receipt. Items should also be tagged or grouped into predefined categories.

9.2 Anticipated Risks and Mitigation Plan

- **Data Accuracy and Reliability:** The ML model might misinterpret or misclassify data from receipts, leading to incorrect financial records.
  - **Risk level:** High, as it might take more data than anticipated to train a model to recognize and classify items appropriately.
  - **Mitigation plan:** We will train the model using a large dataset of receipts that contains various items from different categories, and then test the model to verify the accuracy of the results. An alternative option to manually edit the price, category, date/time, and other metadata fields about the items recorded will be provided, and this would be used as feedback to improve the model.
- **Model Generalization and Adaptability:** The ML model may struggle with adapting to new types of receipts, retailers, or products that it hasn't encountered during training, resulting in misclassifications.
  - **Risk level:** Low, in terms of the POC demonstration. We plan to demonstrate using a receipt that lists commonly purchased items for the purpose of showing the feasibility of the model.
  - **Mitigation plan:** For the first iteration of the model, we plan to target specific stores near campus (e.g., Food Basics, Fortinos, Shoppers Drug Mart) as we will be able to collect the most amount of data from these locations and monitor the accuracy of the model. As the project progresses, we will expand to different categories and stores.
- **Optical Character Recognition (OCR) Limitations:** Poor-quality receipts (faded text, creases, or low resolution) may hinder OCR performance, leading to incomplete or inaccurate data extraction.
  - **Risk level:** Medium, as we plan to use a good quality receipt (i.e., laid out flat on a table, newly printed) for the POC demonstration.
  - **Mitigation plan:** We will initially train the model using good quality receipts. Over the course of the training process, we plan to introduce lower-quality receipts and use advanced pre-processing techniques (e.g., noise reduction, image enhancement) to increase the accuracy of parsing lower-quality data. For the POC demonstration, we plan to use a good quality receipt for the purpose of showing the base functionality of the model.
- **Privacy and Data Security:** Handling sensitive financial data (such as purchase histories and personal information) poses the risk of data breaches or unauthorized access.
  - **Risk level:** Low, as we do not plan to release the application before the POC. We will only be testing locally with our team of developers, so there would be no unauthorized access.
  - **Mitigation Plan:** To mitigate privacy and data security risks during the POC, sensitive financial data will be encrypted both at rest and in transit, with access restricted to authorized developers using role-based controls. Anonymized or synthetic data will be used for testing, and local development environments will be hardened with strong passwords, disk encryption, and up-to-date software.

- **User Experience:** If the app struggles with receipt clarity, formatting inconsistencies, or unusual items, users might become frustrated.
  - **Risk level:** Low, as we plan to make user experience a priority.
  - **Mitigation plan:** We will conduct usability testing as part of our project, where users will test the app and give feedback on points of improvement for the receipt scanning workflow. For the POC demonstration, we plan to show the key functionality of taking a photo of a receipt and displaying the outputs of the model on the screen. Wireframes will be designed prior to development to ensure that the user interface and experience is well thought out.
- **Scalability and Performance:** As the user base grows, handling a large volume of receipt uploads and processing requests in real-time could impact system performance and user experience.
  - **Risk level:** Medium, in terms of performance for the POC plan. We are unsure how fast the model would be able to process an image and return the information that we would need. Scalability is not a risk for the POC plan but is important to consider for future development.
  - **Mitigation plan:** We will run load testing and stress testing simulations to show the app's capacity to handle increasing numbers of users and transactions without degradation in response time or accuracy. This does not apply for the POC demonstration but is important to consider for future development.

10 Expected Technology

The following lists the technology that we expect or are considering to use for the project. Note that these are subject to change over the course of the project.

- **Programming languages**
  - Frontend: React Native, TypeScript, Dart/Flutter
  - Backend: Python, Golang, AWS
- **Libraries**
  - Utility CSS Library: Tailwind, MUI, Bootstrap, Css-in-js, Styled Components
  - Authentication: Auth0, Firebase
- **Tools**
  - Database: MongoDB, Firebase, Amazon DynamoDB
- **OCR models**
  - Tesseract OCR
  - Google Vision API
  - Amazon Textract
- **Linter tools**
  - eslint
  - typescript-eslint
- **Testing frameworks**
  - Jest (Unit)
  - Mocha/Chai (Unit)
  - React Native Testing Library (Unit + Integration)
  - Detox (E2E)
  - Maestro (UI & Flow Tests)
- **Continuous integration (CI)**
  - GitHub Actions for running tests, linting, formatting. More details on CI plan can be found in Section 7.
- **Version control**
  - Git, GitHub
- **Project management**
  - GitHub projects

11 Coding Style/Standards

- Most of the code styling will be done via third party formatters and linters such as Prettier and ESLint. These formatters/linters will be implemented directly into the project itself via Git Hooks and will be triggered on every pre-commit via **Husky**.
- As a basis for committing changes, every commit should be atomic and contain one change pertaining to a specific feature/fix.
- **Commit Messages:**
  - All commit messages should be systematic and descriptive.
  - All commit messages should be written in present imperative tense
    - **Correct:** "Fix typo in README"
    - **Incorrect:** "Fixed typo in README"
  - All commit messages should begin with one of the following prefixes:
    - `feat: -` for feature and enhancements related implementations
    - `fix: -` for bug fixes and tech debt related fixes
    - `chore: -` for project related commits such as bumping project/dependency versions, file renames/relocations, etc.
  - Example commit messages:
    - `feat: generate openapi ts client based on api_spec.yaml`
    - `fix: change color of CTA button from red to blue`
    - `chore: reorganize file structure hierarchy`
- We will use PEP 8 style guides for Python code