Development Plan
Software Engineering

Team #5, Money Making Mauraders
Zhenia Sigayev
Justin Ho
Thomas Wang
Michael Shi
Johnny Qu

Table 1: Revision History

Date	**Developer(s)**	**Change**
09-21	Justin Ho	Added Section 10 (preliminary)
09-21	Thomas Wang	Added Section 6
09-21	Johnny Qu	Added section 8
09-22	Michael Shi	Added section 9
09-22	Thomas Wang	Added Appendix Reflection 1 and 3
09-22	Justin Ho	Added Appendix Reflection 2 and Team charter (half)
...	...	...

This document outlines the development plan for the MES Club Payment Tracking System, a web application designed to streamline the reimbursement process for McMaster Engineering Society (MES) clubs. The current process using Google Forms and spreadsheets is inefficient, error-prone, and difficult to search. This plan provides the roadmap for design and implementation, describes team and workflow organization, lists the expected technologies, and identifies major risks and mitigation strategies.

1 Confidential Information?

This project does not contain industry confidential information.

2 IP to Protect

The McMaster Engineering Society (MES) will be the primary owner of the system and its deployed instance. The student development team will be credited as contributors. Unless MES requests otherwise, the project will be published under an open-source-friendly license allowing MES and future student teams to maintain and extend the system. If MES requests proprietary handling, we will follow the agreement provided by MES and notify course staff.

3 Copyright License

Given that the MES's website source code is open-source, we will be using the MIT License.

4 Team Meeting Plan

- Weekly team meetings: 60 minutes, fixed weekday time (set by team availability) for planning and progress updates.
- Bi-weekly stakeholder check-ins: 30 minutes with MES representative(s) to gather requirements and demo progress.
- Meeting format: rotating chair; agenda created as a GitHub issue before each meeting; minutes recorded in the issue and a short summary added to the project wiki.
- Ad-hoc technical sessions as needed for design decisions or integration tasks.
- Meeting Notes: Will be documented in GitHub Issues linked to the relevant sprint or task.

5 Team Communication Plan

- GitHub Issues for tasks, bugs, feature requests and meeting agendas.
- Pull requests for all code changes with at least one reviewer before merging.
- Discord for quick coordination.
- Email for formal stakeholder communications.
- Weekly status updates posted to a shared project board or README for the MES contact.

6 Team Member Roles

- Project Coordinator/lead: Oversees the overall project direction and ensures milestones are met in a timely fashion. Facilitates communication between faculty advisors and the team, keeps team aligned with project goals and deadlines
- Meeting Chair (rotates weekly): Prepares agenda and leads team meetings. Ensures discussion stays focused and all voices are heard. Summarize action items at end of all meetings.
- Notetaker (rotates weekly): Record meeting minutes, decisions, and assigned tasks. maintains organized project documnetation. Ensures deliverables are well documented, both for technical and non-technical audiences.
- Technical Development (everyone): Ensures coding practices, frameworks, and tools are consistent across team. SUpports integration of all coding components.
- QA/Testing (everyone): Review code correctness, readability, and adherence to standards. Develops and run test cases to validate functionality
- User Experience and Requirements (everyone): Gathers and refines requirements from MES stakeholders. Focuses on usability, interface design, and accessibility.

7 Workflow Plan

- Git branching: single protected `main` branch for production; feature branches named `feature/short-description`; pull requests for merges.
- Pull request policy: descriptive PRs, link to issues, at least one approving review and passing CI.

- Issue management: issue templates for bug, feature, and documentation; labels for priority and component (frontend/backend/ops).
- CI/CD: automatic test runs on PRs; deployments to a staging environment on merges to `develop` (optional) and production on `main` after review.
- Code reviews: enforce linting and basic tests; use GitHub Actions for automated checks (linters, tests, security scanning).

8 Project Decomposition and Scheduling

- We will use GitHub Issues and GitHub Projects to manage work and track progress. Issues will represent tasks, bugs, and feature requests; Projects will be used for sprint planning and high-level roadmaps.
- GitHub Projects board (create and link when available): Project board placeholder
- Capstone Repository: https://github.com/zheniasigayev/MES-Finance-Tracking
- MES Website Repository: https://github.com/McMaster-Engineering-Society/MES-Website-App-Router

The following are the key dates and deadlines for completion of different milestones for this project:

- **09-22** – Project plan, development plan, proof of concept plan, team charter
- **10-06** – Software Requirements Specification + Hazard Analysis
- **10-27** – Validation and Verification Plan
- **11-10** – Design Doc Revision -1
- **11-17** – Proof of concept demos start
- **11-28** – Proof of concept demos end
- **01-19** – Design Documentation due
- **02-01** – Revision 0 presentation starts
- **02-13** – Revision 0 presentations end
- **03-09** – Verification and Validation Report due
- **04-06** – Final Documentation Due

This allows us to break down the development of the MES Finance Tracker into the following stages.

Phase 1 – Scoping (September 22 to October 27)

During this time, we will work to refine our high-level goals into a comprehensible list of requirements. We will be conducting interviews as well as experimentations with existing processes to determine what we can and should address within our project. At the end of this phase, we should have a concrete plan of how we will approach the rest of the term by producing a requirements specification, hazard assessment, and a validation and verification plan.

Phase 2 – Prototyping (October 27 to November 17)

In this phase, we will start initial development on our application, focusing on the features mentioned in our proof of concept plan. The goal of this phase is for our team to ensure our core features are reasonable and achievable by us, and to gather feedback regarding our features.

Phase 3 – MVP (November 17 to February 1)

During this phase, the bulk of the development work needs to be done. The prototype developed in the previous phase should be improved to become a holistic user experience and all the requirements outlined in the SRS should be addressed. During the demo, we should be aiming to collect valuable feedback to improve the project.

Phase 4 – Finalizing (February 1 to April 6)

The goal of this phase is to polish what was built in the previous phase by iterating on feedback. However, we should also be aiming to optimize towards goals outlined in previous phases and work on stretch goals.

9 Proof of Concept Demonstration Plan

Two main high-risk functionalities have been identified regarding the success of our project that, if not completed, severely hinder achievement of project goals:

1. **Financial data extraction from photos and screenshots:**

   Importance Although manual data input will be supported, screenshots and images of financial data provides a much more efficient manner of data entry. As streamlining the financials managing process is one of the project's main goals, this is a functionality of high importance.

   Difficulty of Addressal The proposed input space includes physical photos: receipts, printed statements, as well as digital formats: screenshots of statements, account balances , etc. Although the narrow context (financial data in alphanumerical format) reduce the difficulty of addressing the risk, the wide medium of formats may prove a technical challenge.

2. **Integration with McMaster Engineering Society (MSE) Monorepo:**

   Importance As the MSE is the client body, integration with their existing systems to ensure long-term supportability is critical.

   Difficulty of Addressal As of the planning stage of the project, the unknown nature of the monorepo, presents potential difficulty. Although many group members already have experience integrating with company and other third party repositories, integration still requires thorough communication with MSE representatives and team supervisors.

3. **User Authentication:**

   Importance As the primary users of the applications will be independent clubs, data must be segregated between users, and users must be authenticated to ensure security and privacy of financial data.

   Difficulty of Addressal Authentication logic is likely the most technically challenging part of the project to implement: the difficulty of photo processing comes, to an extent, from tuning to improve the success rate of a chosen method. Authentication, however, involves implementing existing, but technically complex, systems.

A series of minimum Proof of Concept (PoC) functionalities have been proposed to address the risks outlined above:

1. **Text Extraction from Receipt Photos:** Physical print representations of financial data are more noisy than digital screenshots, and are more difficult for both traditional computer vision and machine-learning based computer vision techniques to extract data from. Receipts represent one of the potentially more difficult mediums, and is chosen as a representative case for the PoC.

2. **Basic WebApp hosted within MSE Monorepo:** As part of the project, the MSE is providing the group with rights and access to two (2) existing codebases to expand on. To reduce the magnitude of risk involved with integrating into the MSE's repository, integration will be prioritized as part of the PoC to minimize conflicts and sunken-cost efforts.

3. **User authentication MVP:** As the base functionality for user authentication comprises most of what is required for the project, developing an independent authentication system will be included as part of our PoC functionalities. Integration with the rest of the system is however not yet required.

10 Expected Technology

- **Version Control**: Git, GitHub

- **Specific programming language:** JavaScript with Node.js
- **Specific libraries:** TypeScript, React.js, Next.js, Material UI, react-query, Axios, NextUI, Tailwind CSS, next-auth
- **Specific linter tools:** ESLint, Prettier
- **Specific unit testing framework:** Jest, React Testing Library
- **Investigation of code coverage measuring tools:** Jest
- **Specific plans for Continuous Integration (CI):** GitHub Actions
- **Specific performance measuring tools (e.g., Valgrind), if appropriate:** React Dev Tools
- **Tools you will likely be using:** VSCode with the following extensions; Error Lens, Tailwind CSS Intellisense, Babel JavaScript, Auto Close Tag, Auto Rename Tag, Better Comments

11 Coding Standard

General Principles

- Write clean, readable, and maintainable code.
- Prefer clarity over cleverness.
- Follow the principle of "fail fast, fail clearly".
- All code must be reviewed via pull requests before merging.

Version Control (Git & GitHub)

- Branching: use `main` as the stable branch.
- Feature branches: `feature/<short-description>`.
- Bugfix branches: `fix/<short-description>`.
- Commit messages follow **Conventional Commits** specification

Language & Frameworks (JavaScript / TypeScript / Node.js)

- Use **TypeScript** for code.
- Use `const` and `let` instead of `var`.
- Prefer `async/await` over `.then()`.
- Strongly type props, states, and API responses.
- Avoid `any`; use strict typing.
- Use `===` over `==`

Naming Conventions

- Components should be in PascalCase.
- Variables and functions should be in camelCase.
- Constants should be in UPPER_SNAKE_CASE.
- Files should be in kebab-case.

Formatting & Style

- Follow ESLint + Prettier rules.
- Indentation: 4 spaces.
- Line length: maximum 100 characters.

React / Next.js Guidelines

- Function components with hooks (`useState`, `useEffect`, etc.).
- Use `react-query` for data fetching, not raw Axios calls in components.
- Use Axios in a dedicated service layer (e.g., `services/api.ts`).
- Styling:
  - Prefer Tailwind CSS for layout and utility classes.
  - Use Material UI / NextUI for consistent UI components.
- Co-locate component-specific styles and tests with the component.

Error Handling & Logging

- Always handle API errors with `try/catch`.
- Use centralized error boundaries in React where needed.
- Do not use console logging in production.

Testing (Jest & React Testing Library)

- Minimum 80% code coverage (tracked via Jest).
- Unit tests for services, hooks, and components.
- Naming: test files mirror component/service name (e.g., `component.test.tsx`).

Continuous Integration (GitHub Actions)

- All pull requests into `main` branch must pass:
  - ESLint + Prettier checks.
  - Jest unit tests with coverage reports.
  - Build validation (`next build`).

Tools & Editor Setup (VSCode)

- Required extensions:
  - Error Lens
  - Tailwind CSS Intellisense
  - Babel JavaScript
  - Auto Close Tag
  - Auto Rename Tag
  - Better Comments
- Enable format-on-save with Prettier.
- Run `npm run lint` and `npm run test` locally before committing.

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

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

   (a) Creating a development plan prior to starting a project is crucial to provide a structured roadmap for the team to follow. This makes sure all team members are aligned on project goals, milestones, and deliverables. The plan identifies potential risks, allocate resources effectively, and establishes clear communication channels. By outlining workflows, roles, and responsibilities for the team, we can minimize misunderstandings and inefficienes. Lastly, development plans serve as a reference document to track progress throughout the project's lifecycle.

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

   (a) CI/CD allows for faster development and deployment cycles, as well as improved collaboration among team members. It streamlines the process of integrating code changes and performing manual update tasks such as updating pdf's after changing it's corresponding Tex file. It also helps to catch bugs and issues early in the development process, reducing the risk of introducing errors into production.

   (b) However, setting up and maintaining a CI/CD pipeline can be complex and time-consuming. It requires a certain level of expertise and resources to implement effectively. Additionally, if not properly configured, CI/CD can lead to increased build times and resource consumption.

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

(a) The only disagreement that we've had up until this deliverable is deciding which project to choose. We had a multitude of ideas, including both personally thought of projects and those from the pre-approved list. We resolved this by having a vote and discussion about the pros and cons about each so we can all be satisfied with the final decision. In the end we decided choose the MES Finaicial Tracker project as it had a large enough scope, technically challenging for us to all learn something, and a well outlined project objective.

Appendix — Team Charter

External Goals

Our team's external goals for this project are:

- To develop new technical skills with a variety of software tools.
- To produce a high-quality, useful product for the McMaster Engineering Society and its clubs.
- To successfully complete the course and achieve a high grade.

We have selected these as our external goals because they align with our personal and professional development objectives, all while positively impacting the MES community. We think that the completion of this project will also be an achievement we can share with future employers

Attendance

Expectations

Team members are expected to:

- Attend scheduled meetings and be on time.
- Notify the Project Manager and meeting chair in advance if they cannot attend.
- Act professionally and responsibly when absent (review minutes and follow up on assigned actions).

Preferred notification is the day before; same-day notification is acceptable for emergencies.

Acceptable Excuse

Any one of the following reasons that McMaster University recognises, will also be accepted. Below is a list of McMaster's acceptable extenuating circumstances:

- **Medical issues:** a physical or mental health condition that negatively affected participation or performance. Provide a physician's note or equivalent documentation when available.
- **Family crises:** significant personal or family situations (e.g., bereavement, serious illness) that create hardship.
- **Other personal emergencies:** unforeseen and significant events that disrupt academic or project commitments.

- **Religious observances:** conflicts with religious obligations where accommodation is required.
- **Varsity reasons:** official university sports commitments.
- **Business commitments (part-time students):** job-related duties that prevent attendance.

For urgent or sensitive circumstances where documentation cannot be produced immediately, notify the Project Manager promptly and follow up with documentation as soon as reasonably possible. Trivial reasons (e.g., social events) are not considered acceptable without prior agreement.

In Case of Emergency

If a team member has an emergency and cannot attend a meeting or complete agreed work, follow these courteous steps:

1. Notify the Project Manager and meeting chair as soon as possible (direct message and a short message in the team chat). If unavailable, notify any team lead.
2. Create a short GitHub Issue titled "emergency: ¡YourName¿" describing current status, outstanding tasks, and the expected return time (if known). Link related issues or PRs.
3. Where appropriate, assign a backup or delegate specific issues/tasks to another team member and update issue ownership. Use the handover checklist in the issue description.
4. Share any quick access notes or pointers needed to continue work (location of credentials, relevant commands, how to run tests). Do not post sensitive credentials in chat — use the team's secure credential store or inform the Project Manager privately.
5. If the absence affects an upcoming demo or deadline, request an immediate short meeting or asynchronous decision from the Project Manager about scope adjustments or re-assignment.
6. When able, provide a brief follow-up update and, if requested, supporting documentation. On return, do a short handover with whoever covered your work.

These steps prioritise clear communication and minimise disruption while respecting the privacy of the person experiencing the emergency.

Accountability and Teamwork

Quality

We expect all team members to come prepared for meetings having reviewed information relevent to the meeting. Deliverables should satisfy course requirements and stakeholder requirements, while also promoting clarity and completeness. Team members are encouraged to seek feedback from other members if they are stuck or unsure about the quality of their work.

Attitude

We expect all team members to maintain a positive and respectful attitude towards each other. This includes being open to different ideas, actively listening during discussions, and providing constructive feedback. We will adopt a code of conduct that promotes inclusivity, respect, and professionalism. In case of conflicts, we will address them promptly through open communication and, if necessary, involve the Project Manager or a neutral third party to mediate.

We awant to promote an environment where new and unique ideas are given the space to be heard and considered. We will strive to create a collaborative atmosphere where all team members feel valued and motivated to contribute their best work.

Stay on Track

We will use regular check-ins during meetings to monitor progress and address any issues early. Members will also use the course google calendar to stay informed about deadlines. There will also be clear communication channels where reminders will be sent out for upcoming deadlines or meetings.

Effort will be measured through attendance, participation in meetings, completion of assigned tasks, and quality of work. We will use GitHub metrics (issues closed, commits made) to track contributions.

Rewards for members who perform well may include public recognition during meetings, positive feedback in peer evaluations, and opportunities to take on leadership roles within the team.

Poor performance will be addressed through private conversations to understand any underlying issues.Consequences for not contributing may include reassignment of tasks, reduced responsibilities or consulation with the TA or instructor if necessary.

Team Building

To build team cohesion, we will schedule regular social activities outside of work meetings, such as dinners or casual hangouts. We will also celebrate completion of milestones and achievements together. Additionally, we will encourage open communication and support among team members to create a positive and inclusive team culture.

Decision Making

We will aim for consensus in decision-making, ensuring that all team members have the opportunity to voice their opinions. If consensus cannot be reached, we will use a majority vote to make decisions. In case of disagreements, we will encourage open dialogue to understand different perspectives and find common ground. If necessary, we may ask the TA for their input.