Development Plan
MES-ERP

Team #26, Ethical Pals
Sufyan Motala
Rachid Khneisser
Housam Alamour
Omar Muhammad
Taaha Atif

Table 1: Revision History

Date	Developer(s)	Change
Sept 24th, 2024	Housam & Taaha	Completed the confidential information, IP info, Copyright, Team plan and Workflow plan
Sept 24th, 2024	Omar, Rachid	Changes to part 8, 9, Appendix
April 4, 2025	Sufyan	Revised based on rubric feedback: Added detail to Meeting Plan, Roles, POC Demo Plan (linking to risks), and Team Charter. Clarified Confidential Info section purpose.

Introduction

This Development Plan outlines the key aspects of our, the MES (McMaster Engineering Society) Custom Financial Expense Reporting Platform. The document is structured to provide an overview of the project's goals, workflow, and expected technologies. It includes sections on confidential information handling, intellectual property considerations, copyright licensing, team roles, and communication plans. Additionally, the plan covers our coding standards, Git workflow, issue management, and CI/CD implementation.

The document also provides a project decomposition and scheduling plan, along with a proof of concept demonstration plan designed to validate core functionality and address potential risks early in the development cycle. Finally, the appendices include reflections on the importance of a development plan, the advantages and disadvantages of CI/CD, and our team charter, which outlines our external goals, attendance expectations, accountability, and decision-making processes.

1 Confidential Information

This section outlines the types of sensitive information the MES-ERP system will handle, informing team members about data sensitivity during development. While the primary mechanism for protecting this information involves formal Non-Disclosure Agreements (NDAs) signed with the MES, understanding the nature of the data is crucial for implementing appropriate security measures within the application itself.

This project deals with a large amount of ~~confidentatl~~ confidential information related to the financials of all the MES clubs and organizations. Such details must maintain confidentiality and can only be shared within MES and student groups. All group members will sign an NDA provided by the MES before gaining access to said information. Confidential information includes, but is not limited to, the following:

- **Expense Receipts**: Detailed records of all expenses incurred by the MES clubs and organizations, including receipts for purchases, travel expenses, and other financial transactions.
- **Budget Reports**: Comprehensive reports outlining the budget allocations, expenditures, and financial planning for various MES activities and events.
- **Financial Statements**: Detailed financial statements that provide insights into the financial health of the MES, including balance sheets, income statements, and cash flow statements.
- **Membership Fees and Dues**: Information on membership fees collected from MES members, including payment records, dues schedules, and any outstanding balances.

- **Event Financials**: Detailed financial records for events organized by MES, including income from ticket sales, sponsorships, and expenses related to event logistics and execution.
- **Vendor Contracts**: Agreements with vendors providing goods and services to MES, including payment terms, service agreements, and financial obligations.
- **User Information**: Personal details like names, emails, phone numbers, and potentially banking information for reimbursements.

2 IP to Protect

No IP to protect. All current software projects by the MES are publically available at https://github.com/McMaster-Engineering-Society

3 Copyright License

This project will follow a proprietary licensing agreement, which can be found in the LICENSE file.

4 Team Meeting Plan

We will hold team meetings twice a week, either online or in person, based on availability and need. Meeting times will be coordinated weekly via Discord poll or discussion to accommodate varying schedules. Advisor meetings will take place monthly through Teams unless urgent issues arise. In such cases, meetings may be held in person, depending on the severity of the matter and in consultation with the advisor.

4.1 Meeting Structure

4.1.1 Team Meetings

Each team member will share updates on completed tasks and current work. During this time, others can ask questions or provide feedback. After updates, we will discuss potential next steps and address any concerns as a group. Finally, each member will have an opportunity to raise questions or seek support on any challenges they are facing.

4.1.2 Advisor Meetings

We will begin by providing an overview of the project's progress and potential next steps. The advisor will then have the opportunity to raise any concerns or provide guidance. We will conclude by discussing any questions or clarifications needed regarding the project.

4.2 Meeting Roles

The role of meeting chair will rotate among team members. Prior to advisor meetings, the team will collaboratively decide on the agenda and compile a list of questions and concerns to be addressed.

5 Team Communication Plan

In terms of team communication for this project, we will use ~~d~~Discord as the main ~~messaging platform~~ channel for informal discussion, quick questions, and meeting coordination. Discord will be used for messaging between the group members, to plan meetings, talk about issues, share resources and concepts. In addition, all group members will stay up to date on the GitHub ~~issues~~ Issues section, ~~to track any issue with their commits and be~~ which will serve as the primary tool for tracking specific tasks, bugs, and feature progress. Team members are required to address (fix or simply respond) ~~all their issues.~~ issues assigned to them in a timely manner. Additionally, any merge conflicts or issues pushing or pulling code will be notified to team members via Discord or~~, if urgent,~~ through phone call if urgent.

6 Team Member Roles

We have assigned the following ~~roles~~ procedural roles to team members, detailing their responsibilities. Every month we will meet back to adjust the roles of each person as needed according to what people want to do and what their strengths are. In addition, group members can request a vote to switch to another role at any time.

- **Meeting Chair: Taaha Atif**
  - Organizes and leads team meetings.
  - Prepares the meeting agenda and ensures all topics are covered.
  - Facilitates discussions and ensures that meetings run smoothly and on time.
- **Notetaker: Rachid Khneisser**
  - Takes detailed notes during meetings.
  - Distributes meeting minutes to all team members after each meeting.
  - Ensures that action items and decisions are clearly documented.
- **Reviewer: Sufyan Motala**
  - Reviews all project documents and deliverables for accuracy and completeness.
  - Provides feedback and suggestions for improvements.

    - Ensures that all work meets the project's quality standards.

- **Notetaker/Helper: Omar Muhammad**
    - ~~Reviews all project documents and deliverables for accuracy and completeness (with Rachid).~~ Assists the primary Notetaker during meetings.
    - Talks to anyone else in the group to see if they need assistance for any work and helps coordinate support.
- **Leader: Housam Alamour**
    - Oversees the overall progress of the project.
    - Coordinates tasks and ensures that deadlines are met.
    - Acts as the main point of contact between the team and the advisor.

    - Facilitates the distribution of technical tasks.

In addition to these procedural roles, technical tasks related to frontend development, backend logic, database management, testing, and documentation will be distributed based on member expertise, interest, and project phase needs, coordinated by the Leader.

7 Workflow Plan

7.1 Git Usage

Git will be used to pull/push code (share code) between the group. The following practices will be followed:

- **Branches**: Each feature or bug fix will be developed in its own branch. Branch names will follow a consistent naming convention, such as `feature/feature-name` or `bugfix/issue-number`.
- **Pull Requests (PRs)**: Pull requests will be used to merge changes from feature branches into the main branch. PRs will require at least one peer review before being merged to ensure code quality and adherence to project standards.
- **Commit Messages**: Commit messages will be descriptive and follow a consistent format, detailing what changes have been made, what files are affected, and why these changes are necessary.
- **Tags**: Tags will be used to mark significant milestones and releases in the project. Tags will follow a semantic versioning format, such as `v1.0.0`.
- **Issue Tracking**: Issues will be tracked using a project management tool like GitHub Issues. Issues will be categorized based on their type (e.g., bug, feature request) and priority (e.g., high, medium, low).

- **Project Management Tools**: We will use GitHub Projects to organize and manage tasks, track progress, and ensure that all team members are aligned with the project goals.

7.2 Issue Management

Issues will be managed to ensure smooth progress and timely resolution of problems:

- **Issue Templates**: Templates will be used for creating new issues to ensure all necessary information is provided. Templates will include sections for description, steps to reproduce, expected behavior, and actual behavior.
- **Issue Classification**: Issues will be classified based on their type (e.g., bug, feature request, enhancement) and priority (e.g., high, medium, low).
- **Assignment**: Issues will be assigned to team members based on their expertise and availability. Team members can volunteer for issues or be assigned by the project manager.

7.3 CI/CD Implementation

Continuous Integration and Continuous Deployment (CI/CD) will be implemented to maintain code quality and streamline the development process:

- **Auto-formatting**: On commit, code will be auto-formatted to adhere to the project's coding standards.
- **Linting**: Linting will be performed via CI/CD to catch syntax and style issues early.
- **Static Analysis**: Static analysis will be conducted on committed code to identify potential issues and improve code quality.
- **Automated Testing**: Automated tests will be run on each commit to ensure that new changes do not break existing functionality.

8 Project Decomposition and Schedule

Our project will be scheduled as such:

Requirements Document Revision 0	October 9
Hazard Analysis 0	October 23
V&V Plan Revision 0	November 1
Proof of Concept Demonstration	November 11–22

Design Document Revision 0	January 15
Revision 0 Demonstration	February 3–February 14
V&V Report Revision 0	March 7
Final Demonstration (Revision 1)	March 24–March 30
EXPO Demonstration	April TBD

9 Proof of Concept (POC) Demonstration Plan

The Proof of Concept (POC) demonstration ~~will be a critical part of our project, showcasing~~ aims to demonstrate the feasibility and core functionality of our proposed solution while explicitly addressing key project risks identified early on.

9.1 Key Risks Addressed by POC

The primary risks this POC aims to mitigate are:

- **Core Workflow Viability:** The risk that the fundamental process of user authentication, request submission, and administrative review is technically infeasible or overly complex within the chosen tech stack (Next.js/Supabase).
- **Role-Based Access Control (RBAC) Implementation Risk:** The risk that implementing basic role separation between standard users and administrators is fundamentally difficult or cannot be securely achieved with Supabase Auth.
- **Data Handling and Persistence Risk:** The risk that submitted request data cannot be reliably stored, retrieved, and updated in the database.

While the original plan mentioned risks related to handling diverse receipt inputs, this POC focuses first on validating the core digital workflow before tackling input variability.

9.2 Demo Plans

For our upcoming Proof of Concept (POC) demo, we will showcase the core functionalities of the website. The goal is to demonstrate the key user flows and interactions while keeping the design straightforward and focused on functionality. This demonstration directly addresses the identified risks by proving the viability of the core workflow and basic RBAC. The demo will highlight the following features:

- **Authentication System:** Users will be able to register, log in, and be assigned a role (user or admin) to access their respective dashboards.

- **User and Admin Dashboards:**
  - **User Dashboard:** The user will have access to a form for submitting reimbursement requests. Once a form is submitted, the user can view the status of their request (initially 'Submitted').
  - **Admin Dashboard:** The admin will be able to view all reimbursement requests submitted by users. Admins can update the status of each request (e.g., change 'Submitted' to '~~in progress, accepted, rejected~~ Approved' or 'Rejected'), and these status changes will be reflected on the user's dashboard.

The planned demo flow will involve: (1) Registering a new user account. (2) Logging in as the user and submitting a sample request. (3) Logging out and logging in as an admin. (4) Viewing the submitted request on the admin dashboard. (5) Changing the request status (e.g., to 'Approved'). (6) Logging out and logging back in as the original user to show the updated status. This demo will serve as a basic proof of concept, focusing on demonstrating essential functionalities without incorporating overly complex elements ~~. The emphasis is on ensuring that the primary user interactions, such as registration, dashboard navigation, request submission, request review, and request approval/denial work as intended~~. This will provide a solid foundation for future development and refinement as the project progresses.

9.3 Next Steps and Feedback

As this is a preliminary demonstration, we aim to gather feedback on whether the functionalities outlined above meet the requirements and if any improvements are needed. Any suggestions for enhancements, as well as additional features that may be valuable for the next phase, will be highly appreciated.

9.4 Link to GitHub Project:

https://github.com/Housam2020/MES-ERP

10 Expected Technology

Please note that any specific technologies referenced in this section are merely suggestions and exist mainly as an illustrative example for what the potential product used for each section could be and what it's capabilities should be. These options are not fixed and are subject to be changed as the project continues to be developed.

- **Specific programming language:**

  - **TypeScript**: For this project, we will be using Typescript as it has been set out as a constraint by the supervisor for consistency reasons as many of the previous MES tools use this language. However, its benefits are that it provides static typing, which helps catch errors early and improves code quality. Additionally, it integrates well with Next.js, the other requirement for this project.

- **Specific libraries:**
  - First, we have a constraint that we must use Next.js imposed by the supervisor for similar reasons above as to why Typescript will be used. Our team wants to choose libraries that facilitate the development of a web application, as that is the main goal of our project. We are looking for libraries that help with server-side rendering, static site generation, and routing. We also need libraries that will help us sort through the old financial information as part of this project involves reading and parsing the old Excel documents that were used to keep track of financials for clubs in the past.

- **Pre-trained models and Machine Learning Technologies:**
  - TensorFlow or PyTorch: Frameworks for utilizing pre-trained machine learning models for the machine learning based expense categorization stretch goal.

- **Specific linter tool:**
  - The team will pick a linter tool (ESLint) to ensure code quality and consistency that will help to identify and fix problems in the codebase. It helps identify and fix problems in TypeScript code.

- **Specific unit testing framework:**
  - A unit testing framework such as Jest will be chosen to test the individual components, ensuring that each part of the application functions correctly.

- **Investigation of code coverage measuring tools:**
  - Code coverage tools will be used to measure the percent of the codebase tested, showing the team how to improve test coverage and code quality. This may be potentially done with Jest and GitHub actions. This will help ensure that all parts of the codebase are tested.

- **Specific plans for Continuous Integration (CI):**
  - Continuous Integration (CI) tools will be implemented to automate the testing and deployment process. This will help ensure that code changes are continuously integrated and tested. GitHub Actions has most of the features needed for our development as it automates the testing and deployment process.

- **Specific performance measuring tools:**
  - While not likely that we will need specific performance measuring tools for a web application, as this is not quite a real time system, we still want the tool to be responsive, fast and easy to use for all users. For this reason we will mostly be basing our performance tests off how long pages take to load on different hardware, and if needed we will utilize other tools like Lighthouse to assess the efficiency and speed of the application, identifying areas for optimization.
- **Tools you will likely be using:**
  - We will choose development tools and editors to increase productivity and streamline the development process. We want good "return on investment" for our time to implement and learn how to use any tools in this project. Thus, we will mainly be using tools already used by our team members. This includes Visual Studio Code: This code editor will be used for development. It offers a wide range of extensions and integrations that enhance productivity. We will also extensively be using GitHub and GitHub projects as part of this effort for version control.

11 Coding Standard

We will adopt the following coding standards to ensure code quality and maintainability:

- **Language-Specific Guidelines:**
  - **TypeScript**: Follow official TypeScript guidelines and leverage strict type checking to ensure type safety and consistency.
- **Code Formatting:**
  - Use Prettier for consistent code formatting through the codebase, enforced via pre-commit hooks and CI checks.
- **Commenting and Documentation:**
  - Add inline comments for logic that is complex and needs explanation. Use JSDoc comments for functions and components where appropriate. This will help to improve code readability.
- **Version Control:**
  - Follow the Git branching model (e.g., Gitflow or GitHub Flow) for feature development and bug fixes.
  - Use descriptive commit messages, ~~this can be done by~~ ideally following the Conventional Commits specification.

- **Code Reviews:**
  - Require peer reviews (at least one approval) for all code changes via Pull Requests to ensure quality and adherence to standards.
- **Testing:**
  - Write unit tests (using Jest) for each major feature ~~using a set testing framework~~.
  - Monitor code coverage to ensure an adequate amount of code coverage has been reached (currently set to 90%).
- **Variable Naming:**
  - Use camelCase for variable names and function names. Use PascalCase for component and type names. ~~to ensure consistency and readability.~~ For example, use `totalAmount` instead of `total_amount`.

Appendix — Reflection

This appendix was discussed and completed as a group.

1. Why is it important to create a development plan prior to starting the project (answered as a group)? It is important to create a development plan prior to the project because it can be used to guide us during our project. A development plan can ensure we stay on track and meet specific criteria/checkpoints throughout the timeline of a project. It also reduces confusion since you will know generally what your agenda will look like for upcoming weeks.

2. In your opinion, what are the advantages and disadvantages of using CI/CD (answered as a group)? Using CI/CD has many advantages such as automating the integration, testing, and deployment of code. This saves time by alerting us on issues early on and allows us to more confidently write code knowing issues will be caught. It essentially allows us to always have our project in a working state. A disadvantage to CI/CD comes from the time needed to setup and maintain it. This can take development time away from working on the project itself. Some form of computational costs will also be included to verify that the code works.

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them (answered as a group)? We had disagreements with team member roles, some students have a larger course load this semester, so to keep it reasonable we gave people with less courses this semester slightly more work to do (but we will try to keep everything even if possible). We resolved this by evaluating all of our course work and trying to remain fair to all teammates, we planned and talked the course through and how we would plan it. So we decided to compromise on certain parts and were able to split the work in a way the whole group was happy with and in a way that is fair to the project itself (All group member will do at least ”some” work, where some is defined as an amount of work where all group members feel like the workload was distributed evenly).

Appendix — Team Charter

External Goals

Our team's external goals for this project are to further develop our skills as software engineers and gain hands-on experience in real-world project management and development within a modern web technology stack (Next.js, TypeScript, Supabase). We aim to build a strong foundation of practical skills that we can confidently showcase in our resumes and discuss in interviews. We want to use this project to show our abilities when demonstrating our expertise and contributions to future employers. Delivering a functional and useful tool for the MES is also a key external goal.

Attendance

Expectations

All team members are expected to attend every scheduled team and supervisor/stakeholder meeting on time. If someone is delayed, they must notify the group via Discord in advance. If a member needs to leave early or miss a meeting for a valid reason, they are required to update the team on their progress via text or Discord and stay informed about the work and progress of others by reviewing meeting minutes or catching up with a teammate.

Acceptable Excuse

Valid reasons for missing a meeting or deadline include family emergencies or severe illness that prevents the individual from working. Excuses like having a midterm or an upcoming assignment will not be considered acceptable, as these should be planned for in advance.

In Case of Emergency

In the event of an emergency preventing a member from completing critical work, the work ~~that was initially assigned to the affected team member~~ will be ~~equally~~ distributed amongst the others based on availability and relevant skills, to ~~be completed.~~ ensure project progress is not significantly hindered. The team member that was unable to complete their work last time ~~will~~ may have a slight increase in work for the next deliverable to make up, discussed and agreed upon by the team.

Accountability and Teamwork

Quality

Team members are ~~to be~~ expected to ~~have questions prepared for team meetings as well as~~ come prepared to team meetings with updates on their progress, potential blockers, and questions for the group. They should have a detailed understanding of what they are currently working on and what they have completed. This information is crucial as it will be required ~~to present~~ for discussion during each team meeting. The quality of the deliverables that members bring to the team should be detailed and well done. The team is expected to look at the rubric and project requirements while completing their portions for the deliverables. Code contributions should adhere to the agreed-upon Coding Standard and pass CI checks.

Attitude

All team members are expected to maintain a welcoming and open-minded attitude towards all ideas and suggestions. While discussions on the effectiveness of an idea are encouraged, no idea should be dismissed outright without constructive reasoning. Everyone is expected to collaborate and contribute to the task at hand. If conflicts arise, members involved are expected to communicate their concerns openly and respectfully, ~~and~~ work together to find a resolution as quickly as possible.

Stay on Track

To keep the team on track, we will hold weekly meetings to review current progress and outline upcoming tasks using our GitHub Project board. Each member will provide a brief summary of their contributions for the week. Members who fall behind compared to their peers will receive warnings and offers of support from the team. ~~and be encouraged to meet the group's standards.~~ Persistent lack of contribution, after attempts to resolve the issue internally, will be discussed with the Leader and potentially ~~Those who do not contribute at all will be~~ reported to the professor or TA.

~~We will set a target number of commits for each team member. At the end of the month, the member with the highest number of commits will be rewarded with a free meal of their choice.~~

Team Building

We will ~~have~~ plan bi-weekly meetups where we ~~go for food or do an activity together.~~ can socialize outside of project work, such as grabbing food or participating in a group activity chosen by consensus. This aims to foster camaraderie and improve team dynamics.

Decision Making

All major technical or project direction decisions will be made through a voting process after discussion. If a member strongly disagrees with the outcome, they must present their concerns and explain why an alternative option might be preferable. A revote can then be conducted based on this new information if the group agrees the concerns warrant reconsideration. Minor decisions can be made by the assigned individual or relevant subgroup, keeping the Leader informed.