Development Plan
McMaster Engineering Society Custom Financial Expense Reporting Platform

Team #12, Reimbursement Rangers
Adam Podolak
Evan Sturmey
Christian Petricca
Austin Bennett
Jacob Kish

Table 1: Revision History

Date	Developer(s)	Commit	Change
9/24/2024	Entire Team	Pull Request 13	Initial revision, including all sections, appendix, and reflections
1/3/2025	Adam Podolak	4788818	Added GitHub Project link
1/10/2025	Austin Bennett	4831973	Revised POC Plan

Link to GitHub Project: 4G06 Capstone Project

This document outlines the development plan for the MES finance platform capstone project, providing a structured approach that will help guide us throughout this project. Starting with legal details and privacy agreements, the document also outlines a comprehensive plan for team communication and member roles. Additionally, we detail expected technologies and the programming languages, frameworks and tools that we anticipate using throughout the year. Finally, we reflect on team dynamics and draft a team charter that will help us maintain a professional and productive environment.

1 Confidential Information?

The project contains confidential information that we need to be mindful of. Some information we would need to protect include an individual's bank statement or account details. We are in the process of preparing a confidentiality agreement to protect this sensitive information.

2 IP to Protect

This project does not have any IP to protect.

3 Copyright License

Our team is adopting the MIT license for our project. We feel that this license suits us best with its simplicity and relatively loose restrictions. For more information, see the LICENSE document in the repository or visit https://opensource.org/license/mit

4 Team Meeting Plan

Our team plans to meet at least once a week, either in-person or virtually, depending on the preferences and circumstances of team members. Additional meetings will be scheduled on weeks when deliverables are due or at any time we believe we would benefit from more planning time.

In addition to our regular team meetings, we also plan to meet with our capstone supervisor once a week. These meetings are subject to their availability and may also take place either in-person or online.

To ensure that our meetings are efficient and productive, we have established rotating roles for all team members during meetings. One person will serve as the team liaison, one person will serve as the meeting chair and the rest will take on the role of notetakers. The responsibility of the team liaison is to set the time and place of the meeting. Their task is to ensure that all group members

are available to attend and they are to deal with any scheduling conflicts. The responsibility of the meeting chair is to lead the discussion. They will prepare a list of important topics that must be covered and ensure that we stay on topic. The notetakers are then responsible for documenting key information and keeping track of topics that should be revisited in future meetings. By the end of the meeting, roles will be redistributed and the next meeting will be tentatively scheduled.

5 Team Communication Plan

Team communication plan includes regular availability for communication via Discord chat and calls. This is ideal for quickly sharing ideas and questions and checking statuses of documentation, commits, pull requests, etc. GitHub Issues will be used to keep track of work items and provide traceability by linking PRs to Issues. That said, commit messages will still be used to provide clarity when updates are made. GitHub Projects, especially the Kanban boards, will also help keep track of assignments and progress in a convenient visual way. Finally, as discussed in section 4, in-person and/or virtual (Teams or Discord) meetings will be used to actively discuss major issues, progress, and blockers with the whole team.

6 Team Member Roles

Roles are not currently assigned but several will be used by the team. If the team notices more regular tasks that are needed to keep the project running smoothly, additional roles can be discussed and added.
The first role is notetaker, which would involve taking meeting notes at least during meetings with project heads or stakeholders and possibly during internal team meetings as well. The objective is to record key points discussed and feedback from valuable resources as a reference, without relying on memory to implement the knowledge gained. Notetaker should also record the time and place of meetings to provide a timeline of progress.
Another role is the meeting chair, who would lead discussions with stakeholders, product owners, etc. The chair would have an agenda of items and help keep meetings focused and on track. Major points of discussion should be brought up by the chair, rather than by picking team members arbitrarily.
The next role is the administrator, who would review and submit milestones on Avenue assignments and perform final pull requests and merges on GitHub. This provides a single resource for major changes to the project, preventing confusion regarding that responsibility and making a single person clearly accountable for one of the most important project tasks.
The last role is team liaison, who would be responsible for messaging and setting up meetings with external individuals including stakeholders and product owners. The team liaison role would create a single point of contact for outside

parties that would streamline the communications process and provide accountability.
We anticipate that roles can be assigned by several factors, including mainly personal preferences and skill sets (e.g., attention to detail would be an important skill for an administrator). Roles could also change if team members feel they are not suited for the role; in the case of team liaisons or meeting chairs this update should also be communicated with external stakeholders. Any role assignment or reassignment would hopefully be done on a voluntary basis, but if necessary a vote could be held. The whole team should be in agreement when roles are handed out.

7 Workflow Plan

- Git will be an important tool to facilitate collaboration and improve efficiency of the project. Major features, bugs, and code improvements will be kept track of via Issues assigned to team members, and in turn Issues should be assigned to one of these categories. This provides accountability for who has done what and is a reference for what each member needs to do. Pull requests will be assigned to Issues to track progress.

- Similarly, GitHub Projects will help keep track of assigned tasks and current progress. The Kanban board especially will help facilitate this, providing a visually clear method of doing so. The board also allows tasks to be displayed with their statuses (open, in-progress, etc) and so provide that valuable information.

- Pull requests should be reviewed by at least one, but preferably two reviewers, and preferably reviewers who have worked on similar or connected issues. After a pull request automatic checks and tests will be done to ensure nothing is broken by the new change.

- It is also the expectation that all commits have clear messages detailing what has been changed and what Issue is being addressed.

- Major feature updates will be handled by creating new branches to work on the features, so in case of major issues the main branch can be used to fall back on a viable product.

- Version tags would be used to identify what milestone is being worked towards, or possibly each major feature would entail its own version.

- CI/CD will be used to automatically compile Latex documents when a commit is made and pushed to the main branch. CI/CD may also be used in the future to streamline pushing updates to the live production environment.

- Issues will be used to track team contributions pertaining to meeting and lecture attendance and track TA and Peer feedback on deliverables and

documents. There will be issue templates for lectures, tracking lecture attendance and questions asked. Similarly, there will be templates for supervisor, TA, and team meetings. Finally, there will be templates for peer reviews, and TA feedback on deliverables from the Avenue to Learn rubrics.

8 Project Decomposition and Scheduling

Scheduling

The project will be broken down into high-level phases or milestones, grouped by each deliverable. (This schedule is subject to change as we understand more about the project scope.)

Phases & Deliverables:

Phase 1: Team Formation and Planning

- **Team Formed, Project Selected**
  Date: September 18
- **Problem Statement, POC Plan, Development Plan**
  Date: September 25

Phase 2: Requirements and Analysis

- **Requirements Document Revision 0**
  Date: October 4
- **Hazard Analysis**
  Date: October 20

Phase 3: Verification and Validation Planning

- **V&V Plan Revision 0**
  Date: November 3
- **Proof of Concept Demonstration**
  Date: November 13–24

Phase 4: Design and Development

- **Design Document Revision 0**
  Date: January 17

Phase 5: Demonstration and Review

- **Revision 0 emonstration**
  Date: February 5–February 16
- **V&V Report Revision 0**
  Date: March 6

Phase 6: Final Preparation

- **Final Demonstration (Revision 1)**
  Date: March 18–March 29
- **EXPO Demonstration**
  Date: April TBD

Phase 7: Final Documentation & Complete Deliverables

Final Documentation (Revision 1)
Deliverables:

- Problem Statement
- Development Plan
- Proof of Concept (POC) Plan
- Requirements Document
- Hazard Analysis
- Design Document
- V&V Plan
- V&V Report
- User's Guide
- Source Code

9 Proof of Concept Demonstration Plan

What is the main risk, or risks, for the success of your project? What will you demonstrate during your proof of concept demonstration to convince yourself that you will be able to overcome this risk?

Identified Risks

As a group, we have identified some potential risks specific to establishing a POC demonstration. They are as follows:

- **Integration Failure:** APIs and components may fail to communicate correctly during the demo, leading to incomplete functionality being showcased.
- **Performance Issues:** The POC system might lag or crash under demo conditions due to unoptimized configurations or resource limitations.
- **Demonstration Environment Problems:** Unforeseen technical issues might arise when setting up the POC demo environment, such as hardware or software incompatibilities.

Mitigating Risks

To address these risks during the proof of concept (POC) demonstration, we will focus on the following key areas:

1. **Technical Feasibility**
    - **Functionality Showcase:** Demonstrate core features such as user registration and financial transaction management to validate integration between the frontend and backend.
    - **Pre-Demo Testing:** Conduct comprehensive pre-tests in the demo environment to identify and resolve potential issues in advance.
2. **Performance Assurance**
    - **Resource Optimization:** Use a lightweight version of the database and ensure efficient configurations to handle typical demo scenarios smoothly.
3. **Demonstration Environment Setup**
    - **Prepared Scripts:** Develop step-by-step scripts to ensure the demonstration proceeds smoothly, even if unexpected issues arise.

By addressing these areas in the POC demonstration, we aim to reassure ourselves and stakeholders that we are prepared to mitigate risks and deliver a successful project.

10 Expected Technology

Technologies that are confirmed for the project are as follows

- Git, GitHub and Github Projects for code management

- Frontend: HTML/CSS, TypeScript + React
- Backend: Next.js
- Database: MongoDB

This is a potential list of technologies categorized by use case. None of these are confirmed but serve as a starting point for our considerations.

10.1 Programming Languages

- **TypeScript**: To be used for both frontend (React/Next.js) and backend (Node.js) development for type safety and a better development experience.

10.2 Frameworks and Libraries

- **Next.js**: The main framework for the frontend, allowing for server-side rendering, API routes, and static site generation.
- **React**: Used for building the dynamic user interface.
- **Node.js**: The backend runtime environment for executing JavaScript/TypeScript server-side.
- **Express** (optional): A lightweight web framework to handle backend routing and middleware, though Next.js API routes may suffice.
- **MongoDB**: A NoSQL database for storing financial data and user information.
  - **Mongoose**: An ODM (Object Data Modeling) library for MongoDB to handle schema definitions and interactions more easily.

10.3 Additional Libraries/Technologies

Authentication:

- **NextAuth.js** or **Auth0**: For handling user authentication and session management.

Payment Processing (if needed):

- **Stripe API**: For managing financial transactions within the platform.

Form Validation:

- **Formik** or **React Hook Form**: To ensure smooth validation for user inputs in financial forms.

State Management:

- **Redux** or **Context API**: To manage global state across the application.

Styling:

- **Tailwind CSS** or **Styled-Components**: To provide consistent and responsive UI styling.

Data Visualization:

- **Chart.js** or **Recharts**: For generating financial graphs, charts, and reports.

10.4 Unit Testing & Linting

- **Jest**: For writing unit and integration tests for both the frontend and backend components.
- **React Testing Library**: To test React components in isolation and ensure proper behavior.
- **ESLint**: To maintain code quality and adhere to coding standards. TypeScript plugins will be added to enforce types.

10.5 Code Coverage

- **Istanbul/nyc**: To measure code coverage and help identify untested parts of the code.

10.6 Continuous Integration (CI)

- **GitHub Actions**: For automating CI/CD pipelines. This will include running tests, checking linting errors, and deploying the application.
  - **Prettier**: Used alongside ESLint to enforce consistent code formatting before merging pull requests.

10.7 Performance Measuring Tools

- **Lighthouse**: For checking performance, accessibility, and best practices of the frontend.
- **Postman**: For testing API endpoints manually before they are integrated.
- **Valgrind** (optional): If memory profiling and performance are needed in specific areas of the backend.

10.8 Security and Compliance

- **Helmet.js**: To help secure the Node.js app by setting various HTTP headers.
- **OWASP Guidelines**: Follow web security best practices to ensure the platform is resistant to vulnerabilities like XSS, CSRF, and SQL Injection.

10.9 Version Control and Project Management

- **Git**: Version control for tracking changes and collaborating with team members.
- **GitHub**: For hosting repositories, managing issues, and code reviews.
- **GitHub Projects**: To manage project tasks, milestones, and progress.

10.10 DevOps & Deployment

- **Vercel** or **Netlify**: For deploying the Next.js application (frontend and backend API routes).
- **MongoDB Atlas**: To host the MongoDB database in the cloud.
- **Docker** (optional): For containerizing the application and ensuring a consistent environment across different stages.

10.11 Tools Likely to be Used

- **VSCode**: For coding, with plugins for ESLint, Prettier, and TypeScript.
- **Postman**: For testing REST APIs during development.
- **Figma** (optional): For designing the user interface and planning the visual components of the platform.

10.12 Future Considerations

- **Scaling**: Use cloud services like AWS or Google Cloud for scalability if the application grows in usage.
- **Microservices** (optional): Consider splitting the backend into microservices if the platform requires higher modularity or has different domains such as financial reporting, user management, and payment processing.

10.13 Initial Thoughts on Complexity

The platform involves a complex integration of different technologies, especially handling user data and financial transactions securely. The frontend should be intuitive for club members, while the backend should ensure the safety and scalability of financial data. As we progress, we will revisit the implementation decisions based on project challenges and feasibility.

11 Coding Standard

This project will primarily use JavaScript, so we will leverage existing linters such as ESLint to ensure code quality and consistency. To follow industry best practices, we plan to adopt the widely used AirBnB JavaScript coding standard. This will help maintain readability and uniformity throughout the codebase.

For more details on the AirBnB coding standard, you can refer to the following resource: https://github.com/airbnb/javascript.

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?**

   Christian: I believe the saying "If you fail to plan, you plan to fail" has a lot of truth to it. Not that everything can be anticipated, but even just the act of getting your mind in the habit of attempting to foresee potential problems, and methods of completing certain tasks will positively affect the trajectory of the project the earlier it is completed. Creating an initial plan is the first chance you get at realizing the project fully, and even through multiple iterations, the main objectives should remain invariant and the true form of the final product should start to take shape. Without a plan, things can still go right, but it then becomes a question of how confident can you be that the final product is truly what was needed and

not what was simply completed? Plans act as a beacon for a project.

Jacob: It is important to create a development plan because the plan will provide a roadmap that can be loosely or strictly followed depending on the needs of the project. It also gives clear markers of success and progress to evaluate the actual results against. Simply, it can be used a reference for when there is doubt on how to proceed and also can be presented to outside stakeholders to let them know what they can expect is happening behind the scenes.

Austin: A concrete development plan and especially help to avoid scope creep, in a software project its easy for the stakeholder to ask for additional "small" features that may cut into existing allocating resources. Ultimately, this can cause delays in the project or result in unintended bugs as last second features were not originally accounted for.

Evan: Just like all other projects in life, planning is a crucial step in making sure everything runs smoothly. Thorough preparation allows us to set timelines, allocate resources effectively and outline goals and responsibilities. With a group of five members, it is especially important to create a development plan document so that we are all on the same page and can work productively.

Adam: Creating a development plan can help with defining a clear objective in all aspects of the project. This project has many layers, and ironing out all the details early on will help us further down the road when we actually sit down to code this platform. It also helps with identifying any risks that we may face later on down the road. Identifying the risks early can help us isolate them and establish way to mitigate them so that we aren't scrambling when they do show up. Lastly, it helps with team coordination and team management. Now that we have a plan, it is easier to delegate tasks, making our development process more efficient.

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

Christian: One big advantage of CI/CD is the "granularity" of the approach, allowing for problems to be noticed right away and optimally in a context that is easy to rectify makes debugging for the project overall much less tedious. Problems have a harder time of piling on top of each other and becoming tangled, and you always have a deep understanding of how the different components of your system interact with each other. A disadvantage could be the other side of it's advantage, or "granularity", in the sense that it can make having a big-picture view hard to maintain.

With constant focus on the new development, one also has to consider the context of the whole project. Being able to consider the big-picture perspective of a project is important throughout development to make sure that development progress is in line with project goals.

Jacob: CI/CD is advantageous because it provides automatic checks and packages that prevent spending efforts on running the same tests or downloading the same packages many times when changes are made. It also increases reliability by preventing these pitfalls. However, the main issue with CI/CD is the overhead required to setup all of these automated efforts

Austin: Very very good because it allows frequent builds, deploys and testing, catching bugs early! Additionally, it allows a project to be easily deployed to numerous development / testing / production environments. This can ease the development process on large teams, as well facilitate adoption through easy deployment. The very very bad, is it can add a layer of unnecessary complexity to a project or even a team. Management can often throw CI/CD at every software project when it may not be necessary, on a small team or project, the CI/CD pipeline could become larger or more complex than the project itself!

Evan: An advantage of using CI/CD is the quality assurance aspect. Frequent testing and integration help to catch bugs early in the development cycle which helps save time and improves the quality of code. A disadvantage of using CI/CD is the resources it requires. Setting up CI/CD pipelines can be complex and time-consuming which can require too many resources depending on the scale of the project.

Adam: One of the advantages of CI/CD is the ability to merge code changes and have it automatically build and run tests. This makes pushing updates to the running application much easier, and developers can make edits to the code and have the updates reflected on the user side automatically. This makes development more efficient, allowing us developers to focus on the code itself and not managing releases. One disadvantage of using CI/CD is its initial setup can be quite complex. There may be a steep learning curve for myself considering I've never set up a CI/CD environment. However, putting in the work now will prove to be beneficial in the long run.

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

Christian: We did not have any disagreements for this deliverable. We don't doubt that disagreements could come up in the future, however

with respect to this deliverable, we wanted to keep an open mind and avoid making the development plan unnecessarily specific or restrictive. By doing this, we are all aware of multiple pathways different parts of the project could take, but are also willing to pivot to whatever is necessary without getting attached to anything too early. If we were to have any disagreements, we would solve them as we did in the project selection phase, which was with a democratic vote.

Jacob: There weren't any disagreements but in case of disagreement, the best way to resolve them would probably be to discuss on Discord or in person, if necessary. If the whole team needs to get involved, then it may be necessary to put the decision up to a vote. Fortunately since there are 5 members on the team, a tie vote and deadlock is impossible.

Austin: No disagreements, however just respectful communication and sharing of ideas to arrive at decisions that everyone is happy about. Moving forward approaching development conflicts this way is the most productive, hopefully this sets the routine for the group moving forward.

Evan: We did not have any disagreements during this deliverable. Early on, we established channels for clear and open communication which allowed us to get through this deliverable while facing minimal challenges. We plan to continue to communicate effectively going forward to minimize future issues.

Adam: There were no disagreements when writing this deliverable. The group brainstormed, communicated, and finalized ideas well. The group has communicated well within and outside of this deliverable and if conflicts or disagreements were to arise in the future, friendly discussions would be held to resolve them.

Appendix — Team Charter

External Goals

We viewed our external goals as our foundation to the project, as a misalignment of these goals could make cooperation unnecessarily tough later on. We agreed that a general level project that we could flesh out and polish was the best fit. Working on the MES Finance platform not only allows us to work towards those goals, but also allows us to get more experienced with start-to-finish full stack development, that more closely resembles an industry workflow, than any other course projects thus far. We are not opposed to striving for a 12 or even an award, however these would be achieved as a byproduct of our original goals and not the main priority. Given what is expected of our final product in terms of usability for the MES members, we would ultimately want to have our names on something that is simple but works as expected and mitigates pre-existing problems.

Attendance

Expectations

The expectation is that members attend all meetings, however we understand that life is unpredictable so communication is a top priority in this regard. We expect open communication in terms of weekly availability so that meetings can best fit all group members schedules, however if the situation arises we are prepared to conduct meetings with members missing in the interest of moving the project forward, assuming the missing member has communicated why they cannot attend and acknowledges they will catch themselves up as best as possible in the near future.

Acceptable Excuse

Excuses relating to extraneous circumstances will virtually always be permitted, assuming the group believes it to be in good faith. We are a relatively small group, so trust and confidence in each other is very important, especially given the scope of the project and relevance with respect to our current programming background. We want all members to feel confident being open and honest. If a group member is faced with a circumstance they couldn't reasonably predict, such as family emergencies or sickness, this is acceptable. If a group member randomly cancels their attendance with no explanation and refuses to elaborate, this is not fine and will be discussed. For example, if a group member says they woke up with a bad cough, this is clearly unpredictable so it would be okay. If a group member says that they were awake late last night hanging out with friends and as a result they are too tired to attend a meeting, this is not acceptable.

In Case of Emergency

Assuming it really is an emergency or unworkable circumstance for a given group member, the rest of the group would work to fill in any gaps and make sure that the overall project progress is not significantly compromised. As soon as we are notified, we would delegate the remaining work, set a time frame, and communicate with the missing group member what their responsibilities will be upon being able to work again. If the amount of work missing is small, there will likely be no further action, however if someone is unable to complete a significant portion of what is expected of them, the group member may be asked to complete other tasks in the future to compensate. As for missing meetings, the group members that are able to attend would gather relevant information for the missing member and relay it to them as soon as possible.

Accountability and Teamwork

Quality

We expect all group members come to meetings with either ideas for the project and inquiries related to those ideas, or questions about how to proceed in general. Meetings are times for the group to get organized, plan our future steps, and prepare for the immediate next step. If a member is confused, we ask that they ask questions during meetings to clarify. If a member has an idea, we ask that it is shared and they are open to having the idea construtively criticized. There will be no concrete expectation of what should be brought to each meeting in terms of potentially workable items. As for deliverable work, all work will be reviewed prior to submission, and we expect that all members complete their work assuming that required work from other members will be completed by the time they need it. If a member does not stay on task or develops something different than was expected, regardless of functionality, this will need to be discussed. Over the course of a year-long project, the predictabilty of group members' output is extremely valuable for planning and organization, arguably as valuable as the quality itself. Any interruptions or deviations from what is expected must be communicated immediately.

Attitude

We expect everyone to have a positive attitude and be willing to cooperate on any given day. Given that we have a level of familiarity with each other going in to the course, establishing a concrete code of conduct or conflict resolution plan is not our top priority, however if consistent problems arise in the future as a result of this, we would be willing to implement solutions. As it stands, the goals we outlined in the External Goals section act as a common ground for us all to stay on track, and through effective communication we hope to maintain this, and should any issues arise we would communicate them much the same way we arrived at our common goals initially, in a civil group discussion.

Stay on Track

As of now we do not have any reward system or penalty system in place for the greatest/lowest performers. We believe that this approach would ruin the mutual respect amongst the group and give an illusion of hierarchy which can only have negative implications for the team's productivity, especially given the small size. We are all in similar academic situations and expect it is in everyone's best interest to perform well on this project. Between deliverables there will be an assumption that group members are working as expected, however this will be checked on during meetings to ensure progress is maintained. With consistently timed meetings and encouraged open discussion, we hope to mitigate significant discrepancies in performance by making sure everyone is on the same page and knows they are supported by the rest of the group rather than competing with them. Given that this is the first time we are experiencing software development in this breadth and depth, we're approaching our targets with cautious optimism and carefully managing the precedents we set, as we are all fairly new to this.

Team Building

We have discussed team building activities such as sports watch parties, and celebratory pub nights for bigger milestones along the way. We have been attending more lectures together as a group, as well as shared all our social media contact information so we can communicate on platforms less formal than Discord or MS Teams. This will help make the group feel less foreign to each other by repeated exposure and occasional outings.

Decision Making

As of now, all decisions have been put to a vote. Consensus has not been necessary yet for any of our considerations, however if a situation arises that presents us with a particularly impactful team decision, we may choose to seek consensus. Along with the voting process we state why we vote how we do such that we can come to a compromise that best suits even those who voted for a less popular option, due to why they voted for it.