Development Plan
Software Engineering

Team 4, EventHub
Virochaan Ravichandran Gowri
Omar Al-Asfar
Rayyan Suhail
Ibrahim Quraishi
Mohammad Mahdi Mahboob

Table 1: Revision History

Date	**Developer(s)**	**Change**
September 22, 2025	Ibrahim, Mahboob, Omar	Creation of revision 0
September 22, 2025	Virochaan, Ibrahim	Reflection and Review
...	...	...

This document will outline the development plan for **EventHub**: handling of confidential information; IP protection and copyright licensing; team roles, collaboration, and organization guidelines; project development guidelines; project workflow and version control guidelines; and expected programming tools, technologies, and standards.

This project will be conducted in collaboration with two other Capstone teams to create one large unified product, and as such, the contents in these sections are subject to change based on updated requirements from the project supervisors or design decisions made in conjunction with the collaborating teams.

1 Confidential Information?

This project is not expected to deal with any information confidential to the McMaster Engineering Society.

2 IP to Protect

There is currently no IP which requires protection, and the project is planned to be open-source.

3 Copyright License

This project will use the GNU General Public License version 3 (GPLv3). The license can be found here.

4 Team Meeting Plan

The members of the team are expected to meet at least once weekly virtually through Microsoft Teams. The purpose of these meetings is to provide updates on current work, plan accordingly for submission of deliverables, discuss next steps, and distribute tasks among team members. The structure of this meeting will be as follows.

1. Each team member will provide a 5–6-minute update on the work completed within the last week. After each member has completed their update, the other team members are free to provide feedback.

2. The meeting leader will go over next steps on current deliverables, as well as remind team members of upcoming deliverables

3. If any new work is to be assigned, the meeting leader will split the work into a set of tasks. Team members will then select which tasks they are comfortable with taking

4. The remaining meeting time will be spent on preparation for the upcoming sync meeting with the supervisors

Additionally, a weekly sync meeting will be scheduled with the industry advisor and team supervisor to ensure that the direction of the project remains aligned with the needs of the supervisors. The meeting will ensure the supervisors are kept up to date with the state of the project and allow the supervisors to provide feedback or request changes. If there are no significant updates within the week before the meeting (i.e. during exam season), the meeting will be cancelled with at least 24 hours notice for the supervisors.

5 Team Communication Plan

- Issues: GitHub
- Meetings: MS Teams, Discord
- Meetings (with advisor): MS Teams
- Project Discussion: Discord

6 Team Member Roles

The following administrative and technical roles will be assigned to team members:

6.1 Rayyan Suhail

- Team Liason: Main contact point between the team members and the supervisors. Responsible for sending emails for communication between the team and supervisors as well as scheduling and leading the weekly sync meetings.
- Web UI/UX Developer (Admin Portal): Responsible for frontend design and development of the admin web portal.
- Software Tester (Web): Handles software testing of the web portal logic and UI.

6.2 Ibrahim Quraishi

- Inter-Team Liason: Responsible for communication between capstone teams. Since this project is a sub-component of a larger system for the MES with multiple teams working together, it is important to have a representative for the team who will be responsible for communication between teams to ensure consistency and compatibility between design decisions for the project.

- Full Stack Developer (Mobile): Responsible for development of frontend to backend connections for the end-user mobile application.
- Software Tester (Backend): Handles unit and integration testing of backend services.

6.3 Virochaan Ravichandran Gowri

- Meeting Leader: Responsible for facilitating the weekly meetings between team members. Ensures that team meetings happen consistently and that team members are up to date with deliverables.
- Full Stack Developer (Web): Responsible for development of frontend to backend connections for the admin web portal.
- DevOps/Project Manager: Updated and manages the GitHub Project and GitHub Actions workflows

6.4 Omar Al-Asfar

- Reviewer: Responsible for final review of any main deliverables before final submission to ensure they align with the course and supervisors' expectations.
- Mobile UI/UX Developer: Responsible for frontend design and development of the end-user mobile application.
- Software Tester (Mobile): Handles software testing of the mobile application logic and UI.

6.5 Mohammad Mahdi Mahboob

- Note Taker: Responsible for taking notes summarizing key points from every meeting, including team meetings and supervising meetings. This member is also responsible for creating GitHub Issues for every meeting and providing a summary of attendance and key takeaways.
- Backend Developer: Responsible for development of the backend server for the admin web portal and end-user mobile application.
- Database Designer: Responsible for design for structure of database tables and relations

7 Workflow Plan

The repository will have two primary branches, one for development and one for documentation, responsible for technical development and software documentation respectively.

The average workflow will be as follows:

- Pull changes from primary branch depending on the task at hand
- Create an issue for the assigned task if one does not already exist
- Create a new branch with the naming convention of: [assigned task - name of contributor]
  - If it is a new task, create a branch direcly under the primary
  - If it is a follow up/subtask, create a sub-working branch branch from previous branch
- Commit changes with detailed message
- Create unit tests for changes
- Open a pull request to the primary branch
- Link issue to pull request
- Wait for approval from other members and for testing to pass
- Merge pull request

Issues will be managed through GitHub Issues. Through GitHub Project boards, issues will be linked to notable meetings and milestones, and classified into categories: Meetings, To Do, In Progress, and Done. Each issue will be assigned to the relevant team members, who can provide feedback, report errors, and organize tasks. Issues will be tagged with distinct labels to indicate their priority, timeline, and type (e.g., bug, feature, documentation).

8 Project Decomposition and Scheduling

The project is hosted on GitHub at the following link: https://github.com/users/VirochaanRG/projects/4
A Kanban Board project setup is currently being used to track the progress of the project. The board is divided into four columns: Meeting, To Do, In Progress, and Done. Each task is represented as a card that can be moved across the columns as it progresses through different stages of completion. This visual representation allows team members to easily see the status of each task and identify any bottlenecks in the workflow.

The cards can pertain to:

- Lectures

- Group and Supervisor Meetings
- Project Milestones

The project is scheduled to be completed over the course of two semesters. However, we aim to have a working prototype by January to be tested across various events, including the Fireball Formal, and CALE Conference.

Deliverable	Due Date
Problem Statement and Goals, and Development Plan	September 24th, 2025
Requirements Documentation and Hazard Analysis (Revision 0)	October 6th, 2025
Verification and Validation Plan (Revision 0)	October 27th, 2025
Proof of Concept Report	November 3rd, 2025
Design Document (Revision 0)	November 10th, 2025
Proof of Concept demos	November 17, 2025
Functional Prototype	January

9 Proof of Concept Demonstration Plan

The main challenges we foresee for this project are creating the form builder and integrating user-side and admin-side interfaces for the application. The form-builder needs to be modular and track all data against a user to perform analytics in order to minimize redundancy and improve user experience. The interface integration is also challenging because they demand an entirely different set of features for the client, yet they must interact with the same underlying system.

The following are some significant risks we expect to encounter during development.

1. **Implementation of Branching Logic**
   We expect that the hardest part of the implementation for the custom form builder will be development of some sort of branching logic based on form answers. During development, we risk implementing this feature with too mcuh simplicity that does not meet the needs of the MES, or making the implementation too complicated and difficult to understand.

   To mitigate this risk, we can study existing form building software that implements this functionality to gather ideas of how it should be implemented, as well as study previous forms created by the MES to gauge the required complexity and depth of the feature.

2. **Frontend and Backend Integration**
   Our system is composed of many different subsystems, including the admin mobile and/or web application, end user web and/or mobile

application, the backend server, and the database. A large volume of data such as user information, custom forms, user submissions, and analytical data will need to passed between multple subsystems. Due to this, there is a significant risk of data loss and innacuracies between each system.

To mitigate the risk, it is imperative that features that require development on two or more subsystems are completed as soon as possible, and that work on each subsystem is done simultaneously. This ensures that integration between systems can be completed as quickly as possible and that developers are on the same page when developing features that require integration. This allows more time to be alloted towards testing said features and earlier detection of integration issues.

3. **Application Scalability**
   Our system will be required to deal with a large volume of data and simultaneous users. During development, we will likely only have the capacity to test the system with a few users and much smaller volume of data. Due to this, we risk our project not being scalable to the capacity required by the MES.

   To mitigate this risk, we should follow and conduct research on scalable practices including but not limited to breaking the system into smaller and more maintainable services and using libraries or services with built in scalability.

Some smaller risks associated with the project are as follows.

1. **Integration Between Capstone Teams** Since our project is part of a larger system being developed by multiple capstone teams, we have been asked to work with other teams to integrate our solutions into one application. There are a lot of risks involved with working with so many independent developers on a larger project such as incompatibilities between systems or miscommunication between teams.

   We can mitigate this risk by assigning an inter-team communications lead, who will stay in contact with other teams to ensure developers are on the same page.

10 Expected Technology

At the request of the supervisor, the following tools and technologies must be used for development of the project to ensure compatibility and seamless integration between capstone teams.

- JavaScript, and HTML/CSS shall be used in development of the frontend for both the web-based admin portal, and the user mobile application.

React and Next.js will be used as the primary framework for the admin portal and React Native will be used to develop the mobile application.

- PostgreSQL shall be used as the primary database for storage of any system and user data.

Additionally, the following programming languages, frameworks and libraries are expected to be used during development and may change at the discretion of the team.

- JavaScript along with Node.js may be used in development of the backend server for both the web portal and mobile application. Additionally, Node.js modules such as node-postgres may be used to communicate with the database.
- A styling library/framework such as Tailwind or Bootstrap may be used to simplify UI styling.
- Figma will be used for design and mockups of the mobile and web application UIs.
- A REST API will be used for communication between the frontend applications and the backend server. The default Fetch API provided within React and React Native may be used on the frontend, along with Node.js modules such as Express to handle requests on the backend server.
- Jest will be used as the primary unit testing framework for testing both the frontend and backend code with JavaScript, including unit testing and code coverage. Additionally, the React Testing Library may be used to test frontend components.
- A linter such as Prettier will be used to ensure the source code adheres to a specific standard.

GitHub will be used as the project repository for documentation, project planning and administration, and source code management. The repository can be found here:
https://github.com/VirochaanRG/MES-Event-Management-System.

- A GitHub Issue will be created for each attended lecture, tutorial, and team meeting outlining attendance, and key takeaways.
- A GitHub Issue will be created for each course deliverable, such as documentation, and code submissions.
- GitHub Projects will be used to host a Kanban Board of all GitHub Issues and their status.

- GitHub Actions will be used as the main CI platform. CI will be used to maintain code styling requirements and perform regression tests on pull requests before they can be merged. Additionally, GitHub Actions may be used build and deploy source code into a staging environment.
- Git will be used as the source code management platform for making code changes. The specific development environment in which these changes are made, such as the IDE, are subject to the preferences of the team members.

11 Coding Standard

The source code for the project will be subject to the following standards.

11.1 Language Style

The Airbnb JavaScript Style Guide and Airbnb React/JSX Style Guide will be used as the main styling convention for JavaScript code, as it is one of the most widely used styling conventions. This styling guide can be integrated into the CI/CD pipeline via ESLint and GitHub applications

11.2 Git Standards

The following branching model should be used when commiting to the repository.

- main branch: production ready code
- Feature/Deliverable branch: Code for each new feature/deliverable being completed

Any code changes should be made under a sub-branch under the appropriate feature/deliverable branch and should only be merged in via a pull request containing a description of what was changes and why. Once a feature is complete, a final pull request requiring all members of the team to review should be made before pulling into main.

A Git worktree should be used for the main branch and each feature/deliverable branch being worked on. This is to avoid constant switching of branches when pulling new changes and code stashes.

Appendix — Reflection

Virochaan Ravichandran Gowri's Reflection:

1. **Why is it important to create a development plan prior to starting the project?**
   Creating a development plan prior to starting the project is important because it provides us with a structured roadmap of how we can deliver this project. By including deadlines as well as roles and responsibilities we are ensuring we have certain guidelines on how best to work as a group together and when be begin the technical parts of our project we have a general understanding of the components we are going to work on. Also by defining meeting communication structures we can ensure that our group understands their responsibilities to each other and if/when issues crop up, we are well equipped to deal with them. Finally we also including some Project Management and workflow considerations in this plan. This will be crucial when we start developing the project to ensure we don't end up conflicting with one another and actually keep a good track of the work being done. By having all of this defined at the start of the project we are able to work efficiently and have documentation to fall upon if we get stuck during the development of the project.

2. **In your opinion, what are the advantages and disadvantages of using CI/CD?**
   By using CI/CD primarily through Gitub Actions we gain many benefits throguh automatic QA and easier integration. We can easily compile files and create workflows for testing different components and including linting that ensures that we maintain a high code quality. Through this system we can ensure that even if multiple developers are working on the code the system will remain stable as changes will only be implemented if they pass through the workflows without errors. The main disadvantages are that there could be a learning curve to understanding this new technology as sometimes it can not be the most intuitive. There is also extra effort required in creating and maintaining the workflows as well as debugging the failed builds to understand what went wrong. Overall if used effectively CI/CD can be a powerful tool that can help us work better as a group and ensure we have a stable development platform.

Omar Al-Asfar's Reflection:

1. **Why is it important to create a development plan prior to starting the project?**
   Creating a development plan prior to starting a project is crucial in defining structure and direction. This allows us to establish clear goals, roles, timelines, and milestones prior to working on the technical aspects of the project. It helps in aligning expectations and ensures that everyone is on the same page. This allows the team to stay organized, and divide respon-

sibilities more effectively based on respective skill sets. Moreover, it helps in highlighting potential areas of concern which may not be as clear during brainstorming. Overall, a development plan acts as a guideline, enabling clear direction from the early stages of the project, while also serving as a reference point to revisit during confusion to remain on track.

2. **In your opinion, what are the advantages and disadvantages of using CI/CD?**
   The primary advantage of CI/CD is that it streamlines the integration and deployment process. By automating testing, building, and deployment, it ensures that code changes are consistently validated, making it possible to discover and address issues quickly. In similarly collaborative environemnts, it allows for mulitple people to work on the same code with seamless integration, resulting in a more efficient workflow. At the same time, there are also some disadvantages to consider, mainly in the complexity involved. Setting it up can be very time-consuming and confusing for new users, or with complex projects. The time spent on maintaing the pipeline may be better spent on other taks for smaller scale projects.

Ibrahim Quraishi

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

   It is important to create a development plan prior to starting the project so that the developers can set an agreed upon starting point for the project. If there are no rules or expectations set in place before beginning development, it will be extremely difficult to build up the momentum to get the project into a stable workflow due to disalignment between developer preferences stemming from disagreements or miscommunication. By having a solid development plan prior to starting the project, everyone will have an idea of where to start, what skills they are expected to have, and how to build onto that starting point.

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

   Using CI/CD in a project provides several advantages, as it automates alot of monotonous work that would typically require a dedicated person to take on that role. For example, CI/CD can automate integration and regression testing, automate compilation and deployment of the software, and provide code analysis and style checking. Some disadvantage of CI/CD is that it may take alot of time and resources to setup initially, and also slows down the development process by requiring developers to wait for the CI/CD pipeline to complete for every small change they make.

Rayyan Suhail

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

   Before starting a project, a development plan gives direction and structure. Without it, it's common for team members to misunderstand priorities. A plan makes it clear who is in charge of what and helps divide the work into manageable steps with reasonable deadlines. It also makes it simpler to make adjustments if circumstances change. In my experience, when a project begins without a plan, we typically spend more time figuring out what to do next rather than moving forward.

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

   One of the main advantages of CI/CD, in my opinion, is that it significantly reduces the stress caused by integration. We can constantly test and deploy changes in smaller chunks rather than having to deal with a massive merge at the end, which keeps the project stable and reduces bugs. The disadvantage is that setting up CI/CD requires time and work, particularly in the beginning when you want to test something quickly. However, because it allows for collaboration and minimizes last-minute issues, I believe the advantages outweigh the disadvantages overall.

Group Reflection:

1. **What disagreements did your group have in this deliverable, if any, and how did you resolve them?**
   Our group experienced some minor disagreements during the development of this project primarily on the division and size of the roles and responsibilities and how best to structure the workflows. When discussing the technical roles we were unsure on how granular we wanted to be and how much responsibility we wanted to concentrate in each roles. We ended up splitting up the roles based on the interest of each group member and if they had prior experience. We also made sure that we didn't end up having anyone with too much responsibility as it would be unfair and could prove challenging for that person. We also had a discussion on the different technologies that we could use. Everyone had their own preferences and experiences using different technologies for full-stack development. To resolve this we researched what technology stacks worked best together and also had discussions with the supervisor to get his opinions as he also had a technical background. Then we got together and decided on the best stack for our use case. By having open and constant communication we managed to easily overcome these minor disagreements ensuring that we

could have a succesful deliverable and set the standard for the rest of the project.

Appendix — Team Charter

External Goals

The primary goal of our team is to create a convenient and reliable solution that we can apply to adress a real-world problem. The team aims to have the project used in real MSE events, and potentially even by McMaster as a whole in the future. At the same time, the team aims to achieve an A+ in the course, and have it serve as a valuable talking point in future interviews. Finally, we intend to have fun while developing new skills that will be useful in our future careers.

Attendance

Expectations

Our team expects all members to attend all scheduled meetings on time, and remain for the entire duration. In cases where a member is unable to attend, they are expected to inform the team well in advance with justification, so we can reschedule if possible or proceed accordingly. If a member plans to leave a meeting early, they should notify the team by the beginning of the meeting at the latest, so we can adjust the agenda if necessary. Continous unexcused absences will not be tolerated, and will be addressed by the team.

Acceptable Excuse

Acceptable excuses for missing a meeting or deadline includes severe illness, family emergencies, or significant accidents. These constitute cases where missing a meeting unannounced is understandable. Other excuses, such as unavoidable commitments are only acceptable given that the member gave prompt notice.

Unacceptable excuses generally includes any avoidable or last-minute situations. For example, forgetting about the meeting or deadline, oversleeping, or prioritizing other work over this project.

In Case of Emergency

In case of an emergency, the team member should inform the team as soon as possible, ideally before the meeting or deadline. After discussing the situation, the team will determine how to act:

- In the event of a missed meeting, the team will reschedule or proceed without them.
- In the event of being unable to complete their part of a deliverable, the team will then discuss how to redistribute tasks to accommodate the absence. If the work is unable to be completed by the others on time, the member should inform the course instructor and request relief.

Accountability and Teamwork

Quality

Our team's quality expectations are as follows:

- Meetings:
  All members are expected to do their due diligence and prepare for future meetings as discussed in the piror ones. This may include reviewing/studying material, completing milestones, and preparing resources necessary for discussion. Everyone is expected to have an understanding of the current state of the project in order to contribute meaningfully to discussions.

- Deliverables:
  All members are expected to complete their part of deliverables to the best of their ability. Members are expected to adhere to predetermined guidelines and standards. If a member is struggling or unconfident in their part, they are expected to communicate this to the team as soon as possible. Upon completion, all members are expected to thoroughly review the deliverable for quality before submission, and especially focus on areas that were highlighted for improvement. All feedback will be respectful but honest, and all members are expected to take constructive criticism professionaly.

Attitude

Our team expects all members to maintain a positive and respectful attitude. We value open communication, active listening, honesty, and constructive feedback. At our first meeting, we discussed our goals and expectations. We agreed that while we aim for a complex project and a high grade, we also want this experience to be enjoyable.

We recognize that not all members will feel as motivated or free to work as the rest, and we agreed to be flexible and understanding as long as everyone demonstrates a positive attitude and an honest effort.

All members are expected to:

- Contribute ideas in a respectful manner.
- Maintain an open attitude to other opinions.
- Complete work with a level of quality and effort up to the agreed upon standards.
- Exhibit honesty about challenges so the team can accomodate.

In the event of conflicts or disagreements, we will address them through a conflict resolution plan:

- Initiate an open and respectful discussion.
- Allow all members to voice their opinions and concerns.
- Remain objective. The focus is on the issue, not the person.
- If consensus can't be reached, a majority vote will be held.
- Everyone will accept the results of the vote.
- If necessary, we will seek the assistance of a neutral third party to help mediate.

Stay on Track

To keep the team on track, we will employ the following methods:

- Regular Meetings: We will hold weekly meetings to discuss progress, address challenges, and plan next steps. With this, we can ensure everyone is aware of their responsibilities and track progress.
- Clear Roles and Responsibilities: Each member has defined roles and responsibilities, ensuring accountability for specific tasks. The team will aim to divide responsibilties as fairly as possible.
- Performance Metrics: We will set target metrics for attendance, commits, and task completion. We will use tools like GitHub Issues and Projects to track tasks, deadlines, and progress, making contributions transparent.
- Feedback and Rewards:
  - Above Expectations: Members who consistently meet or exceed expectations will be recognized by the team and to supervisors. We may also consider small rewards, such as choosing a team activity.
  - Below Expectations: Members who consistently fall short of expectations will be addressed privately. If this is a regular occurence, concequences may include being assigned more work, or approaching a TA or instructor.

Team Building

To ensure team cohesion, we have decided to set up a Discord server for casual communication outside of meetings. We will also meet in person at least once a week in some manner, from attending lectures and events to simply meeting up for food. Finally, we will celebrate all milestones and achievements, no matter how small to boost team morale and motivation.

Decision Making

Decisions will be made in our group by consensus whenever possible. A conclusion everyone agrees on is important to ensure all members are motivated to invest time and effort into the project. If a consensus cannot be reached, the differing opinons will initially justify their decision and attempt to convince the others. If that does not work, a majority vote will be held.

Disagreements will be handled through open discussion. All members will have equal opportunity to voice their opinoins, and will remain calm and respectful. The priority will be given to the project goals outlined at the beginning of the course as opposed to personal feelings. If necessary, a neutral third party may be consulted to help mediate and resolve conflicts.