AppCraft Project Plan

## Table of Contents

[**Table of Contents 1**](#_pkqjxg5vfn51)

[**Introduction 1**](#_xqf880qvv5kr)

[**1.0 Project Vision 2**](#_qkoec1usrzbj)

[**2.0 Team 2**](#_7pnu4ltmo6qc)

[2.1 Client Information 2](#_r6kdo61td5vy)

[2.2 Stakeholder Interest 3](#_9wld9r8pmlh5)

[2.3 Team Members 3](#_r9oz8nlz97rk)

[2.2.1 Contact Information 3](#_rkostze3duf6)

[2.2.2 Role and Responsibilities 4](#_mfcn6yyces5o)

[**3.0 Team Process Model 4**](#_k5nymlgn5o81)

[**4.0 Definition of Done 4**](#_8b9ljsey7noi)

[**5.0 Project schedule 5**](#_z5528fnfl04l)

[5.1 Weekly Schedule 5](#_jfbtm9q3a1qp)

[5.2 Time Tracking 5](#_nkpvf3d1ei3o)

[**6.0 Project management 6**](#_3277g1fwbe0u)

[6.1 Task Allocation 6](#_wecuv9mm2tyq)

[6.2 Progress Tracking 6](#_j1p3pb5lvcw2)

[6.3 Task Reporting 7](#_183emltdywk9)

[6.4 How do we review the project 7](#_eadkez2deaat)

[6.5 Backlog Management 8](#_nmqta2mhg48v)

[6.6 Git Usage Policy 8](#_db9aqvrjo6nv)

[6.6.1 Version Control and Repository Management 8](#_8c5lad5snk9w)

[6.6.2 Branching 8](#_kqig386cz9iv)

[6.6.3 Commit Frequency and Standards 8](#_nzjshds6nurb)

[6.6.4 Conflict Resolution 8](#_qc6zwou6dj8w)

[**7.0 Selection criteria 10**](#_7bnrj8anr9kz)

[**7.0 Alternative Analysis 10**](#_ndhsfwonm0z3)

[7.1 Platform selection analysis 10](#_vubuj3j3lhvd)

[**8.0 Conclusion 11**](#_ofdsxqalan8l)

[**9.0 Appendix 12**](#_67lywuv51als)

# Introduction

The project involves developing a project management software tool tailored for agile projects, specifically for managing a single project. The tool will include features such as task management, product backlog management, sprint board management, and team member management, accessible via the internet.

The purpose of this project is to create a cost-effective, in-house solution for project management that will cater to the current and future needs of our software start-up. The tool will help streamline project management processes and enhance team productivity.

The primary users will be the internal team members of the start-up, including project managers, developers, and other stakeholders who need to track project progress and manage tasks efficiently.

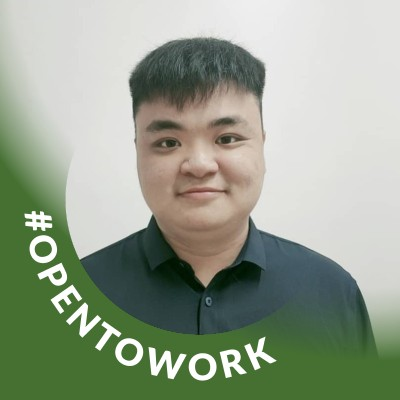
# 1.0 Project Vision

Our vision is to create a universally accessible educational platform that empowers students from all corners of the globe with free, high-quality educational resources.

# 2.0 Team

This section provides an overview of the key stakeholders involved in the project, including client details and comprehensive information about the development team. It introduces the client and outlines the roles, responsibilities, and contact information of each team member, highlighting their contributions to the project’s success.

## 2.1 Client Information



**Client Name and Affiliation:**Mr Chong Chun Yuan, Project Manager at Monash.

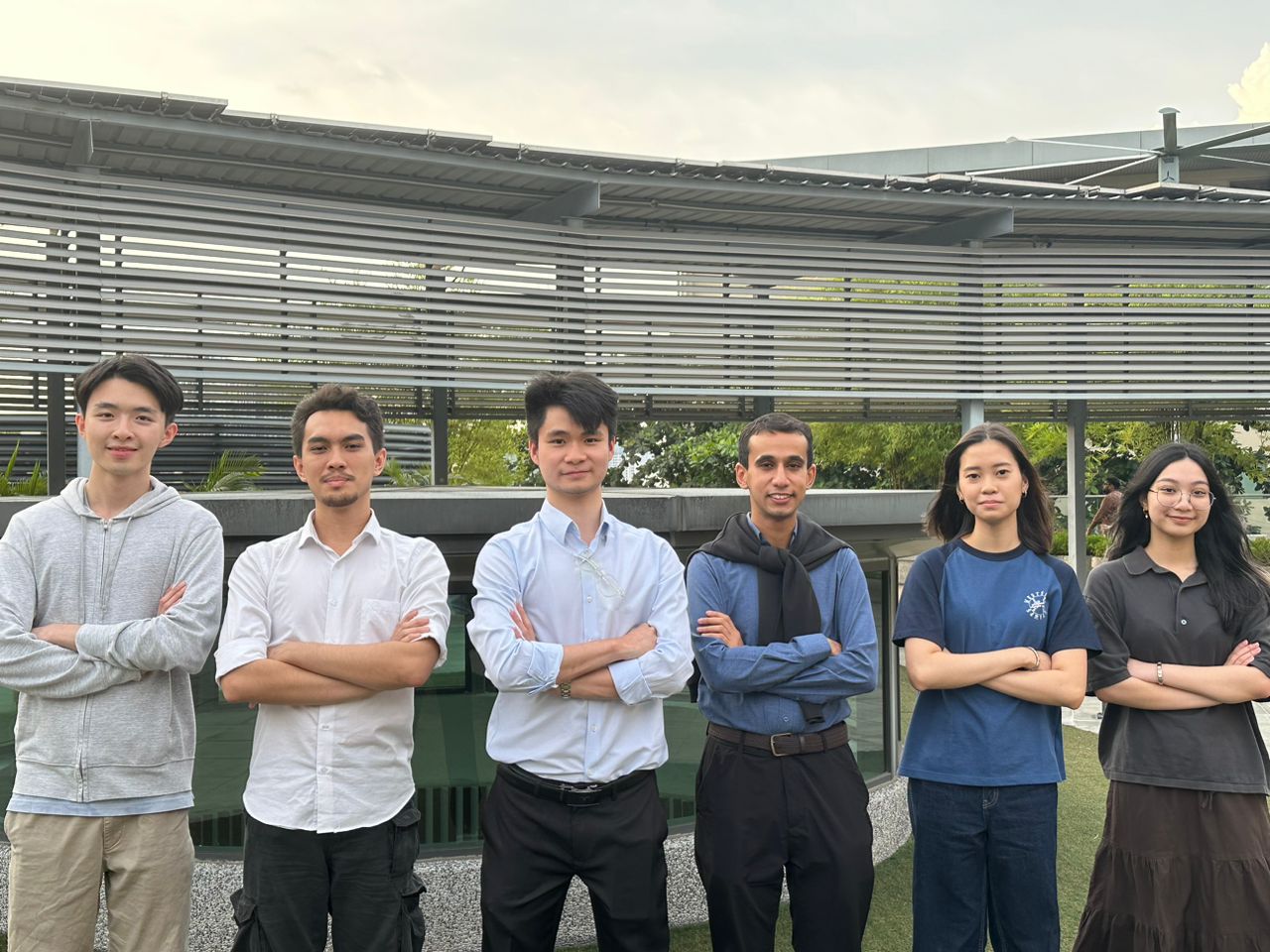
**Contact Details:**

[chong.chunyuan@monash.edu](mailto:chong.chunyong@monash.edu)

## 2.2 Stakeholder Interest

| Aspects | Interest |
| --- | --- |
| Accessibility | Able to access through internet |
| Budget | Low budget |
| Integration | Seamless integration with existing tools |
| User Experience | Easy to use |

## 2.3 Team Members



We are **AppCraft**, a team of dedicated professionals working together to deliver this project. Our team members include Yuen Kei Foong, Adji Ilhamhafiz Sarie Hakim, Teh Ming Dong, Mohanad Al-Mansoob, Charmaine Chee Hing Yi, and Arielle Ocampo Dela Cruz, each contributing unique skills and expertise to achieve our project goals.

### 2.2.1 Contact Information

1. Teh Ming Dong - Lead Developer, Backend:   
   [mteh0004@student.monash.edu](mailto:mteh0004@student.monash.edu)
2. Adji Ilhamhafiz Sarie Hakim - Quality Assurance (QA):   
   [ahak0006@student.monash.edu](mailto:ahak0006@student.monash.edu)
3. Charmaine Chee Hing Yi - Documentation, Backend:  
   [cche0258@student.monash.edu](mailto:cche0258@student.monash.edu)
4. Yuen Kei Foong - UI/UX:  
   [kyue0009@student.monash.edu](mailto:kyue0009@student.monash.edu)
5. Arielle Ocampo Dela Cruz - Product Owner:  
   [adel0037@student.monash.edu](mailto:adel0037@student.monash.edu)
6. Mohanad Al-Mansoob - Scrum Master:  
   [malm0022@student.monash.edu](mailto:malm0022@student.monash.edu)

### 2.2.2 Role and Responsibilities

| **Roles** | **Responsibility** |
| --- | --- |
| Project Manager | Oversees the project timeline, manages client communication and ensures the smooth team collaboration. |
| Lead Developer | Oversees the technical direction, mentoring other developers and ensures the code quality. |
| Quality Assurance (QA) Tester | Conducts thorough testing of the software to identify and report bugs to ensure the software meets the required quality standards. |
| UI/UX Designer | Designs the user interface and user experience, ensuring the software is user-friendly and visually appealing. |
| Backend Developer | Develops and maintains the server-side logic, database management and API integration. |
| Frontend Developer | Implements the user interface and ensures seamless integration with backend services. |

# 3.0 Team Process Model

The team will adopt a [**Scrum** process model](#esjuiqj9i5jh) with three sprints, each lasting two weeks. The project will be developed iteratively, with regular reviews and adjustments based on feedback.

Each sprint will begin with a planning session, followed by daily stand-ups, and end with a sprint review and retrospective.  
  
The team will use Jira for task management, GitLab for version control, and Google Drive for documentation. Whatsapp will be used for communication, and Figma for design collaboration.

# 4.0 Definition of Done

Checklist:

The code is complete and adheres to coding standards.

All associated test cases are passing successfully. (if we have test cases?)

The code has been reviewed and approved by at least one other developer. (or qa)

Documentation is updated to reflect the new feature or changes.

The feature is deployed to the staging environment and validated by the QA team.

Review and Testing:   
 Every feature must undergo a review and thorough testing before it can be marked as   
 "done."

Approval Process:

The Project Manager and the Lead Developer must approve the completion of tasks,   
 ensuring all criteria in the Definition of Done are met.

# 5.0 Project schedule

This section outlines the overall timeline and key milestones for the project, detailing important deadlines and deliverables. It includes a breakdown of weekly meetings, sprint cycles, and key activities, ensuring that the team remains aligned and on track. The schedule provides a clear structure for task allocation, progress tracking, and regular check-ins, allowing for iterative development while ensuring timely completion of project goals.

## 5.1 Weekly Schedule

A weekly stand-up meeting will be held every Tuesday from 10:00 AM to 12:00 PM. This meeting will allow the team to review the progress of the week, discuss any challenges in greater detail, and ensure that all tasks are on track. It will also provide an opportunity to align on priorities for the upcoming week, ensuring that everyone is clear on their objectives and responsibilities.

## 5.2 Time Tracking

Time Tracking Approach:

[The team will use an Excel spreadsheet named “Contribution logs”](#piah9a4wcjs5) for tracking time spent on each task. This spreadsheet will serve as a contribution log where each team member records the time they start and complete tasks.

The Excel file will also track planned start and completion dates for tasks against actual dates. This will help monitor progress and identify any discrepancies early, allowing for timely adjustments.

# 

# 6.0 Project management

This section details the strategies and tools used to effectively manage the project, including methods for tracking progress, managing the backlog, monitoring time, and implementing version control. The project management approach ensures that tasks are prioritised, deadlines are met, and the codebase remains organised and high-quality.

## 6.1 Task Allocation

User Story Completion

Our approach to task allocation is designed to ensure that user stories are completed sequentially, prioritising efficiency and value. We will work through user stories one at a time, focusing on finishing each before moving on to the next. This approach helps maintain clarity and ensures that each task is given the attention it requires.

**Self-Allocation Based on Abilities:**

Team members will allocate tasks to themselves according to their individual strengths and expertise. This self-allocation strategy leverages each member’s unique abilities, ensuring that tasks are handled by the most qualified person. By aligning tasks with personal skills, we aim to maximise productivity and the quality of our deliverables.

**Handling Difficulties:**

If a difficulty or blocker arises that hinders the team’s progress, the Scrum Master will step in to assist. The Scrum Master’s role is to help remove obstacles, provide guidance, and ensure that the team can continue working smoothly. This proactive approach to problem-solving minimises disruptions and keeps the project on track.

**Prioritisation of User Stories:**

We will prioritise user stories based on their value to the client. The most valuable and impactful user stories, as identified by the client, will be tackled first. This ensures that the most critical features and functionalities are delivered early, aligning the project’s progress with the client’s needs and expectations.

## 6.2 Progress Tracking

**Creation and Assignment:**

Each user story and associated task is created as an issue in Jira. These issues are categorised by type (e.g., story, bug, task) and are linked to specific sprints or epics. Team members can self-assign tasks based on their expertise, or tasks can be assigned by the Scrum Master or Product Owner as needed.

**Progress Monitoring:**

Jira’s Kanban or Scrum boards provide a visual representation of task progress. Tasks move through columns such as "To Do," "In Progress," "In Review," and "Done," making it easy to see the current status at a glance. Each task’s details, including descriptions, comments, and attachments, are updated in real-time to reflect the latest progress.

Milestones: Key milestones in this project include the completion of each sprint, the delivery of major features, and the final project handover.

## 6.3 Task Reporting

**Status Updates**

Team members are expected to update the status of their tasks regularly. Jira’s dashboard features and custom reports allow us to generate real-time insights into sprint progress, task completion rates, and overall project efficiency and balance. Burn-down charts help in visualising the rate of task completion against the sprint timeline.

**Weekly Stand-ups**

During weekly [stand-up meetings](#qnqbu9wt26l2), team members provide verbal updates on their Jira tasks, highlighting what was completed, what is in progress, and any blockers encountered. These updates are reflected in Jira to keep the entire team informed.

## 6.4 How do we review the project

**Peer Reviews**

Once a task is marked as "In Review," it triggers a peer review process. Other team members can review the work, provide feedback, and request changes if necessary. This collaborative review process is facilitated directly within Jira through comments and code review tools (e.g., github Co-pilot).

**Sprint Reviews:**

At the end of each sprint, completed tasks are reviewed during the sprint review meeting. The team demonstrates completed user stories to stakeholders, using Jira to track which stories meet the Acceptance Criteria and which may need further refinement.

**Retrospectives:**

Post-sprint, the team conducts a retrospective to discuss what went well and what could be improved. Jira’s reporting tools help identify patterns in task completion, such as recurring blockers or tasks that took longer than expected, allowing the team to continuously improve their processes.

## 

## 6.5 Backlog Management

Our team [uses Jira to manage our project backlog](#6mj6cbrt7886), where the task will be stored, prioritised and tracked throughout the project lifecycle in the backlog section as shown in the appendix. The product owner will manage the project backlog manually based on the estimated story point, task priority from client side and feasibility in a time constraint environment.

Prioritisation: Backlog items will be prioritised based on their importance and urgency, as determined during sprint planning sessions. (by the scrum master??)

Review Frequency: The backlog will be reviewed and updated at the start of each sprint and as needed during sprint planning or daily stand-up meetings.

## 6.6 Git Usage Policy

### 6.6.1 Version Control and Repository Management

[GitLab will be used for version control](#qjsm7tb5rxxh), with all code stored in a central repository accessible to all team members. The repository will be organised and structured to support seamless collaboration and easy navigation.

### 6.6.2 Branching

The team will use a Git branching strategy, with separate branches for (features, development, and production to maintain code integrity?)

### 6.6.3 Commit Frequency and Standards

Developers are expected to commit their code frequently to ensure continuous integration. Commit messages should be clear, concise, and follow a standardised format, including the task or issue ID, a brief description, and any relevant details (e.g., "TASK-123 —- Implemented user authentication"). This helps maintain project organisation and traceability.

All code changes must undergo a peer review process before being merged into the main branch. This ensures that the codebase remains clean, efficient, and free from errors.

### 6.6.4 Conflict Resolution

What is a Git Conflict? When you're working on a project with others, everyone might be making changes to the same files. Git is a tool that helps track those changes and combine everyone's work into one final version. However, sometimes two people might make different changes to the same part of a file. This is called a Git conflict.

Merge conflicts will be addressed promptly by the developers involved, with support from the Lead Developer as needed to ensure continuity and maintain code quality.

Here are some tips to avoid Git conflicts:

* Talk to your team about who is working on which parts of the project. This helps avoid multiple people making changes to the same files.

Pull Changes Regularly:

* Frequently pull the latest changes from the main branch to stay updated with what others are doing. This makes conflicts less likely.

Use Feature Branches:

* Work on your own branch for new features or changes. When your work is done, merge it back into the main branch. This keeps things organised and helps isolate changes.

Commit Small and Often:

* Make small, frequent commits instead of big, infrequent ones. This makes it easier to resolve conflicts and track changes.

# 

# 7.0 Selection criteria

# 7.0 Alternative Analysis

## 7.1 Platform selection analysis

| Aspects | Desktop Application | Mobile Application | Web Application |
| --- | --- | --- | --- |
| Team experience | Low | High | Medium |
| Accessibility | Offline and online as long as it is installed in the device | Offline and online as long as it is installed in the mobile device | Anywhere with internet connection |
| Offline access | Limited functionality, because to prevent crashes with other people’s update | Limited functionality, because to prevent crashes with other people’s update | Limited functionality, only able to access the loaded feature from the browser buffer. |
| Platform | Linux and windows | IOS and Android | Web browser |
| Performance | Depending on internet speed, server and PC performance due to the OS clock speed and available ram usage. | Depending on internet speed, server and mobile performance | Depending on internet speed and server. |
| Maintenance | Updates need to be pushed to each installed instance | Updates need to be pushed to each installed instance | Centralised updates on the server, instantly available to users |
| Integration Complexity | High due to the need for platform-specific APIs and system-level integration | Medium, requires integration with mobile OS features and APIs | Low to medium, primarily involves integration with web services and databases |
| API Utilisation | Extensive use of system-level APIs, may require different APIs for different OS | Utilises mobile-specific APIs (e.g., GPS, camera) | Mostly involves RESTful APIs, web services, and third-party libraries |
| Testing and Debugging | Complex due to the need for testing on multiple OS versions and configurations | Requires testing across multiple devices and OS versions | Easier with browser-based debugging tools, but needs cross-browser testing |
| Continuous Integration (CI) | More complex CI setup needed for different platforms and OS versions | Medium complexity, requires different pipelines for Android and iOS | Simplified CI/CD pipeline with web hosting services and cloud platforms |
| Third-Party Libraries | Requires careful selection of libraries compatible with specific OS | Mobile-specific libraries and SDKs needed, depending on the platform | Wide range of web libraries and frameworks, generally easier integration |

Based on the comparative analysis, the web application stands out as the most suitable choice for development due to its lower integration complexity and broader accessibility. Unlike desktop and mobile applications, which require platform-specific APIs, tools, and extensive testing across multiple devices and operating systems, a web application allows for a more streamlined development process. The use of web services and RESTful APIs simplifies the integration of third-party libraries and services, reducing the overall development time and effort. Moreover, the deployment process for a web application is significantly easier, as updates can be instantly pushed to the server and accessed by all users without the need for individual installations or app store submissions. This continuous integration and deployment process is not only more efficient but also more cost-effective, making the web application an ideal choice for delivering a robust and accessible project management tool that meets the diverse needs of its users.

## 7.2 Programming analysis

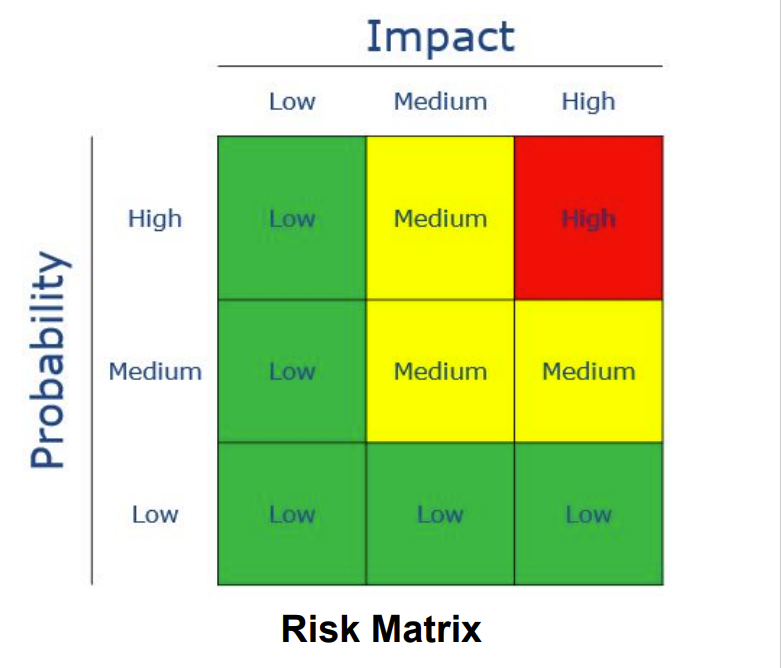
| Aspects | Java | JavaScript | Python |
| --- | --- | --- | --- |
| Team experience | Medium | Low | High |
| Usage and application | Best for large-scale enterprise applications, Android apps, and backend systems. | Dominant in web development (frontend and backend), especially for interactive and dynamic websites. | Ideal for data analysis, machine learning, scripting, and web development (with frameworks like Django, Flask). |
| Performance | Highly performant and suitable for applications requiring high concurrency and stability but known as RAM hungry due to no effective garbage collection | Fast execution in browsers; Node.js extends its use to server-side for moderate performance needs. | Slower runtime compared to Java; best for non-performance-critical applications.  Does not supports multithreading, and building native mobile applications. |
| Integration | All can be integrated effectively using frameworks and microservices. | | |
| Community and supports | All has rich ecosystem with extensive libraries and strong supports. | | |
| Web Frameworks | Spring | React, Angular | Django, Flask |
| Database | MongoDB, firebase | MongoDB, firebase | Firebase |

xRecommendation

# 

# Risk Management

We will be register our risk management based on the table shown below



| ID | Date Raised | Risk Description | Likelihood | Impact to the project | Severity | Mitigating Action | Contingency Plan |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 13/08/2024 | Mental Health Crisis  When a team member experiences a severe mental health crisis, such as a mental breakdown or intense pressure | LOW | This could affect team productivity, lead to delays in task completion, and potentially cause a decrease in overall project quality | Low | * Encourage consultation sessions with Teaching Assistants * completing work on time to avoid built up stress towards the deadline. * Have a supportive environment that allows team members to have open discussion about their workload and stress level. | * Provide support by offloading some of their tasks and responsibilities * Redistribute the affected member’s task to other group mates equally. |
| 2 | 13/08/2024 | Personal Issues  When a team member receives distressing news or faces personal challenges that significantly affect their ability to focus and work efficiently. | LOW | Team member's efficiency might decrease significantly, leading to delays in their assigned tasks, which could impact the project timeline. | Low | * Allow flexibility in work hours to accommodate personal circumstances. * Regularly check in on team members to offer support and understand if any personal issues are affecting their work. | * Redistribute the affected member’s task to other group mates equally. * Offer support by having bonding sessions as a team to ensure team members feel valued and supported. |
| 3 | 13/08/2024 | Technical debt  Accumulation of shortcuts, incomplete implementation and substandard code that will affect the quality of the product in the long run. | MEDIUM | Accumulating technical debt can lead to future maintenance challenges, increased debugging time, and potential delays in project delivery due to unforeseen technical issues. | MEDIUM | * Follow good programming principles and practices, create branches in GIT for backup and avoid hardcoding. * Have code review sessions as a team, so feedback can be provided and ensure high code quality | * In case of critical technical debt issues, always prioritise refactoring over developing new features. * Reassign task among members to focus on minimising technical debt |
| 4 | 13/08/2024 | Technical issue  Unexpected problems with software, hardware, or network systems. (e.g system crashes, software bugs, hardware failures, or connectivity problems…) | MEDIUM | Critical errors in team members' devices could result in lost work, decreased productivity, and potential delays in project timelines. | MEDIUM | * Run diagnosis on the computer to identify any potential problem in your device. * Ensure team members regularly backup files and data * Update and ensure software and systems are up to date. * Have an alternative hardware or backup systems | * Switch to an alternative hardware or backup systems to continue working * Seek for IT support immediately * If a software is down, identify the next best alternative to continue working temporary |
| 5 | 20/08/2024 | Miscommunications between members  Misunderstandings or lack of clear communication between team members that can occur due to unclear instructions, assumptions, or failure to share important information. | MEDIUM | Lead to misunderstandings regarding tasks and requirements, resulting in errors, rework, and inefficiencies that can delay the project. | MEDIUM | * Ask for clarification between members. Write everything down in black and white. * Schedule daily stand ups to keep everyone informed about project progress, tasks, and any changes. | * address it immediately by clarifying the issue with all involved members. * facilitate a resolution through open discussion or involve a mediator, such as the project manager, to help resolve the issue. |
| 7 | 20/08/2024 | Project purpose is undefined  Team member are unable to make a clear purpose when they are making the project | LOW | An unclear project purpose can lead to confusion among team members, misaligned efforts, and ultimately a product that does not meet client expectations, risking project failure. | LOW | Talk with the client and seek more information. |  |
| 8 | 20/08/2024 | Accident  Team member gets into a non-fatal car accident. | LOW | A team member’s involvement in a non-fatal accident could result in their temporary absence, affecting the team's capacity and potentially causing delays. | LOW | Encourage safe practices and avoid unsafe/risky circumstances. | * Redistribute the affected member’s task to other group mates equally. |
| 9 | 20/08/2024 | Mild disease  Team member contracts a mild disease (e.g common cold) | LOW | could lead to temporary reduced productivity or absence, impacting the project timeline. | LOW | Maintain good personal hygiene, encourage a good work-life balance | * Redistribute the affected member’s task to other group mates equally. |
| 10 | 20/08/2024 | Conflicting client’s requirements. (what we received and client’s imagination is different)  A situation where the requirements provided by the client differ from what the client actually envisions or expects. | MEDIUM | Differences between client expectations and delivered work can lead to significant rework, increased costs, and potential delays in project delivery. | MEDIUM | * Consult with the client and show them the prototype.Obtain formal approval from the client to ensure alignment. * Implement regular feedback loops with the client throughout the project to ensure that their expectations are being met * Maintain clear and detailed documentation of all requirements and client communications. | * schedule an immediate meeting with the client to discuss and reconcile the differences. |
| 11 | 21/08/2024 | The team might accidentally infringe on copyrights or patents, leading to legal disputes. | MEDIUM | Legal disputes from accidental infringement could result in project delays, increased legal costs, and damage to the project's reputation. | MEDIUM | Educate team members on intellectual property laws and the importance of originality in work. |  |

# 8.0 Conclusion

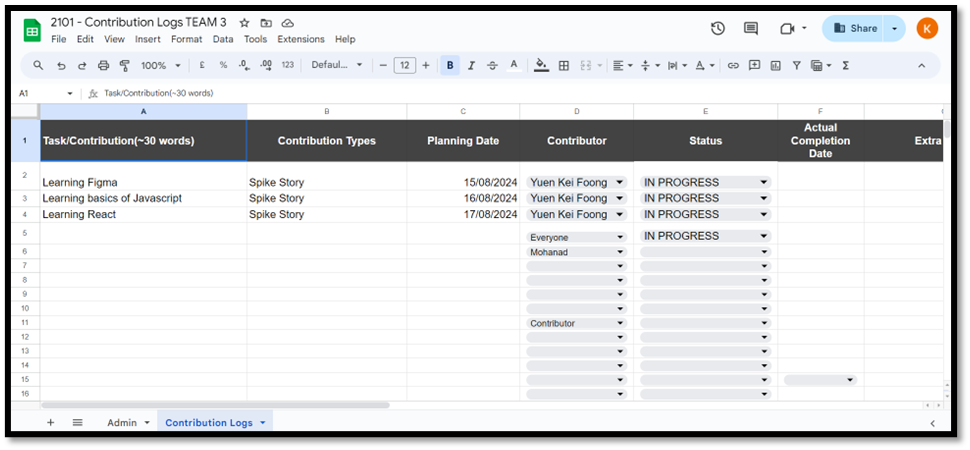
In conclusion, the AppCraft project plan outlines a structured approach to developing a robust project management tool tailored for agile workflows. Through clear definitions of roles, responsibilities, and processes, our team is well-prepared to deliver a high-quality product that meets the needs of our client.

We have established a strong foundation with a well-defined process model, effective time and task management strategies, and a focus on continuous improvement through iterative development. The inclusion of tools like Jira for task management, GitLab for version control, and Figma for design collaboration will ensure that our workflow remains efficient and our codebase remains organised and maintainable.

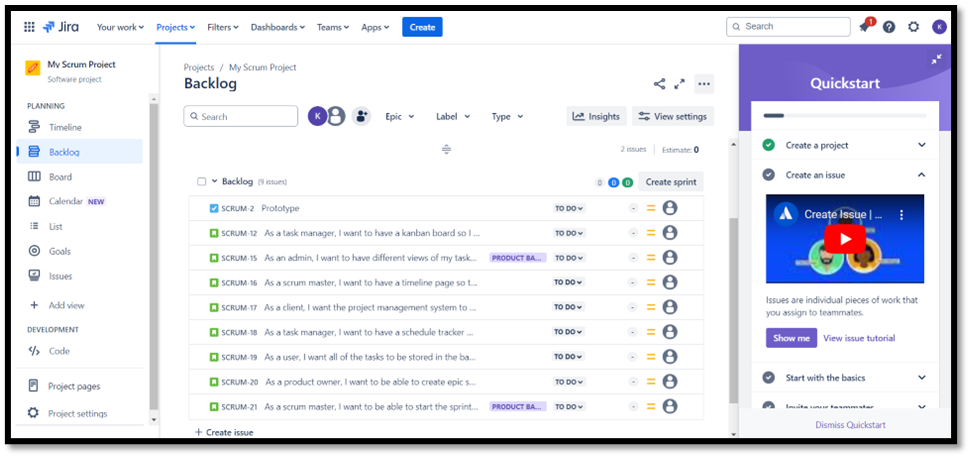
As we move forward, we will continue to monitor our progress closely, adapt to any challenges that arise, and maintain our focus on delivering a product that not only meets but exceeds expectations. This project plan can serve as our guide, ensuring that all team members are aligned and working towards a common goal.

# 9.0 Appendix

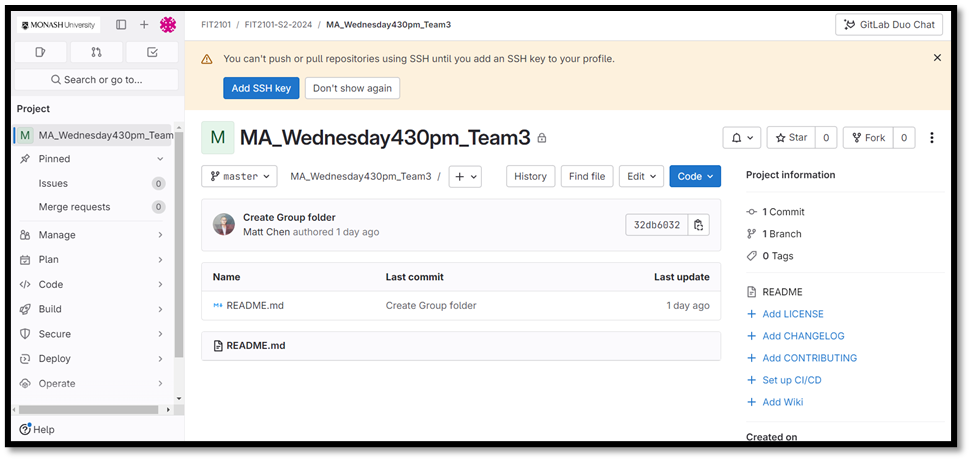
Appendix A: Time Tracking using Contribution Logs excel file



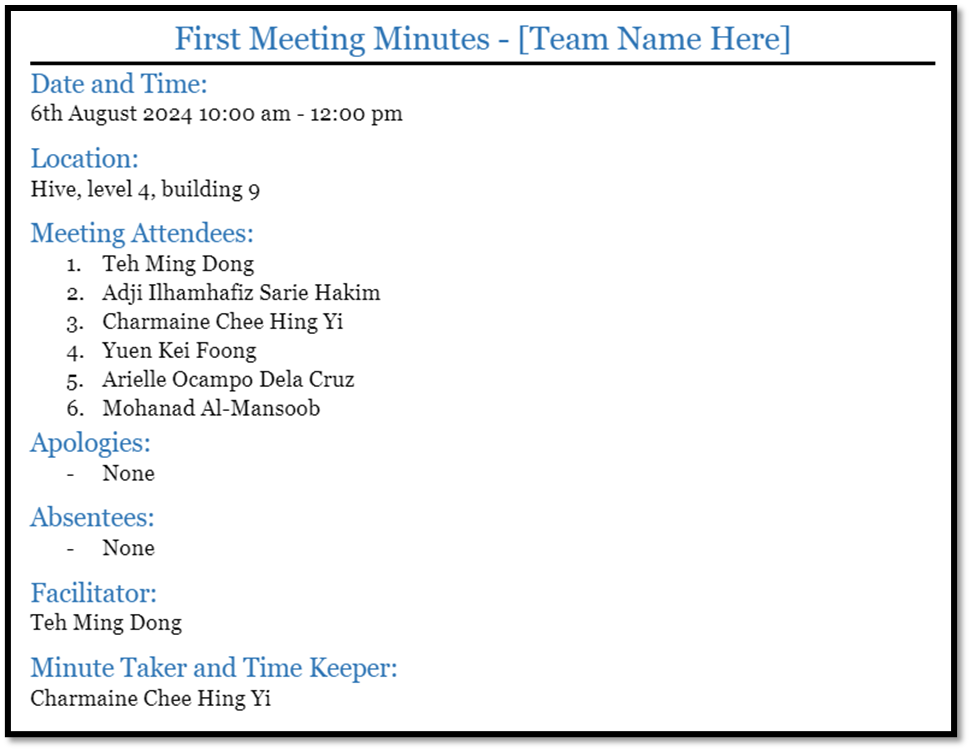
Appendix B: Jira Setup and Usage Guidelines

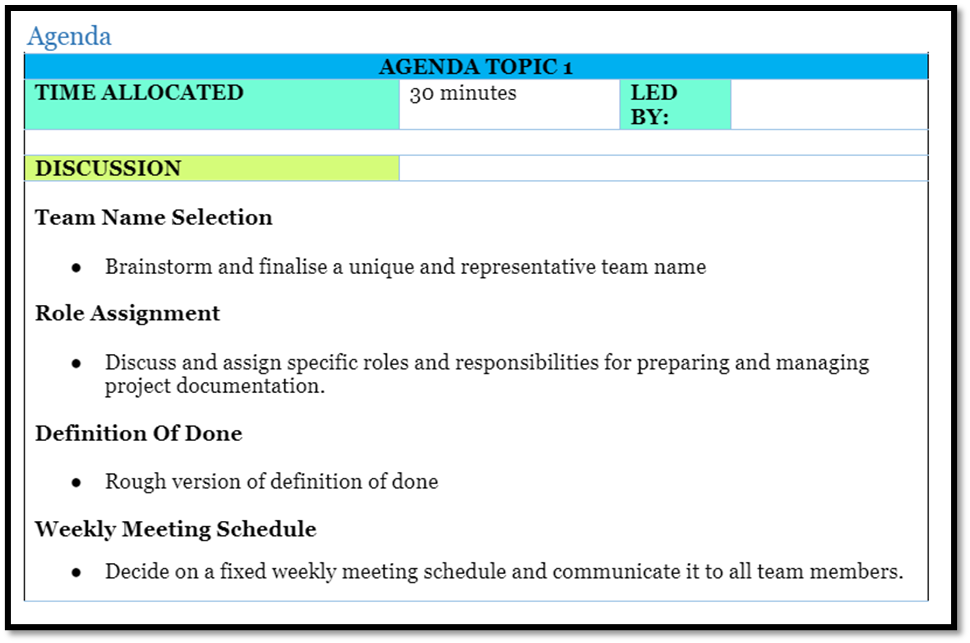


Appendix C: Gitlab Usage Policy Details



Appendix D: Stand-up Meeting Agenda Documents





Appendix E: Scrum process model

