APPCRAFT

PROJECT INCEPTION

2024

Monash University Malaysia

MA_Wednesday430pm_Team3

MONASH University

App Craft Project Plan

Table of Contents

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.

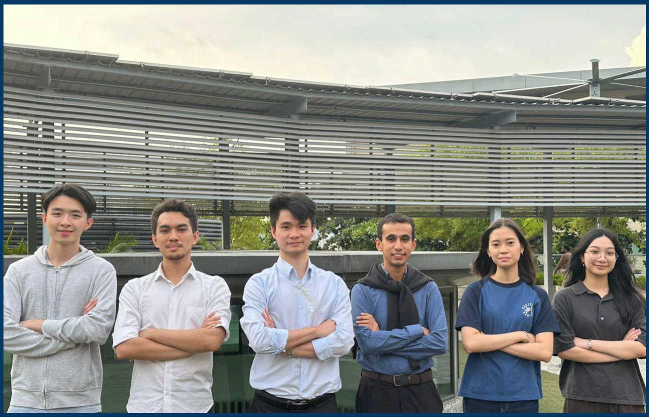
Contact Details:
chong.chunyuan@monash.edu

2.2 Stakeholder Interest

These interests provide a foundation that is essential in the process of decision-making throughout the project's development, acting as a guide to focus on not only meeting the stakeholder's needs but also aligning with the team's current capabilities and resources at hand.

Aspect	Interest
Accessibility	The software should be accessible through the Internet, enabling team members to use it from anywhere in an efficient manner.
Budget	The cost of project development must fall within a low budget, with the team utilising cost-effective solutions for deployment, maintenance and other stages of development.
Integration	The software should have seamless integration with existing tools, which means that it should be able to work well with other existing tools already in use by the team to maximise workflow efficiency.
User Experience	It is of utmost importance for the software to be easy to use. It should be user-friendly and ensure that all users, regardless of their level of expertise, will be able to navigate and gain a firm grasp of how to utilise its features effectively.

2.3 Team Members



We are **AppCraft**, a team of dedicated professionals working together to deliver this project. Our team members starting from left:
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.

2.3.1 Contact Information

Members	Roles & Contact Information
Adji Ilhamhafiz Sarie Hakim	**Quality Assurance (QA) Tester** **Scrum Master (Sprint 2)** 📞 **+60 11 2373 4967** ahak0006@student.monash.edu
Arielle Ocampo Dela Cruz	**Frontend Developer** **Product Owner (Sprint 1)** 📞 **+60 16 876 3846** adel0037@student.monash.edu
Charmaine Chee Hing Yi	**Backend Developer** **Product Owner (Sprint 3)** 📞 **+60 19 341 3190** cche0258@student.monash.edu
Mohanad Al-Mansoob	**Risk & ethical Manager** **Scrum Master (Sprint 1)** 📞 **+967 776 260 065** malm0022@student.monash.edu
Teh Ming Dong	**Lead Developer** **Product Owner (Sprint 2)** 📞 **+60 12 368 8837** mteh0004@student.monash.edu
Yuen Kei Foong	**UI/UX Designer** **Scrum Master (Sprint 3)** 📞 **+60 10 312 9988** kyue0009@student.monash.edu

2.3.2 Role and Responsibilities

Roles	Responsibility
Product owner	Oversees the project timeline, manages client communication and ensures smooth team collaboration. Work closely with clients to gather requirements and transform them into user stories that outline specific product functionalities
Lead Developer	Oversees the project's technical direction, mentors other developers ensures code quality, and takes a proactive approach to identifying and resolving technical debt.
Quality Assurance (QA) Tester	Conducts thorough software testing 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.
Scrum Master	Facilitates the Scrum process by organising and leading Scrum ceremonies (such as daily stand-ups, sprint planning, reviews, and retrospectives).
Ethics and risk manager	Identifies, assesses, and mitigates potential risks that could impact the project's success. Ensures the project adheres to ethical standards and industry regulations, addressing concerns related to data privacy, security, and user rights.

3.0 Team Process Model

The team adopts a modified Scrum process model, an adaptation of the original Scrum model. Originally, there should be daily stand-up meetings that last 15 minutes that the team will attend to report on their progress; however, we opted to replace the daily stand-up meetings with weekly meetings that lasted at least 2 hours. Moreover, our modified Scrum model is limited to only three sprints, each lasting two weeks, while the original has no sprint limit.

Our modified Scrum process also placed the role of product owner and scrum master within the team, unlike the original where the product owner is the client and the scrum master is a person that is outside the team. In addition to having them (product owner & scrum master) involved in the development process of the application since the scale of the project that our team is handling is quite minuscule. Please refer to Appendix A for more detailed information about the scrum process model.

Both in our modified Scrum model and the original, the Product Owner will interview the client to be aware of what the client intended where they would write a user story based on the client's requirements (yet in our model, the interview will be conducted by the whole team member). Furthermore, the Product Owner is tasked to organise the sprint such as selecting tasks from the product backlog and assigning them to the developers. The Scrum Master is responsible for keeping track of the progress of the project while keeping the team under control such that they do not overengineer the product. Moreover, the Scrum Master is responsible for preparing the place for meetings and updating the sprint board.

4.0 Definition of Done

Code Definition of Done

Checklist:

- ☐ The code is complete and adheres to coding standards.
- ☐ All associated test cases are passing successfully.
- ☐ The code has been reviewed and approved by at least one developer or QA member
- ☐ Documentation is updated to reflect the new features or changes.
- ☐ The feature is deployed to the staging environment and validated by the QA team.

Review and Testing:

- ☐ Identified all of the edge test cases in the test plan.
- ☐ units/integration test cases must have passed all of the given acceptance criteria from the code.

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.

Jira, Project management tool Definition of Done

Checklist:

- ☐ The user story and tasks have passed the review process by at least one developer or the QA team
- ☐ Completed all of the acceptance criteria
- ☐ The features are free of bugs
- ☐ Update team members on WhatsApp

Review and Testing:

- ☐ Features must undergo rigorous testing by the QA team before being considered "Done"

Git Definition of Done

Checklist

- ☐ The code is peer-reviewed by members before committing to Git.
- ☐ Merge the current branch with the main branch followed by updating the version number
- ☐ Once the code is integrated the main branch will be merged with the master branch
- ☐ Update version number in the main branch after merging with the current branch, before merging into the master branch.

  For the simplified version (current -> main - > master)
- ☐ It follows the git usage policy in section 6.6.

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 priorities for the upcoming week, ensuring that everyone is clear on their objectives and responsibilities.

5.2 Time Management

Time tracking is essential to ensure that the project progresses smoothly and meets deadlines. In our project, we will implement several Agile techniques to manage time effectively, including Planning Poker, Sprint Retrospectives, and Burndown Charts.

Planning Poker is a collaborative estimation technique used to measure the amount of time or effort required for each task. During a Planning Poker session, team members discuss the task at hand and independently assign a numerical value, usually from the Fibonacci sequence (1, 2, 3, 5, 8, 13...), to represent the estimated effort. After revealing their estimates, team members discuss any differences to reach an agreement, leading to more accurate and realistic time estimates. This collaborative approach ensures that all team members share a common understanding of the work involved and are aligned on the effort required.

Sprint Retrospective is another crucial Agile practice that we will use to refine our time management processes. At the end of each sprint, the team will gather to reflect on what went well, what didn't, and how we can improve in the next sprint. This retrospective allows the team to identify any time management issues, such as tasks that took longer than expected or bottlenecks that slowed progress. By addressing these issues, the team can continuously improve their time estimation and planning processes, making future sprints more efficient and effective.

Finally, we will use a Burndown Chart to track our progress during each sprint. A Burndown Chart is a visual representation of the work remaining in a sprint versus the time left to complete it. It shows a downward trend as tasks are completed, ideally leading to zero work remaining by the end of the sprint. The chart helps the team monitor their progress in real time, quickly identifying if they are on track to meet their goals or if adjustments are needed. If the team is falling behind, the Burndown Chart can prompt discussions during the daily stand-up meetings, allowing for timely interventions and course corrections.

In addition to these Agile practices, the team will use an Excel spreadsheet named "Contribution Logs" to track 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 details of the contribution logs can be found in Appendix B. The Excel file will also track planned start and completion dates for tasks against actual dates. By comparing planned versus actual timelines, we can monitor our progress and identify any discrepancies early. This early detection allows for timely adjustments, ensuring that we stay on track and meet our project deadlines.

By integrating these Agile concepts—Planning Poker, Sprint Retrospective, Burndown Chart—along with a detailed time-tracking system through the Contribution Logs, we can effectively manage our time, continuously improve our processes, and ensure the successful completion of our project. These practices not only help us manage time efficiently but also promote transparency, collaboration, and adaptability within the team.

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 Management

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 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 Management

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.

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, 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. For more information about how we document the stand-up meetings, please refer to Appendix C.

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, where the task will be stored, prioritised and tracked throughout the project lifecycle in the backlog section as shown in Appendix D. 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.
Review Frequency: The backlog will be reviewed and updated at the start of each sprint and as needed during sprint planning or weekly stand-up meetings.

6.6 Git Usage Policy

6.6.1 Version Control and Repository Management

GitLab will be used for version control, 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. For more information about how we store the code in Gitlab, please refer to Appendix E.

6.6.2 Branching

The team will implement a structured Git branching strategy to maintain code integrity throughout the development process. Each team member will be required to create and work within their branch. This approach ensures that all changes are isolated and can be thoroughly tested and reviewed before being integrated into the main codebase.

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 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

This section outlines the factors that have guided the team to choose the tools for our project that align with our project goals, team capabilities and stakeholder interests. The selection criteria include the following:

Team project management tool
When choosing a team project management tool, we focused on looking for a solution that facilitates efficient task management, team collaboration and project tracking. It needed to support agile methodologies such as sprint planning.

Platform
To choose the platform, we evaluated options such as web application, mobile application and desktop application. We considered factors such as accessibility, ease of maintenance, as well as client preference to ensure that the platform chosen would provide a satisfactory and consistent user experience across different devices and environments.

Programming language
Regarding the programming languages to be used in software development, we considered a variety of options such as Java, Python, Ruby etc. The criteria for selection included team familiarity and expertise with the languages, ease of learning and suitability for the project.

Framework (frontend, backend, database)
We considered several frameworks for frontend development, backend development as well as database management. The ideal selection of framework or backend-as-a-service for the team needs to have strong community support for a better learning experience in rapid development so that we can easily integrate and spend more time customising the application for the client's needs.

Web hosting
For web hosting, we sought a solution that offers reliability, scalability and ease of use. It should be able to handle various levels of traffic without performance degradation. In addition, it should be able to support the technologies chosen for development

8.0 Alternative Analysis

In this section, we'll discuss the comparative evaluation of the different platforms available for the AppCraft project. The analysis primarily focuses on the viability, challenges, and benefits of each alternative.

8.1 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 on the device	Offline and online as long as it is installed on the mobile device	Anywhere with an internet connection
Offline access	Limited functionality, to prevent crashes with each update	Limited functionality, to prevent crashes with each 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	Depending on internet speed, server and device 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

Aspects	Desktop Application	Mobile Application	Web Application
			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 good quality and accessible project management tool that meets the diverse needs of its users.

MONASH University

8.2 Programming analysis

Aspects	Java	JavaScript	Python	Go	Ruby	Dart
Team experience	Medium	Low	High	Low	Low	Low
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).	A popular choice for Backend as a service (BAAS), cloud computing and microservices.	Popular in web development (Rails framework); known for ease of use and productivity. Often used by startups.	Ideal for cross-platform mobile development (Flutter); can also be used for web and desktop apps, though less common.
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 support multithreading, and building native mobile applications.	Known for high performance and efficient concurrency; excellent for scalable backend services.	Moderate performance; Ruby on Rails abstracts many details, which can slow down execution in complex apps.	Fast performance on mobile (with Flutter); Dart is compiled ahead-of-time (AOT) for high efficiency.

Integration	All can be integrated effectively using frameworks and microservices.			Integrates well with cloud platforms and backend services;	Integrates well with web technologies and databases via Rails; also used for automation and scripting tasks.	Integrates smoothly with mobile hardware and services; can be used with Firebase for backend integration.
Community and supports	All have a rich ecosystem with extensive libraries and strong support.			Growing community in cloud and backend services; solid libraries for networking and concurrency.	Strong community support for Ruby on Rails, which simplifies many web development tasks.	Growing community with flutter; the libraries are expanding rapidly.
Web Frameworks	Spring	React, Angular	Django, Flask	No major web frameworks, but can be used with minimalistic frameworks like Gin, Echo for backend services.	Ruby on Rails dominates; perfect for rapid web development with a convention over configuration approach.	Flutter for mobile/web; Dart's web frameworks are still maturing but are gaining traction (e.g., Aqueduct).
Database	MongoDB, firebase	MongoDB, firebase	Firebase	MongoDB with SQL for NoSQL	MongoDB with NoSql, MySQL/ PostgreSQL via ActiveRecord in Rail	Typically used with Firebase for mobile apps; can work with any backend database when needed.

Based on our analysis, we've decided to use JavaScript and Python for our project. The team has strong experience with Python, making it the best choice for backend development. Python is known for its flexibility in areas like data analysis, machine learning, and backend work. While it might not be the fastest language, this isn't a big issue for us since our project doesn't need high-speed computing. Python also integrates well with databases, especially when using frameworks like Django and Flask, which will help our application scale and handle data efficiently.

For the front end, JavaScript is essential. It's fast in browsers and can also be used on the server side with Node.js, making it perfect for creating interactive and responsive web interfaces. Although the team has less experience with JavaScript, it's widely used in web development and has strong community support, along with many libraries that make development easier. This will help us quickly build and connect our frontend with cloud services and backend systems. By using Python for the backend and JavaScript for the front end, we can create a well-rounded and scalable solution that meets our project's needs.

MONASH University

8.3 Framework and Service Analysis

Framework	Frontend/ Backend/ Back-end as a service	Language	Ease of Use	Performance	Community Support	Usage
Django	Backend	Python	Clear documentation with admin interface for quick development	Moderate due to some complexity overhead	Strong community with extensive third-party packages and plugins.	Ideal for web application in rapid development.
Flask	Backend	Python	Easy to learn and use. minimalistic and flexible, allowing developers to structure projects	High performance due to its lightweight nature; can be optimised easily with external libraries.	Strong community support, though smaller than Django; lots of extensions available for added functionality.	Ideal for small to medium RESTful APIs and projects.
Spring	Backend	Java	Syntax learning is hard for everything to be able to integrate with frontend and others.	High performance; well-optimised for large-scale,	Large and strong community with a vast ecosystem of plugins and modules.	Perfect for enterprise-level web applications, large-scale systems, and services.
React	Frontend	JavaScript TypeScript	Easy to learn and use. Many reusable components and resources.	High performance; virtual Document Object Model (DOM) ensures efficient updates and rendering.	Large community with many tutorials, plugins and libraries.	Ideal for dynamic single-page web applications.
Angular	Frontend	JavaScript, TypeScript	requires familiarity with TypeScript.	High performance with real DOM	Strong community with backing from Google; many tools and libraries.	Ideal for a large user base of web applications with complex data binding and business logic.
Vue.js	Frontend	JavaScript,	Very easy to use due to	High performance due to	Growing community with	Ideal for up to medium-sized

		TypeScript	its simplicity from good templates.	optimised virtual DOM.	strong support, easy-to-find plugins and resources.	projects.
Node.js + Express	Backend	JavaScript	Easy to learn for JavaScript developers; minimalistic with a focus on building fast, lightweight web servers.	High performance; known for its non-blocking, event-driven architecture, making it excellent for I/O-bound applications.	Massive community with extensive resources, libraries, and plugins; widely used in full-stack JavaScript development.	Ideal for backend services, APIs, and real-time applications; used in full-stack development with JavaScript on both client and server sides.
Gin/Echo	Backend	Go	Easy to use due to good documentation and simplicity in setting up projects.	High performance due to Go’s efficient concurrency model.	Growing community, particularly in cloud services and microservices; solid support from Go developers.	Best for building high-performance backend services, APIs, and microservices; great for cloud-native applications
Ruby on Rails	Backend	Ruby	Very easy to use with a focus on convention over configuration.	Moderate performance; some overhead due to its abstractions, but can be optimised.	Strong community with many libraries and extensive documentation	Best for rapid development of web applications, especially for startups and MVPs
Flutter	Frontend	Dart	Easy to learn, especially for mobile developers; a single codebase for multiple platforms simplifies development.	high performance on mobile due to AOT compilation	Rapidly growing community with strong support from Google. Many resources can be found.	Ideal for cross-platform mobile app development.
AWS Amplify	Backend as-a-service	JavaScript, TypeScript, Python, Swift, Kotlin, Java,	Easy to use with a broad range of tools for backend development, including serverless functions, storage, and	High performance, leveraging AWS's global infrastructure; optimised for various use cases, including mobile and web.	Large community with extensive resources, integration with AWS services, and strong enterprise support.	Best for building scalable mobile and web applications with strong integration into AWS services,

		Flutter, React Native	authentication.			
Firebase	Backend as-a-service	JavaScript, TypeScript, Java, Python, Go, C++, Swift, Dart	easy to use; managed backend services with powerful SDKs; allow developers to focus on frontend logic	High performance; optimised by Google for real-time applications and efficient backend processing.	Strong and growing community; extensive documentation and integration with Google Cloud services.	Ideal for rapidly building mobile and web apps; great for real-time features, authentication, databases, and serverless functions.

We chose React and Firebase for our project because they offer a rapid development environment and have extensive resources available, including numerous tutorials and a large library of tools and components. Our primary goal is to focus on the front end, an area where we have less experience and expect a steep learning curve. React is well-known for its ease of use, especially for developers new to front-end development. Its component-based architecture allows us to build dynamic user interfaces efficiently, while its huge community support ensures that we can find solutions to challenges quickly.

On the backend, we opted for Firebase as a Backend-as-a-Service (BaaS) solution. Firebase abstracts much of the complexity associated with backend development, such as managing servers, databases, and authentication. This allows us to concentrate on building the front end without getting bogged down in backend details. Firebase's integration with Google Cloud also provides scalability and reliability, which is essential as our application grows.

In comparison, we considered using the MEAN (MongoDB, Express, Angular, Node.js) and MERN (MongoDB, Express, React, Node.js) stacks, which are popular for full-stack JavaScript development. While these stacks offer a cohesive environment for both frontend and backend development, we decided against them because of the additional complexity they introduce. The MEAN stack, with Angular as the frontend framework, has a steeper learning curve due to its more opinionated and structured nature, which can be challenging for developers who are not familiar with TypeScript or Angular's architecture. The MERN stack, while using React for the front end, would still require us to manage a Node.js and Express-based backend, which adds complexity that we wanted to avoid.

By choosing React and Firebase, we can streamline our development process, allowing us to focus on mastering frontend development with React while relying on Firebase to handle the backend tasks. This approach aligns with our project goals, enabling us to develop quickly and efficiently without being overwhelmed by the challenges of full-stack development.

MONASH University

8.4 Project Management Analysis

Tools	Key Features	Ease of Use	Collaboration	Quality Management	Task management	Best use case	Chart Representation	Difference Compared to Others
Jira	Agile boards, backlog management, roadmaps, issue tracking.	Moderate; requires some setup and learning curve.	Strong collaboration tools, including comments, notifications, and mentions; ideal for Agile teams.	Quality control through issue tracking, customizable workflows, and integration with testing tools.	Excellent task management with detailed issue tracking, custom workflows, and automation.	Best for software development teams, especially those following Agile methodologie s; suitable for large teams.	Supports various chart representations including burndown charts, Gantt charts (with plugins), and sprint reports.	**Specialised for Agile teams**: Unlike most other tools, Jira is specifically designed for software development teams following Agile methodologies, with features like sprint planning and backlog management that are deeply integrated into its workflow.
Trello	Kanban boards, task cards, checklists, Power-Ups.	Very easy to use; intuitive drag-and-drop interface.	Simple collaboration through task assignments, comments, attachments, and notifications.	Basic quality management via checklists and task cards; Power-Ups can extend capabilities.	Simple task management with easy-to-use boards and card systems, great for small teams.	Great for small teams, startups, and personal task management; ideal for simple project tracking and to-do lists.	Kanban boards are the primary chart representation; support basic progress tracking and task visualisation.	**Simplicity and Flexibility**: Trello stands out for its simplicity and visual approach with Kanban boards. Unlike tools like Jira or Microsoft Project, Trello is highly intuitive with minimal setup, making it accessible to non-technical users.
ClickUp	Task management, goal tracking, docs, timelines.	Easy to use; feature-rich interface can	Strong collaboration through comments,	Quality management with	Versatile task management with features	Great for teams of all sizes,	Offers Gantt charts, timelines,	**All-in-One Solution**: ClickUp combines features from various

		be overwhelming	task assignments, shared docs, and real-time chat.	customizable workflows, automation, and task dependencies.	like goal tracking, timelines, and task automation.	especially those needing a versatile and comprehensive project management solution.	workload views, and custom dashboards for detailed project tracking.	tools into one platform, offering everything from goal tracking to document collaboration, making it broader in scope than Jira, Trello, or Asana.
Monday.com	Custom workflows, automation, time tracking, dashboards.	Easy to moderate; customizable but may require setup.	Strong collaboration with shared boards, task assignments, and communication within tasks.	Quality management via automation, custom workflows, and reporting tools.	Comprehensive task management with time tracking, automation, and custom workflows.	Best for teams needing a highly customizable project management tool; suitable for diverse industries and remote teams.	Supports Gantt charts, timelines, and custom dashboards for visual project management and reporting.	**Customization and Visual Tracking**: Monday.com differs by offering a high degree of customization in workflows and visual project tracking, which is more flexible than Asana and Trello but less specialised for Agile than Jira.
Notion	Wiki, notes, task management, databases, kanban boards.	Easy to use; highly customizable but requires initial setup.	Strong collaboration on shared documents, databases, task boards, and real-time editing.	Basic quality management through databases and task tracking; customization is needed for more advanced features.	Flexible task management with Kanban boards, to-do lists, and databases.	Best for teams looking for an all-in-one workspace; ideal for knowledge management, documentation, and lightweight project management.	Kanban boards and database views are the primary chart representations customizable views allow for personalised tracking.	**Documentation and Flexibility**: Notion excels in combining project management with knowledge management (wiki and databases), unlike other tools that primarily focus on task and project management.

We chose Jira over other project management tools due to its specialised features tailored for Agile software development teams. Jira provides comprehensive task management through detailed issue tracking, custom workflows, and automation, making it ideal for managing complex projects with multiple sprints and detailed requirements. Its strong collaboration tools, including comments, notifications, and real-time updates, ensure effective communication within the team, which is crucial for Agile methodologies. Additionally, Jira's integration with quality management tools allows for better task tracking and resolution, ensuring that the final product meets high-quality standards. The support for various chart representations, such as burndown charts and Gantt charts, further enhances project visibility and progress tracking. Unlike other tools like Trello or Notion, which are more generalised and simple, Jira is specifically designed for Agile teams, offering features deeply integrated into its workflow that are essential for software development projects. This makes Jira the most suitable choice for our team, particularly given our need for a tool that can handle complex project management tasks efficiently while supporting Agile practices.

8.5 Web-Host server analysis

Web Host	Key Features	Storage	Bandwidth	Custom Domain
Github pages	Static site hosting, version control, custom domains via CNAME	1 GB	Unlimited	Yes
000WebHost	cPanel-like interface, PHP, MySQL support	300 MB	3GB/ month	No
Wix	Drag-and-drop site builder, templates, SEO tools	500 MB	1 GB/month	No
Freehostia	1-click apps install, clustered servers	250 MB	6 GB / month	No
AwardSpace	1-click CMS instals, free domain for subdomains	1 GB	5 GB /month	No

When comparing various free web hosting options, GitHub Pages stands out for its unique offering of static site hosting with version control and the ability to use custom domains via CNAME. It provides 1 GB of storage with unlimited bandwidth, making it a good choice for us focused on hosting static websites compared to others.

8.6 Recommendation

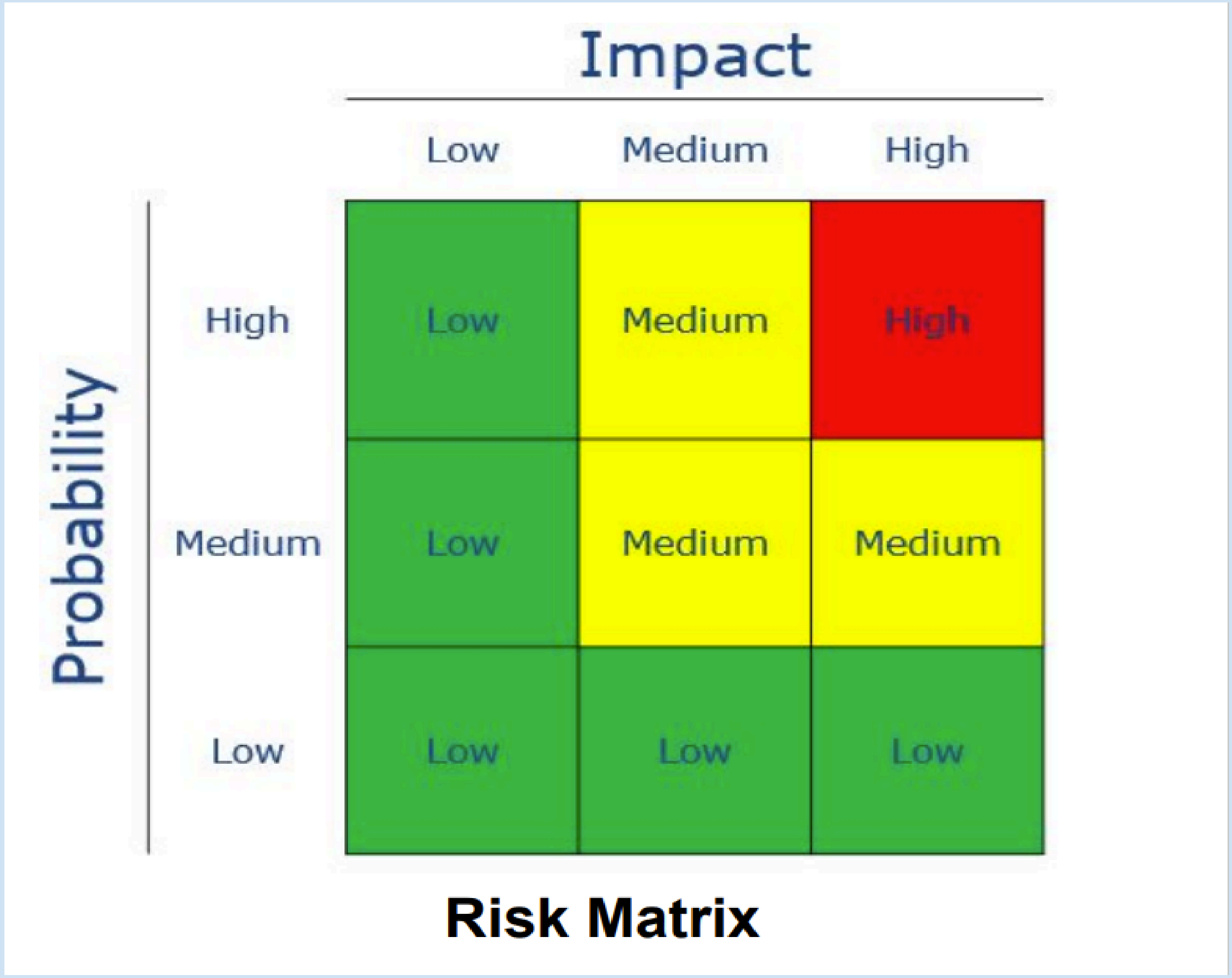
Based on our thorough analysis, we recommend using React for the frontend, Firebase as the backend-as-a-service (BaaS), and GitHub Pages for hosting services. React offers a powerful, flexible, and widely adopted framework for building dynamic user interfaces, making it an excellent choice for creating responsive and interactive applications. Firebase, with its robust BaaS capabilities, simplifies backend management, providing seamless integration with real-time databases, authentication, and cloud storage, which accelerates development time and reduces the need for server maintenance. Finally, GitHub Pages is an ideal hosting solution for static sites, offering reliable performance with unlimited bandwidth and custom domain support, all integrated within the developer-friendly GitHub ecosystem. This combination of tools will ensure a streamlined development process, efficient backend management, and reliable hosting, ultimately delivering a high-quality product to the client.

8.7 Cost of changing different frameworks

Switching from React and Firebase to another framework and backend, such as Angular, Vue.js, Next.js, SvelteKit, Django, or Ruby on Rails, will impact the project in several ways. The learning curve may vary, with Angular and Django introducing more complexity, while Vue.js and Next.js might be easier transitions. Integration with Firebase alternatives like Supabase or a serverless backend (e.g., AWS Lambda) would require significant changes in backend logic, potentially slowing development.

Risk Management

This section will outline the risk that our team registers for this project. We will be using the risk matrix to determine the severity of the consequences of the risk.



Probability \ Impact	Low	Medium	High
High	Low	Medium	High
Medium	Low	Medium	Medium
Low	Low	Low	Low

Risk Matrix

ID	Date Raised	Risk Description	Likelihood	Impact to the project	Severity	Mitigating Action	Contingency Plan	Risk Owner	Monitoring Strategy
1	13/08/2024	Mental Health Crisis Team member experiences 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.	Product Owner	- Encourage team members to watch out for signs of stress or burnout among their peers and report any concerns to the project manager

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.	Scrum Master	- Encourage team members to look out for one another
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	Lead Developer	- Maintain and regularly review a technical debt log, where developers record instances of 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	Product Owner	- Implement real-time system monitoring tools to keep track of software, hardware, and network performance such as OpManager
5	20/08/2024	Miscommunications between members miscommunication within team members that resulted in unclear minimum viable product (MVP)	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 weekly 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.	Product Owner	- Implement a practice of distributing brief meeting recaps after every significant discussion

6	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.	Product Owner	- Track attendance and participation in meetings or work sessions because sudden absences could indicate that a team member has encountered an issue like an accident.
7	20/08 /2024	Minor Illness 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.	Product Owner	- Monitor the usage of sick leave days. An increase in sick leave could indicate that an illness
8	20/08 /2024	Conflicting client's requirements. (what we received and client's imagination is different)	MEDIUM	Differences between client expectations and delivered work can lead to significant rework, increased costs, and potential	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	- schedule an immediate meeting with the client to discuss and reconcile the differences.	Product Owner	- Schedule regular check-ins with the client to discuss project progress and ensure that the

		A situation where the requirements provided by the client differ from what the client actually envisions or expects.		delays in project delivery.		the project to ensure that their expectations are being met - Maintain clear and detailed documentation of all requirements and client communications.			development is aligned with their expectations.
9	21/08/2024	Copyright or Patent Infringement 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. - Maintain detailed documentation of all third-party materials used in the project, including their licences and sources.	- immediately stop the usage of the infringing material and remove it from the project. Replace it with compliant materials.	Risk Ethics Manager	- Periodically review all materials used in the project to ensure they comply with copyright and patent laws, including verifying the licences of any third-party software, images, or content.

9.0 Conclusion

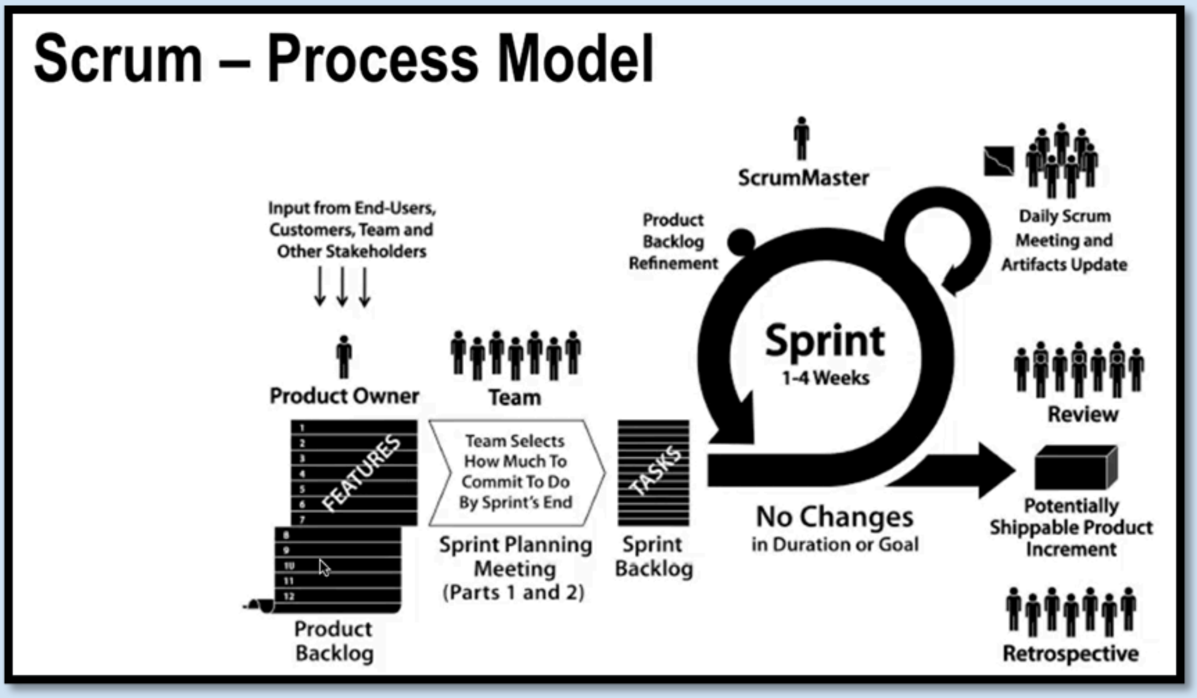
In conclusion, the AppCraft project plan outlines a structured approach to developing a strong project management tool tailored for agile workflows. Through clear definitions of roles, responsibilities, and processes, our team is well-prepared to deliver a high-quality product that meets the needs of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

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.

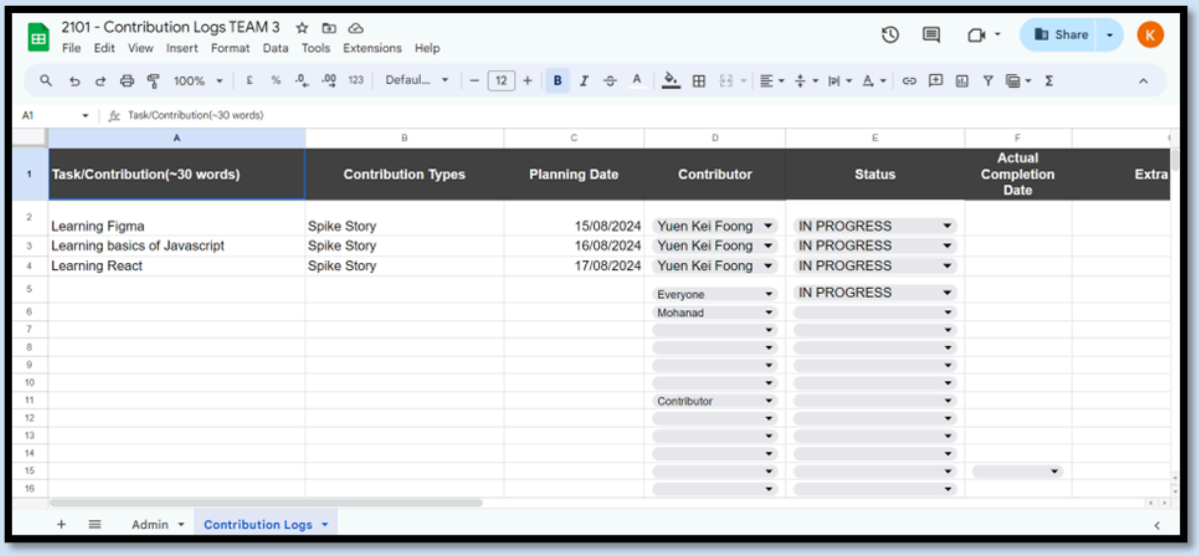
As we move forward, we will continue to monitor our progress closely, adapt to any challenges that arise, and maintain our focus on delivering a product that not only meets but exceeds expectations. This project plan can serve as our guide, ensuring that all team members are aligned and working towards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

10.0 Appendix

Appendix A: Scrum process model


Scrum – Process Model
Input from End-Users, Customers, Team and Other Stakeholders
Product Owner
Team
FEATURES
Product Backlog
Team Selects How Much To Commit To Do By Sprint's End
Sprint Planning Meeting (Parts 1 and 2)
TASKS
Sprint Backlog
Product Backlog Refinement
ScrumMaster
Sprint
1-4 Weeks
Daily Scrum Meeting and Artifacts Update
No Changes in Duration or Goal
Review
Potentially Shippable Product Increment
Retrospective

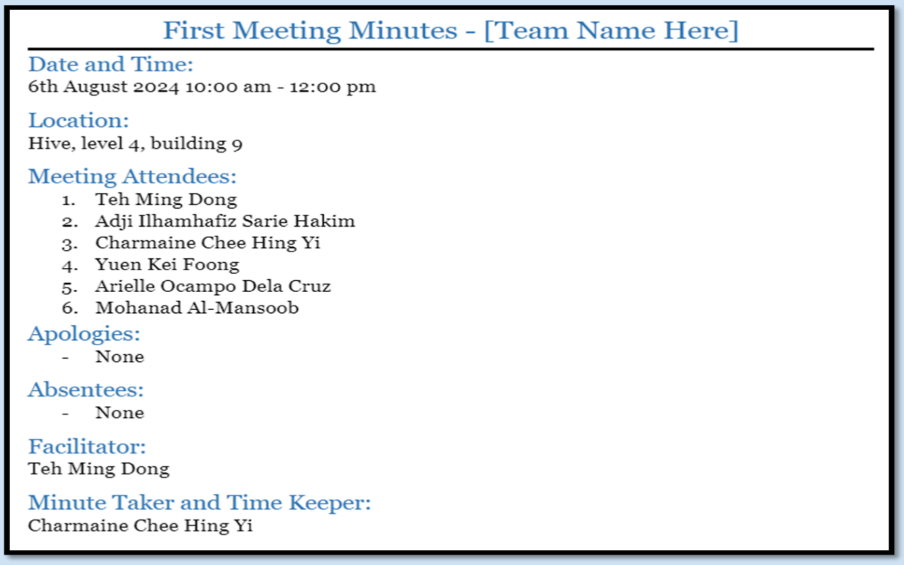
Appendix B: Contribution Logs


2101 - Contribution Logs TEAM 3

	A	B	C	D	E	F	
1	Task/Contribution(~30 words)	Contribution Types	Planning Date	Contributor	Status	Actual Completion Date	Extra
2	Learning Figma	Spike Story	15/08/2024	Yuen Kei Foong	IN PROGRESS		
3	Learning basics of Javascript	Spike Story	16/08/2024	Yuen Kei Foong	IN PROGRESS		
4	Learning React	Spike Story	17/08/2024	Yuen Kei Foong	IN PROGRESS		
5				Everyone	IN PROGRESS		
6				Mohanad			
7							
8							
9							
10							
11				Contributor			
12							
13							
14							
15							
16							

File Edit View Insert Format Data Tools Extensions Help
Share
A1
Task/Contribution(~30 words)
Admin
Contribution Logs

Appendix C: Stand-up Meeting Agenda Documents



First Meeting Minutes - [Team Name Here]

Date and Time:
6th August 2024 10:00 am - 12:00 pm

Location:
Hive, level 4, building 9

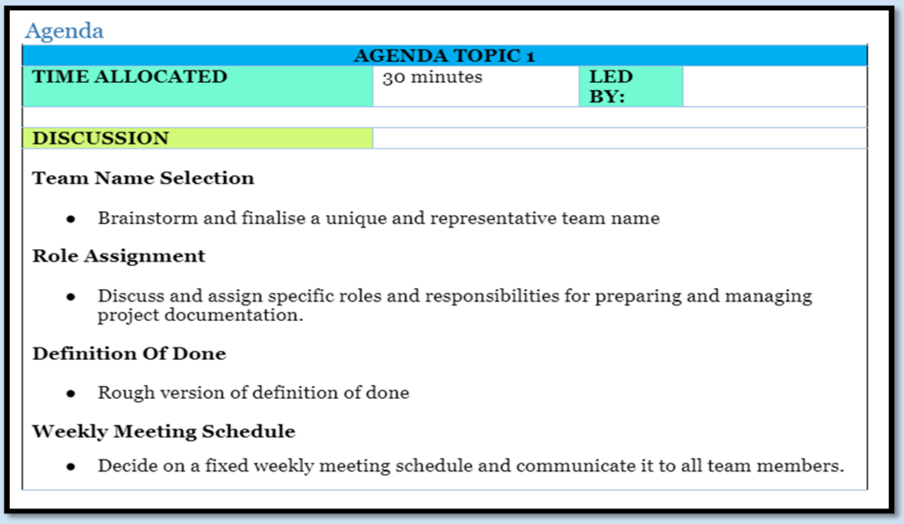
Meeting Attendees:
1. Teh Ming Dong
2. Adji Ilhamhafiz Sarie Hakim
3. Charmaine Chee Hing Yi
4. Yuen Kei Foong
5. Arielle Ocampo Dela Cruz
6. Mohanad Al-Mansoob

Apologies:
- None

Absentees:
- None

Facilitator:
Teh Ming Dong

Minute Taker and Time Keeper:
Charmaine Chee Hing Yi



Agenda

AGENDA TOPIC 1			
TIME ALLOCATED	30 minutes	**LED BY:**	
DISCUSSION			

Team Name Selection

- Brainstorm and finalise a unique and representative team name

Role Assignment

- Discuss and assign specific roles and responsibilities for preparing and managing project documentation.

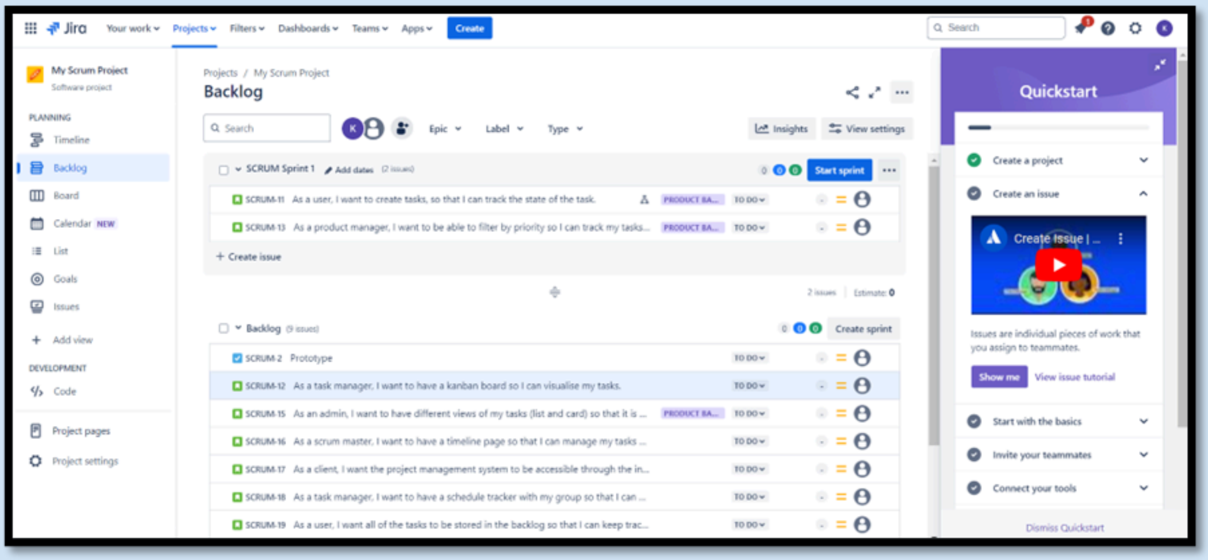
Definition Of Done

- Rough version of definition of done

Weekly Meeting Schedule

- Decide on a fixed weekly meeting schedule and communicate it to all team members.

Appendix D: Jira usage



SCRUM Sprint 1 Add dates (2 issues) — Start sprint

SCRUM-11	As a user, I want to create tasks, so that I can track the state of the task.	PRODUCT BA...	TO DO
SCRUM-13	As a product manager, I want to be able to filter by priority so I can track my tasks...	PRODUCT BA...	TO DO

+ Create issue

2 issues | Estimate: 0

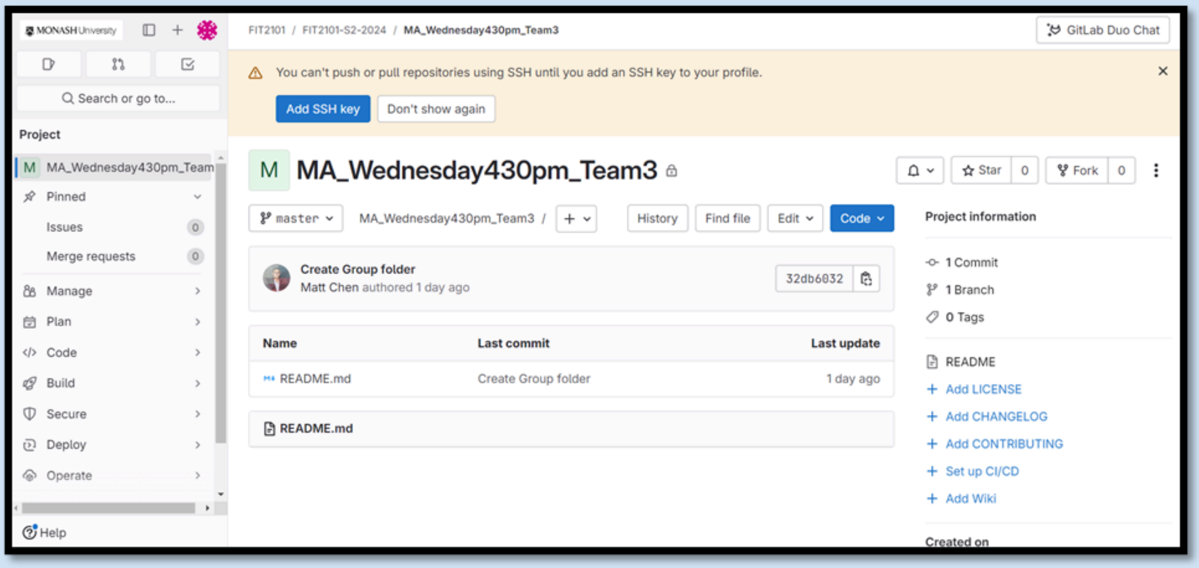
Backlog (9 issues) — Create sprint

SCRUM-2	Prototype		TO DO
SCRUM-12	As a task manager, I want to have a kanban board so I can visualise my tasks.		TO DO
SCRUM-15	As an admin, I want to have different views of my tasks (list and card) so that it is ...	PRODUCT BA...	TO DO
SCRUM-16	As a scrum master, I want to have a timeline page so that I can manage my tasks ...		TO DO
SCRUM-17	As a client, I want the project management system to be accessible through the in...		TO DO
SCRUM-18	As a task manager, I want to have a schedule tracker with my group so that I can ...		TO DO
SCRUM-19	As a user, I want all of the tasks to be stored in the backlog so that I can keep trac...		TO DO

Issues are individual pieces of work that you assign to teammates.

Show me View issue tutorial

Appendix E: Git Usage



You can't push or pull repositories using SSH until you add an SSH key to your profile.

MA_Wednesday430pm_Team3

Create Group folder
Matt Chen authored 1 day ago
32db6032

Name	Last commit	Last update
README.md	Create Group folder	1 day ago

README.md

AI declaration

Ai is used for grammar checking. The whole paragraph is thrown into chatgpt as input and restructured the passage and shortened down into table form.