SENG3011 Management Information
Bobby Tables

Project Management

This project is of high importance to us so we have put a great deal of care into making sure we are able to **effectively work together** as a team throughout the term, produce a **high-quality system**, and **stay on top of deliverables.**

Roles & Responsibilities

We have allocated responsibilities to each member to ensure the work is divided equally, optimise our team's efficiency, and allow everyone to sharpen different skill sets.

Role	Member	Responsibilities
Project Manager	Michael Chen	● Michael is in charge of **making sure the project is on track**. To do this, he will **monitor** how each member is going with their tasks and **resolve any blockers** they encounter to enable them to do their best work. ● He will analyse each individual's contribution and **provide feedback** when necessary to keep the project running smoothly. ● On behalf of our team, he will be the person in charge of **discussing with the mentor** to raise issues, seek feedback, and sort out any uncertainties. ● Furthermore, he will set the dates of **meetings**, decide whether additional **ad hoc stand-ups** are necessary, and constantly **review deliverable specifications.** ● While the team is working on their tasks for the upcoming deliverable, Michael will be **looking ahead** to ensure we are also **prepared for the next one.**
Scrum Master	Jamie Domingo	● While Michael is in charge of the project from a more high-level perspective, Jamie's duties revolve around the **current deliverable** and **managing weekly sprints** throughout the term. ● Jamie is responsible for all the **project management tools** used by the team. She will **set up and maintain** the team's communication channels, file management space, and sprint board. ● Jamie will **run the weekly meetings**, taking notes to **keep record of the team's progress**, and continuously update the team's **living documents** to ensure they are always accurate. ● In team meetings, she will conduct a **sprint review**, encourage a **discussion** of what tasks were completed, and will execute a **sprint planning** session to ensure all members are aware of their tasks for the upcoming week.

Product Manager	Jingbang Zhou (Ben)	• Ben is responsible for the **overall system** that the team will produce. • He is in charge of reviewing **deliverable specifications**, monitoring how we meet **customer requirements**, and ensuring we are providing a **positive user experience**. • Ben will **conduct meetings with the client** when necessary to discuss their needs and how our system can provide them with value. • Thus, Ben will look at our product from a **holistic** perspective to ensure we maintain a **customer-centric focus** and will serve as the **team's communication channel with the customer.** • From the team's side, Ben is in charge of ensuring our deliverables are of **high quality**. He will **review our reports, fix formatting issues, and conduct testing** according to our user acceptance criteria to improve our system's user experience.
Backend Technical Lead	Cooper Bartlett	• Cooper is the **technical lead** for the **backend** of our system. He will make decisions on the **backend architecture**, **adjust** it according to customer requirements, and he will also be in charge of **setting it up**. • He will **inform** the team about how the backend communicates with other components, and ensure each team member **understands how it works**, and how to make **valuable code contributions** according to their tasks. • He will **sort out any bugs** that arise within the backend, help the team with **technical issues**, and **produce explanations and diagrams** when necessary.
Frontend Technical Lead	William Sieber	• William and Cooper share similar responsibilities, however, William is responsible for the **frontend architecture** of our system. • In addition to **setting up the architecture, helping the team** make code contributions, and **fixing frontend bugs**, William will work with Ben, our product manager, to ensure the **customer has a smooth and positive user experience.** • Thus, in addition to maintaining the frontend code base, William is in charge of our **product's design system** and will make sure the customer can use the system according to their needs. • Both Cooper and William will maintain the team's **code repository.**
Developer	Everyone	• As software engineers, every team member will be allocated **software tasks.**

		• We will make **code contributions** to our repository, ensure we are **writing clean code** and following **good software practices**. • This includes **writing and running tests** frequently to ensure our system is working as intended. • We all share the goal and responsibility to **build a high-quality system** that caters to the **customer's needs** and provides **value to our users.**

Project Plans

As a team, we have established a plan to stay on track for this first deliverable. After submission, we will evaluate how it went through a **retrospective** and modify it accordingly. This way, we can make sure that we are constantly **reflecting, optimising how we work together,** and **setting ourselves up for success.**

Every Monday at 8pm, we will hold a **team meeting**. In this meeting, we will do a **sprint review** of the past week using our **Jira sprint board** and take turns to individually discuss what we have each completed. At the end of this meeting, we will **close the sprint and start planning** the next week ahead. We will also use this time to make preparations for our mentor meeting the following Tuesday at 7pm. We have chosen to work using an **agile approach** because quick, speedy iterations will allow us to **deliver value faster, test early** and **continuously improve our solution.**

We have decided to use the following project management tools:

- **Jira** will be used to keep track of our **weekly sprints**, conduct our **sprint reviews and sprint planning**, and observe our **project roadmap**.
  We will have a **Kanban board** with the following columns: To Do, In Progress, Review, Done. Each deliverable will be represented by an **Epic** consisting of multiple **'Issues'** or tasks. Issues will have a deadline and a person assigned to keep track of it. This will allow us to **visualise our progress** throughout the sprint.
  We chose to use Jira as it makes **agile project management** easy to **execute and maintain**. Although it can be quite complicated to learn at first, we decided to use it as it provides plenty of features, can **link to our Github** repository, and is encouraged by the course admin. Furthermore, it is used in industry and gaining exposure to it now would be helpful for us in the future.
- Our **Confluence** space will be used as our **file management system**. This is where we will store our **meeting minutes, brainstorms and drafts**. We will also utilise the Confluence **space shortcuts** to quickly access our team documents and make the most of the mentions and **notifications platform** to keep each other informed.
  We decided to use Confluence as not only was it prescribed to us by the course, it has many **templates** to speed up our workflow. We will use templates such as 'Sprint Planning', 'Meeting Minutes' and 'Retrospectives' to keep our space **neat, consistent and organised**. We will allocate each deliverable a folder to keep track of all relevant documents and notes. We found that Confluence was the best tool to use to maintain our **living documents** as it is easy to update and keep track of.
- For **team communication**, we have decided to use the **Discord** application. This is where we will

hold offline discussions and run our weekly meetings. We chose Discord because it is very **lightweight**, **easy to use**, and everyone already has it. Furthermore, it is extremely convenient that the **voice channel is open 24/7**, as there is no need to schedule and start a call. Through the use of **different channels and threads**, Discord allows us to organise our discussions so that information is not lost and messages are easy to locate.

- **Microsoft Teams** is the prescribed tool to **communicate with our mentor**. We will also use this application to send **calendar invitations** for our sprint reviews and conduct our Tuesday mentor meetings. Since Teams is already connected to our UNSW accounts, we found it the best choice for all **scheduling purposes**.
- **Google Drive** will be used for conducting **collaborative work** with each other, such as rough drafts, brainstorming and individual work. We have chosen to use a combination of Google Drive and Confluence as we believe the features they provide will serve us in different ways. Since Google Drive uses a more **traditional approach** to writing, we will utilise it to curate our **deliverable submissions**. This application was chosen as the team is extremely **familiar** with its interface and we favour **not having to formally publish documents,** unlike Confluence. However, Confluence will still be used for file management and organisation.
- A **Github** repository will allow us to manage our code as the project progresses. We will utilise its features to optimise our **version control, automate our testing and review each other's work**. We chose this application over Gitlab and BitBucket as we all have Github accounts, we have used it for many **previous projects**, and it can easily be **integrated with our Jira board**.

Software Practices

To ensure our product is high-quality and functional, we have adopted the following good software practices to adhere to.

We will use a separate **'master' branch** for our **production-ready code**. We will set up **Github Actions** to automate the building, testing and deployment of our code when branches are merged in. This ensures the code in production will not crash and prevents **code regression**. By automating this, we are also able to work more **efficiently** and can **catch bugs** as soon as they arise.

When completing a **Jira ticket**, we will **identify the issue in our branch name as a prefix**, followed by a concise description of what the code in the branch does. For example, if the Jira ticket was "BT-14", we would name our branch: "BT-14/description-of-what-branch-does". Similarly, commit messages will also be written in this format. This allows us to keep track of how our issues are progressing and **keep our code organised**. We can also **see who is assigned** to the code by reviewing the ticket in Jira. When browsing through our code history, it will be easy to read as the branches will all be written in a **consistent format**.

When we have completed the code on our branch and it is ready to be pushed into master, we will first **create a pull request** so that others can review what we have written. This will ensure that all changes made to our system are **visible** to all team members. Having another person to review it will motivate us to write **cleaner, more readable and optimised code**. Sharing changes, suggestions and explanations will also encourage a **culture of learning** and will ultimately help us all to become better programmers. Only when a team member has **approved** the pull request, most likely one of the technical leads, will the code

be **allowed to be merged into master**. Again, Github Actions will ensure that no **merge conflicts** exist, the code is **building successfully**, and it is passing our **automated test suite**.

The technique we will use for merging our code will be the **Squash method**. This ensures that when we review the history, it reflects the changes made by the branches **accurately** and we can **easily roll back** to different versions if needed. We have chosen this technique over Rebasing as we want to avoid having to deal with **complicated history errors** and **confusing merge conflicts**. Although the Rebasing method is linear and more aesthetic, we favour how **precise** the Squash method **visualisation** is in the code history.

Other software practices we will adopt to avoid code smells are:

- Rather than writing everything into one long file, we will **isolate components** and abstract helpers and functions away into **importable modules**. This will prevent **code bloat** as we add more functionality throughout the term.
- We will use a **consistent file naming system** to keep our code **organised**.
- When possible, we will create **reusable functions and components** to avoid **duplicate code**.
- We will use **themes and style files** to ensure our design system is **consistent** and make it easier to create a **positive user experience**.
- Variables will be given **meaningful names** and we will follow **code conventions**, such as initialising constants with uppercase variable names.
- We will ensure that when we are making queries to our API, the queries are **not fetching unnecessary information** and are thus **optimised for performance**.