(CMP020N204S) Software Engineering

PROJECT: GAME TIPS & TRICKS

Written By. "CHECKPOINT"

'CHECKPOINT' GROUP MEMBERS:

Keisha Geyrozaga (GEY23581805)
Anderson Ricardo Gomes Ballestroz (GOM21551647)
Angelo Bongon (BON22529894)
Mohammad Rohan (ROH22609719)

Table of Contents

PDF SECTION TASK ALLOCATION:

(1) User Stories: ANGELO (completed 15/02/2025)

(2) Use Case/ Activity / Sequence UML Diagrams: ROHAN (completed 16/01/2025)

(3) Entity-Relationship / Class UML Diagrams: ANGELO (completed 18/02/2025)

(4) Wireframes / Colour Schemes: KEISHA (completed 19/02/2025)

(5) GitHub Link & Kanban Board Link: KEISHA (completed 17/02/2025)

(6) Kanban Board Screenshots + Meeting Records: ANDERSON (completed 20/02/2025)

System Design & User Experience (UX)

This section outlines the foundational structure and user interactions of our application. It includes user stories, diagrams mapping system behaviour, and design elements that define how we expect users to interact with our application.

User Stories

These outline the key interactions from a user's perspective, describing how different types of users engage with the application. They serve as a foundation for development, ensuring features align with real user needs. We have based our user stories off the core features we aim to implement in our system application.

1. **Searchable Database of User-Submitted Game Tips**

   As a gamer, I want to search for game tips by title, genre, or keyword, so that I can quickly find the relevant strategies to be able improve my skills on the game.

2. **Community-Driven Rating and Feedback System**

   As a community member, I want to rate and leave feedback on user-submitted tips, so that I can help highlight the most useful/relevant strategies for other new (& experienced) players.

3. **User Authentication and Profiles**

   As a user, I want to create and manage my own personal profile, so that I can track my progress, badges, and achievements on the platform.

4. **Gamification Elements (Badges, Leaderboards, Points)**

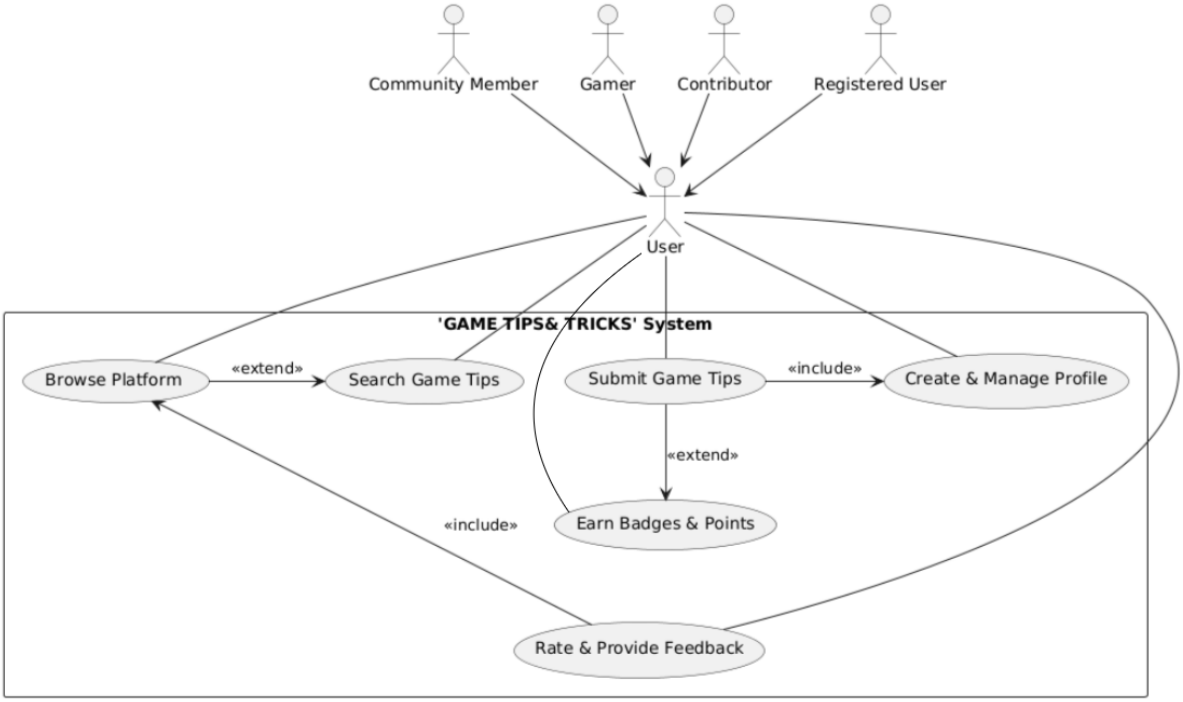
   As a contributor, I want to earn points and badges for submitting tips and engaging with the community, so that I feel motivated to participate actively without feeling bored.

5. **Visually Appealing and User-Friendly Interface**

   As a user, I want to browse the platform easily across different devices with a clean and intuitive interface, so that I can seamlessly access game tips anytime, anywhere.

Use Case Diagram

This diagram provides us with a visual representation of system interactions, this diagram relies on user stories to highlight the relationships between users and the platform's core functionalities, thus, providing us with a clear overview of expected user behaviours (and actions).



This use case diagram illustrates how users interact with the core functionalities of our "GAME TIPS & TRICKS" application via PlantUML. Four distinct user types (Community Member, Gamer, Contributor, and Registered User) are generalised under a single 'User' actor category, as they share common interactions with the system. Use cases are encapsulated within the "GAME TIPS & TRICKS" subsystem boundary.

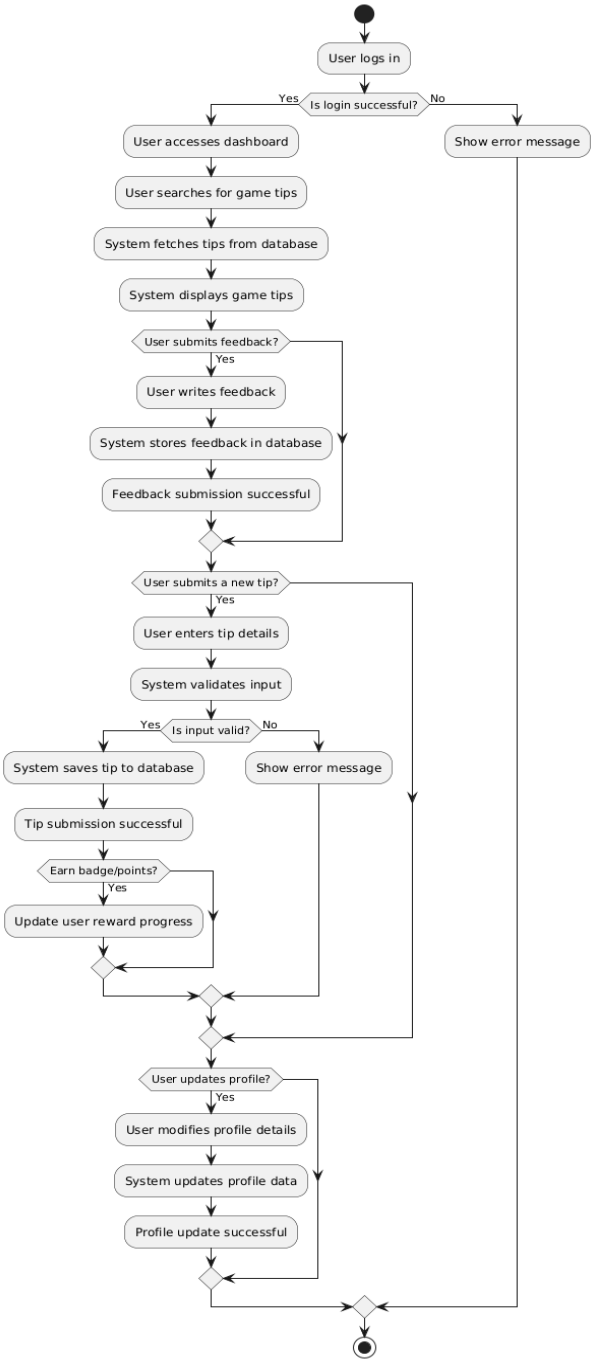
To further refine the relationships between use cases, we have incorporated <<include>> and <<extend>> associations to reflect system dependencies and optional functionalities:

- ◊ <<include>> implies mandatory interactions between use cases, such as 'Submitting Game Tips' requiring users to 'Create & Manage Profile', as contribution features are exclusive to registered users.
- ◊ <<extend>> relationships imply optional or conditional interactions, such as 'Earning Badges & Points' extending from' Submitting Game Tips', since users may engage with contributions without necessarily participating in the reward system.

These relationships provide a clearer understanding of how different functionalities intereact with one another to streamline system design, so we can clearly define user-system interactions and refine our implementation approach to ensure core functionalities align with project requirements.

Activity Diagram

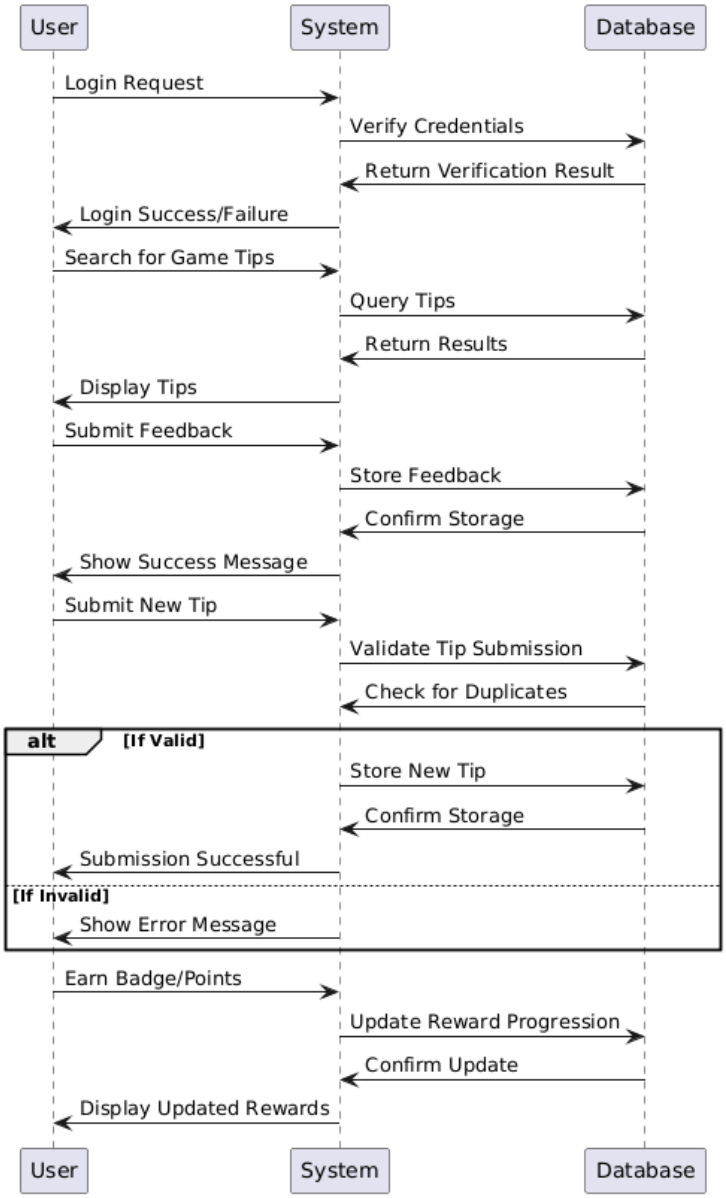
Expanding on the use case diagram, the activity diagram aims to map out the step-by-step workflow of various processes as it illustrates user actions, decision points, and system responses throughout different tasks. It essentially provides a visual breakdown of the application from start to finish.



Using PlantUML, we mapped out user interactions and illustrated the step-by-step process of a user interacting with the 'GAME TIPS & TRICKS' platform start to finish on thsi activity diagram. It highlights key processes such as searching for tips, submitting feedback, contributing new tips, and managing user profiles, while incorporating decision points that determine system responses. Invalid inputs, unsuccessful logins, and other edge cases are accounted for, ensuring a structured and intuitive user experience. With the addition of a reward progression system, users can now earn badges and points for their contributions, reinforcing engagement and incentivising participation.

Sequence Diagram

Building upon the use case and activity diagrams, the sequence diagram provides a detailed, time-ordered representation of interactions between system components. It illustrates how different parts of the application communicate to execute specific functions, ensuring a clear understanding of data flow and system behaviour.

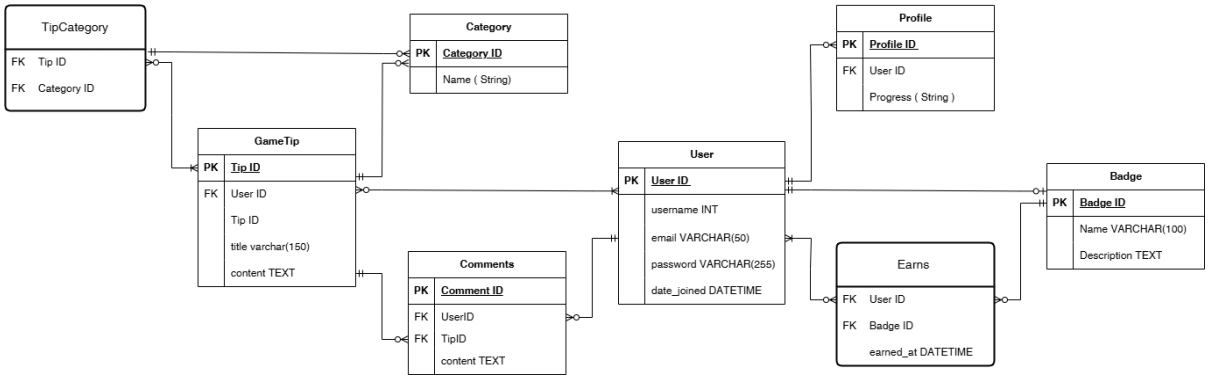


Visualising system interactions, this diagram aims to capture hwo the user, system and database systems interact with one another within the 'GAME TIPS & TRICKS' platform. It demonstrates how user requests, e.g., searching game tips, can trigger sequential system processes, leading to DB queries and responses. Each interaction follows a clear request-response pattern, ensuring data integrity and user feedback. The alt block shows what happens when a user submits a game tip.

This visualisation is particularly useful in the context of our project, as it aligns with the scaffolding code set up in Visual Studio Code (VSC) and the backend architecture deployed using Docker. By mapping out these interactions, the diagram helps ensure a structured development process, guiding how frontend and backend components integrate within our full-stack application.

Entity-Relationship Diagram

Mapping out the data structure of our application, the ERD illustrates how different entities, such as users, posts, and interactions, are related. This visual representation ensures efficient database design, enforcing relationships and constraints to maintain data integrity while supporting the system's core functionalities. Though, this is an optional section for the PDF requirements, it would be beneficial to include it for when we start database development and relying on SQL to store data.



At the core of this ERD is the '**User**' entity, which represents registered users. Each user has a '*User_ID*' as a primary key (PK) and attributes such as *username*, *email*, *password*, and *date_joined*, which store essential information. The '**User**' entity has a one-to-one (1:1) relationship with Profile, meaning each user has exactly (and only) one profile. The '**Profile**' entity includes '*Profile_ID*' as its PK and additional attributes, like *progress*, to track the user's engagement in the system.

A single '**User**' entity can create multiple '**GameTip**', establishing a one-to-many (1:M) relationship between '**User**' and '**GameTip**'. The '**GameTip**' entity represents tips posted by users, containing '*Tip_ID*' as its PK, *title*, and *content*. Since a single '**GameTip'** entity can belong to multiple '**Category'** types, a many-to-many (M:M) relationship can be established through the '**TipCategory**' relational table, which holds foreign keys ('*Tip_ID*' and '*Category_ID*'). The '**Category**' entity contains a '*Category_ID*' as its PK and a *name* attribute that should categorise different types of game tips based on the game genre, as well as playing styles (e.g., casual, competitive, etc.)

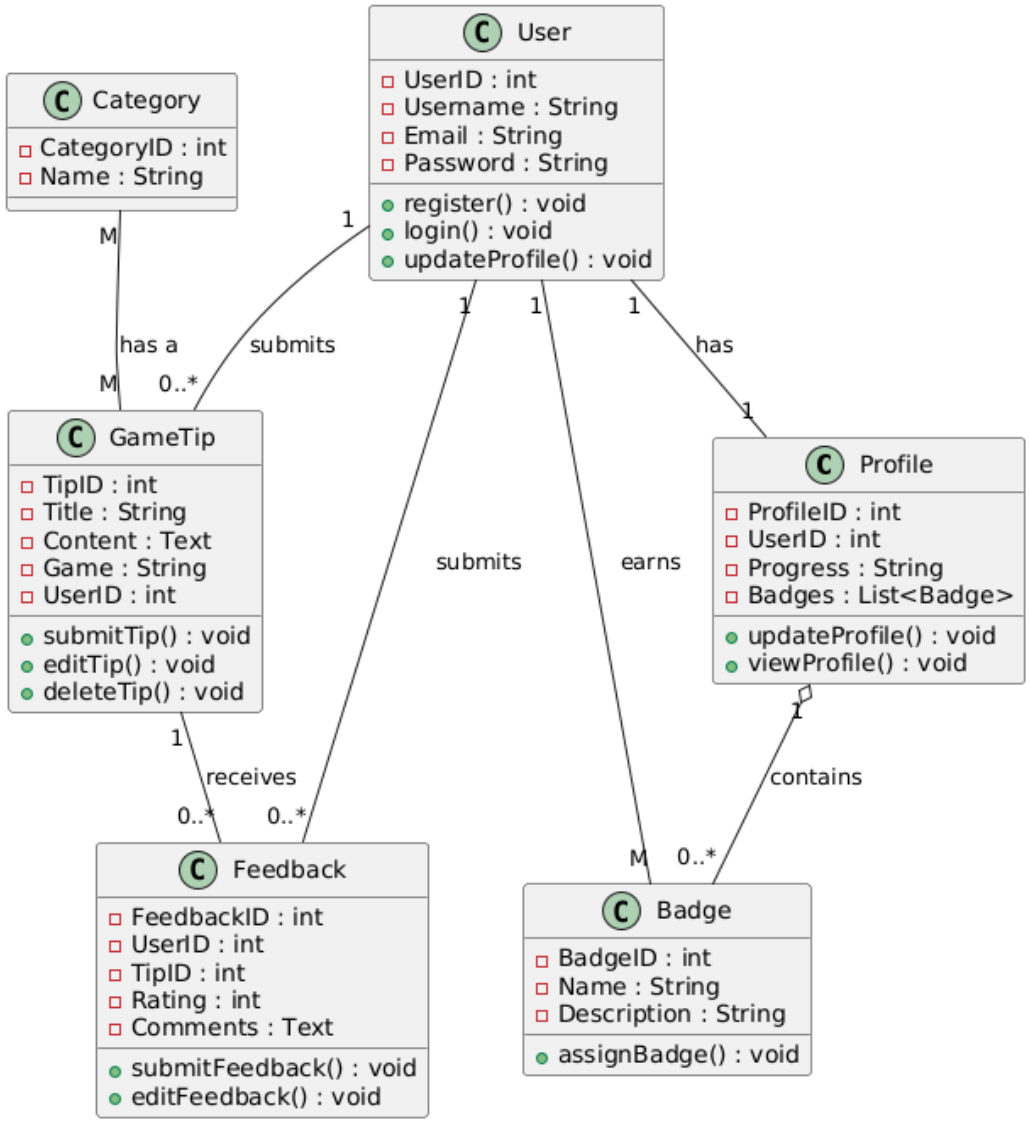
However, user engagement extends beyond posting tips as users can also provide '**Comment**' on tips posted by others. The '**Comment**' entity contains '*Comment_ID*' as its PK with attributes *rating* and *comments*. A '**User**' entity is able to leave multiple comments to create a one-to-many (1:M) relationship between '**User**' and '**Comment**', while each '**GameTip'** can receive comments, leading to another one-to-many (1:M) relationship with the '**GameTip**' entity.

Additionally, users can earn '**Badge**' as achievements within the system. The '**Badge**' entity includes 'Badge_ID' as its PK and attributes such as name and description to define different types of badges. Since users can earn multiple badges and each badge can be awarded to multiple users, a many-to-many (M:M) relationship exists between '**User**' and Badge, facilitated by the '**Earns**' relational table, which records when a user earns a badge (*earned_at*).

Unlike the '**User**', '**GameTip**', and '**Comment**' entities, which represent core functionalities of the platform, '**TipCategory**' and '**Earns**' exist purely to manage many-to-many (M:M) relationships and do not contain additional attributes beyond foreign keys. This can be seen in the class diagram as relational tables lack the methods that regular classes possess.

Class Diagram

By defining the system's architecture, the class diagram outlines key objects, attributes, and their interactions within the application. This UML model provides a structured view of how different components communicate, guiding the backend implementation and ensuring scalability.



The Class Diagram provides an object-oriented view of the system, focusing on the attributes and behaviours of each entity rather than just their relationships. Unlike the ERD, which includes relational tables to handle many-to-many (M:M) associations, the class diagram simplifies this by using direct associations between '**GameTip**' and '**Category**' as well as '**User**' and '**Badge**'.

Additionally, methods are defined for each class, outlining the system's functionality, such as user authentication, tip submission, and feedback management. The diagram also incorporates multiplicity, associations, and encapsulation (private/public/protected attributes & methods) to reflect how we believe objects will interact and function within the project's application system.

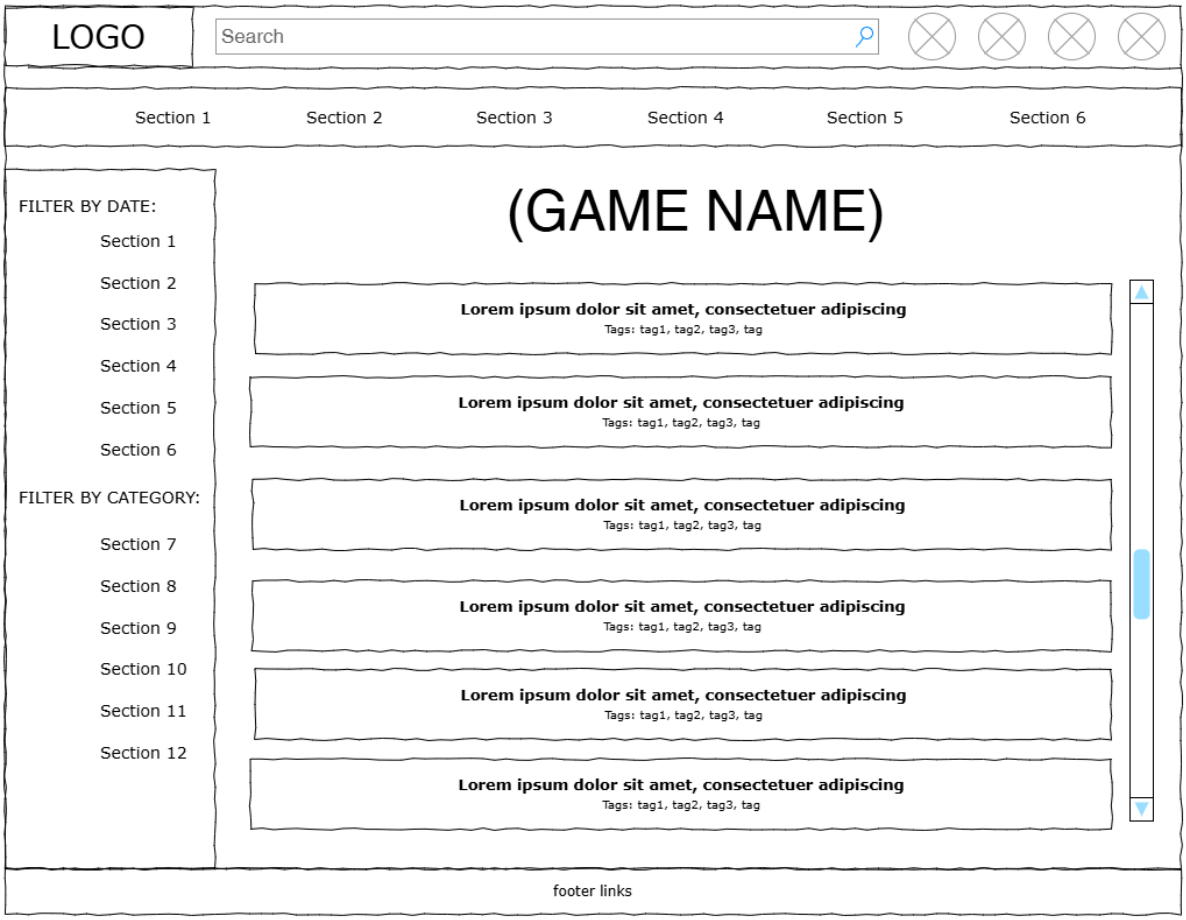
Overall, the ERD and Class Diagram work together to define the system's structure, relationships, and functionalities. Both of these UMLs help provide us with a helpful, comprehensive blueprint of classes to prepare us for the implementation stage of our project as we will soon be relying on JavaScript, HTML, and CSS as our main programming languages to push out functional webpages.

User Interface (UI) & Visual Design

This section focuses on the structural and aesthetic elements of the application, including low-fidelity webpage wireframes and chosen colour scheme. As a team, we undersyand that a well-defined visual design is essential for ensuring usability, accessibility, and a seamless user experience (UX) for our userbase. Carefully curating wireframes that meet project objectives and fulfil planned goals will help us to create an intuitive and engaging interface that enhances user interaction (UI).

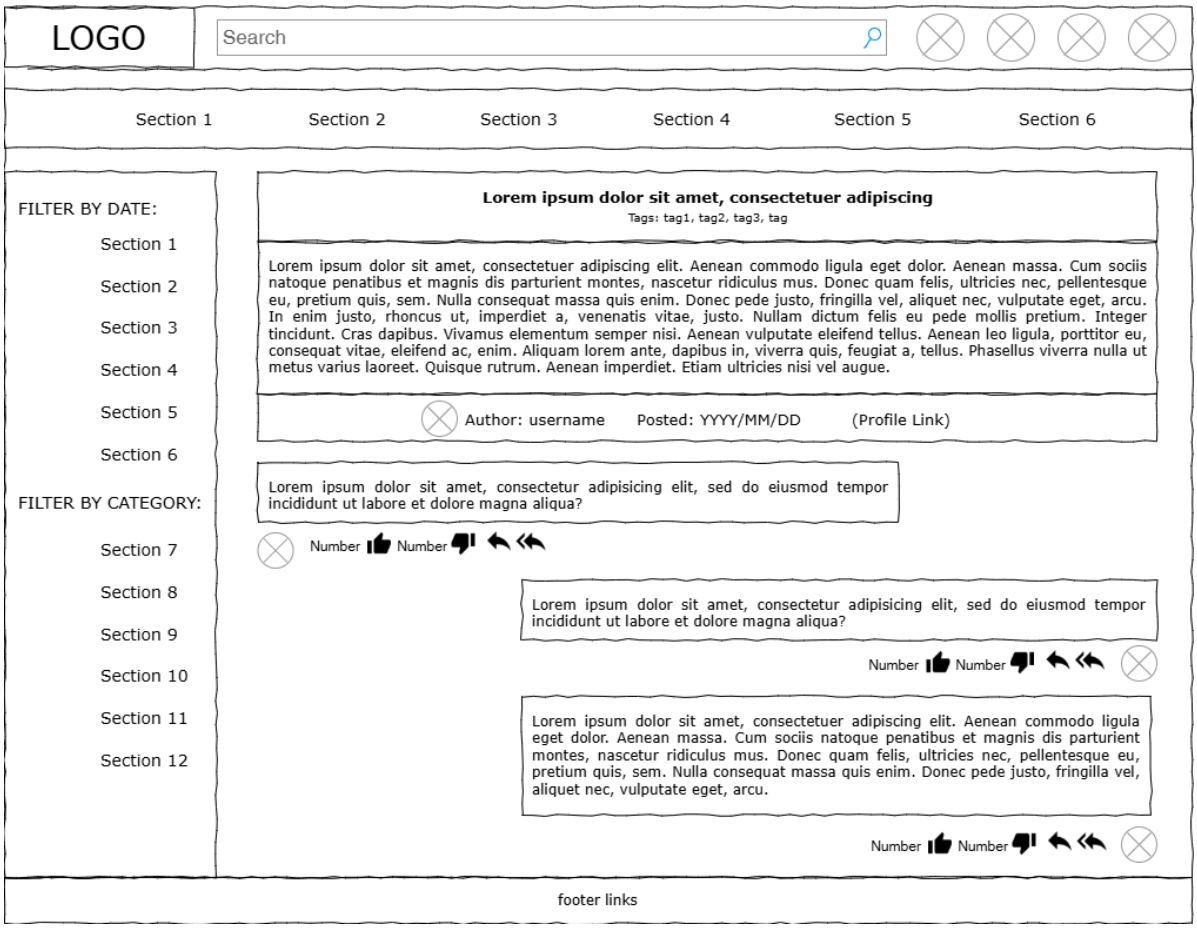
Wireframes

Before moving into detailed UI design, an initial layout is planned to map out the structure and navigation of the application. Wireframes serve as a visual blueprint, ensuring that key interface elements are logically positioned to create a smooth and intuitive user experience. To ensure that all group project requirements and core functionalities are met, five webpages have been modelled in the desired low-fidelity wireframe format to demonstrate what we want our website to look like.

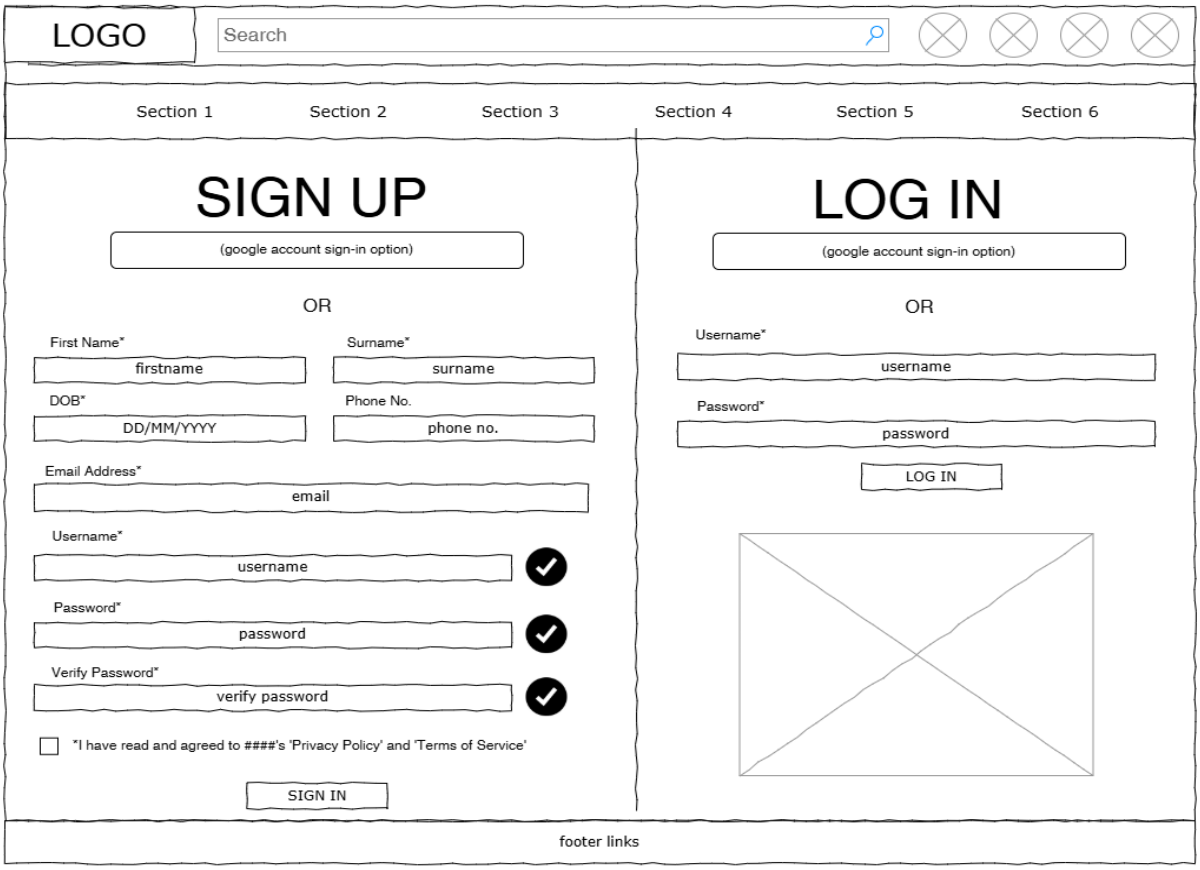


Homepage – Displays game-related content, filters by date/category, and shows a list of game tips

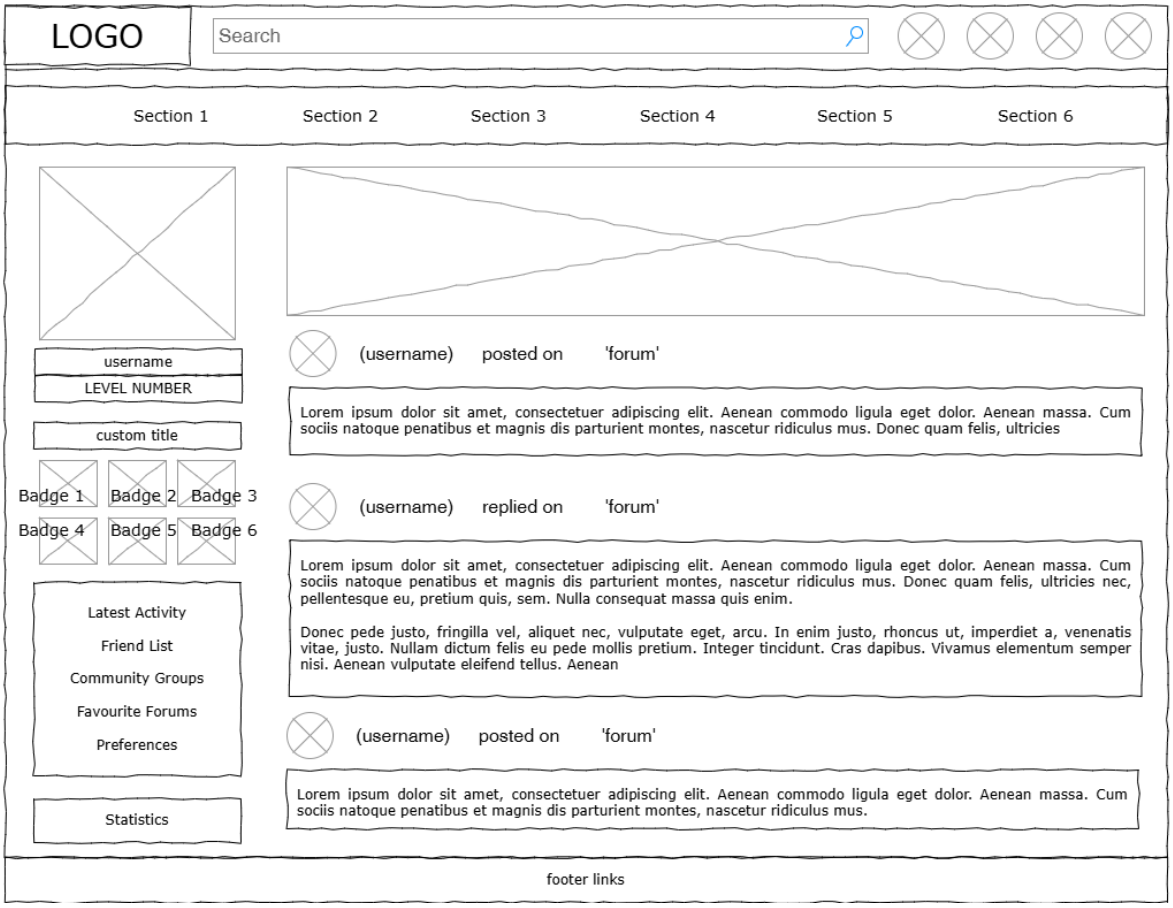
Additional Header Details – Provides context to the planned buttons on the header bar



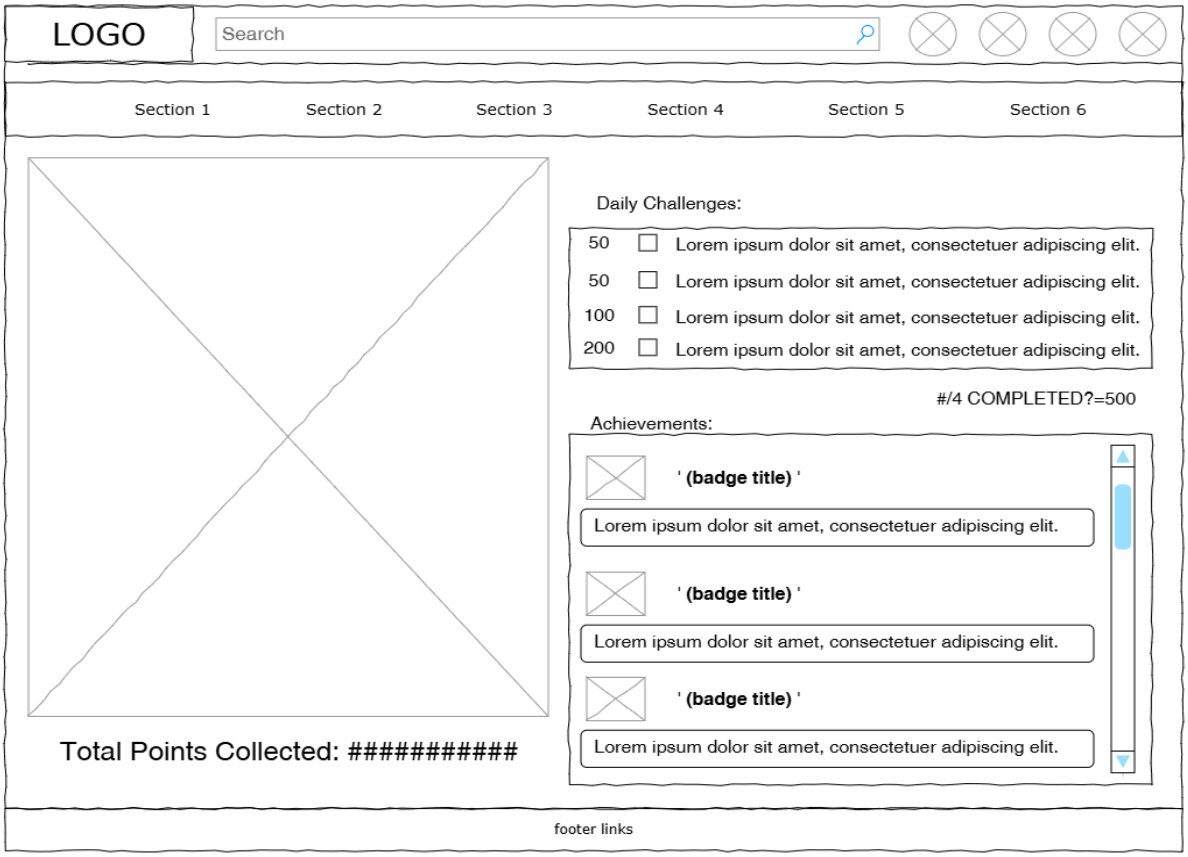
Forum Page – Displays user discussions, filters for browsing topics, and adds interactive elements



Signup & Login Page – Allows users to register/log in with their username and password



User Profile Page – Shows user details, badges, recent activity & community interactions



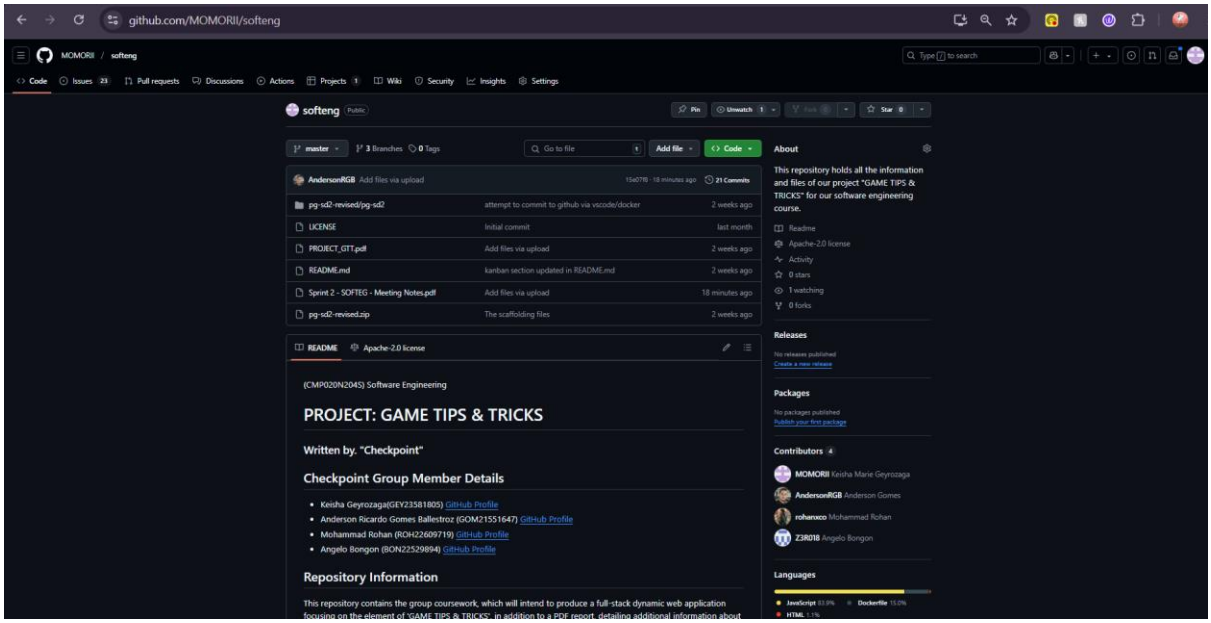
User Statistics Page – Shows a user's achievements, daily challenges, and total points collected

These wireframes clearly demonstrate core functionalities and user interaction flow as required.

Project Management & Documentation

This section outlines the tools and methodologies used to manage the project efficiently, track progress, and maintain clear documentation—ensuring smooth collaboration and development.

GitHub Repository Preview & Link



As part of the project, we were tasked with creating and setting up a repository on GitHub dedicated to collaborative development and version control, in addition to containing the scaffolding files critical to building our web application. Since last accessing the repository, we have added additional documents that expand upon our project and provide further details on its development, including refined user stories, updated design artifacts, and documentation of key implementation processes.

The link below will take you to the repository, named ‘softeng’, which contains a customised README file, the licensing (Apache 2.0), and lastly, the scaffolding files (both as a ZIP and a folder of the files uploaded from ‘Docker’ via ‘Visual Studio Code’. All four of our group members have access and regularly contribute to the GitHub repository to ensure changes have been adequately made.

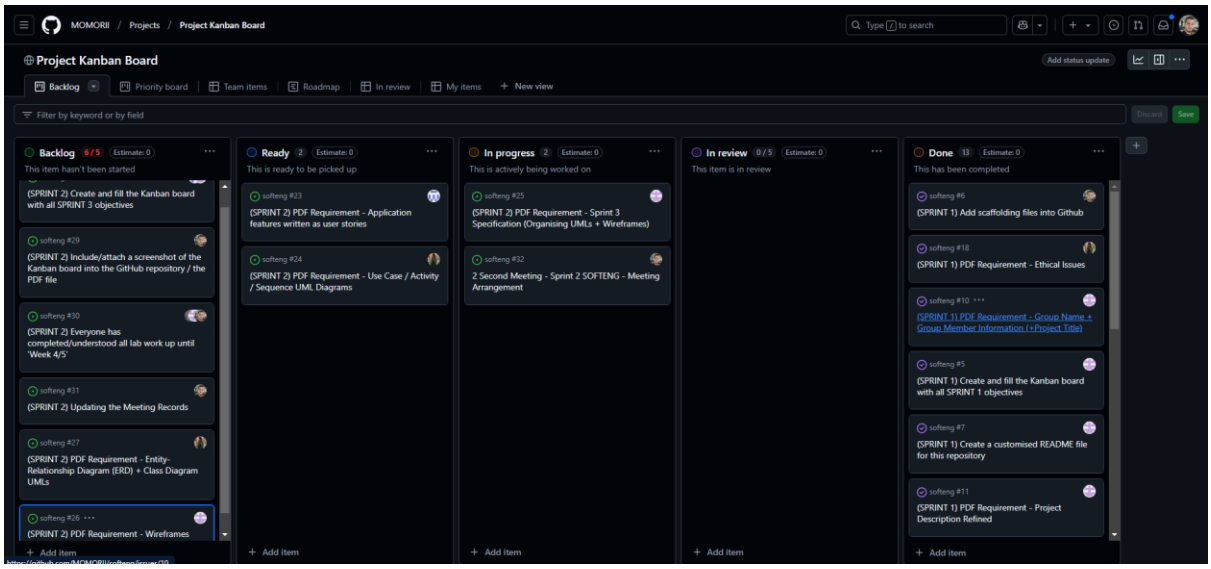
LINK: https://github.com/MOMORII/softeng

Commits on the project were mostly completed by the product owner, Keisha, who was tasked with editing the README file and ensuring that all members were able to access group resources shared on the repository, including the project kanban board.

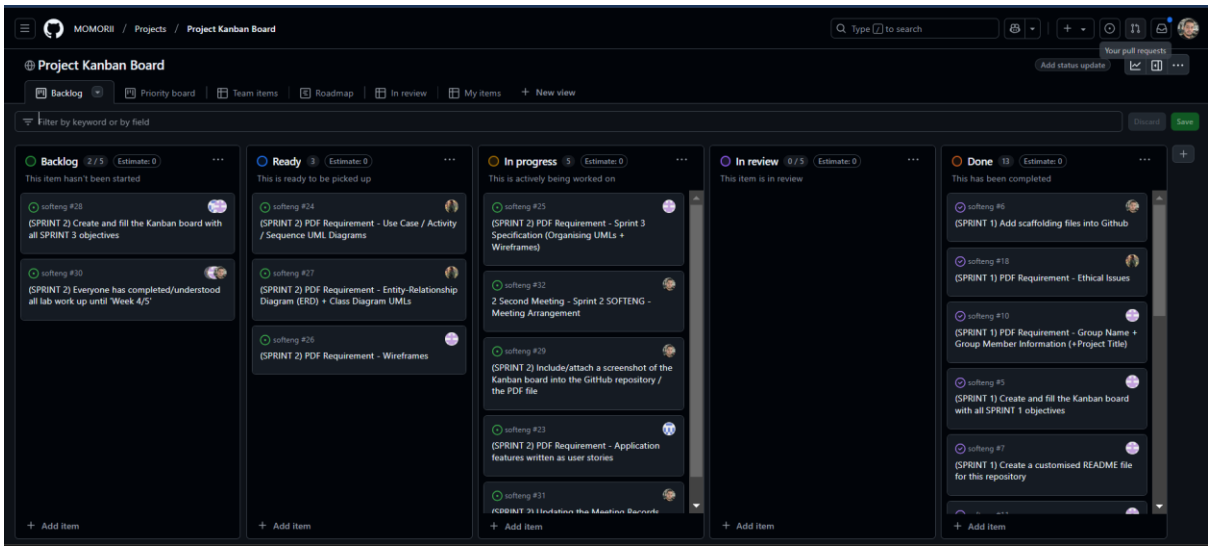
Kanban Board

As per the requirements for ‘SPRINT 2’, we were tasked with creating checklist tasks for ‘SPRINT 3’, so we’d be fully equipped to kickstart the development process and understand our individual responsibilities (allocated/assigned tasks), in addition to the responsibilities we must uphold as a group (maintaining communications via quick updates on our WhatsApp group chat and/or formal discussions on Microsoft Teams).

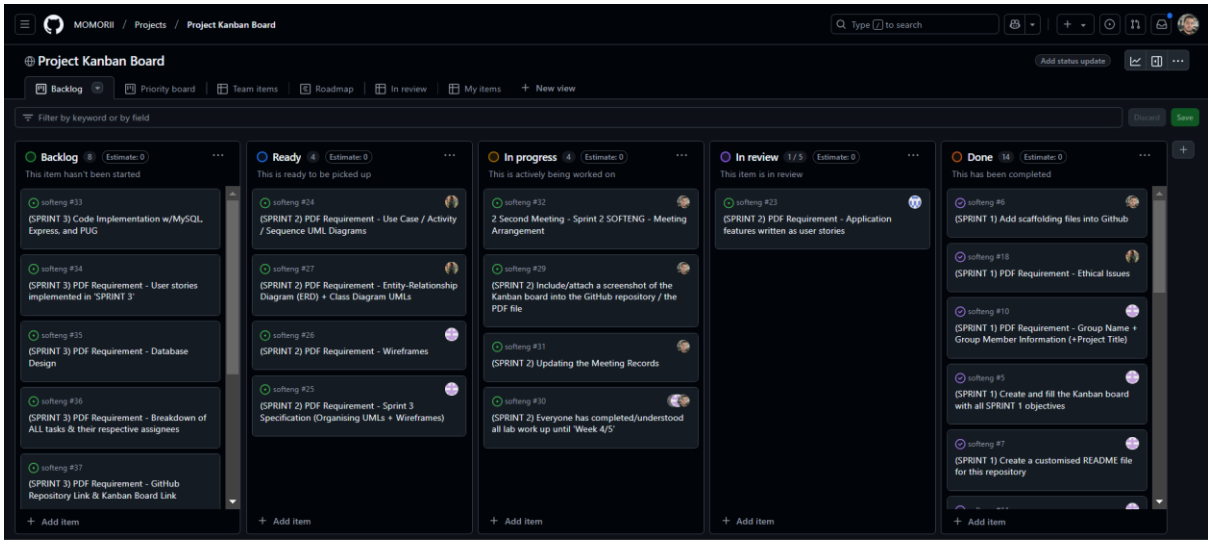
Here is the **link** to the Kanban board: https://github.com/users/MOMORII/projects/2/views/1



Screenshot 1 (Before Starting 'SPRINT 2')



Screenshot 2 (Mid-Progress for 'SPRINT 2')



Screenshot 3 (After Completing 'SPRINT 2')

Meeting Records

Sprint 2

These meeting records serve to document key details of each team meeting conducted via Teams, including member attendance, communication with absent members, date and time, meeting format (in-person or online), objectives, and discussion summaries.

First Team Meeting

Date and Time	**First Team Meeting** (13/02/2025 14:10 – 15:15)
Project Name	**GAMES TIPS & TRICKS**
Meeting Goal	• Review Sprint 1 • Plan Sprint 2
Facilitator	Anderson Ricardo Gomes Ballesteroz
Note taker	Anderson Ricardo Gomes Ballesteroz
Attendees	• **Keisha Geyrozaga** (GEY23581805) – Product Owner • **Anderson Ricardo Gomes Ballesteroz** (GOM21551647) – Scrum Master • **Mohammad Rohan** (ROH22609719) – Team Member • **Angelo Bongon** (BON22529894) – Team Member
Roundtable Updates (each group member to contribute)	• **Keisha:** Confirmed that Sprint 1 has been completed successfully. GitHub and Kanban board are up to date. No additional feedback was provided by the lecturer. • **Anderson:** Emphasized that the meeting should define clear tasks for Sprint 2. Highlighted the need for the team to coordinate work efficiently. • **Rohan:** Will focus on UML diagrams and database structures. Expressed confidence in completing ERD and Class Diagrams before the next meeting. • **Angelo:** Will structure application features as user stories and assist Keisha in updating the Kanban board.

Discussion points	**What was the feedback from the lecturer?** • "Everything is fine", no additional feedback was given. **What, if any, additional tasks do you need to do following the review?** • Start Sprint 2 tasks immediately to meet the deadline. **What are the features you are going to focus on for Sprint 3?** • Implementing UI wireframes. • Structuring the UML diagrams for backend integration. • Setting up database architecture based on ERD. • Adding user authentication and gamification elements. **Who is going to start creating the artifacts for these?** • **Wireframes:** Keisha • **UML Diagrams (Use Case, Activity, Sequence):** Rohan • **ERD & Class Diagram:** Rohan • **User Stories:** Angelo • **Kanban Board for Sprint 3:** Keisha & Angelo • **Meeting Records & GitHub Update:** Anderson **How and when will the group review and approve the artifacts?** • Artifacts will be reviewed in the second meeting (19/02/2025). • Each team member should upload their work to GitHub and notify the group for feedback before the next meeting.
Actions (list tasks and assign a group member)	• **Write user stories in PDF** – Angelo • **Create UML Diagrams (Use Case, Activity, Sequence)** – Rohan • **Design Wireframes** – Keisha • **Create ERD & Class Diagram** – Rohan • **Update Kanban Board with Sprint 3 tasks** - Angelo & Keisha • **Update Meeting Records** – Anderson • **Upload Screenshot of Kanban Board to GitHub** – Anderson

Meeting Minutes

Second team meeting

Date and Time	**Second Team Meeting** (19/02/2025 13:50 – 14:55)
Project Name	**GAMES TIPS & TRICKS**
Meeting Goal	• Review Sprint 2 progress • Finalize deliverables and ensure all tasks are complete
Facilitator	Anderson Ricardo Gomes Ballesteroz
Note taker	Anderson Ricardo Gomes Ballesteroz

Attendees	• **Keisha Geyrozaga** (GEY23581805) – Product Owner • **Anderson Ricardo Gomes Ballesteroz** (GOM21551647) – Scrum Master • **Mohammad Rohan** (ROH22609719) – Team Member • **Angelo Bongon** (BON22529894) – Team Member
Roundtable Updates (each group member to contribute)	• **Keisha:** Completed wireframes, uploaded them to GitHub, and integrated them into the project PDF. Ready for feedback. • **Anderson:** Updated meeting records and added the Kanban screenshot to GitHub and PDF. • **Rohan:** Created UML diagrams and ERD/Class Diagrams. Requested a review before final submission. • **Angelo:** Completed user stories and structured them in the PDF. Updated the Kanban board with Sprint 3 tasks.
Discussion points	**What was the feedback from the team on the created artifacts?** • **Wireframes:** Approved. Minor UI changes will be addressed in Sprint 3. • **UML Diagrams & ERD:** Some labels need minor adjustments before final upload. • **User Stories:** Well structured, no changes needed**.** **Finalizing Sprint 2:** • All artifacts must be uploaded to GitHub and the final PDF version. • Final quality check and submission.
Actions (list tasks and assign a group member)	• **Minor fixes on UML Diagrams & ERD** – Rohan • **Upload final PDF version with all artifacts** – Keisha • **Final review of project documentation** - Entire Team • **Submit Sprint 2 deliverables** - Keisha & Anderson • **Ensure all meeting records are properly formatted and included in the PDF** – Anderson • **Cross-check Kanban board tasks with the meeting records for consistency** – Anderson

Conclusion

Through careful planning, collaboration, and the use of industry-standard tools, we have successfully structured the development of our 'GAME TIPS & TRICKS' application. By implementing a clear project management approach, designing an intuitive user experience (UX), and establishing a robust system architecture, we have laid the foundation for a functional and engaging web application.

Moving forward, we will continue refining our features, addressing feedback, and ensuring the final product meets both technical and user-focused requirements for this project.