CEBU INSTITUTE OF TECHNOLOGY UNIVERSITY

COLLEGE OF COMPUTER STUDIES

Software Requirements Specifications

for

CyberKids

Change History

Version	Date	Amendment	Author
1.0	3/14/2025	Initial	Cultura
1.1	3/18/2025	Updated Modules	Baguio Cultura Dedumo Vequiso
1.2	3/21/2025	Updated Modules (relevant to the problem)	Baguio Cultura Dedumo Vequiso
2.0	5/18/2025	Finalized for Capstone 1	Cultura
3.0	8/27/2025	Applying the changes, standards, and conventions used in the refactored requirements.	Baguio Cultura Dedumo

Table of Contents

1. Introduction

1.1. Purpose

CyberKids is an interactive puzzle and strategy game designed to immerse elementary school students in real-world cybersecurity scenarios. It aims to address the increasing vulnerability of children to online threats by providing an engaging and effective learning environment focused on password security, phishing awareness, and online privacy. The intended audience includes grades 5-6 elementary students, computer teachers, system administrators, and developers responsible for maintaining and improving the platform.

1.2. Scope

CyberKids is an educational game designed to teach students about cybersecurity best practices through a series of engaging missions within the Roblox platform. By leveraging Roblox's immersive environment, CyberKids transforms essential cybersecurity conc epts into interactive, game-based challenges that promote active learning and critical thinking.

What the Software Will Do (Core Functionalities**)**

1. **Multiplayer Exploration**: Players control a character navigating a virtual world with different digital zones representing online platforms in each level.
2. **Mission-Based Learning:** Players complete three major cybersecurity challenges:
    - Data Leak Investigation – Identifying safe and unsafe personal information to share.
    - Password Fortress Defense – Constructing strong passwords to protect against hacking attempts.
    - Cyber Escape Room – Identifying phishing and scam attempts to escape a locked digital room.
3. **Teacher Dashboard**:
    - Educators can track student progress and view mission completion rates.
    - Teachers can assign specific missions and review student performance.

1.3. Definitions, Acronyms and Abbreviations

This section provides definitions of key terms, acronyms, and abbreviations relevant to the implementation of the CyberKids system.

Acronyms:

Acronyms used in this document shall be interpreted as follows:

- API – Application Programming Interface
- CORS – Cross-Origin Resource Sharing
- CRUD – Create, Read, Update, Delete
- CI/CD – Continuous Integration / Continuous Deployment
- JWT – JSON Web Token
- SQL – Non-Relational Database Query Language
- REST API – Representational State Transfer API
- UI – User Interface

Abbreviations:

Abbreviations used in this document shall be interpreted as follows:

- JS – JavaScript
- DB – Database
- UX – User Experience
- HTTP – HyperText Transfer Protocol
- SQL – Structured Query Language
- URL – Uniform Resource Locator

Definitions:

Definitions used in this document shall be interpreted as follows:

Term	Definition
Authentication	The process of officially recognizing an educational institution or program as meeting specific quality standards.
Authorization	The process of granting or denying access to specific resources or functionalities.
Backend	The server-side part of a software application, responsible for data processing, logic, and database interactions
Cybersecurity	The practice of protecting systems, networks, and programs from digital attacks.
Database	An organized collection of structured information, or data, typically stored electronically.
Encryption	The process of converting data into a secure format to prevent unauthorized access.
Front-end	The user-facing part of a web application that handles the visual presentation and interaction.
Git	A distributed version control system for tracking changes in source code.
GitHub	A web-based platform for version control and collaboration using Git
Malware	Malicious software designed to harm, exploit, or disrupt computer systems.

React.js	A JavaScript library for building user interfaces.
Scams	Fraudulent schemes used to deceive people into providing money, personal details, or access to sensitive data.
Spring Boot	A Java-based framework for building microservices and web applications.
Vite	A build tool for modern web development.
Wireframe	A visual guide that represents the skeletal framework of a website or application.

1.4. References

[1] IEEE Computer Society. (1998). IEEE 830-1998: Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998. Source: IEEE Xplore Digital Library.

[2] React. (2019). Glossary. Source: https://legacy.reactjs.org/docs/glossary.html

[3] Axios Contributors. (2024). Axios API Reference. Source: https://axios-http.com

[4] National Institute of Standards and Technology (NIST). (2018). Cybersecurity Framework. Source: https://www.nist.gov/cyberframework

2. Overall Description

2.1. Product perspective

These modules ensure a seamless experience for students while providing teachers with tools to monitor progress.

Modular Decomposition

Module 1: Gamified Cybersecurity Game Levels

This module delivers interactive, educational gameplay designed to teach students key cybersecurity concepts through immersive challenges.

- Gamified Online Privacy and Safety
  Players analyze digital case files and sort information into “Safe to Share” vs. “Not Safe to Share” categories, reinforcing critical thinking about data privacy.

- Gamified Password Security
  Players collect password fragments from treasure chests and assemble strong passwords. A simulated hacker bot attempts to crack weak passwords, encouraging players to build robust defenses.

- Gamified Phishing and Scam Awareness
  Players navigate a virtual escape room, identifying phishing emails, fake login pages, and scam messages using social engineering tactics. Successful identification unlocks security codes to escape the room.

Module 2: Student Real-Name Integration

Enables students to input their real names within the Roblox game, linking their in-game activities to identifiable profiles accessible by teachers for clearer monitoring and personalized support.

- Allows students to register and update their real names directly inside the game interface
- Replaces anonymous Roblox usernames with real names in the teacher dashboard for better student identification
- Ensures data consistency between gameplay activity and student records
- Supports privacy and data security by restricting access to real names to authorized teachers only

Module 3: Teacher Web Dashboard

Equips teachers with comprehensive tools to monitor student engagement and performance, facilitating data-driven instruction and support.

- Teachers can lock access to specific game modules
- If a student needs remediation or challenge, teachers can reassign them to replay or skip particular levels.
- View detailed student progress reports and analytics
- Teachers can create class sections by assigning a grade and section name.

Module 4: Notification System

Enhances teacher responsiveness by providing real-time alerts and updates on student activities and system events through a notification interface.

- Automatically generate notifications based on key student actions such as challenge completion.
- Display unread and read notifications for easy tracking of important events

2.2. User characteristics

CyberKids is designed primarily for elementary and junior high school students, introducing them to cybersecurity concepts in a fun, interactive way. The platform helps students develop safe online habits through gamified learning experiences, while teachers and parents can track their progress and guide them toward better digital awareness

.

1. Student Users

The primary users of CyberKids are young learners who are becoming more engaged with technology and the internet. The platform is tailored to their learning style through an intuitive, interactive, and engaging game-based format.

- **Need for Early Cybersecurity Awareness**

  Many students lack proper guidance on how to stay safe online, making them vulnerable to phishing, scams, and cyberbullying. CyberKids fills this gap by introducing essential cybersecurity knowledge through engaging missions and challenges that teach digital safety without overwhelming young users with technical jargon.

- **Interactive and Gamified Learning**

  Traditional learning materials may not hold the attention of young students. CyberKids ensures engagement by integrating reward systems, level progression, and real-time feedback, making cybersecurity education fun and rewarding. Students can unlock new levels and earn badges as they progress through different cybersecurity challenges.

- **Age-Appropriate Content**

  Unlike general cybersecurity platforms designed for adults, CyberKids presents lessons in a way that is simple, visual, and interactive, making it easier for young learners to grasp concepts like password security, phishing awareness, and safe social media usage.

2. Teachers and Educators

Teachers play a crucial role in guiding students' learning experience and ensuring they understand the importance of cybersecurity. CyberKids supports teachers by offering a dashboard that tracks student progress and performance.

- **Classroom Integration**

  Teachers can integrate CyberKids into their curriculum as an interactive learning tool to supplement traditional cybersecurity education. The platform provides structured challenges that align with key digital literacy learning goals.

- **Student Progress Monitoring**

  Educators can view student scores, track progress across different cybersecurity topics, and identify areas where students need improvement. This data-driven approach helps teachers offer targeted guidance to students struggling with certain cybersecurity concepts.

- **Facilitating Cyber Safety Discussions**

  Teachers can use CyberKids as a starting point for classroom discussions about online safety, responsible internet use, and digital ethics, ensuring students apply what they learn in real-life scenarios.

3. Parents and Guardians

Parents are often concerned about their children's online activities and exposure to digital threats. CyberKids provides them with a way to ensure their children are learning safe online behaviors through an engaging and structured platform.

- **Parental Supervision and Insights**

  Parents can monitor their child's progress through reports on completed missions, scores, and areas needing improvement. This enables parents to have meaningful conversations about cybersecurity at home.

- **Encouraging Safe Internet Habits at Home**

  With CyberKids, parents can actively participate in reinforcing safe online behaviors, helping their children apply cybersecurity lessons beyond the digital classroom.

User Roles and Privileges

1. Students (Players)

- **Description:** Students in Grades 5-6 at CIT-University who participate in the game to learn about cybersecurity concepts.
- **Roles & Privileges:**
    - Access and play all game missions.
    - View their own scores and rankings on the leaderboard.
    - Customize their profile (avatar and display name only).

2. Teachers

- **Description:** Educators who monitor and assess student progress in CyberKids.
- **Roles & Privileges:**
    - Access the Teacher Dashboard to track students' progress, scores, and mission completion rates.
    - Generate performance reports to assess learning outcomes.
    - View leaderboard standings for class rankings.

3. Admin

- **Description:** System administrators responsible for managing and maintaining the CyberKids application.
- **Roles & Privileges:**
    - Maintain system functionality and ensure a smooth gaming experience.
    - Manage technical updates, bug fixes, and security measures.
    - Oversee user management, including teacher and student accounts.
    - Ensure leaderboard accuracy and resolve any scoring discrepancies.

2.4. Constraints

The development of CyberKids is subject to various technical, regulatory, and operational constraints that can impact the system design, operation, and deployment. Such constraints need to be critically analyzed to ensure the system is pragmatic, effective, and compliant with the relevant standards.

Regulatory Policies

CyberKids must comply with data protection law and education code in a bid to protect user information and enable the consistency of career guidance services. The system should be compliant with:

- The Data Privacy Act of 2012 (RA 10173) intends to protect students' information, grades, and career options by guaranteeing confidentiality and security.

Hardware Limitations

The system is designed to run on desktop and laptop computers that meet the minimum specifications required by Roblox, the platform hosting the game. To ensure a smooth and enjoyable gaming experience, users should have devices with decent hardware capabilities as outlined below:

- **Minimum System Requirements:**
  The game requires a computer with at least Windows 7, macOS 10.11, or equivalent operating system. A processor with a minimum of 1.6 GHz, 1 GB of RAM, and a dedicated graphics card supporting DirectX 9 or OpenGL 2.1 is necessary to run the game without lag.

- **Recommended System Specifications:**
  For optimal performance, especially with multiple game modules and smooth rendering, devices with a dual-core processor (2.5 GHz or higher), 4 GB of RAM or more, and a modern GPU supporting DirectX 10 or higher are recommended.

- **Resource Usage Optimization:**
  The system is optimized to balance visual quality and performance, aiming to prevent excessive CPU and memory consumption. This ensures that even mid-range hardware can provide a seamless gameplay experience.

- **Justification:**
  These hardware requirements align with Roblox's official recommendations and ensure that students at CIT-University can access and enjoy the cybersecurity game modules without performance bottlenecks or crashes. Ensuring minimum specs helps avoid frustration and supports uninterrupted learning.

Interfaces to Other Applications

The system operates independently and does not integrate with external applications, but this comes with certain constraints:

- No Social Media Integration – Unlike other modern educational tools, the system does not connect with social media platforms for sharing progress or achievements.
- Internal Data Storage Only – All user data, including scores, progress, and mission completion information, is stored within an internal database. This limits external access but ensures security and control over student information.

Parallel Operation

While the game is designed as a multiplayer experience, multiple students can play simultaneously. However, there are some operational constraints:

- Independent Sessions – Each student's game session runs independently, meaning their progress does not interfere with other users.
- Scalability Considerations – Although the system allows multiple concurrent users, excessive simultaneous logins may require server performance optimization.

Audit Functions

The system includes mechanisms for monitoring student activity and progress, though there are some limitations:

- Teacher Dashboard Tracking – Teachers can monitor student progress through a dashboard, but they can only view the data and cannot modify student scores or mission completion records.
- Session Logs and Data Retention – The system logs game sessions, scores, and mission completion data, but long-term storage may be limited based on server capacity.

Control Functions

Different user roles have varying levels of access and control over the system, but there are restrictions in place:

- Administrative Oversight – Admin users have full control over system maintenance, bug fixes, and security updates. They ensure the system remains functional and up to date.
- Limited Teacher Modifications – While teachers can monitor student progress, they do not have the ability to alter student scores or modify gameplay mechanics.

Reliability Requirements

The system aims to be stable and reliable, though certain factors may impact its performance:

- Crash Prevention – The system must function without unexpected crashes during gameplay to ensure a smooth user experience.
- Data Security and Retrieval – Student progress and leaderboard scores must be securely stored and retrievable in case of system failures.

Criticality of the Application

Although the system is designed primarily as an educational tool, reliability remains a key priority:

- Not a Mission-Critical System – The game serves as a supplementary educational resource rather than a system that directly impacts grades or official records.
- Importance of Seamless Learning – Despite not being mission-critical, the system must remain stable to maintain student engagement and educational value.

Safety and Security Considerations

Ensuring user security and data protection is a fundamental requirement of the system:

- Collection of Minimal Personal Data – Students provide only their real names (no other PII such as birthdates or addresses). These names are used solely for teacher identification and are stored securely in the Railway MySQL database.
- Data Privacy Compliance – Before collecting any real-name information, the student's faculty adviser and supervising professor will sign a data privacy consent letter, outlining the purpose, storage, and retention policies for the names collected.

2.5. Assumptions and dependencies

The development and functionality of CyberKids rely on the following assumptions and dependencies. Any changes to these factors may impact the system requirements and require modifications to the software.

Assumptions

1. **Hardware Availability**
    - The game is assumed to run on standard desktop computers available at CIT-University.
    - Devices used must have basic input peripherals (keyboard and mouse).
2. **Operating System Compatibility**
    - The system assumes that devices will run on Windows (Google Chrome, Firefox, Edge). If a specific OS is not supported, additional development effort may be required.
3. **Internet Connectivity**
    - The leaderboard and teacher dashboard assume a stable internet connection for real-time updates.
4. **User Digital Literacy**
    - Students (Grades 5-6) are assumed to have basic computer skills, such as navigating a game interface, using a mouse/keyboard, and following on-screen instructions.

Dependencies

1. **Database System**
    - The leaderboard, student progress tracking, and user profiles depend on a functional database (e.g., MySQL, Firebase).
2. **Web-Based Frameworks and Technologies**
    - The game relies on web-based technologies (e.g., React, JavaScript, HTML5, CSS) for execution.
3. **Security Policies**
    - The system follows CIT-University's IT security policies to prevent unauthorized access and protect student data.
    - Changes in security regulations may require updates to authentication and access control mechanisms.
4. **Server Availability**
    - The leaderboard and teacher dashboard depend on a centralized server for data storage and retrieval.
5. **Third-Party Libraries and APIs**
    - The system may use external libraries or APIs for password encryption, data storage, and UI enhancements.
    - If these libraries become deprecated, alternative solutions must be integrated.

3. Specific Requirements

3.1. External interface requirements

3.1.1. Hardware interfaces

The system is designed to operate on standard computing hardware used at CIT-University, ensuring compatibility with a wide range of devices. The hardware interface requirements include:

- **Client-side Requirements**
  - **Devices:** Desktop and Laptop with a modern web browser.
  - **Browsers:** Chrome, Firefox, Edge, and Safari (latest stable versions).
  - **Internet Connection:** Minimum 5 Mbps for smooth interaction.

- **Supported Devices** – The software will run on desktop and laptop computers with Windows operating systems, ensuring accessibility for students. No support for mobile or tablet devices is currently planned.

- **Minimum Hardware Specifications** – The system must be optimized to function on low-end to mid-range computers, requiring at least:
  - Processor: Intel Core i3 (or equivalent) with a 2.0 GHz clock speed
  - RAM: 4 GB minimum, 8 GB recommended for optimal performance
  - Storage: At least 2 GB of available disk space for installation and data storage
  - Graphics: Integrated GPU support (no dedicated graphics card required)

- **Peripheral Device Support** – The system will support standard input and output devices, including:
  - Keyboard and Mouse: Required for user interactions in the system

3.1.2. Software interfaces

The system will interact with various software components to ensure smooth functionality, scalability, and maintainability, including operating systems, database management systems, development frameworks, and cloud deployment services.

- **Operating System Compatibility**
  The software will be compatible with the following operating systems:

  - Windows 10 and later – Officially supported and thoroughly tested for stability and security.

- **Database Management System (DBMS)**
  The system will store all user progress, scores, and logs in a relational database with cloud-based hosting for high availability and reliability. Supported databases include:

  - MySQL 8.0 or later – Primary relational database for structured data storage.
  - PostgreSQL 13 or later – Alternative database option offering enhanced scalability and robustness.
  - Railway SQL Database – Cloud-hosted database solution providing managed services, automated backups, and scalability.

- **Frameworks and Development Tools**

  The system leverages modern development frameworks and tools for efficient development, scalability, and maintainability:

  - Game Engine: Roblox – The primary game engine hosting the cybersecurity modules, deployed on Roblox servers.
  - Backend: Java Spring Boot – Handles API requests, business logic, and integrates with the database layer.
  - Frontend: React.js – For building the user interface components of student and teacher dashboards. Additional frontend frameworks include:
    - Shadcn UI – For accessible and customizable UI components.
    - Tailwind CSS – Utility-first CSS framework used to rapidly build and style responsive layouts.
  - Data Management: Hibernate ORM – Manages database interactions through object-relational mapping, ensuring data consistency and integrity.

- **APIs**

  Communication between the frontend and backend services is facilitated through RESTful APIs, enabling modularity and separation of concerns.

- **Containerization and Deployment**

  - Docker – Used to containerize the backend and database services, ensuring consistent environments across development, testing, and production.
  - Render – The backend services are deployed on Render, providing scalable and reliable cloud hosting.
  - Vercel – The frontend React application is deployed on Vercel, enabling fast global content delivery and seamless continuous deployment pipelines.

- **Cloud Infrastructure**

  The system utilizes Railway for hosting the relational database, leveraging Railway's cloud infrastructure for high availability, security, and automatic scaling.

3.1.3. Communications interfaces

The CyberKids requires connectivity to various network services to enable multiplayer interactions, leaderboard updates, and real-time event handling.

Internet Connectivity Requirements

An active internet connection is essential for:

- Live updates of game content and user progress
- Leaderboard synchronization and ranking updates
- Cloud-based data storage and retrieval
- Real-time notifications and event handling within dashboards

Protocols and Technologies Used

- **HTTP/HTTPS**
  Standard protocols for secure communication between clients (web dashboards and game clients) and backend servers, ensuring encrypted data transfer.
- **RESTful APIs**
  The backend exposes RESTful endpoints that handle CRUD operations, user authentication, game progress updates, and leaderboard data management.
- **WebSocket Protocol**
  Enables persistent, full-duplex communication channels between clients and servers, facilitating real-time features such as:
  - Instant leaderboard updates
  - Real-time notifications
  - Live event handling within the teacher and student dashboards
- **Axios**
  A promise-based HTTP client used in the frontend React applications for making asynchronous API requests to the backend, including fetching user data, submitting progress, and managing profile settings.

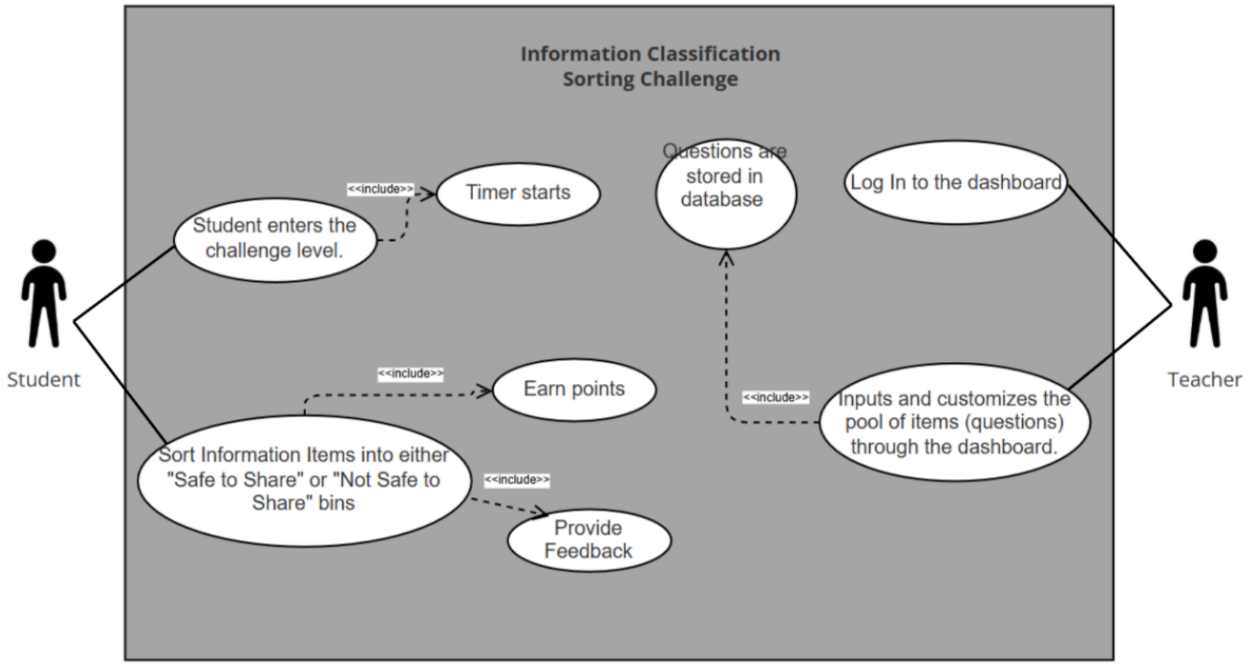
- **Routing and Navigation**
    - React Router handles client-side routing within the web dashboards, enabling smooth navigation between different views such as student progress pages, teacher analytics, and notification panels.

3.2. Functional requirements

Module 1: Gamified Cybersecurity Game Learning Module

Transaction 1.1 Information Classification Sorting Challenge

***Use Case Diagram** (Information Classification Sorting Challenges)*

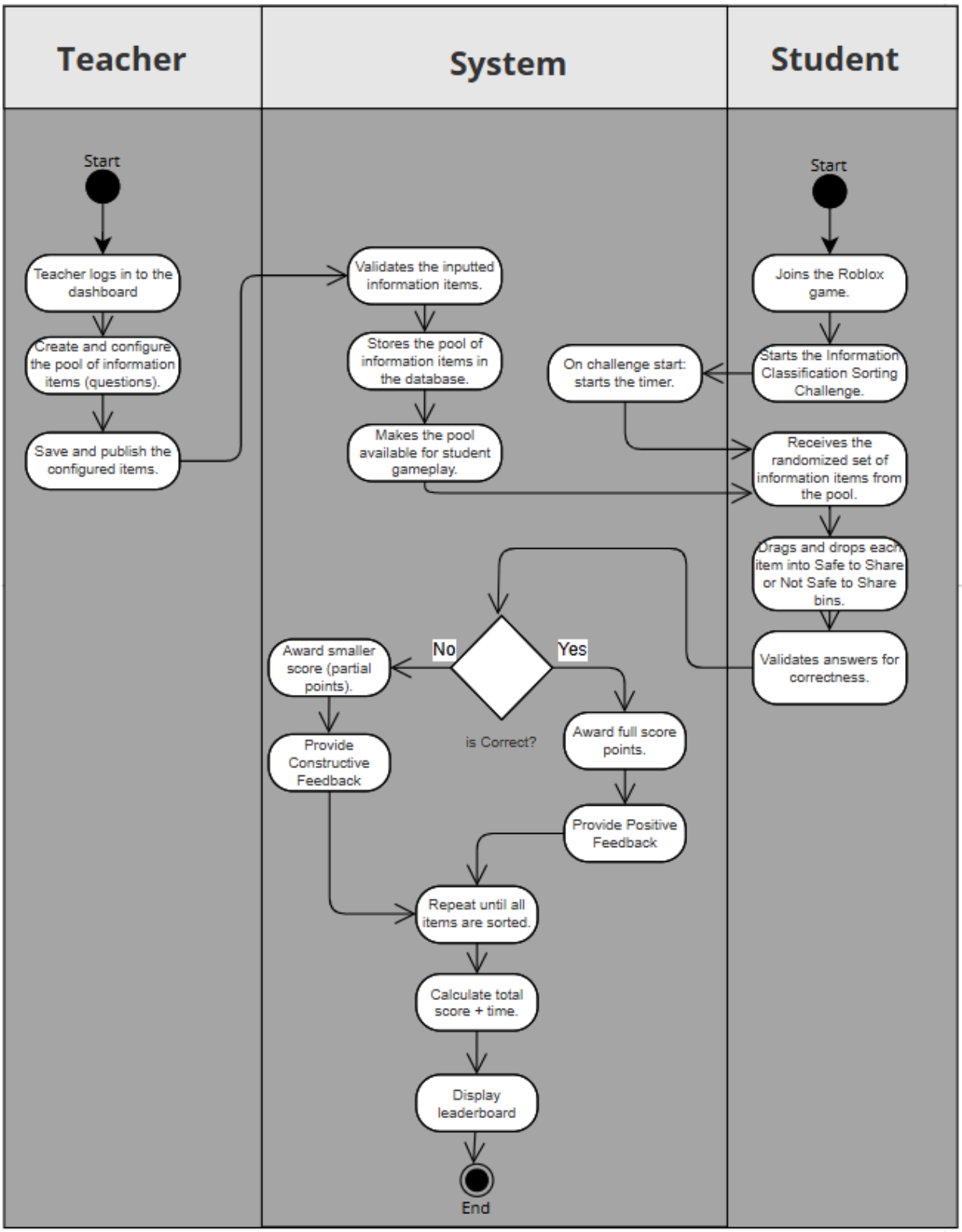


Use Case Description

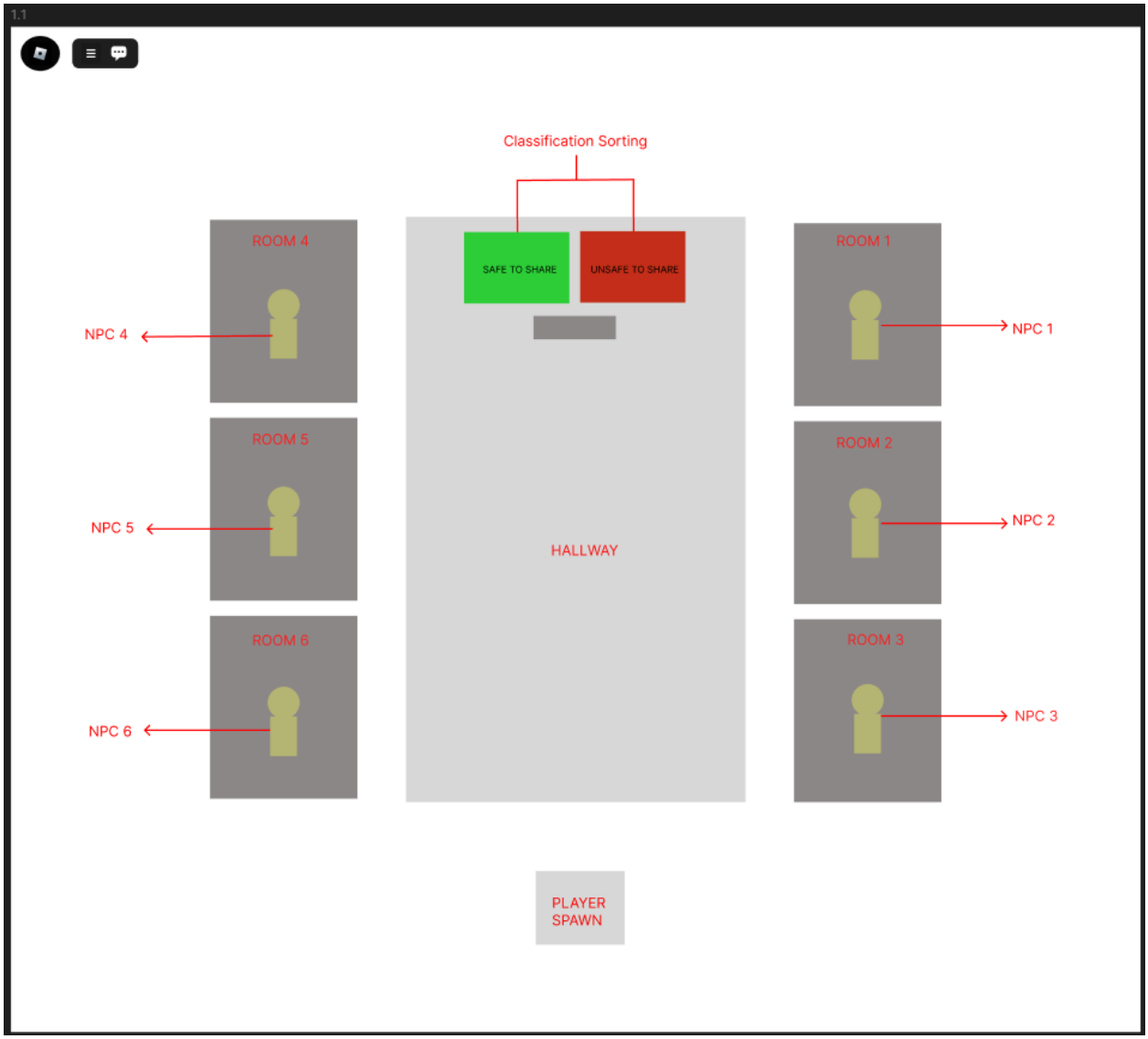
Use Case ID	UC-001
Use Case Name	*Information Classification Sorting Challenges*
Actor	Student, Teacher
Description	This use case allows students to participate in an interactive sorting challenge where they categorize digital items as either Safe to Share or Not Safe to Share. The system tracks their performance by measuring accuracy and completion time, awards points, and submits the results to the database via a REST API. Teachers configure the pool of information items beforehand using the teacher's dashboard.
Flow of Events	1. Student enters the challenge level. 2. System starts the challenge timer. 3. System retrieves the pool of information items (configured by the teacher) and displays them randomly to the student. 4. Student drags and drops each item into either *Safe to Share* or *Not Safe to Share* bins. 5. System validates each drop and records the student's choices. 6. Student completes all items.
Precondition	● Student must be registered and logged into the Roblox game. ● Teacher must have configured the pool of information items via the dashboard.

Postcondition	● Student's results (accuracy, completion time, score, points) are recorded in the database. ● Student receives performance feedback and points are updated. ● Data becomes available for the teacher's reports and progress tracking.

Activity Diagram (*Information Classification Sorting Challenges)*

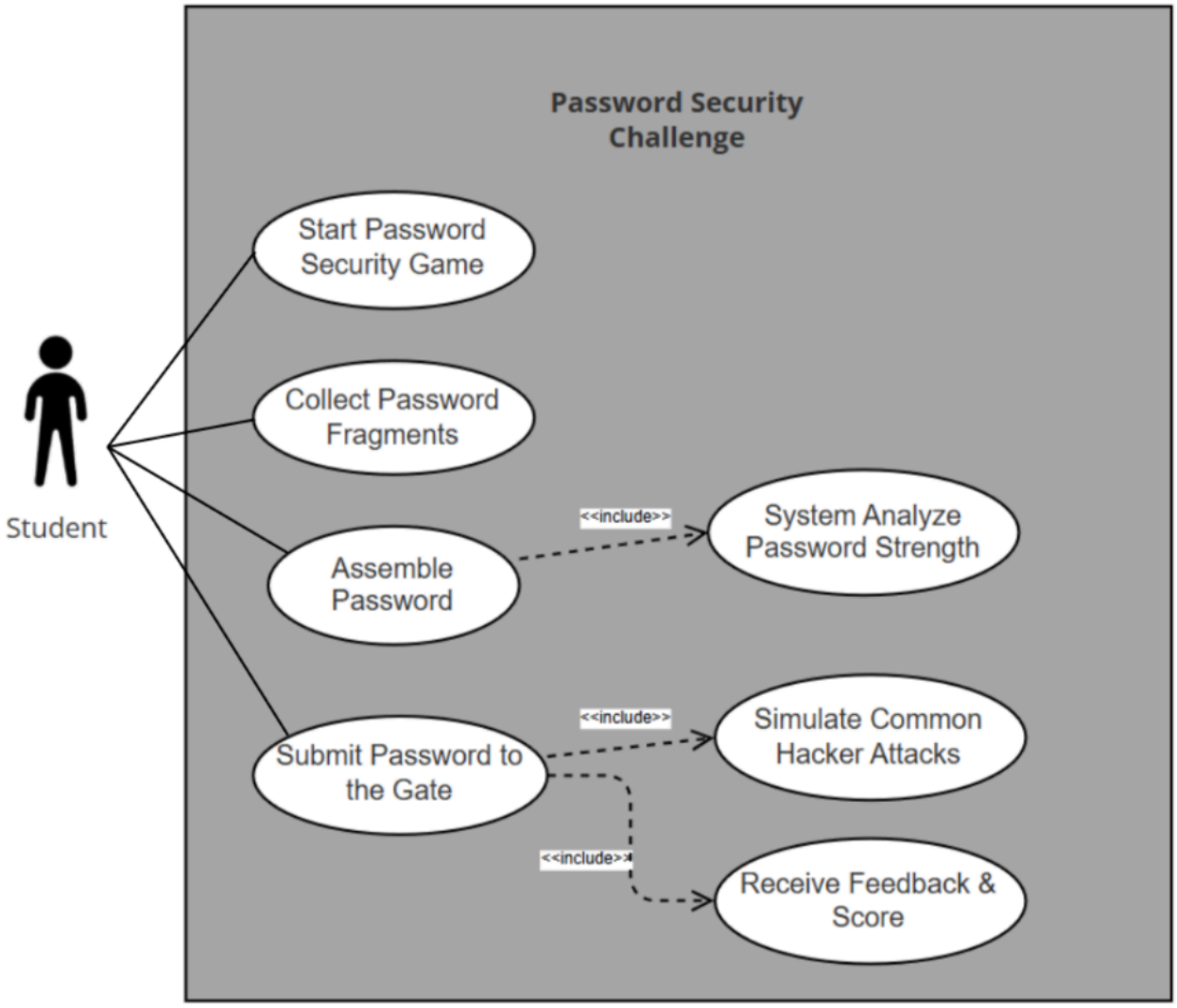


Wireframe *(Information Classification Sorting Challenges)*



Transaction 1.2 Password Security Challenge

***Use Case Diagram** (Password Security Challenge)*

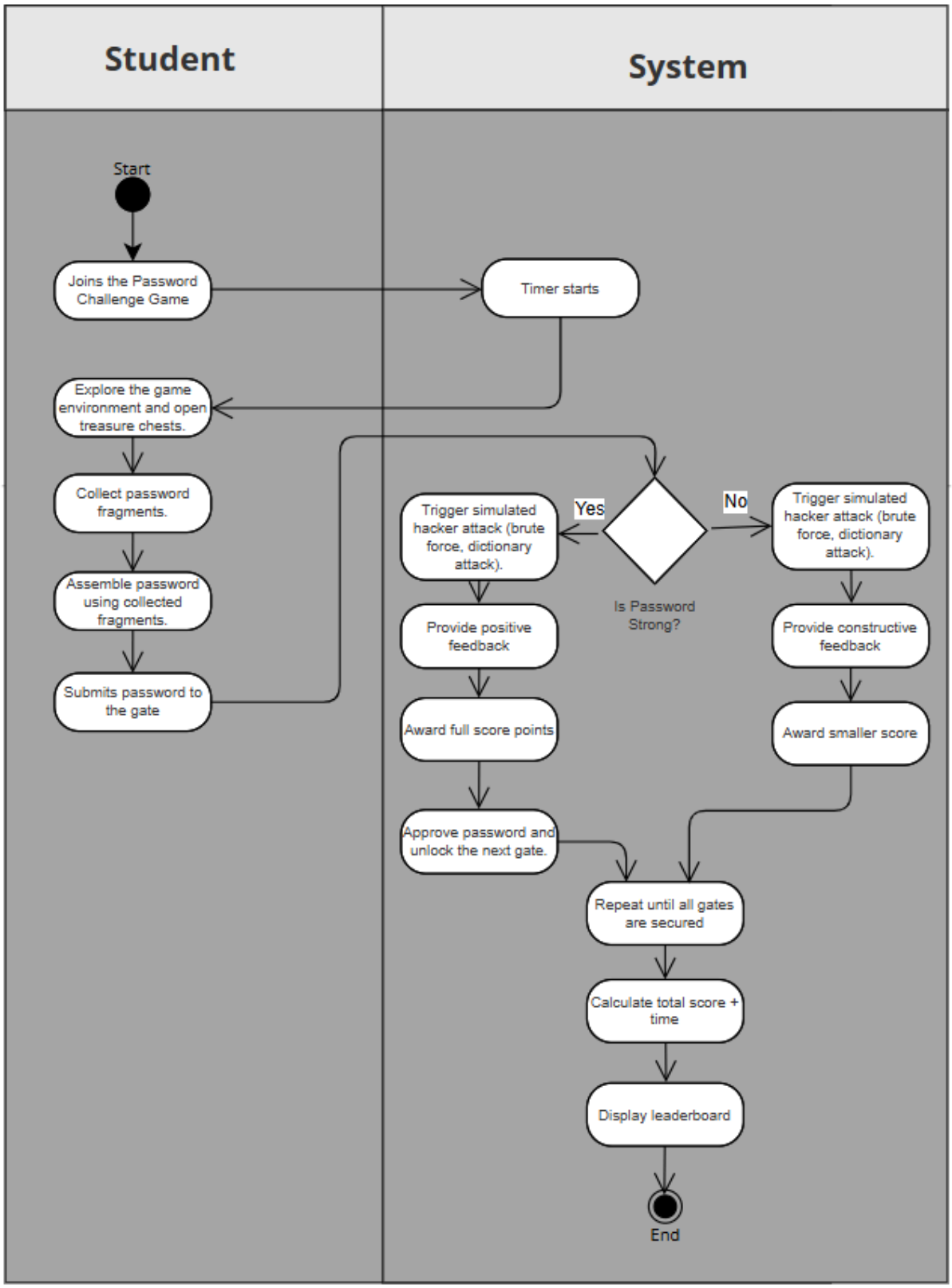


Use Case Description

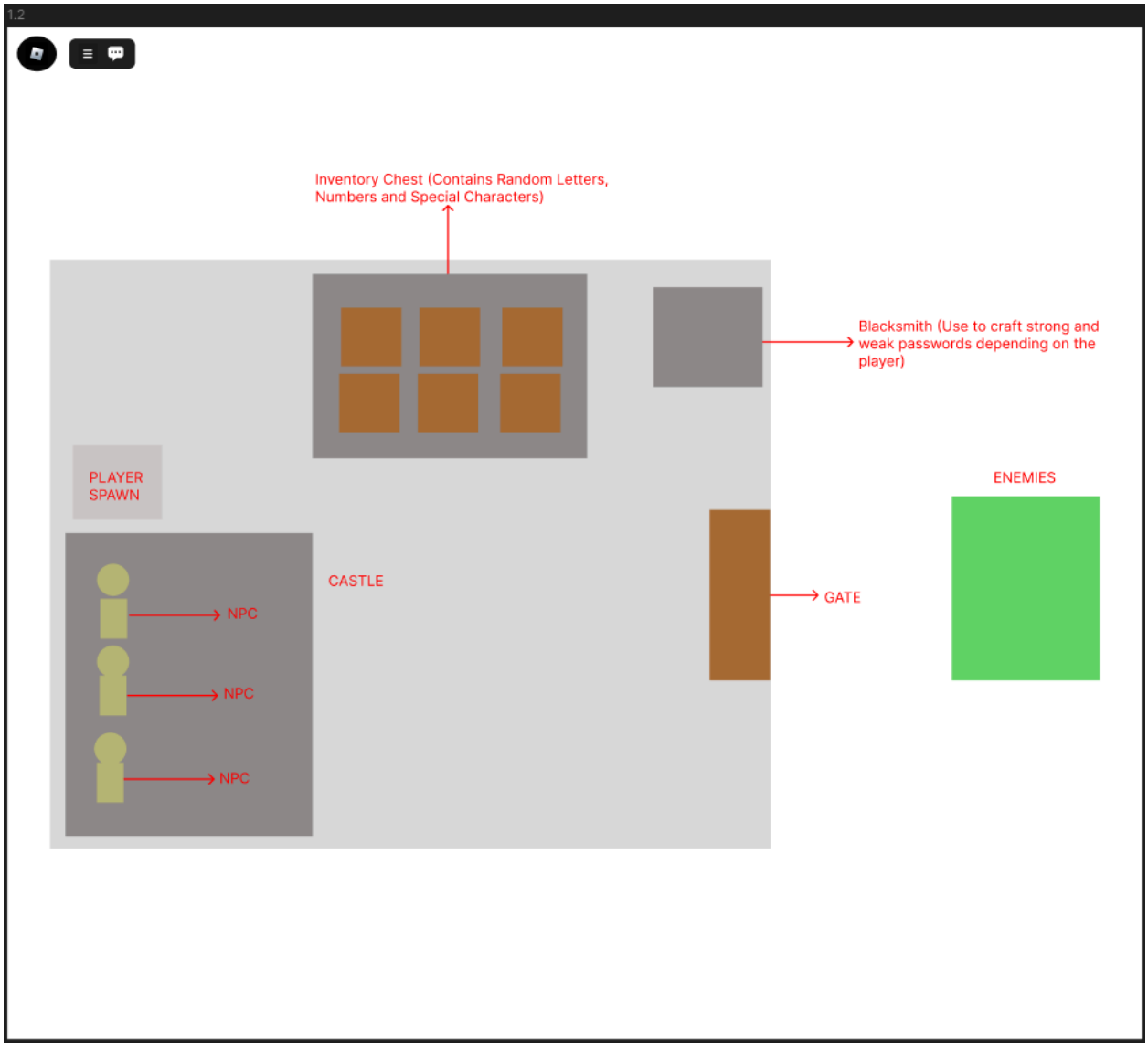
Use Case ID	UC-002
Use Case Name	*Password Security Challenge*
Actor	Student
Description	The Password Security Game is an interactive challenge where the student collects password fragments from treasure chests, evaluates their strength, and assembles a secure password to unlock the final gate. The system simulates real-world hacker attack methods (e.g., brute force, dictionary attacks) to demonstrate the risks of weak passwords. Players must reinforce their password if it is vulnerable, learning password best practices through immediate feedback and gameplay consequences.
Flow of Events	1. Student enters the Password Security Game. 2. System initializes the challenge and displays treasure chests containing password fragments. 3. Student opens chests and collects fragments. 4. System presents password assembly interface. 5. Student selects and arranges fragments to form a password. 6. System evaluates password strength. 7. System simulates hacker attack attempts. 8. System awards points, provides positive feedback, and submits results via REST API.
Precondition	● Student must be registered and logged into the Roblox game. ● Student must finish the prerequisite level which is Level 1.

Postcondition	The system records the player's progress and updates their score and leaderboard position.

Activity Diagram *(Password Security Challenge)*

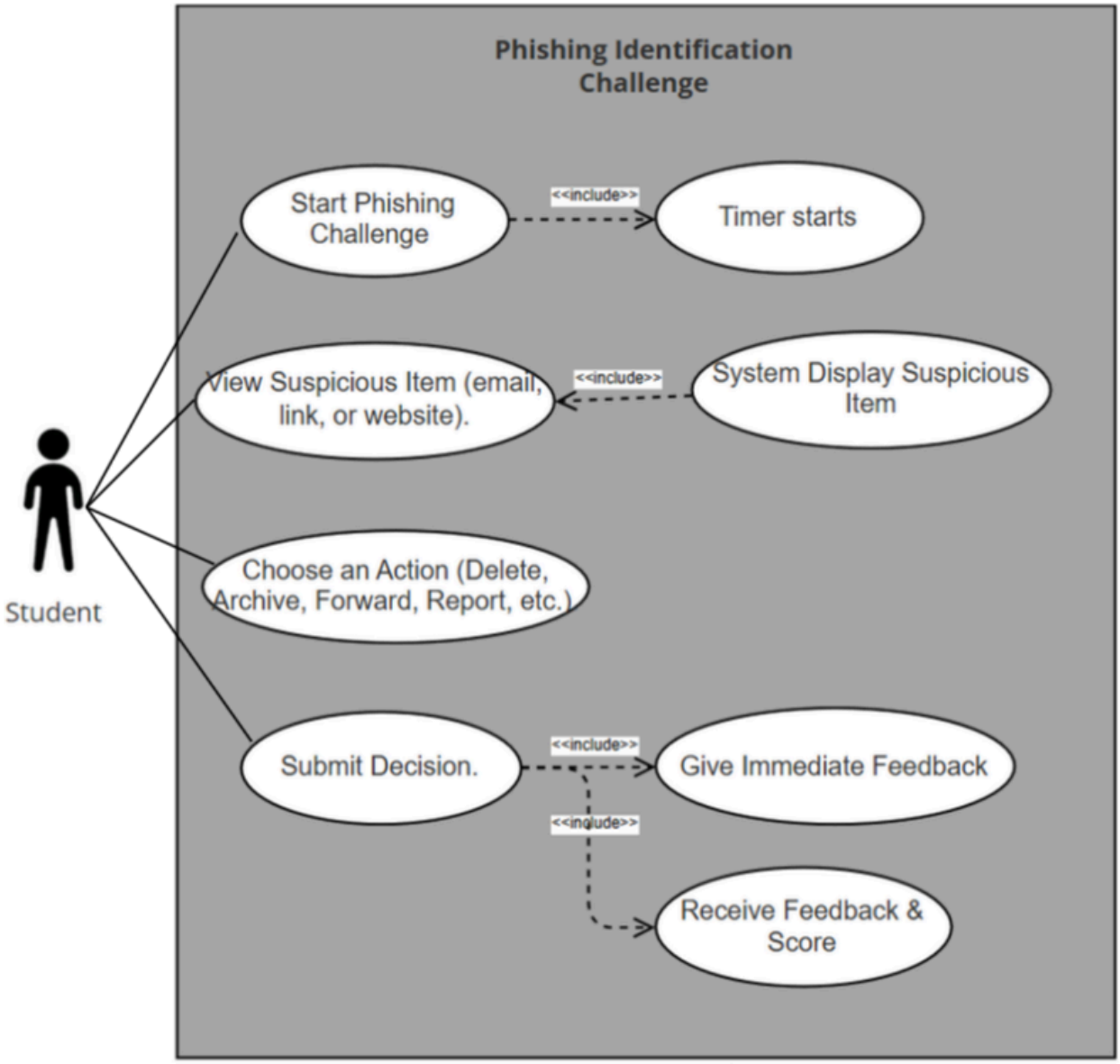


Wireframe *(Password Security Challenge)*



Transaction 1.3 Phishing Identification Challenge

***Use Case Diagram** (Phishing Identification Challenge)*

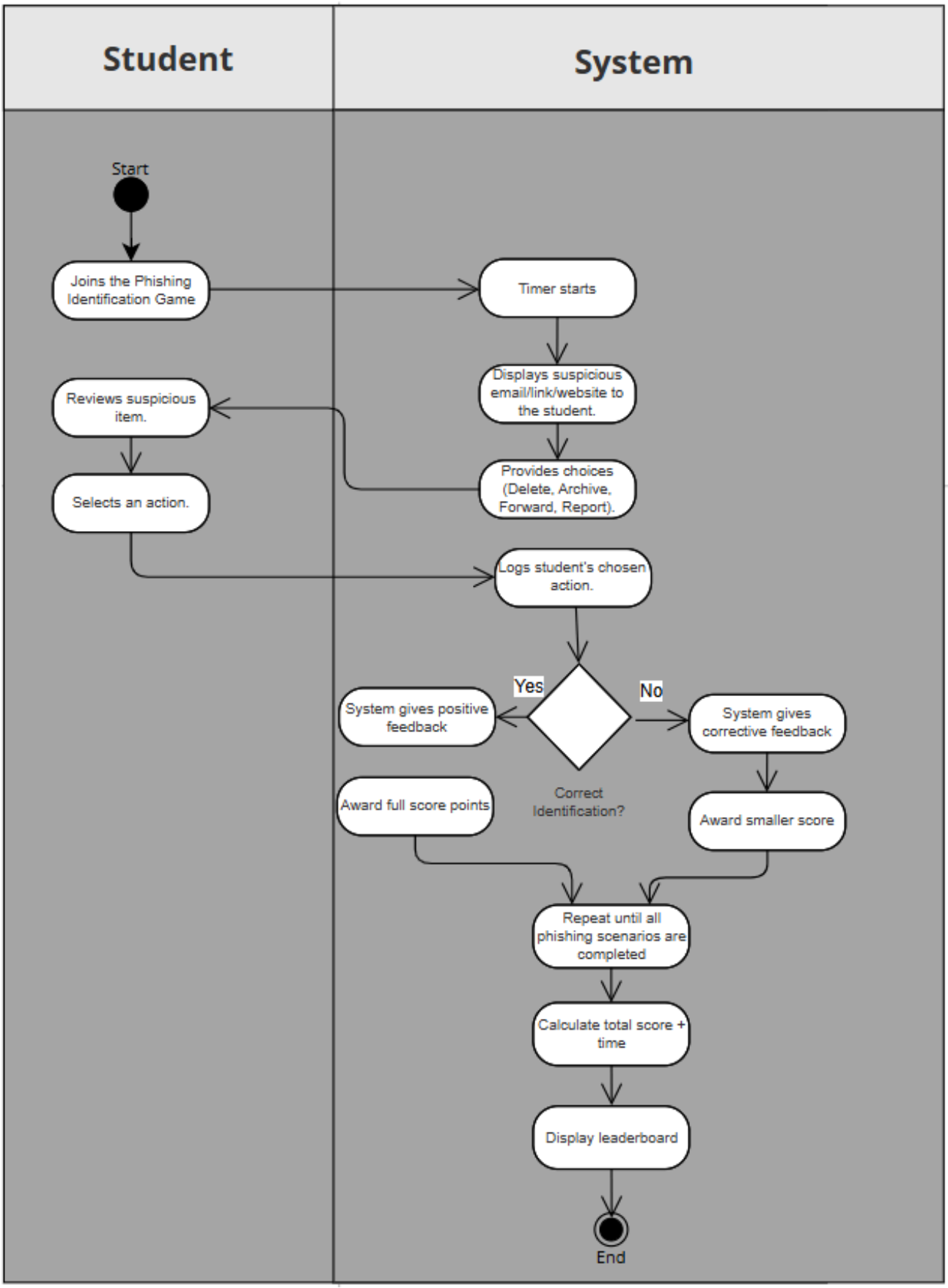


Use Case Description

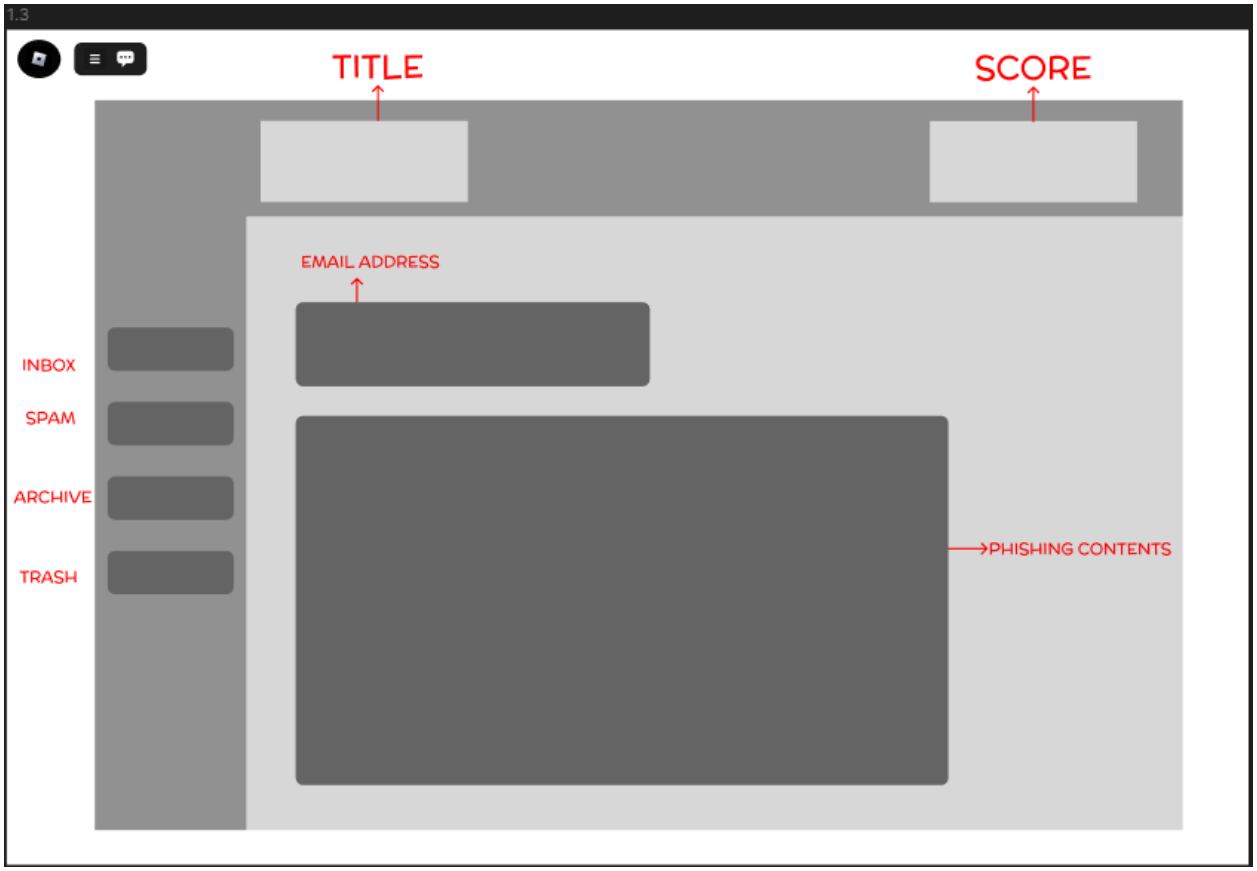
Use Case ID	UC-003
Use Case Name	*Phishing Identification Challenge*
Actor	Student
Description	This use case describes how a student participates in the Phishing Identification Challenge. The student inspects simulated emails, websites, and links within the escape room environment and decides how to act (e.g., delete, archive, forward, flag as phishing, or interact). Each action is logged by the system with the item ID, correctness of the action, and timestamp. The system evaluates the student's decisions, assigns scores, and provides immediate feedback. Correct identifications increase the score and progress, while incorrect actions result in lower scores but still give feedback. Final scores are tallied and submitted to the leaderboard service for ranking among peers.
Flow of Events	1. The student enters the virtual escape room and is presented with phishing scenarios. 2. The system displays a suspicious item (email, link, or webpage). 3. The student reviews the item for phishing indicators. 4. The system provides multiple possible actions (delete, archive, forward, report/flag as phishing, etc.) depending on the scenario type. 5. The student selects the most appropriate action. 6. The system checks the correctness of the action. 7. The system calculates the total score and

	sends it to the leaderboard service. 8. The leaderboard is updated, and the student can view their standing.
Precondition	● Student must be registered and logged into the Roblox game. ● Student must finish the prerequisite level which is Level 2.
Postcondition	The system records the player's progress and updates their score and leaderboard position.

Activity Diagram *(Phishing Identification Challenge)*

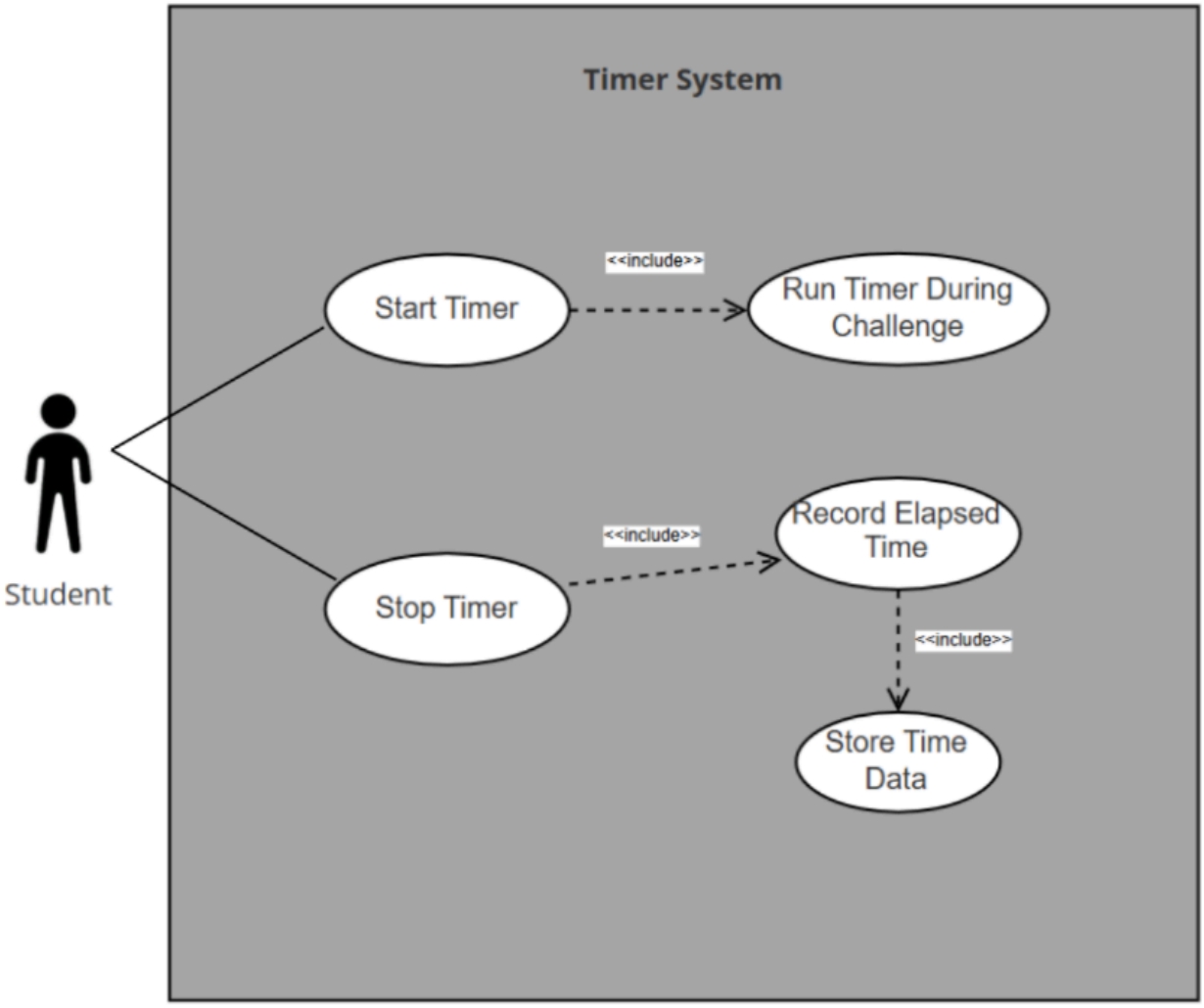


Wireframe *(Phishing Identification Challenge)*



Transaction 1.4 Timer System

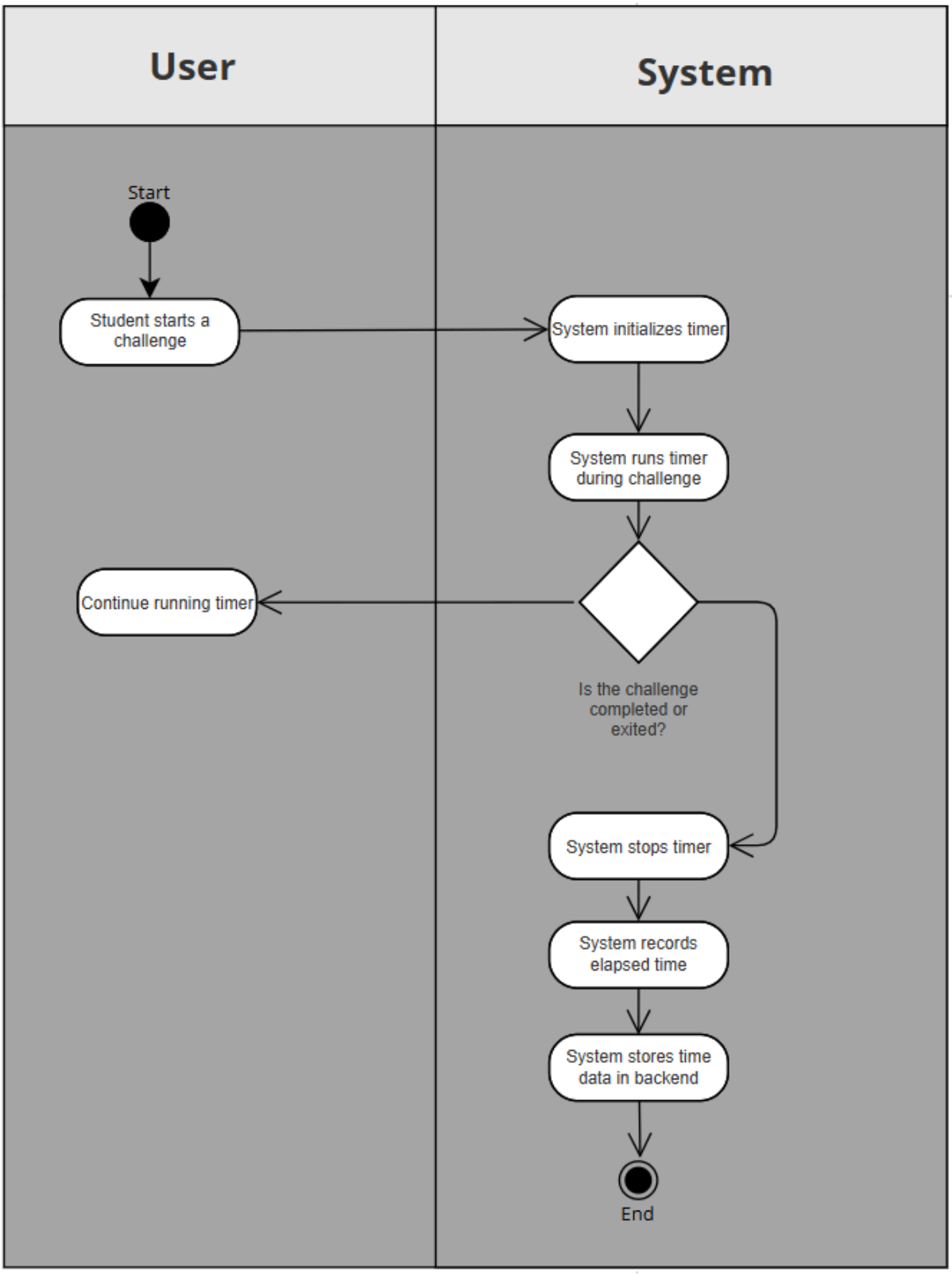
Use Case Diagram (*Timer System****)***



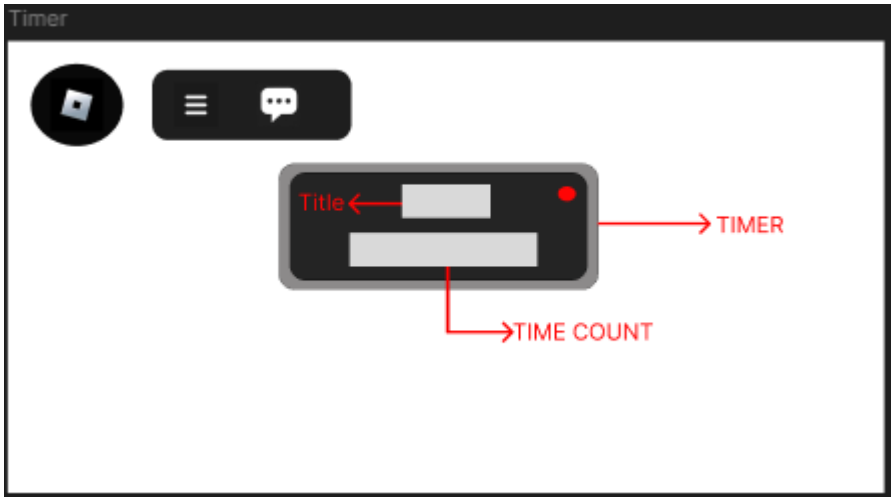
Use Case Description

Use Case ID	UC-004
Use Case Name	*Timer System*
Actor	Student
Description	The Timer System tracks the time spent by a student during each challenge. When a student starts a challenge, the system automatically initializes a timer. The timer runs until the student completes or exits the challenge, after which the total elapsed time is recorded. This recorded time is then stored in the backend and used as a factor in the scoring system and leaderboard ranking.
Flow of Events	1. The student starts a challenge level. 2. The system initializes the countdown timer at the start of the challenge. 3. The student either completes the challenge or exits before completion. 4. The system automatically stops the timer. 5. The system records the elapsed time for that challenge level in the database or log.
Precondition	The student must have started a challenge (Online Privacy, Password Security, or Phishing Awareness).
Postcondition	● The student's total completion time for the challenge is recorded. ● The time data is stored in the backend for scoring and ranking purposes. ● The recorded time is made available to the leaderboard for display alongside the score.

***Activity Diagram** (Timer System)*

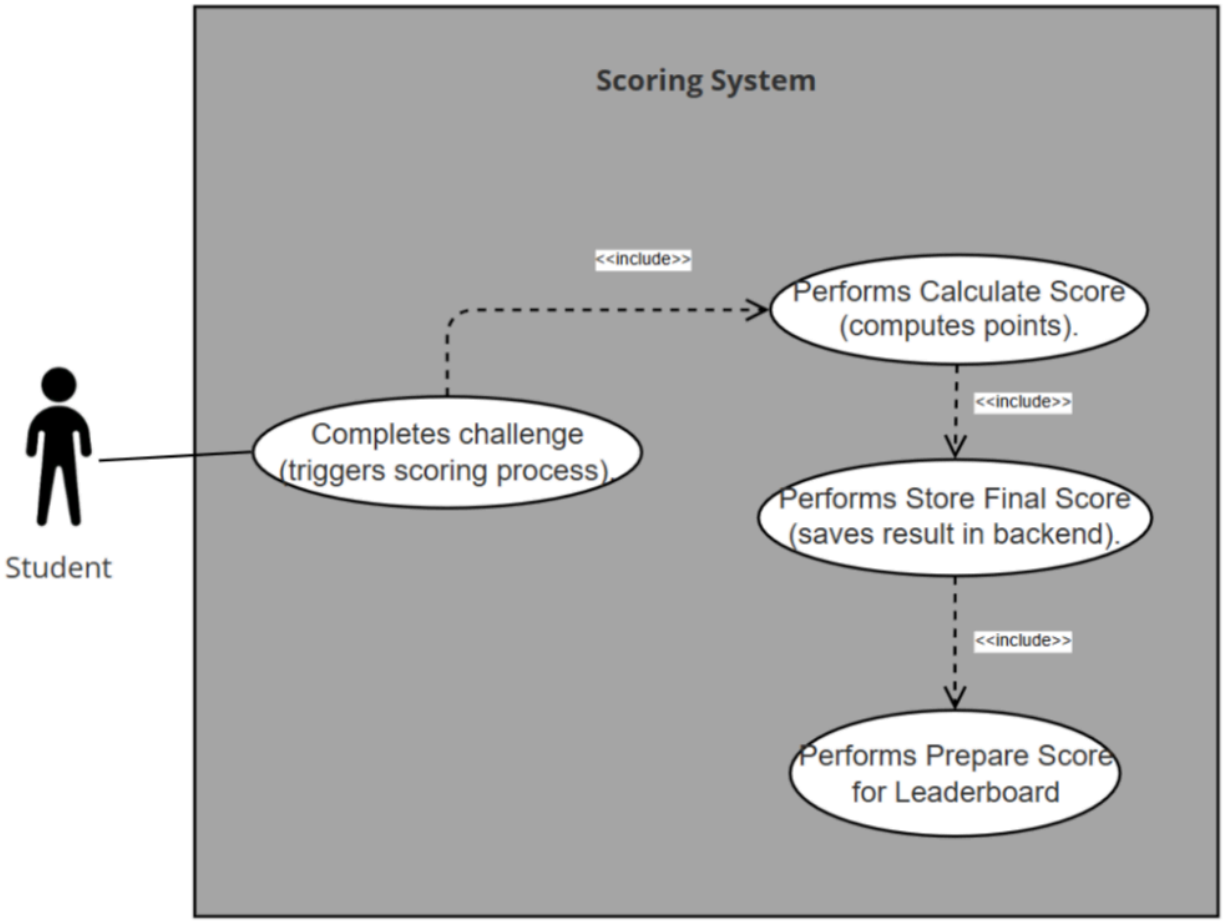


***Wireframe** (Timer System)*



Transaction 1.5 Scoring System

Use Case Diagram *(Scoring System)*

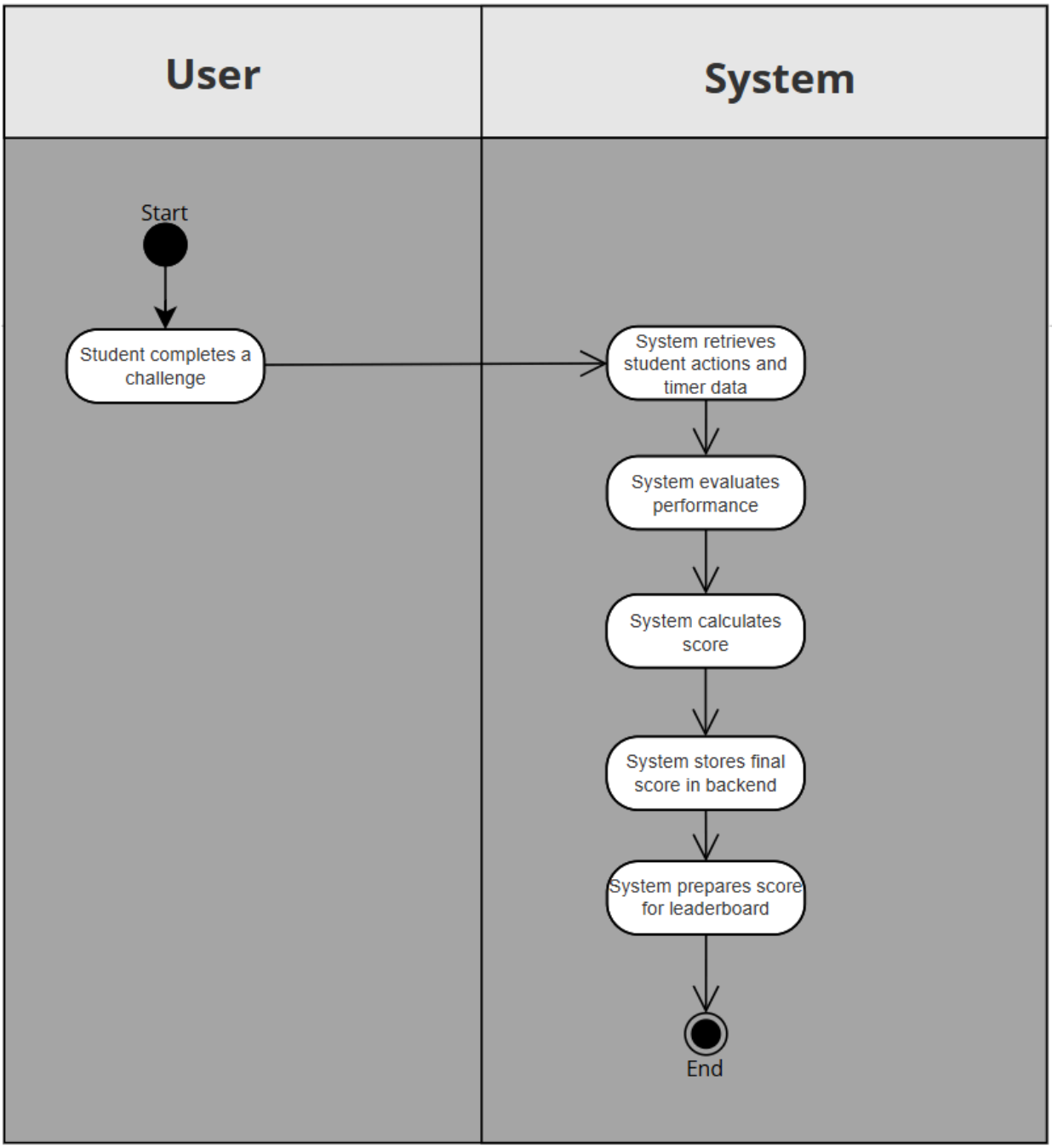


Use Case Description

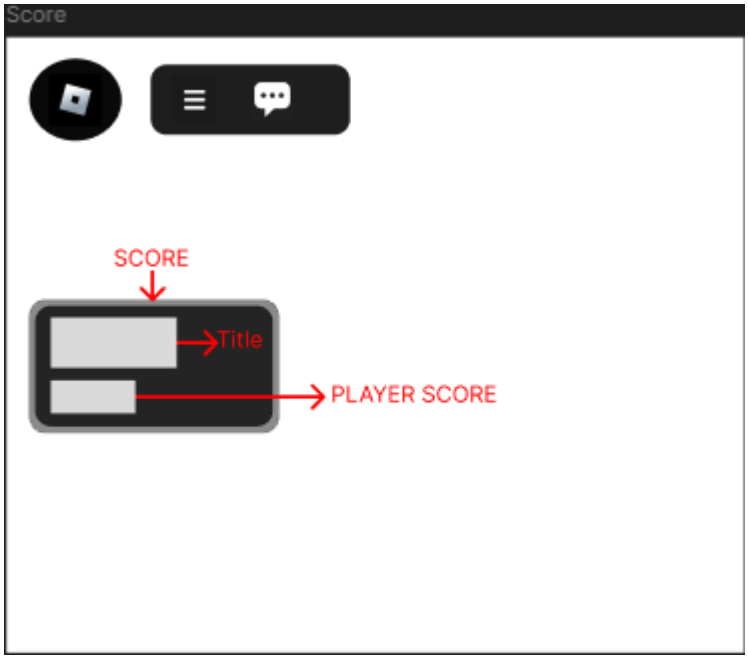
Use Case ID	UC-005
Use Case Name	*Scoring System*
Actor	Student
Description	The Scoring System allows the game to calculate and assign points to students after completing a challenge. Once a challenge is finished, the system evaluates the student's performance based on completion. Correct actions increase the score, while incorrect actions may result in deductions or no points. The system then compiles the total score, stores it in the backend, and prepares it to be reflected in the leaderboard.
Flow of Events	1. The student starts and completes a challenge level. 2. The system retrieves the student's recorded actions (correct/incorrect attempts) and total completion time from the Timer System. 3. The system awards points for correct actions and applies deductions (if any) for incorrect attempts. 4. The system compiles the total score for the challenge. 5. The system stores the finalized score in the backend database. 6. The system prepares the score to be displayed in the leaderboard system.
Precondition	The student must have played and completed a challenge (Online Privacy, Password Security, or Phishing Awareness).
Postcondition	● The student's score is calculated and finalized

	based on performance. ● The score is stored in the backend for record-keeping. ● The score is ready to be displayed and ranked in the leaderboard system.

Activity Diagram *(Scoring System)*

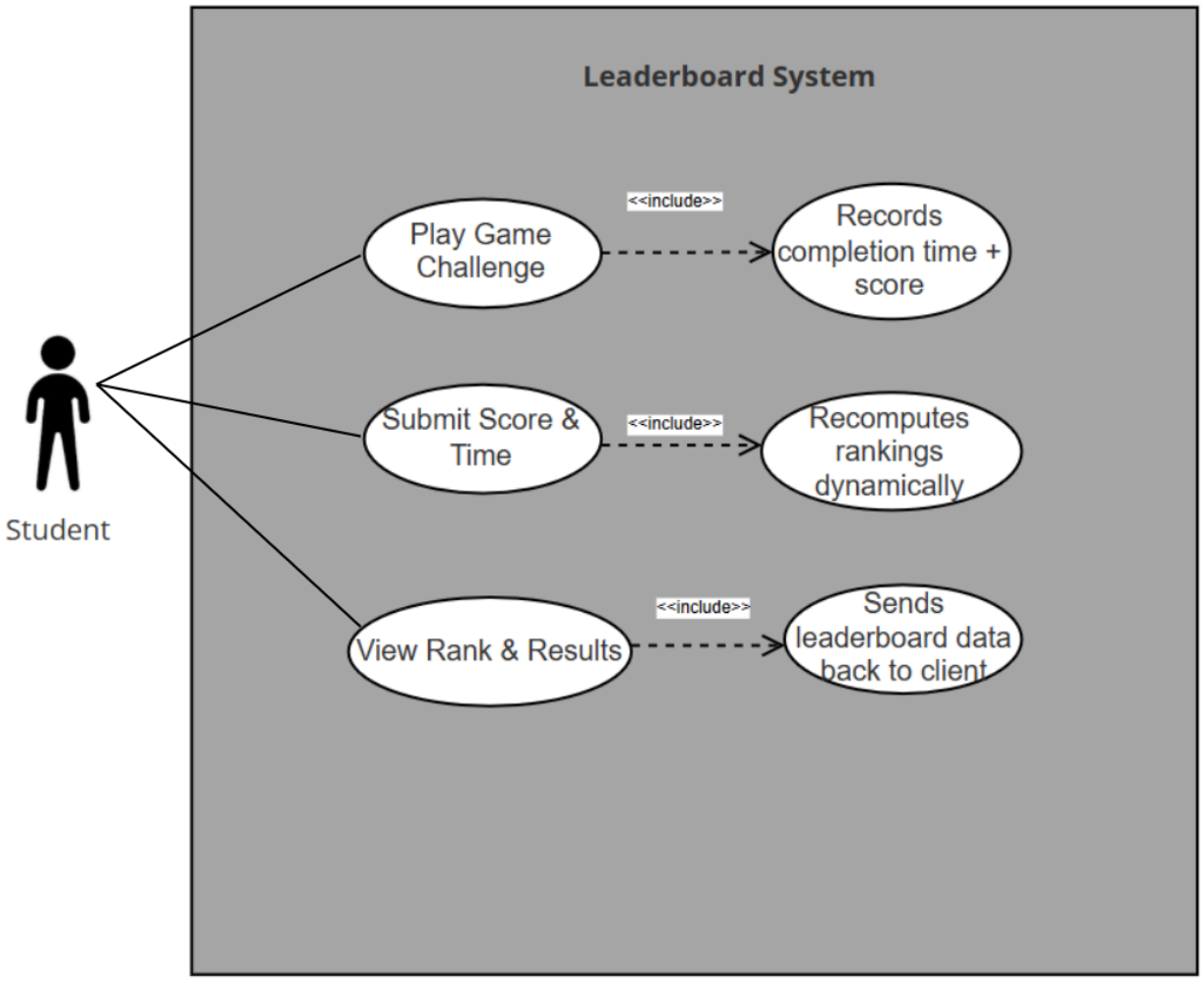


Wireframe *(Scoring System)*



Transaction 1.6 Leaderboard System

***Use Case Diagram** (Leaderboard System)*

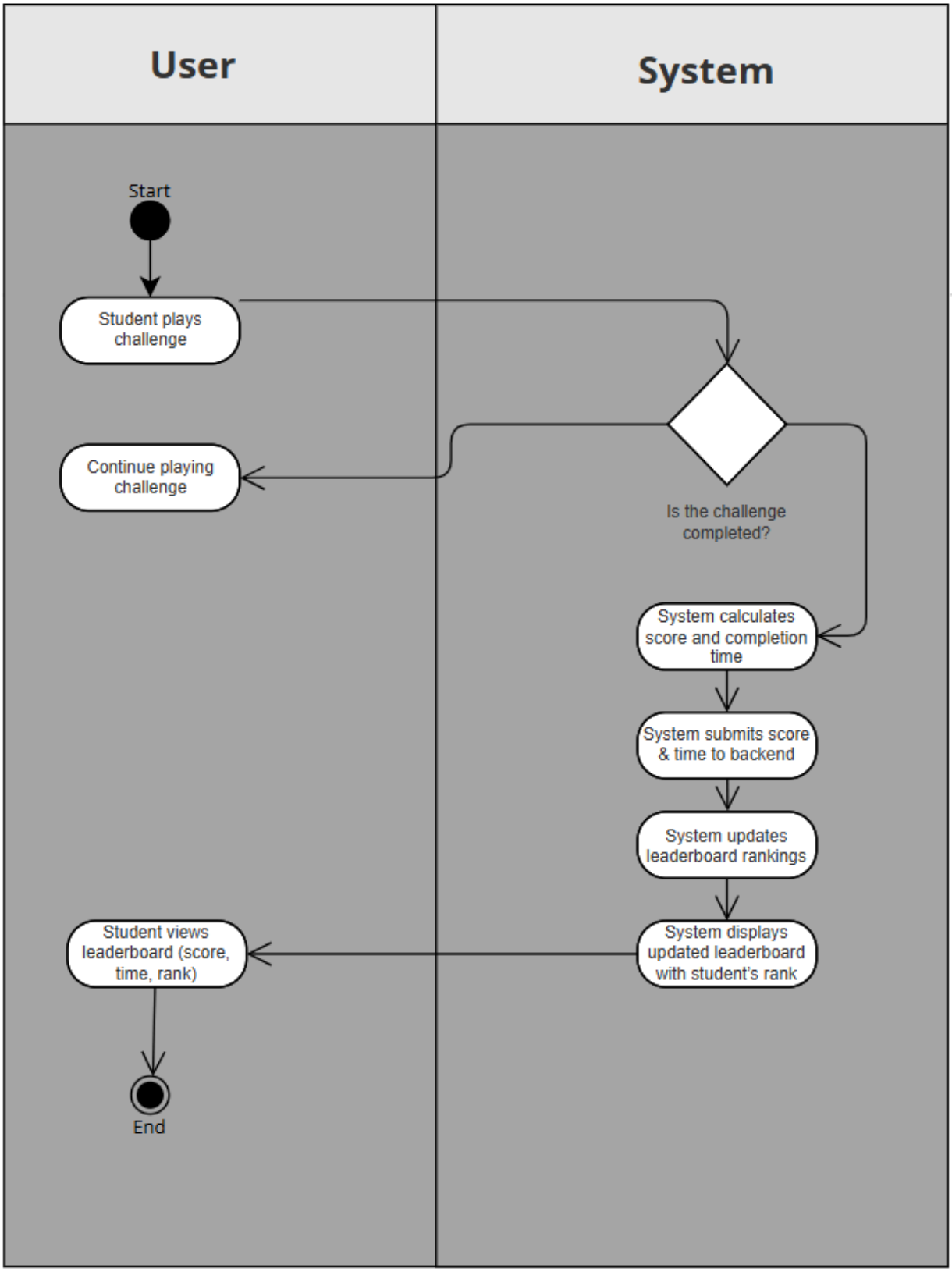


Use Case Description

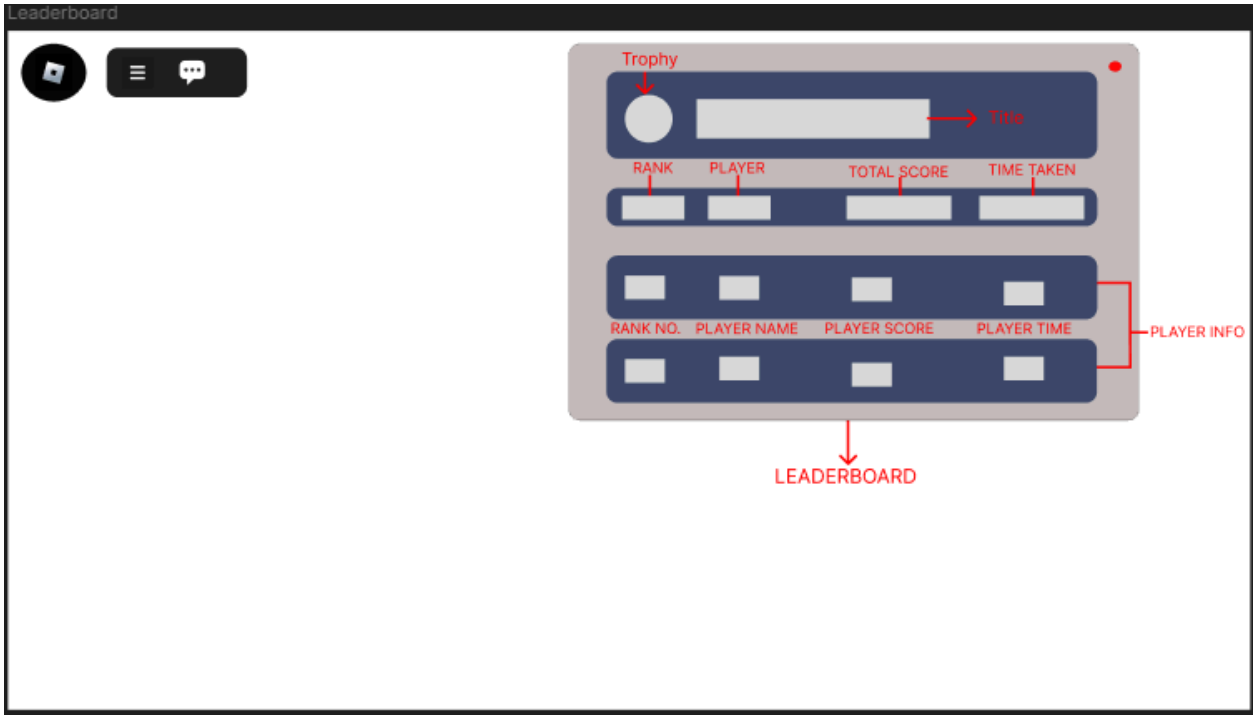
Use Case ID	UC-006
Use Case Name	*Leaderboard System*
Actor	Student
Description	The Leaderboard System allows students to view their performance ranking at the end of each game challenge. After completing a challenge, the system records the student's score and completion time, updates the leaderboard rankings in real time, and displays the updated results. This provides students with feedback on their progress and motivates them by showing their standing compared to other players.
Flow of Events	1. **Student** completes a challenge (Online Privacy, Password Security, or Phishing Awareness). 2. **System** calculates the final score based on accuracy, completion time, and difficulty. 3. **System** sends the score and completion time to the backend database. 4. **System** retrieves the updated leaderboard data. 5. **System** displays the leaderboard showing rankings, total scores, and completion times. 6. **Student** views their rank and compares performance with other players.
Precondition	The student must have completed a game challenge (Online Privacy, Password Security, or Phishing Awareness).
Postcondition	● The student's score and completion time are recorded in the system.

	● The leaderboard is updated with the latest ranking. ● The student can view their own rank and compare it with others.

Activity Diagram *(Leaderboard System)*



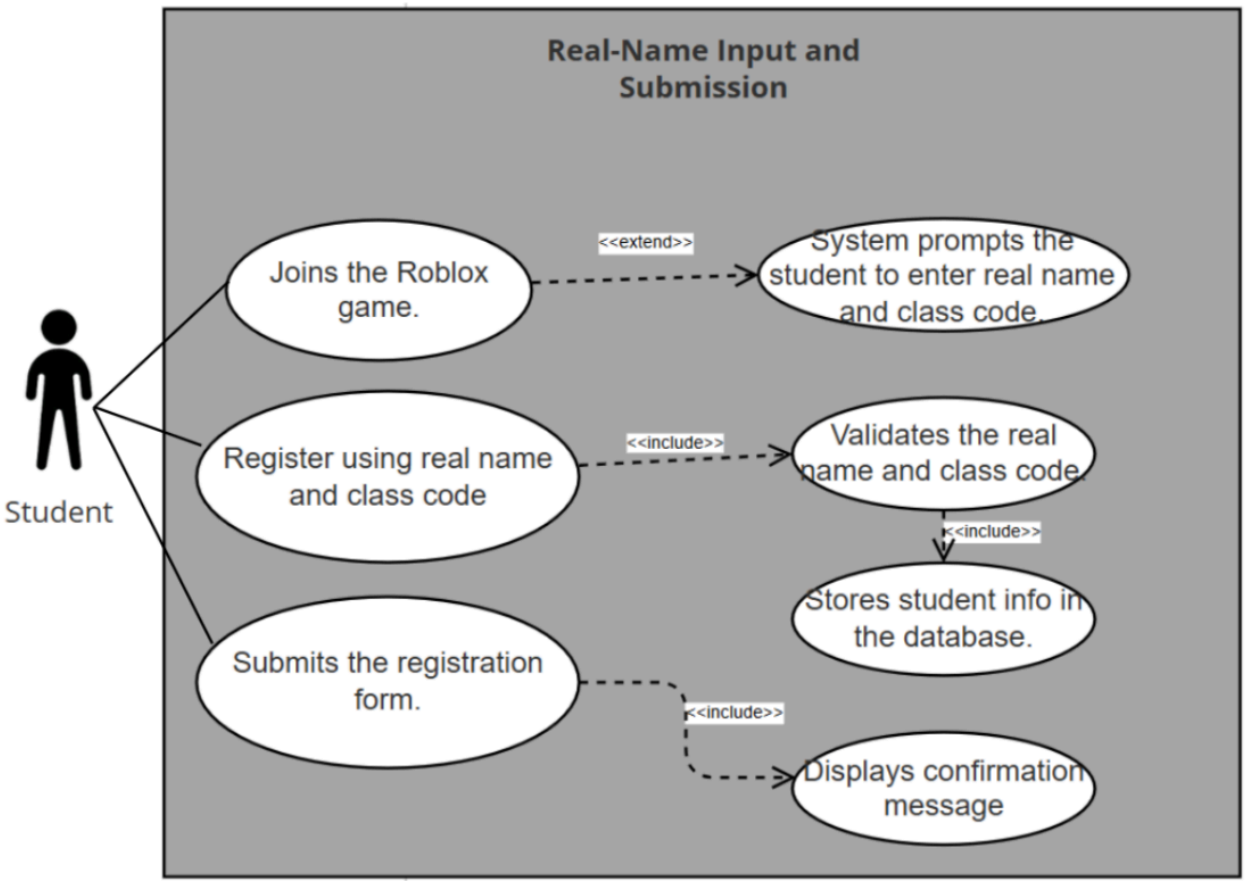
Wireframe *(Leaderboard System)*



Module 2: Student Real-Name Integration

Transaction 2.1 Real-Name Input and Submission

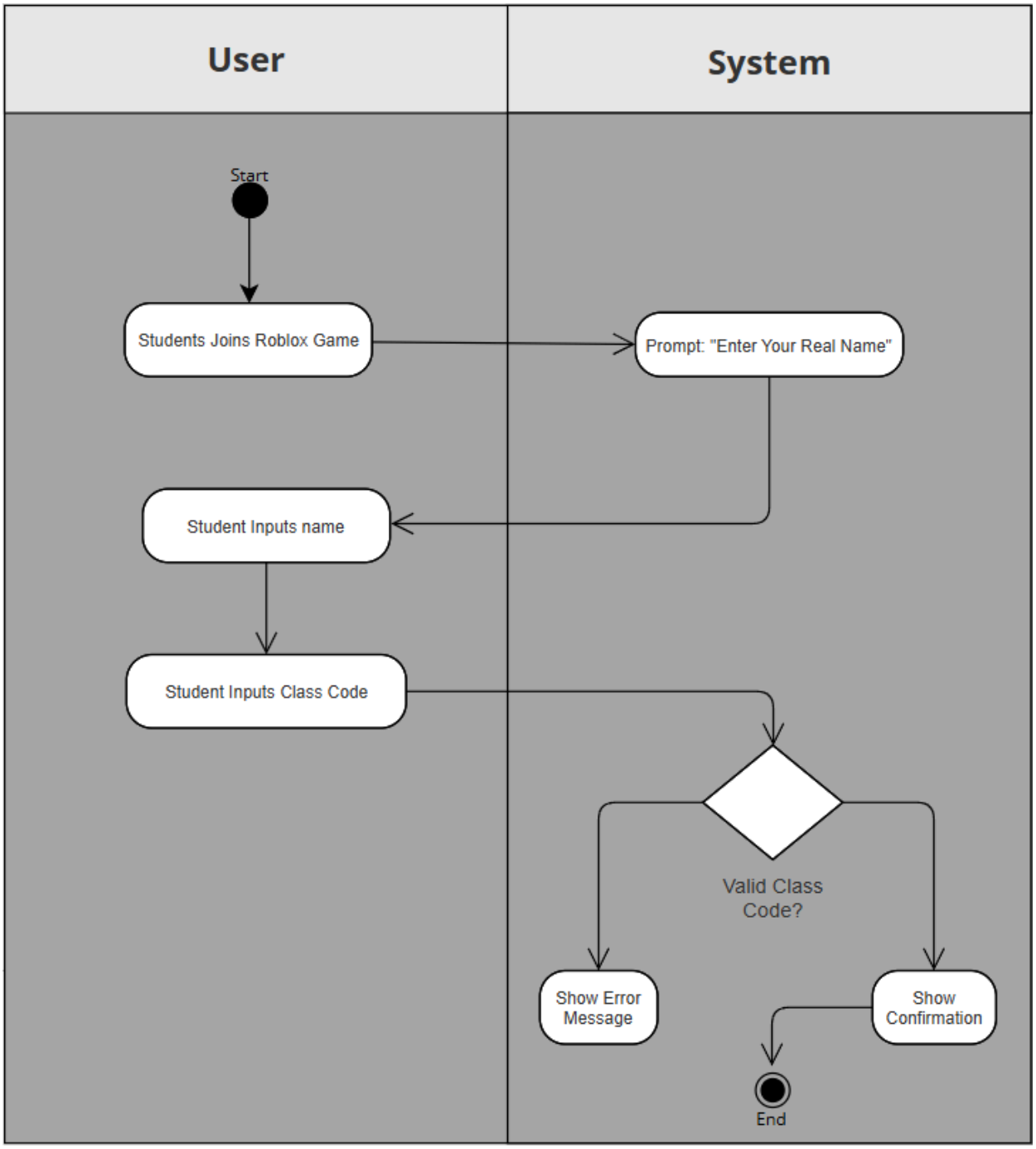
***Use Case Diagram** (Real-Name Input and Submission)*



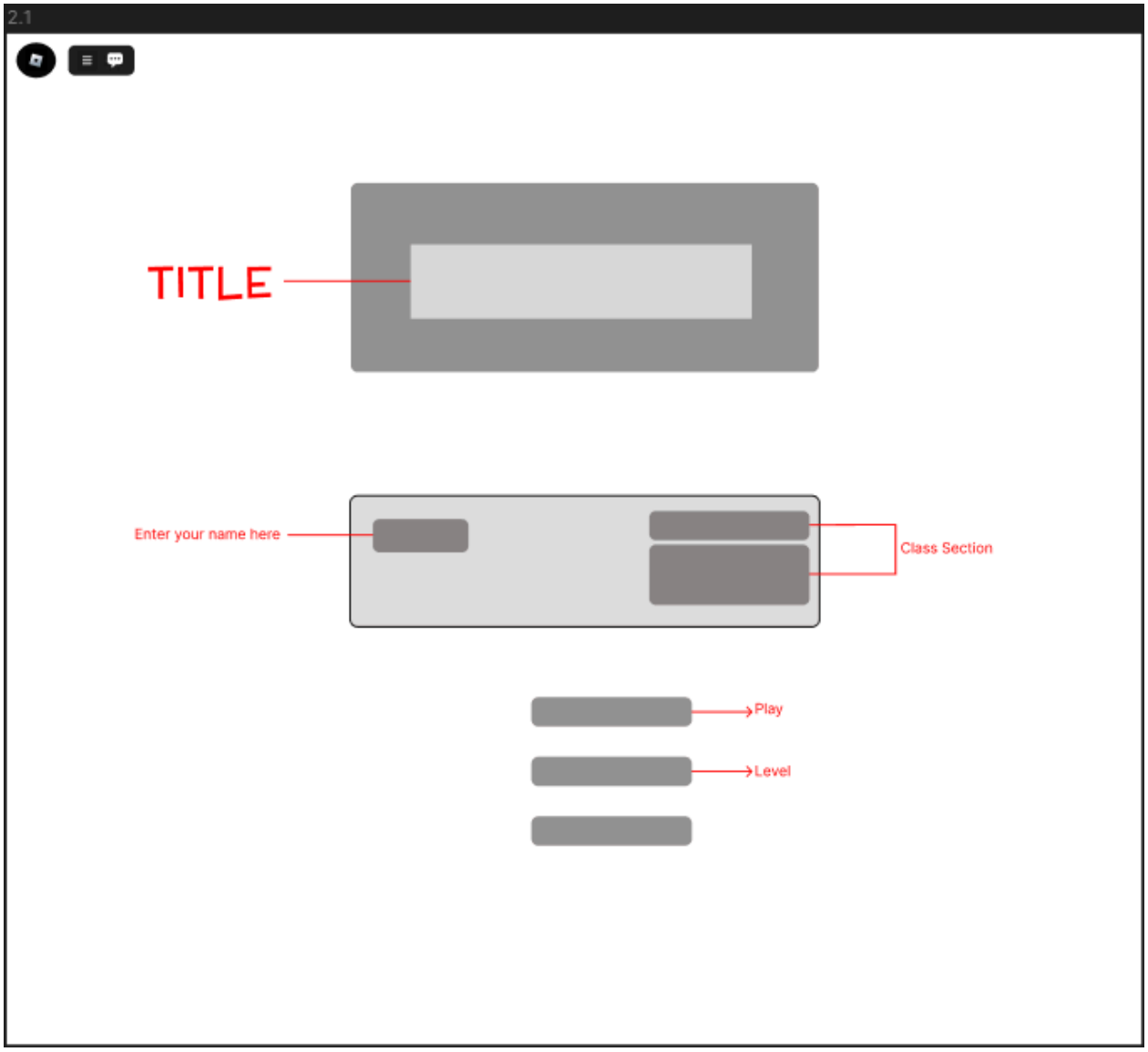
Use Case Description

Use Case ID	UC-001
Use Case Name	*Real-Name Input and Submission*
Actor	Student
Description	This use case allows students to input and submit their real names and class code upon joining the Roblox game. This real name is used for identifying students in reports, tracking progress, and displaying their achievements.
Flow of Events	1. Student joins the Roblox game. 2. The system prompts the student to enter their real name. 3. Student inputs their real name. 4. Student inputs their proper class code. 5. System displays a confirmation message: “You have been registered.”
Precondition	● Student has successfully joined the Roblox game. ● The system is running and accessible. ● The class code for each section already exists in the system (created by the teacher). ● Student has not yet registered with their real name in the system.
Postcondition	● Student’s **real name** is successfully stored in the system database. ● Student is linked to their **designated class section** via the provided valid class code. ● System confirms registration by displaying: *“You have been registered.”* ● Student can now be tracked in progress monitoring, and achievement displays.

Activity Diagram *(Real-Name Input and Submission)*

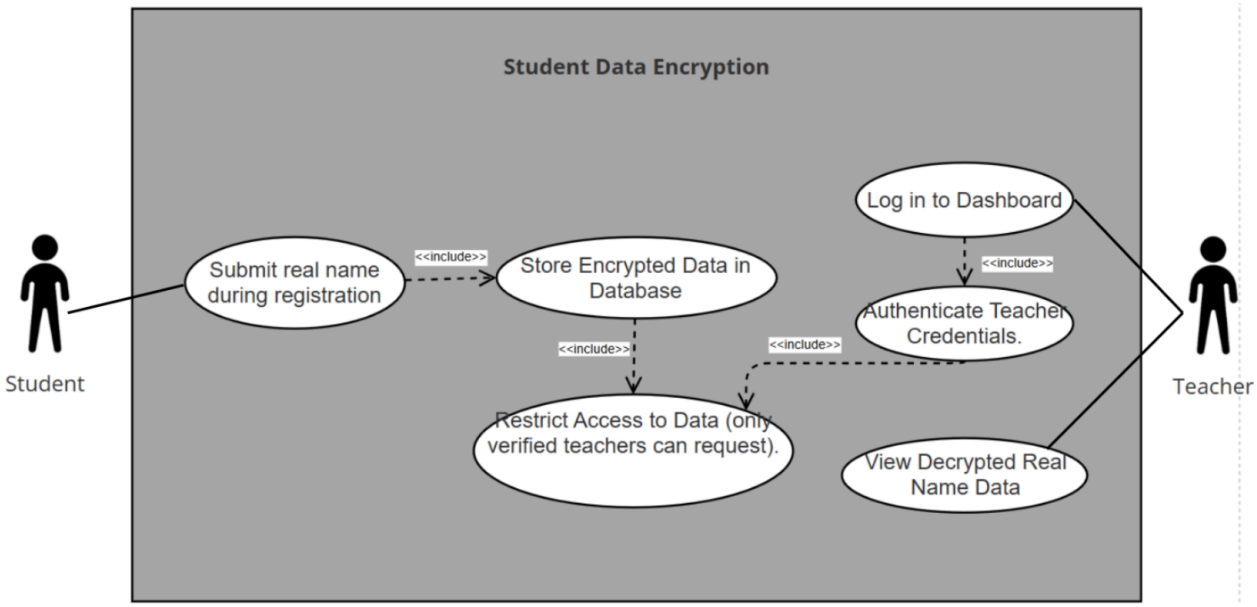


Wireframe *(Real-Name Input and Submission)*



Transaction 2.2 Student Data Encryption

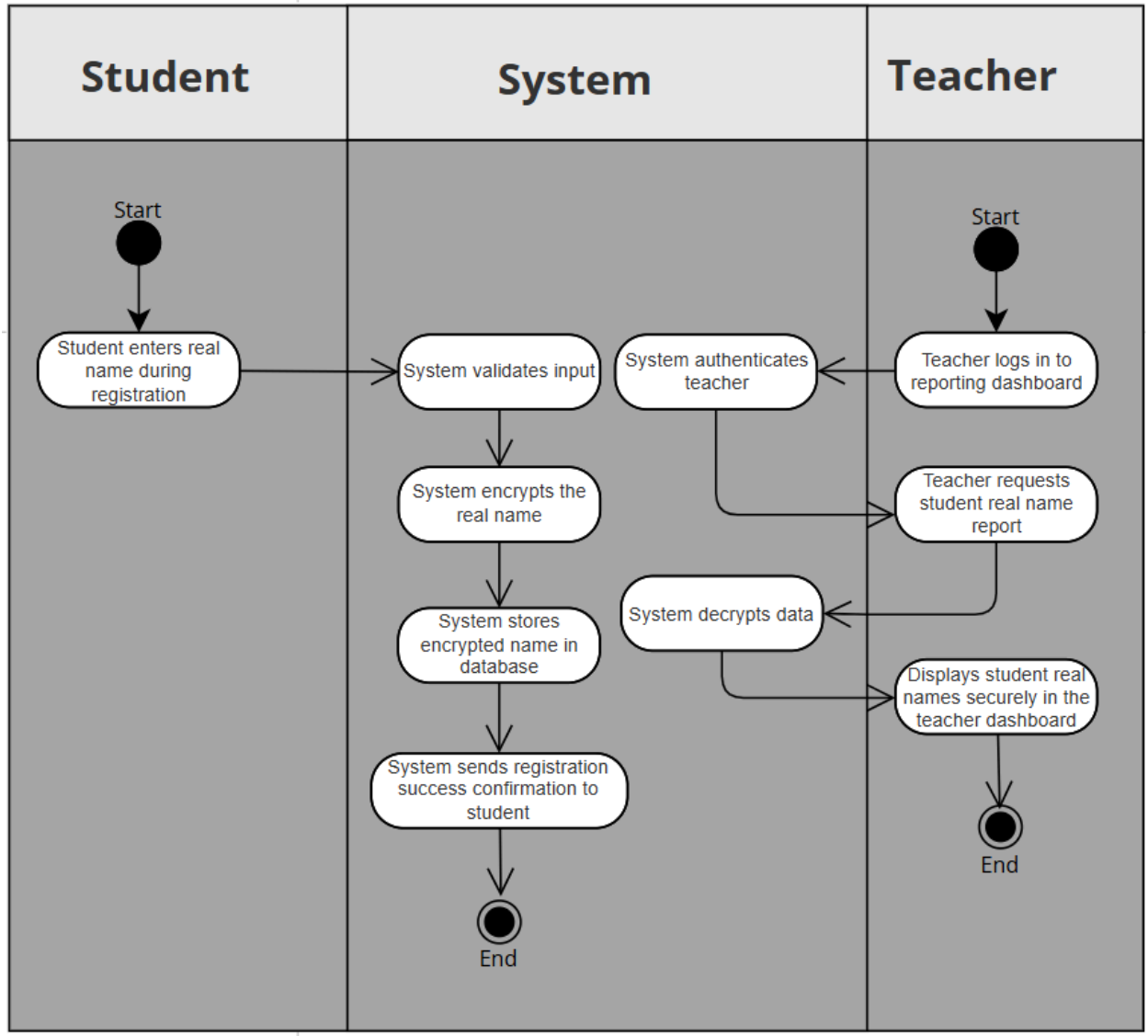
***Use Case Diagram** (Student Data Encryption)*



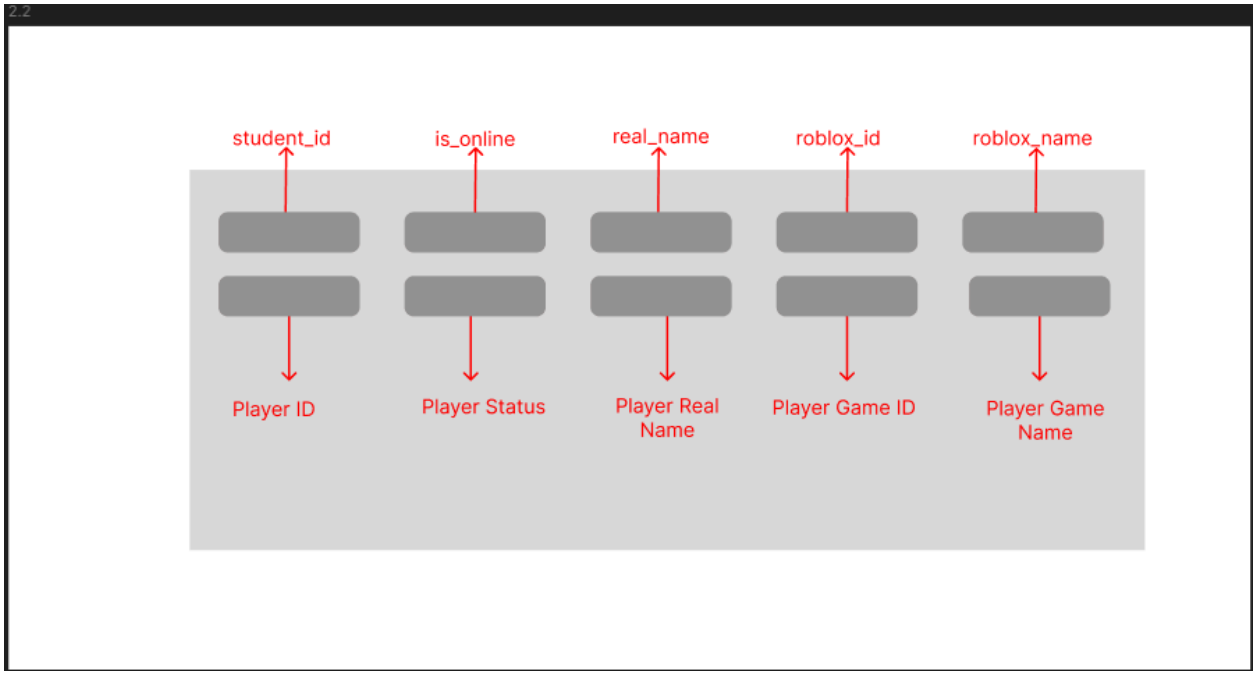
Use Case Description

Use Case ID	UC-001
Use Case Name	*Student Data Encryption*
Actor	Student, Teacher
Description	The system ensures that all submitted real names from students are encrypted upon submission and securely stored in the database. Only authorized teachers can access these real names through the reporting dashboard. The system prevents unauthorized access, ensuring confidentiality and compliance with data privacy standards.
Flow of Events	1. The student submits their real name during registration or verification. 2. The system receives the input and triggers the encryption module. 3. The system encrypts the student's real name using a secure algorithm. 4. The encrypted real name is stored in the database. 5. When an authorized teacher accesses the reporting dashboard, the system decrypts the data and displays the real name securely.
Precondition	● The student must be logged into the system. ● The student must provide their real name during the registration or verification process.
Postcondition	● The student's real name is securely encrypted and stored in the database. ● Only authorized teachers can view decrypted names via the reporting dashboard.

Activity Diagram *(Student Data Encryption)*



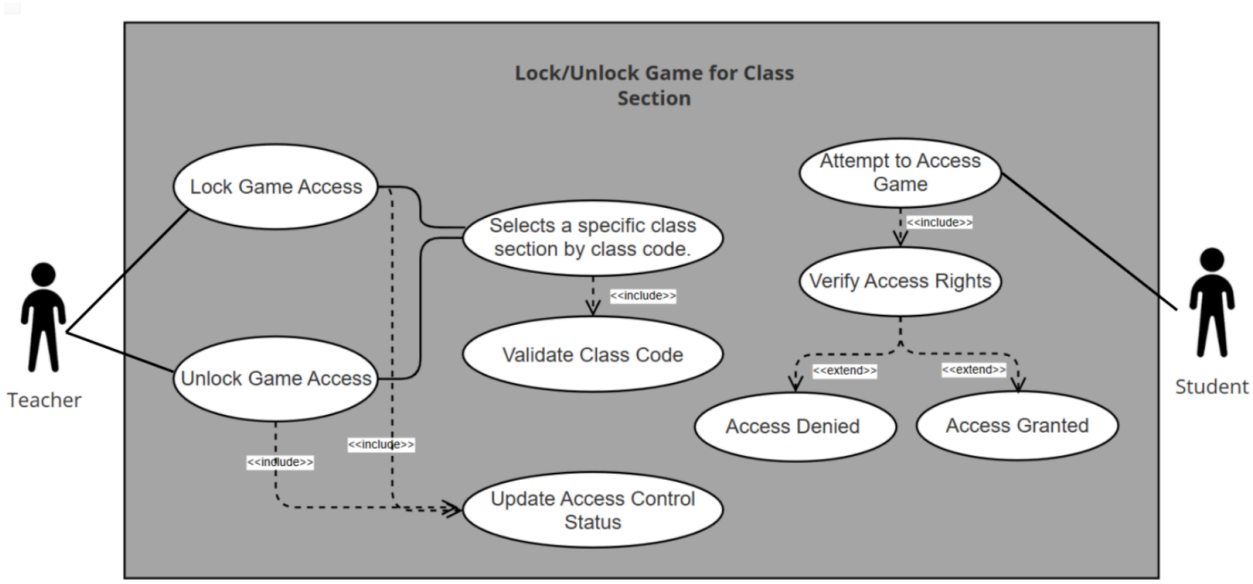
Wireframe *(Student Data Encryption)*



Module 3: Teachers Dashboard

Transaction 3.1 Lock/Unlock Game for Class Section

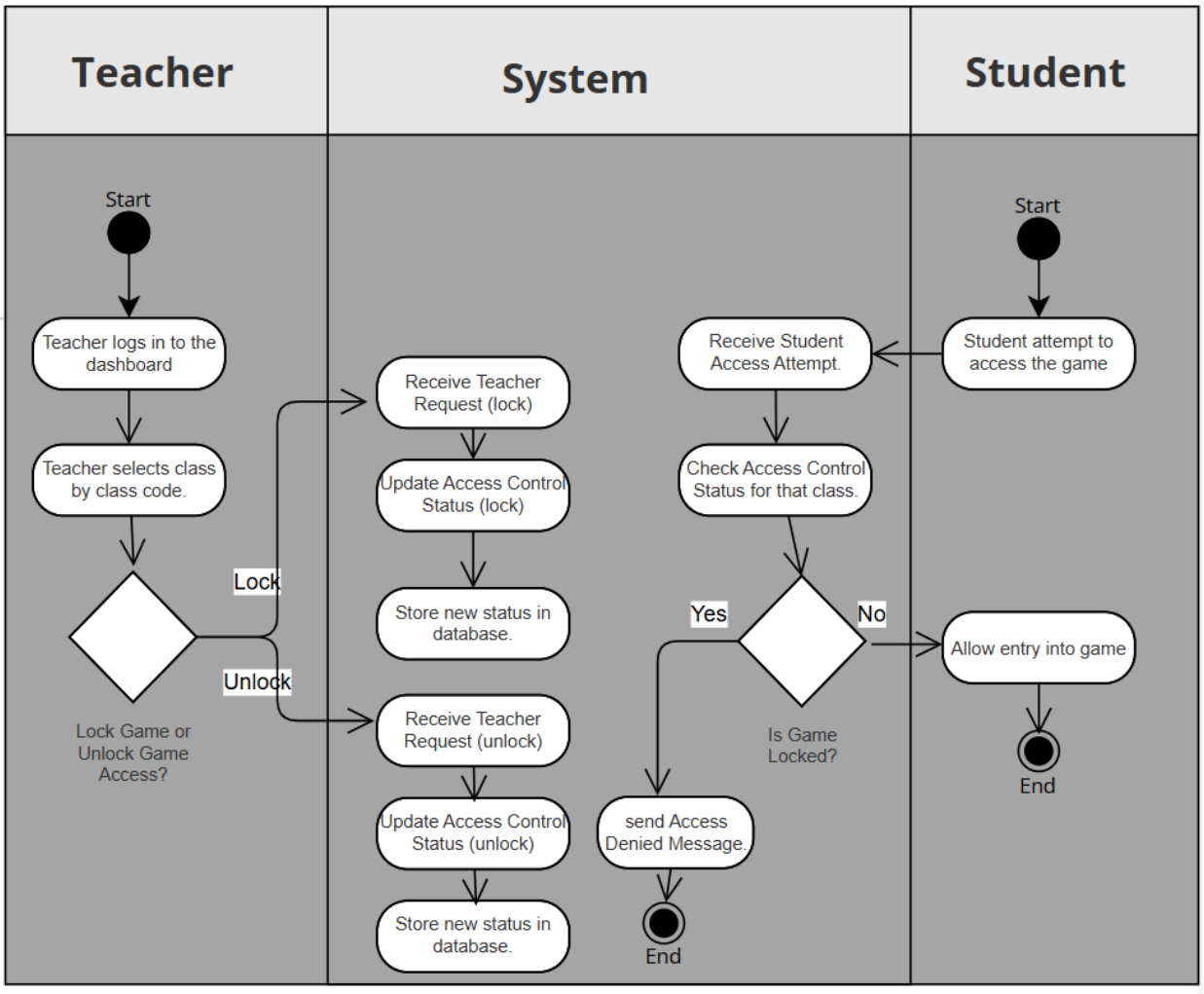
***Use Case Diagram** (Lock/Unlock Game for Class Section)*



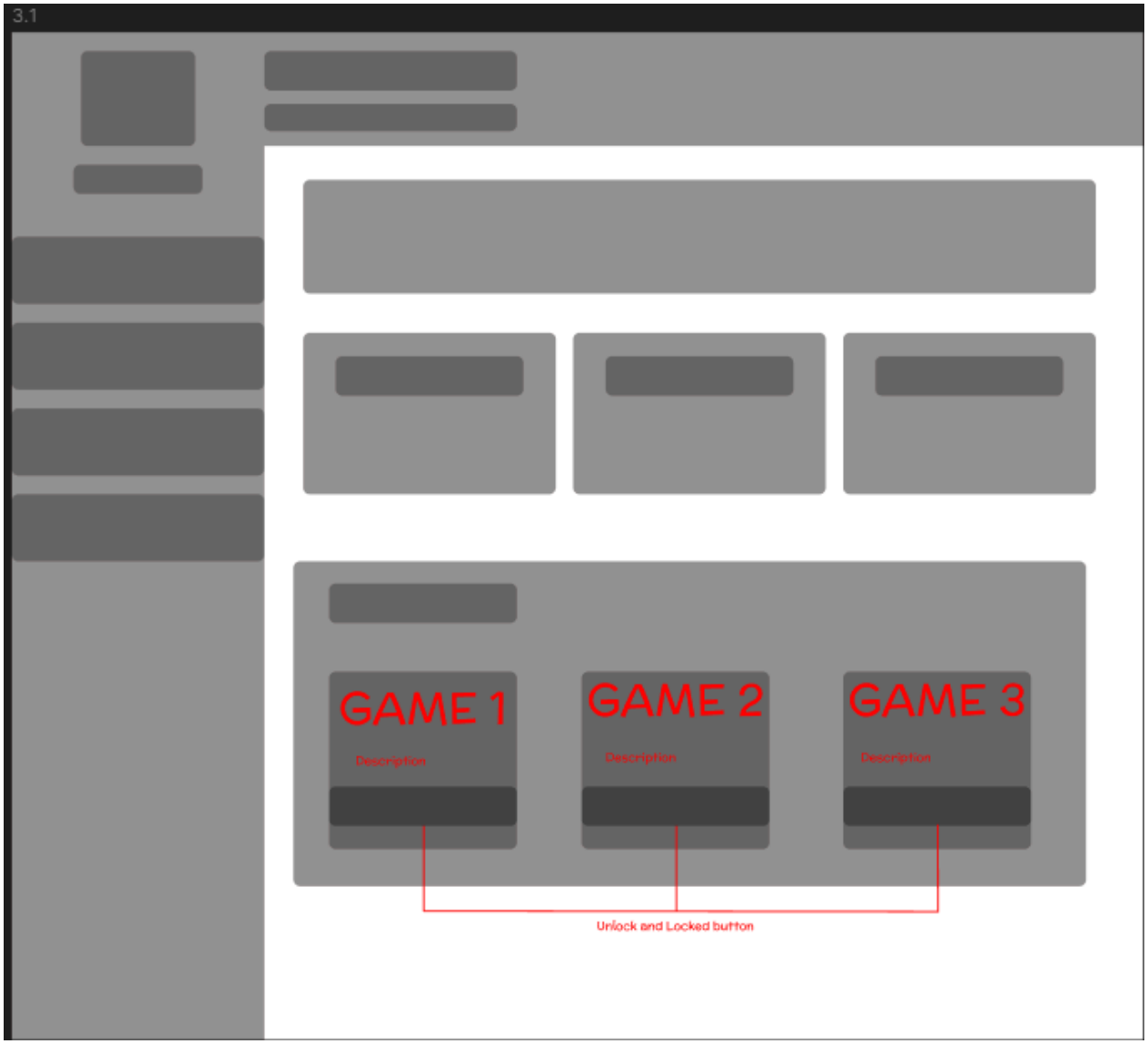
Use Case Description

Use Case ID	UC-001
Use Case Name	*Lock/Unlock Game for Class Section*
Actor	Teacher, Student
Description	Allow teachers to lock or unlock the game for a specific class section by class code, restricting student access accordingly.
Flow of Events	1. Teacher selects “Lock Game” / “Unlock Game” option. 2. Teacher enters/selects class code for the section. 3. System validates the class code. 4. System marks the section as locked/unlocked. 5. Students in the section are prevented from accessing the game until unlocked. 6. System displays confirmation to teacher.
Precondition	● Teacher is logged into the system. ● Class code for the section has been created and assigned to students.
Postcondition	● If the game is **locked**: Students in that section cannot access the game. ● If the game is **unlocked**: Only students in the designated section can access, others are restricted.

Activity Diagram *(Lock/Unlock Game for Class Section)*

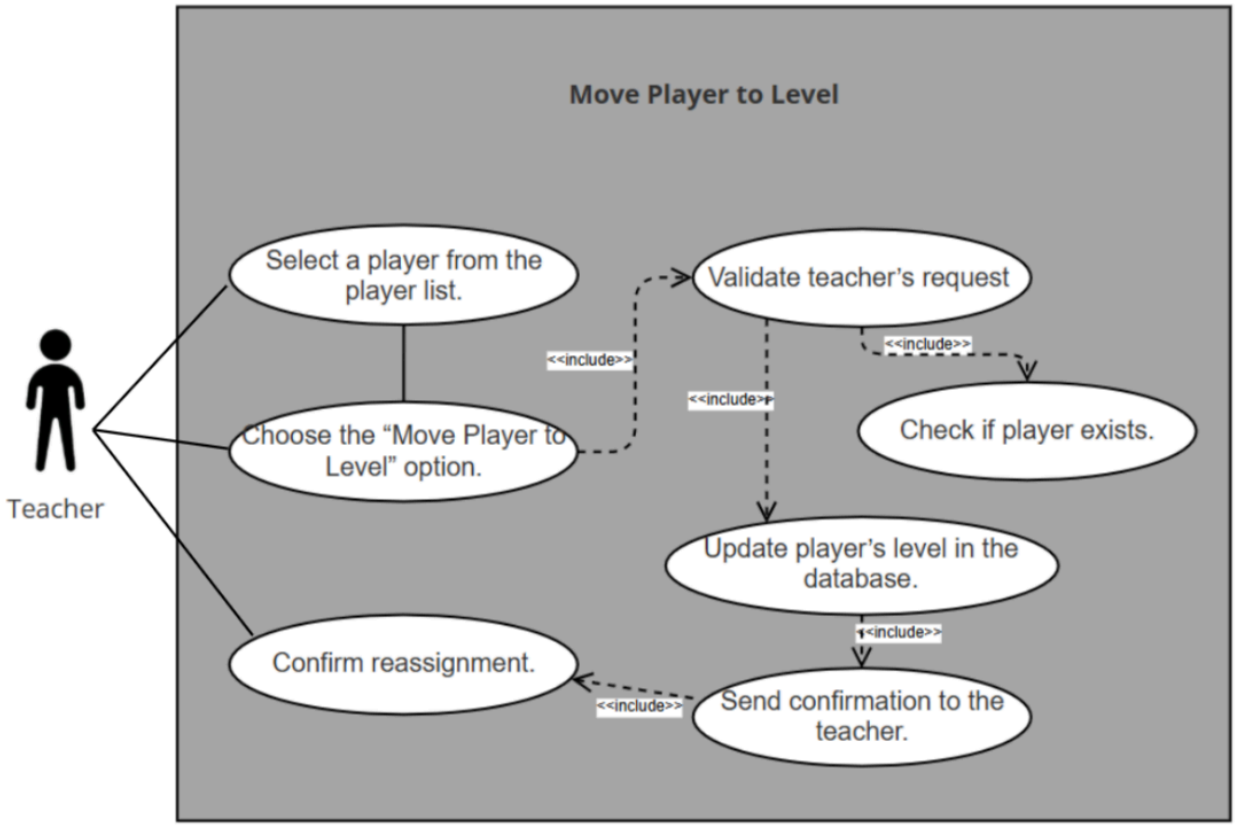


Wireframe *(Lock/Unlock Game for Class Section)*



Transaction 3.2 Move Player to Level

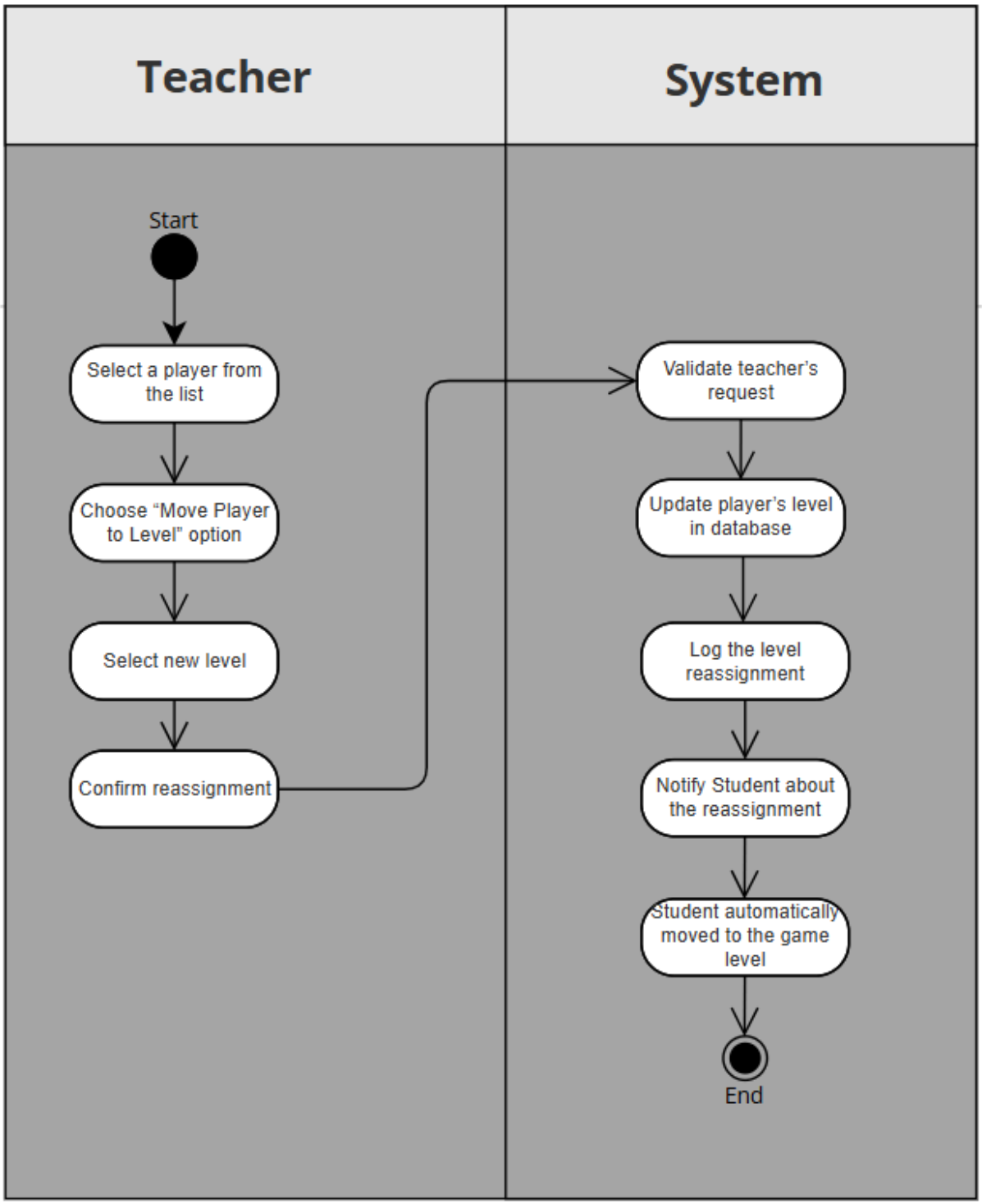
Use Case Diagram *(Move Player to Level)*



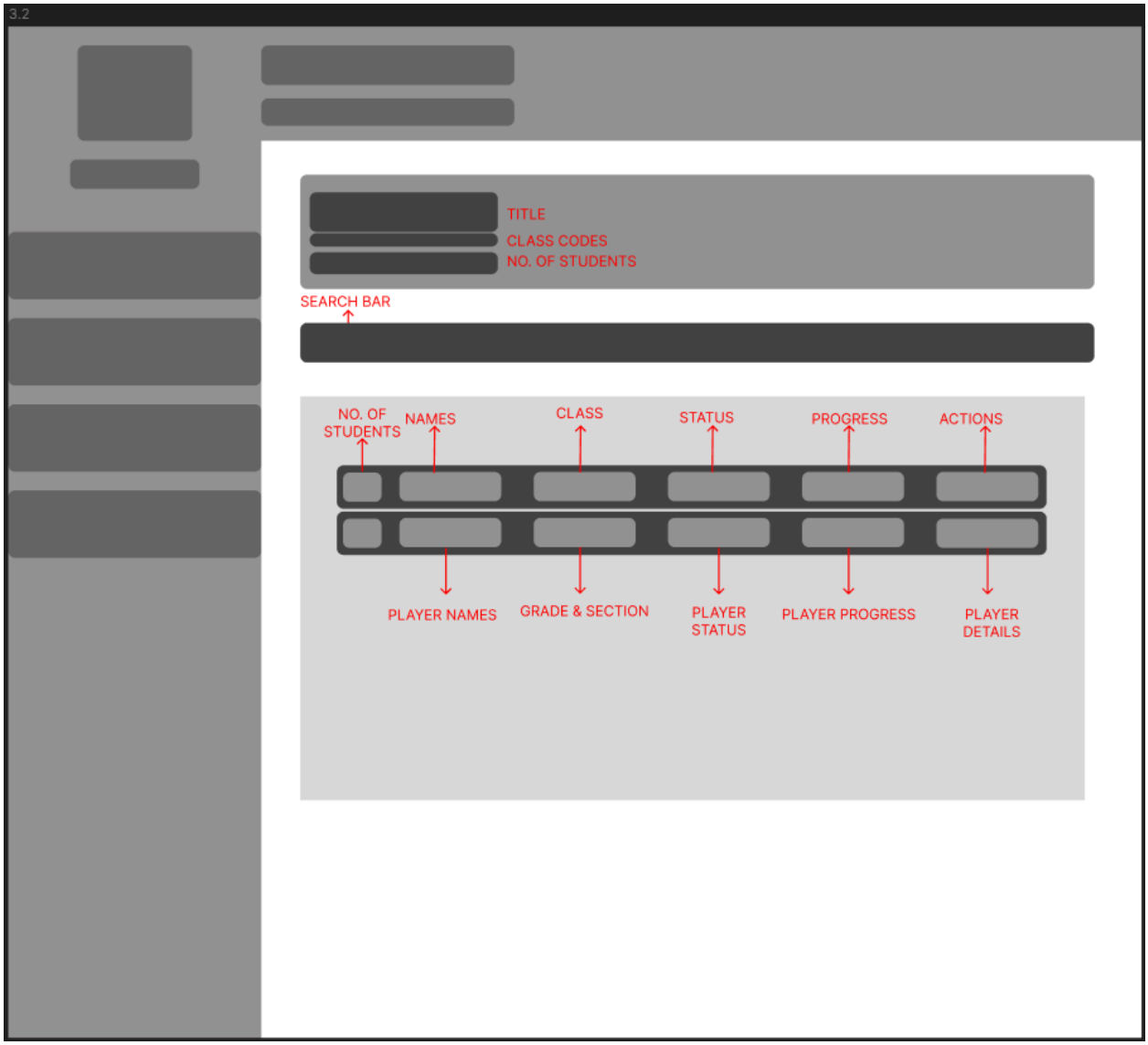
Use Case Description

Use Case ID	UC-002
Use Case Name	*Move Player to Level*
Actor	Teacher
Description	Allow teachers to lock or unlock the game for a specific class section by class code, restricting student access accordingly.
Flow of Events	1. Teacher navigates to the dashboard and selects a specific player. 2. Teacher chooses the “Reassign Level” option. 3. System retrieves the current level and available levels. 4. Teacher selects the new desired level. 5. Teacher confirms the reassignment. 6. System validates the reassignment (e.g., level exists, reassignment is allowed). 7. System updates the player’s level in the database. 8. System notifies the player and updates the teacher’s dashboard.
Precondition	● Teacher is logged into the system. ● Player is already registered and currently assigned to an existing level.
Postcondition	● Player’s level assignment is updated successfully. ● The new level assignment is immediately reflected in the teacher’s dashboard. ● The player is notified and placed in the reassigned game level in real time.

Activity Diagram *(Move Player to Level)*

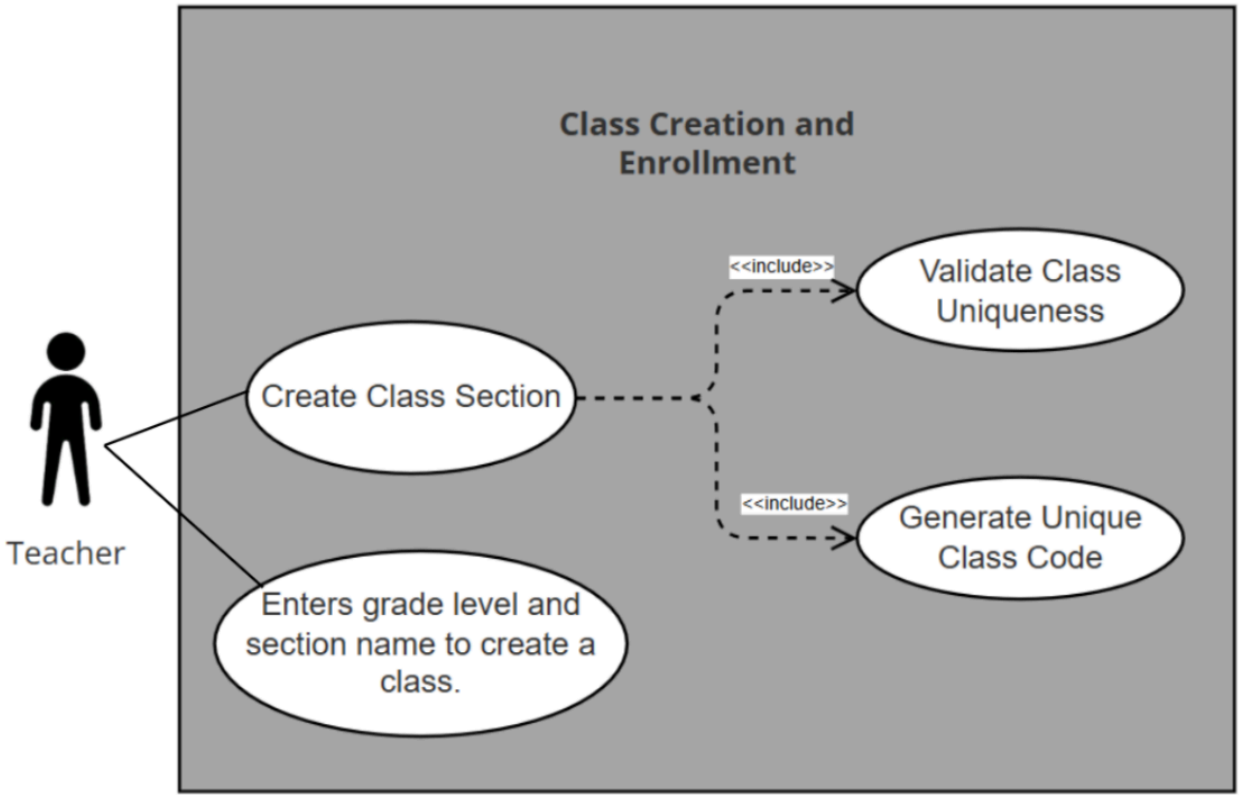


***Wireframe** (Move Player to Level)*



Transaction 3.3 Class Creation and Enrollment

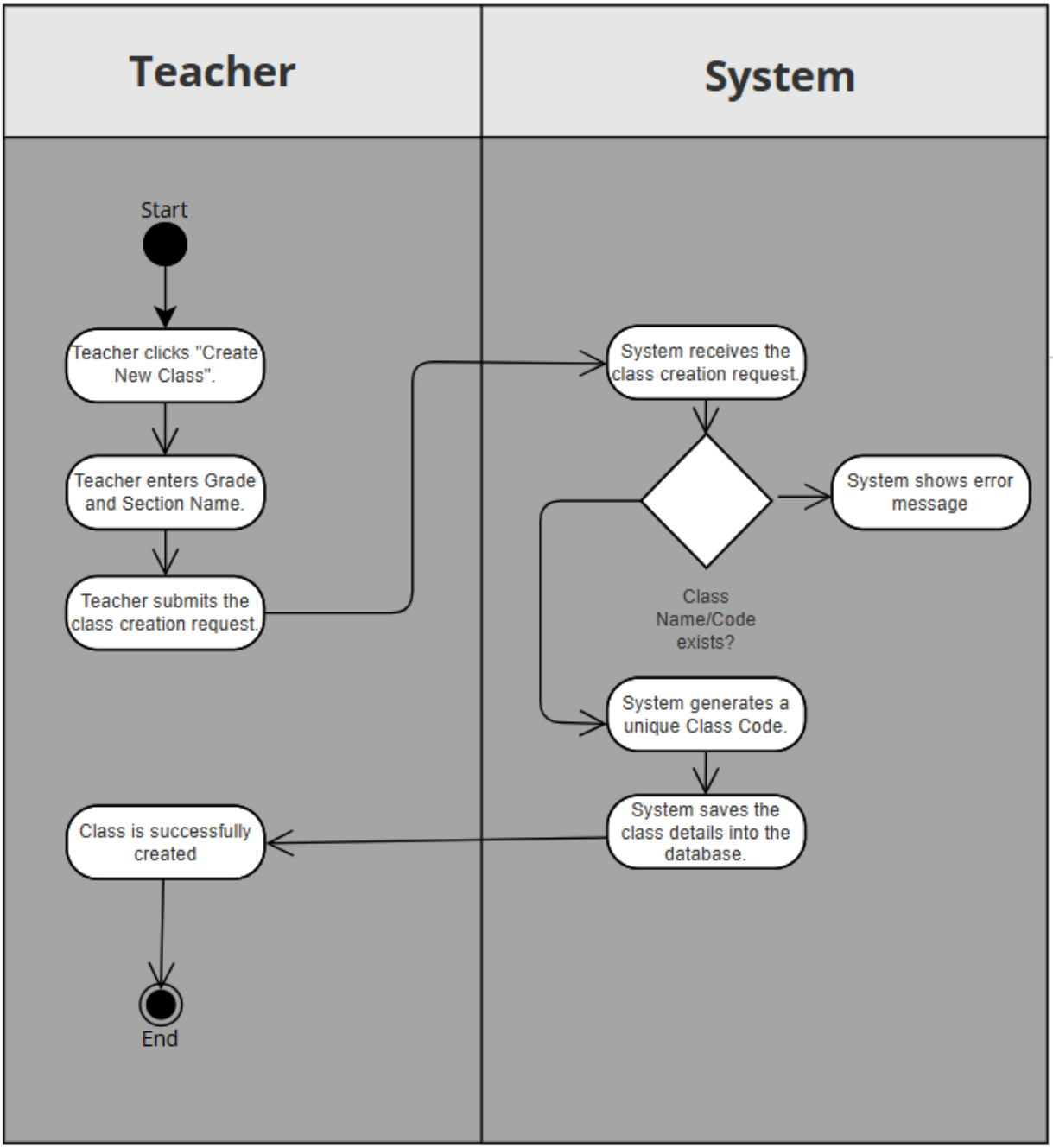
Use Case Diagram** (Class Creation and Enrollment**)



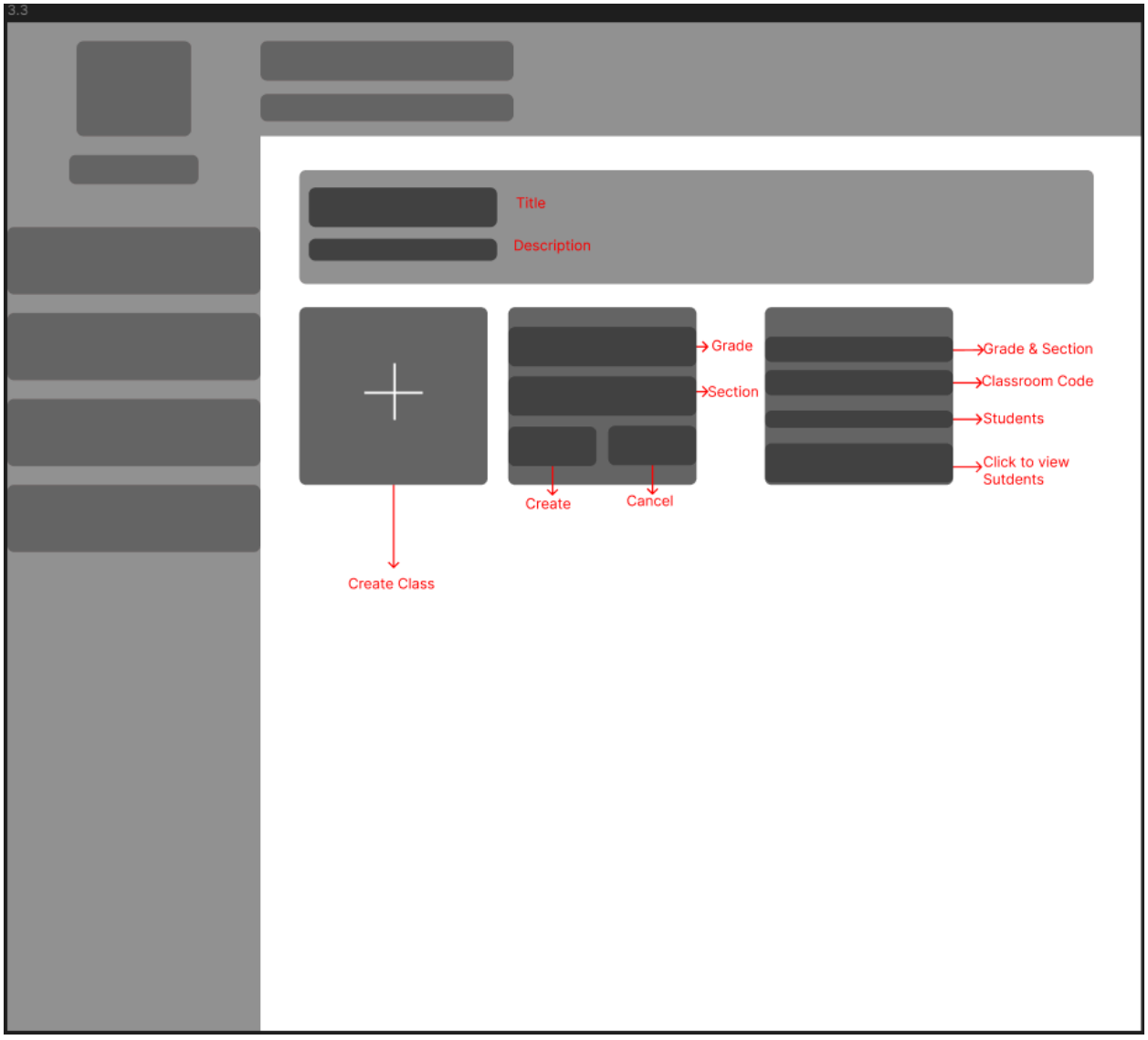
Use Case Description

Use Case ID	UC-003
Use Case Name	*Class Creation and Enrollment*
Actor	Teacher
Description	This use case allows teachers to create class sections by assigning a grade and section name. The system ensures each class name and code is unique to prevent duplicate classes.
Flow of Events	1. Teacher selects the option “Create Class Section.” 2. Teacher enters the grade level and section name. 3. System generates a unique class code for the new section. 4. System checks if the class name and code are unique. 5. If unique, system stores the class information in the database. 6. System displays a confirmation message: “Class section created successfully.”
Precondition	The teacher must be logged into the system.
Postcondition	● A new class section is successfully created and stored in the system. ● Each class has a unique identifier (name and code). ● The teacher and students can later enroll in or be assigned to the class.

Activity Diagram *(Class Creation and Enrollment)*

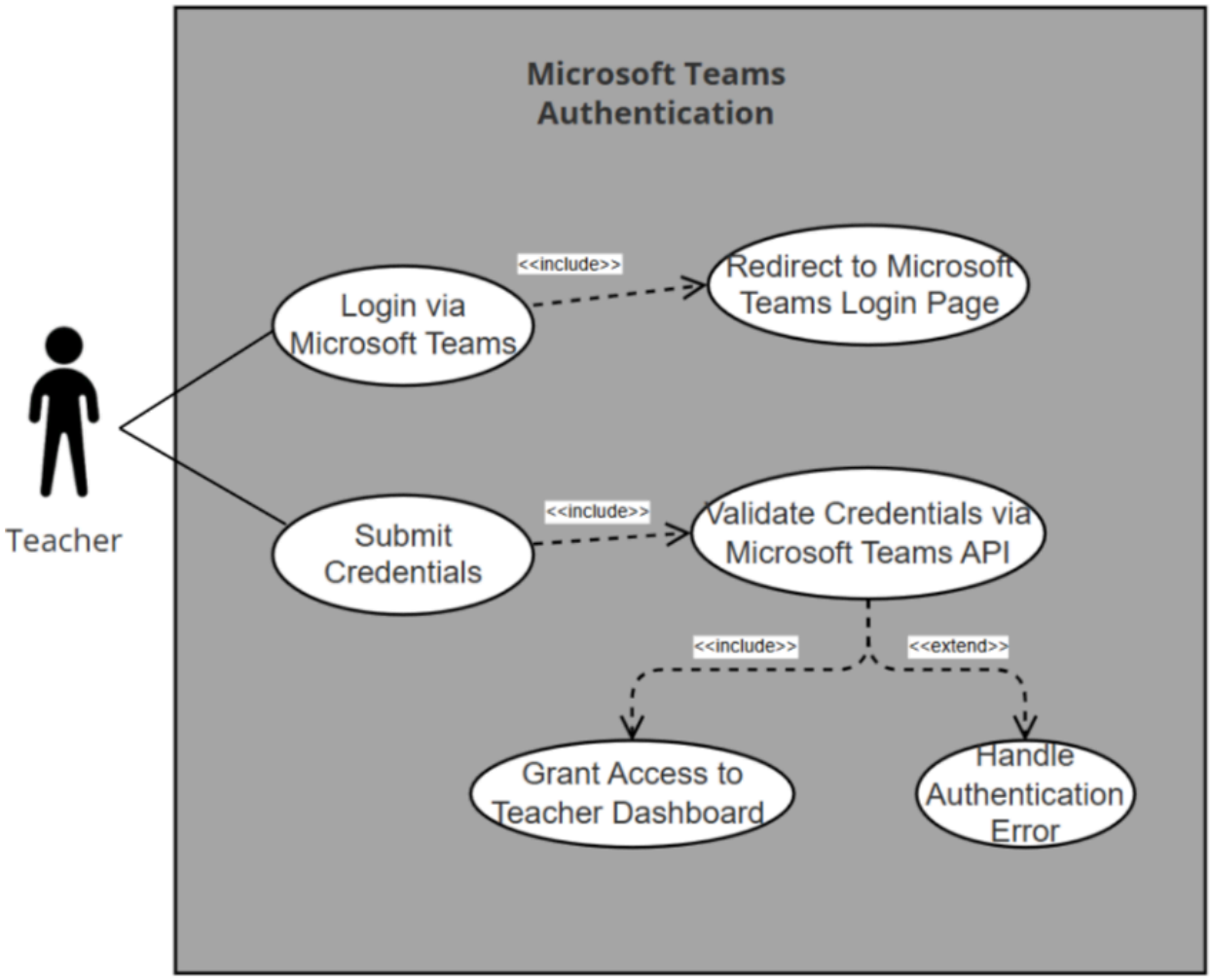


Wireframe *(Class Creation and Enrollment)*



Transaction 3.4 Microsoft Teams Authentication

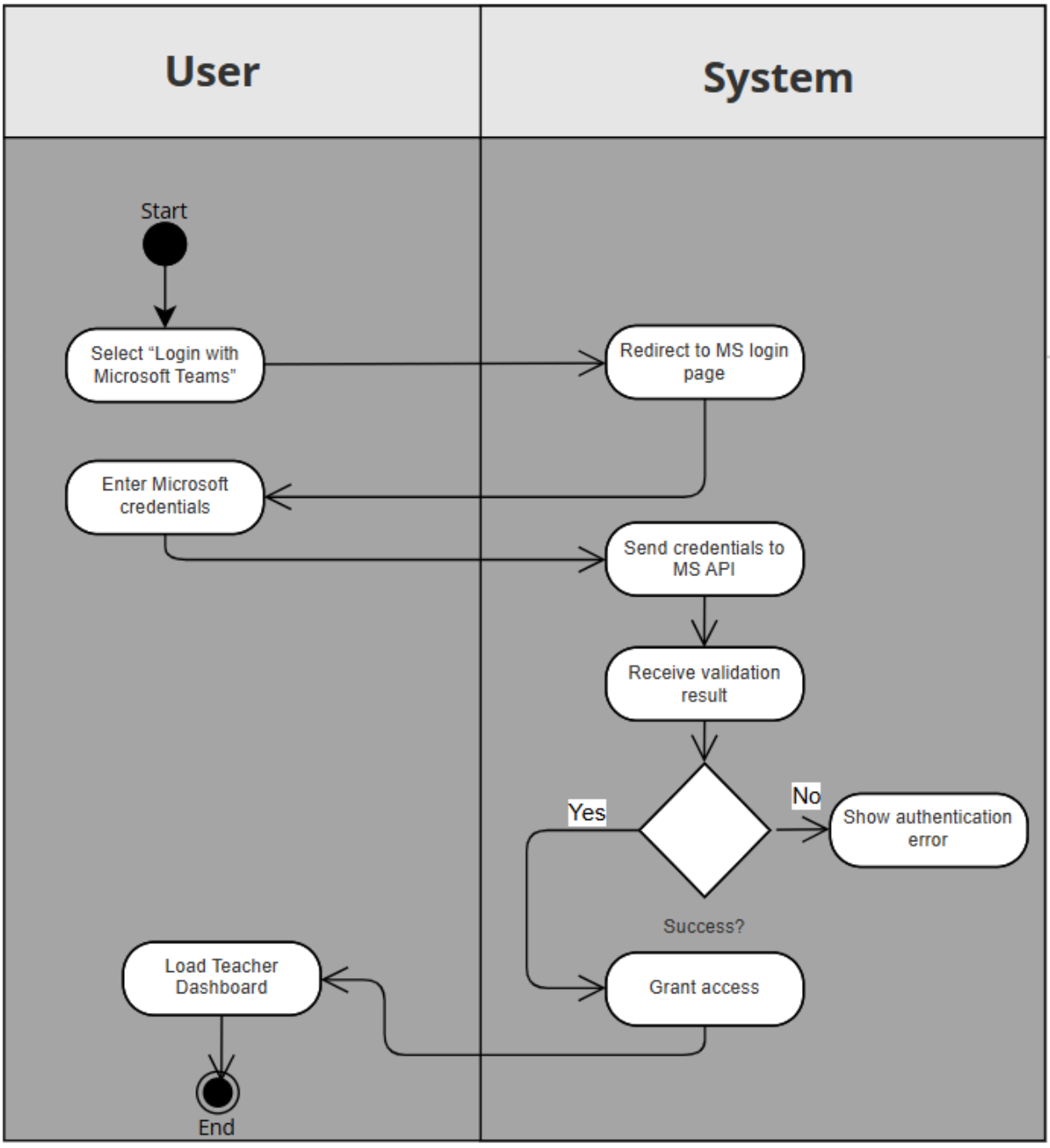
Use Case Diagram *(Microsoft Teams Authentication)*



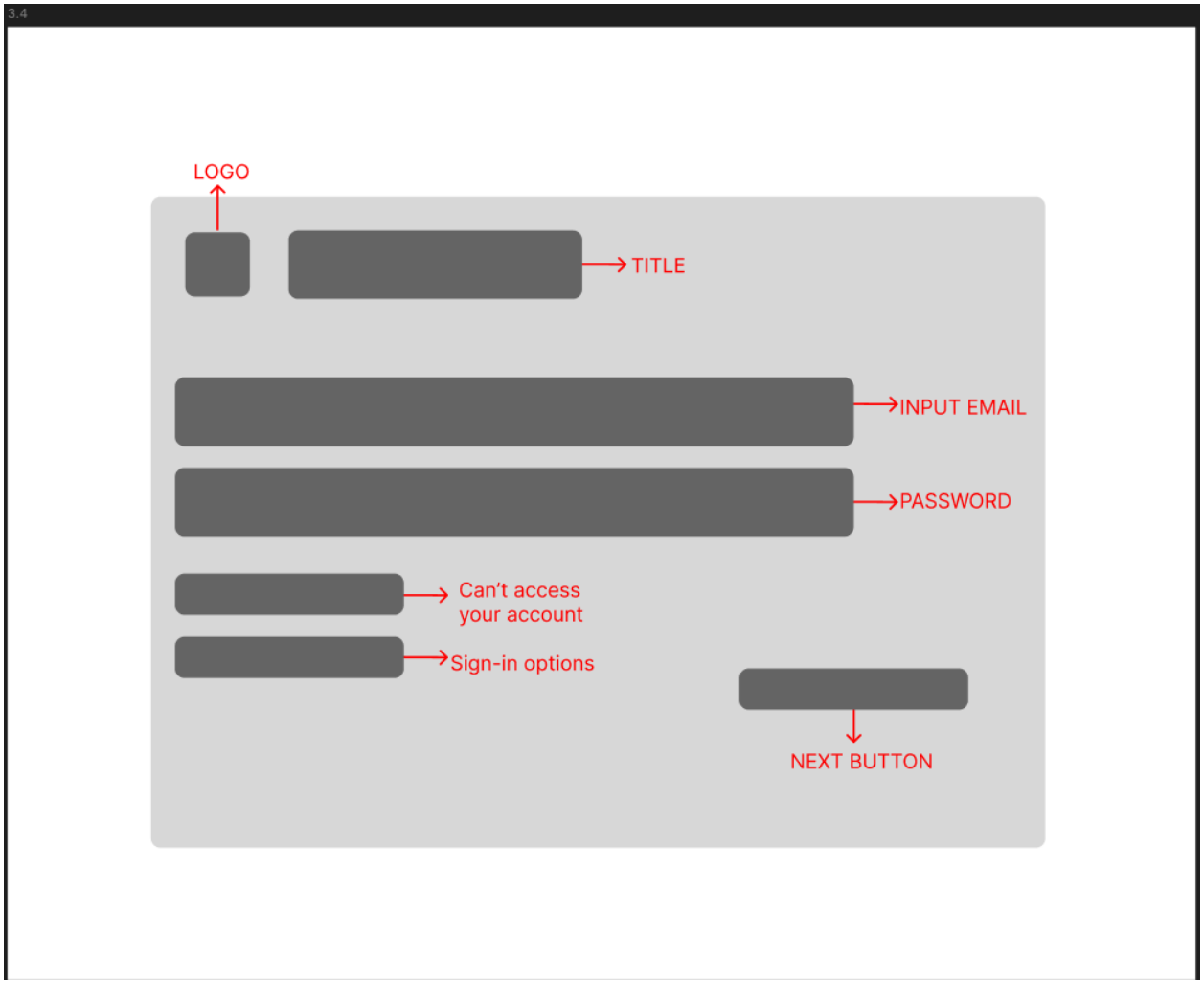
Use Case Description

Use Case ID	UC-004
Use Case Name	*Microsoft Teams Authentication*
Actor	Teacher
Description	This use case allows a student or teacher to log in to the system using their Microsoft Teams account credentials. The system communicates with Microsoft's authentication service to validate the login request and grants access if credentials are valid.
Flow of Events	1. User selects **"Login with Microsoft Teams"** option. 2. System redirects user to Microsoft Teams authentication service. 3. User enters their Microsoft Teams credentials (username & password). 4. Microsoft validates credentials. 5. If valid, Microsoft returns an **authentication token** to the system. 6. System uses token to retrieve user's basic profile information. 7. System creates a session and grants access. 8. User is redirected to their **Dashboard**.
Precondition	User has a valid Microsoft Teams account (username & password).
Postcondition	● User is logged into the system. ● User profile is retrieved from Microsoft Teams (basic info like name, email). ● Session is created and access is granted.

Activity Diagram *(Microsoft Teams Authentication)*

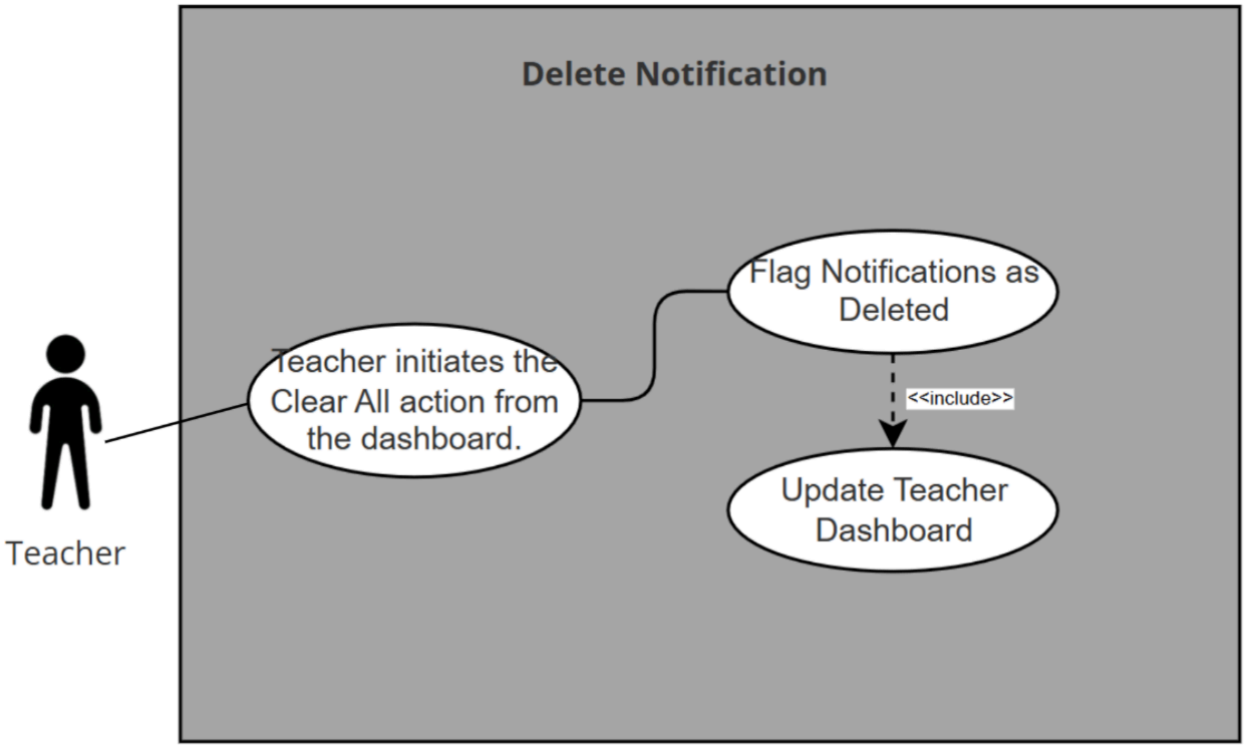


Wireframe *(Microsoft Teams Authentication)*



Transaction 4.1 Delete Notification

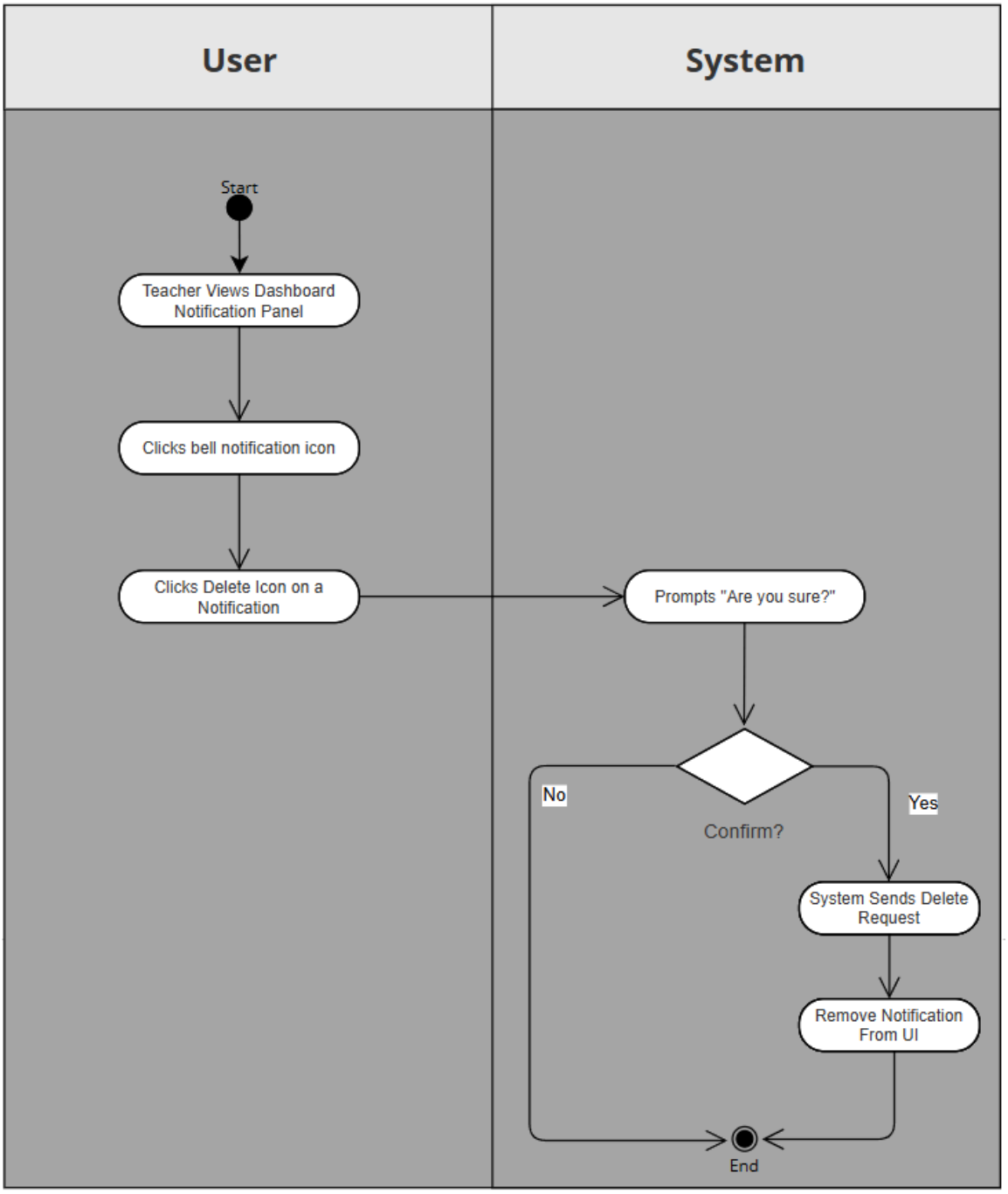
***Use Case Diagram** (Delete Notification)*



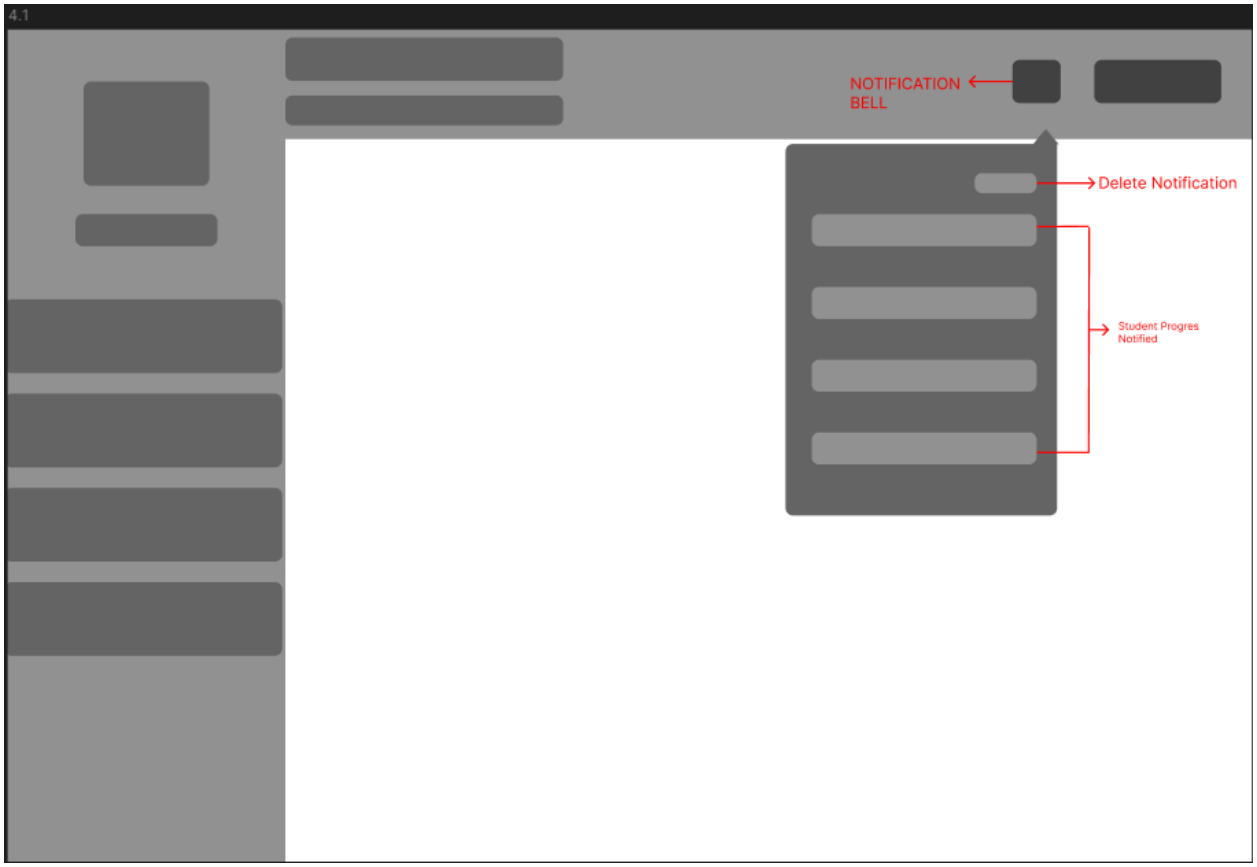
Use Case Description

Use Case ID	UC-001
Use Case Name	*Delete Notification*
Actor	Teacher
Description	This use case allows the teacher to clear all displayed real-time notifications from their dashboard using the “Clear All” function. The notifications are removed from the teacher’s view but are not permanently deleted from the database; instead, they are flagged as cleared to maintain historical records.
Flow of Events	1. Teacher selects the **“Clear All”** option from the notification panel. 2. System prompts confirmation (e.g., “Are you sure you want to clear all notifications?”). 3. Teacher confirms the action. 4. System flags all retrieved notifications as **“cleared”** in the database. 5. System removes the notifications from the teacher’s dashboard view. 6. Teacher sees an updated clean dashboard with no notifications displayed.
Precondition	● The teacher is logged into the system. ● Notifications exist and are currently displayed on the teacher’s dashboard.
Postcondition	● Notifications are no longer displayed on the teacher’s dashboard. ● All cleared notifications remain flagged in the database (not permanently deleted).

Activity Diagram *(Delete Notification)*

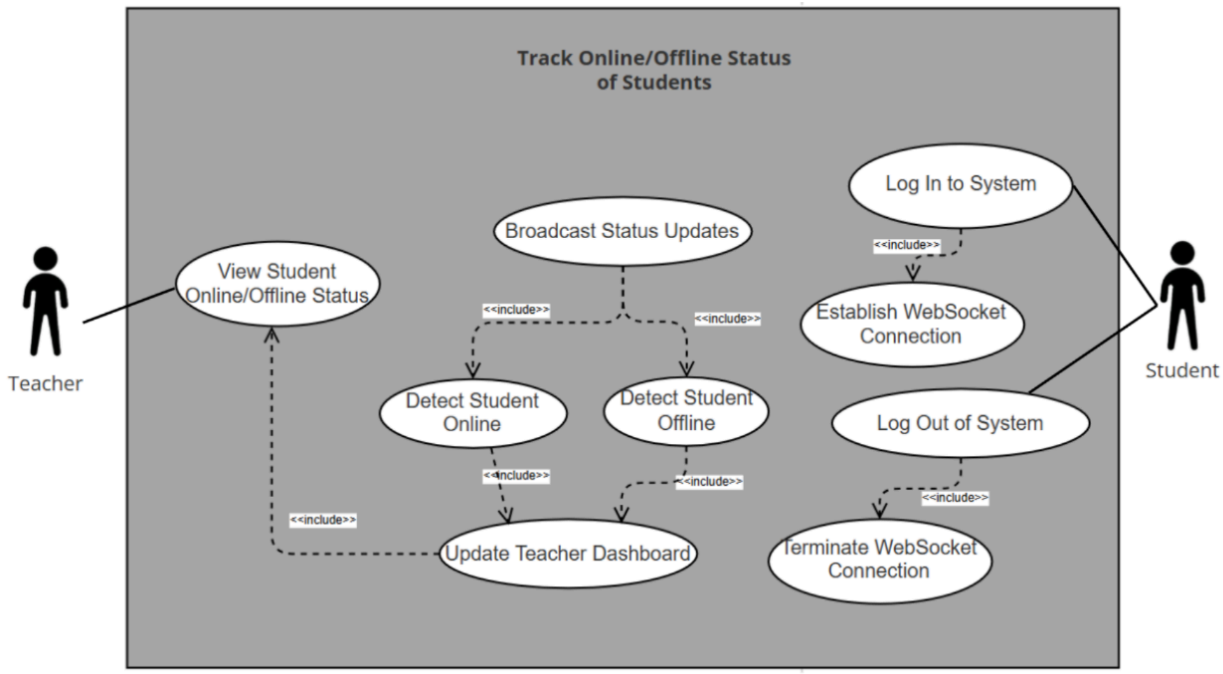


***Wireframe** (Delete Notification)*



Transaction 4.2 Track Online/Offline Status of Students

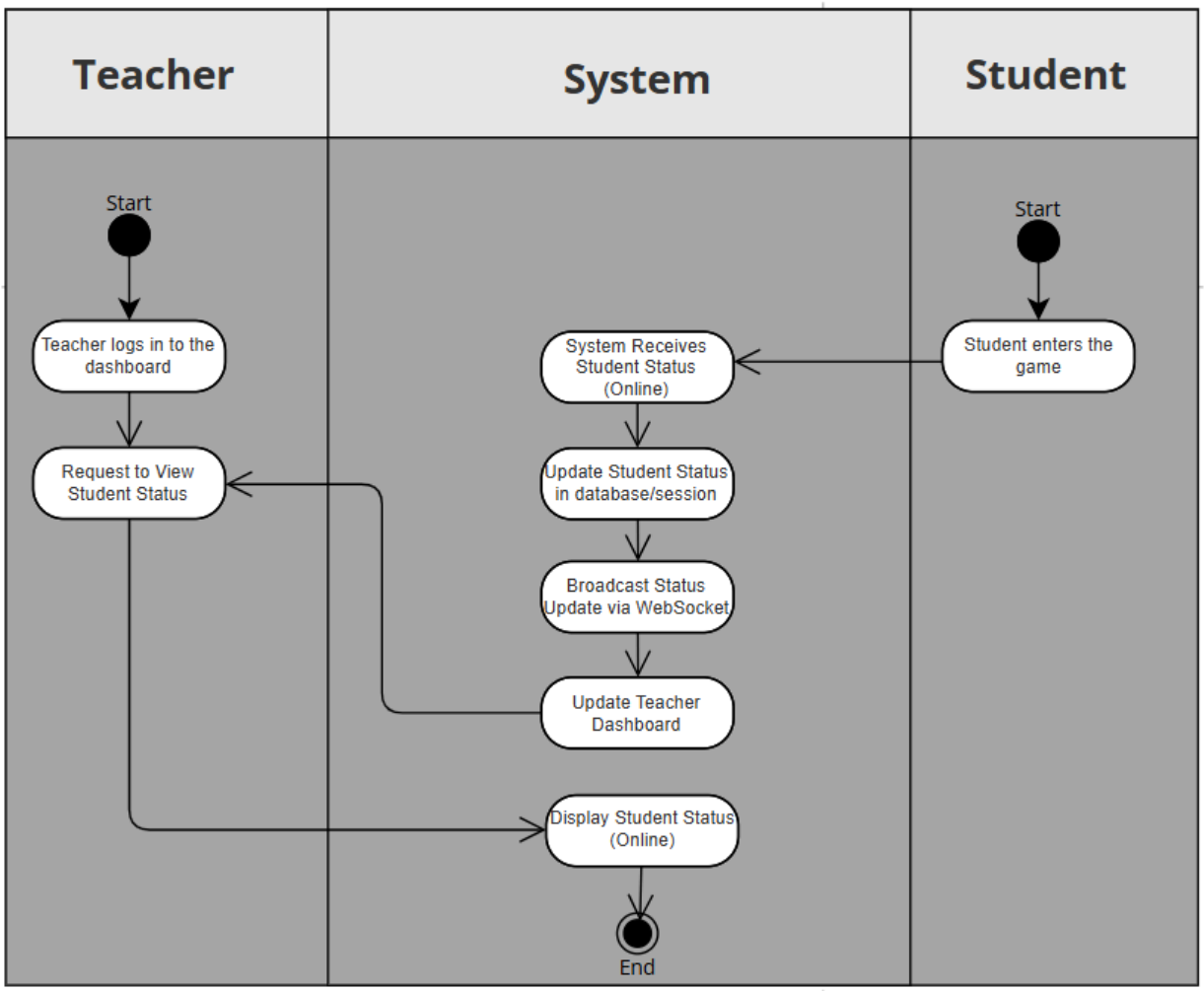
Use Case Diagram (*Track Online/Offline Status of Students)*



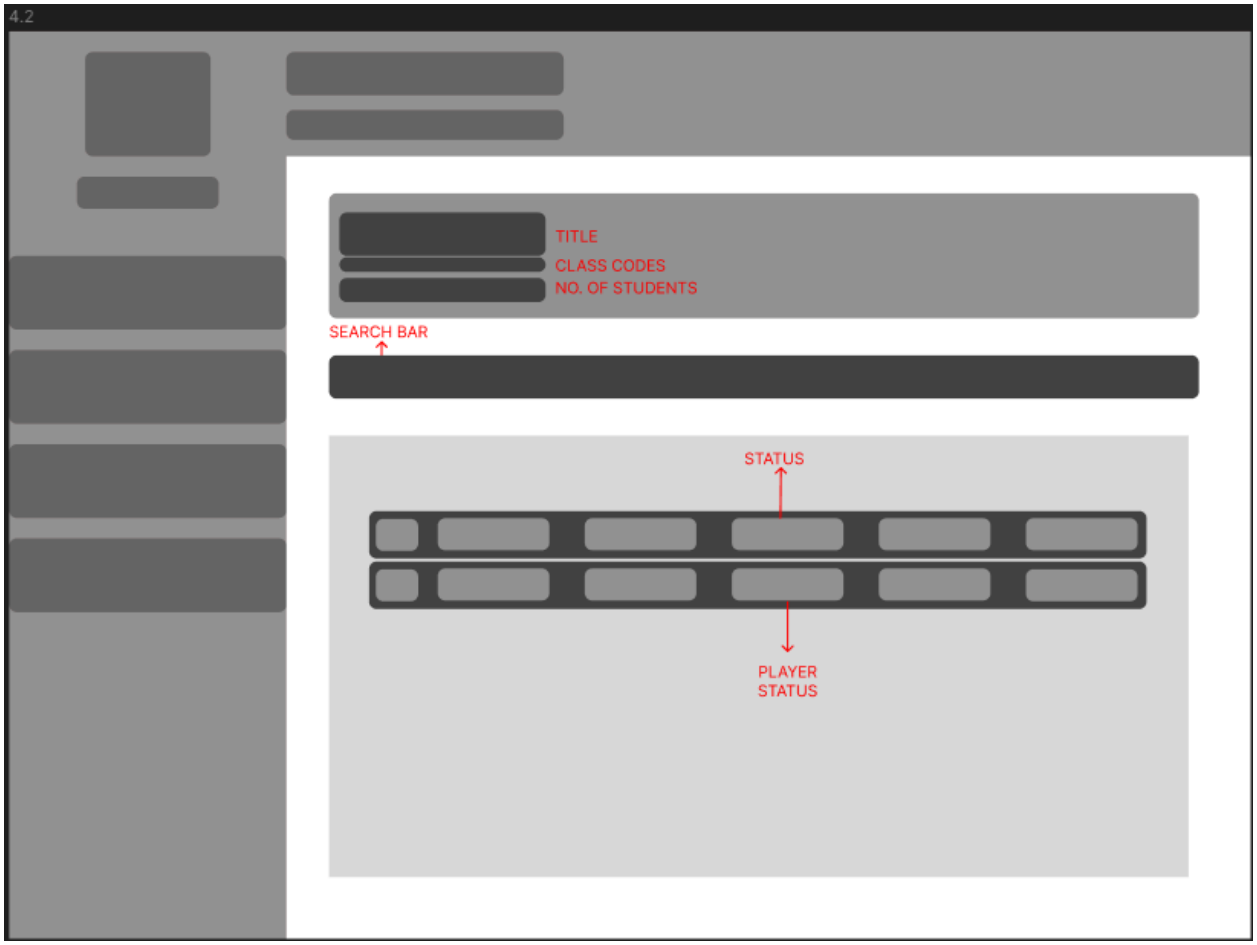
Use Case Description

Use Case ID	UC-002
Use Case Name	*Track Online/Offline Status of Students*
Actor	Teacher, Student
Description	The system provides real-time visibility of students' online/offline status through WebSocket connections. When a student logs in or out, the teacher's dashboard automatically updates without needing a page refresh.
Flow of Events	1. Teacher logs into the dashboard. 2. The system establishes a WebSocket connection to track real-time student activity. 3. Students log in/out of the application. 4. The system detects changes in student connection status (online/offline). 5. The system updates the teacher's dashboard instantly to reflect the current status of each student. 6. Teacher views the updated student presence information on the dashboard.
Precondition	● Teacher must be logged into the dashboard. ● Students must have valid accounts in the system.
Postcondition	Teacher can view an up-to-date list of students' online/offline status in real-time.

Activity Diagram (*Track Online/Offline Status of Students)*

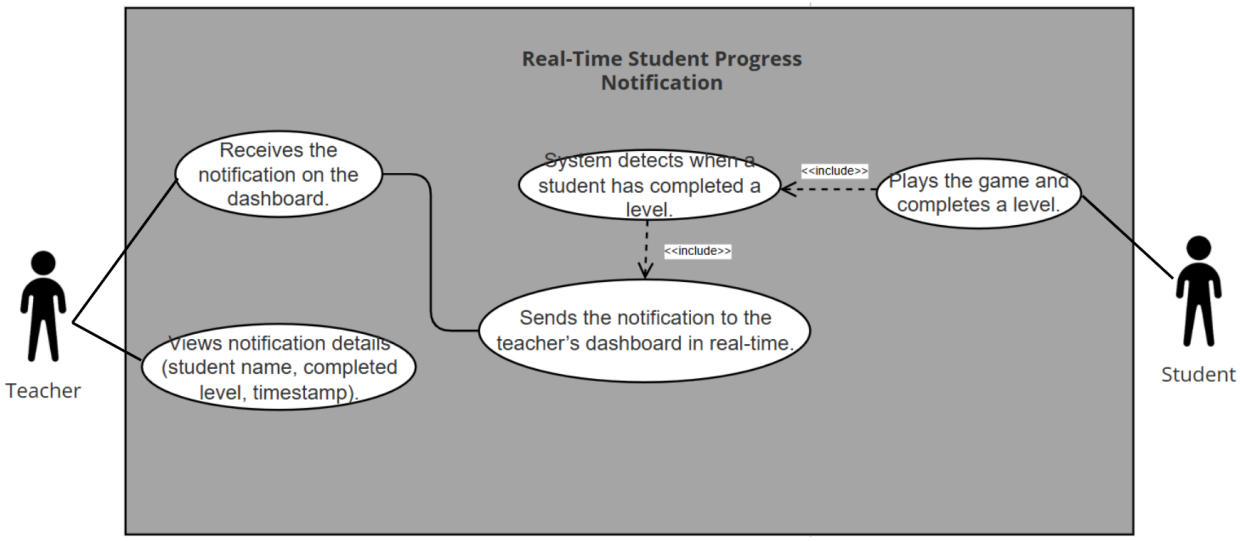


Wireframe (*Track Online/Offline Status of Students)*



Transaction 4.3 Real-Time Student Progress Notification

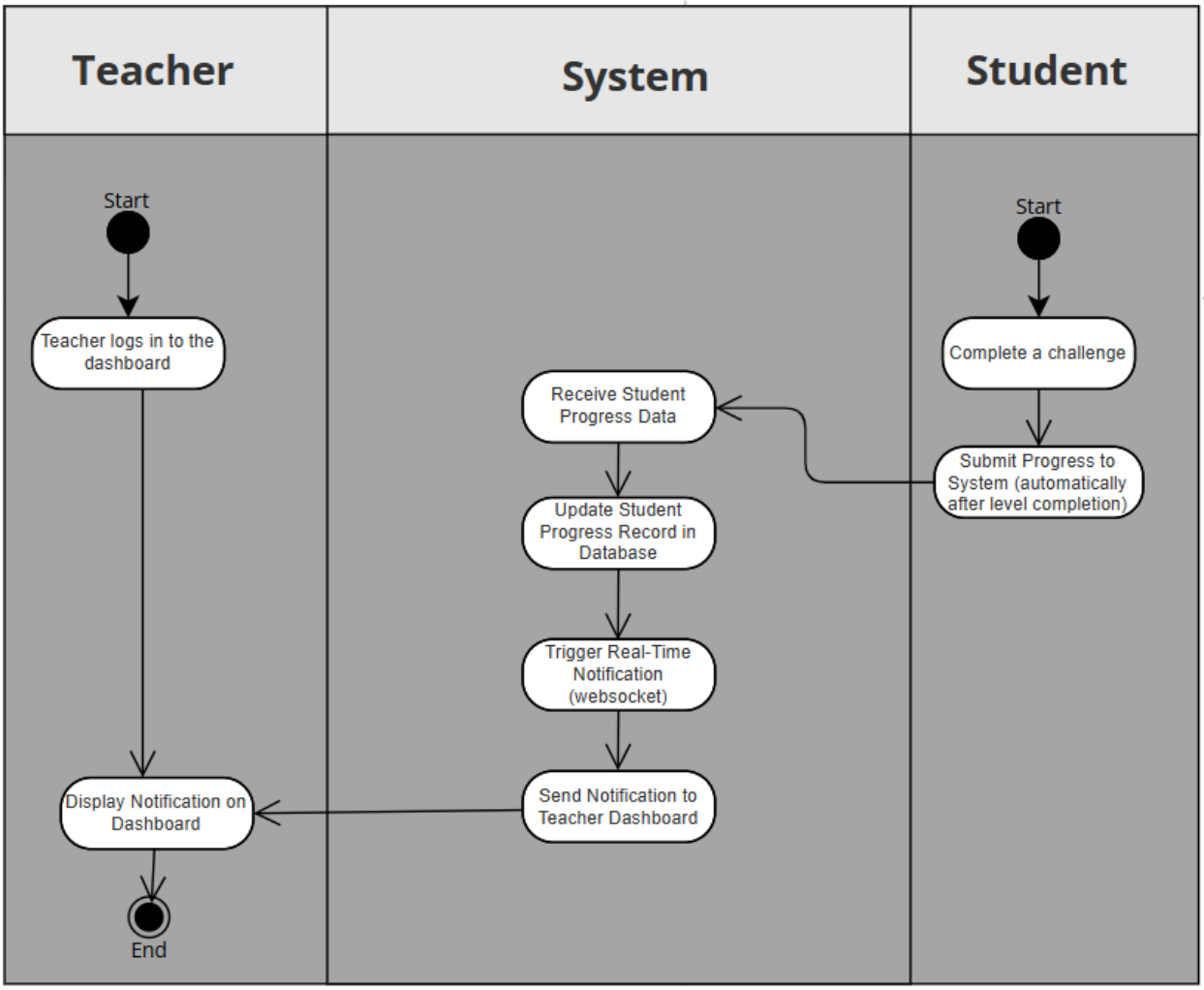
***Use Case Diagram** (Real-Time Student Progress Notification)*



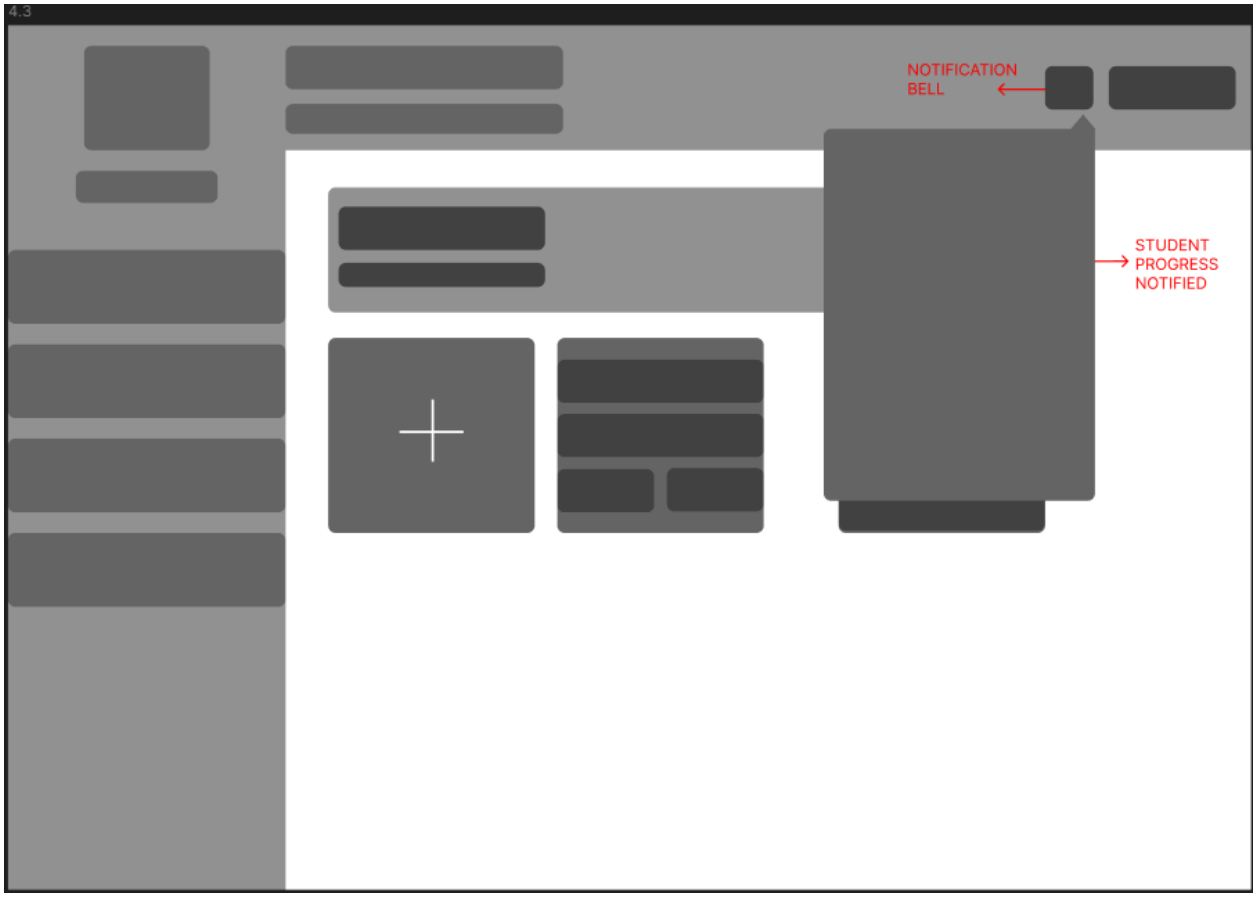
Use Case Description

Use Case ID	UC-003
Use Case Name	*Real-Time Student Progress Notification*
Actor	Teacher, Student
Description	This use case describes how the system notifies the teacher in real-time whenever a student completes a level. The notification is instantly displayed on the teacher's dashboard, enabling teachers to monitor and track student progress seamlessly without manual refresh or updates.
Flow of Events	1. The student completes a game level. 2. The system detects the level completion event. 3. The system validates the student's identity and class section. 4. The system generates a notification containing the student's name, class section, and completed level. 5. The system pushes the notification to the teacher's dashboard in real-time. 6. The teacher views the notification instantly on their dashboard.
Precondition	● The teacher must be logged into the system and have access to the teacher dashboard. ● The student must be actively playing the game within a valid class section.
Postcondition	A notification of the student's level completion is displayed on the teacher's dashboard.

Activity Diagram *(Real-Time Student Progress Notification)*

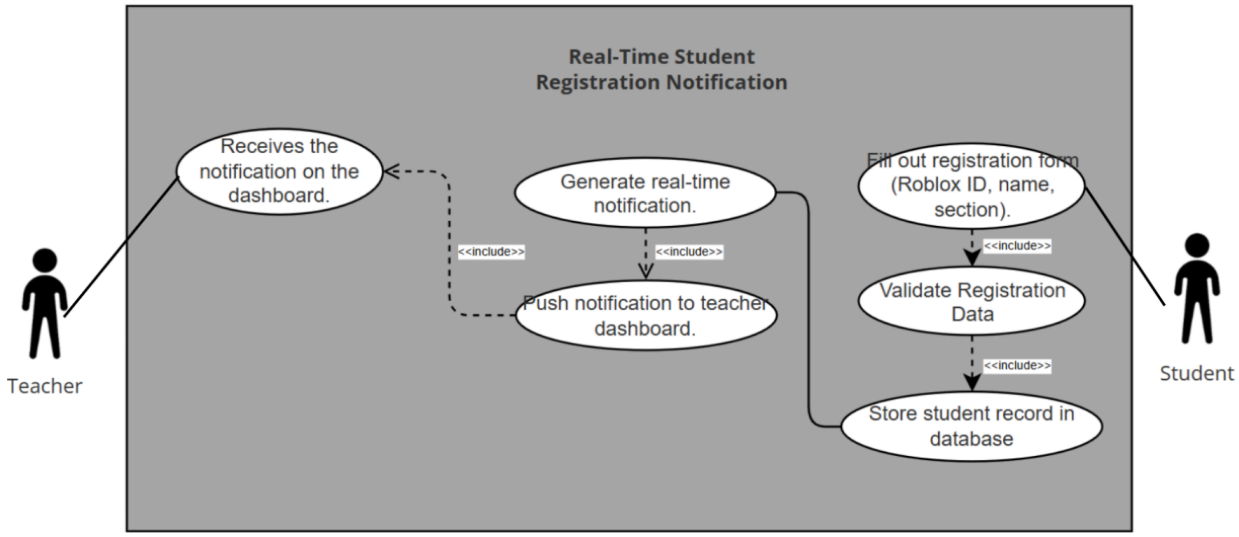


Wireframe *(Real-Time Student Progress Notification)*



Transaction 4.4 Real-Time Student Registration Notification

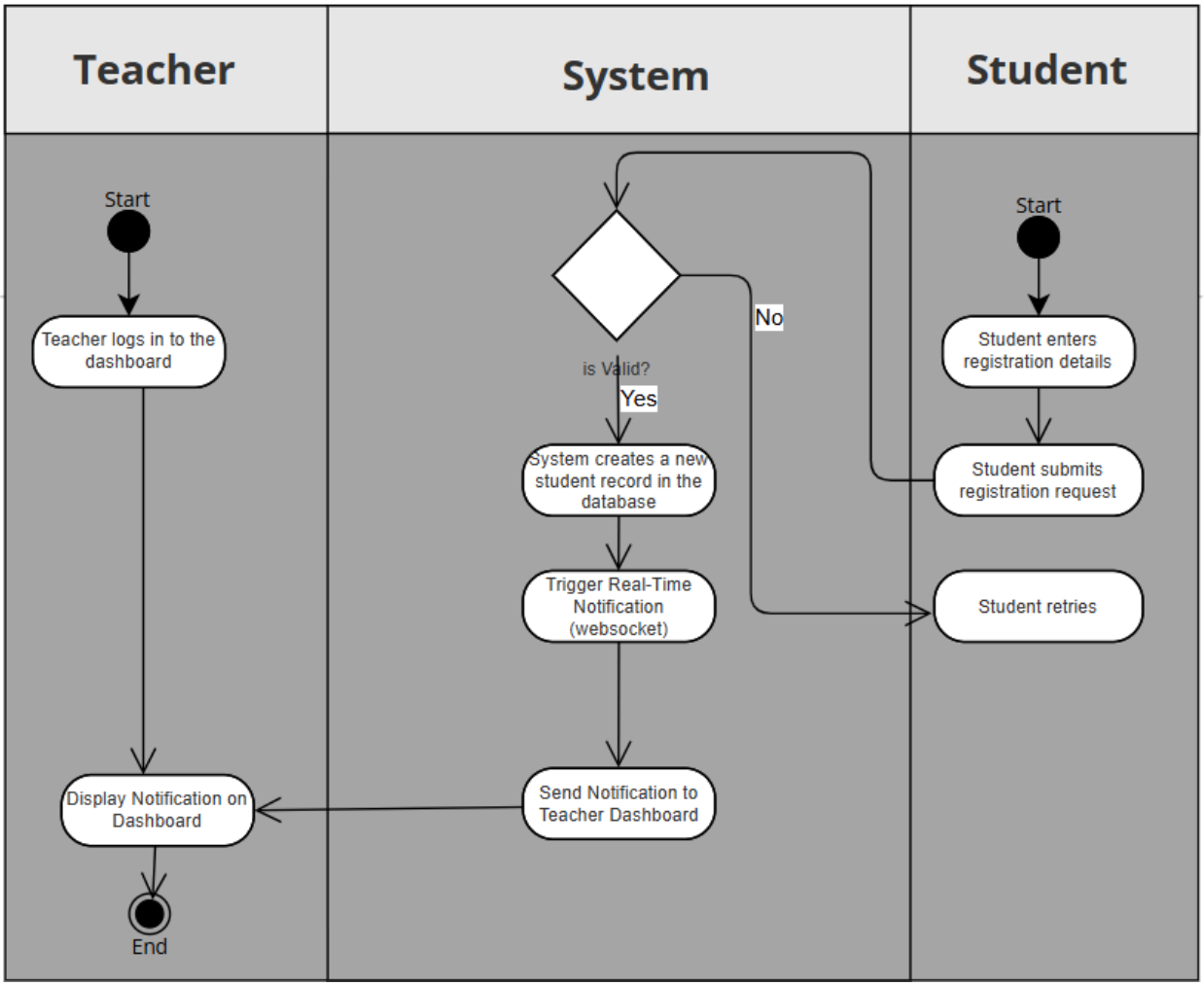
***Use Case Diagram** (Real-Time Student Registration Notification)*



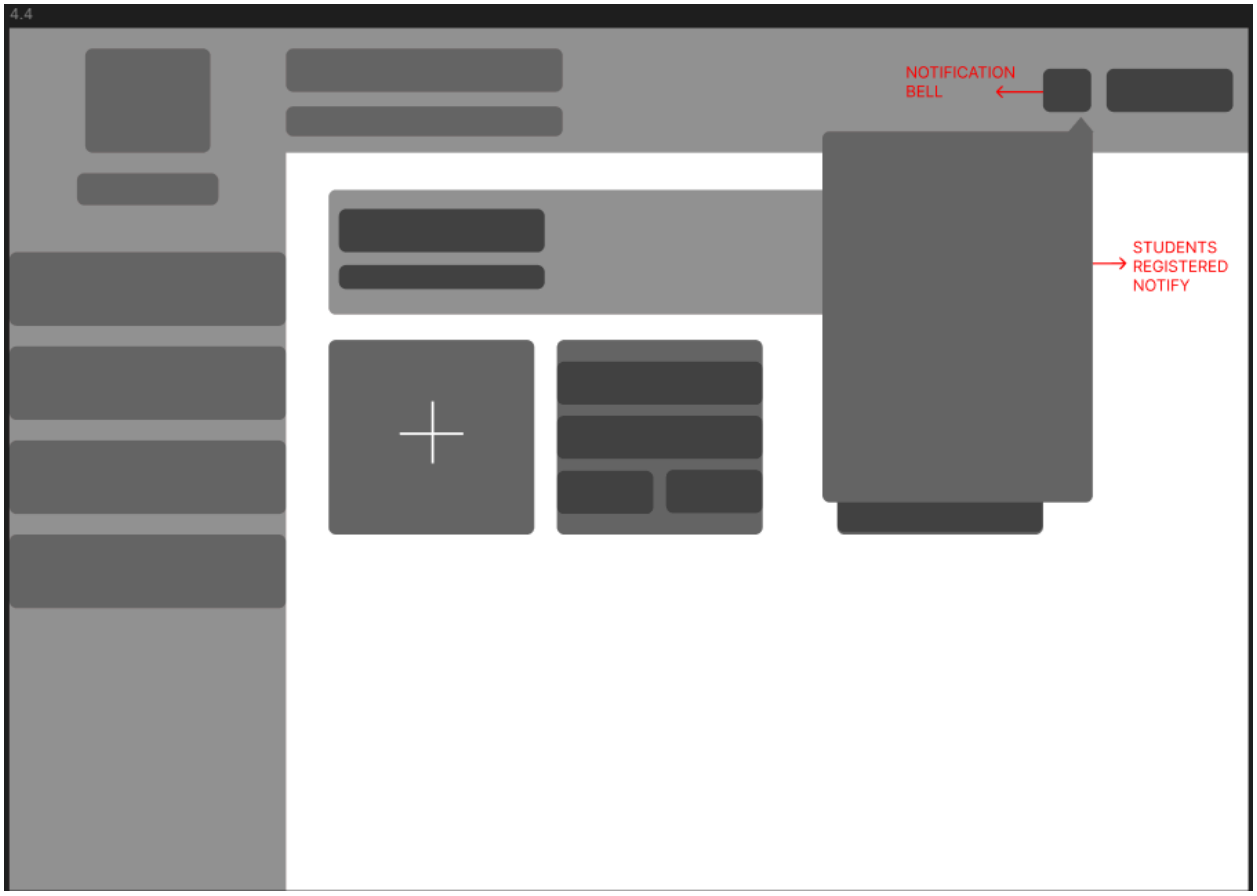
Use Case Description

Use Case ID	UC-004
Use Case Name	*Real-Time Student Registration Notification*
Actor	Teacher, Student
Description	This use case describes the process where a student registers in the system and the system automatically generates and delivers a real-time notification to the teacher's dashboard, ensuring the teacher is immediately aware of new student registrations.
Flow of Events	1. **Student** initiates registration by providing required details (e.g., name, section, credentials). 2. **System** validates the input details. 3. **System** stores the new student account in the database. 4. **System** generates a real-time notification of the new registration. 5. **System** pushes the notification to the **Teacher's dashboard**. 6. **Teacher** views the notification and acknowledges the new student registration.
Precondition	The teacher must be logged in and have access to the dashboard.
Postcondition	● A new student account is successfully stored in the system. ● The teacher instantly receives a notification of the new registration. ● The teacher's dashboard reflects the updated student registration status.

Activity Diagram *(Real-Time Student Registration Notification)*



Wireframe *(Real-Time Student Registration Notification)*



3.3. Non-functional requirements

Performance

1. **Game Responsiveness**
    - Initial Load Time: The CyberKids experience should load the main menu on Roblox within 10 seconds on a standard classroom computer (Intel i3, 4 GB RAM, integrated graphics) over a typical school Wi-Fi connection (20 Mbps).
    - Level Transition: Once a student selects a mission, the new level must begin streaming and be playable within 3 seconds. Roblox's builtin asset streaming and the game's optimized terrain/models support this target.
2. **Leaderboard Updates**
    - Polling-Based Sync: The leaderboard refreshes via a standard HTTP GET request to the API. After a student completes a mission and their score is saved, the client will poll the leaderboard endpoint every 5 seconds and update the display. This ensures no more than a 10-second delay in showing new scores.

Security

1. **User Authentication**
    - **Unique Credentials:**
        - All users authenticate via the Roblox platform. On first dashboard access (teacher) or real-name registration (student), the system issues a backend JWT (JSON Web Token) for subsequent API calls.
        - Teachers and admins log in with email/password credentials stored in the Railway MySQL database; passwords are hashed using bcrypt with a salt.
    - **Role-Based Access Control:**
        - **Student Role**: Can only read and write their own game progress, real-name record, and view leaderboard.
        - **Teacher Role**: Can read data for all students in their class, generate reports, lock/unlock modules, reassign levels, and manage notifications—but cannot modify student scores.
2. **Data Protection**
    - **Real-Name Storage:** Students' real names are stored alongside their Roblox IDs in the Railway MySQL database. No other personal data (e.g., birth dates, addresses) is collected.
    - **Minimal Data Collection:** Only the real name and Roblox ID are collected; no additional personally identifiable information (PII) is stored.
3. **Access Control**
    - Students can only access their own game progress and leaderboard rankings.
    - Teachers can only access data related to their assigned students.

The admin has full system control but cannot alter student scores manually.

Reliability

1. **System Availability**
    - CyberKids runs on Roblox's cloud platform, which offers approximately **99.7% uptime** for published experiences.
    - If the Roblox service is unavailable (planned maintenance or outage), students will be unable to play the game until the platform is restored. Leaderboard and progress synchronization will resume automatically once Roblox comes back online.

2. **Error Handling & Recovery**
    - **Backend Resilience:** All critical data (timers, scores, real names, notifications) are stored in the Railway MySQL database. The Spring Boot API handles retries for transient database errors up to three times with exponential back-off. Failed transactions return clear error messages to the client for user feedback.
    - **Script Errors:** Lua runtime errors are caught using protected calls (`pcall`). If an error occurs during a module, the game waits up to **5 seconds** to attempt an automatic reload of that module. If reload fails, the player sees a clear error message with an option to return to the main menu.