PROJECT TITLE: CodeQuest#

Student Name: Soohang Lingkhim Limbu

London Met ID: 23056769

College ID: np05cp4s240036@iic.edu.np

Internal Supervisor: Mr. Kushal Tamang

External Supervisor: Mr. Samuel Sherpa

Assignment Due Date:

Assignment Submission Date:

Word Count:

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Acknowledgement

I would also like to thank and express my sincere gratitude to everyone who supported and guided me throughout the development of this Final Year Project.

First, I am very grateful to both my external and internal supervisors who provided professional insights, practical perspectives, constructive feedback and timely suggestions that strengthened the quality of this project. Their support greatly enhanced my understanding of the subject matter. I would also like to thank my friends and classmates who offered their assistance, motivation and cooperation.

Table of Contents

1 Introduction

Learning to program can be overwhelming for beginners, especially with text-heavy, non-interactive materials. Many new C# learners struggle with basic concepts and lack immediate guidance, which lowers confidence and slows progress.

To solve this, the proposed project introduces an interactive, game-style web application that teaches C# through structured lessons, coding tasks, quizzes, and a simple 2D battle system. An in-app AI tutor will provide instant, easy-to-understand help whenever students get stuck, making learning more engaging and supportive.

Figure 1:CodeQuset

1.1 Problem Scenario:

Traditional programming instruction often presents several challenges:

- Students are expected to read long tutorials or watch lengthy videos without much hands-on practice.
- Beginners commonly face syntax errors and logical mistakes but lack immediate explanations.
- Many existing platforms treat all learners the same way, without adjusting to their pace or weaknesses.
- When students repeatedly attempt a task and continue failing, it leads to frustration and discouragement.
- Most introductory C# resources are not interactive or game-like, which can make the learning process feel tiring.

These issues affect first-year university students, self-learners preparing for internships, and anyone trying to build a foundation in C#.

1.2 Solution:

- The project is a web-based learning platform that teaches C# through levels in a simple 2D battle game.
- Students read short lessons and then take quizzes or complete small coding tasks.
- Getting answers right lets their character attack the enemy.
- Wrong answers cause the player to take damage or receive helpful hints.
- If a student keeps struggling, an NPC tutor appears to guide them.
- This tutor gives hints, explains what went wrong, and helps the student understand the concept.
- An API such as OpenAI is used to generate explanations that fit each student's specific mistakes.
- Roslyn runs on the backend to check C# code, find errors, and give clear feedback.
- Altogether, the platform aims to make learning C# more fun, less stressful, and more supportive.

2 Aim and Objectives:

2.1 Aim

To build a gamified, web-based learning platform that teaches C# through lessons, interactive challenges, AI-powered assistance, and a simple 2D battle system to increase engagement and understanding.

2.2 Objectives:

The main goal of this project or the objective of this project is as follows:

- Design a structured set of C# learning levels, ranging from basic syntax to OOP and collections.
- Develop a React-based frontend for lessons, quizzes, code editors, and the 2D game interface.
- Build an ASP.NET Core backend that handles account management, level progression, scoring, and data storage.
- Integrate Roslyn to evaluate and analyze user-submitted C# code and generate useful feedback.
- Implement a chatbot tutor that uses an LLM API to answer questions and assist learners who fail a level.
- Develop the 2D game mechanic that ties quiz performance to player and monster HP, and level completion.
- Test the platform for usability, performance, and overall learning effectiveness.

Figure 2: Aim and Objectives

3 Expected Outcomes and Deliverables

- A working web platform where users can learn and practice C# interactively.
- A full set of C# learning levels, each containing lessons and assessments.
- A 2D battle interface linked to quiz outcomes (correct answers = player attack; wrong answers = monster attack).
- An NPC tutor that provides guidance through OpenAI API integration.
- A Roslyn-based module that analyzes C# code submissions and returns feedback.
- Backend APIs for authentication, quiz handling, scoring, and chatbot communication.
- A functioning database storing user progress, scores, and submitted code.
- Complete system documentation (design diagrams, API documentation, testing results).
- Final FYP report and presentation.

4 Project Risks, Threats, and Contingency Plans

Risk	Threats	Contingency Plans
Game integration complexity	Embedding MonoGame into a web environment may be difficult	Switch to a web-based 2D system using HTML canvas or lightweight animation library.
API usage limitations	AI responses may be slow	Cache common explanations, limit chatbot usage, and provide built-in fallback hints
Roslyn too advanced	Full code analysis is heavy	Only use it for basic syntax checking or remove completely
Time constraints	Multiple subsystems (AI, game, code analysis) may be time-consuming	Prioritize essential features first; add animations and extras later

Table 1:PRoject Rislk & Conteingency Plan

5 Methodology

A software development methodology is a way of managing a software development project. This typically addresses issues like selecting features for inclusion in the current version, when software will be released, who works on what, and what testing is done. (Young, 2013) The project will be developed using an iterative, Agile-style approach, which emphasizes gradual development, continuous improvement, and flexibility.

Why Use an Agile-Style Approach?

Agile is chosen because the project involves several interconnected modules whose behaviors depend on iterative testing and user feedback. The gamified learning interface, Roslyn code analysis, and AI chatbot may require multiple adjustments to achieve an intuitive and seamless user experience.

Agile allows the developer to build, test, evaluate, and improve the system in short cycles, ensuring that issues are identified early and enhancements can be integrated continuously. (Young, 2013)

Figure 3: Agile Method

6 Resource Requirements

6.1 Hardware Requirement:

- Desktop or laptop (workstation)
- Internet Connection

6.2 Software Requirements:

- **Frontend:** React, , VS Code
- **Backend:** ASP.NET Core, .NET SDK
- **Code Analysis:** Roslyn compiler platform
- **Game:** MonoGame (if used) OR HTML5 canvas/WebGL libraries
- **Database:** SQL Server or PostgreSQL
- **Caching:** MemoryCache (ASP.NET In-memory cache), Redis
- **Framework :** Blazor UI
- **Other Tools:** Git, GitHub, Postman, Docker (optional)
- **AI Integration:** Access to OpenAI API or equivalent LLM service

7 Entity Relationship Diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system. (Lucid Software Inc, 2025)

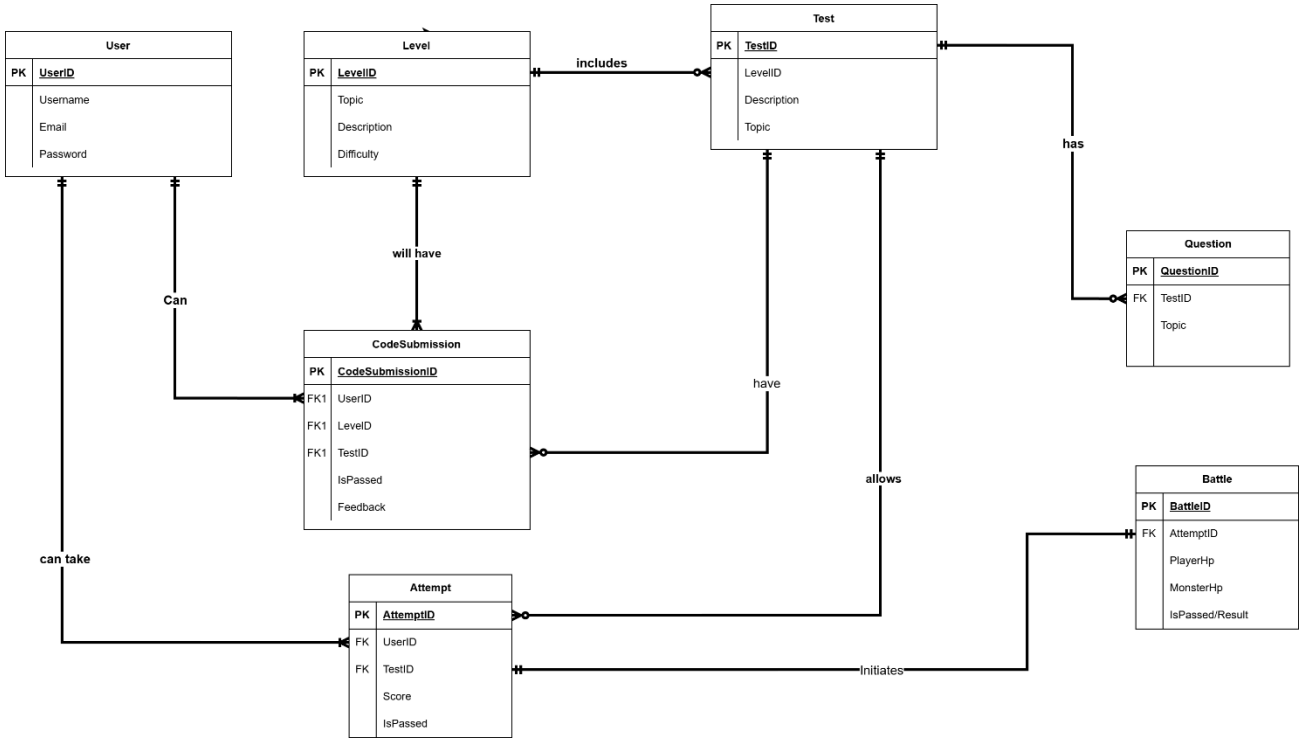

Figure 4:ERD

8 Work Breakdown Structure (WBS)

A WBS provides a hierarchical view of your project's scope, translating overall strategies and objectives into specific goals, workflows, project phases, and action plans. (Coursera Staff, 2025)

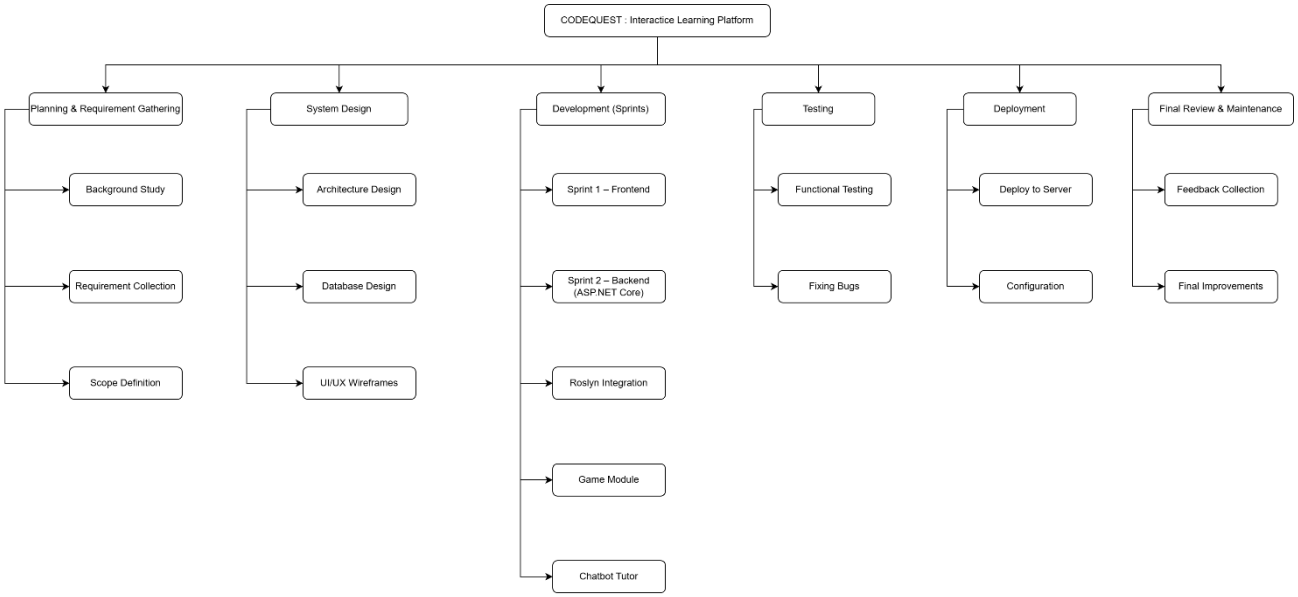

Figure 5: WBS

9 Milestones

Project Milestone, or also known as checkpoints, refers to a significant point or event within a project timeline that marks the completion of a major phase, deliverable, or objective.

- **Milestone 1 – Topic Finalization**

  Finalize the project theme, objectives, and overall scope for the C# gamified learning platform.

- **Milestone 2 – Proposal Submission**

  Submit the formal proposal outlining the aims, requirements, expected outcomes, Gantt chart, and WBS.

- **Milestone 3 – System Analysis**

  Prepare system analysis artifacts including SRS, ERD, and UML diagrams to define the platform's functionalities.

- **Milestone 4 – System Design**

  Complete UI/UX wireframes, database design, and system architecture as the blueprint for development.

- **Milestone 5 – Sprint Development Phase**

  Carry out iterative development across five sprints: frontend setup, backend development, Roslyn integration, game module, and chatbot tutor.

- **Milestone 6 – System Testing**

  Test all modules for functionality, performance, and stability, fix issues for smooth operation.

- **Milestone 7 – Review & Feedback**

  Refine and update the project based on supervisor feedback and final checks.

- **Milestone 8 – Deployment & Final Improvements**

  Deploy the system, perform final adjustments, and prepare for final submission and presentation.

10 Gantt Chart:

A gantt chart is a horizontal bar chart that visually maps out a project over time. It displays tasks down the vertical axis and time intervals across the horizontal axis.

Each task bar on the gantt chart shows:

- When work starts
- How long it'll take
- When it's due
- How much progress has been made (LaPrad, 2025)

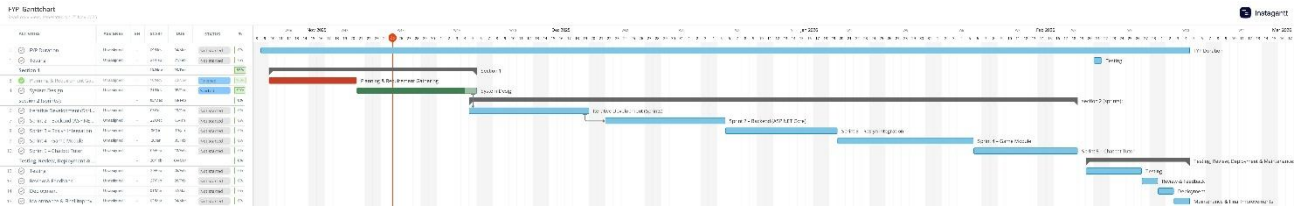

Figure 6: Gantt Chart

11 Conclusion

In summary, this project aims to create a more engaging and supportive way for students to learn C#. By combining a web-based platform with simple 2D game mechanics, the system turns each lesson into an interactive experience that encourages steady progress. Features such as instant feedback, code analysis through Roslyn, and a helpful in-game tutor ensure that learners aren't left confused when they make mistakes. The structured development plan, supported by clear milestones and a detailed Gantt chart, provides a practical path from concept to completion. With the right balance of education and gameplay, the project is designed to make learning more approachable, motivating, and effective for students.

12 References

Coursera Staff, 2025. *Coursera.* [Online] Available at: https://www.coursera.org/articles/work-breakdown-structure?msockid=3faad288af3262bf37abc40eae8c6315
[Accessed 27 11 2025].

Kruger, G., 2025. *Preforce.* [Online] Available at: https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document
[Accessed 28 11 2025].

LaPrad, L., 2025. *TeamGantt.* [Online] Available at: https://www.teamgantt.com/what-is-a-gantt-chart
[Accessed 28 11 2025].

Lucid Software Inc, 2025. *Lucid Software Inc.* [Online] Available at: https://www.lucidchart.com/pages/er-diagrams
[Accessed 27 11 2025].

Paradigm, 2025. *Paradigm.* [Online] Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/
[Accessed 28 11 2025].

Patil, A., 2024. *Orhanergun.* [Online] Available at: https://orhanergun.net/understanding-high-level-design-an-introduction-for-beginners
[Accessed 29 11 2025].

Young, D., 2013. *Software Development Methodologies.* [Online] Available at: https://www.researchgate.net/publication/255710396_Software_Development_Methodologies
[Accessed 28 11 2025].

13 Appendix

13.1 Methodology

Benefits of using an Agile Approach:

- Flexibility to Adjust Requirements
  Agile allows changes throughout development. This is important because of the game mechanics, chatbot behavior, and code-analysis features may require frequent refinement.
- Early and Continuous Testing
  Each iteration includes testing, meaning bugs and design issues are caught early—especially helpful for complex features like Roslyn integration or the 2D battle system.
- Faster Delivery of Working Features
  Agile produces small, functional modules in each sprint, allowing visible progress and ongoing feedback from supervisors and testers.
- Better User Experience
  User feedback can be quickly applied to improve lesson flow, gameplay difficulty, and UI design—critical for an educational app.
- Lower Project Risk
  Breaking the project into small cycles reduces the chance of large failures and helps manage complex integrations step-by-step.

13.1.1 Agile Development Phases:

Planning & Requirement Gathering

- Identify core features such as lessons, quizzes, leveling system, 2D battle mechanics, and chatbot tutor.
- Define user roles and learning flow.

System & Game Design

- Create UML diagrams and database schema.
- Decide how the 2D game interface will be implemented (MonoGame or canvas-based).
- Design lesson structure, quiz format, and level progression logic.

Iterative Development

- Sprint 1 – Frontend (React, C#):
  - Build UI for lessons, quizzes, code editor, and battle screen .
- Sprint 2 – Backend (ASP.NET Core):
  - Implement APIs for user authentication, quizzes, scoring, and progression .
- Sprint 3 – Roslyn Integration:
  - Add code parsing and automated feedback for coding challenges.
- Sprint 4 – Game Module:
  - Develop HP logic, damage system, basic 2D animations, and level completion mechanics.
- Sprint 5 – Chatbot Tutor:
  - Integrate LLM API, implement prompt constraints, and create UI for NPC helper.

Testing

- Unit testing for backend logic.
- Functional testing for quizzes, Roslyn code evaluation, and battle system.
- Usability testing to ensure smooth gameplay and learning experience.

Review & Feedback

- Present sprint outputs to supervisor/test users.
- Gather feedback to refine game mechanics, quiz difficulty, and chatbot responses.

Deployment

- Deploy a working prototype to a web server.
- Set up user accounts and track progress for demonstration purposes.

Maintenance & Final Improvements

- Fix bugs, polish animations/UI, and optimize API performance.
- Prepare final documentation, diagrams, and presentation materials.

13.2 ERD

Relationships:

1. **User => CodeSubmission (1 to Many)**

   One User can have many CodeSubmissions

2. **Level => CodeSubmission**

   One level will have many CodeSubmissions

3. **Test => CodeSubmission**

   One Test can have many CodeSubmission.

4. **User => Attempt**

   One User can take multiple Attemps.

5. **Test => Attempt**

   Each Test allows multiple Attempts.

6. **Attempt => Battle**

   Each Attempt only initiates one Battle

7. **Level => Test**

   A Level includes many Tests.

8. **Test => Question**

   A Test has many Questions

13.3 SRS

An SRS (System Requirements Specification) serves as a clear blueprint for software development, outlining the purpose of the software, what it will do, and the specific functional and non-functional requirements it must meet. It links user stories to concrete requirements and ensures all stakeholders are aligned with the roadmap that guides development efficiently and transparently. (Kruger, 2025)

13.4 System Overview

The project is a web-based learning platform that teaches C# through interactive lessons, quizzes, coding tasks, and a simple 2D battle game. Students progress through level by completing tasks and quizzes. The aim is to make learning more engaging, supportive and less overwhelming for begginers.

13.5 Functional Requirements

13.5.1 User Features

- Users can register and log in.
- Users can view C# lessons divides into topics and levels
- Users can complete quizzes and coding tasks for each lesson.

13.5.2 Quiz and Coding System

- Each lesson must include quizzes or coding questions.
- A built-in code editor lets users write and submit C# code.
- Correct answers trigger in-game action (e.g., attack).
- Incorrect answers reduce player health or give hints.

13.5.3 Game Battle Module

- Each level includes and simple 2D battle scenario.
- Player and monster health update based on quiz outcomes.
- Completing a battle unlocks the next lesson or level.

13.5.4 NPC Tutor System

- If a user fails multiple attempts, he can consult with the ai chatbot.
- The provided answers and explanations through ai API.

13.5.5 Admin Panel

- Admin can add/edit lessons, quizzes and game content.

13.6 Non-Functional Requirement

13.6.1 Performance

- Lessons and quizzes should be loaded within a few seconds.
- Code analysis must respond promptly.

13.6.2 Usability

- Interface should be simple and beginner friendly.
- Game elements should enhance learning, not distract.

13.6.3 Security

- User accounts must be protected with secure authentication.
- API keys must not be exposed to the client side.

13.6.4 Reliability

- Systems should handle multiple users without failure.
- Basic fallback messages must appear if the API is unavailable.

13.7 Use Case Diagram

A UML or also known as use case diagram is that visually represents the functional requirements of a system. It shows who interacts with the system(actors) and what the use cases do. (Paradigm, 2025)

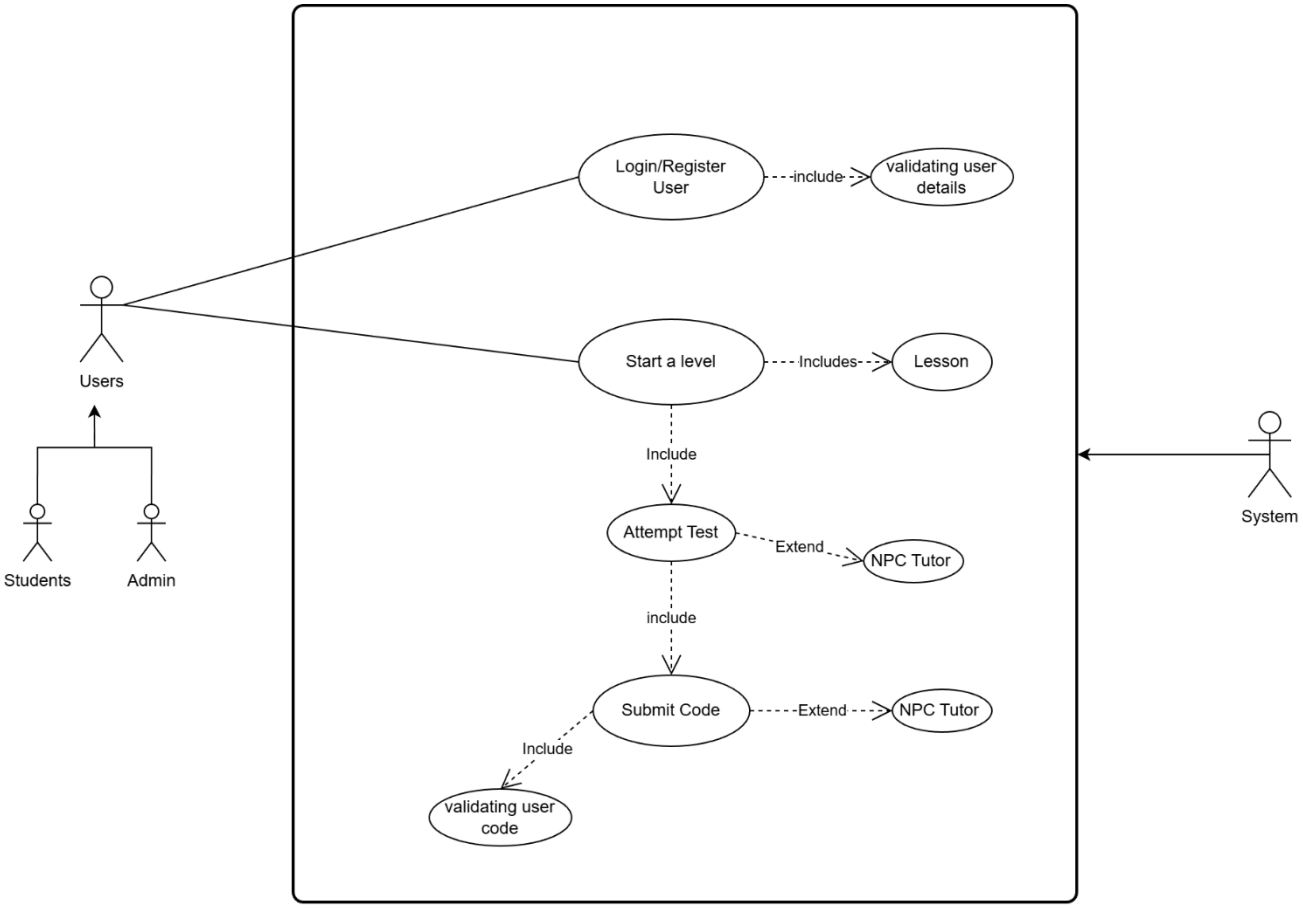

Figure 7: Use Case Diagram

13.8 High Level Description

High-Level Design in software engineering is a conceptual blueprint. Much like an architect's plan for a building, HLD outlines the software system's architecture, its modules, and the interaction between them, all at a high level. (Patil, 2024)

Use Case 1: Login/Register User

Actor: User (Student or Admin)

Description:

The user logs into the system or creates a new account. The system verifies the provided credentials to ensure authenticity and prevent unauthorized access.

13.8.1 Use Case 2: Start a Level

Actor: Student
Description:
The student selects a level to begin learning. The system loads the lesson content and unlocks the corresponding test. Within the level, students can read lessons and later attempt the test related to that level.

13.8.2 Use Case 3: Lesson

Actor: Student (through Start Level)

Description:

The student views structured learning content such as explanations, examples, and topic descriptions. The lesson prepares the student for the upcoming test.

13.8.3 Use Case 4: Attempt Test

Actor: Student

Description:

The student answers test questions for the chosen level. The system records responses, validates answers, and calculates the test outcome. Tutor help may be triggered if the student needs assistance.

13.8.4 Use Case5: Submit Code

Actor: Student

Description:

The student writes and submits C# code for a test or coding challenge. The system analyzes and validates the submitted code, checking syntax and correctness. NPC Tutor support may be provided for debugging or hints.

13.8.5 Use Case 6: Validating User Code

Actor: System

Description:

The system analyzes and evaluates the student's submitted code using internal logic or a compiler. It provides feedback, identifies errors, and determines whether the submission meets the expected behavior.

13.8.6 Use Case 7: NPC Tutor

Actor: System (as helper)

Description:

An AI-driven tutor provides hints, explanations, or guidance whenever the student struggles during lessons, tests, or code submission. This support only appears when needed or requested.