THE UNIVERSITY OF DANANG
UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: INFORMATION TECHNOLOGY

SPECIALTY: SECURITY

PROJECT TITLE:

DEVSPELL - A LEARNING PROGRAMMING PLATFORM WITH RPG STYLE

Instructor: **Dr. VO DUC HOANG**
Student: **NGUYEN TUAN**
Student ID: **102210194**
Class**:** **21TCLC_DT2**

Da Nang, 6/2025

DA NANG UNIVERSITY
UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY INFORMATION TECHNOLOGY

SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom - Happiness

GRADUATION PROJECT COMMENT

I. General information:

1. Student name: Nguyen Tuan
2. Class: 21TCLC_DT2 Student ID: 102210194
3. Topic title: DEVSPELL – A learning programming platform with RPG style
4. Instructor: Vo Duc Hoang
   Academic title/ degree: Doctor of Philosophy

II. Reviews of graduation project

1. About the urgency, novelty, usability of the topic: (2 points)
   ……
   ……
2. About the results of solving the tasks required by the project: (4 points)
   ……
   ……
3. About the form, structure and layout of the graduation project: (2 points)
   ……
   ……
4. The topic includes scientific value/article/problem solving of the enterprise or school: (1 point)
   ……
   ……
5. Existing shortcoming need to be supplemented or modified:
   ……
   ……

III. Spirit and attitude of the student (1 point):

……
……

IV. Evaluation:

1. Evaluation point: …./10
2. Suggest: ☐ Defense permitted/ ☐ Edit to defend/ ☐ Defense not permitted

Da Nang, date … month … year 2025

Instructor

SUMMARY

Topic title: DEVSPELL – A Learning programming platform with RPG style
Student name: Nguyen Tuan
Student ID: 102210194
Class: 21TCLC_DT2

In the context of rapid technological advancement—especially in the fields of Artificial Intelligence (AI) and Information Technology—programming has become a highly valuable and essential skill across many industries. However, traditional programming education often lacks interactivity, practical engagement, and learner motivation, making it difficult for many individuals to stay committed. These limitations frequently result in early dropout, poor learning outcomes, and an inability to apply knowledge in real-world scenarios.

Meanwhile, the demand for skilled programmers continues to rise with the growth of AI and automation. Yet, there remains a significant gap between learners and actual industry needs. Most current e-learning platforms fail to personalize learning paths or adapt to individual learning paces and skill levels. As a result, learners often struggle to engage effectively with the content.

This thesis aims to address these issues by developing an RPG-style (Role-Playing Game) programming learning platform called DEVSPELL. The platform combines core programming education with gamification elements such as quests, levels, rewards, and rankings to create an engaging and interactive learning experience. The project not only focuses on teaching programming knowledge but also promotes logical thinking and problem-solving skills while motivating users through game-based incentives.

DA NANG UNIVERSITY
UNIVERSITY OF SICIENCE AND TECHNOLOGY
FALCUTY INFORMATION & TECHNOLOGY

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom - Happiness

GRADUATION PROJECT REQUIREMENTS

Student Name: NGUYEN TUAN Student ID: 102210194

Class: 21TCLC_DT2 Major: Security

1. *Topic title: DEVSPELL – A LEARNING PROGRAMMING PLATFORM WITH RPG STYLE*
2. *Project topic:* ☐ *has signed intellectual property agreement for final result*
3. *Content of the explanations and calculations:*

Content of the explanations contains 6 parts

- **Introduction**: This chapter gives information about the context and purpose of the project as well as giving the scope of the problems which will be focused on the thesis.
- **Chapter 1**: Theories and Technologies
- **Chapter 2**: Chatbot System
- **Chapter 3**: System analysis and design
- **Chapter 4**: System deployment and evaluation
- **Conclusion**

4. *Drawings, charts (specify the types and sizes of drawings):*
   - Use case diagram
   - Activity diagram
   - Sequence diagram
   - Architecture structure diagram
   - Chatbot structure
   - Database diagram
5. *Instructor:*

   Dr. Vo Duc Hoang, Information Technology Faculty, Danang University of Science and Technology, The University of Danang, Vietnam.
6. *Date of assignment: 24/3/2025*
7. *Date of completion: 1/6/2025.*

Da Nang, date … month… year 2025

Head of Division Software Engineering **Instructor**

PREFACE

To achieve meaningful results in this project, I have been fortunate to receive support and encouragement from many individuals. With deep appreciation and sincerity, I would like to express my heartfelt gratitude to everyone who has accompanied and supported me throughout my academic journey and research.

First and foremost, I would like to thank all the lecturers at the University of Science and Technology – The University of Da Nang, particularly those in the Faculty of Information Technology. Your dedication and the valuable knowledge you have imparted over the years have laid a solid foundation for me to undertake and complete this graduation project.

I am especially grateful to **Dr. Vo Duc Hoang**, my project supervisor, for his enthusiastic guidance, generous sharing of expertise, and unwavering support throughout the course of this work. His mentorship and encouragement have been instrumental in the successful completion of this project.

Despite my best efforts, due to limitations in knowledge and experience, there may be shortcomings or errors in this report. I sincerely welcome any feedback, comments, or suggestions from teachers and readers so that I can continue to improve and grow.

Thank you all once again.

COMFIRMATION

I guarantee:

1. The contents of this senior project are performed by myself following the guidance of instructor Dr. Vo Duc Hoang.
2. All references used in this senior project thesis are quoted with the author's name, project name, time and location to publish clearly and faithfully.
3. All invalid copies educated statue violation or cheating will be borne the full responsibility by myself.

Student Performed

PROJECT TASK ASSIGNMENT

Tran Van Dat's Task Assignment Table

No.	Task Name
1	Back-end development
2	Database design
3	Chatbot solution design and integrate
4	Build CI/CD pipeline
5	Front-end development (Achievement, User profile, Certificate)

Nguyen Tuan's Task Assignment Table

No.	Task Name
1	Front-end development (User page & Admin page)
2	Integrate API with front-end
3	Fine-tune chatbot model

Common Task Assignment Table

No.	Task Name
1	Research Topic, Market Study, and Requirements Analysis
2	Documentation and Thesis Report Writing
3	System Architecture Planning and Design Collaboration

TABLE OF CONTENT

LIST OF PICTURES

LIST OF TABLES

LIST OF SYMBOLS, ACRONYM

No	Acronym	Explanation
1	RPG	Role-playing game
2	JS	JavaScript
3	XP	Experience
4	SQL	Structured Query Language
5	CI/CD	Continuous Integration/Continuous Deployment
6	SSE	Server Sent Events
7	VM	Virtual Machine

INTRODUCTION

1. Reason for Choosing the Topic

In the context of rapid technological advancement—especially in the fields of Artificial Intelligence (AI) and Information Technology—programming has become a highly valuable and essential skill across many industries. However, traditional programming education often lacks interactivity, practical engagement, and learner motivation, making it difficult for many individuals to stay committed. These limitations frequently result in early dropout, poor learning outcomes, and an inability to apply knowledge in real-world scenarios.

Meanwhile, the demand for skilled programmers continues to rise with the growth of AI and automation. Yet, there remains a significant gap between learners and actual industry needs. Most current e-learning platforms fail to personalize learning paths or adapt to individual learning paces and skill levels. As a result, learners often struggle to engage effectively with the content.

This thesis aims to address these issues by developing an RPG-style (Role-Playing Game) programming learning platform called **DEVSPELL**. The platform combines core programming education with gamification elements such as quests, levels, rewards, and rankings to create an engaging and interactive learning experience. The project not only focuses on teaching programming knowledge but also promotes logical thinking and problem-solving skills while motivating users through game-based incentives.

2. Objectives and Tasks of the Project

- Objectives:
    - Build an e-learning platform supporting programming learning in an RPG style, helping learners improve programming skills through challenges and gamification.
    - Create a flexible, scalable learning system that benefits both beginners and experienced learners.

- Tasks:
    - Develop a web-based application.
    - Design system architecture and build main functional modules.
    - Implement continuous integration/deployment to optimize website workflow.
    - Build main features:
        - Course management: Search, enroll, and track learning progress.
        - Access lessons & exercises: View lecture content, do interactive exercises with detailed guidance.
        - Gamification system: Earn EXP, level up, complete quests, and receive rewards.
        - Leaderboard & achievements: View ranking and unlock badges.
        - Interaction & support: Use a chatbot for learning assistance.
        - Item shop: Purchase items to support learning.
        - Share learning progress: Display personal profiles and achievements.
        - Admin content management: Add, edit courses, exercises, and items.

3. Expected Results

- Online programming learning platform: Provide an environment that helps users approach programming knowledge in a fun and effective way.
- Course and exercise system: Include well-structured courses with lectures combined with practical exercises such as quizzes, fill-in-the-blank, and direct coding.
- Gamification mechanism: Integrate levels, leaderboards, and rewards to motivate learners, helping maintain interest and continue practicing programming skills.
- Management system: Allow administrators to manage course content, adjust exercise difficulty, and optimize user experience.
- Learning support chatbot: Provide a virtual assistant to explain exercises, suggest solutions, and support learners during their practice.

4. Practical and Scientific Significance

Practical significance: The platform offers a new approach to learning programming, helping learners maintain motivation through tasks, achievements, and rewards. This can significantly improve course completion rates compared to traditional platforms. The system also has potential to expand to support other fields beyond programming, diversifying online education forms.

Scientific significance: The project applies educational principles combined with gamification to optimize the learning process. The RESTful API design ensures integration and scalability, while data processing solutions improve performance and user experience. Additionally, the AI learning support chatbot shows the potential of artificial intelligence in education, opening many directions for future research and development.

Chapter 1: THEORIES AND TECHNOLOGIES

1.1. Continuous Integration and Continuous Deployment (CI/CD)

Continuous Integration (CI) refers to the process of automatically integrating code changes from multiple contributors into a shared repository several times a day. Each integration is verified by an automated build and testing process, which helps detect errors quickly and ensures that new code does not break the existing system. By integrating frequently, teams reduce the complexity of merging code and enhance collaboration among developers.

Continuous Deployment (CD) builds upon CI by automating the release process. Once the code passes all stages of the testing pipeline, it is automatically deployed to production or a staging environment without manual intervention. This practice significantly reduces the time between writing code and delivering it to users, allowing for faster feedback, quicker iteration, and more robust software.

The CI/CD pipeline typically includes stages such as:

- **Code commit**: Developers push code changes to a version control system (e.g., Git).
- **Build**: The application is compiled or bundled.
- **Test**: Automated tests (unit, integration, end-to-end) are executed.
- **Deploy**: The application is deployed to testing, staging, or production environments.

In the context of this project, CI/CD is implemented using **GitHub Actions** for automating workflows, **Docker** for consistent environment packaging, and **Nginx** for deployment. This pipeline ensures that new features and updates are integrated smoothly and deployed efficiently, minimizing downtime and enhancing the platform's reliability.

CI/CD not only improves the development workflow but also aligns with DevOps principles by fostering a culture of automation, collaboration, and continuous improvement.

1.2. Docker

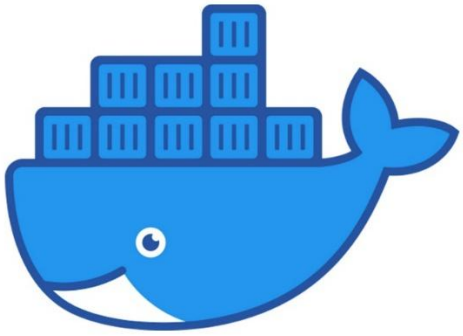

Figure 1.1 Docker official image

Docker [1] is an open-source platform designed to automate the deployment, scaling, and management of applications through **containerization**. It allows developers to package applications along with all their dependencies, libraries, and configuration files into lightweight, portable containers that can run consistently across various environments.

Traditionally, running an application across different systems (e.g., development, testing, and production) can lead to issues due to differences in environment setups. Docker addresses this problem by creating a standardized unit of software—the container—which ensures that an application behaves the same regardless of the host system.

A Docker container is based on a Docker image, which defines the application's environment, including the operating system, application code, runtime, system tools, and libraries. These images are created using a file called a Dockerfile, where the build steps and configurations are defined.

Key components of Docker include:

- Docker Engine: The core part of Docker that creates and runs containers.
- Dockerfile: A script that contains instructions to build a Docker image.
- Docker Compose: A tool that allows defining and running multi-container Docker applications using a YAML file.

- Docker Hub: A cloud-based repository where users can publish, share, and download Docker images.

In this project, Docker is used to containerize both the frontend (developed with ReactJS and Vite) and backend (developed with NestJS). It provides several advantages:

- Consistency between development and production environments.
- Simplified deployment, as containers can be easily deployed and managed on any server with Docker installed
- Scalability, as new containers can be launched quickly to handle increased load.
- Isolation, ensuring that services run independently without interfering with each other.

1.3. Caching

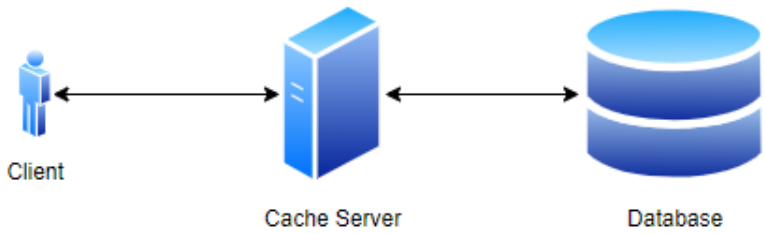

Figure 1.2 Caching data between Database and Client

Caching is a technique used in software systems to store frequently accessed data in a temporary storage layer—called a cache—so that future requests for the same data can be served more quickly. By reducing the need to repeatedly access the original data source, caching improves system performance, reduces latency, and decreases the load on databases and APIs.

There are several types of caching, including:

- **In-memory caching**: Stores data in RAM for extremely fast access. Common tools for this approach include Redis and Memcached.
- **Client-side caching**: Stores data on the user's device, often using browser storage mechanisms like localStorage or service workers.
- **Server-side caching**: Stores processed responses or query results on the server to avoid redundant processing.
- **Database caching**: Keeps frequently accessed query results in memory to reduce the load on the database engine.

In this project, **Redis** is used as the primary caching solution. Redis is an in-memory data store known for its high performance and support for various data structures such as strings, hashes, lists, and sets. It is well-suited for caching due to its low-latency data access and simplicity of use.

The caching system in the project supports several use cases:

- **Reducing database load** by caching query results for frequently accessed data such as course listings, user progress, or leaderboard rankings.
- **Improving response time** for API calls by serving cached data instead of querying the database every time.
- **Temporary data storage**, such as storing tokens, session data, or pre-processed chatbot responses.

Caching is particularly important in this e-learning platform to ensure a smooth and responsive user experience, especially when handling a growing number of users and real-time features such as gamification progress, leaderboards, and chatbot interactions.

1.4. Cron Jobs

A **Cron Job** is a time-based job scheduler commonly used in Unix-like operating systems. It allows developers and system administrators to automate the execution of scripts, commands, or programs at specified intervals or on a recurring schedule—such as every minute, hour, day, or week.

Cron jobs are defined using a **cron expression**, which is a string consisting of five or six fields that represent the time and date when the job should run. For example, a cron expression like 0 0 * * * means the task will execute every day at midnight.

Cron jobs are useful for automating repetitive tasks such as:

- Backing up databases or files
- Sending scheduled notifications or emails
- Cleaning up temporary data
- Generating periodic reports
- Updating statistics or syncing data between services

In the context of this project, cron jobs are used to support various backend operations that need to run regularly or at specific intervals. Examples include:

- **Resetting daily missions** and rewards for users.
- **Updating user activity** based on user progress or activity.
- **Clearing expired cache entries** from Redis to optimize memory usage.

1.5. REST API

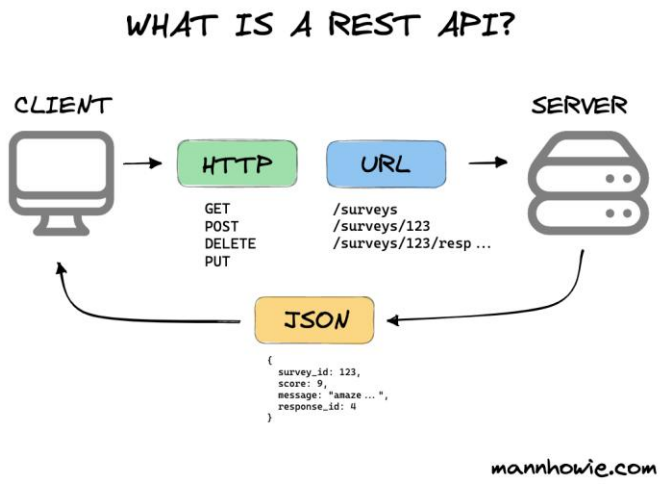

Figure 1.3 REST API Demonstrate

REST API stands for **RE**presentational **S**tate **T**ransfer **API**. It is a type of **API (Application Programming Interface)** that allows communication between different systems over the internet. **REST APIs** work by sending requests and receiving responses, typically in **JSON format**, between the client and server.

REST APIs use **HTTP methods** (such as GET, POST, PUT, DELETE) to define actions that can be performed on resources. These methods align with **CRUD (Create, Read, Update, Delete)** operations, which are used to manipulate resources over the web.

Key Features of REST APIs

- **Stateless**: Each request from a client to a server must contain all the information the server needs to fulfill the request. No session state is stored on the server.
- **Client-Server Architecture**: RESTful APIs are based on a client-server model, where the client and server operate independently, allowing scalability.
- **Cacheable**: Responses from the server can be explicitly marked as cacheable or non-cacheable to improve performance.
- **Uniform Interface**: REST APIs follow a set of conventions and constraints, such as consistent URL paths, standardized HTTP methods, and status codes, to ensure smooth communication.
- **Layered System**: REST APIs can be deployed on multiple layers, which helps with scalability and security.

1.6. OAuth 2.0

OAuth 2.0 [2] is an open standard for access delegation, commonly used as a way for internet users to grant websites or applications access to their information on other websites but without giving them the passwords. It provides a secure and

standardized way for third-party applications to access user resources hosted on a web service.

At its heart, OAuth 2.0 is about **delegated authorization**. Instead of a user sharing their credentials (like a username and password) directly with a third-party application, OAuth 2.0 allows the user (Resource Owner) to grant that application (Client) limited access to their resources (e.g., photos, contacts, profile information) hosted on a Resource Server (e.g., Google, Facebook, Twitter). This access is granted via an Access Token, which is issued by an Authorization Server.

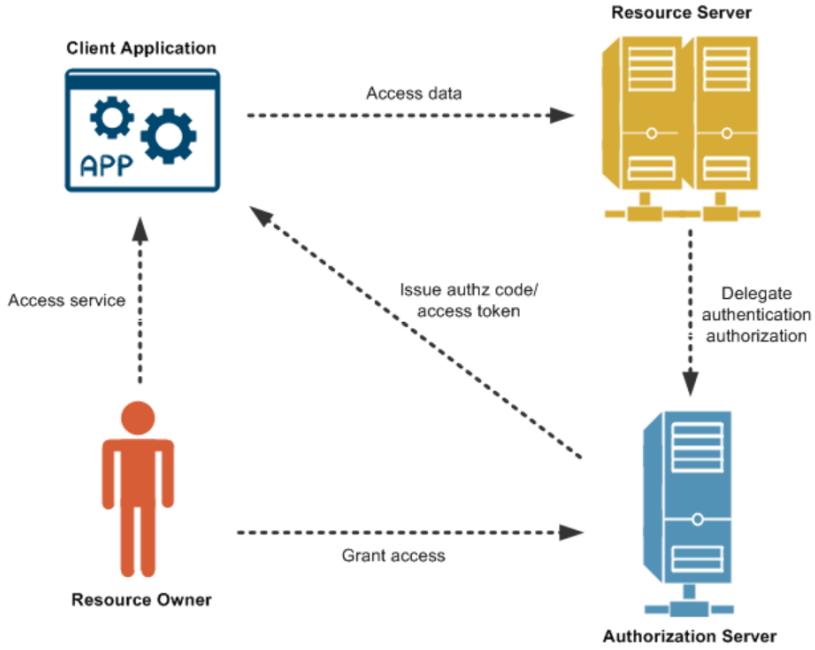

Figure 1.4 OAuth 2.0 structure

Key Actors in the OAuth 2.0 Framework:

- **Resource Owner:** The user who owns the data and can grant access to it.
- **Client (Third-Party Application):** The application that wants to access the Resource Owner's data. This could be a web application, mobile app, or desktop application.
- **Authorization Server:** The server that authenticates the Resource Owner and issues Access Tokens after obtaining authorization from the Resource Owner. This server is often, but not always, the same as the Resource Server.
- **Resource Server:** The server hosting the protected resources (e.g., an API for accessing user data). It accepts and validates Access Tokens from the Client.

1.7. JSON Web Tokens

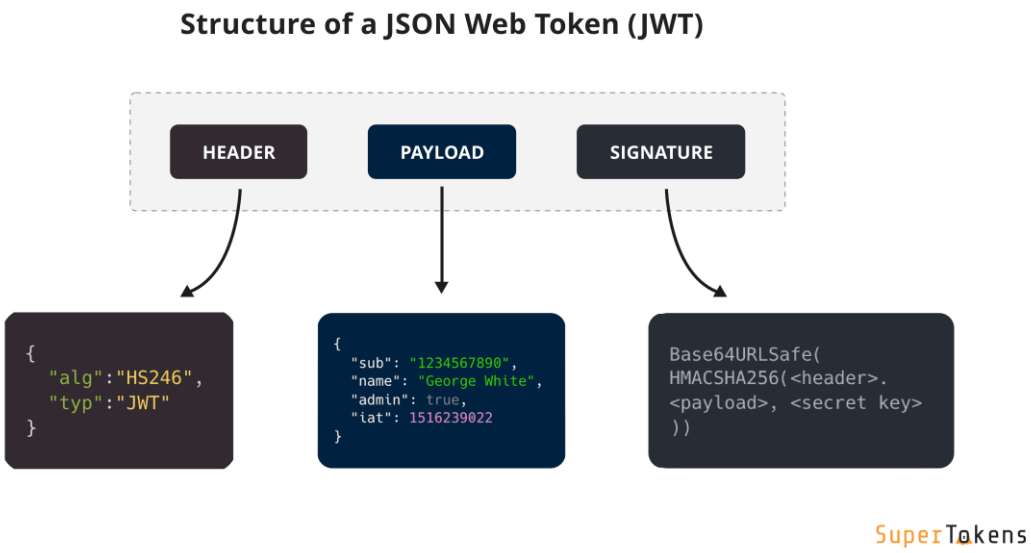

Figure 1.5 JSON Web Token Structure

Authentication and authorization are fundamental aspects of web application security. They ensure that users are who they claim to be and have the right permissions to access specific resources. Traditionally, this was handled through session-based authentication, where user information was stored on the server. However, as applications became more complex and distributed, traditional authentication methods relying on opaque tokens or session-based approaches faced limitations. Validating these tokens often required multiple database lookups and complex server-side logic, leading to performance issues and scalability challenges. The stateful nature of these tokens meant servers had to maintain session information, which could quickly become unwieldy as the number of users and devices grew.

JSON Web Tokens (JWTs) [3] emerged as a more flexible solution. JWTs are self-contained, stateless tokens that carry all the necessary information for authentication and authorization within the token itself. This eliminates the need for servers to maintain session data, allowing for more efficient and scalable authentication across distributed systems. By encoding the user's claims and permissions directly into the token, JWTs can be validated locally without requiring multiple database lookups or complex server-side logic. This streamlined approach improves performance and enables seamless authentication and authorization, even in

microservices architectures or across different domains. The stateless nature of JWTs also makes them more resilient to system failures, as there is no central point of failure for authentication.

JWTs or JSON Web Tokens are most commonly used to identify an authenticated user. They are issued by an authentication server and are consumed by the client-server (to secure its APIs).

1.8. Server Sent Events – SSE

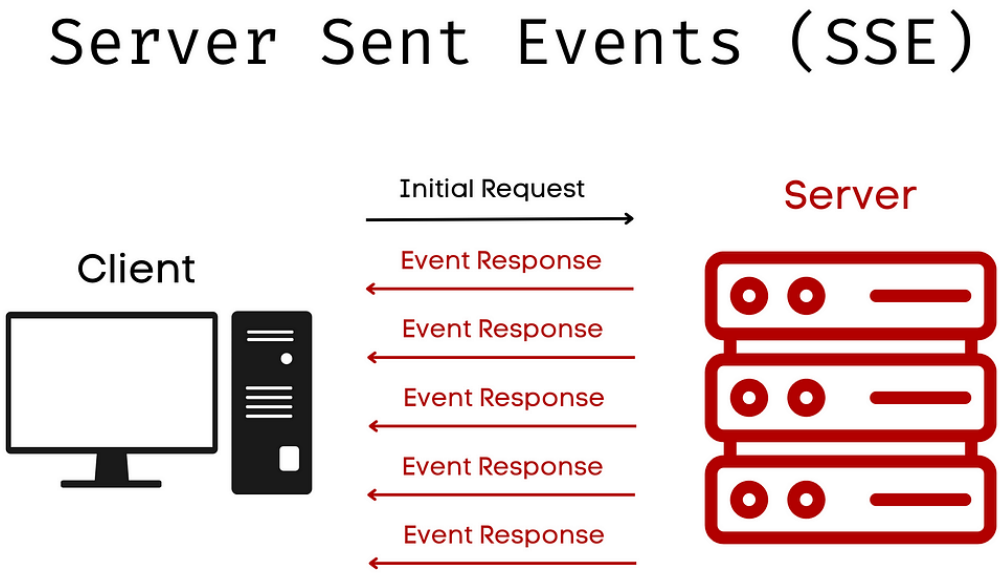

Figure 1.6 A simple demonstrate of the SSE

Server-Sent Events (SSE) [4] is a web development technology that allows servers to push real-time updates to web clients over a single, persistent HTTP connection. Unlike WebSockets, SSE operates in a unidirectional manner, where data flows only from the server to the client. This makes SSE lightweight and ideal for scenarios requiring real-time updates without the complexity of two-way communication.

SSE uses a text-based protocol to send data as a stream of events, making it easy to implement and debug in various web applications.

SSE offers several benefits that make it a popular choice for real-time web applications:

- **Ease of Use**: SSE is natively supported in most modern browsers via the EventSource API, simplifying its adoption.
- **Efficient Resource Usage**: By maintaining a single HTTP connection, SSE reduces the overhead compared to traditional polling.
- **Automatic Reconnection**: Built-in reconnection support ensures stable performance even on unreliable networks.
- **Text-Based Protocol**: The simplicity of its text-based format makes SSE easy to integrate into existing systems and troubleshoot during development.

1.9. Vector similarity

Vector similarity [5] is a fundamental concept used to quantify the likeness between items represented as numerical vectors, often called embeddings. This is particularly important in applications like semantic search (finding documents or items with similar meanings), recommendation systems (suggesting items similar to what a user has liked), and anomaly detection (identifying items that are significantly different from the norm).

The core idea is to represent items (such as words, sentences, documents, images, or user preferences) as points in a multi-dimensional space. The "closer" two vectors are in this space, the more similar the items they represent are considered to be. The method used to calculate this "closeness" or "distance" is determined by a similarity metric.

Here's a more detailed look at the common vector similarity metrics:

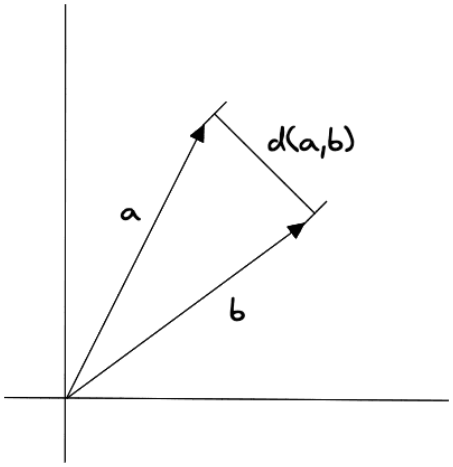

Figure 1.7 Euclidean Distance

- Euclidean Distance (L2 Norm):
    - Concept: This is the most intuitive measure of distance, representing the straight-line or "as-the-crow-flies" distance between the tips of two vectors. Imagine two points in a 2D or 3D space; Euclidean distance is the length of the line segment connecting them. This concept extends to higher-dimensional spaces.
    - Calculation: If you have two vectors, a = (a_1, a_2, ..., a_n) and b = (b_1, b_2, ..., b_n), the Euclidean distance is calculated as:

$$d(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

    - Interpretation: A smaller Euclidean distance indicates greater similarity. A distance of 0 means the vectors are identical.
    - Use Case: Often used when the magnitude of the vectors is important.

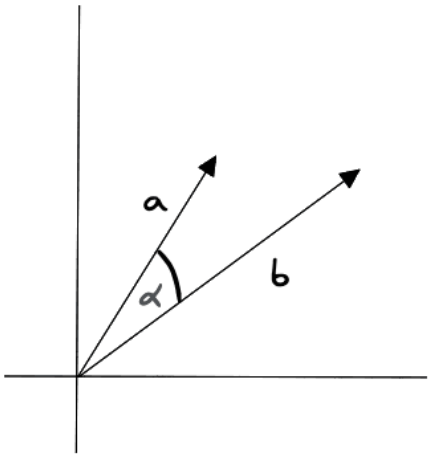

Figure 1.8 Cosine Similarity

- Cosine Similarity:
    - Concept: This metric measures the cosine of the angle between two vectors. It primarily focuses on the orientation of the vectors rather than their magnitude. If two vectors point in the same general direction, their cosine similarity will be high, even if they have different lengths.
    - Calculation: For vectors a and b, the cosine similarity is calculated as:

$$sim(a,b) = \frac{a \cdot b}{\|a\| \cdot \|b\|}$$

Where a · b is the dot product of a and b, and ||a|| and ||b|| are the magnitudes (or lengths) of vectors a and b, respectively.

- Interpretation: The value ranges from -1 to 1.
  - 1: Vectors are identical in orientation (angle is 0°).
  - 0: Vectors are orthogonal (angle is 90°), indicating no similarity in orientation.
  - -1: Vectors point in opposite directions (angle is 180°).
- Use Case: Widely used in text analysis and information retrieval where the relative frequency of terms (orientation) is more important than the document length or absolute term counts (magnitude). For example, two documents discussing the same topic might have very different lengths but should still be considered similar.

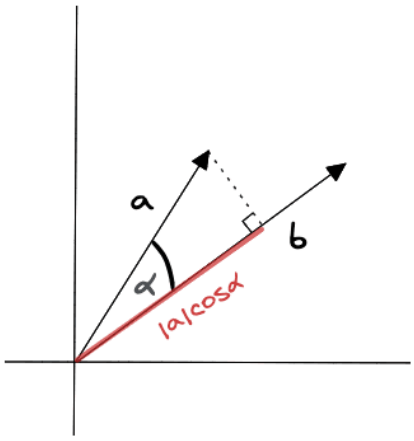

Figure 1.9 Dot Product Similarity

- Dot Product Similarity:
  - Concept: The dot product measures the projection of one vector onto another. It is influenced by both the angle between the vectors and their magnitudes.
  - Calculation: For vectors a = (a_1, a_2, ..., a_n) and b = (b_1, b_2, ..., b_n), the dot product is:

$$a \cdot b = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + a_3 b_3 + \ldots + a_n b_n$$

  - Interpretation:

- A larger positive dot product generally indicates greater similarity.
- A dot product of 0 indicates orthogonality.
- A negative dot product indicates that the vectors point in generally opposite directions. Unlike cosine similarity, the dot product is not normalized by the vector magnitudes, so longer vectors with the same orientation will have a larger dot product.
- Use Case: Can be effective when the magnitude of the vectors carries meaningful information in addition to their orientation. It's computationally simpler than cosine similarity if vectors are already normalized (in which case it becomes equivalent to cosine similarity).

Chapter 2: CHATBOT SYSTEM

2.1. Overview of the Chatbot System

The system enables learners to ask natural language questions related to course content. Their queries are processed through an AI pipeline that classifies, routes, and answers based on either vector-based document retrieval or lesson data. The core AI-powered functionality involves understanding learner queries, retrieving relevant data, and generating human-like responses using the LLM model and supporting components.

2.2. System Architecture

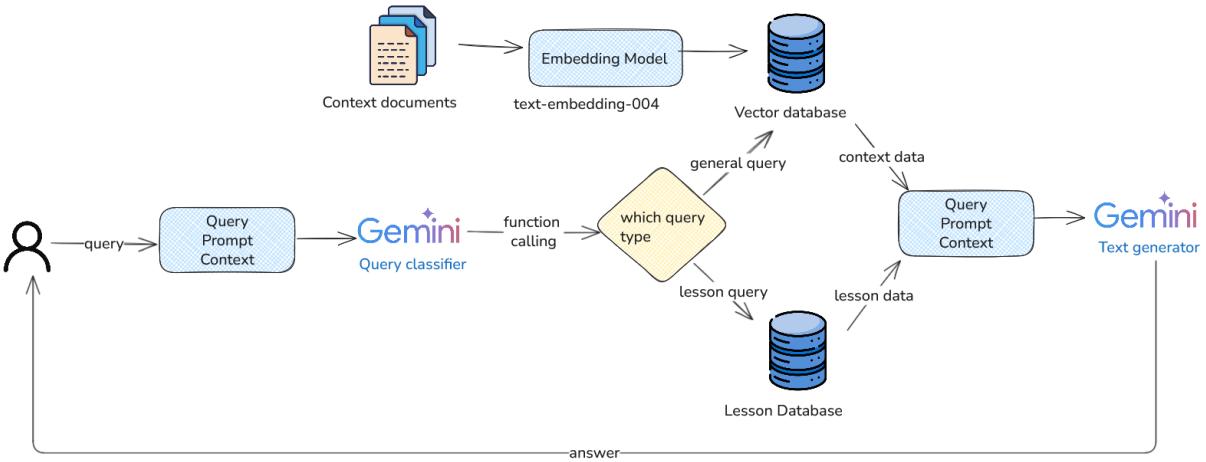

Figure 2.1 AI chatbot architecture

The architecture comprises the following key components:

- **Query Classifier**
  This component determines the intent of the learner's question (e.g., general or lesson-specific). Powered by Gemini, it was selected for this critical classification task due to several key advantages: its free API access makes it cost-effective for educational applications, while still delivering robust natural language understanding capabilities that are more than sufficient for intent classification. The model demonstrates strong performance in parsing educational queries and distinguishing between different question types,

whether students are asking broad conceptual questions or seeking clarification on specific lesson content. Additionally, Gemini's reliable response times and stable API ensure consistent user experience, while its ability to handle diverse question formats and educational contexts makes it well-suited for the varied ways students might phrase their inquiries. The combination of zero cost, dependable performance, and adequate sophistication for classification tasks makes Gemini an optimal choice for this component, allowing resources to be allocated to other aspects of the learning platform while maintaining effective query routing and intent recognition.

- **Query Prompt Context**

  Components of *Query Prompt Context:*
  - User's current query: The latest question or request made by user.
  - System prompt: predefined instruction or behavior guidance
  - Conversation history: Recent dialogue turns to preserve context, coherence, and continuity in the conversation.
  - Additional retrieved context: This might be from a vector DB (semantic search) or the lesson database, depending on query type.
- **Embedding Model (text-embedding-004)** [6] [7]

  This model converts context documents into embeddings and stores them in a vector database for similarity-based search.
- **Vector Database**

  Stores vector representations of educational content and is queried for contextually relevant information.
- **Lesson Database**

  Stores structured lesson-related content for direct retrieval when the query targets specific lessons.
- **Text Generator (Gemini)**

  Generates answers based on the retrieved context and original learner query.

2.3. Query flow and Handling

2.3.1. User query input

The learner inputs a question via the frontend interface. The question is wrapped into a Query Prompt Context and passed to the Gemini model for classification.

2.3.2. Query classification

LLM analyzes the query and determines its type through a function-calling [8] mechanism:

- **General query**: Related to broad concepts of knowledge.
- **Lesson query**: Specific to a course lesson's content.

2.3.3. Context retrieval

- **General Queries**: The system searches the Vector Database for semantically relevant documents using embeddings generated by the embedding model. The top-matching results are returned as context.
- **Lesson Queries:** The system fetches the relevant lesson data directly from the Lesson Database.

2.3.4. Answer generation

The context data, combined with the original query, is passed into a second *Query Prompt Context* which is processed by LLM. LLM generates a coherent, contextually accurate response that is returned to the learner.

2.4. Technologies and Parameters

Table 2.1 Chatbot technologies and configuration parameters

Component	**Technology**	**Configuration Parameter**
LLM	Gemini-2.0-flash-lite	
Embedding Model	Text-embedding-004	dim=512
Database	Postgres	
Vector search	Pgvector	Similarity Metric = L2 distance Threshold= 0.7

2.5. Evaluate system

2.5.1. Overview

The evaluation system checks if the AI chatbot is working correctly by testing two main areas: answer quality, system speed. We use simple tests and measurements to make sure learners get helpful responses quickly.

2.5.2. Test overview

2.5.2.1. Answer quality

- **Correct Information**: Are the answers factually right?
    - Target: 90% of answers should be correct
    - How we check: Compare answers with course materials
- **Right Topic**: Does the chatbot understand what learners are asking about?
    - Target: 95% correct understanding
    - How we check: Test with sample questions

2.5.2.2. System performance

- **Response Speed**: How fast does the chatbot answer?
    - Target: Less than 3 seconds for most questions
    - How we check: Automatic timing measurement
- **System Uptime**: Is the chatbot available when students need it?
    - Target: Works 99% of the time
    - How we check: Monitor system availability

2.6. Advantages of the system

- **Natural Interaction**: Learners engage through conversational queries without needing structured inputs.
- **High Accuracy**: Embedding-based search improves retrieval precision for general queries.
- **Real-time Feedback**: Code compilation and instant answers streamline the learning process.
- **Scalability**: The architecture allows for expanding the context base and integrating other LLMs in the future.

Chapter 3: SYSTEM ANALYSIS AND DESIGN

3.1. Actors

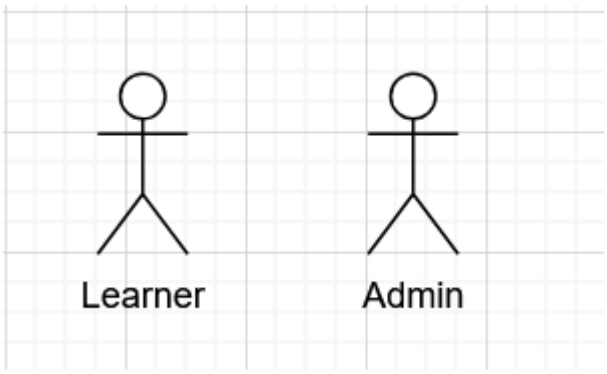

Figure 3.1: Actors in system

The application is designed with two primary actors.

- Learner: Uses the platform to access courses, complete exercises, earn rewards, and get AI-powered help to improve programming skills.
- Admin: Manages courses, exercises, and users, monitors progress and statistics, and maintains the overall platform operation.

3.1.1. Actor "Learner"

Learners can use the application with the following features:

- **Register/ log in/ verify account/ forgot password:** To use the service, users must register a new account, log in, and verify their identity. They can also reset their password in case it is forgotten.
- **User profile management:** Users can update their own profiles and view the profiles of other learners.
- **Learn course:** Learners can access and study various courses that are structured to support progressive learning.
- **Do exercise:** During the lesson, users can practice through interactive exercises to reinforce their understanding, users can open solution if they get stuck.

- **Interact with chatbot:** A built-in chatbot assists learners by answering questions about the website, explaining lessons, and providing guidance during their learning journey.
- **Use item:** Learners can use collected or purchased items to enhance their learning experience or unlock special features.
- **Shopping**: A virtual shop is available where users can purchase in-app items using earned or purchased currency.
- **Level up:** As learner's complete activities and gain experience points, they can level up to unlock new content and features.
- **Gain achievement:** Users can earn achievements for completing milestones, encouraging continued engagement and motivation.
- **Use spell book:** The spell book feature allows learners to quickly review summarized knowledge from the lessons they have completed.
- **Give feedback:** Users can submit feedback about lessons, features, or overall application experience to help improve the system.
- **Get completed course certificate:** Upon finishing a course, learners can receive a certificate of completion as recognition of their efforts.

3.1.2. Actor "Admin"

- **Application status statistics:** Administrators can monitor the overall status and performance of the application. This includes tracking user activity, course participation, system usage metrics.
- **Course and exercise management:** Administrators can create, update, and delete courses and exercises. They define the course structure, lesson content, and configure exercise types to ensure a comprehensive learning path.
- **Spell book management:** Admins manage the creation and association of spell books with corresponding lessons. They can edit or update summaries to ensure accurate and helpful review content for learners.
- **Daily reward management:** Administrators configure the daily reward system, including reward types, chest drop rates, and item probabilities. This encourages consistent user engagement through gamification.
- **Shop and item management:** Admins control the shop system by adding, editing, or removing items. They define item categories, prices, and effects to balance in-app economy and user progression.

- **Global notification:** The administrator can send system-wide notifications to users, such as platform updates, announcements, or event alerts. Notifications can be scheduled or sent in real time.
- **Feedback management:** Admins have access to user-submitted feedback, allowing them to monitor user satisfaction, identify issues, and plan future improvements to the platform. For admins, this like a to-do lists.

3.2. Course platform structure

3.2.1. Course structure

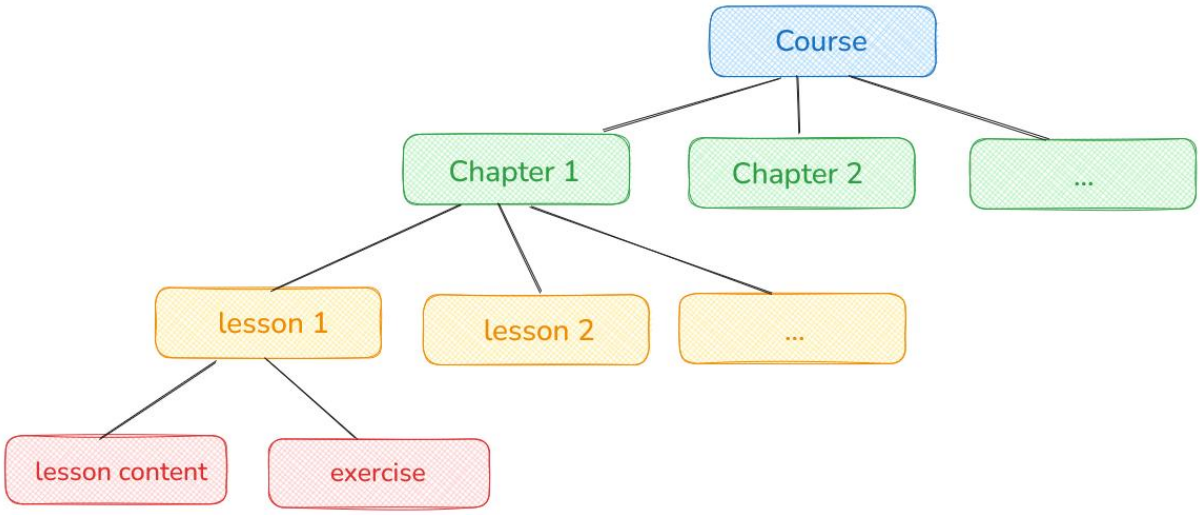

Figure 3.2 Course structure

The course is organized in a **hierarchical structure** with multiple levels, designed to guide learners through a logical and progressive learning path:

1. **Course:** The top level represents the entire course. A course is a complete learning unit that contains multiple chapters grouped by topic or theme.
2. **Chapter:** Each course consists of several chapters. A chapter serves as a major section within the course, grouping together related lessons. This helps segment the course into manageable learning phases.
3. **Lesson:** Within each chapter, there are multiple lessons. A lesson is the smallest standalone learning unit and includes specific topics or skills the learner needs to acquire. Each lesson contains two main components:

    - **Lesson Content**: The instructional material or theory part, such as text, images, videos, or interactive elements.

- **Exercise**: A set of practice questions or interactive tasks that allow learners to apply what they have just learned.

3.2.2. Exercise type

To enhance learner engagement and assess understanding, the platform supports three main types of exercises:

- **Coding exercise:**
  - This exercise type allows learners to write, edit, and run code directly within the platform.
  - The coding environment supports multiple file management, enabling users to simulate real-world coding scenarios.
  - Learners can compile and execute code inside the web interface, with instant feedback provided based on test cases or expected output.
- **Multiple Choice Exercise:**
  - Learners are presented with a question and a list of predefined answer choices.
  - They must select the correct answer(s) from the options.
- **Question and Answer Exercise:** This format allows learners to submit free-text responses to open-ended questions.

3.3. Use cases

3.3.1. General use case

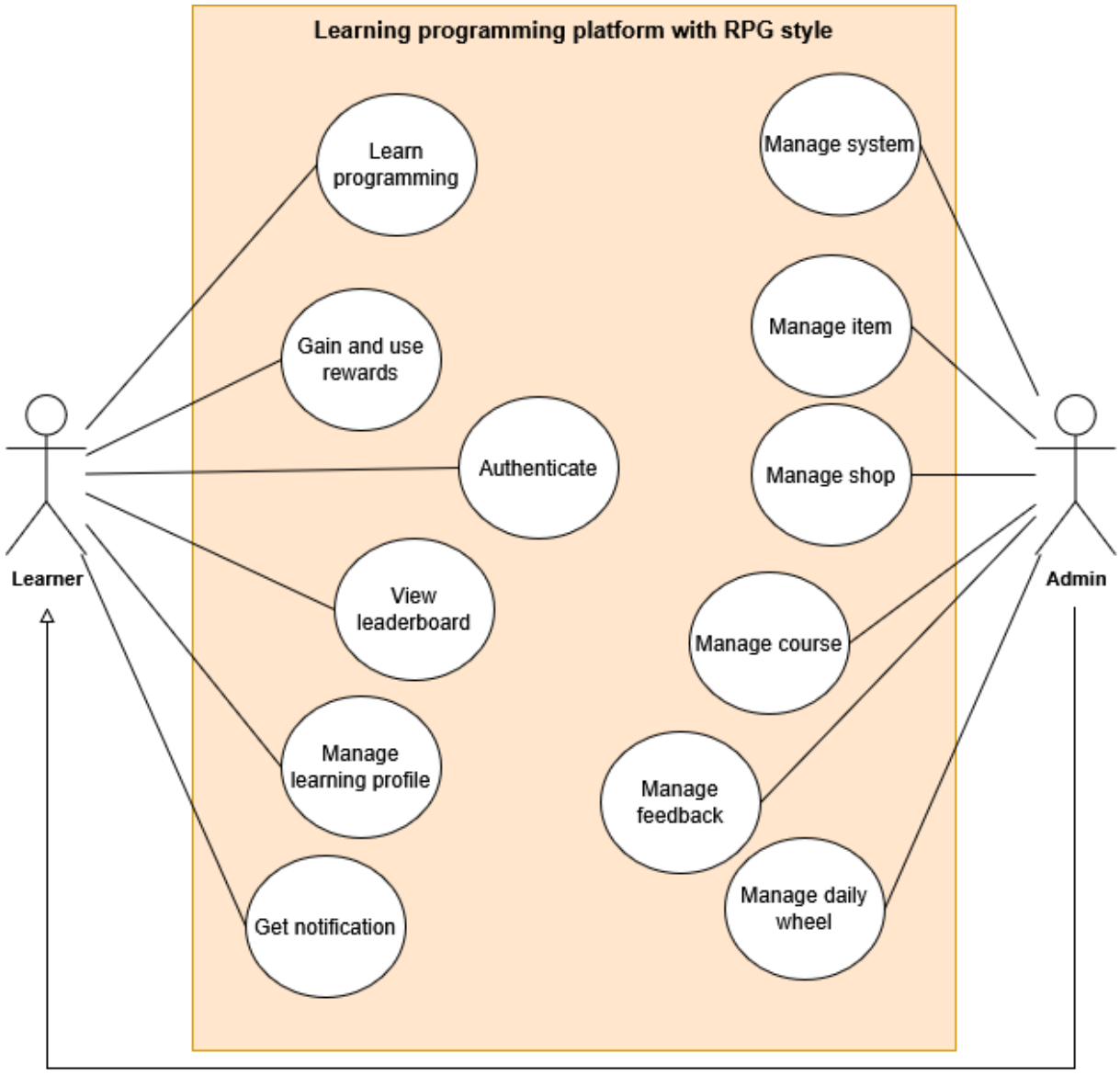

Figure 3.3 General use case diagram

3.3.2. Learner

3.3.2.1. Use case Authenticate

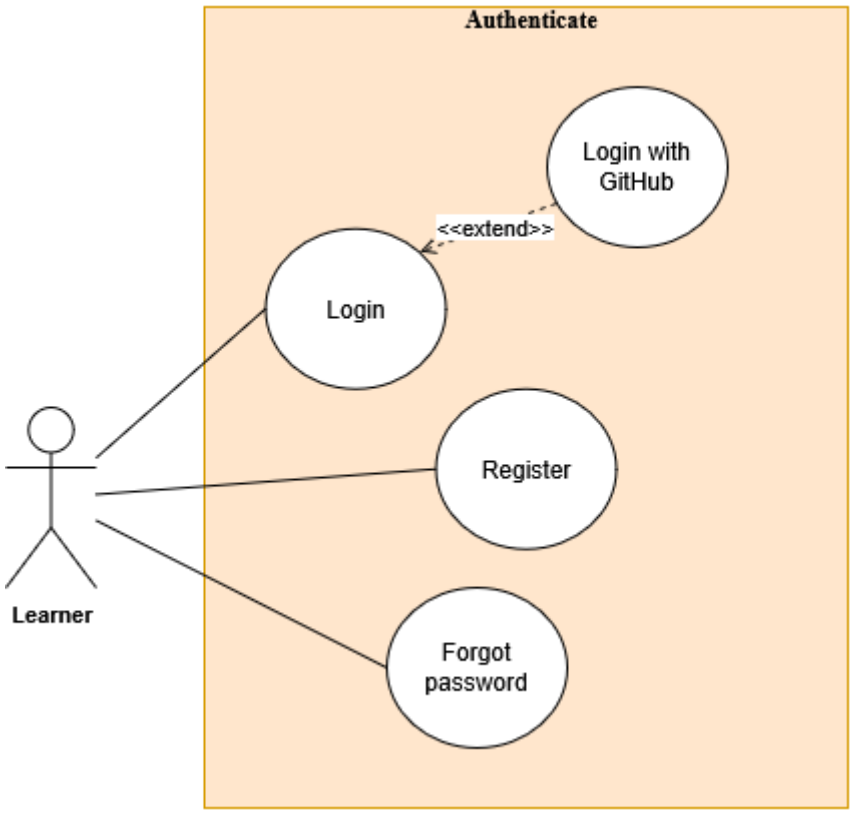

Figure 3.4 Use case Authenticate

Table 3.1 Use case Login

Use Case ID	**UC01**
Use Case Name	Login with username/password
Description	As a user want to login to the platform with their own credentials
Actor(s)	Learner, Admin
Trigger	User goes to the landing page. Click **Sign in** button
Pre-Condition(s)	- User already input their credentials to the input field - User information already signed up before
Post-Condition(s)	User login successfully
Basic flow	1. User go to landing page

	2. Click the **Sign in** button 3. User input the credentials 4. System validates the information 5. Redirect user to the homepage
Exception flow	- The system validates the login failed and displays a message - The user chooses to cancel login button

Table 3.2 Use case Login/Register with GitHub

Use Case ID	**UC02**
Use Case Name	Login/Register with GitHub
Description	As a user want to login/register to the platform with their GitHub account
Actor(s)	Learner
Trigger	Learner goes to the landing page. Click **Sign in with GitHub** button
Pre-Condition(s)	Learner has GitHub account.
Post-Condition(s)	Learner login/register successfully
Basic flow	1. Learner go to landing page 2. Click the **Sign in with GitHub** button 3. System redirect user to GitHub page to accept the policies (if sign in for the first time) 4. System validates the information 5. Redirect user to the homepage
Exception flow	- Learner not accept the GitHub policies to sign up.

Table 3.3 Use case register

Use Case ID	**UC03**
Use Case Name	Register new account
Description	As a user, I want to create a new account
Actor(s)	Learner
Trigger	Learner goes to the landing page. Click **Sign up** button
Pre-Condition(s)	The account does not exist in the system yet
Post-Condition(s)	Learner registers successfully
Basic flow	1. Learner goes to landing page 2. Click the **Sign up** button 3. Learner enters the need of information 4. System validates user information 5. System sends verify code to learner email 6. Learner enter the verification code 7. Redirect learner to the homepage
Exception flow	- Learner enters wrong verify code - Learner register account that already exist in the system - Learner credentials does not meet the system requirement
Non-Functional Requirement	- The learner's password must be hashed in database - Password has at least 8 characters or numbers and one special case character

Table 3.4 Use case Forget password

Use Case ID	**UC04**
Use Case Name	Forgot password
Description	As a learner, I want to reset my password so that I can continue access my account if I'm forget my password
Actor(s)	Learner, Admin
Trigger	Learner goes to the landing page. Click f**orget password** button
Pre-Condition(s)	Learner already has the account in system
Post-Condition(s)	Change user password successfully
Basic flow	1. Learner goes to the landing page 2. Click the **forget password** button 3. Learner enters their email, click **Send OTP** button 4. Learner enters the verification code 5. Enter new password 6. System validates new information 7. Redirect learner to login page
Exception flow	- Learner enter email that not have in the system - Learner enter wrong verification code

3.3.2.2. Use case Learn programming

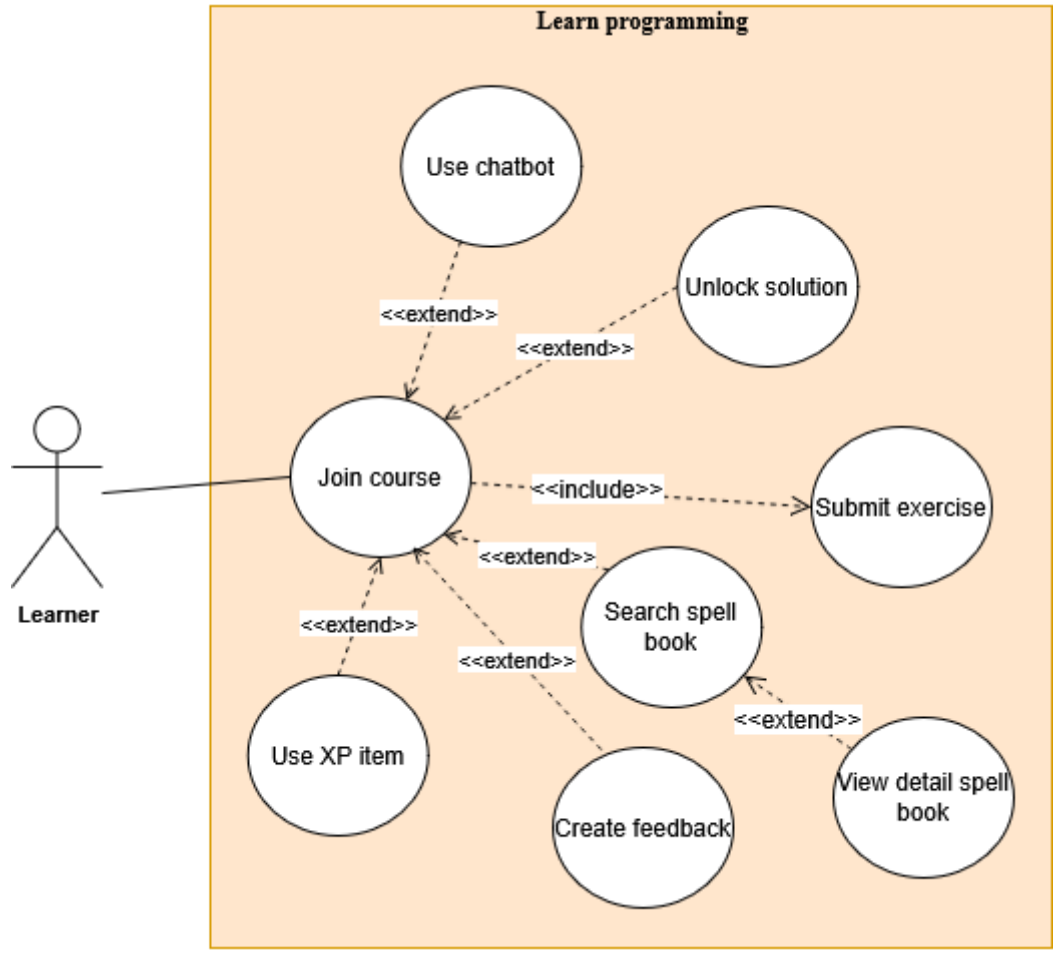

Figure 3.5 Use case Learning programming

Table 3.5 Use case Join course

Use Case ID	**UC05**
Use Case Name	Join course
Description	As a learner, I want to join a course to start learning and access its content and exercises
Actor(s)	Learner
Trigger	Learner navigates to the course list and selects a course to join or enroll in
Pre-Condition(s)	- Leaner is logged into the platform - The desired course exists in the system

Post-Condition(s)	- Learner is enrolled in the course. - Learner can access course materials (lessons, exercises)
Basic flow	1. Learner finds and selects a course 2. Learner clicks the **Enter code** button 3. System verifies learner eligibility 4. System redirect use to the nearest unlearn lesson

Table 3.6 Use case Submit exercise

Use Case ID	**UC06**
Use Case Name	Submit exercise
Description	As a learner, I want to submit the exercise answer to complete the lesson
Actor(s)	Learner
Trigger	Learner clicks **Submit** button
Pre-Condition(s)	- Leaner is logged into the platform - The desired lesson exists in the system
Post-Condition(s)	- Learner stats updated - Learner course progress update
Basic flow	1. Learner submits the exercise 2. System verifies learner answer 3. System update learner information 4. Notify back to learner about the lesson status
Alternative flow 1	2.1 System check learner's answer is incorrect or not. If yes, move to step

	4
Alternative flow 2	2.2 System check if learner use the free unlock solution or not. If yes, move to step 3
Alternative flow 3	2.3 System check if learner already submit the exercise or not. If yes, move to step 4
Exception flow	In step 2.1, if learner's answer is incorrect, give the error

Table 3.7 Use case Unlock solution

Use Case ID	**UC07**
Use Case Name	Unlock solution
Description	As a learner, I want to unlock solution if I can't solve the exercise, so that I can understand the lesson clearly and move to next one
Actor(s)	Learner
Trigger	Learner clicks **Solution** button
Pre-Condition(s)	- Leaner is logged into the platform - Learner is in lesson detail page
Post-Condition(s)	Solution unlocked
Basic flow	1. Learner clicks **Solution** button 2. System checks if learner can open solution or not 3. Open solution for user
Alternative flow 1	2.1 If learner is not complete lesson and have item to unlock, open modal asks if user want to spend item to open 4. User accept to use item. Move to step 3
Alternative flow 2	2.2 If learner is not complete lesson and learner not have item to unlock, open modal asks if leaner want to buy and use or use free unlock solution or

	not 4.1 Learner accept to use buy and use item. Move to step 3
Alternative flow 3	2.3 If leaner is not complete lesson and leaner not have item to unlock, open modal asks if leaner want to buy and use or use free unlock solution or not 4.2 Learner accept to use free unlock solution. Move to step 3
Exception flow	In step 4.1, if learner not have enough money to buy, send the error out

Table 3.8 Use case Use chatbot

Use Case ID	**UC08**
Use Case Name	Unlock chatbot
Description	As a learner, I want to use chatbot if I can't understand the lesson, so that I can ask and get a deep understanding in the lesson
Actor(s)	Learner
Trigger	Learner clicks **Chatbot** button
Pre-Condition(s)	- Leaner is logged into the platform - Learner is in lesson detail page
Post-Condition(s)	Chatbot unlocked
Basic flow	1. Learner click **Chatbot** button 2. System checks if user can open chatbot or not 3. Open chatbot for user
Alternative flow 1	2.1 If learner is not complete lesson and learner have item to unlock, open modal asks if learner want to spend item to open 4. Learner accept to use item.

	Move to step 3
Alternative flow 2	2.2 If learner is not complete lesson and learner not have item to unlock, open modal asks if learner want to buy and use item or not 4.1 Learner accept to use buy and use item. Move to step 3
Exception flow	In step 4.1, if learner not have enough money to buy, send the error out

Table 3.9 Use case Use XP item

Use Case ID	**UC09**
Use Case Name	Use XP item
Description	As a learner, I want get bonus experiences when finish lesson, so that I can level up faster
Actor(s)	Learner
Trigger	Learner clicks **use** button
Pre-Condition(s)	- Leaner is logged into the platform - Learner is in lesson detail page - Learner has XP item to use
Post-Condition(s)	Notify learner use XP item successfully
Basic flow	1. Learner clicks the XP item icon 2. System pops up the XP item description modal 3. Learner clicks **use** button 4. System verifies user and notifies learner has use item successfully

Table 3.10 Use case Search spell book

Use Case ID	**UC10**
Use Case Name	Search spell book
Description	As a learner, I want a feature to search my learned concept but no need to find out in previous lesson, so that I can save my time.
Actor(s)	Learner
Trigger	Learner starts to searching
Pre-Condition(s)	- Leaner is logged into the platform - Learner is in lesson detail page - Learner is in spell book component - Learner completes at least one lesson that have spell book support
Post-Condition(s)	Spell book title list response
Basic flow	1. Learner search spell book keyword in search input 2. System checks if there is any learner's spell book available 3. Response spell book title for learner

Table 3.11 Use case View detail spell book

Use Case ID	**UC11**
Use Case Name	View detail spell book
Description	As a learner, I want to view detail the spell book so that I can review learned my knowledge easily
Actor(s)	Learner
Trigger	Learner clicks any spell book in spell book list
Pre-Condition(s)	- Leaner is logged into the platform

	- Learner is in lesson detail page - Have a list of learner's spell book appear
Post-Condition(s)	Detail spell book data response
Basic flow	1. Learner clicks any spell book title in spell book list 2. System get detail of the spell book 3. Show spell book detail to learner

Table 3.12 Use case Send feedback

Use Case ID	**UC12**
Use Case Name	Send feedback
Description	As a learner, I want to send my feedback about the course I'm learning, so that the platform owner can know the issue as soon as possible
Actor(s)	Learner
Trigger	Learner clicks **Send** button
Pre-Condition(s)	- Learner is logged into the platform - Learner is in the feedback page
Post-Condition(s)	System saves new feedback of user successfully
Basic flow	1. Learner chooses the feedback type 2. Learner input their feedback into the input field and click **Save** button 3. System saves new feedback of user

3.3.2.3. Use case Gain and use rewards

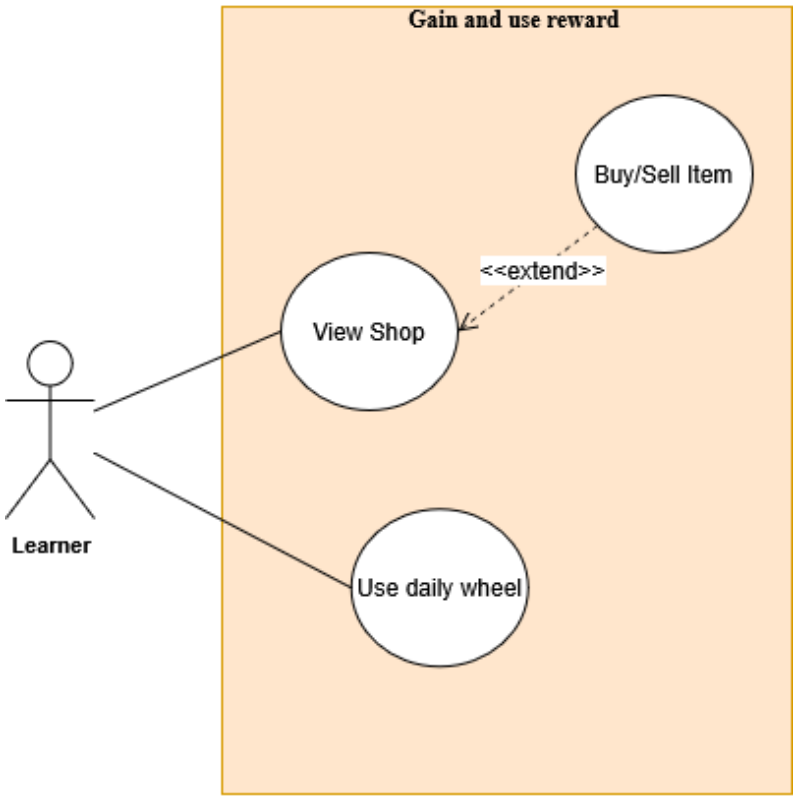

Figure 3.6 Use case Gain and use reward

Table 3.13 Use case View shop

Use Case ID	**UC13**
Use Case Name	View shop
Description	As a learner, I want to view shop so that I can know what is price of the item I want to buy/sell so that it can support my learning path
Actor(s)	Learner
Trigger	The learner click **Shop** icon in the user header
Pre-Condition(s)	- Learner is logged into the platform
Post-Condition(s)	Show shop information successfully
Basic flow	1. User click **Shop** icon in the user

	header 2. System response with item description and price information
Alternative flow 1	2.1 If item is not in shop but learner have it, system show that item and learner' quantity of that item, not show the buy/sell button
Alternative flow 2	2.2 If item is not in shop but learner not have it, not show that item information in shop

Table 3.14 Use case Buy/Sell item

Use Case ID	**UC14**
Use Case Name	Buy/Sell item
Description	As a learner, I want to buy or sell item in shop so that it can support my learning path
Actor(s)	Learner
Trigger	Learner clicks **Sell/Buy** button on that item
Pre-Condition(s)	- Learner is logged into the platform - Learner is in shop screen
Post-Condition(s)	System updates new information successfully
Basic flow	1. Learner clicks **Sell/Buy** button 2. System update with new information of learner 3. System response new information to learner

Table 3.15 Use case Use daily wheel

Use Case ID	**UC15**
Use Case Name	Use daily wheel
Description	As a learner, I want to use the daily wheel so that I can get more reward when complete lesson
Actor(s)	Learner
Trigger	Learner spins the wheel
Pre-Condition(s)	- Learner is logged into the platform - Learner is complete the wheel target of the day - First time use the wheel in the day - Learner is in the spin wheel page
Post-Condition(s)	Update new learner information with the reward
Basic flow	1. Learner clicks the wheel to spin 2. The wheel spinning 3. System show reward for learner 4. System updates new learner information with the reward

3.3.2.4. Use case Manage learning profile

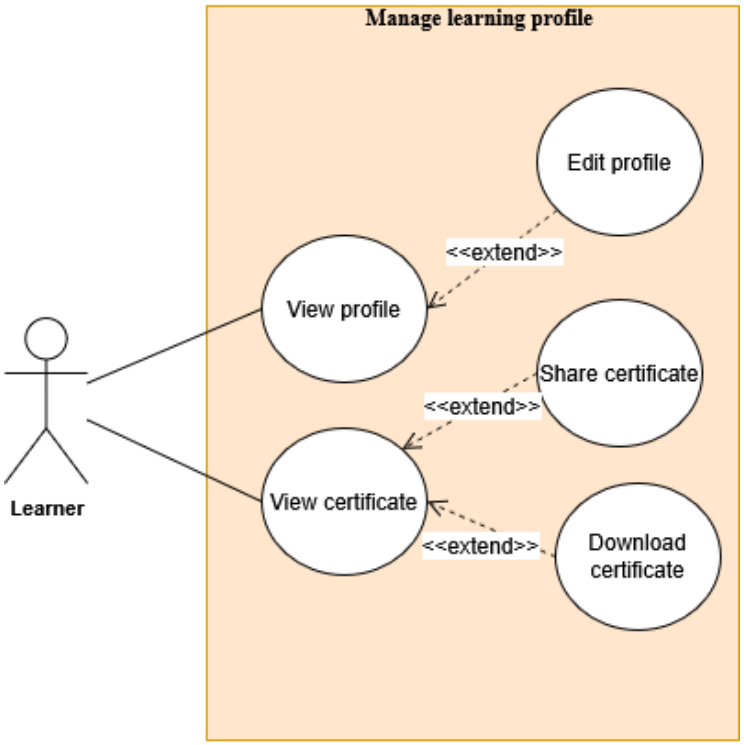

Figure 3.7 Use case Manage learning profile

Table 3.16 Use case Manage profile

Use Case ID	**UC16**
Use Case Name	Manage profile
Description	As a learner, I want to modify my information so that I can make up my profile
Actor(s)	Learner
Trigger	- Learner uploads new avatar - Learner updates their information
Pre-Condition(s)	- Learner is logged into the platform - Learner is in profile page
Post-Condition(s)	System updates new user information
Basic flow	1. Learner updates their information 2. Learner clicks **Save** button

Alternative flow	1.1 Learner upload new avatar
Non-functional requirements	Upload file must be in image format

Table 3.17 Use case View certificate

Use Case ID	**UC17**
Use Case Name	View certificate
Description	As a learner, I want to see my certificate after I finish the course, so that I have something to prove that I already finish it
Actor(s)	Learner
Trigger	Learner clicks the completed course image in their profile
Pre-Condition(s)	- Learner is logged into the platform - Learner is in their own profile page - Learner already finish at least one course
Post-Condition(s)	Show the certificate
Basic flow	1. Learner clicks the completed course in learner profile 2. System redirect user to the learner certificate course page
Non-functional requirements	Only learner can see their own course certificate

Table 3.18 Use case Download certificate

Use Case ID	**UC18**
Use Case Name	Download certificate
Description	As a learner, I want to download my course certificate so that I can use it to make up my portfolio
Actor(s)	Learner

Trigger	Learner clicks **Download certificate** button
Pre-Condition(s)	- Learner is logged into the platform - Learner is in certificate page
Post-Condition(s)	Send a download certificate request to learner
Basic flow	1. Learner clicks the **Download certificate** button 2. System sends a download request to learner

Table 3.19 Use case Share certificate

Use Case ID	**UC19**
Use Case Name	Share certificate
Description	As a learner, I want to share my certificate to another social network, so that I can prove that I already finish the course
Actor(s)	Learner
Trigger	Learner clicks **Share certificate** button
Pre-Condition(s)	- Learner is logged into the platform - Learner is in certificate page
Post-Condition(s)	Copy public certificate URL to learner clipboard
Basic flow	1. Learner clicks **Share certificate** button 2. System saves public certificate URL to learner clipboard

3.3.2.5. Use case View leaderboard

Table 3.20 Use case View leaderboard

Use Case ID	**UC20**
Use Case Name	View leaderboard
Description	As a learner, I want to download my course certificate so that I can use it to make up my portfolio
Actor(s)	Learner
Trigger	Learner clicks **Download certificate** button
Pre-Condition(s)	- Learner is logged into the platform - Learner is in certificate page
Post-Condition(s)	Send a download certificate request to learner
Basic flow	1. Learner clicks the **Download certificate** button 2. System sends a download request to learner

3.3.2.6. Use case Get notification

Table 3.21 Use case Get notification

Use Case ID	**UC21**
Use Case Name	Get notification
Description	As a learner, I want to get notification from this platform so that I can catch up with all new events or track my progress inside the platform
Actor(s)	Learner
Trigger	Learner already login/ Click **notification** icon
Pre-Condition(s)	- Learner is logged into the platform

Post-Condition(s)	Receive newest notification about their progress or new platform events
Basic flow	1. Learner takes some action in the platform 2. System notices the action should be notify or not.
Alternative flow	1.1 Admin create new global notification

3.3.3. Admin

3.3.3.1. Use case Manage system

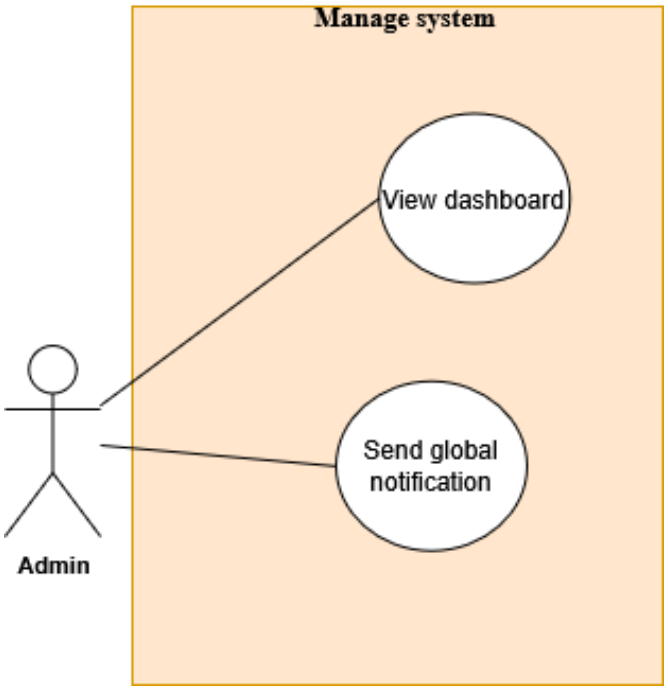

Figure 3.8 Use case Manage system

Table 3.22 Use case View dashboard

Use Case ID	**UC22**
Use Case Name	View dashboard
Description	As an admin, I want to
Actor(s)	Learner
Trigger	Learner clicks **Download certificate** button
Pre-Condition(s)	- Learner is logged into the

	platform - Learner is in certificate page
Post-Condition(s)	Send a download certificate request to learner
Basic flow	3. Learner clicks the **Download certificate** button 4. System sends a download request to learner

Table 3.23 Use case Send global notification

Use Case ID	**UC23**
Use Case Name	Send global notification
Description	As an Admin, I want to send global notification to all user, so that I can make sure all active user can receive latest events of the platform
Actor(s)	Admin
Trigger	Admin click **Send** button
Pre-Condition(s)	- Admin is logged into the platform - Admin is inside the send global notification page
Post-Condition(s)	System saves all notifications and sends notification to all active user
Basic flow	1. Admin input the notification content 2. Admin click **Send** button 3. System saves all notifications 4. System sends notification to all active user

3.3.3.2. Use case Manage course

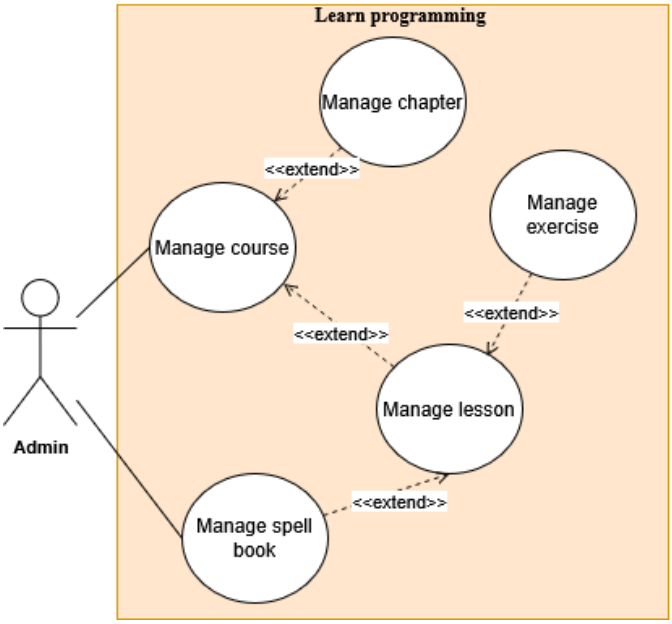

Figure 3.9 Manage course

Table 3.24 Use case Manage course

Use Case ID	**UC24**
Use Case Name	Manage course
Description	As an admin, I want to manage my course
Actor(s)	Admin
Trigger	Admin click **Update** button
Pre-Condition(s)	- Admin is logged into the platform - Admin is in admin course detail page
Post-Condition(s)	Notify update new course information successfully
Basic flow	1. Admin update new course information 2. System validates the information 3. System saves new course information

Table 3.25 Use case Manage chapter

Use Case ID	**UC25**
Use Case Name	Manage chapter
Description	As an admin, I want to manage course's chapter
Actor(s)	Admin
Trigger	Admin click **Add/ Update/ Delete** button Admin drag chapter position
Pre-Condition(s)	- Admin is logged into the platform - Admin is in admin chapter list page
Post-Condition(s)	Notify update chapter information successfully
Basic flow	1. Admin update/ add/ delete/ drag position chapter 2. System validates the information 3. System saves course information

Table 3.26 Use case manage lesson

Use Case ID	**UC26**
Use Case Name	Manage lesson
Description	As an admin, I want to manage chapter's lesson
Actor(s)	Admin
Trigger	Admin click **Add/ Update/ Delete** button
Pre-Condition(s)	- Admin is logged into the platform - Admin is in admin lesson detail page
Post-Condition(s)	Notify update lesson information successfully

Basic flow	1. Admin update/ add/ delete lesson 2. System validates the information 3. System saves course information
Alternative flow	1.1 Admin update/add spell book 1.2 Admin update/add exercise Move to step 2

Table 3.27 Use case Manage exercise

Use Case ID	**UC27**
Use Case Name	Manage exercise
Description	As an admin, I want to manage lesson's exercise
Actor(s)	Admin
Trigger	Admin click **Save** button
Pre-Condition(s)	- Admin is logged into the platform - Admin is in admin lesson detail page
Post-Condition(s)	Notify update lesson information successfully
Basic flow	1. Admin update lesson exercise information 2. System validates the information 3. System saves course information
Non-functional requirements	Can't update to another exercise type

Table 3.28 Use case Manage spell book

Use Case ID	**UC28**
Use Case Name	Manage spell book
Description	As an admin, I want to manage spell book so that I can change the content inside it
Actor(s)	Admin
Trigger	The admin click **Save** button in admin's spell book detail page
Pre-Condition(s)	- Admin is logged into the platform - Admin is in admin's spell book page
Post-Condition(s)	System updates new information successfully
Basic flow	1. Admin search for the spell book title 2. System response a list of spell book 3. Admin click the spell book title from the spell book list to see detail 4. Admin update new information of that spell book 5. Admin click **Save** button 6. System update with new spell book information

3.3.3.3. Use case Manage item

Table 3.29 Use case Manage item

Use Case ID	**UC29**
Use Case Name	Manage item
Description	As an admin, I want to manage item so that I can change the stats of each item
Actor(s)	Admin
Trigger	The admin click **Save** button on the item
Pre-Condition(s)	- Admin is logged into the platform - Admin is in admin's item page
Post-Condition(s)	System updates new information successfully
Basic flow	1. Admin click **Update** icon on the item to change it stats 2. System pops up item update modal 3. Admin input new item information 4. Admin click **Save** button on the item to save the changed 5. System update with new item information

3.3.3.4. Use case Manage shop

Table 3.30 Use case Manage shop

Use Case ID	**UC30**
Use Case Name	Manage shop
Description	As an admin, I want to manage which item should be displayed in shop and which prices is suitable for it by the time so that I can improve my platform
Actor(s)	Admin

Trigger	Admin click **Save** button on the item
Pre-Condition(s)	- Admin is logged into the platform - Admin is in admin shop page
Post-Condition(s)	System updates new information successfully
Basic flow	1. Admin clicks **Edit Prices** on the item 2. Admin update new prices for the item 3. System update with new item information
Exception flow	In step 3, system will response error if user not input correct prices format

3.3.3.5. Use case Manage Daily wheel

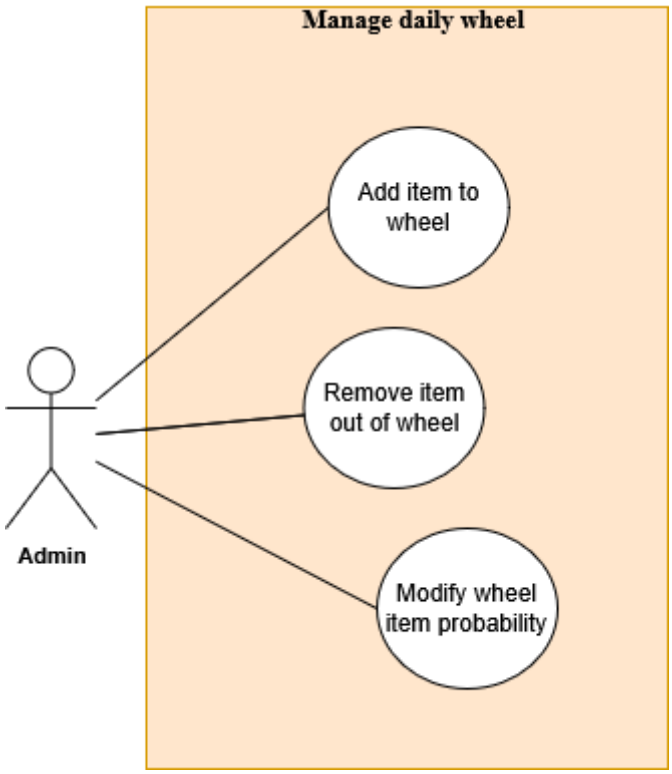

Figure 3.10 Use case Manage daily wheel

Table 3.31 Use case Manage daily wheel

Use Case ID	**UC31**
Use Case Name	Manage daily wheel
Description	As an admin, I want to manage which item should be displayed or not in wheel so that I can adjust the wheel more flexible
Actor(s)	Admin
Trigger	Admin click an item that he want to action with
Pre-Condition(s)	- Admin is logged into the platform - Admin is in admin wheel page
Post-Condition(s)	System updates new information successfully
Basic flow	4. Admin pick an item he wants to action with 5. System updates new wheel information
Alternative flow 1	1.1 Admin pick an item from the item list add to wheel
Alternative flow 2	1.2 Admin pick an item from the wheel item list to remove from wheel
Exception	Server send error if non-functional requirement doesn't meet
Non-Functional Requirement	- Can't add an item to the list if it is the same as both the first and last item in the wheel. - Can't add an item that already have in the wheel

Table 3.32 Use case Change wheel possibilities

Use Case ID	**UC32**
Use Case Name	Change wheel possibilities
Description	As an admin, I want to manage the possibilities all the item in wheels, so that I can adjust the wheel more flexible
Actor(s)	Admin
Trigger	Admin click on the item he wants to edit possibilities
Pre-Condition(s)	- Admin is logged into the platform - Admin is in admin wheel page
Post-Condition(s)	System updates new information successfully
Basic flow	1. Admin pick the item he wants to change it possibilities. 2. Admin input new possibilities 3. System re-config all remain wheel item possibilities 4. System updates new information about the wheel
Non-Functional Requirement	If you edit the probabilities of multiple items at once, their total must equal 100%

3.3.3.6. Manage feedback

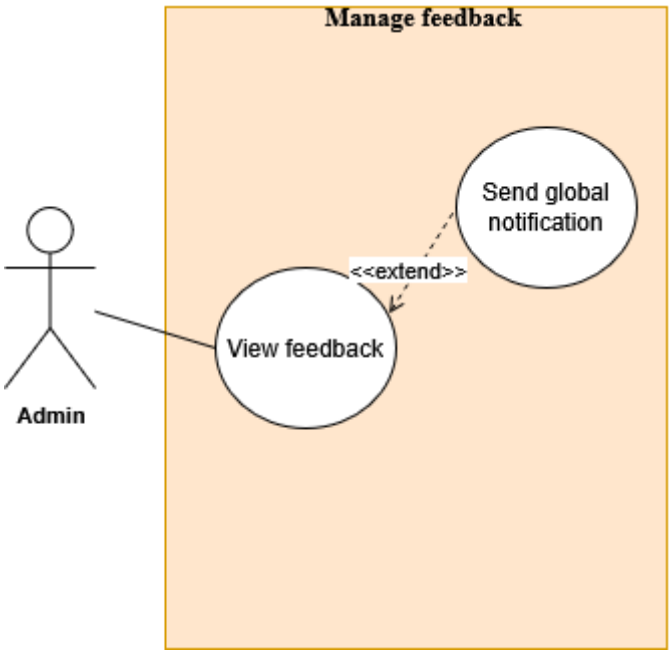

Figure 3.11 Use case Manage feedback

Table 3.33 Use case View feedback

Use Case ID	**UC33**
Use Case Name	View feedback
Description	As an admin, I want to see all learner feedback so that I can know what to improve my website performance
Actor(s)	Admin
Trigger	Admin navigate to admin feedback page
Pre-Condition(s)	- Admin is logged into the platform - Admin is in admin homepage
Post-Condition(s)	System shows all learner feedbacks
Basic flow	1. Admin navigate to admin feedback page 2. System shows all learner feedbacks
Alternative flow	1.1 Admin filter the feedback base on their types

Table 3.34 Use case Resolve feedback

Use Case ID	**UC34**
Use Case Name	Resolve feedback
Description	As an admin, I want to know which learner feedbacks I already resolved, which is not, so that I can manage my website more effectively
Actor(s)	Admin
Trigger	Admin click checkbox on specific feedback
Pre-Condition(s)	- Admin is logged into the platform - Admin is in admin feedback page
Post-Condition(s)	System updates new information successfully
Basic flow	1. Admin click the feedback checkbox to resolve 2. System updates newest information

3.4. Activity diagrams

3.4.1. Authentication

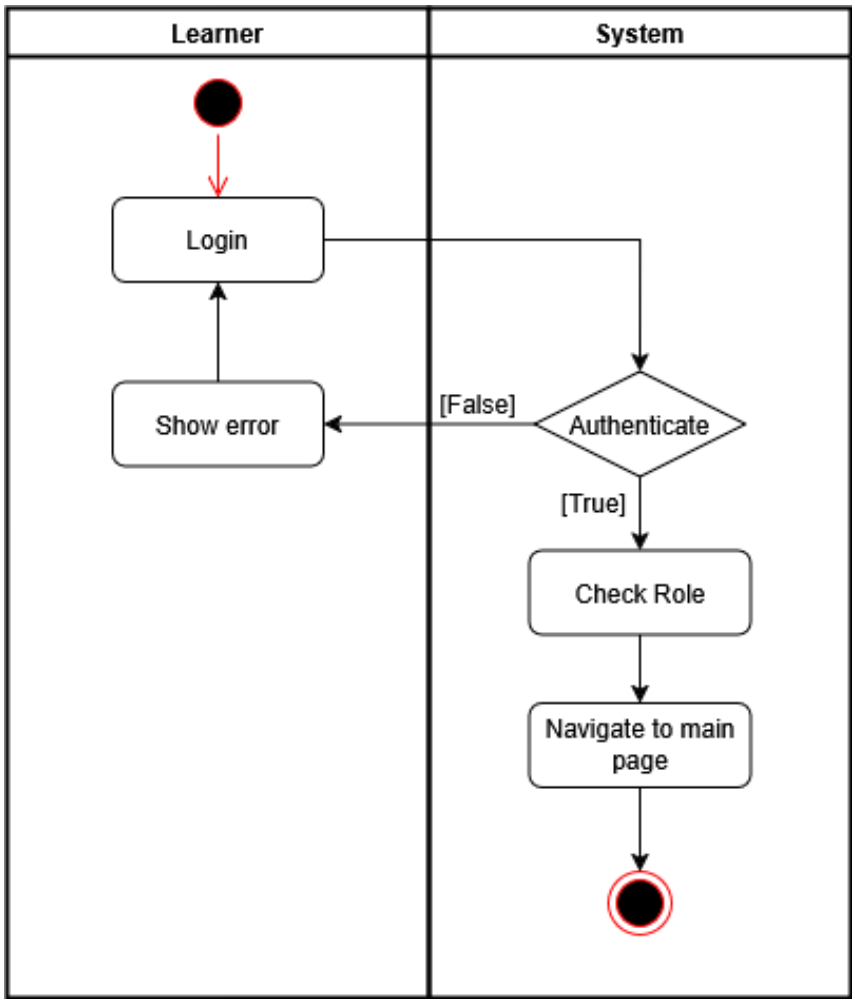

Figure 3.12 Activity diagram: Login

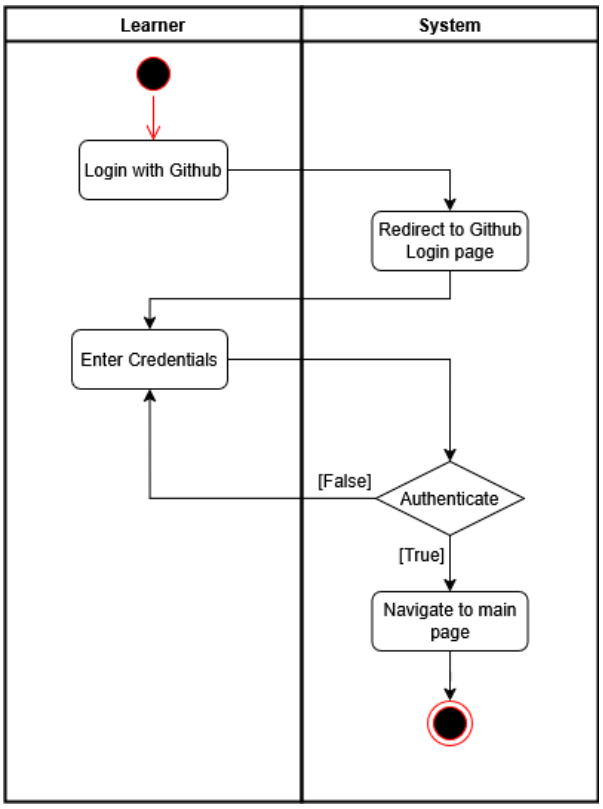

Figure 3.13 Activity diagram: Login with GitHub

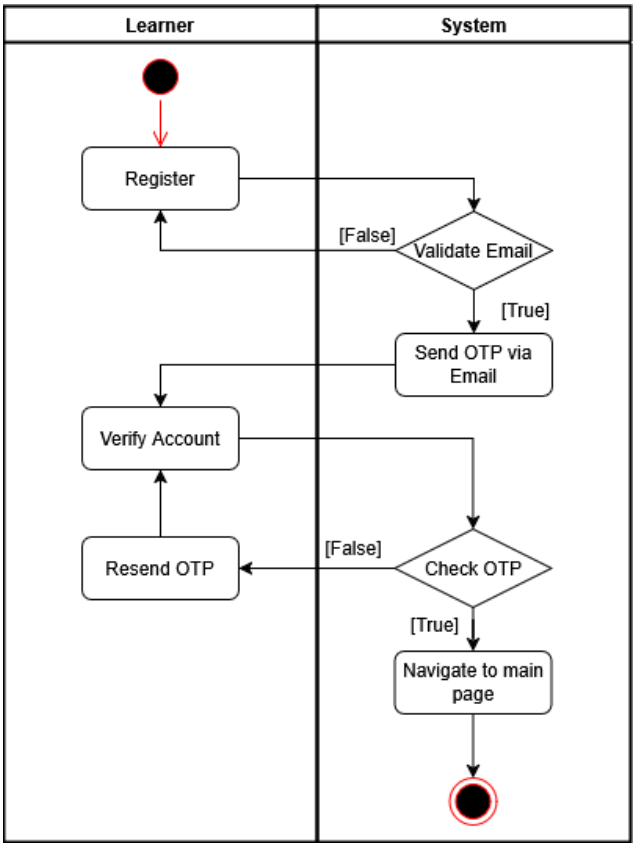

Figure 3.14 Activity diagram: Register

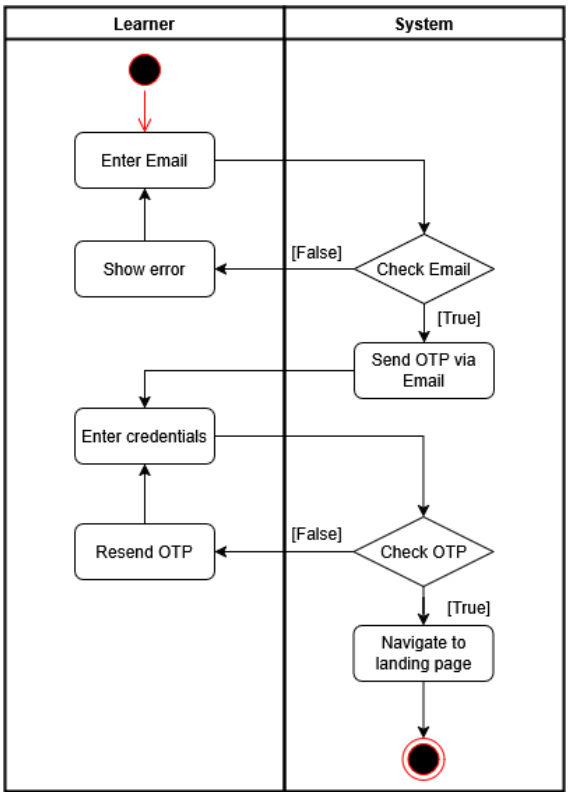

Figure 3.15 Activity diagram: Forgot password

3.4.2. Course

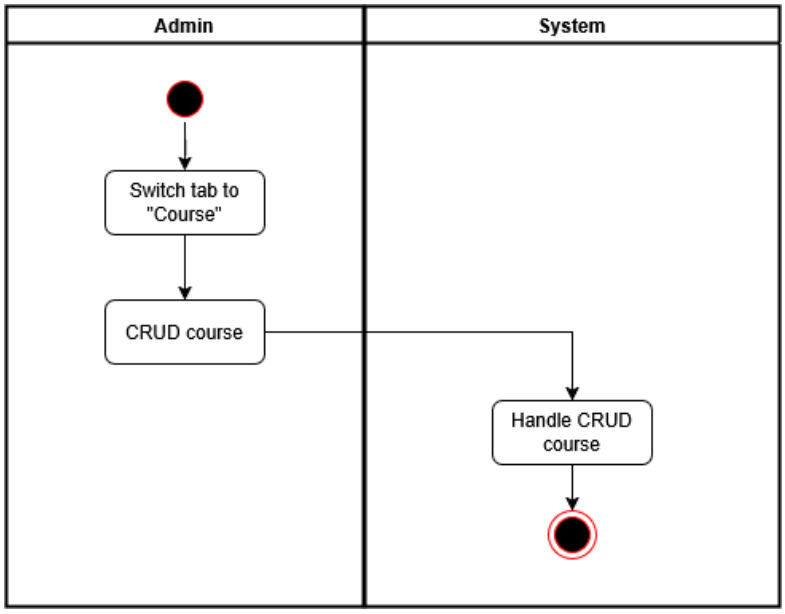

Figure 3.16 Activity diagram: Manage course

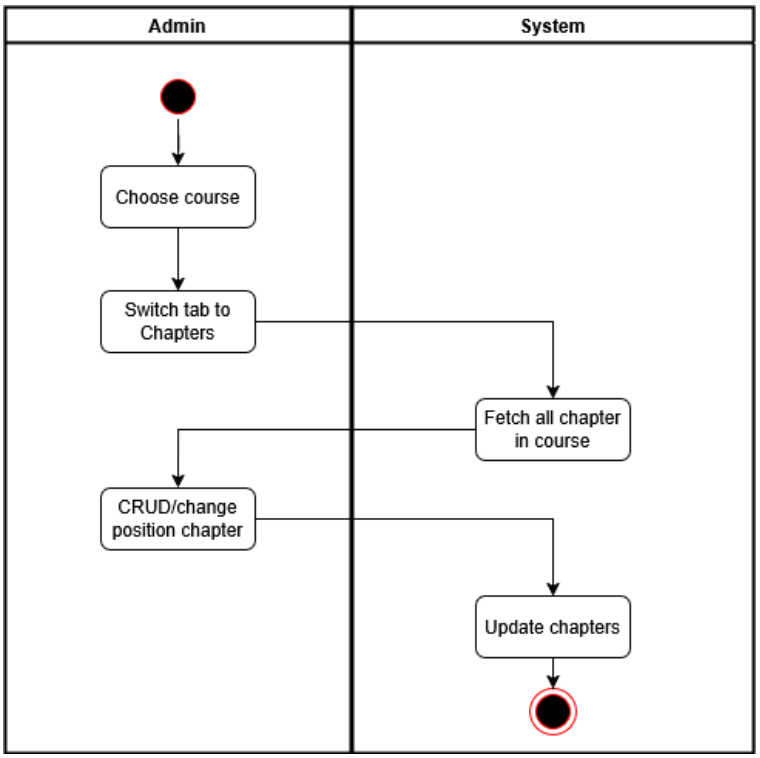

Figure 3.17 Activity diagram: Manage chapter

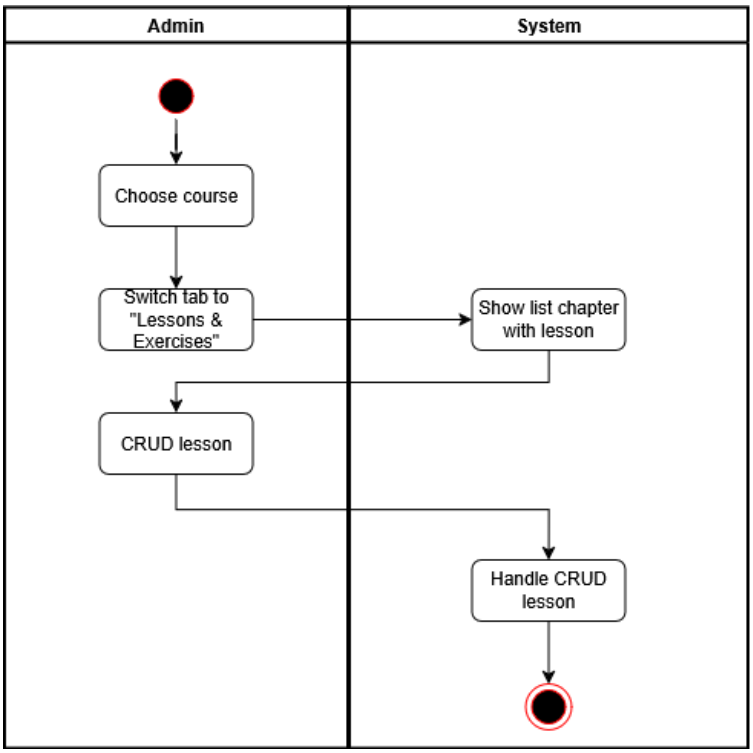

Figure 3.18 Activity diagram: Manage lesson

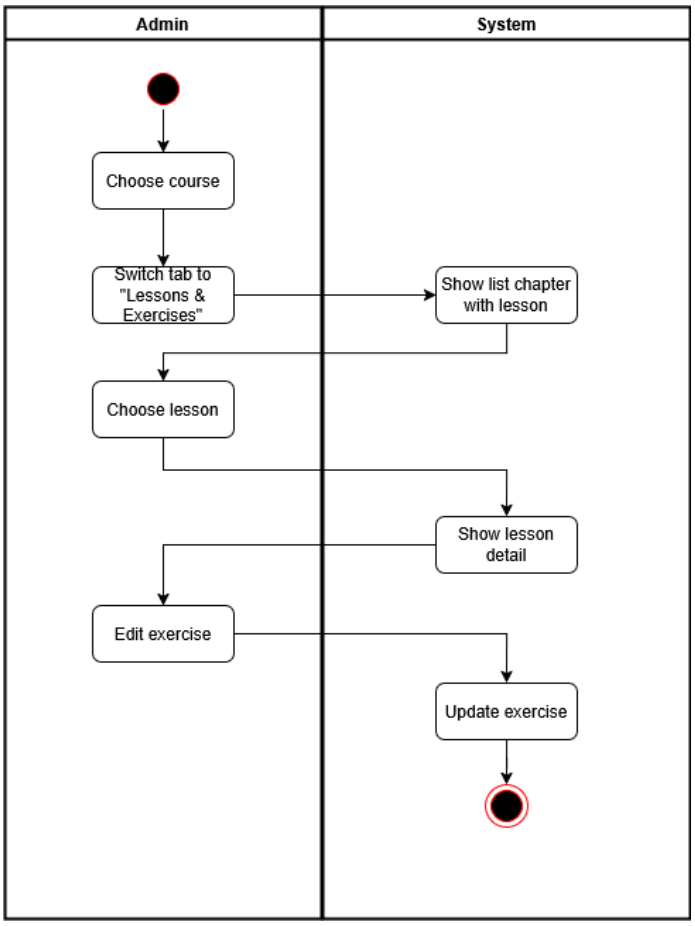

Figure 3.19 Activity diagram: Manage exercise

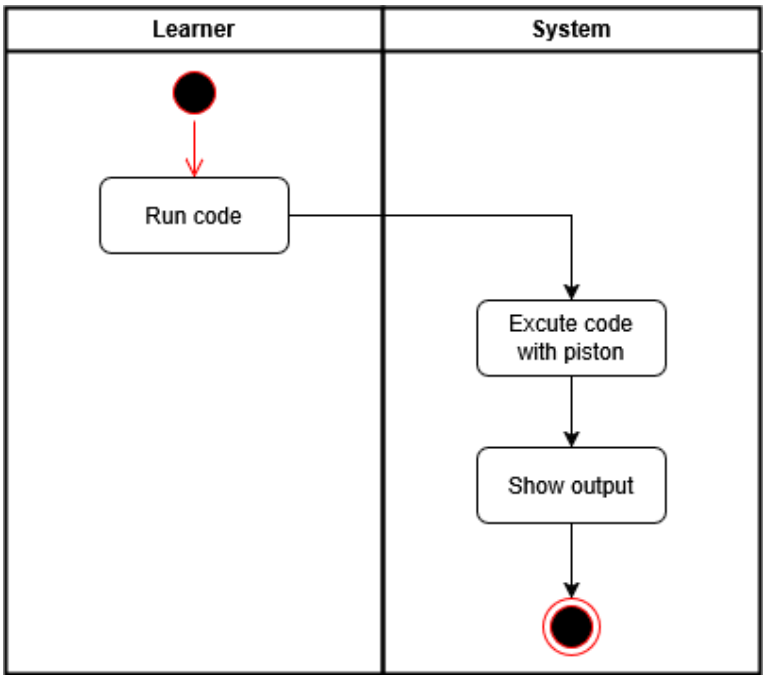

Figure 3.20 Activity diagram: Execute code

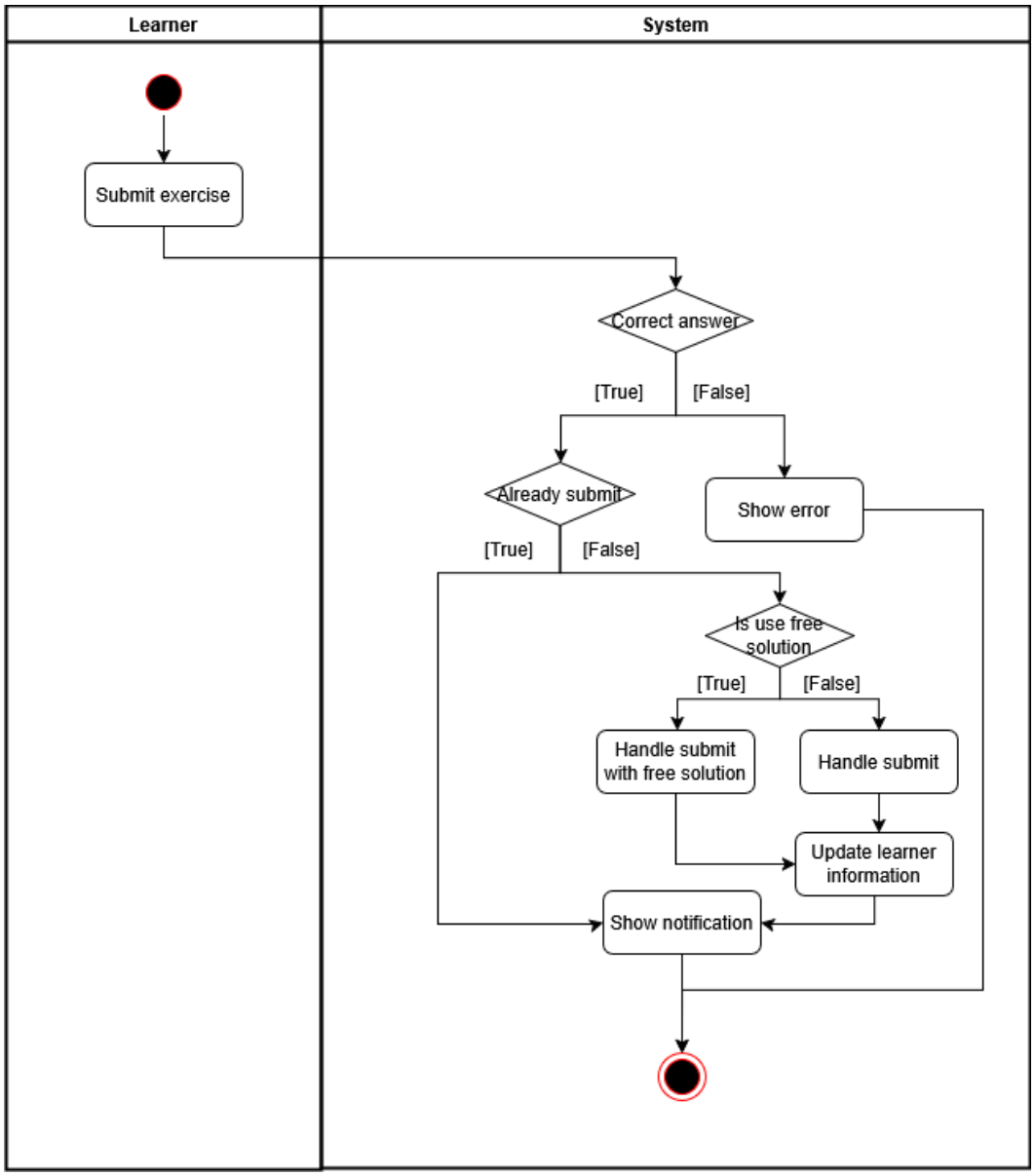

Figure 3.21 Activity diagram: Submit exercise

3.4.3. Item

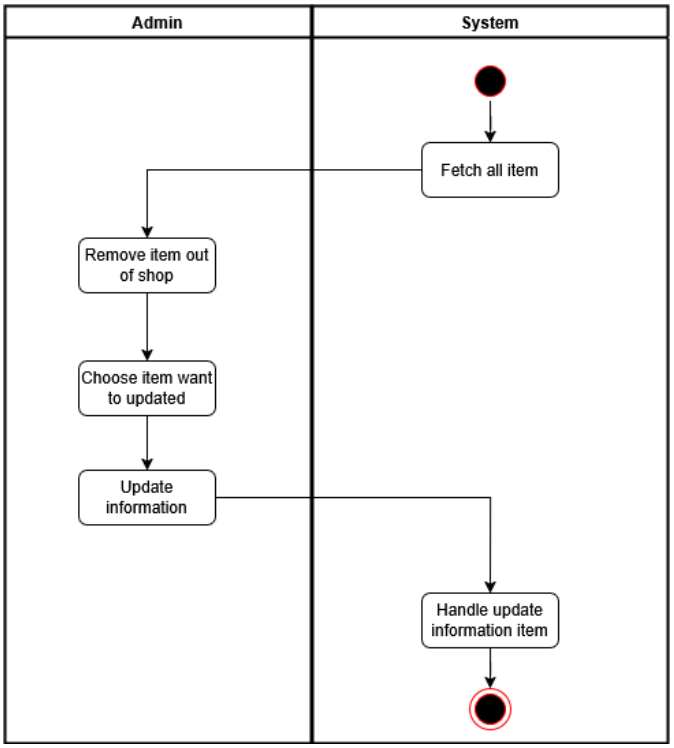

Figure 3.22 Activity diagram: Update item

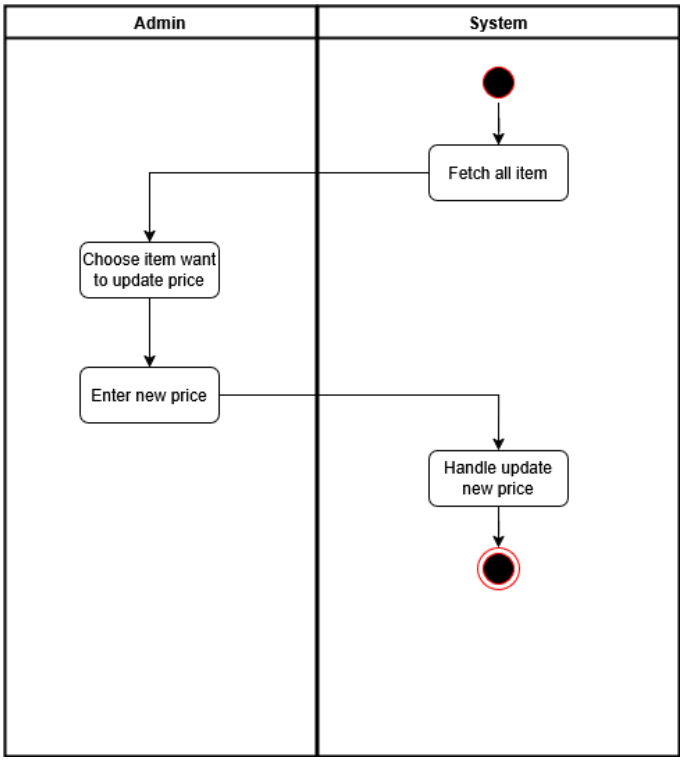

Figure 3.23 Activity diagram: Update item price

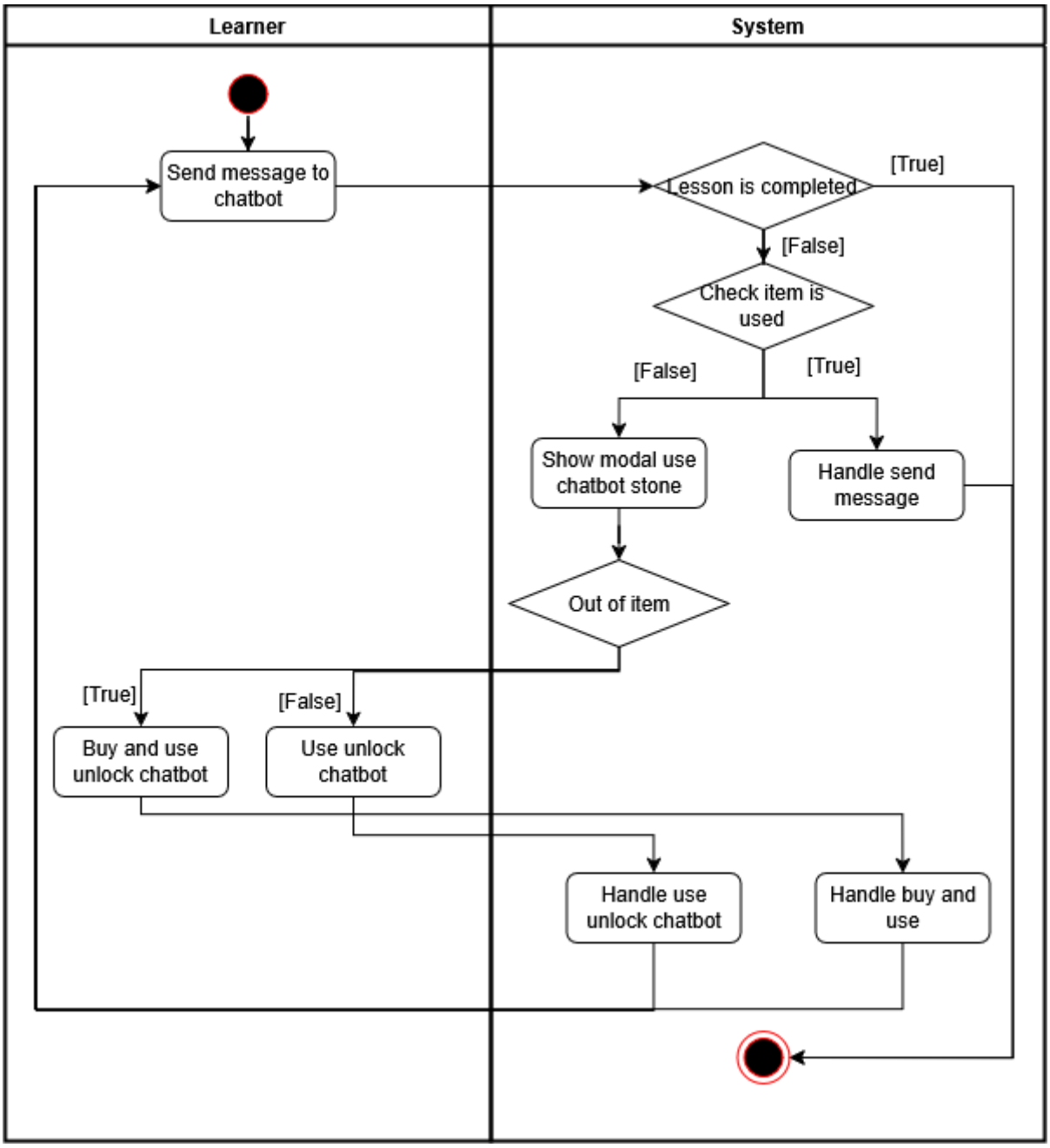

Figure 3.24 Activity diagram: Use chatbot

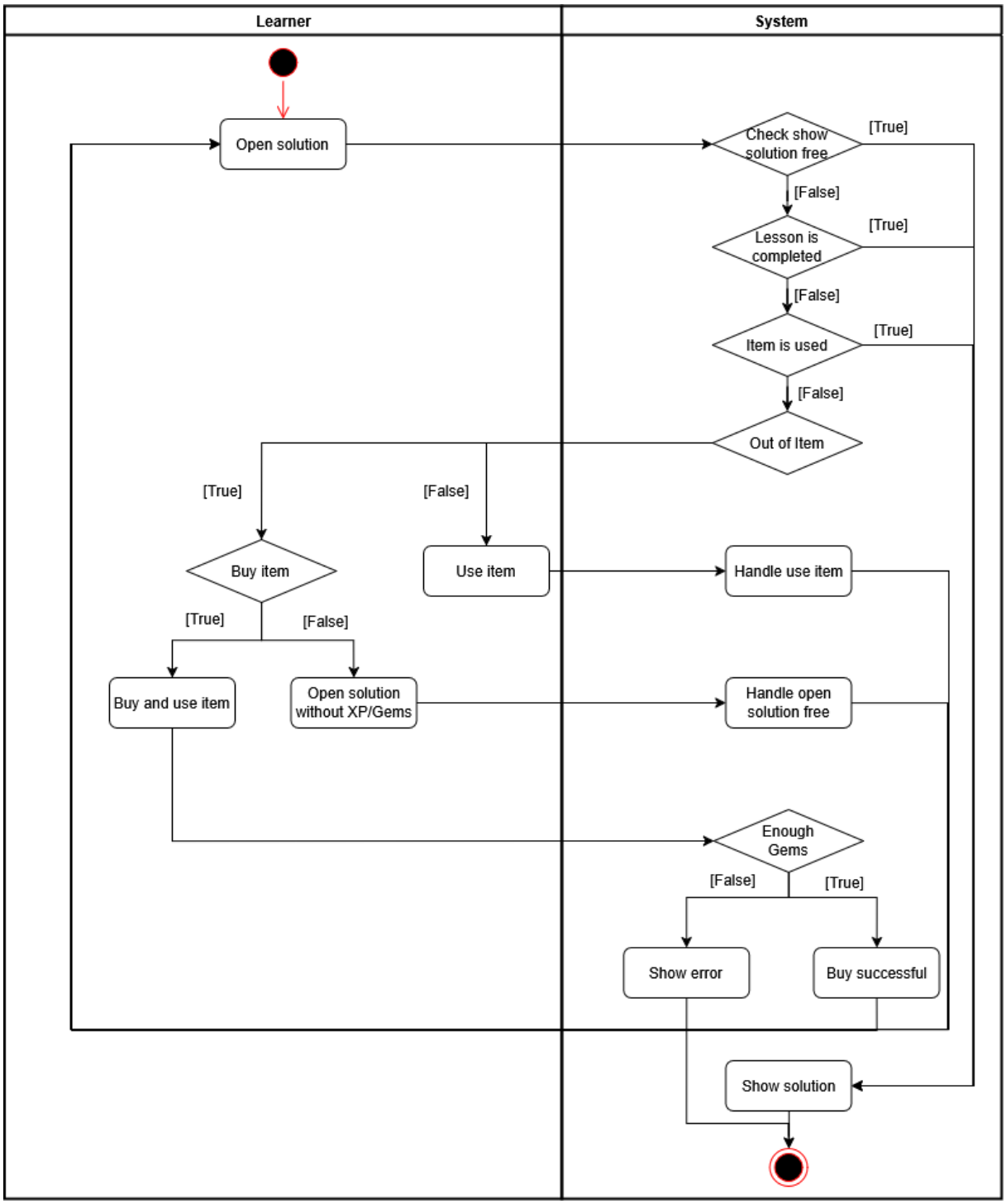

Figure 3.25 Activity diagram: Open solution

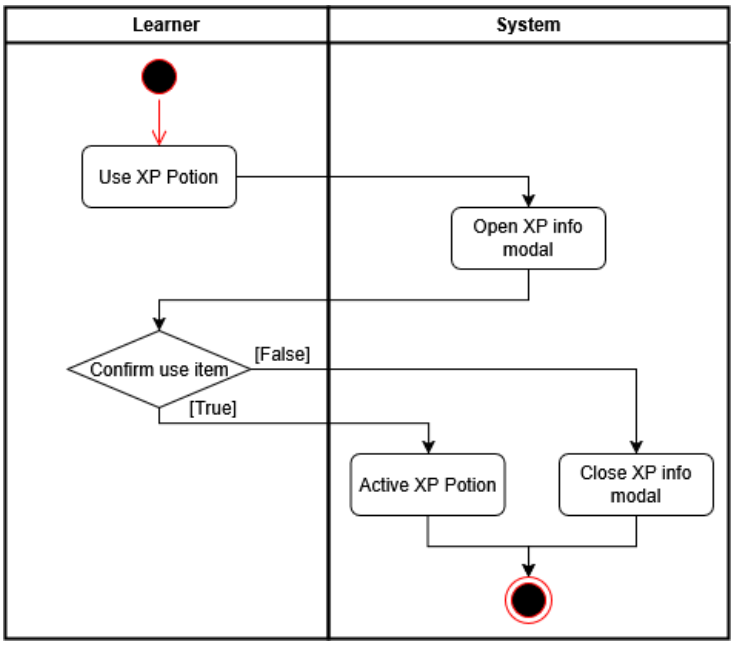

Figure 3.26 Activity diagram: Use bonus XP item

3.4.4. Shop

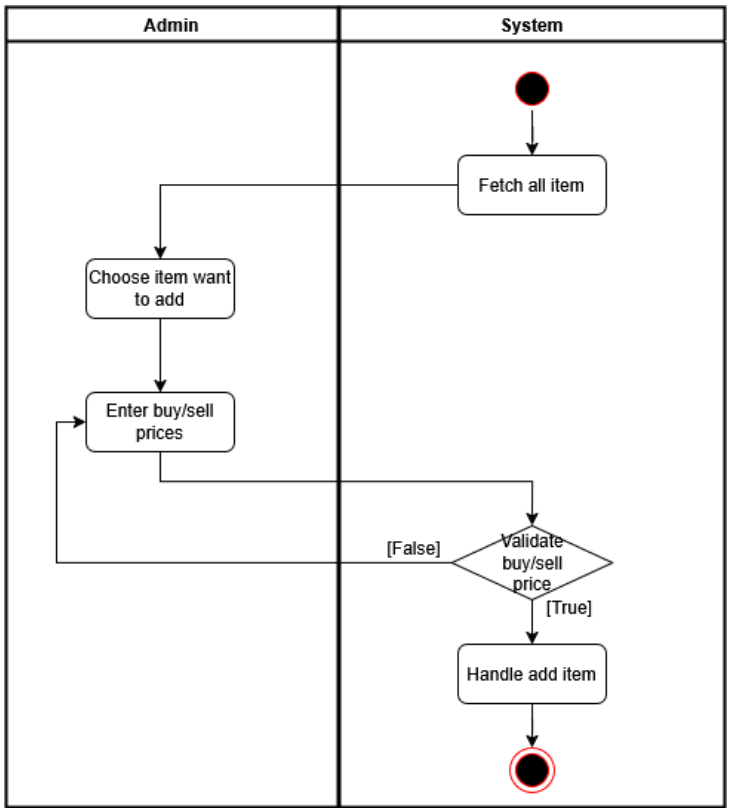

Figure 3.27 Activity diagram: add item to shop

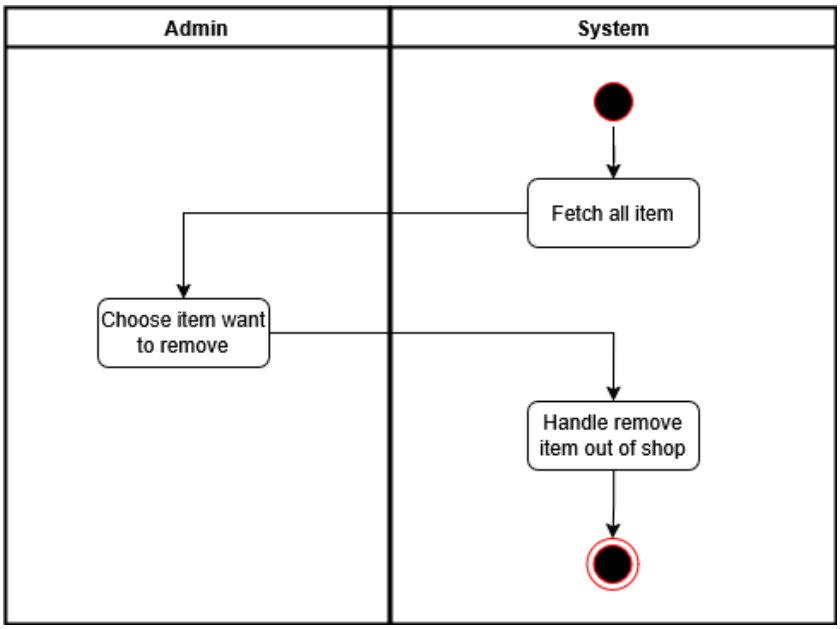

Figure 3.28 Activity diagram: Remove item out of shop

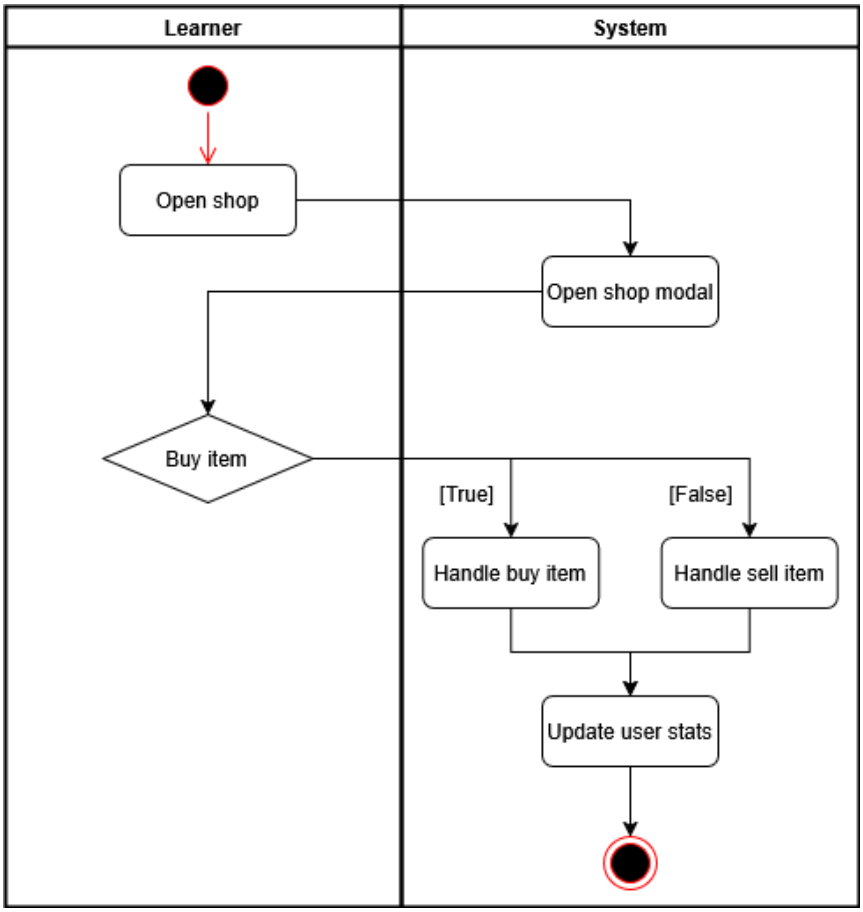

Figure 3.29 Activity diagram: Buy/Sell item

3.4.5. Achievement

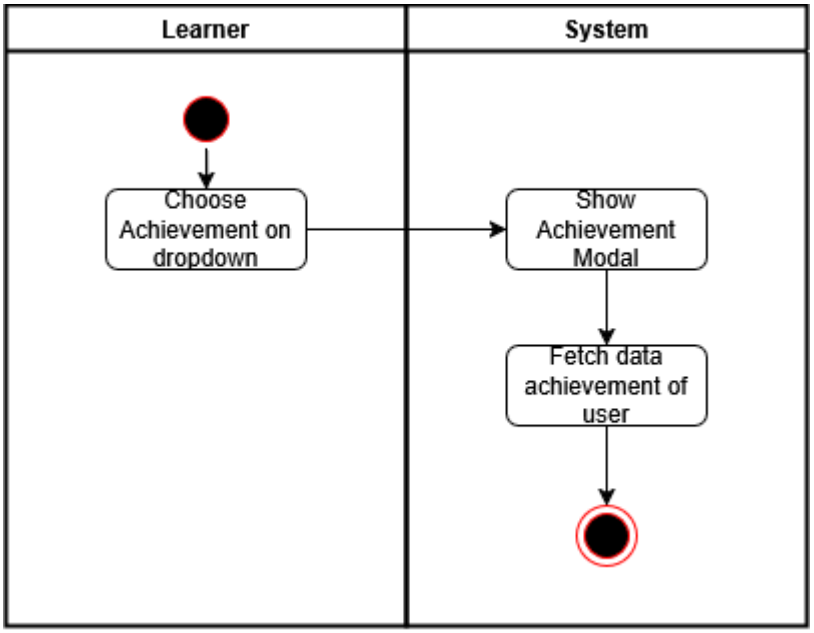

Figure 3.30 Activity diagram: Track achievement progress

3.4.6. Leaderboard

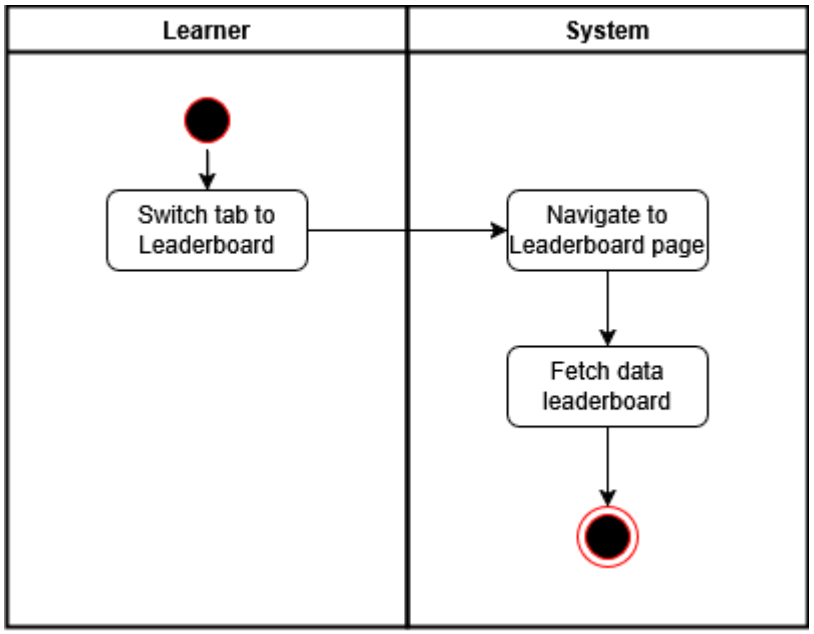

Figure 3.31 Activity diagram: View leaderboard

3.4.7. Profile

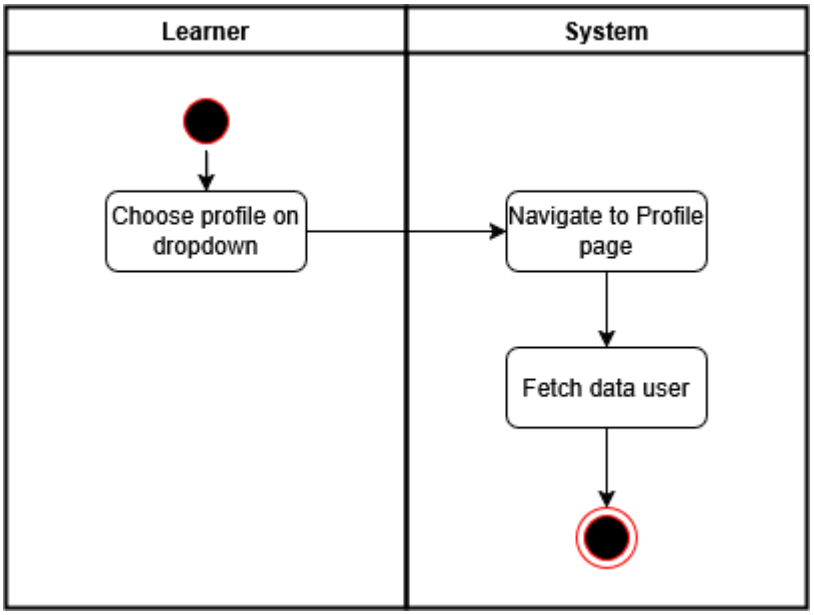

Figure 3.32 Activity diagram: view profile

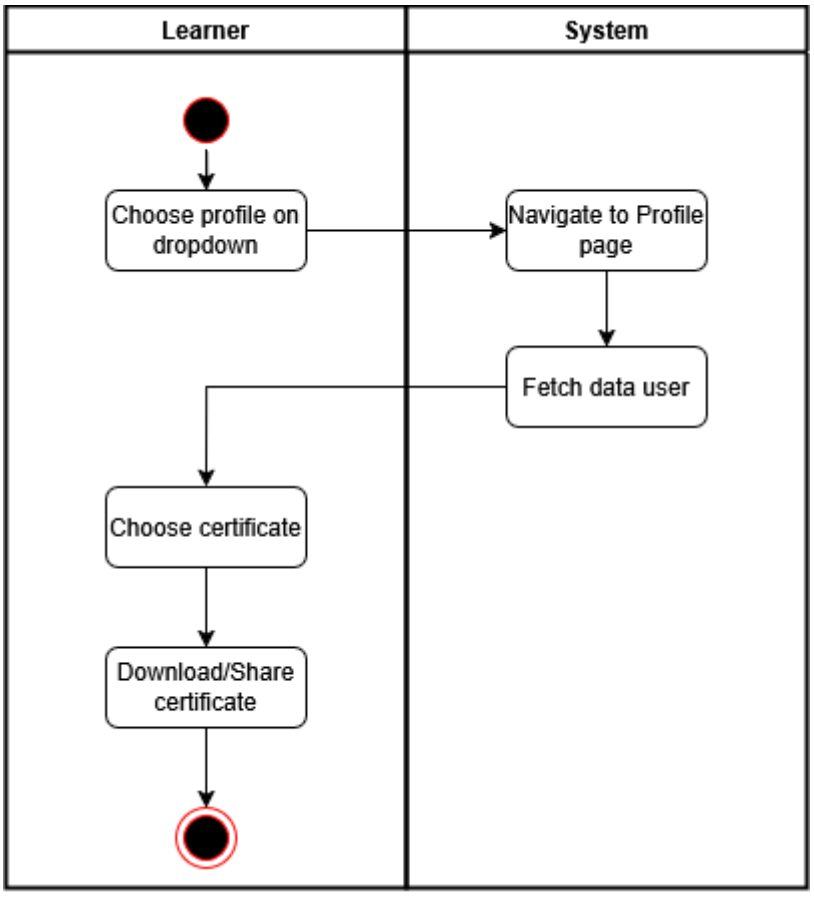

Figure 3.33 Activity diagram: Download/Share certificate

3.4.8. Notification

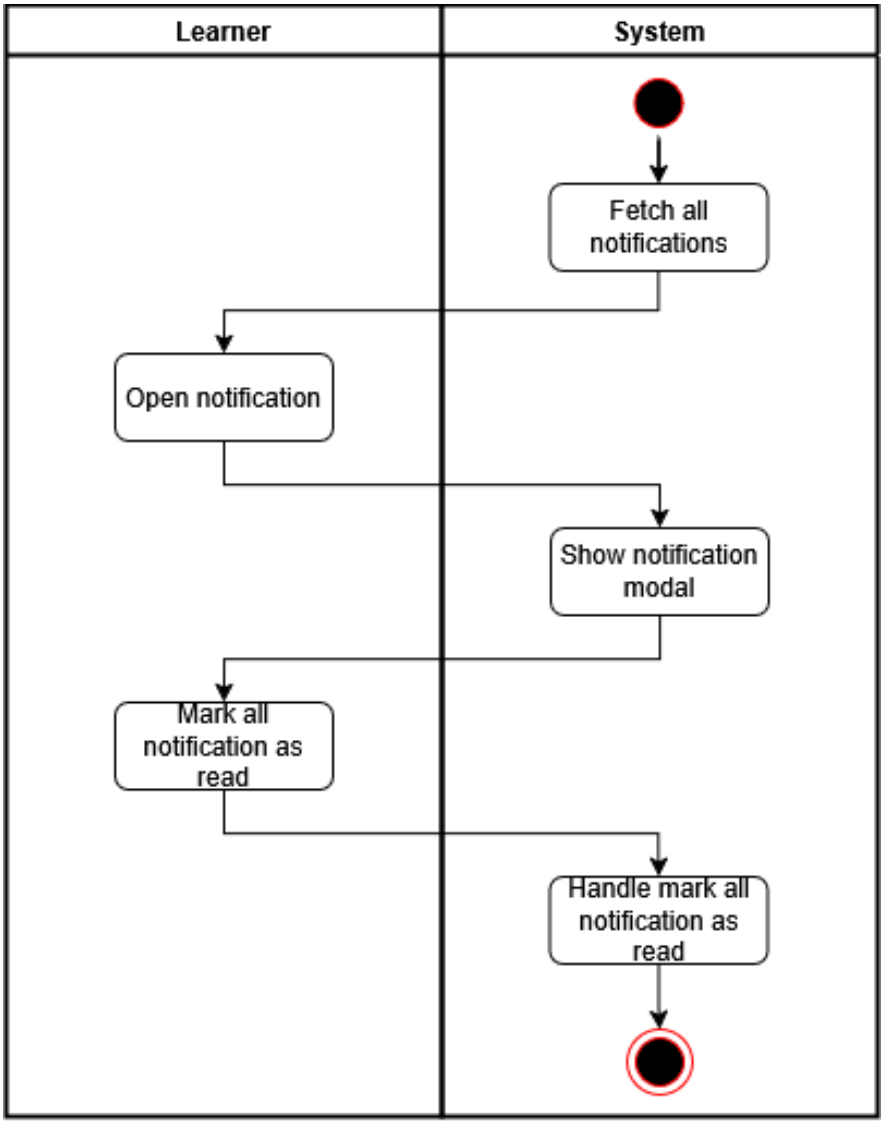

Figure 3.34 Activity diagram: Get notification

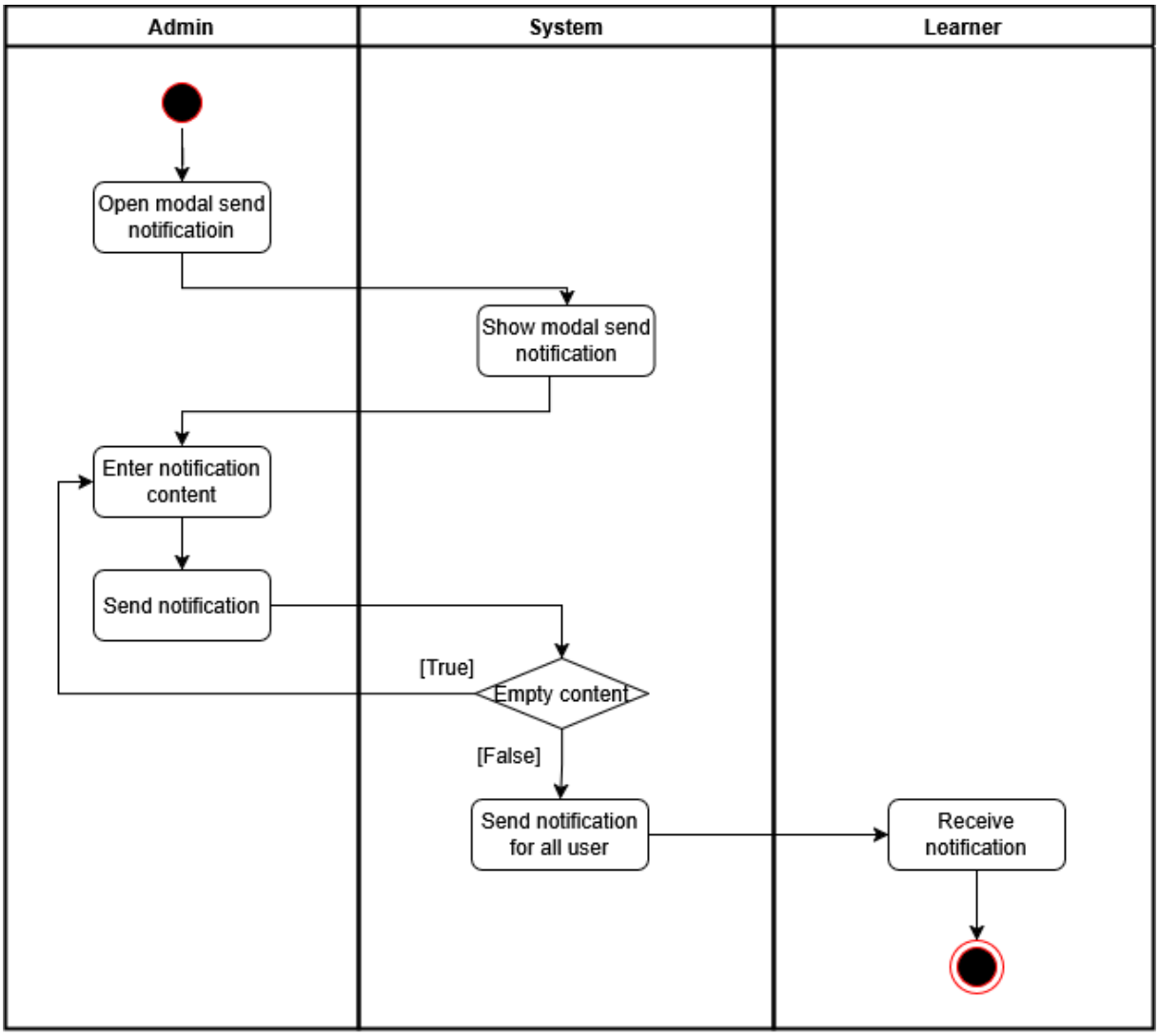

Figure 3.35 Activity diagram: Send global notification

3.4.9. Spell book

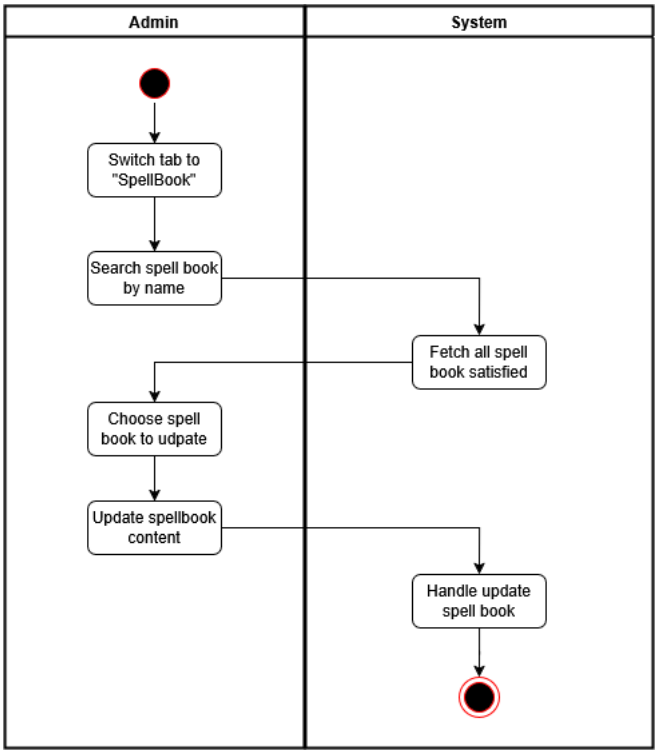

Figure 3.36 Activity diagram: Update spell book

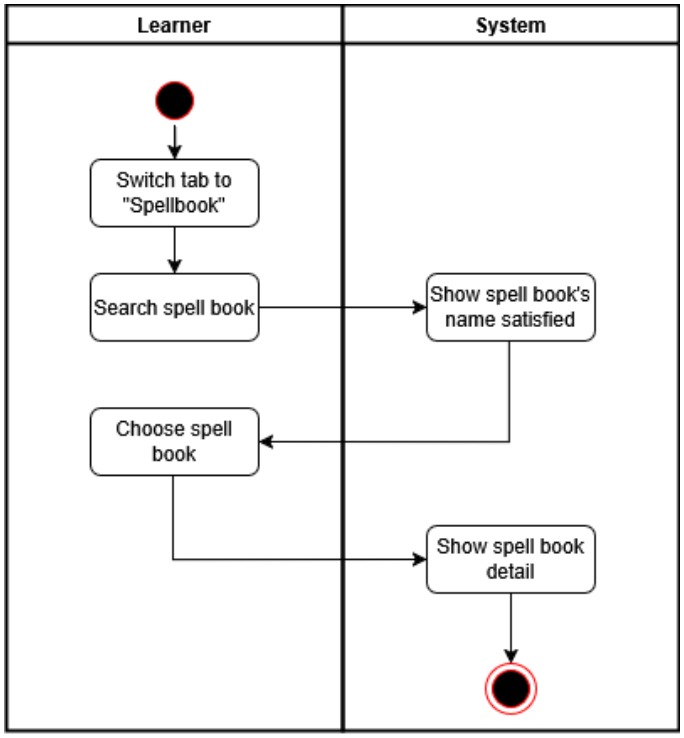

Figure 3.37 Activity diagram: view spell book

3.4.10. Daily wheel

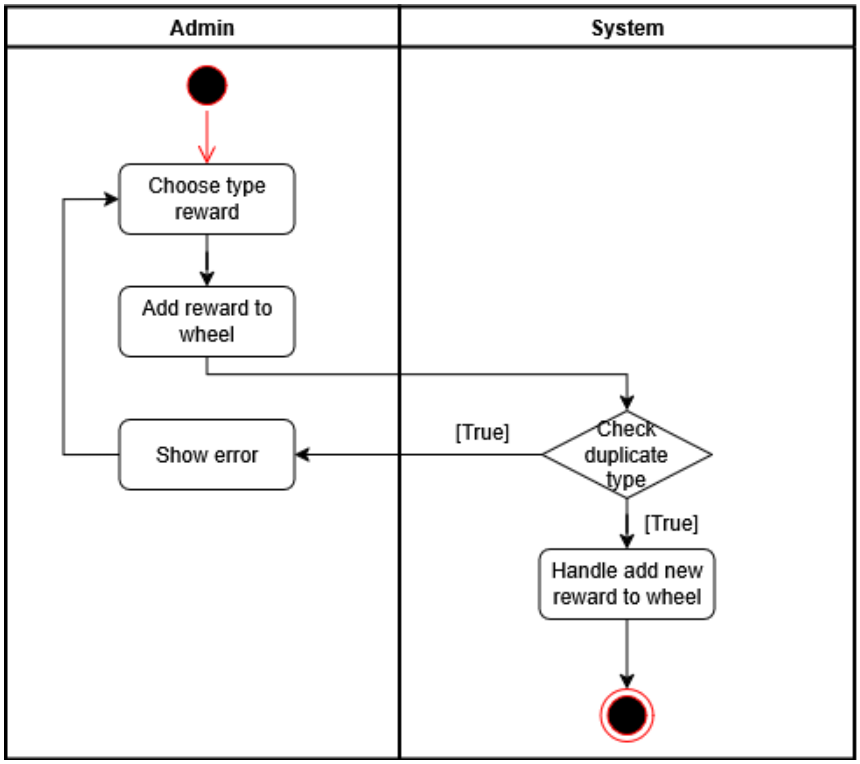

Figure 3.38 Activity diagram: Add daily wheel item

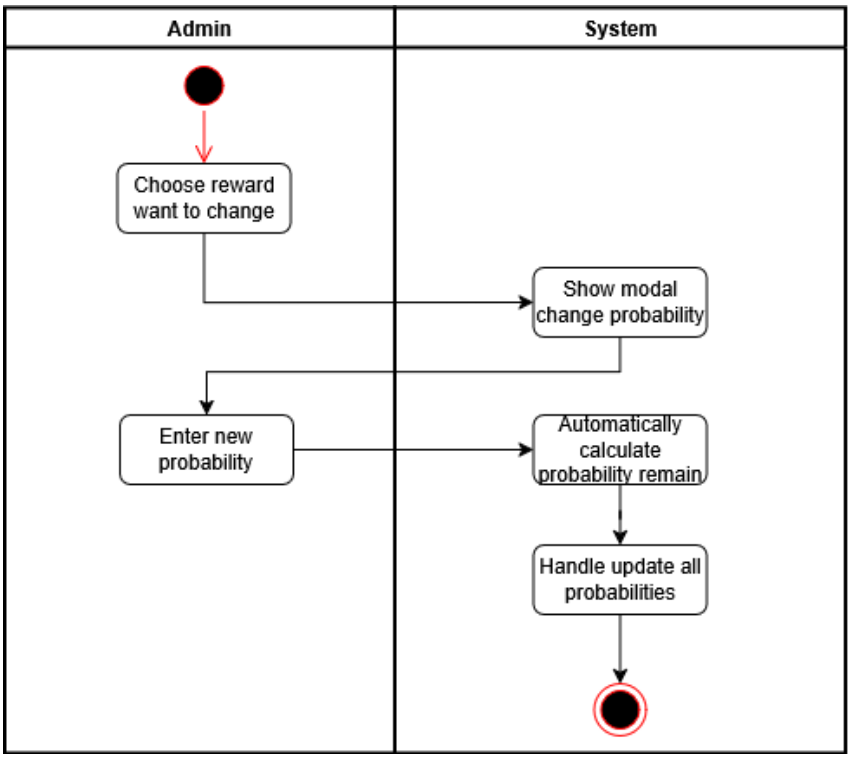

Figure 3.39 Activity diagram: Update daily wheel item probability

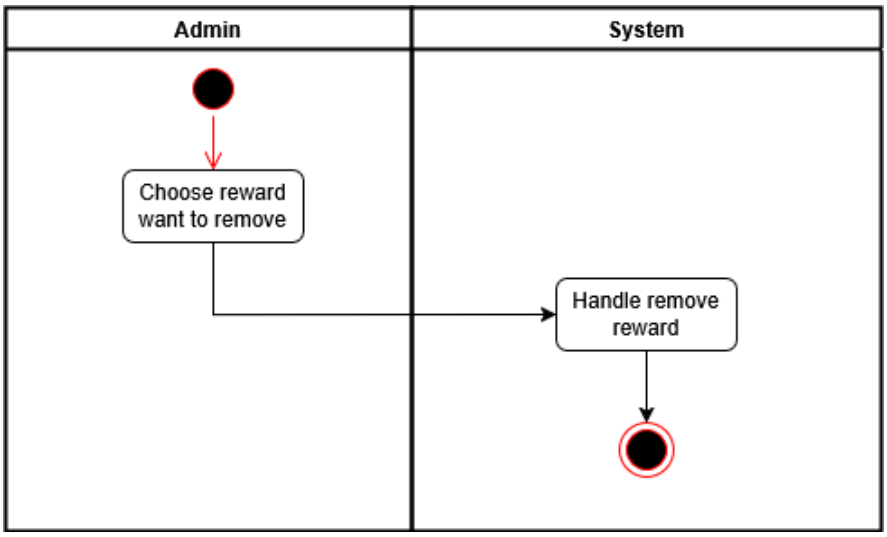

Figure 3.40 Activity diagram: Remove daily wheel item

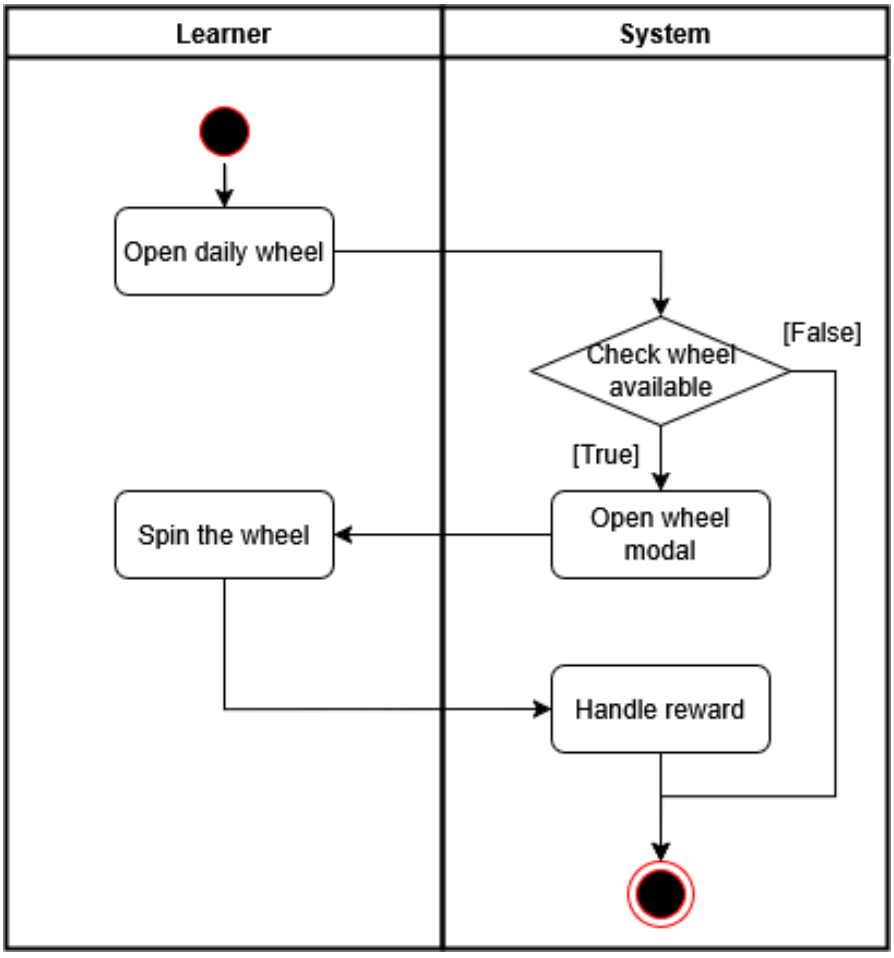

Figure 3.41 Activity diagram: Spin daily wheel

3.4.11. Dashboard

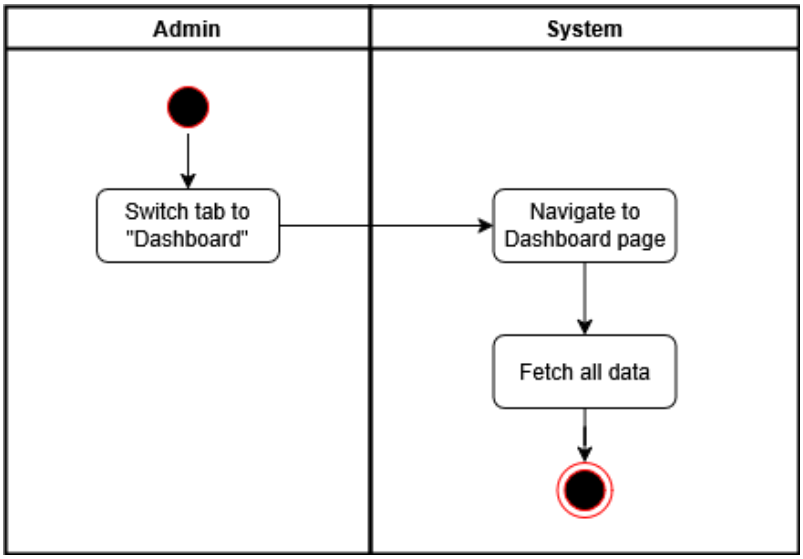

Figure 3.42 Activity diagram: View dashboard

3.4.12. Feedback

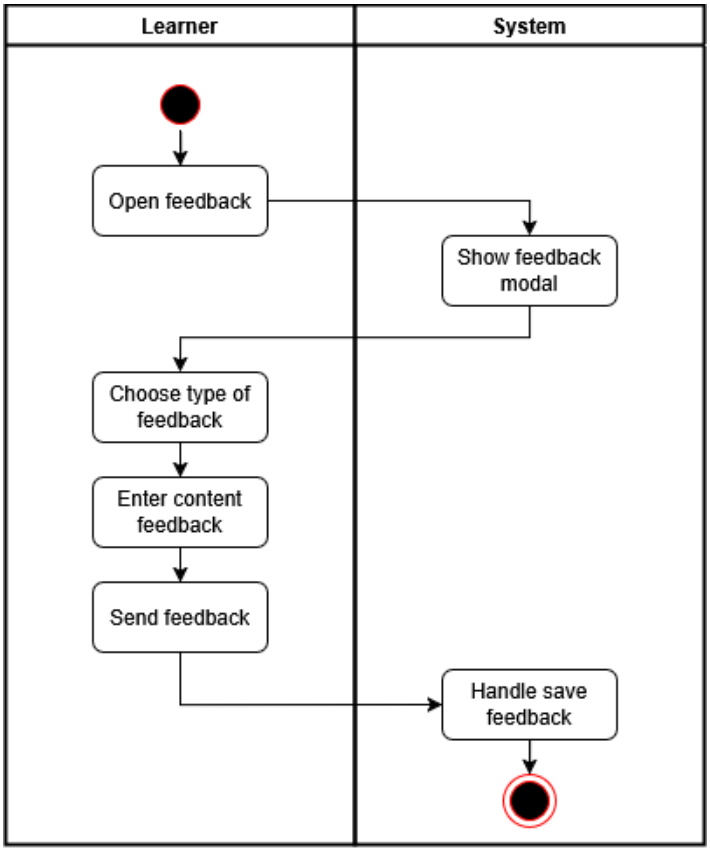

Figure 3.43 Activity diagram: Create feedback

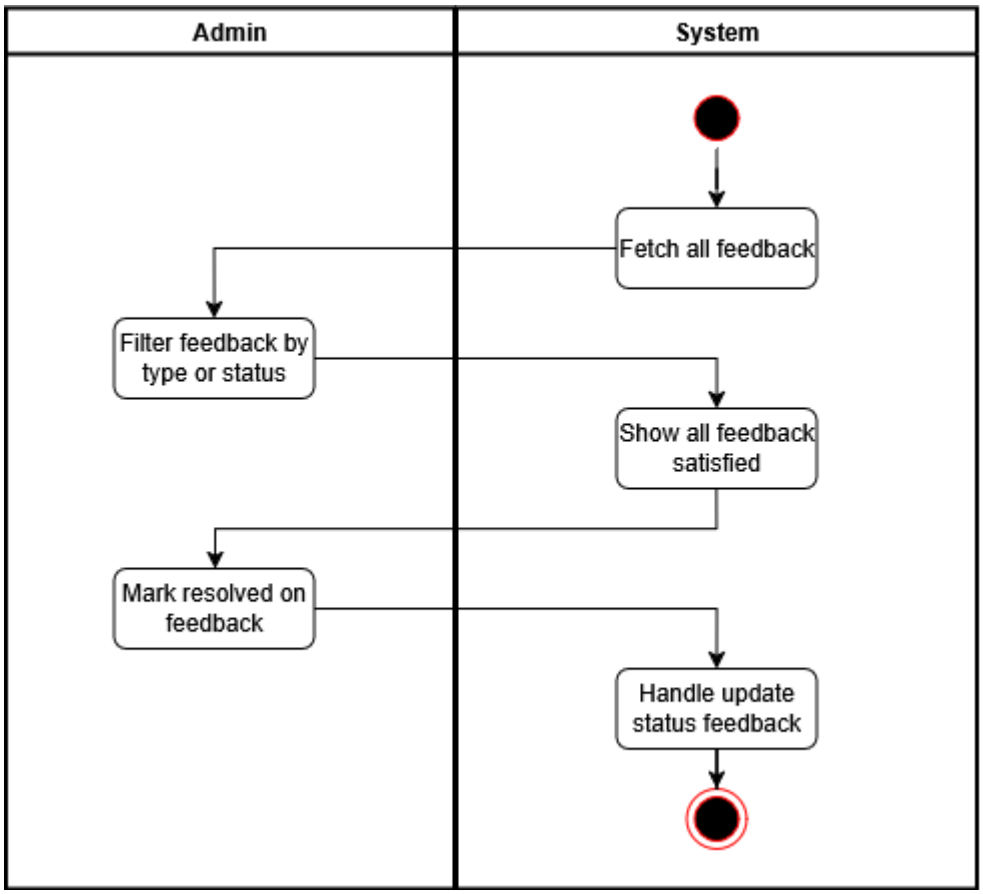

Figure 3.44 Activity diagram: Handle feedback

3.5. Sequence diagrams

3.5.1. Authentication

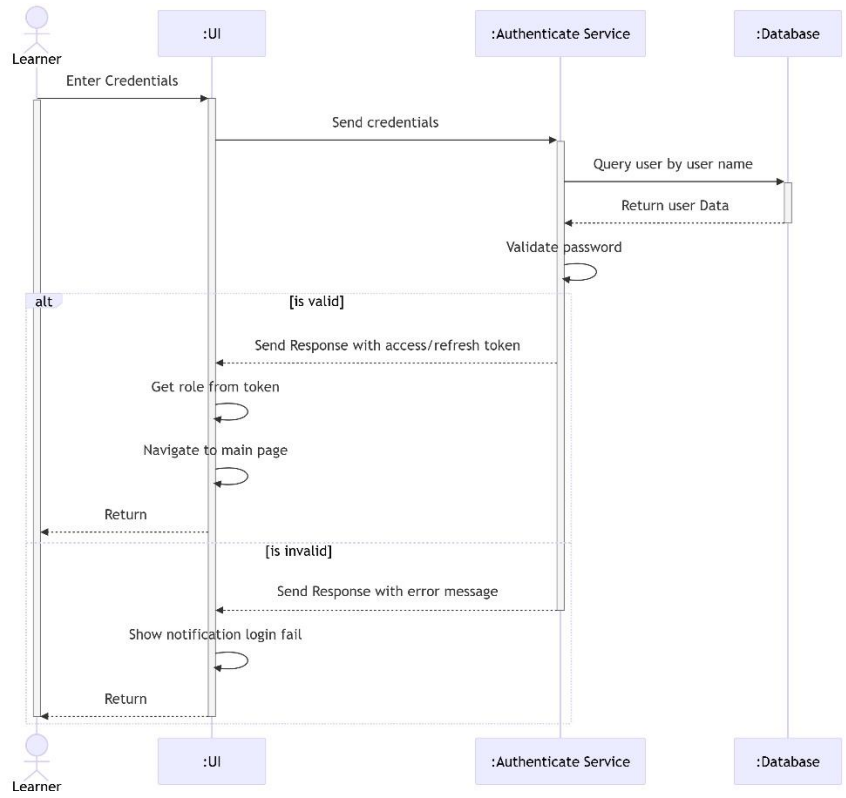

Figure 3.45 Sequence Diagram: Login

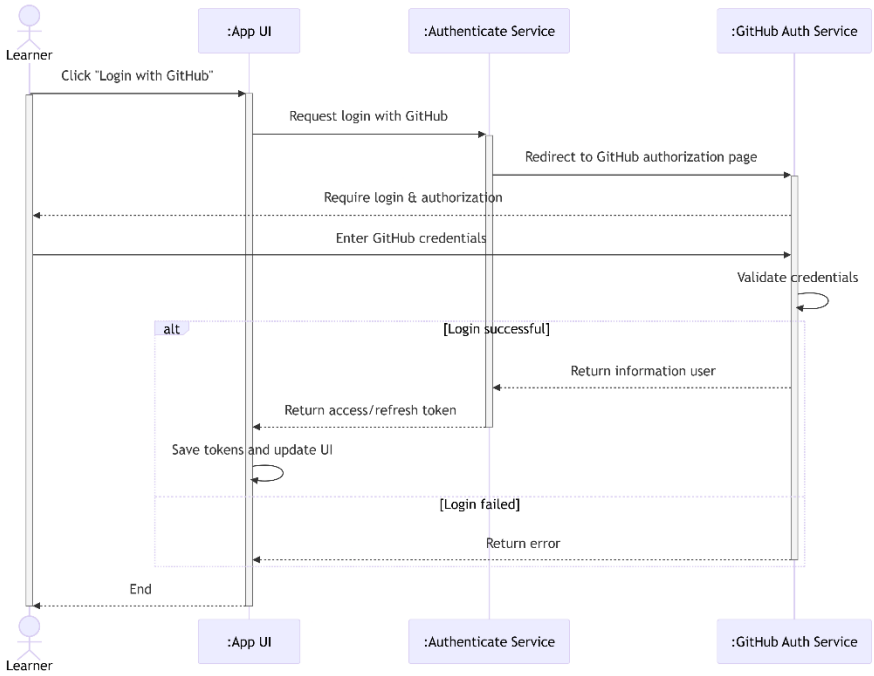

Figure 3.46 Sequence Diagram: Login with GitHub

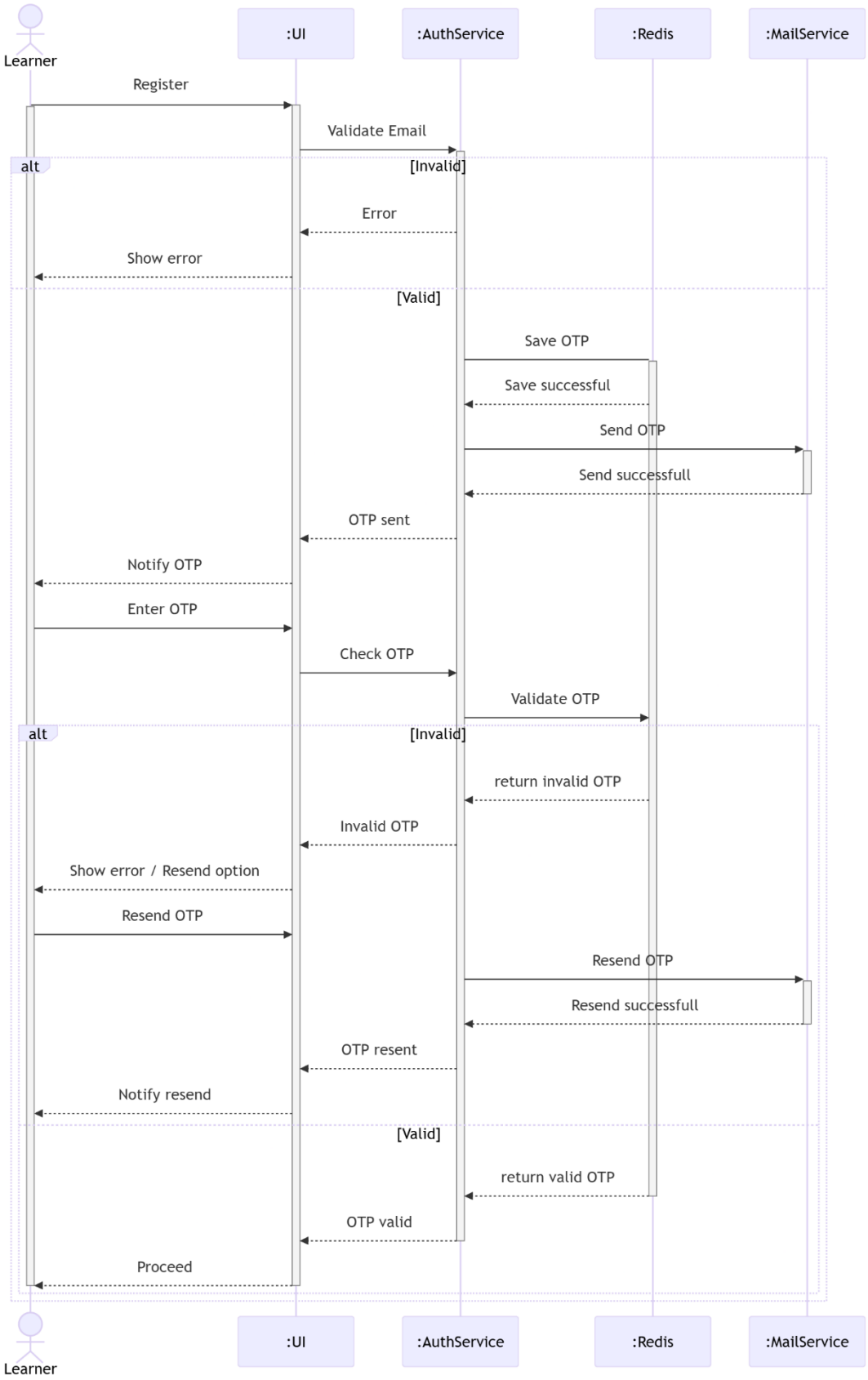

Figure 3.47 Sequence Diagram: Register

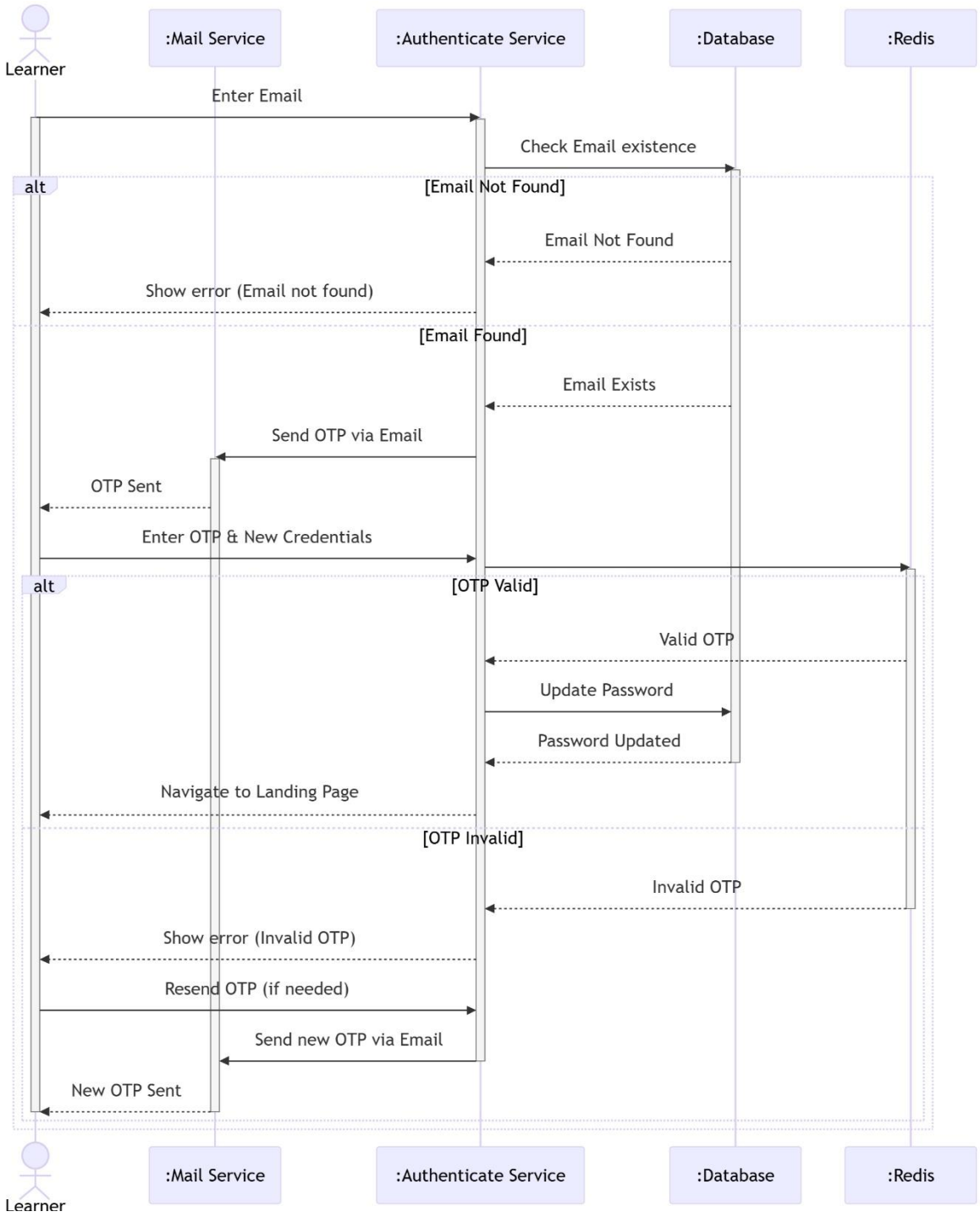

Figure 3.48 Sequence Diagram: Forgot password

3.5.2. Course

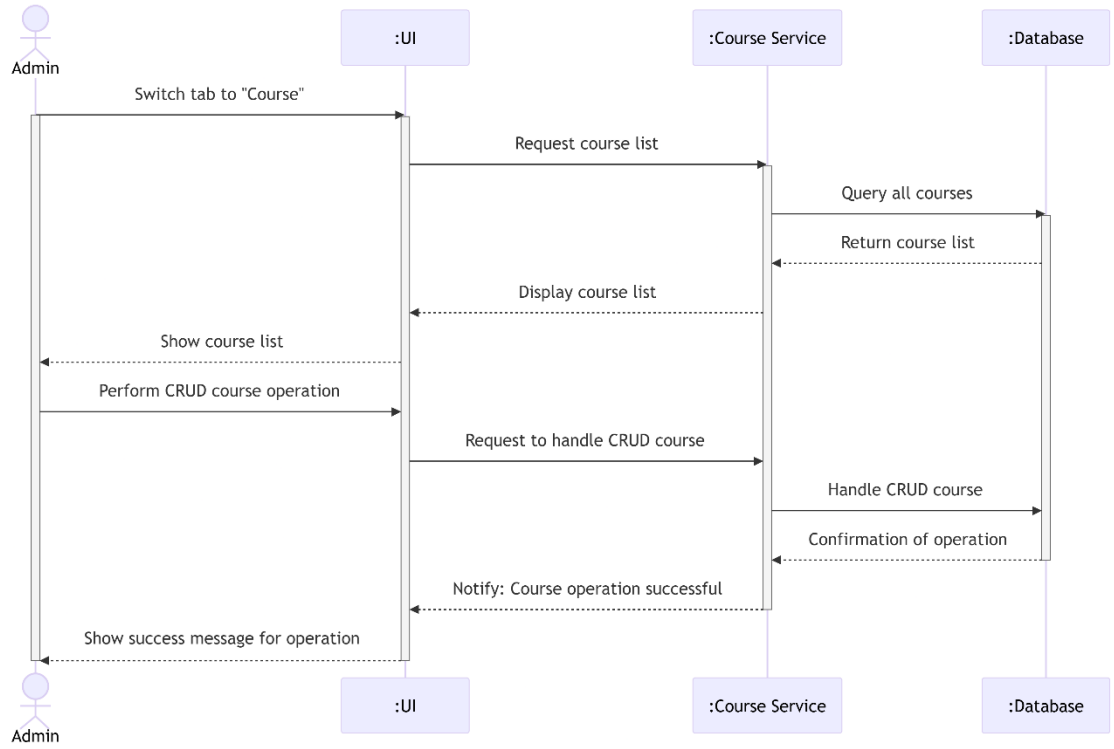

Figure 3.49 Sequence Diagram: Manage course

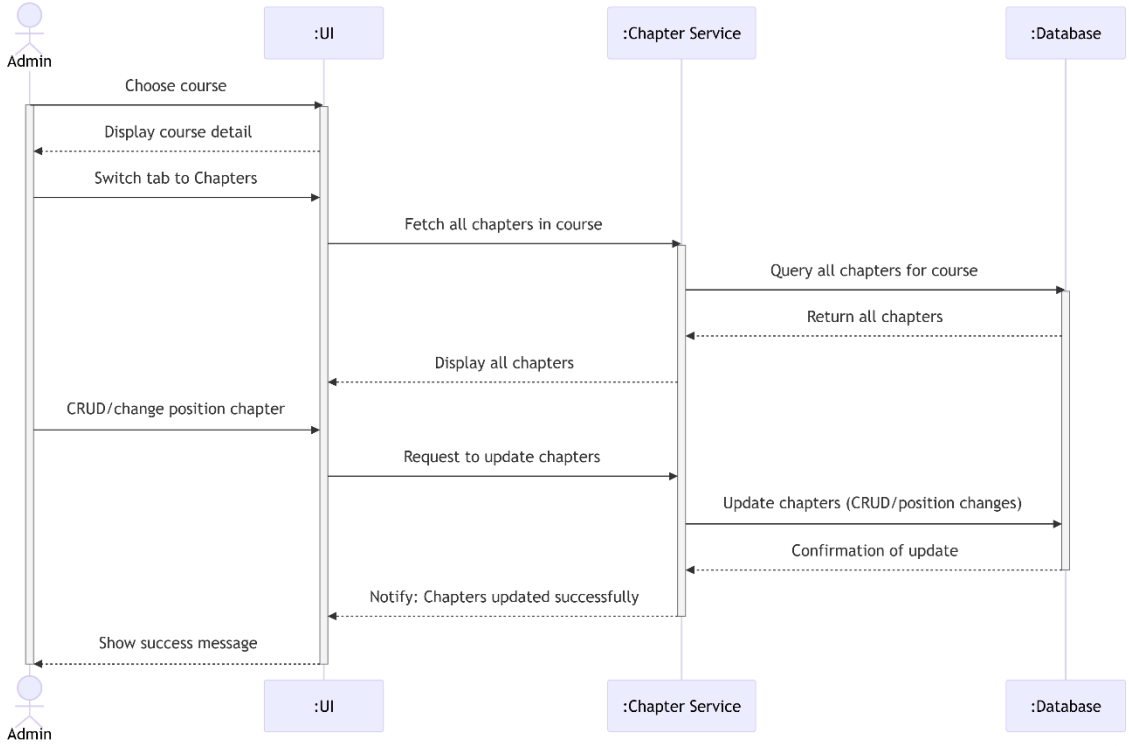

Figure 3.50 Sequence Diagram: Manage chapter

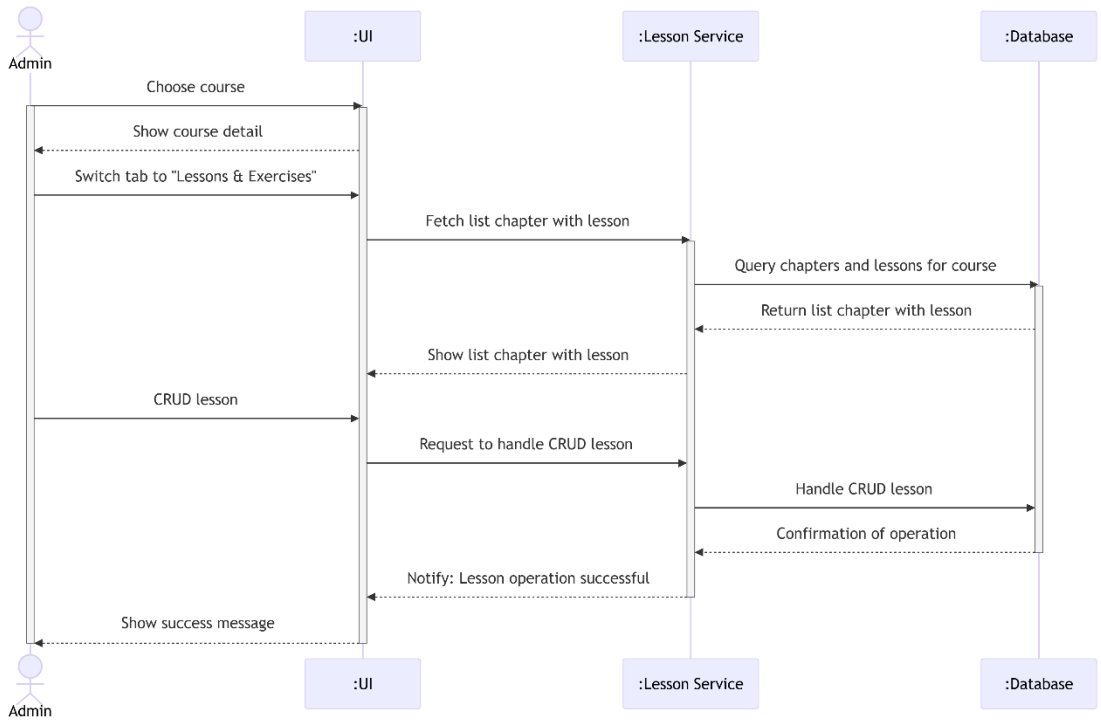

Figure 3.51 Sequence Diagram: Manage lesson

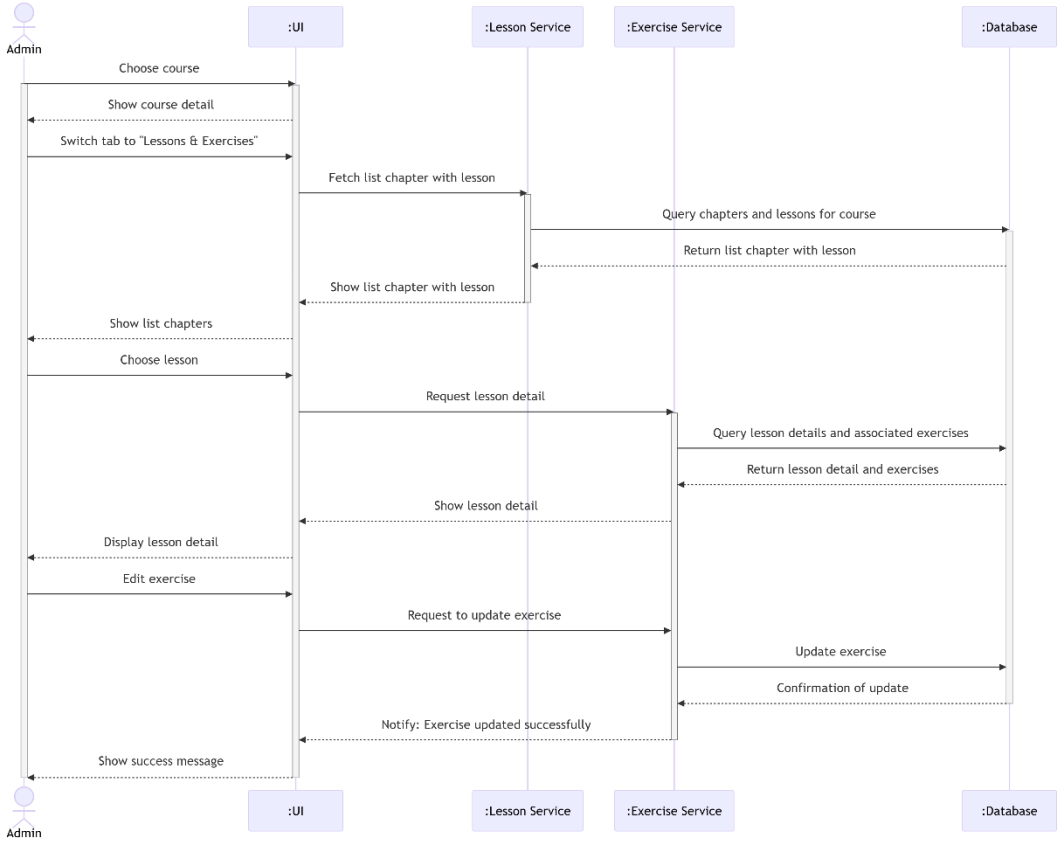

Figure 3.52 Sequence Diagram: Manage exercise

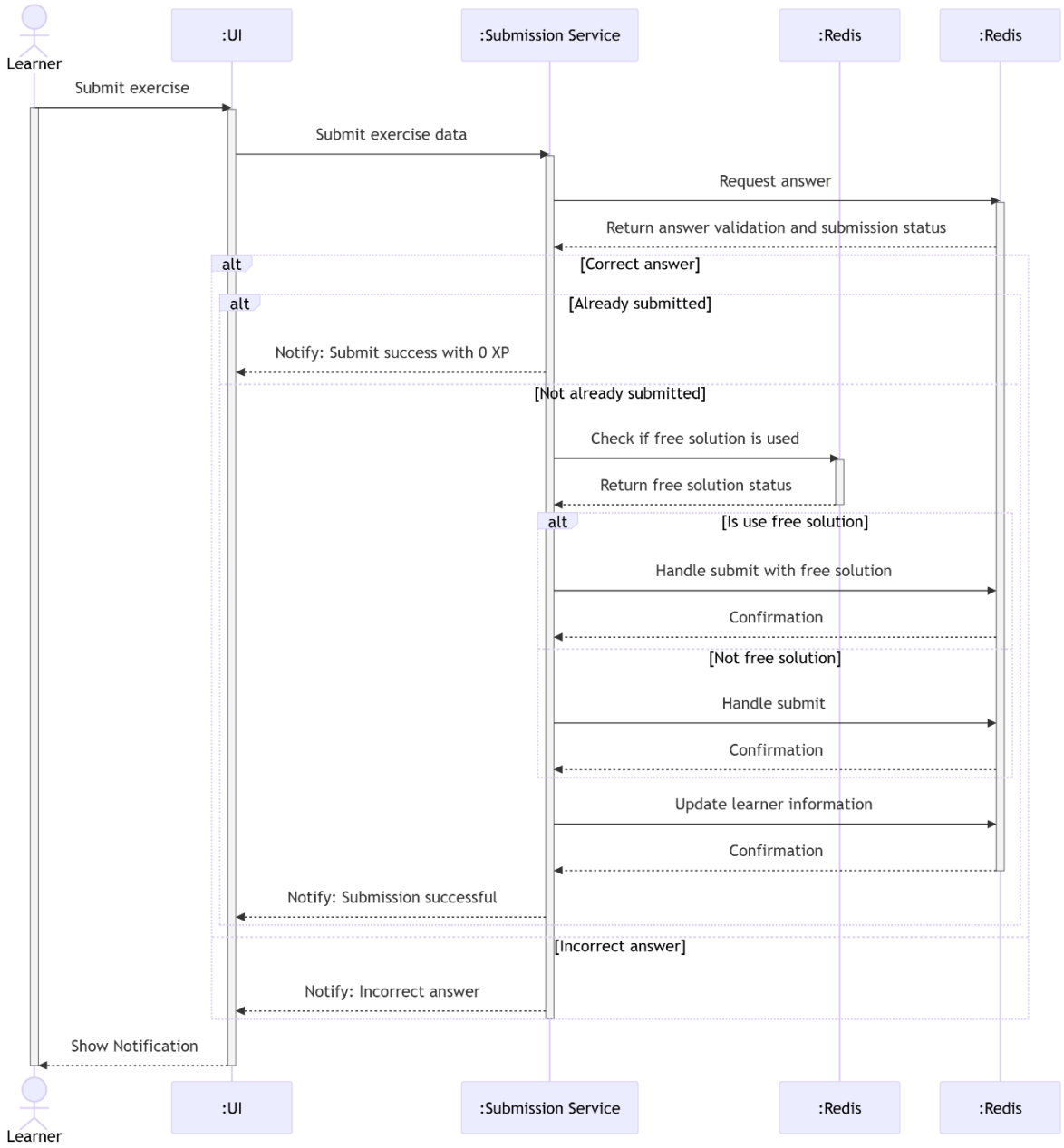

Figure 3.53 Sequence Diagram: Submit exercise

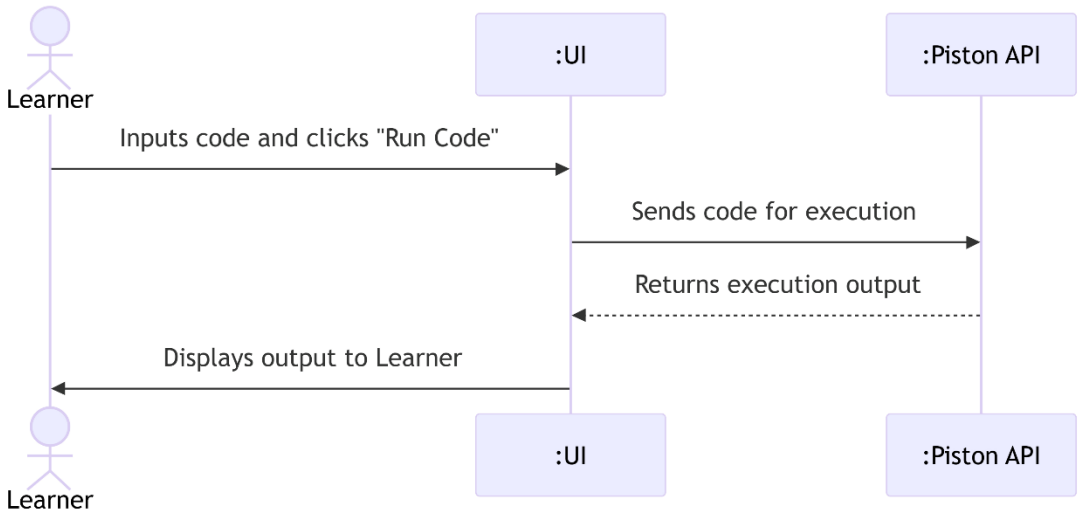

Figure 3.54 Sequence Diagram: Execute code

3.5.3. Item

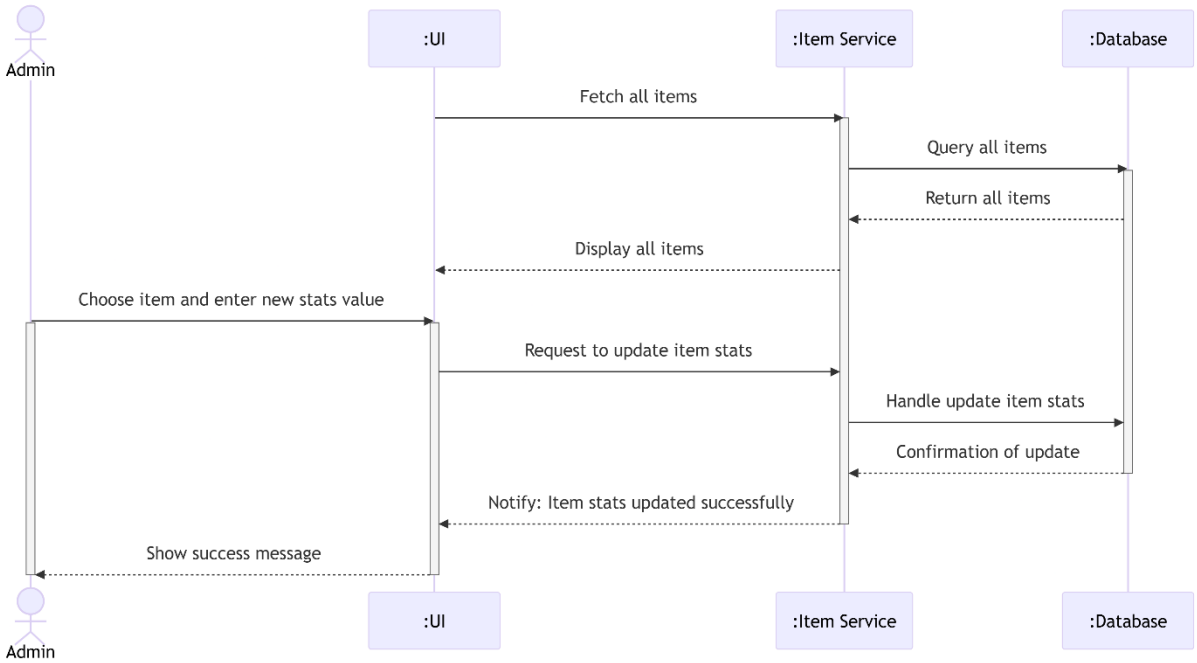

Figure 3.55 Sequence Diagram: Manage item

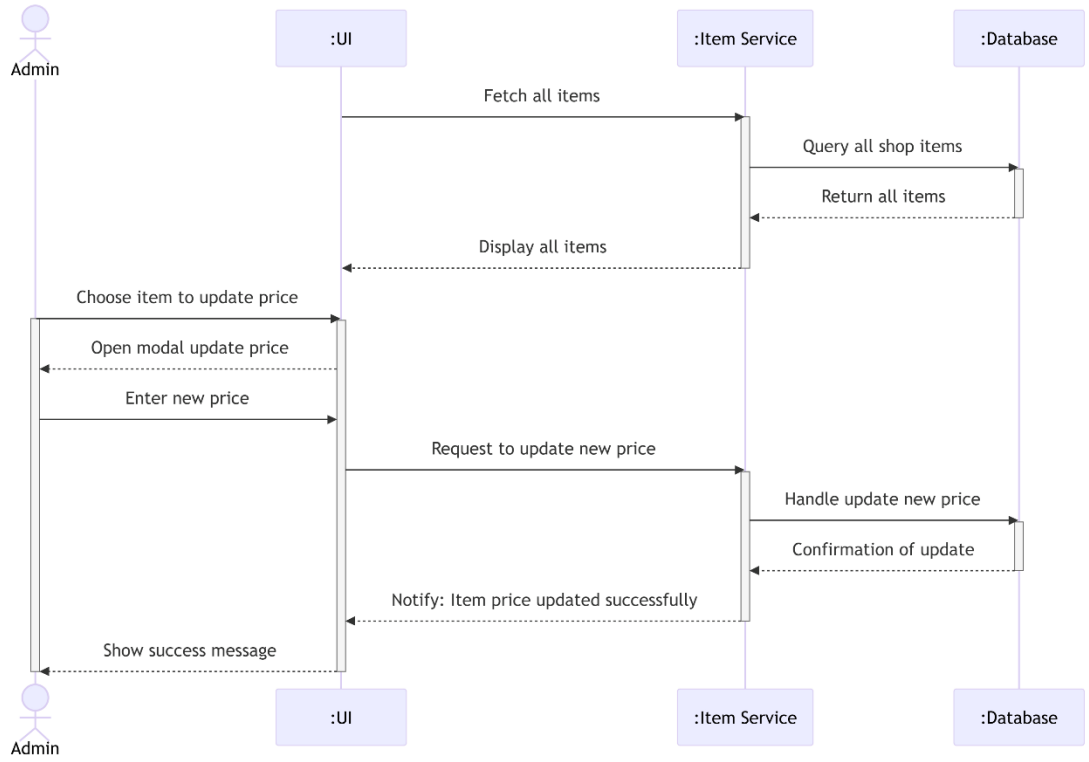

Figure 3.56 Sequence Diagram: Edit item price

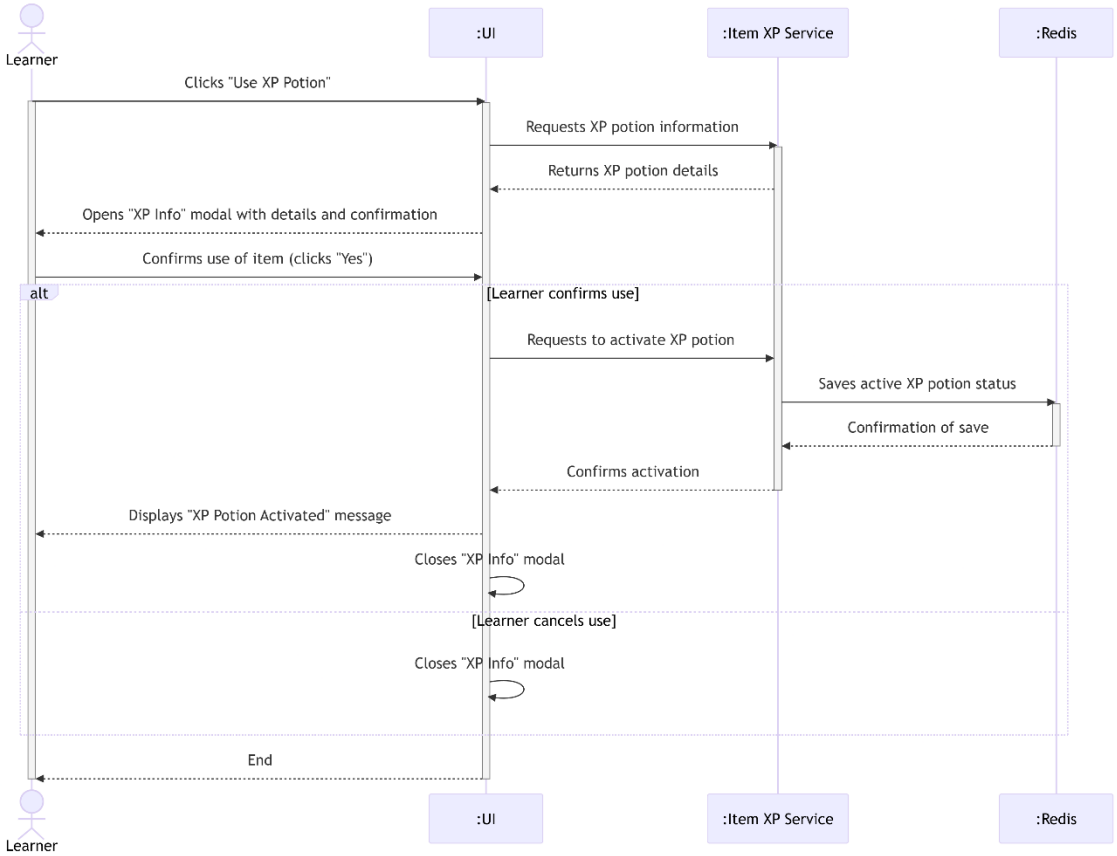

Figure 3.57 Sequence Diagram: Use bonus XP potion

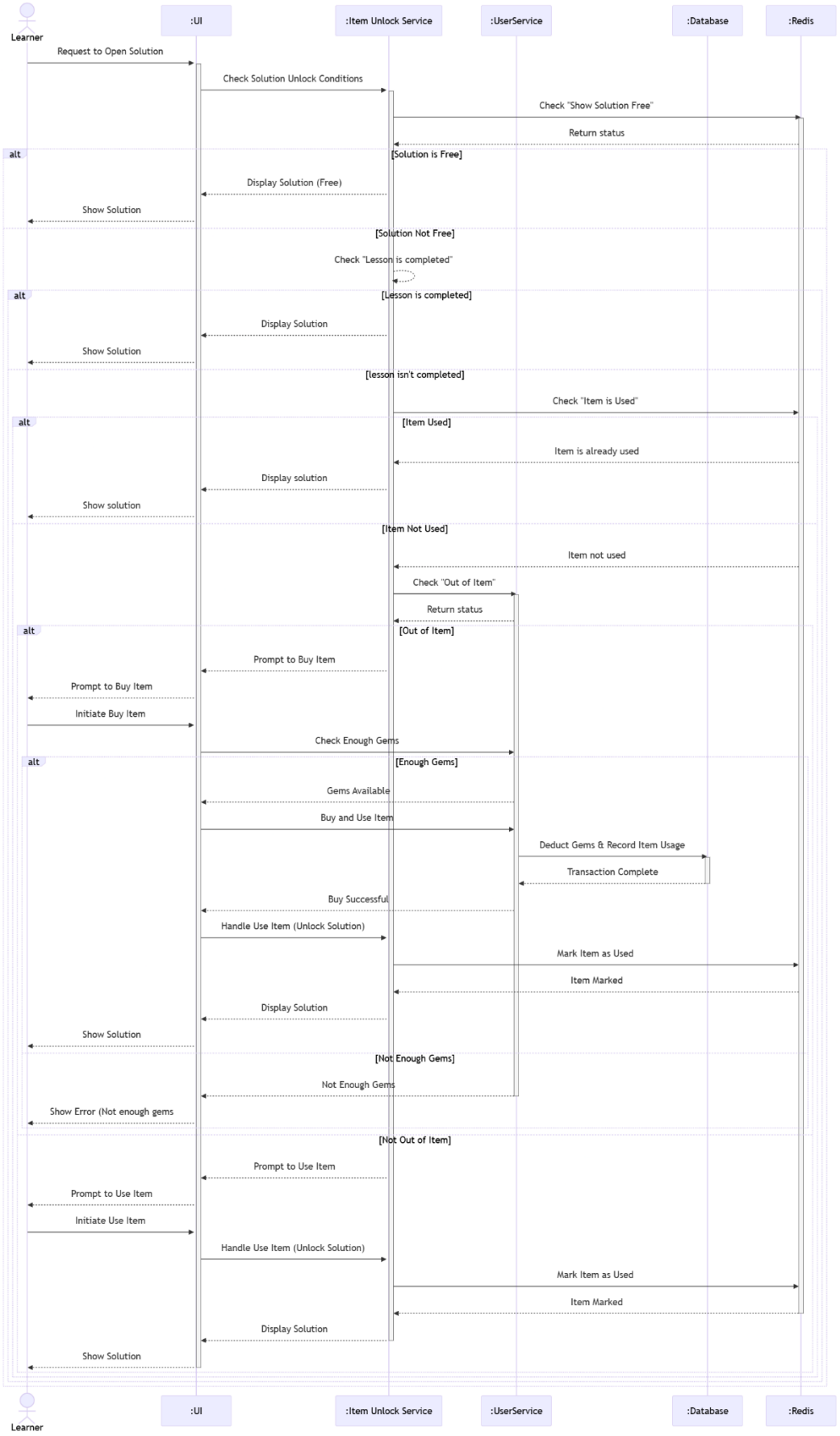

Figure 3.58 Sequence Diagram: Unlock solution

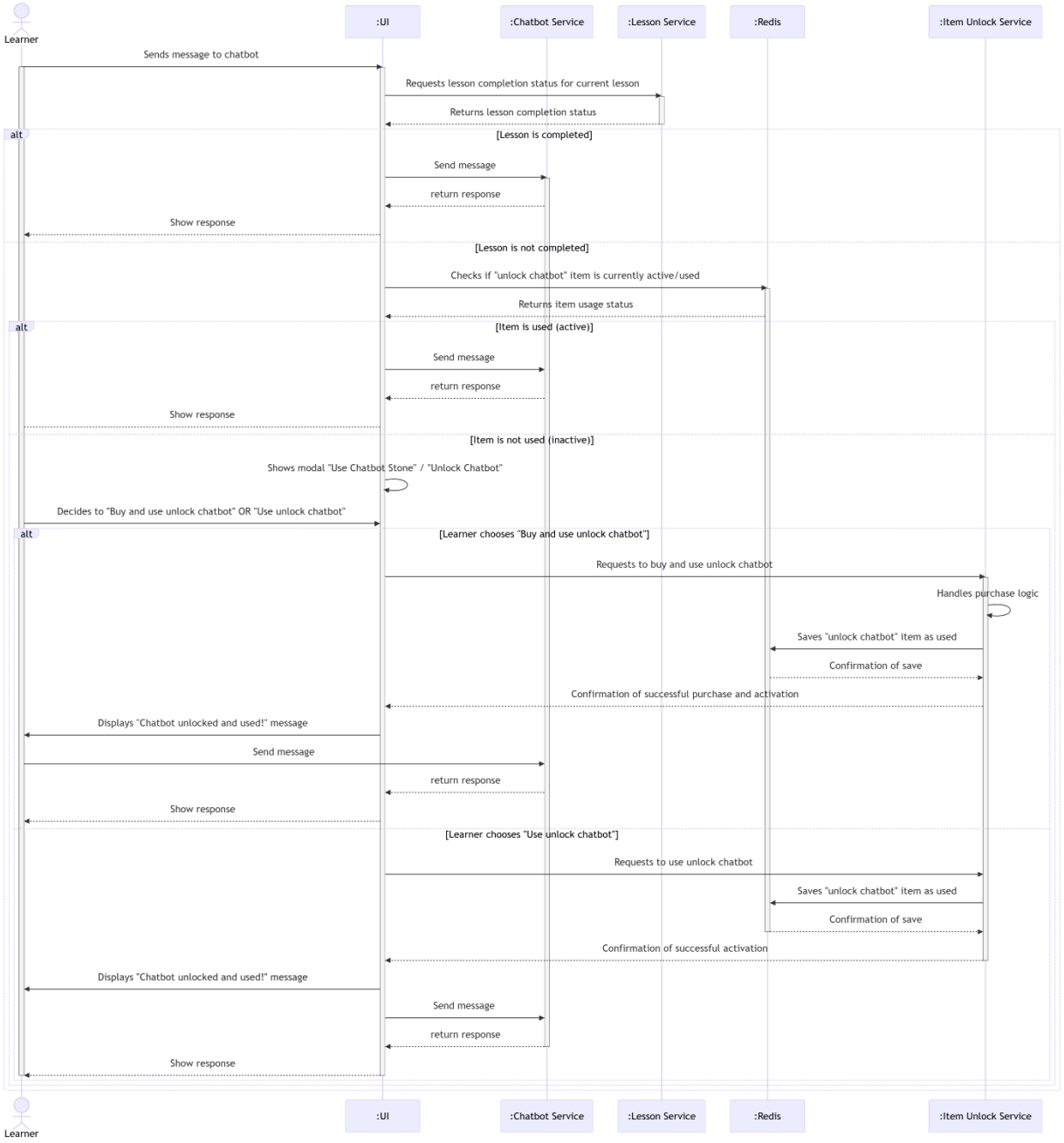

Figure 3.59 Sequence Diagram: Unlock chatbot

3.5.4. Shop

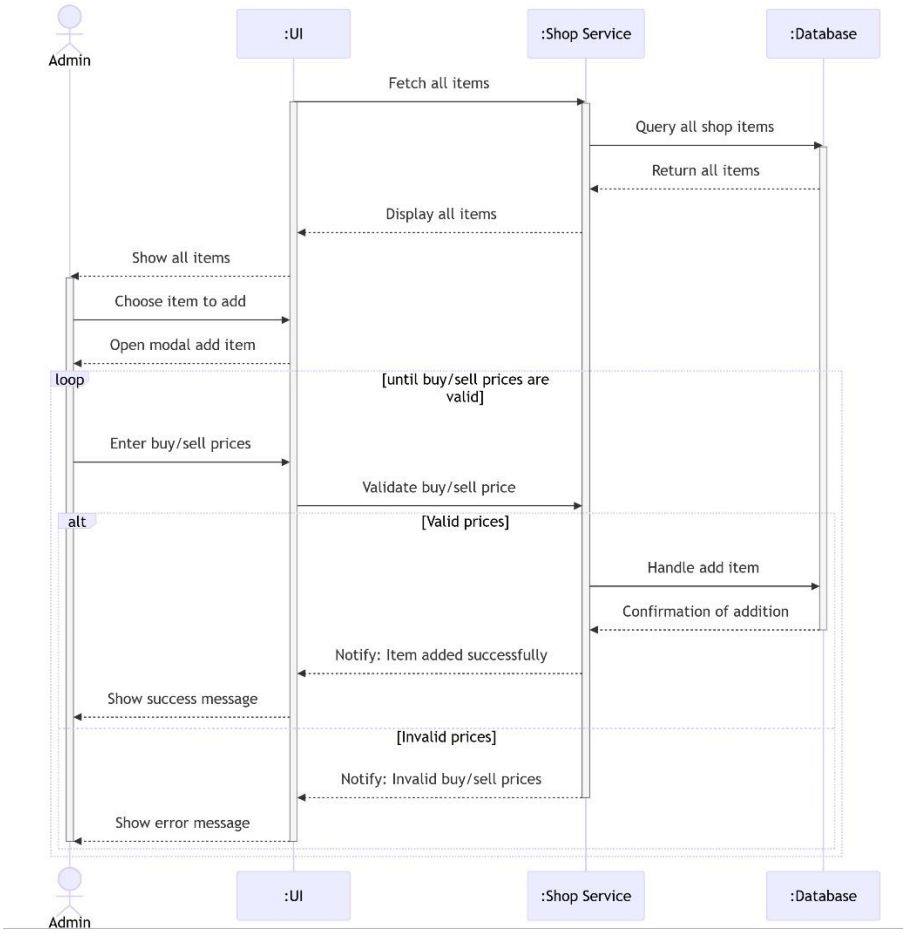

Figure 3.60 Sequence Diagram: Add item to shop

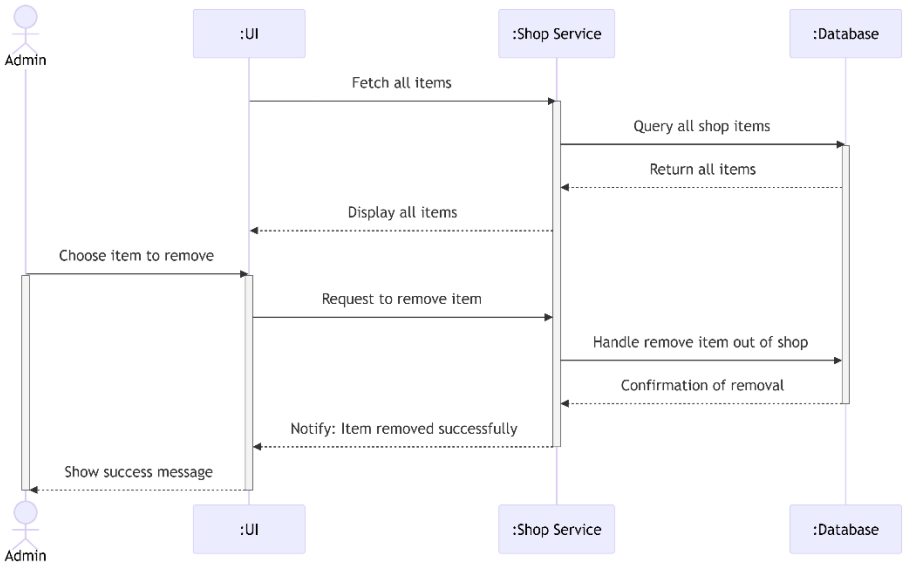

Figure 3.61 Sequence Diagram: Remove item out of shop

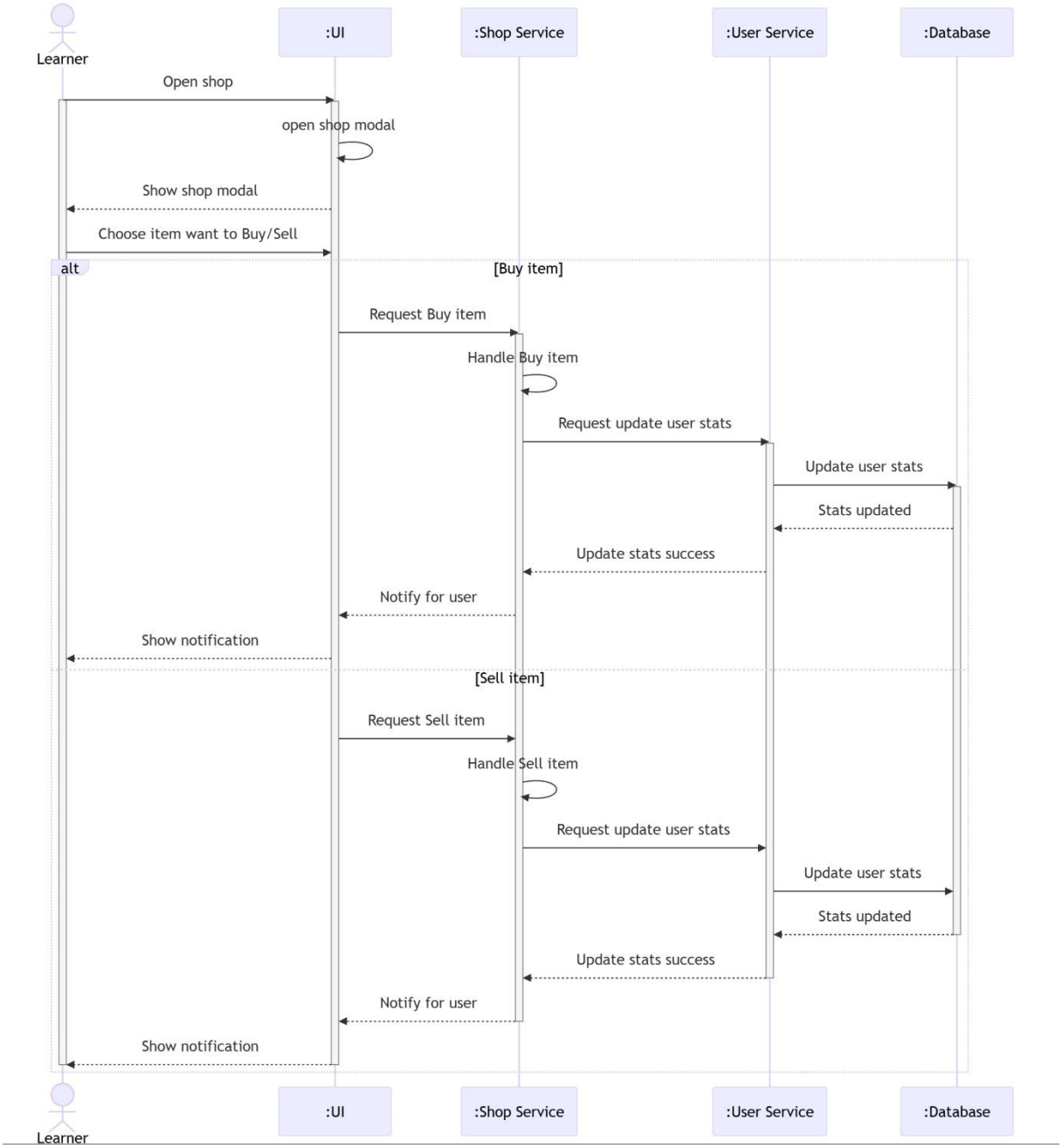

Figure 3.62 Sequence Diagram: Buy/Sell item

3.5.5. Achievement

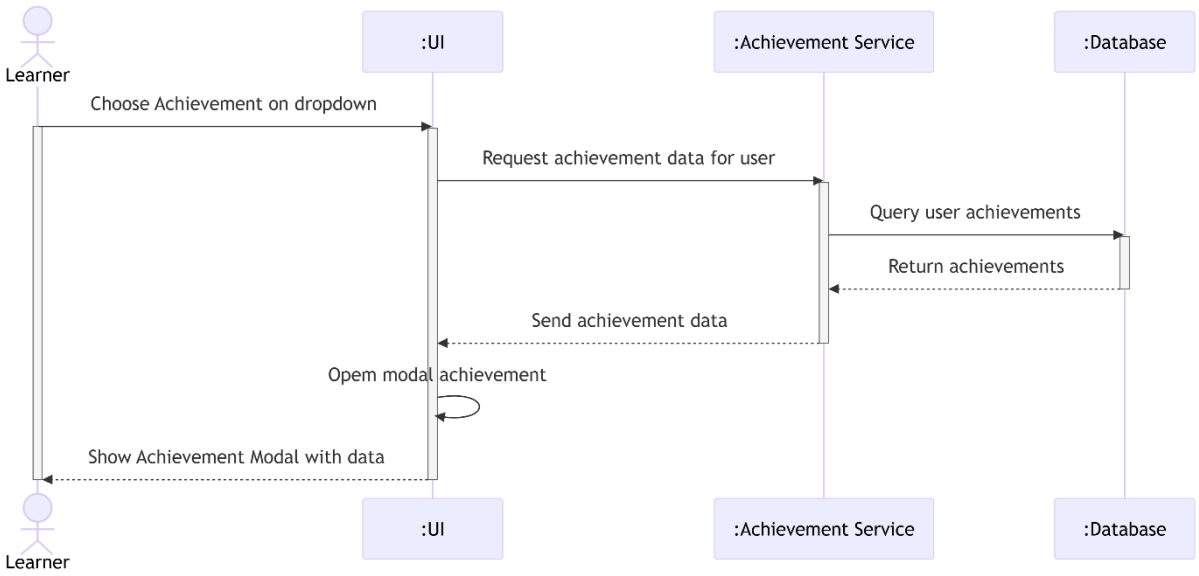

Figure 3.63 Sequence Diagram: View achievement

3.5.6. Leaderboard

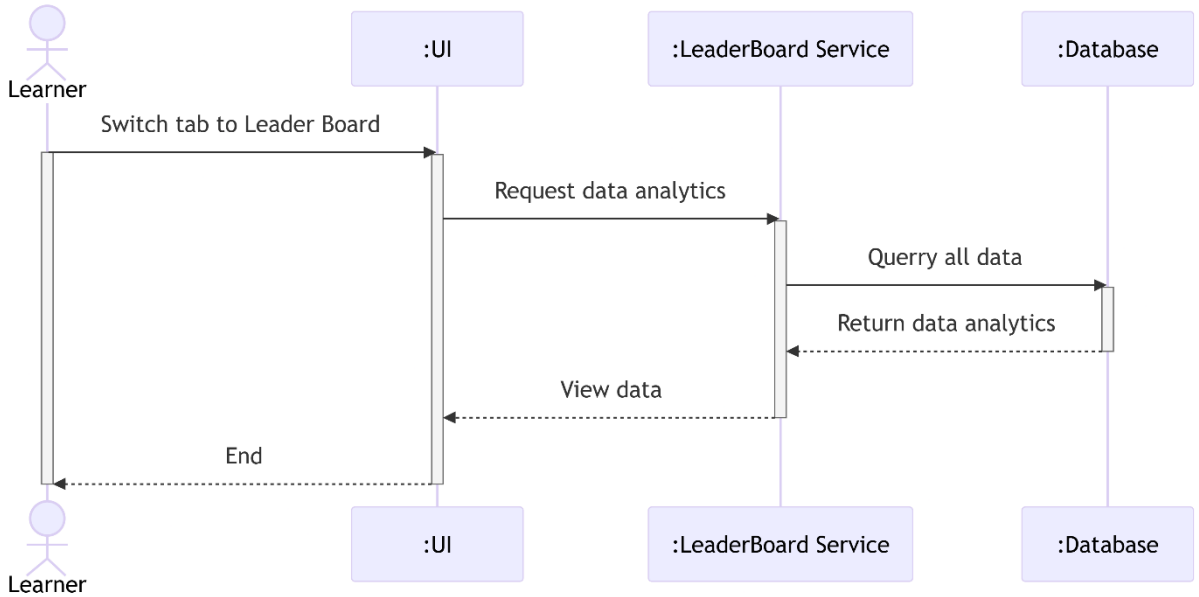

Figure 3.64 Sequence Diagram: View leaderboard

3.5.7. *Profile*

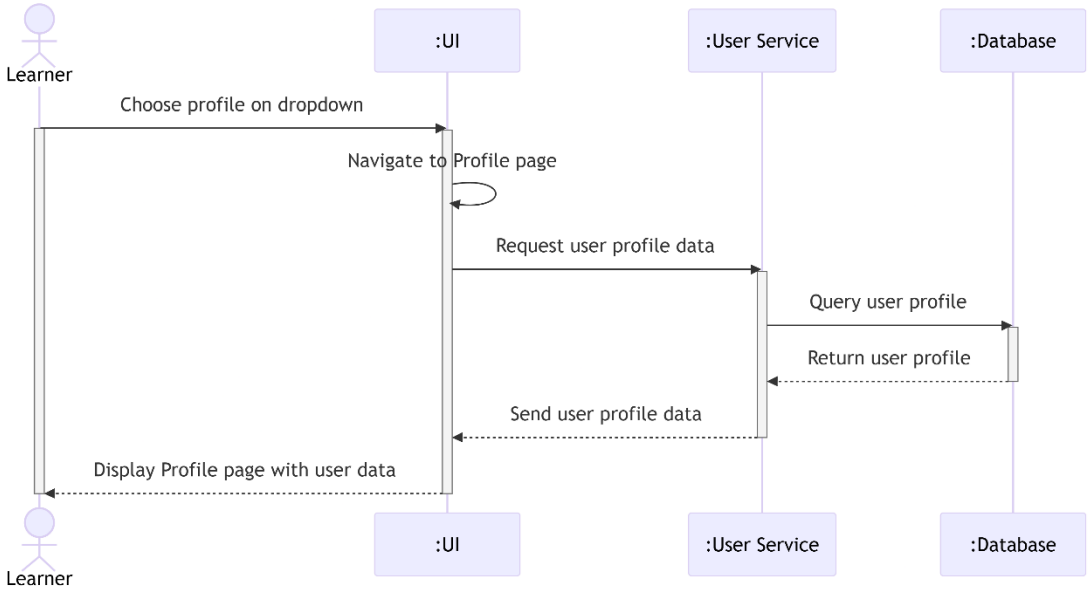

Figure 3.65 Sequence Diagram: View profile

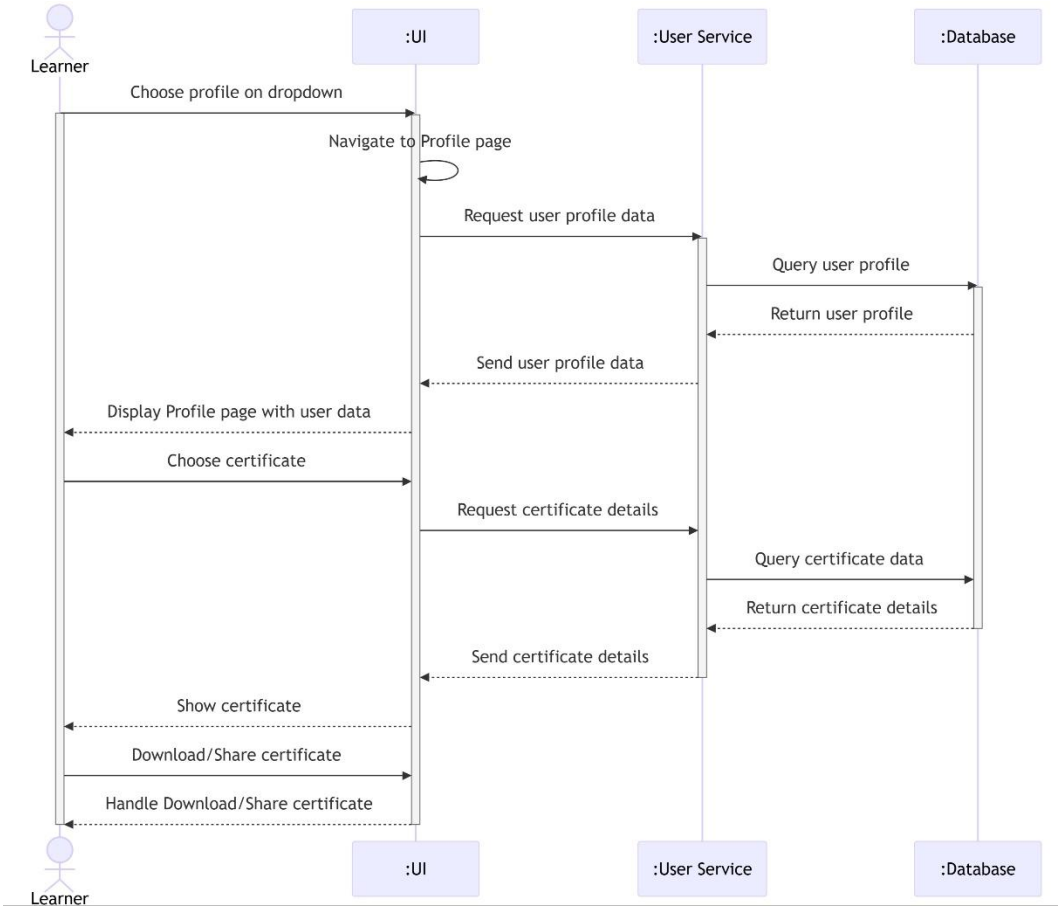

Figure 3.66 Sequence Diagram: Download/Share certificate

3.5.8. Notification

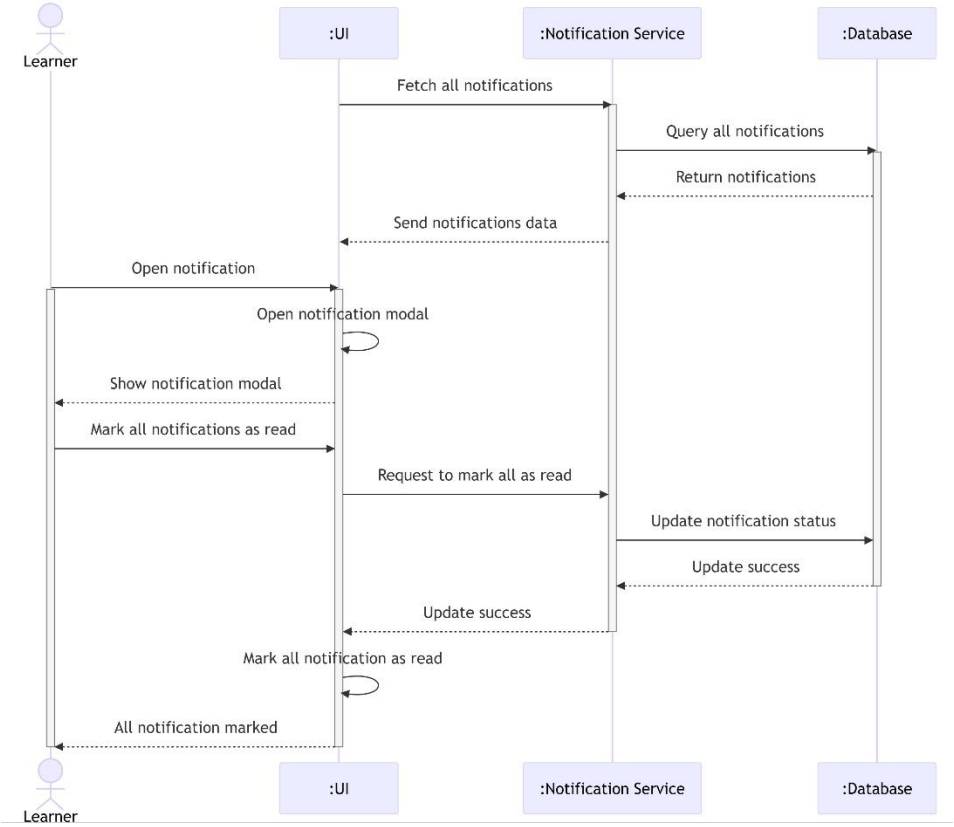

Figure 3.67 Sequence Diagram: Get notifications

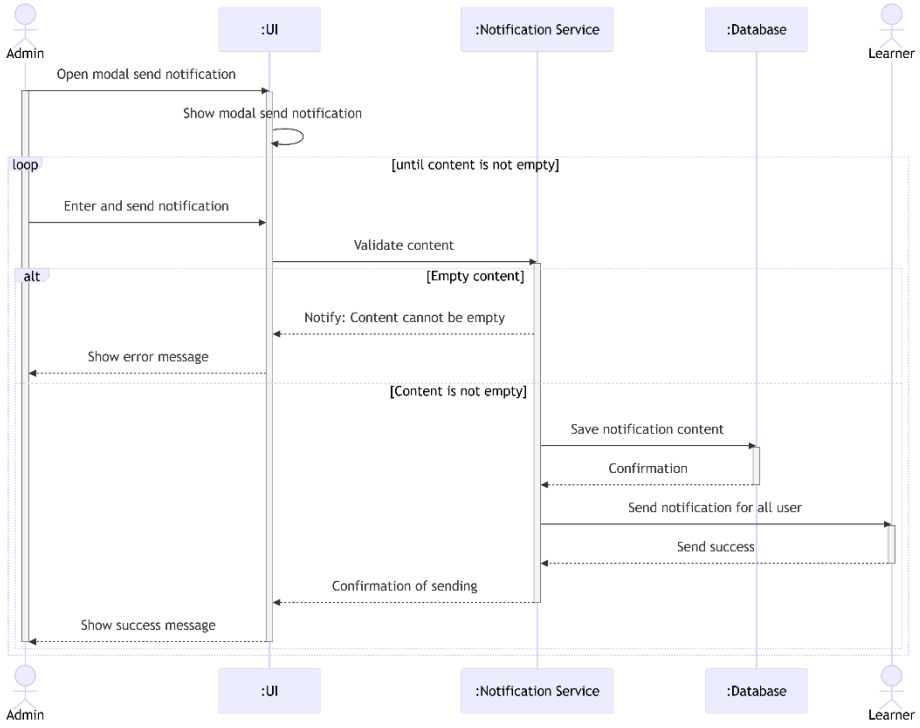

Figure 3.68 Sequence Diagram: Send global notification

3.5.9. Spell book

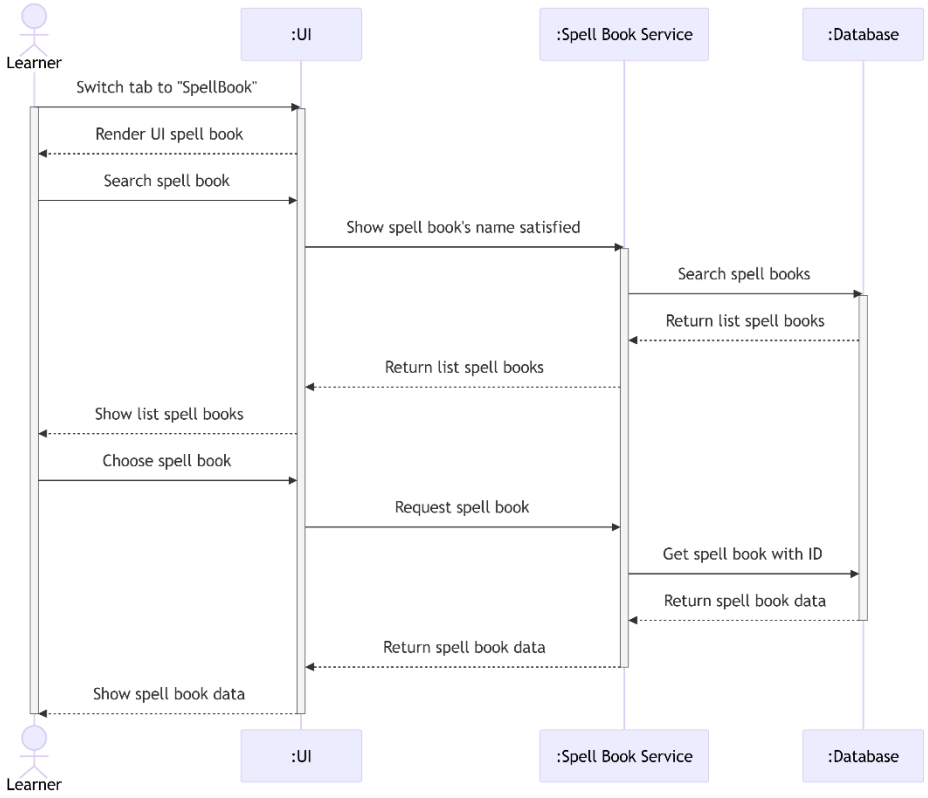

Figure 3.69 Sequence Diagram: Use spell book

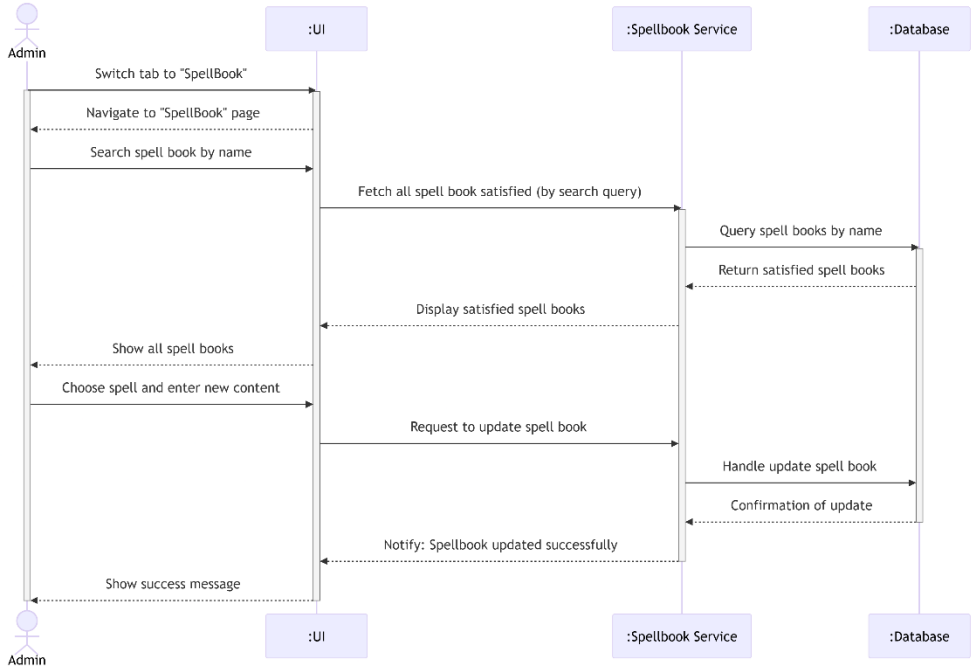

Figure 3.70 Sequence Diagram: Manage spell book

3.5.10. Daily wheel

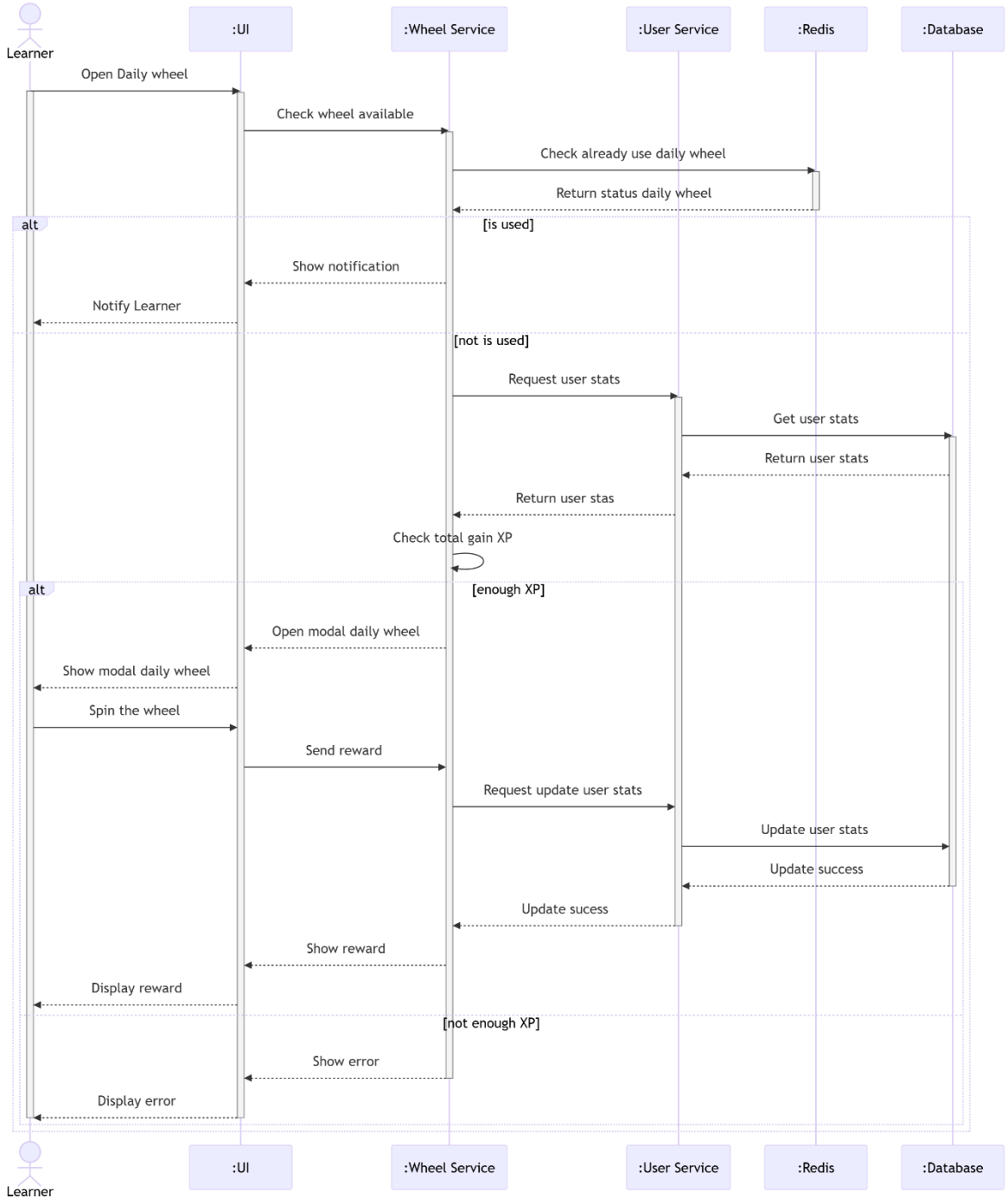

Figure 3.71 Sequence Diagram: Open daily wheel

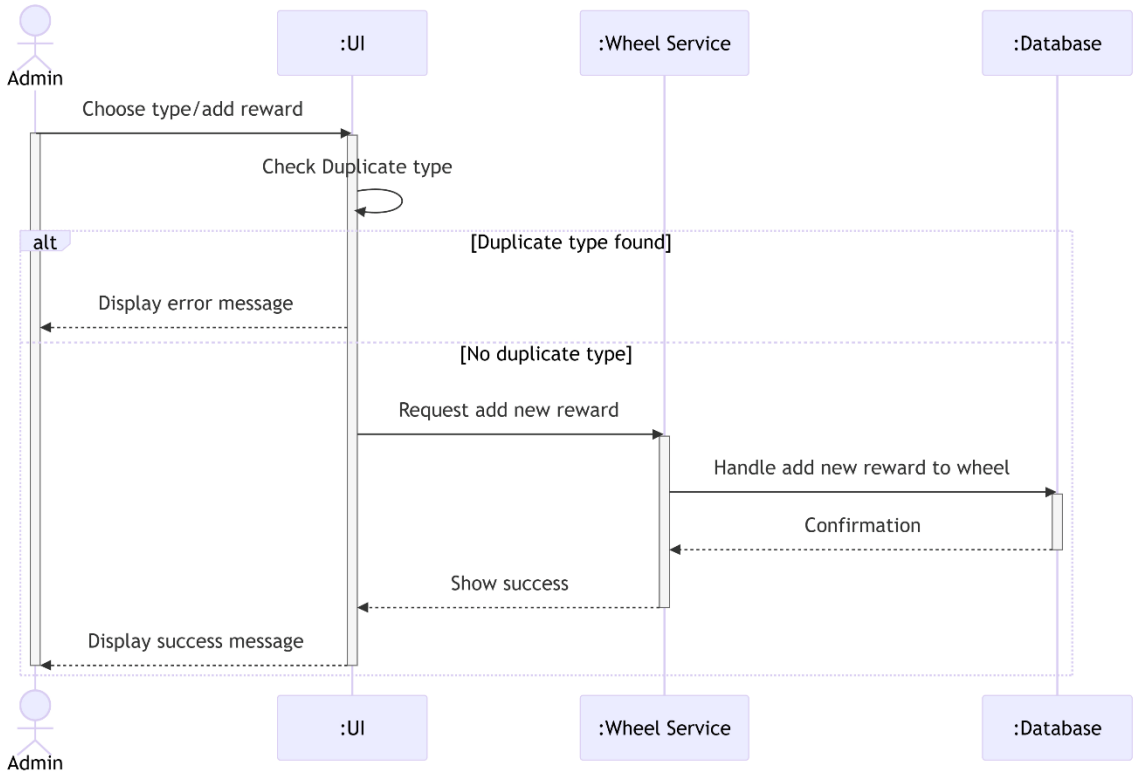

Figure 3.72 Sequence Diagram: Add wheel item

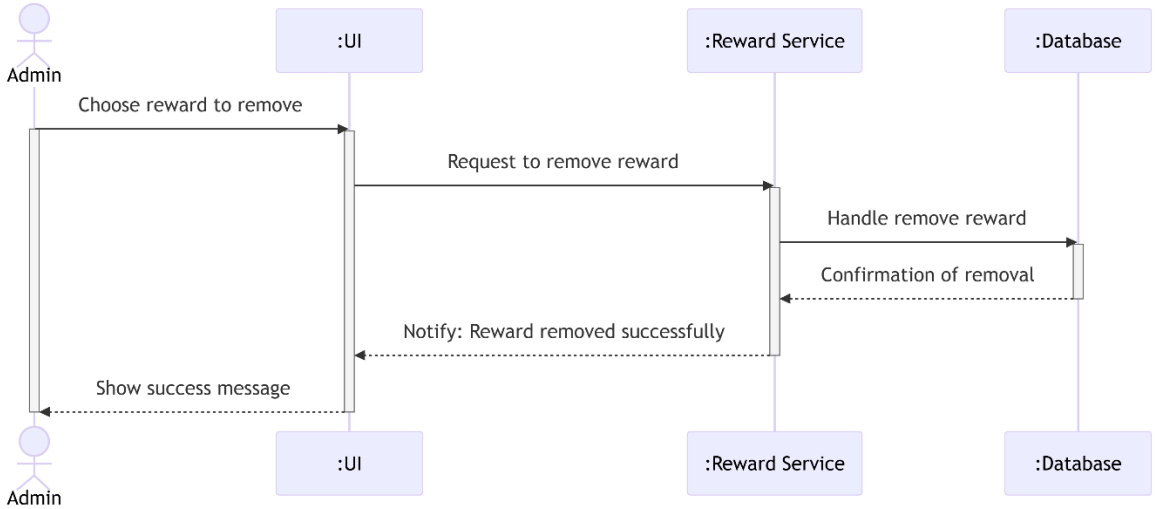

Figure 3.73 Sequence Diagram: Remove wheel item

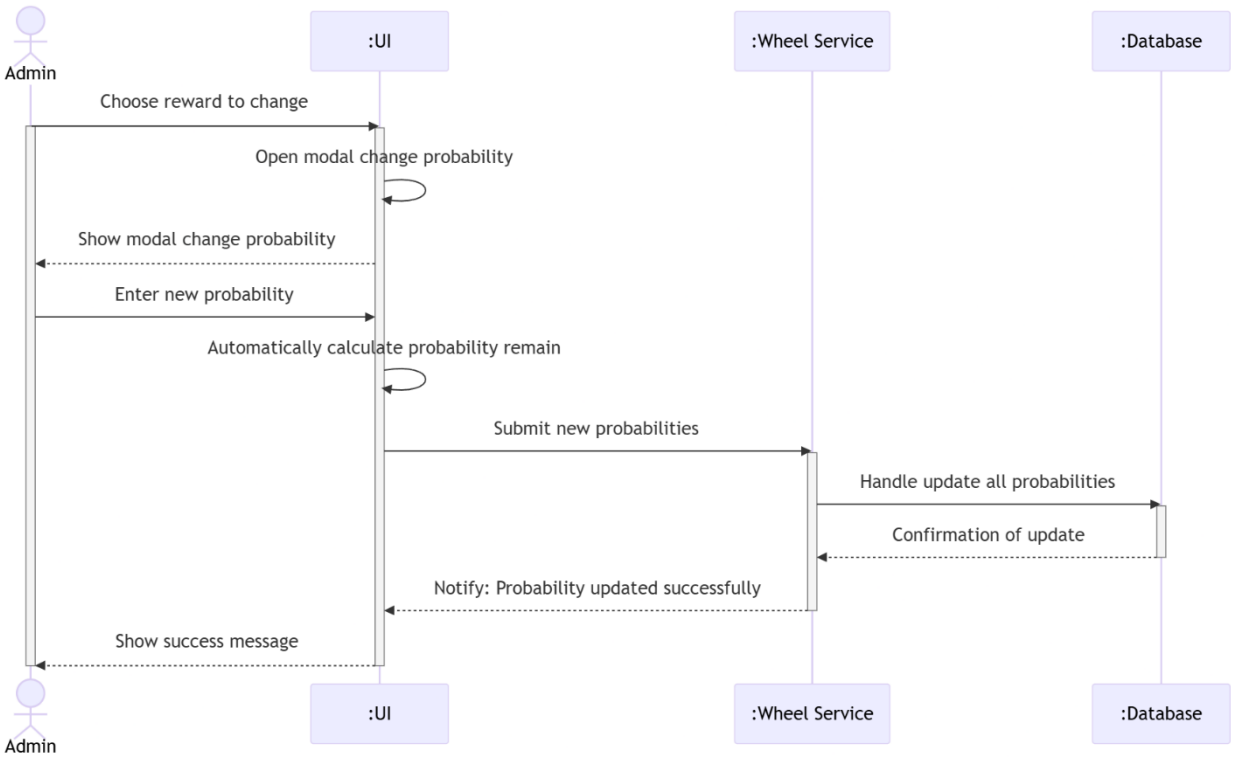

Figure 3.74 Sequence Diagram: Change wheel item probability

3.5.11. Dashboard

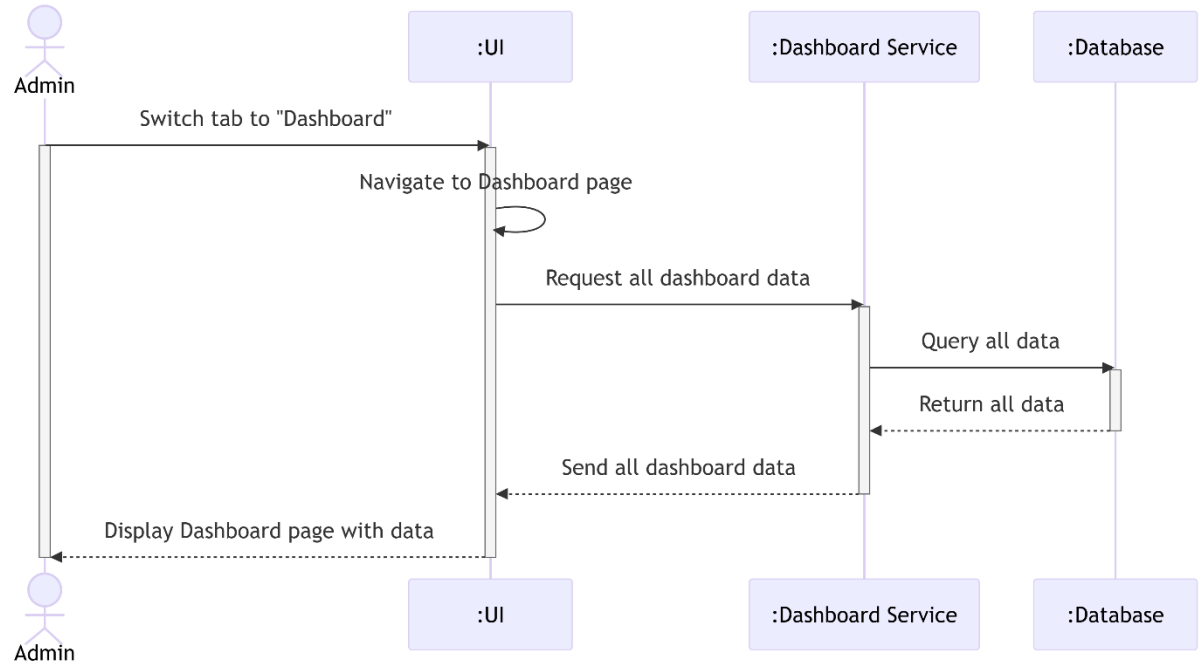

Figure 3.75 Sequence Diagram: View dashboard

3.5.12. Feedback

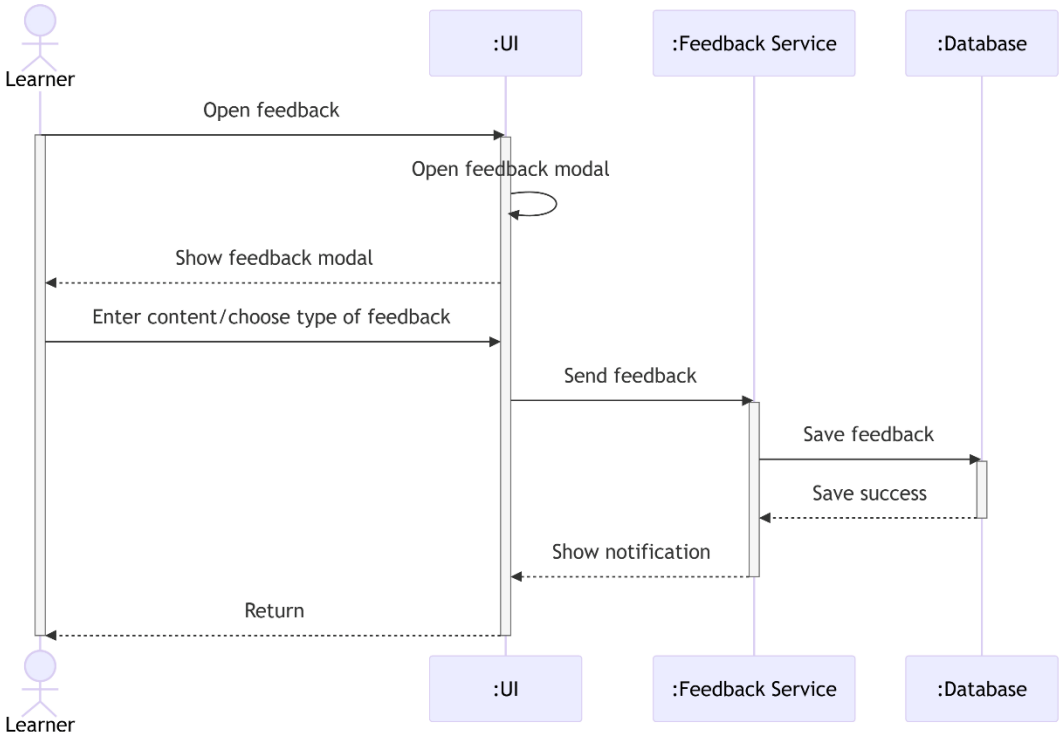

Figure 3.76 Sequence Diagram: Create feedback

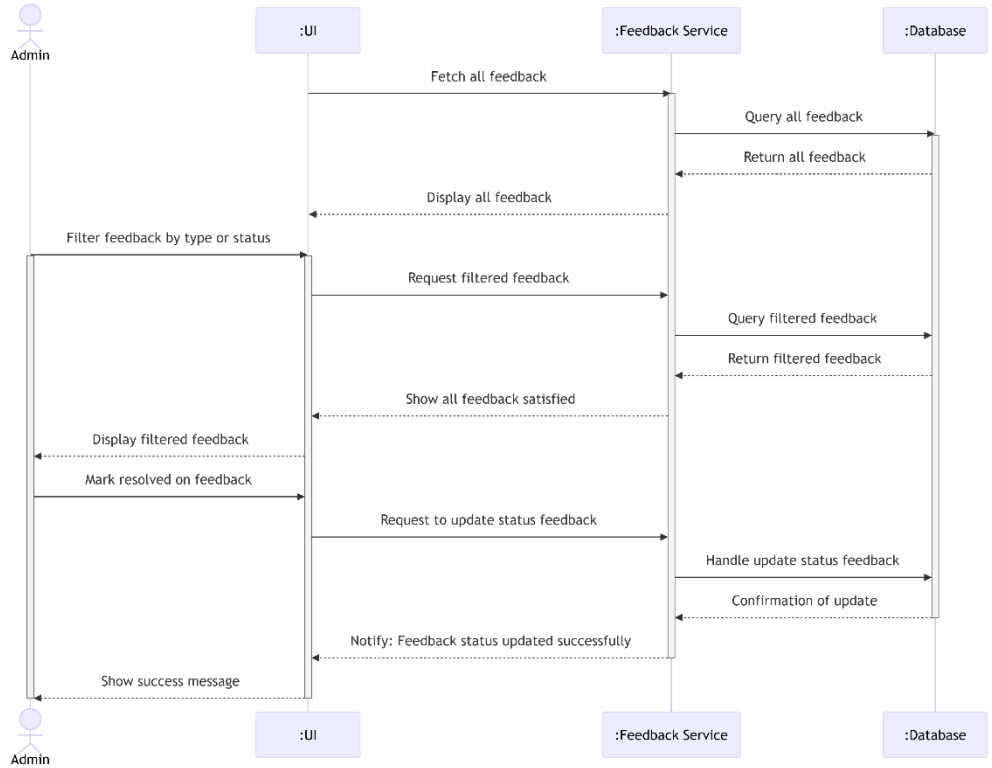

Figure 3.77 Sequence Diagram: Resolve feedback

3.6. Database diagram

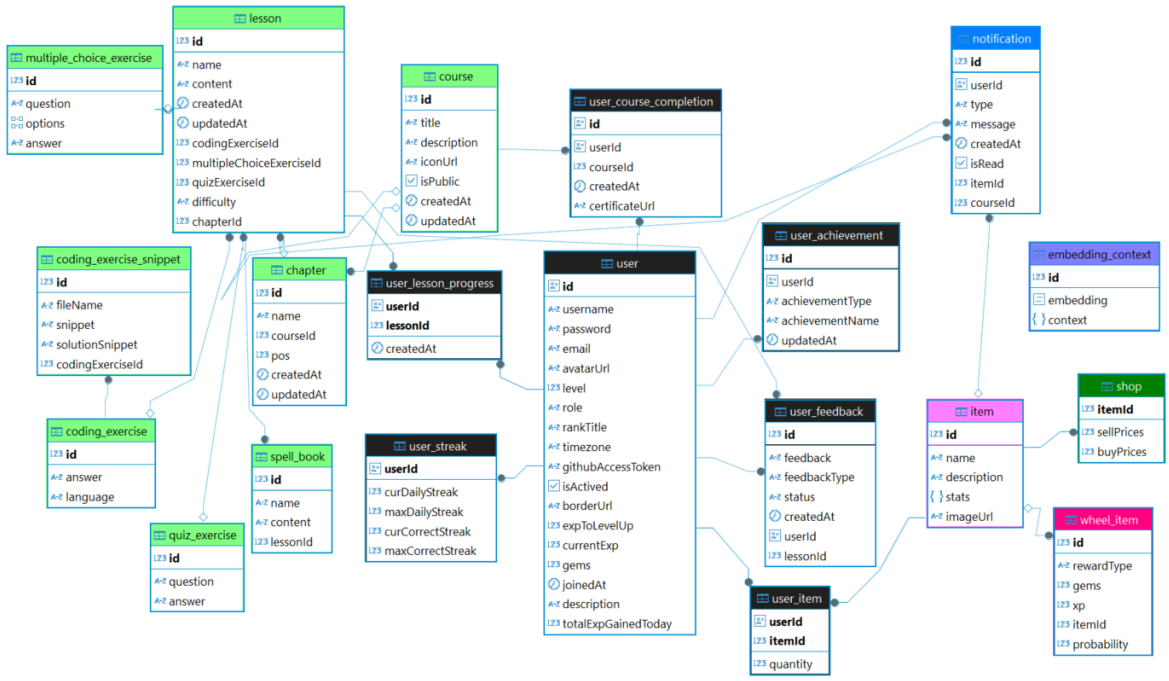

Figure 3.78 Database diagram

3.6.1. Table "user"

Table "user" saves user general information

Table 3. 35 Table user

Column name	Data type	Description
id	uuid	primary key
username	varchar	username of user
password	text	hashed password
email	varchar	email of user
avatarUrl	varchar	avatar url of user
level	int	level of user
rankTitle	varchar	current rank title of user
timezone	varchar	user time zone
githubAccessToken	text	github access token of user

		(if sign in with Github)
isActived	bool	flag for checking if user is verify account or not
borderUrl	varchar	rank border of user
expToLevelUp	int	the require exp need to level up
currentExp	int	current user experience
gems	int	current user gems
joinedAt	timestamp	user join platform date
description	text	user profile description
totalExpGainedToday	int	total user exp gain in one day
role	enum	user role

3.6.2. Table "user_lesson_progress"

Table "user_lesson_progress" saves lesson that user already finished

Table 3.36 Table user_lesson_progress

Column name	Data type	Description
userId	uuid	primary key, foreign key of user table
lessonId	int	primary key, foreign key of lesson table
createdAt	timestamp	the timestamp when user finished lesson

3.6.3. Table "user_course_completion"

Table "user_course_completion" saves course that user already finished

Table 3.37 Table user_course_completion

Column name	Data type	Description
id	uuid	primary key
userId	uuid	foreign key of user table

courseId	int	foreign key of course table
createdAt	timestamp	the timestamp when user finished course
certificateUrl	text	user course certificate url

3.6.4. Table "user_achievement"

Table "user_achievement" saves all user achievement

Table 3.38 Table user_achievement

Column name	Data type	Description
id	int	primary key
userId	uuid	foreign key of user table
achievementType	enum	type of the achievement
achievementName	varchar	name of current achievement
updatedAt	timestamp	latest time user gets new achievement name

3.6.5. Table "user_feedback"

Table "user_feedback" saves all feedback from user

Table 3.39 Table user_feedback

Column name	Data type	Description
id	int	primary key
userId	uuid	foreign key of user table
lessonId	int	foreign key of lesson table
feedback	text	feedback content
feedbackType	enum	feedback type
createdAt	timestamp	the timestamp when user create new feedback
status	enum	feedback status (resolve/open)

3.6.6. Table "user_streak"

Table "user_streak" saves all user daily streak and correct streak

Table 3.40 Table user_streak

Column name	Data type	Description
userId	uuid	primary key, foreign key of user table
curDailyStreak	int	current user daily streak
maxDailyStreak	int	highest user daily streak
curCorrectStreak	int	current user correct streak
maxCorrectStreak	int	highest user correct streak

3.6.7. Table "user_item"

Table "user_item" saves item quantity that each users have

Table 3.41 Table user_item

Column name	Data type	Description
userId	uuid	primary key, foreign key of user table
itemId	int	primary key, foreign key of item table
quantity	int	amount of item that users have

3.6.8. Table "course"

Table "course" saves main information about course

Table 3.42 Table course

Column name	Data type	Description
id	int	primary key
title	varchar	course title
description	text	course description
iconUrl	text	course image url
isPublic	bool	flag to check if course is private or public
createdAt	timestamp	the date course created
updatedAt	timestamp	the latest date course updated

3.6.9. Table "chapter"

Table "chapter" saves chapters inside course

Table 3.43 Table chapter

Column name	Data type	Description
id	int	primary key
name	varchar	lesson name
courseId	int	foreign key of course
pos	float	position of each chapter
createdAt	timestamp	the date chapter created
updatedAt	timestamp	the latest date chapter updated

3.6.10. Table "lesson"

Table "lesson" saves lessons inside chapter

Table 3.44 Table lesson

Column name	Data type	Description
id	int	primary key
chapterId	int	foreign key of chapter table
name	varchar	lesson name
content	text	lesson content
createdAt	timestamp	the date lesson created
updatedAt	timestamp	the latest date lesson updated
codingExerciseId	int	foreign key of coding_exercise table
multipleChoiceExerciseId	int	foreign key of multiple_choice_exercise
quizExerciseId	int	foreign key of quiz_exercise
difficulty	enum	lesson difficulty level

3.6.11. Table "coding_exercise"

Table "coding_exercise" save the lesson coding exercise

Table 3.45 Table coding_exercise

Column name	Data type	Description
id	int	primary key
answer	text	coding exercise answer
language	varchar	the code language

3.6.12. Table "coding_exercise_snippet"

Table "coding_exercise_snippet" save all snippet of the coding exercise

Table 3.46 Table coding_exercise_snippet

Column name	Data type	Description
id	int	primary key
filename	varchar	snippet file name
snippet	text	snippet code template
solutionSnippet	text	snippet code solution
codingExerciseId	int	foreign key of coding_exercise

3.6.13. Table "multiple_choice_exercise"

Table "multiple_choice_exercise" saves the lesson multiple choices exercise

Table 3.47 Table multiple_choice_exercise

Column name	Data type	Description
id	int	primary key
question	text	multiple choice question
options	text[]	list of choices answer
answer	text	correct answer

3.6.14. Table "quiz_exercise"

Table "quiz_exercise" saves the lesson quiz exercise

Table 3.48 Table quiz_exercise

Column name	Data type	Description
id	int	primary key
question	text	quiz question
answer	text	correct answer

3.6.15. Table "spell_book"

Table "spell_book" saves the lesson spell book

Table 3.49 Table spell_book

Column name	Data type	Description
id	int	primary key
lessonId	int	foreign key of lesson table
content	text	spell book content
name	text	spell book title

3.6.16. Table "notification"

Table "notification" saves all notification from server send to user

Table 3.50 Table notification

Column name	Data type	Description
id	int	primary key
userId	uuid	foreign key of user table
itemId	int	foreign key of item table
coursed	int	foreign key of course table
type	varchar	notification type
message	text	notification content
createdAt	timestamp	the date notification created
isRead	bool	flag check if notification is read or not

3.6.17. Table "item"

Table "item" saves all item data and it stats

Table 3.51 Table item

Column name	Data type	Description
id	int	primary key
name	varchar	item name
description	text	item description
stats	jsonb	item stats
imageUrl	varchar	item image url

3.6.18. Table "shop"

Table "shop" saves all item in shop and it prices

Table 3.52 Table shop

Column name	Data type	Description
itemId	int	primary key, foreign key of item table
sellPrices	int	item sell prices
buyPrices	int	item buy prices

3.6.19. Table "wheel_item"

Table "wheel_item" saves all item in reward wheel

Table 3.53 Table wheel_item

Column name	Data type	Description
id	int	primary key, foreign key of user table
rewardType	enum	wheel reward type
gems	int	amount of reward gems
xp	int	amount of reward xp
itemId	int	foreign key of item
probability	float	item reward rate

3.6.20. Table "embedding_context"

Table "embedding_context" saves general context of the platform, for chatbot use purpose

Table 3.54 Table embedding_context

Column name	Data type	Description
id	int	primary key
embedding	vector	embedding vector of the context
context	jsonb	context and the answer

Chapter 4: SYSTEM DEPLOYMENT AND EVALUATION

4.1. Technology stack

- **Back-end:** NestJS
- **Front-end:** ReactJS, Tailwind, CSS [9] [10]
- **Deployment:** Docker, Nginx, Remote Virtual Machine, Github Action [11]
- **API Document & Testing:** Swagger
- **Cache:** RedisCloud [12]
- **Database:** Supabase(PostgreSQL), S3 storage [13] [14]
- **Code execution:** Piston API [15]
- **Large language model:** Gemini
- **Text embedding model:** Text-embedding-004
- **Mail service:** Brevo [16]

4.2. System overview

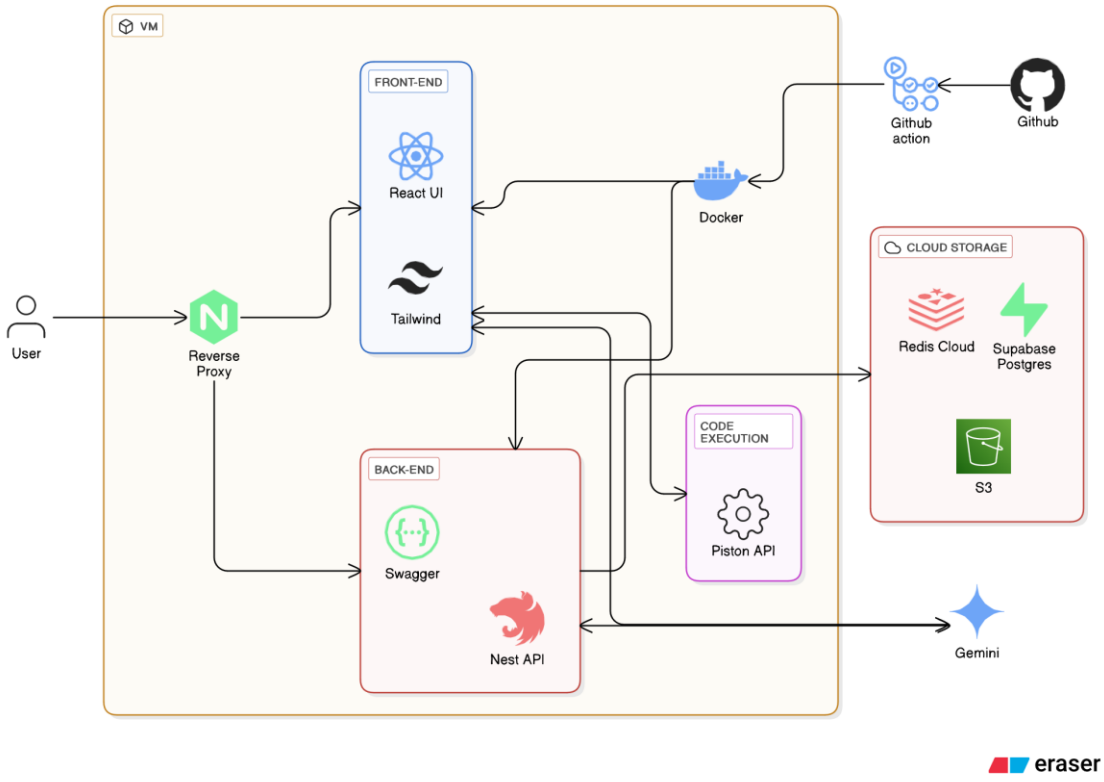

Figure 4.1 System overview

This system is a full-stack web application deployed on a Virtual Machine (VM), comprising a React-based frontend, a NestJS backend, external code execution and AI services, and integrated cloud storage. The system is designed to be modular, scalable, and developer-friendly, leveraging modern DevOps and cloud-native tools.

- **Front-end:**
  - **React UI** is used to build a responsive, component-driven user interface for interacting with the application.
  - **Tailwind CSS** provides a utility-first approach for styling, enabling rapid UI development with minimal custom CSS.
- **Back-end:**
  - **NestJS** serves as the main backend framework, offering a structured and scalable way to build APIs using TypeScript.
  - **Swagger** integration automatically documents all available API endpoints, making the system more maintainable and accessible for developers.
- **Reverse Proxy (Nginx):**
  - Nginx acts as a **reverse proxy**, directing traffic from users to the appropriate service (React frontend or NestJS backend) based on the request path.
  - This improves security, performance, and maintainability by centralizing routing and HTTPS termination.
- **Code Execution:**
  - The system integrates with the **Piston API**, an external service for running user-submitted code snippets in multiple languages.
  - This provides secure and isolated code execution, useful for coding platforms, compilers, or educational tools.
- **Cloud Storage:**
  - **Supabase (Postgres)**: Used for relational data storage such as users, lessons, and progress tracking.
  - **Redis Cloud**: Serves as a caching layer or session manager to improve response time and reduce database load.
  - **Amazon S3**: Stores static assets and files

- **Docker + GitHub Actions:**
    - The application uses **Docker** to containerize both frontend and backend services, ensuring consistent environments across development and production.
    - **GitHub Actions** automates CI/CD workflows such as linting, building Docker images, and deploying to the VM.

This platform is configured with two separate environments—**development** and **production**—to ensure efficient testing, debugging, and deployment. The development environment facilitates rapid iteration and debugging during the build phase, while the production environment is optimized for stability, performance, and secure end-user access.

Table 4.1 Deployment Environment

	Front-end	Back-end
Development	https://dev.gaumeodathanh.me	https://dbe.gaumeodathanh.me/api
Production	https://gaumeodathanh.me	https://be.gaumeodathanh.me/api

4.3. System features

4.3.1. Assets

All design assets used in this project are sourced from free Figma community resources and have been customized to align with the platform's branding and user experience requirements.

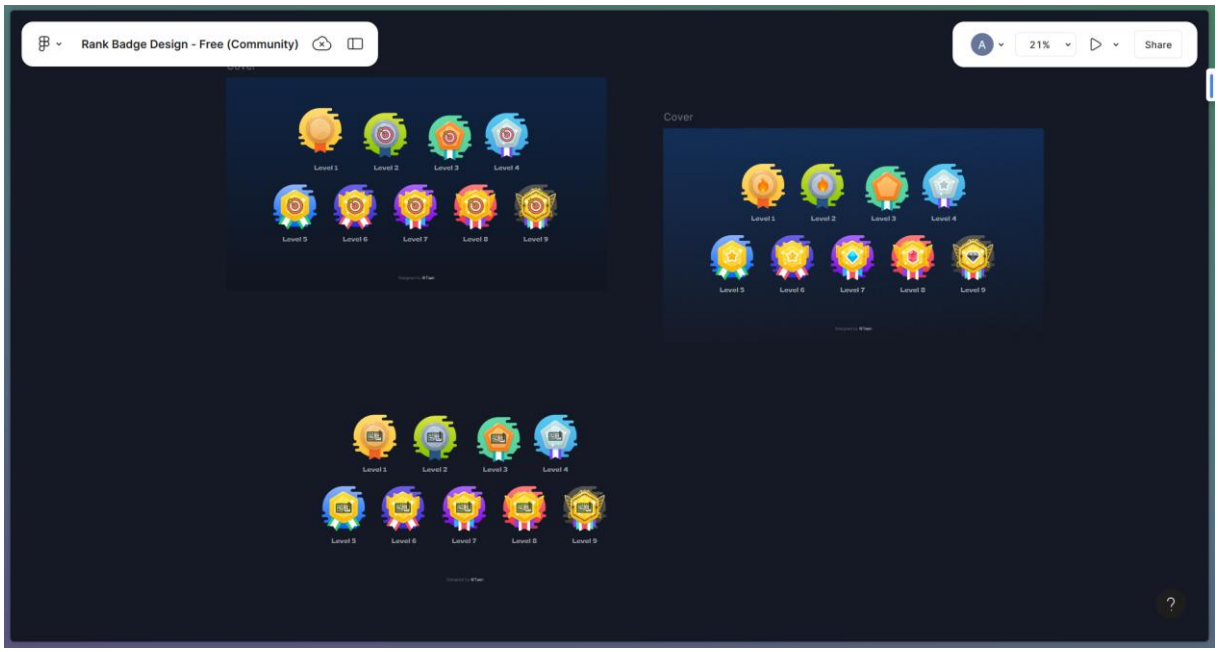

Figure 4.2 Achievement Badge (source)

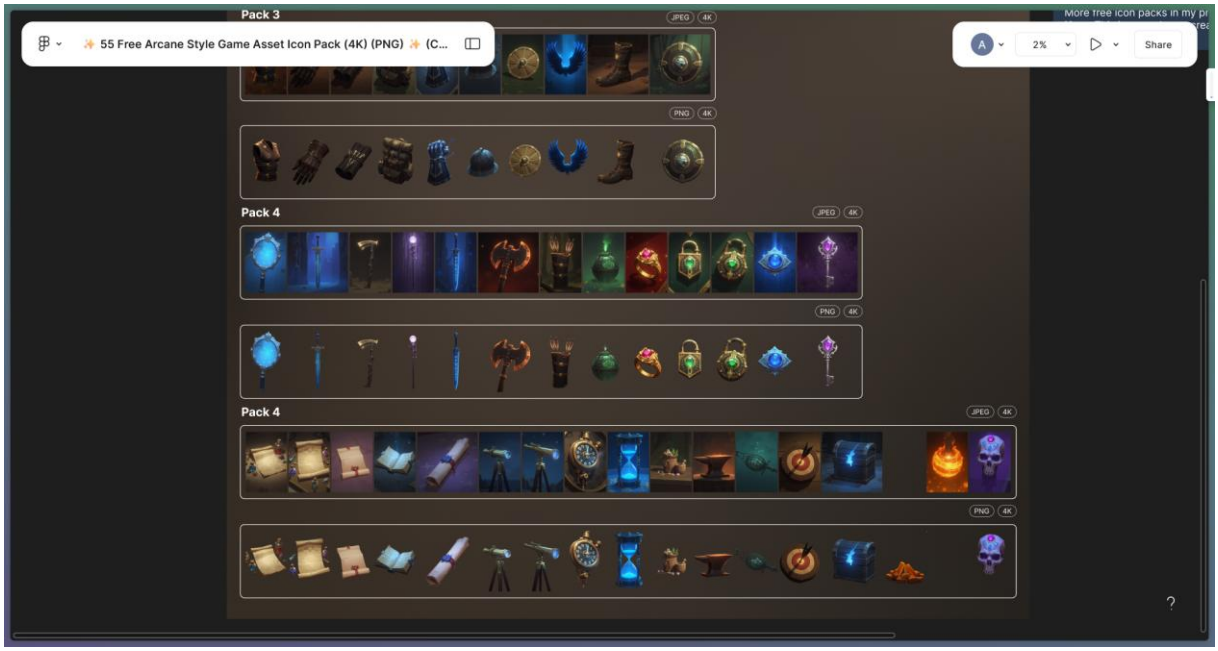

Figure 4.3 Item assets (source)

4.3.2. Learner

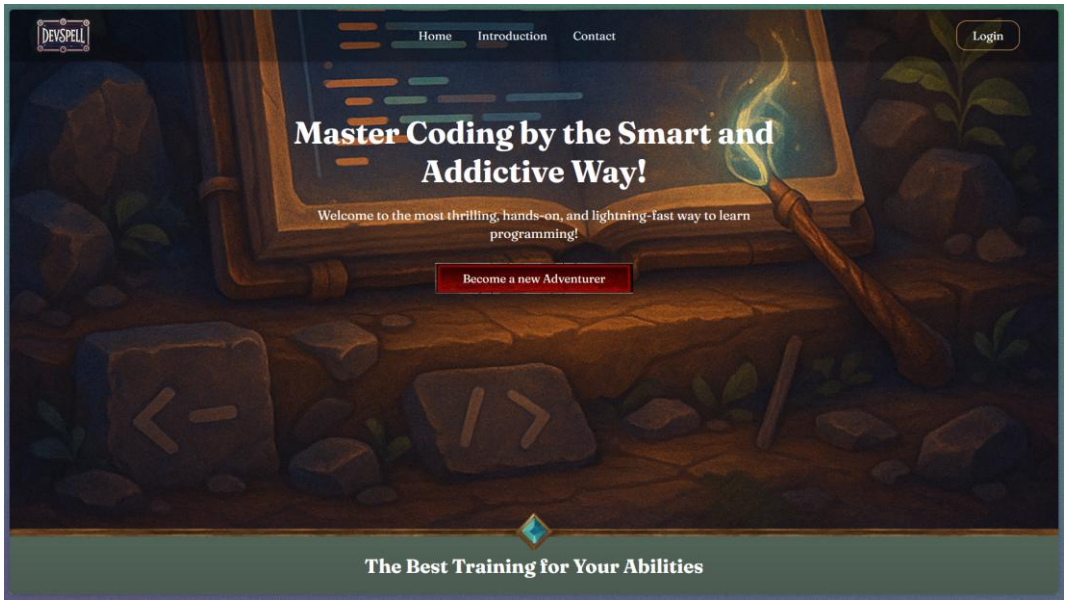

Figure 4.4 Landing page

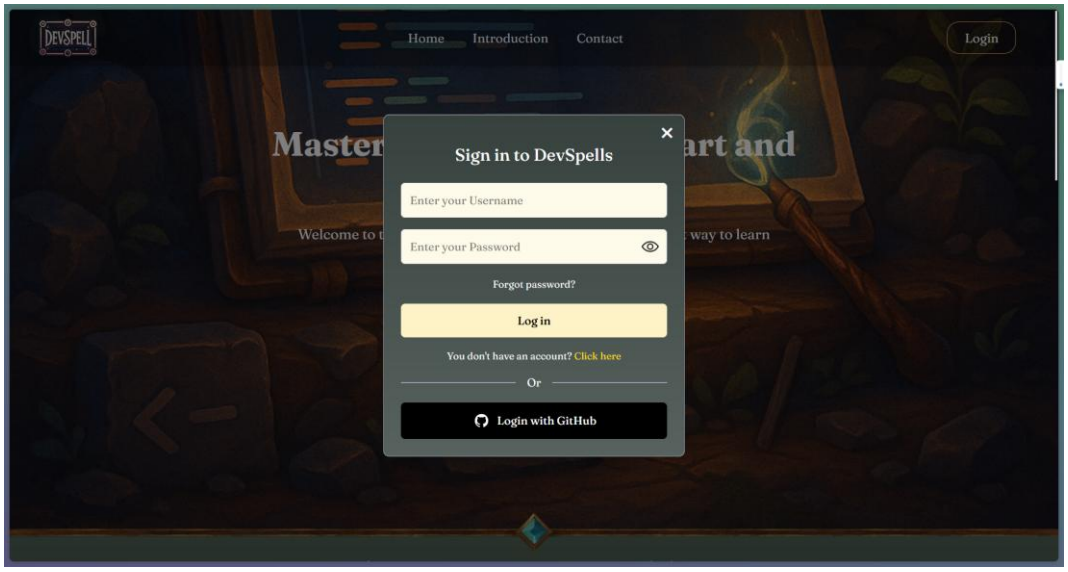

Figure 4.5 Sign in page

In **Sign in page**, users can log in using their username and password or sign in with a GitHub account. The page also provides options to reset a forgotten password or create a new account. Navigation links to the Home, Introduction, and Contact pages are available at the top.

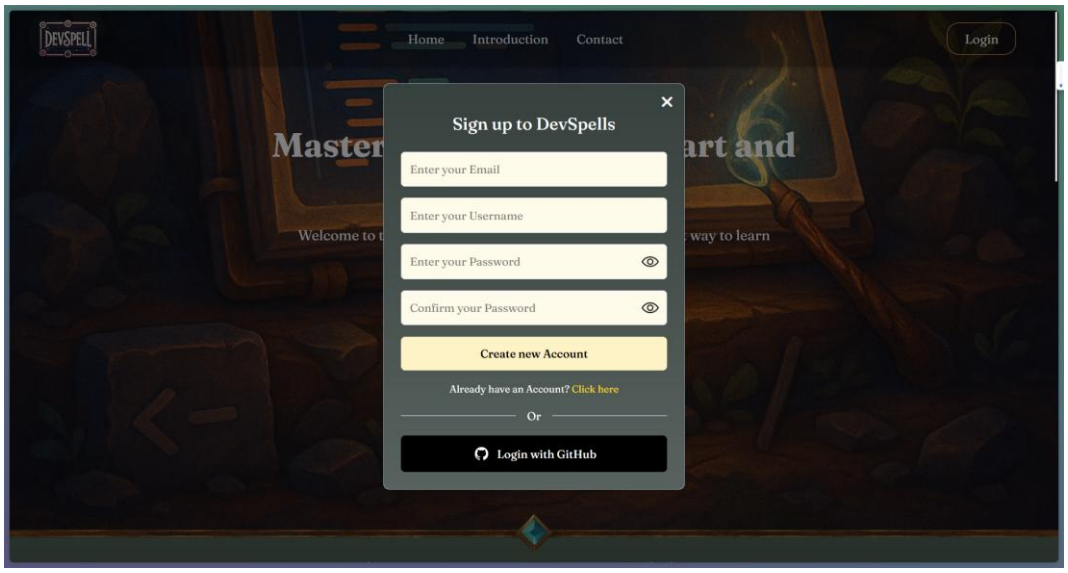

Figure 4.6 Sign up page

Sign-Up Page is used for account creation. New users can sign up by entering their email, username, and password (with confirmation). There's also an option to register using a GitHub account. A link is provided for users who already have an account to switch to the login screen.

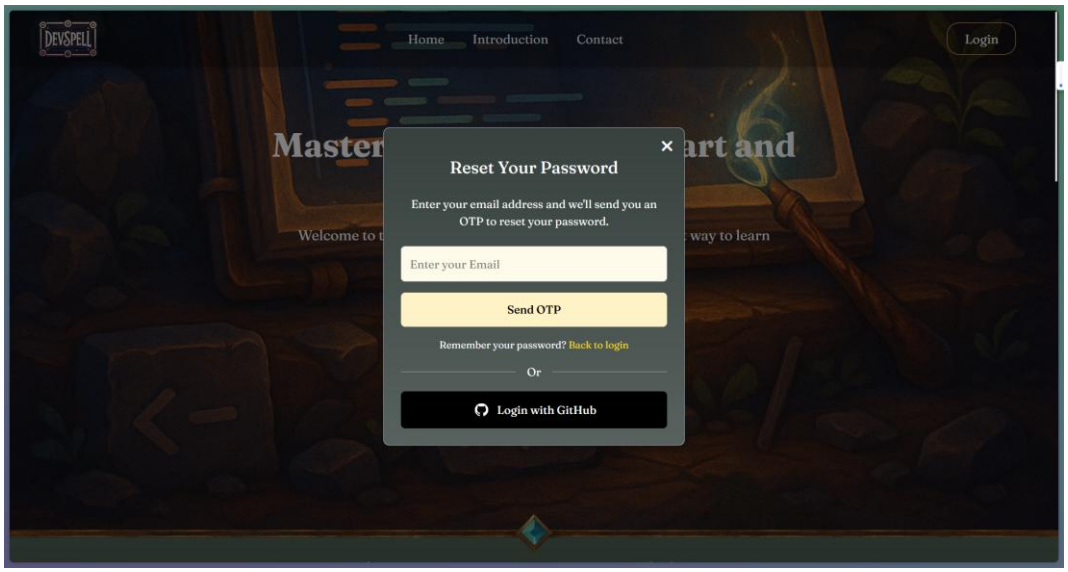

Figure 4.7 Reset password page

Reset Password Page allows users to reset their password by entering their email address. An OTP (One-Time Password) will be sent to their email to verify the request. A link is available to return to the login page, and GitHub login is also supported.

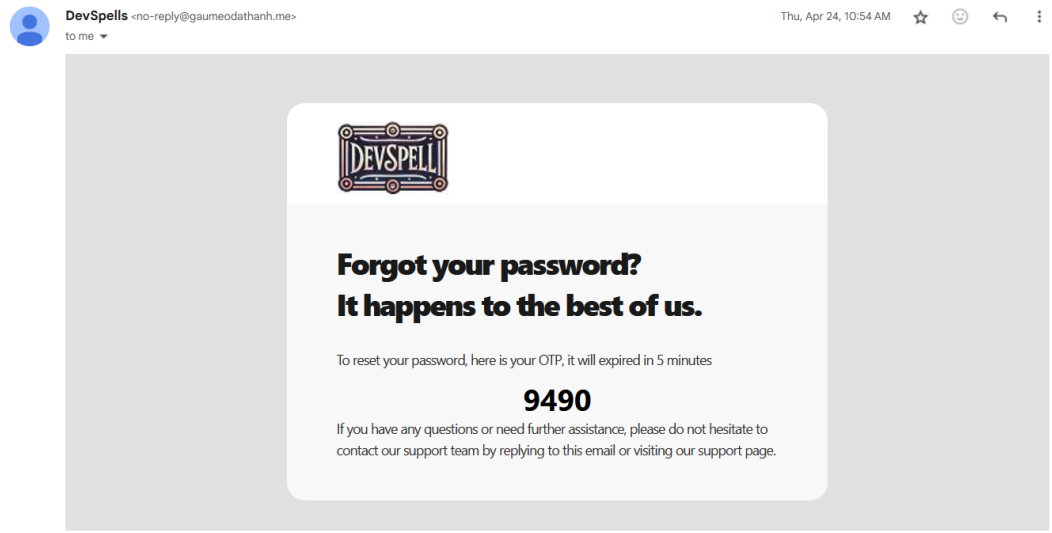

Figure 4.8 Forgot password confirmation mail

Password Reset OTP mail, this email is sent when a user requests to reset their password. It contains a 4-digit OTP that is valid for 5 minutes. The email includes a brief message of reassurance and contact support info in case assistance is needed.

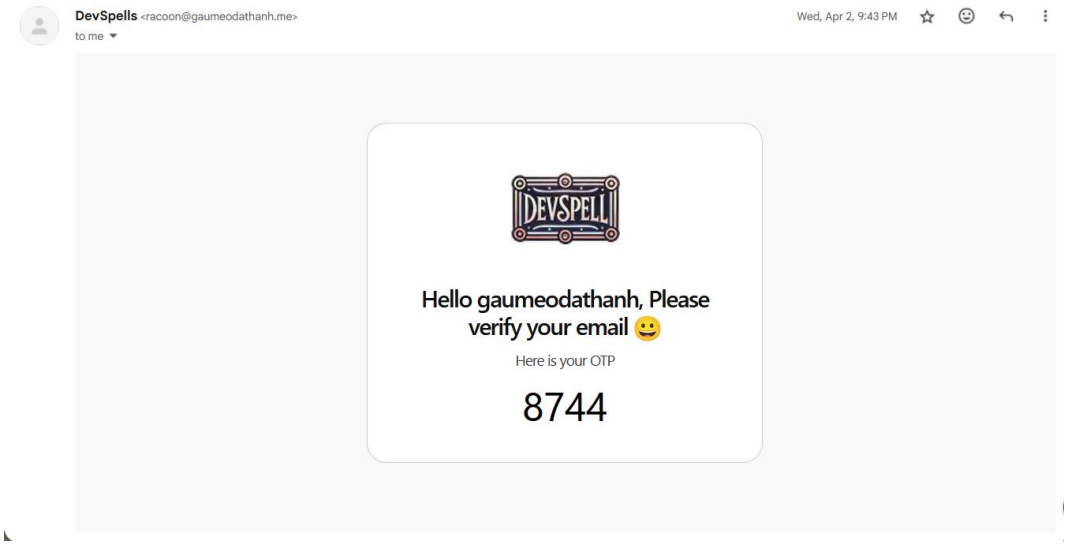

Figure 4.9 Verify account mail

Email Verification OTP mail, this email is sent to verify a user's email during account registration. It includes a friendly greeting and a 4-digit OTP for confirming the email address.

Figure 4.10 User homepage

User homepage allows users view their enrolled courses, track progress, and continue learning. It displays the career learning path, listing available courses with chapters and lessons. Each course card shows the completion status and a button to enter the course. Users can also access navigation links like Courses and Leaderboard, and see their current level and XP progress at the top right.

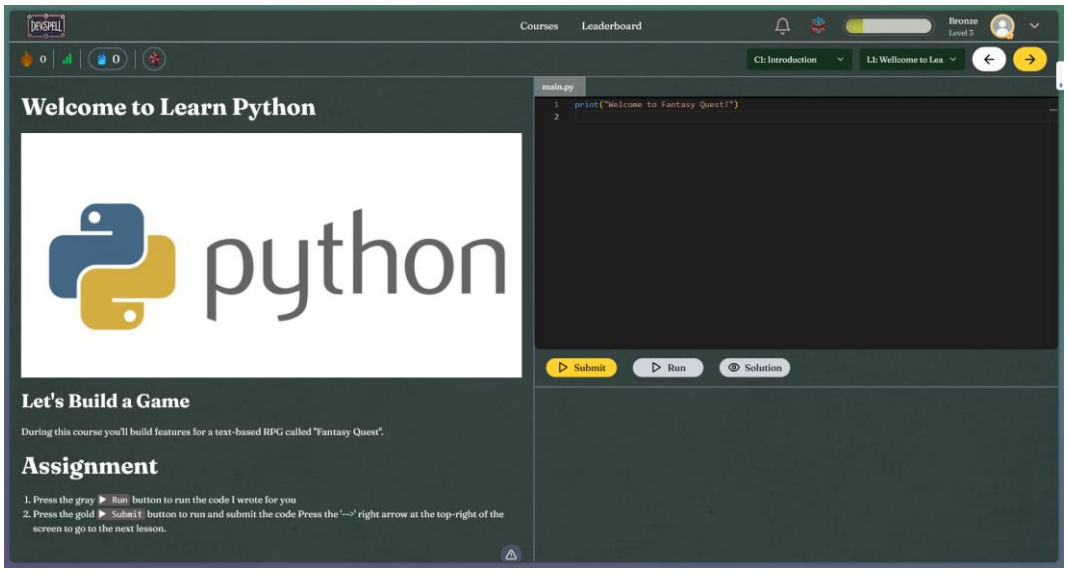

Figure 4.11 Learning page with code exercise

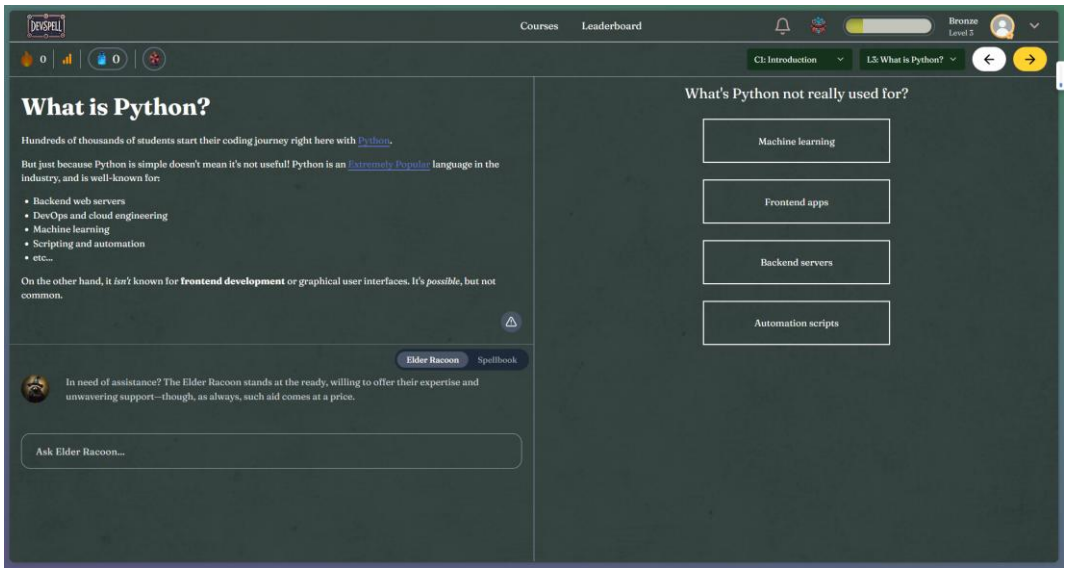

Figure 4.12 Learning page with multiple choices exercise

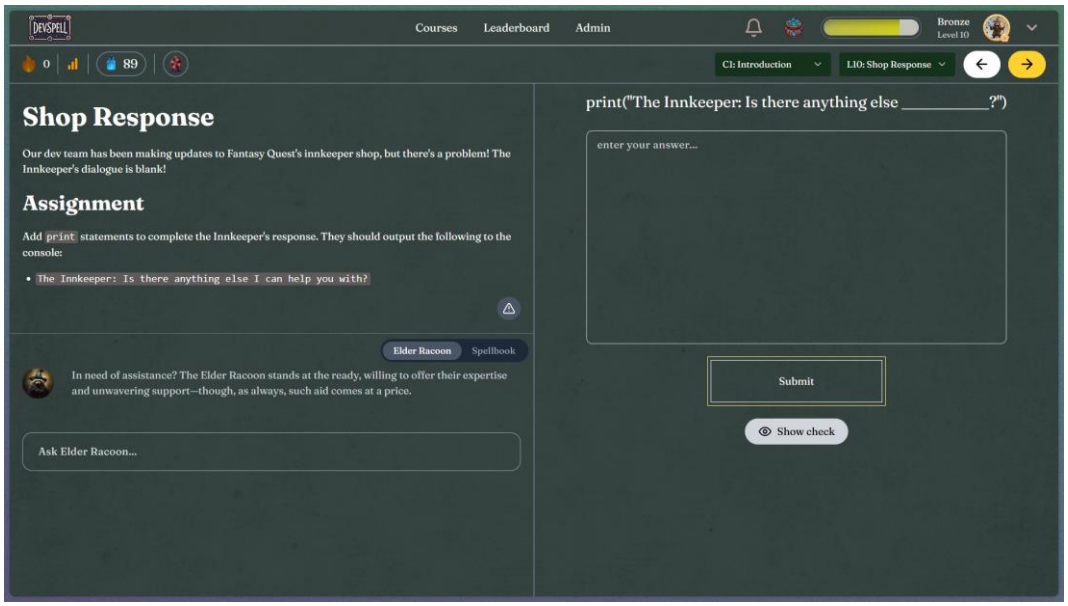

Figure 4.13 Learning page with question-and-answer exercise

This is the **Learning page** in DevSpells. The left side displays the lesson content, such as instructions, explanations, or media. The right side contains the interactive exercise area. Currently, the platform supports three types of exercises:

- Coding exercises – Users write and run code directly in the editor (shown in current screenshot)
- Question and answer – Users type written responses
- Multiple choice – Users select the correct answer from a list of options

Key Features:

- Coding Exercise Controls:
  - Submit button to check answers and progress
  - Run button to execute code and test functionality
  - Solution button to reveal hints or complete answers
- Navigation & Progress:
  - Navigation arrows to move between lessons (visible in top-right)
  - Structured lesson progression with clear assignment steps
- Gamification Elements:
  - XP potion item reward wheel for motivation
  - Lesson difficulty indicators displayed in left corner
  - User daily streak tracking shown in left corner
  - Level-based progression system (visible in header)

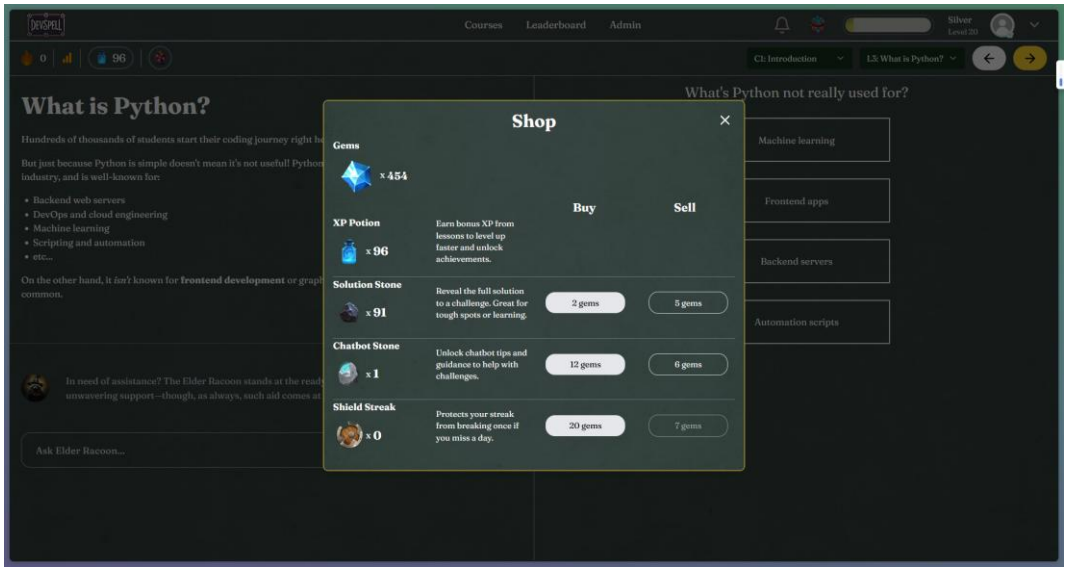

Figure 4.14 Shop

With **Shop,** Users can buy and sell items using in-game currency called gems. Available items include:

- XP Potion: Earn bonus XP to level up faster.
- Solution Stone: Unlock full solutions for the lesson.
- Chatbot Stone: Access chatbot hints and tips for guidance.
- Shield Streak: Prevents streak loss if a day is missed.

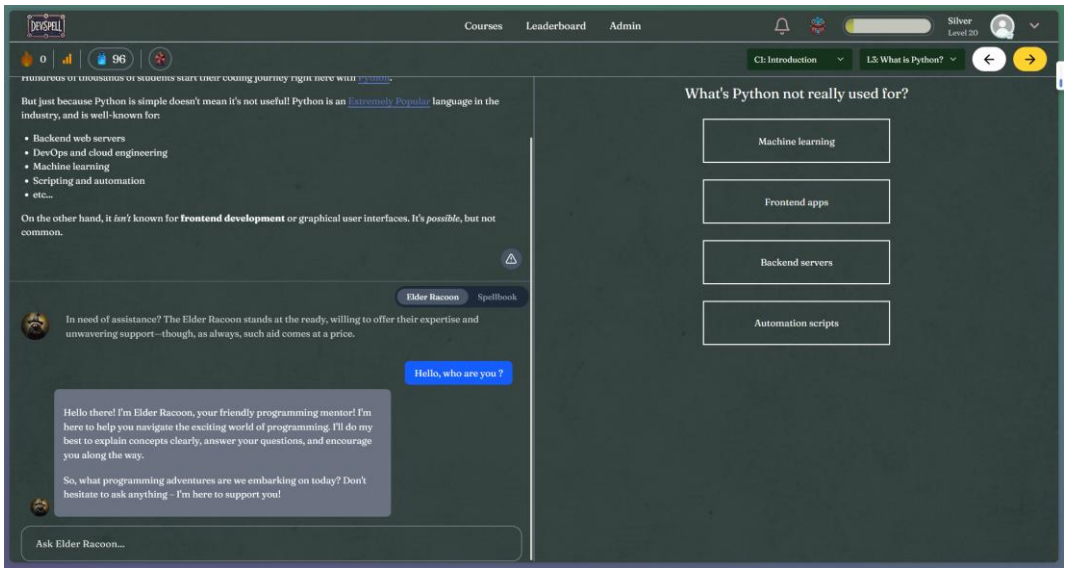

Figure 4.15 Chatbot

Chatbot integrated into the online learning platform that serves as a personalized tutor across all course subjects. The friendly assistant helps users navigate platform features, clarify complex concepts, and resolve misunderstandings about their current lessons. Chatbot provides instant answers to course-related questions, adapts explanations to each user's skill level, and guides learners through their educational journey with conversational support and encouragement.

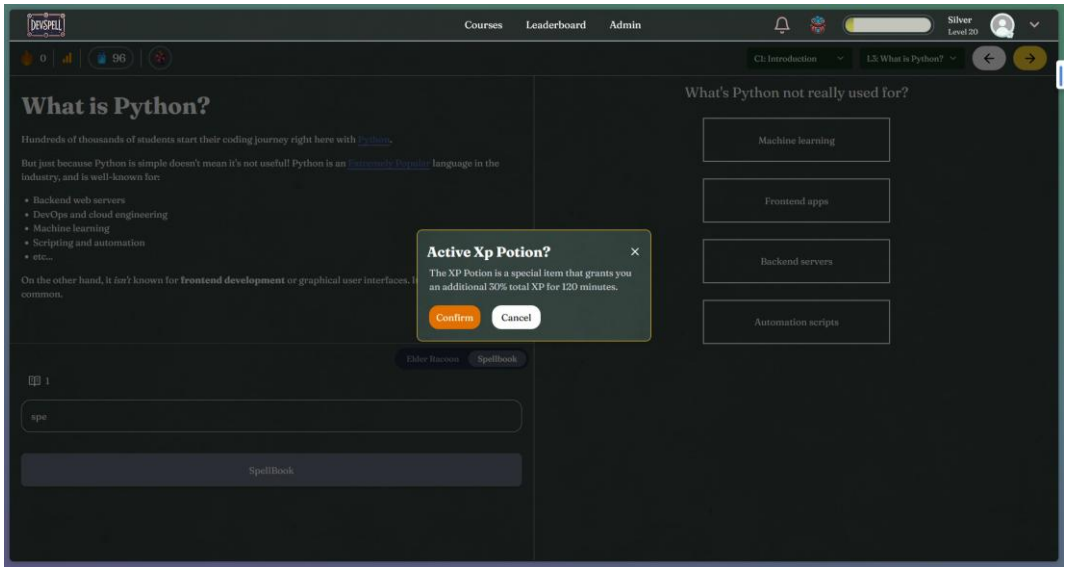

Figure 4.16 Use bonus XP item

The Active XP Potion is a timed boost that increases the experience points users earn from completing lessons and activities. When activated, it runs for amount of

time and multiplies the XP gained during that period. Users can choose to confirm activation to start the boost or cancel to save it for later. This feature tracks the remaining time and automatically expires after the duration ends, requiring users to activate a new potion to continue receiving bonus XP.

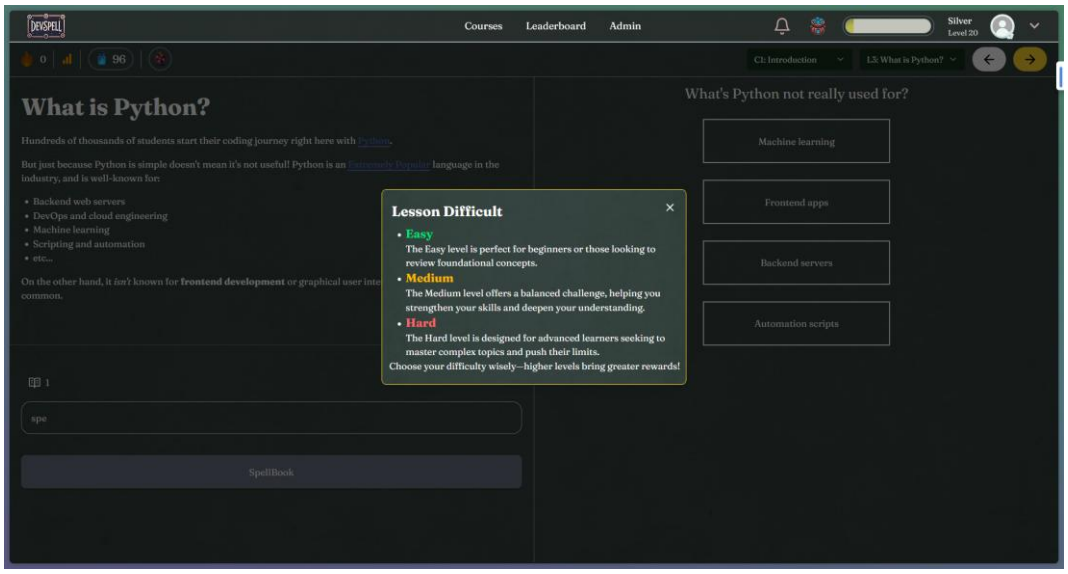

Figure 4.17 Lesson Difficult description

The lesson difficult level determines the reward amount users receive upon completion, with harder lessons providing greater XP and incentives. Users can see the difficulty rating before starting a lesson but must complete it at the assigned level set by the course administrators.

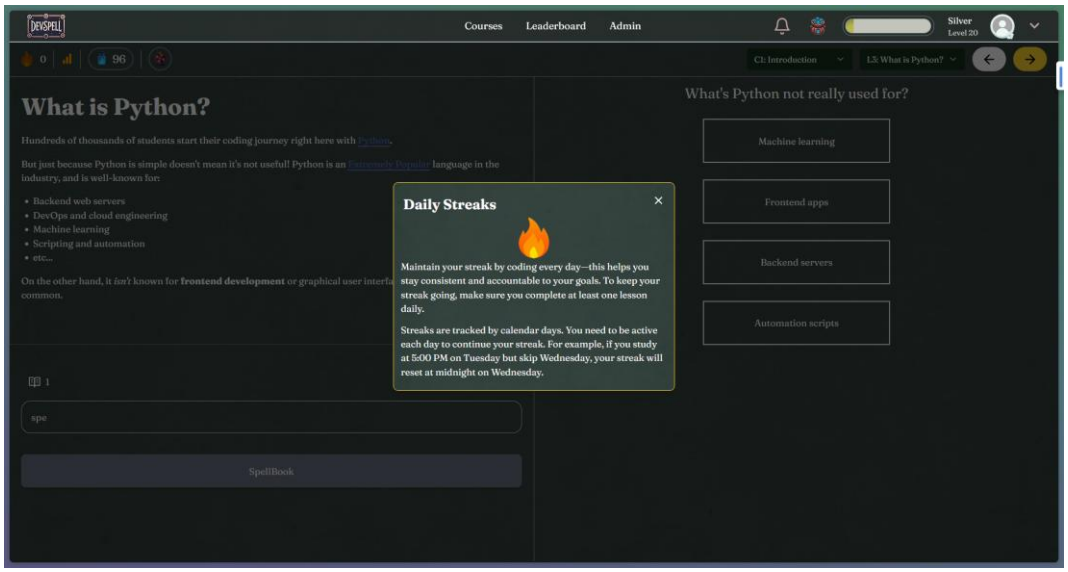

Figure 4.18 Daily streak description

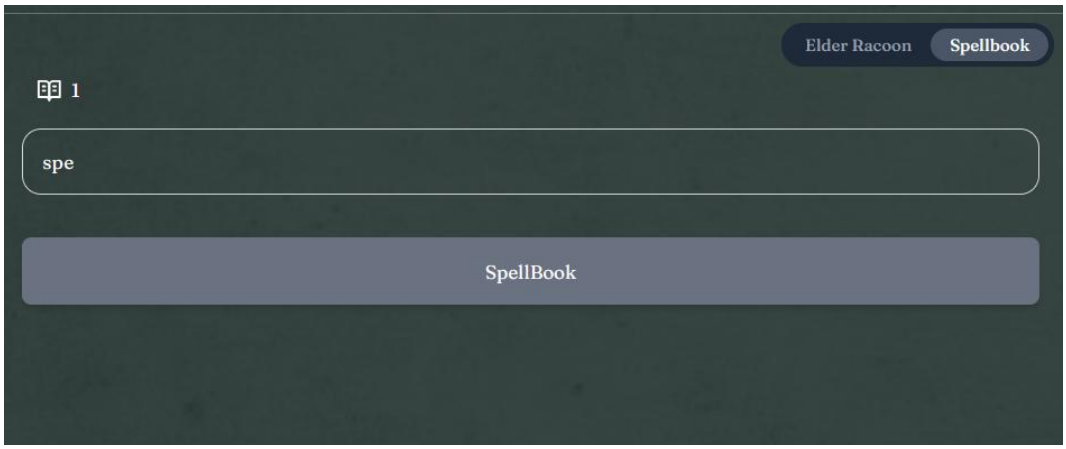

Figure 4.19 Search spellbook

Search spellbook is a search and reference tool that allows users to quickly look up terms, concepts, or keywords from their courses. Users can type in the search field to find relevant information, definitions, or explanations related to their current studies. This feature serves as a digital glossary or knowledge base, providing instant access to course-related content without having to navigate through multiple lessons.

Figure 4.20 Global leaderboard page

The Global Leaderboard ranks user's platform-wide based on their levels and daily activity. It displays two categories: daily lesson completions and all-time level. Users can see their ranking position, profile, username, and current level compared to other learners, creating motivation through friendly competition and progress tracking.

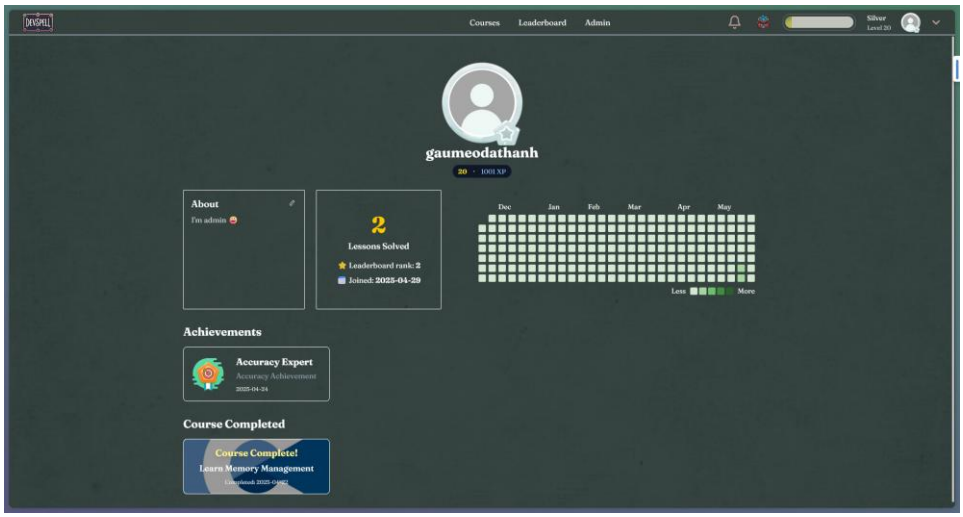

Figure 4.21 User profile page

User profile shows learning progress including level, lessons completed, and leaderboard rankings. It features an activity heatmap that displays daily engagement from both platform learning and GitHub contributions (when logged in with GitHub). Users can view their achievements and completed courses for a complete overview of their development activity.

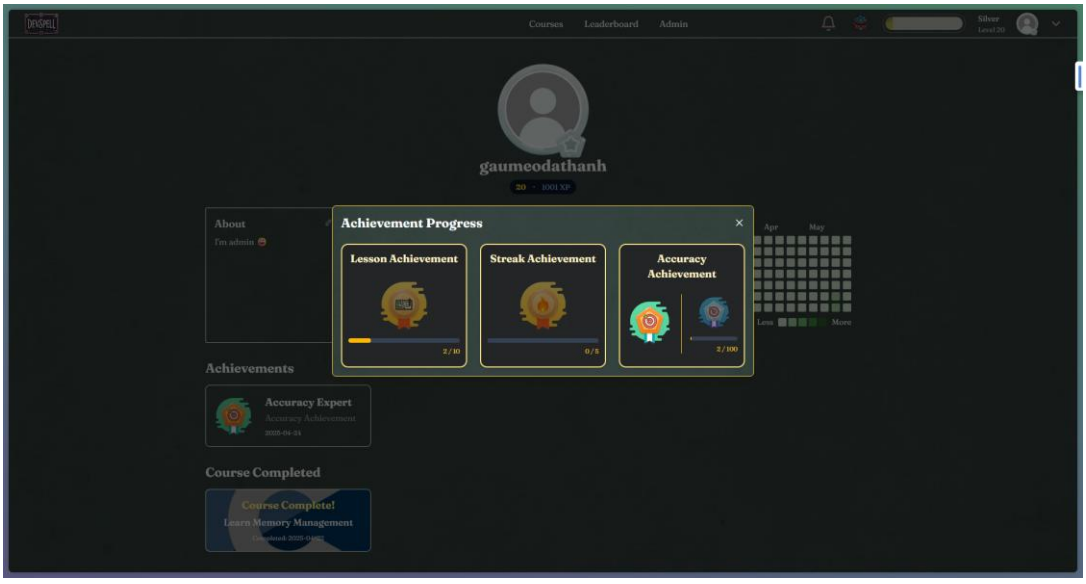

Figure 4.22 Achievement progress page

Achievement progress page displays progress tracking for three achievement types: Lesson (lesson completion), Streak (daily consistency), and Accuracy (Accuracy streak). Shows current progress with visual indicators and completion percentages for each goal.

Figure 4.23 Certificate page

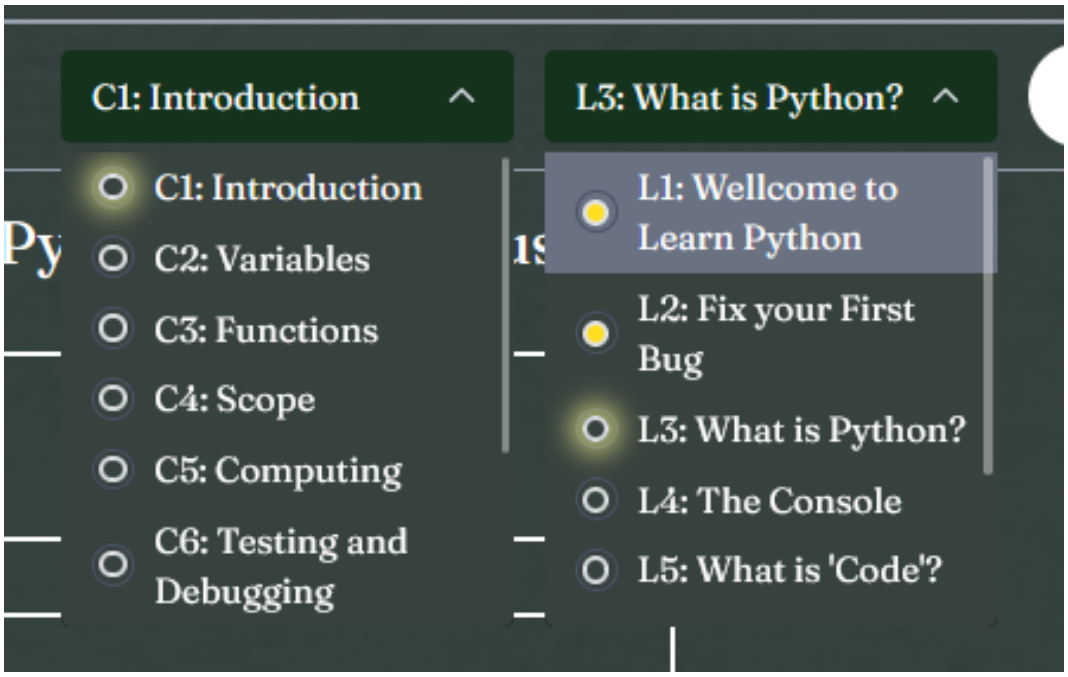

Figure 4.24 List chapter and lesson

The interface displays a hierarchical course structure with chapters and lessons. Chapters (C1, C2, etc.) contain multiple lessons (L1, L2, etc.) that can be expanded or collapsed for easy navigation. Completion status is indicated by colored circles - completed lessons show as filled/green circles while incomplete ones remain empty. Users can track their progress through the course sequence and access specific lessons within each chapter section.

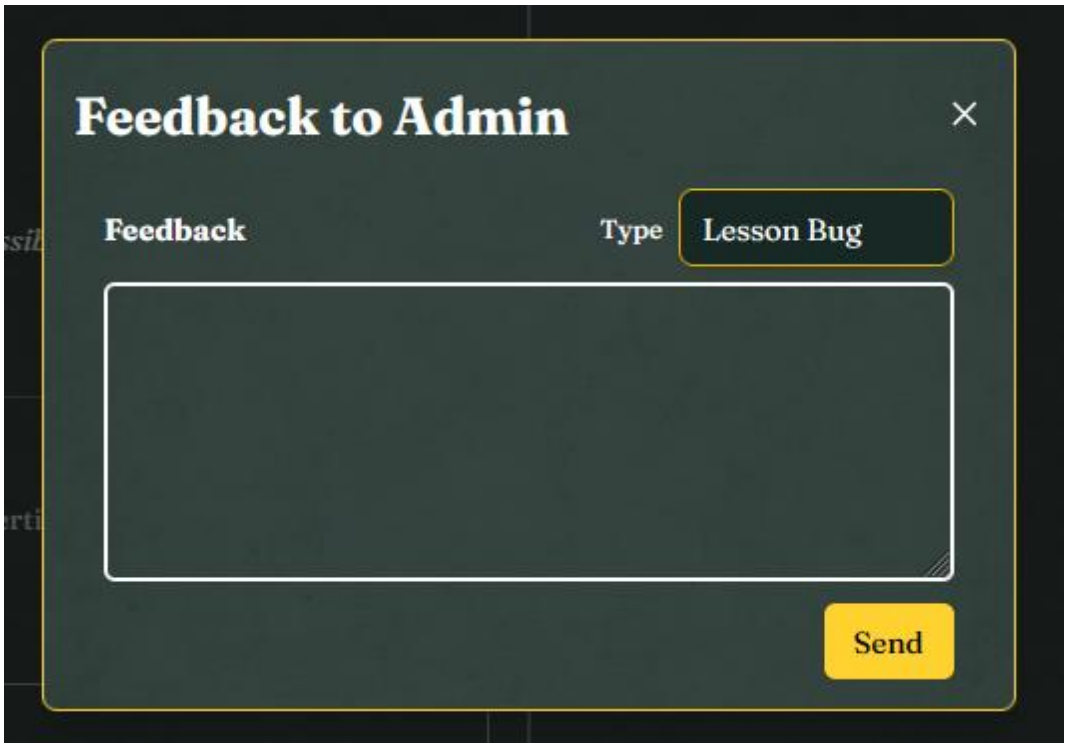

Figure 4.25 User send feedback page

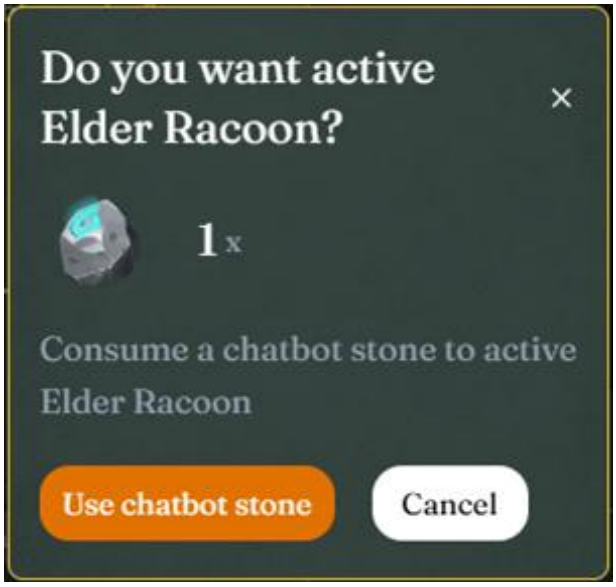

Figure 4.26 Use unlock chatbot item

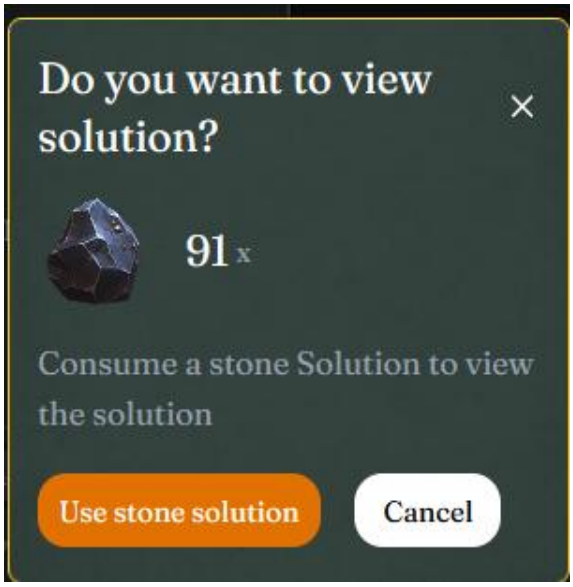

Figure 4.27 Use unlock solution item

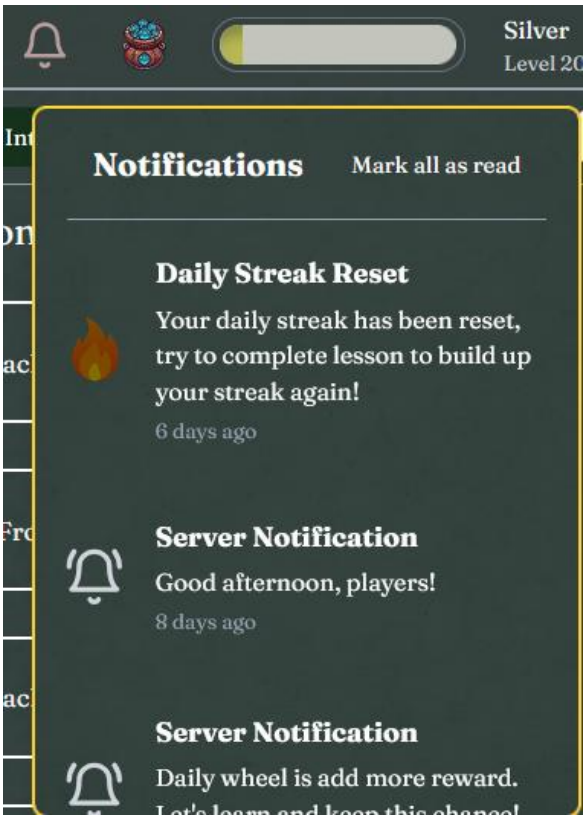

Figure 4.28 User notifications

The platform's **notification system** delivers various types of alerts to keep users engaged and informed:

- **Great Job! You're on a Streak!** - Celebrates when users maintain consistent daily learning activity
- **New Notification** - General alerts for platform updates or announcements
- **Daily Streak Protected** - Confirms when users activate streak protection to maintain their learning streak
- **Congratulations! Course Completed** - Rewards users for finishing entire courses
- **Lesson Achievement Unlocked** - Notifies when users earn specific lesson-based achievements
- **Daily Streak Unlocked** - Alerts when users reach new streak milestones
- **Server Notification** - System-wide announcements from administrators
- **Lucky Wheel Unlocked** - Informs users when they've earned access to reward wheels
- **Daily Streak Reset** - Warns users when their learning streak has been broken

4.3.3. Admin

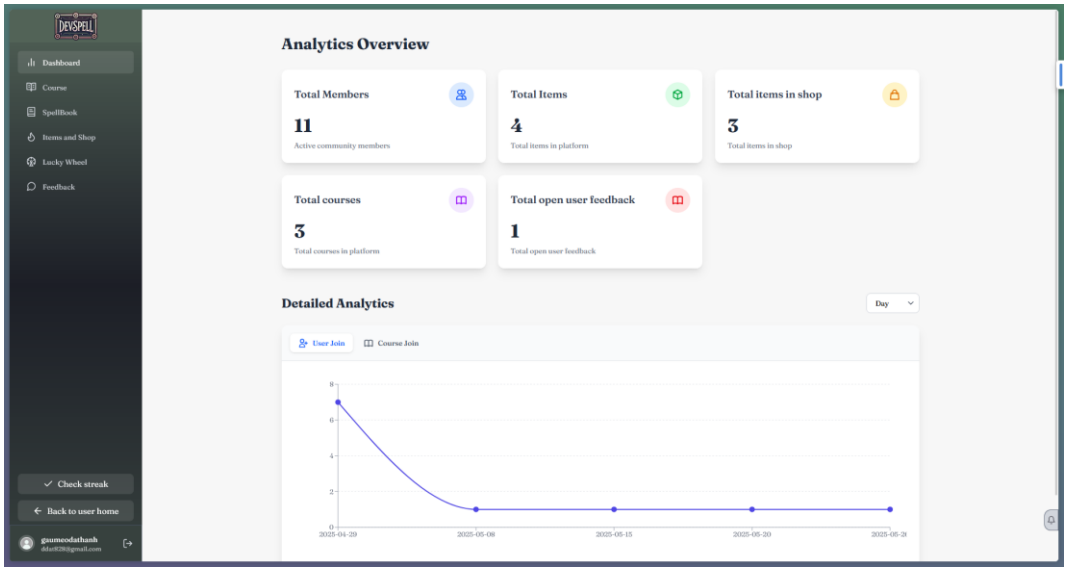

Figure 4.29 Admin dashboard page

The Dashboard provides administrators with key platform metrics and performance tracking:

- **Total Members** - Shows current user count
- **Total Items** - Displays available platform items
- **Total Items in Shop** - Lists purchasable items
- **Total Courses** - Tracks available courses
- **Total Open User Feedback** - Shows pending feedback
- **Detailed Analytics Charts** - Interactive graphs with User Join and Course Join data tracking
- **Time Period Filtering** - Allows admins to view data by day, week, or custom ranges

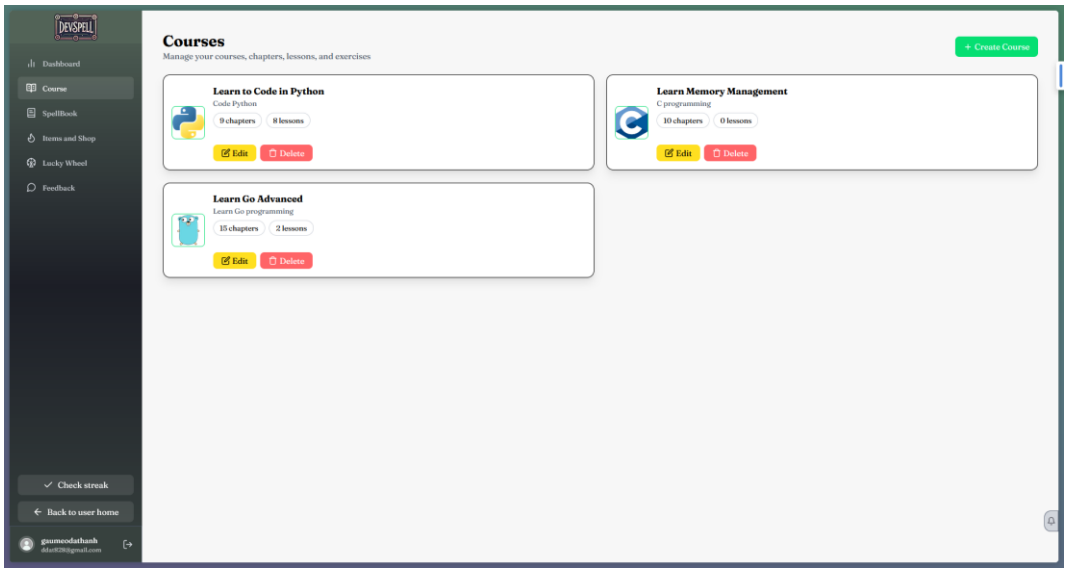

Figure 4.30 Admin course management page

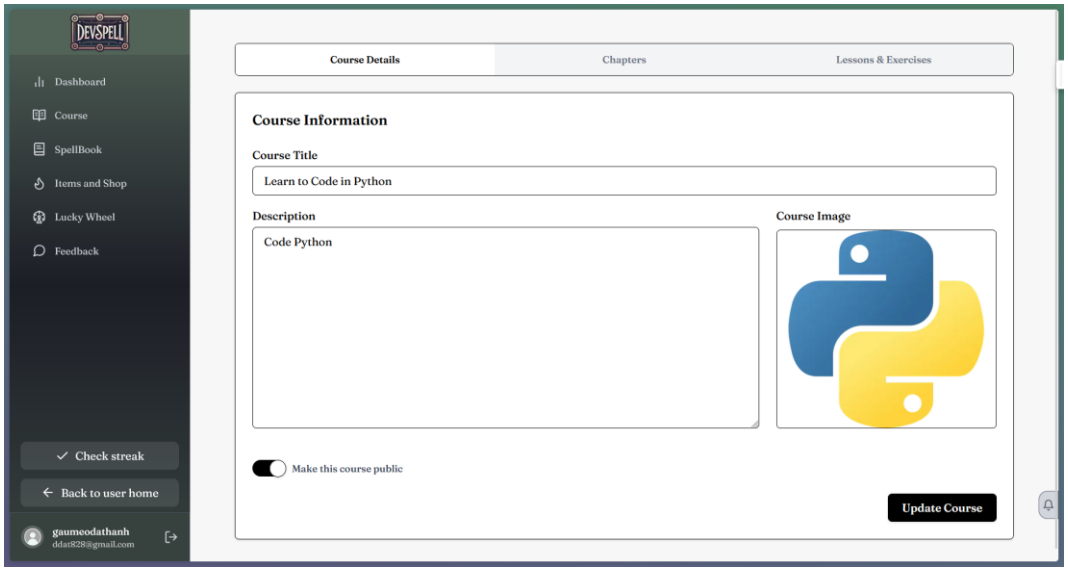

Figure 4.31 Admin edit course page

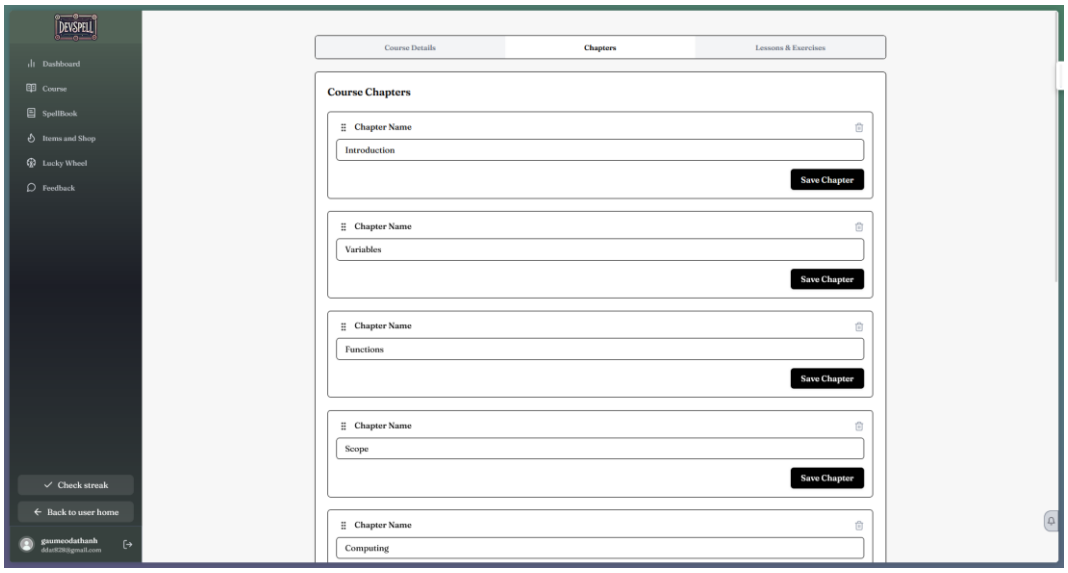

Figure 4.32 Admin edit chapter page

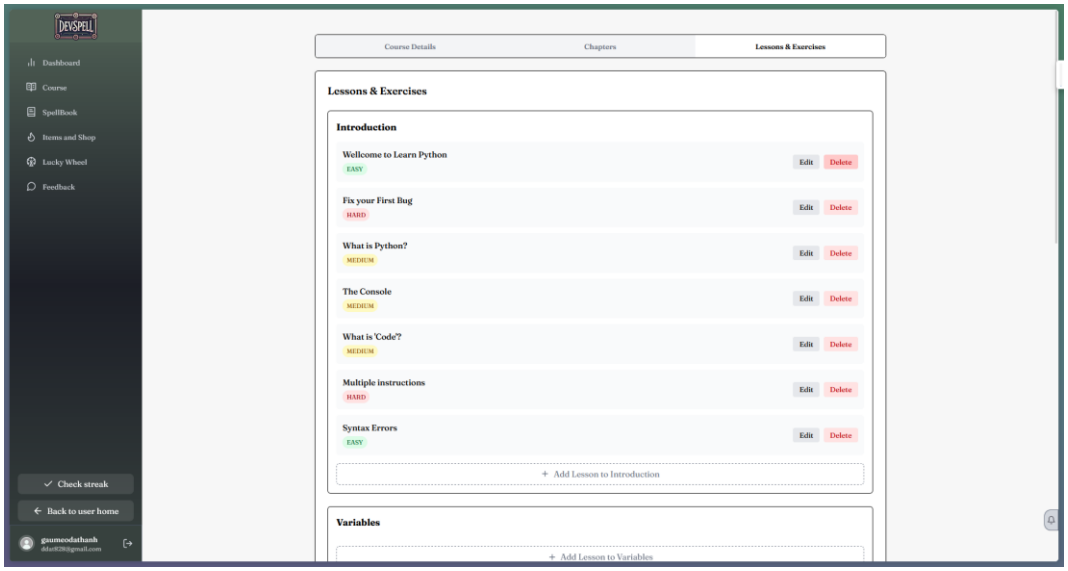

Figure 4.33 Admin edit lesson list page

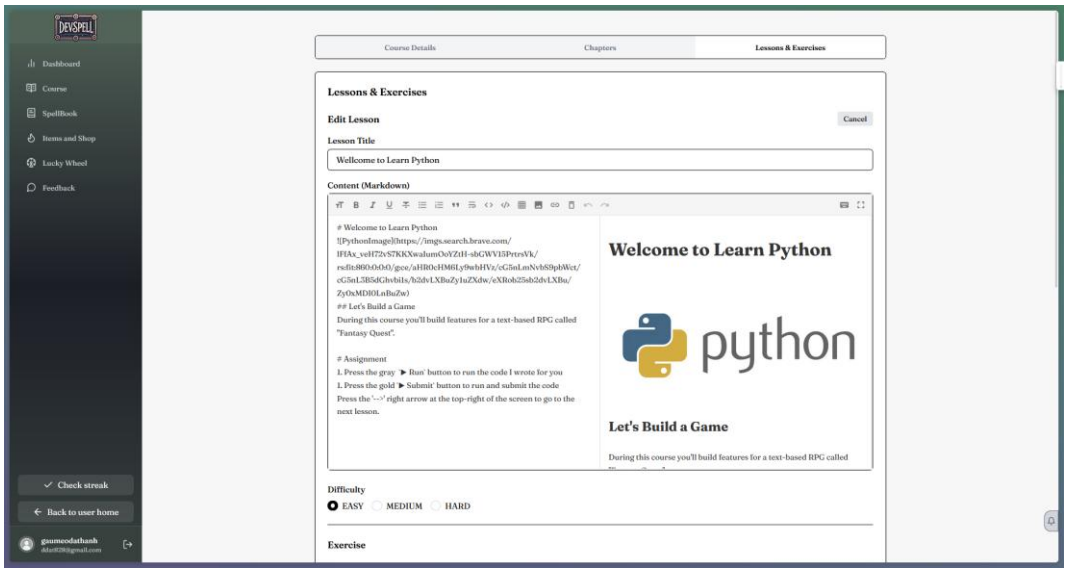

Figure 4.34 Admin edit lesson detail page

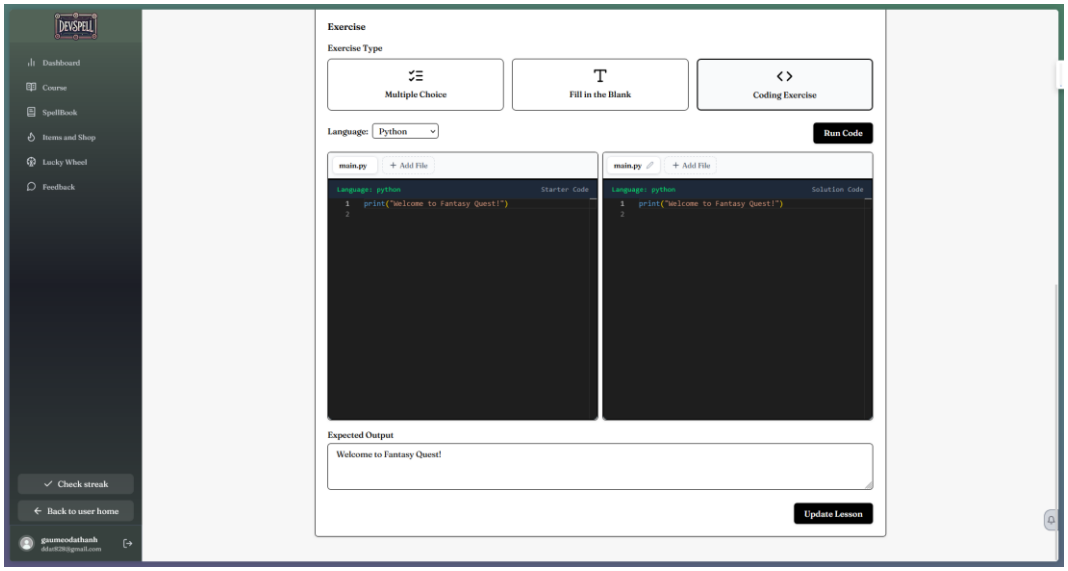

Figure 4.35 Admin edit coding exercise

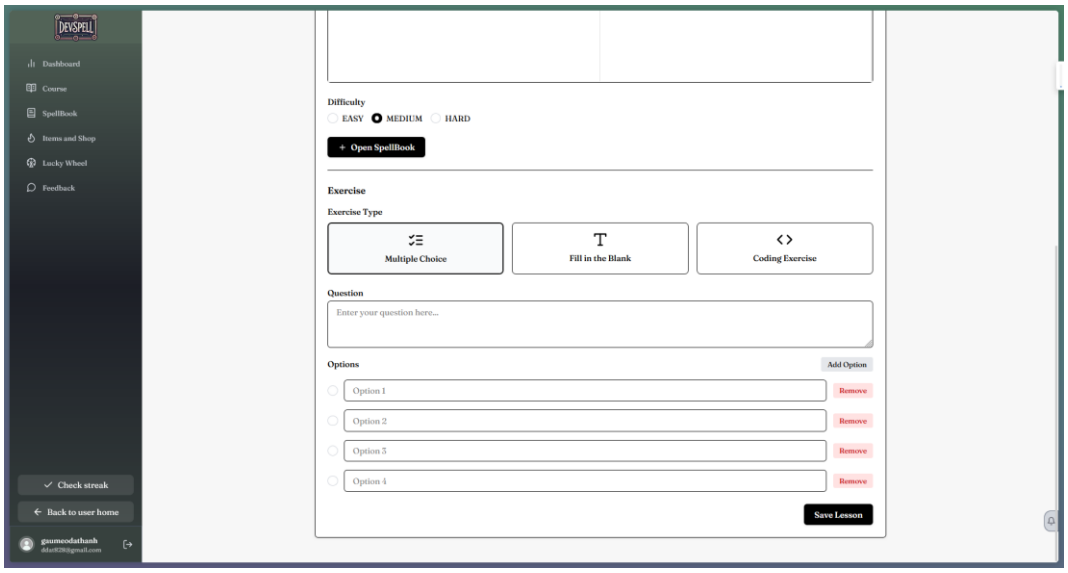

Figure 4.36 Admin edit multiple choice exercise

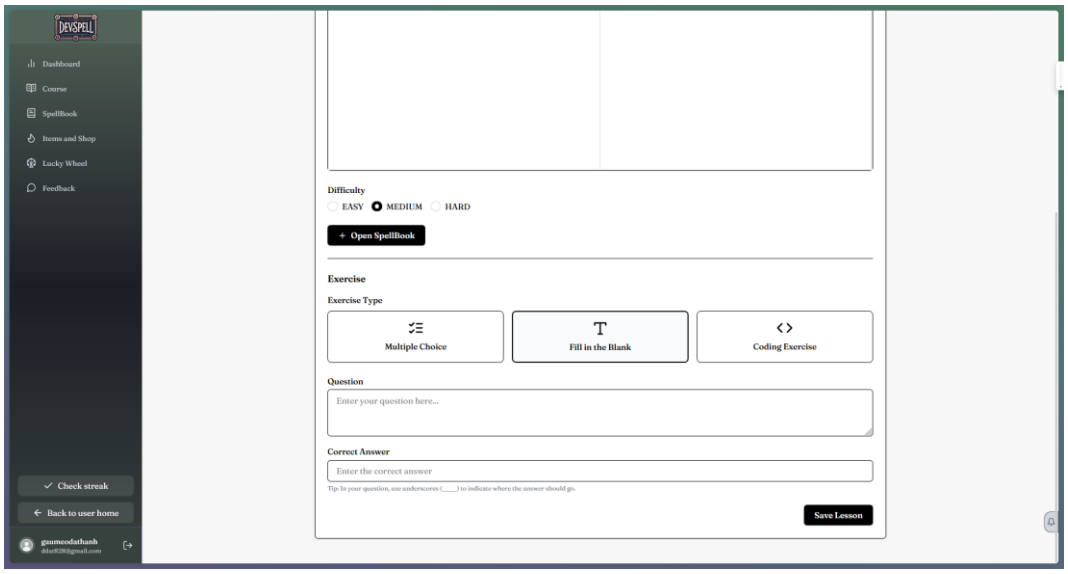

Figure 4.37 Admin edit fill in the blank exercise

The course management provides administrators with comprehensive tools to create and manage educational content:

- **Course Overview** - Display all available courses with chapter/lesson counts, edit and delete options
- **Course Details Tab** - Configure course title, description, cover image, and visibility settings
- **Chapters Tab** - Create and organize course chapters with editable names and save functionality

- **Lessons & Exercises Tab** - Manage individual lessons within chapters, including:
    - **Lesson Configuration** - Edit lesson titles and assign difficulty levels (Easy, Medium, Hard)
    - **Lesson Content Management** - Create and modify theoretical content, explanations, and instructional material
    - **Exercise Content Editor** - Design interactive coding exercises, quizzes, and practical assignments
    - **Difficulty Adjustment** - Set appropriate challenge levels that determine XP rewards
    - **Content Structure** - Organize lessons within chapter sections (Introduction, Variables, Functions, etc.)
    - **Lesson Operations** - Edit, delete, and add new lessons with full content control
    - **Exercise Validation** – Execute code for validate coding exercise

The system enables administrators to create complete learning experiences by combining theoretical lessons with hands-on exercises, ensuring proper difficulty progression and comprehensive skill development.

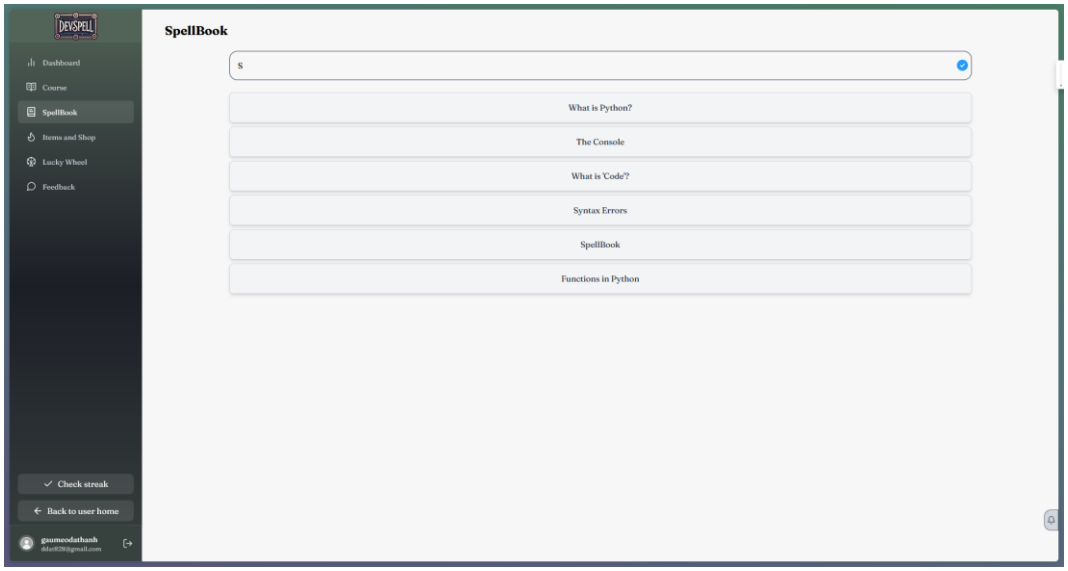

Figure 4.38 Admin search spellbook page

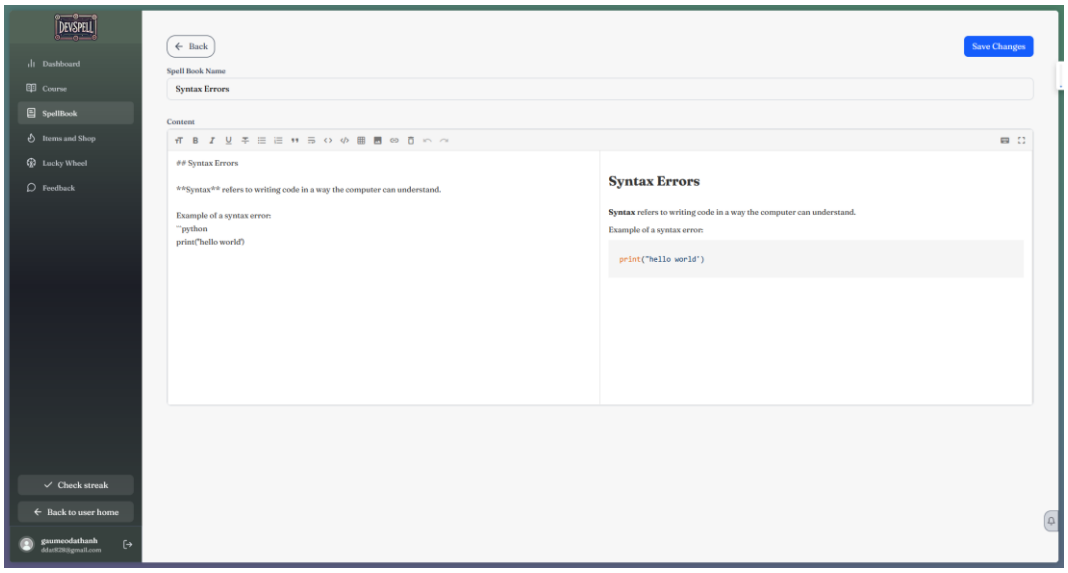

Figure 4.39 Admin edit spellbook page

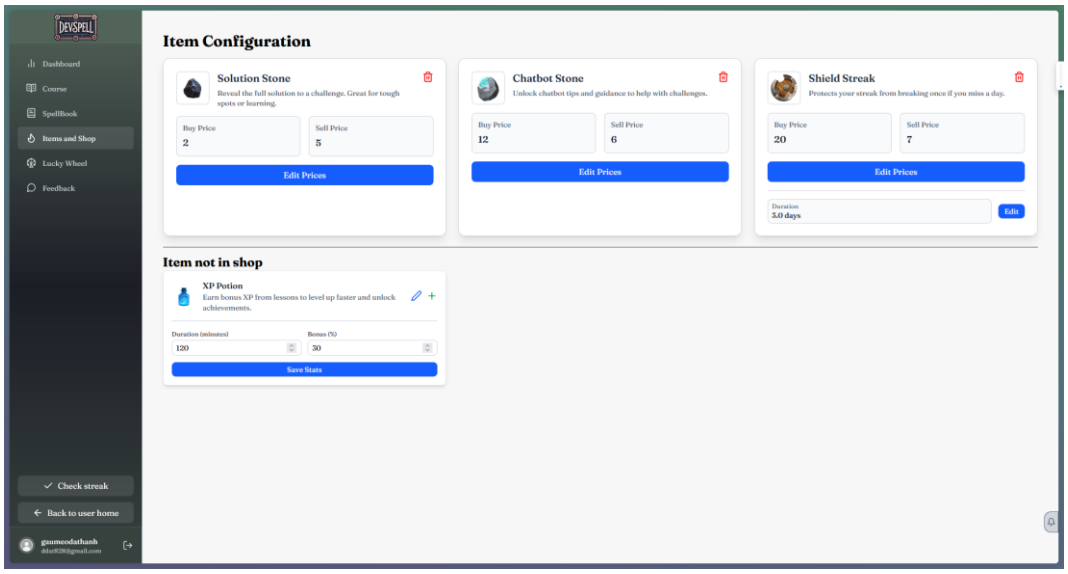

Figure 4.40 Admin item and shop management page

The **Item management** system allows administrators to control the in-game shop and reward items:

- **Shop Items Management** - Configure items available for purchase including:
  - **Solution Stone** - Helps users get solutions to challenges
  - **Chatbot Stone** - Unlocks advanced chatbot guidance features
  - **Shield Streak** - Protects user learning streaks from breaking
  - **Pricing Control** - Set buy and sell prices for each item
  - **Item Editing** - Modify item descriptions, effects, and availability

- **Item Operations** - Full administrative control over shop inventory:
  - **Add New Items** - Create custom items with unique properties
  - **Remove Items** - Delete items from shop completely
  - **Edit Item Information** - Update names, descriptions, and functional effects
  - **Stats Adjustment** - Modify item effectiveness, duration, and benefits
  - **Price Management** - Control both purchase and sell-back values

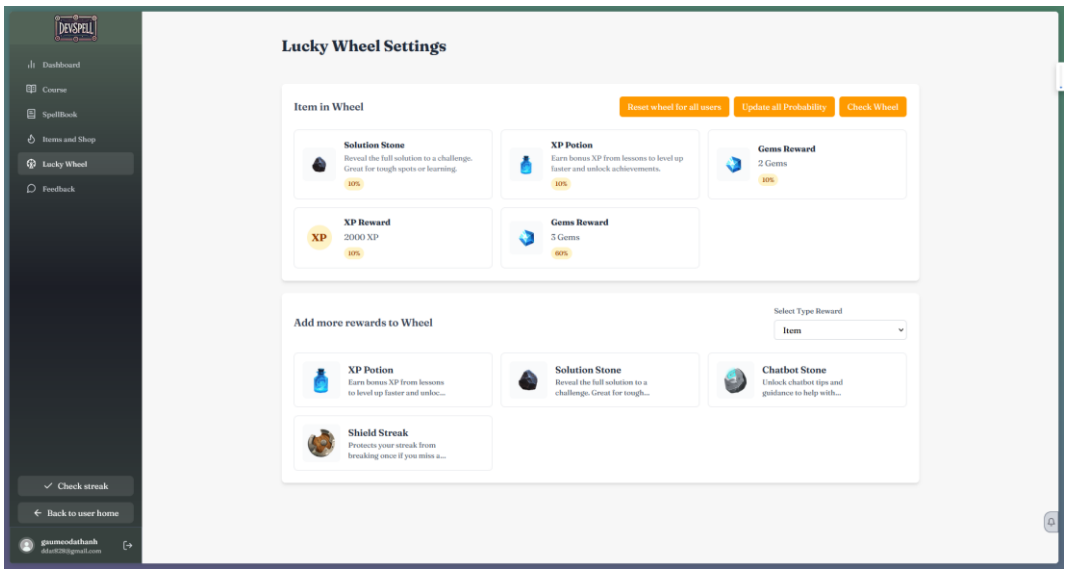

Figure 4.41 Admin edit lucky wheel page

The **Lucky Wheel management** system allows administrators to configure the reward wheel mechanics with three reward types:

- **Reward Type Management** - Configure three distinct reward categories:
  - **Item Rewards** - Physical items like potions, stones, and streak protectors
  - **XP Rewards** - Experience point bonuses for leveling progression
  - **Gem Rewards** - Premium currency for shop purchases
- **Wheel Configuration Control** - Manage reward distribution:
  - **Edit Probabilities** - Adjust percentage chances for each reward type
  - **Add More Rewards** - Select from available items, set XP amounts, or configure gem quantities
  - **Reward Balancing** - Control the mix of items, experience, and premium currency
  - **Remove Rewards** - Delete specific rewards from wheel rotation

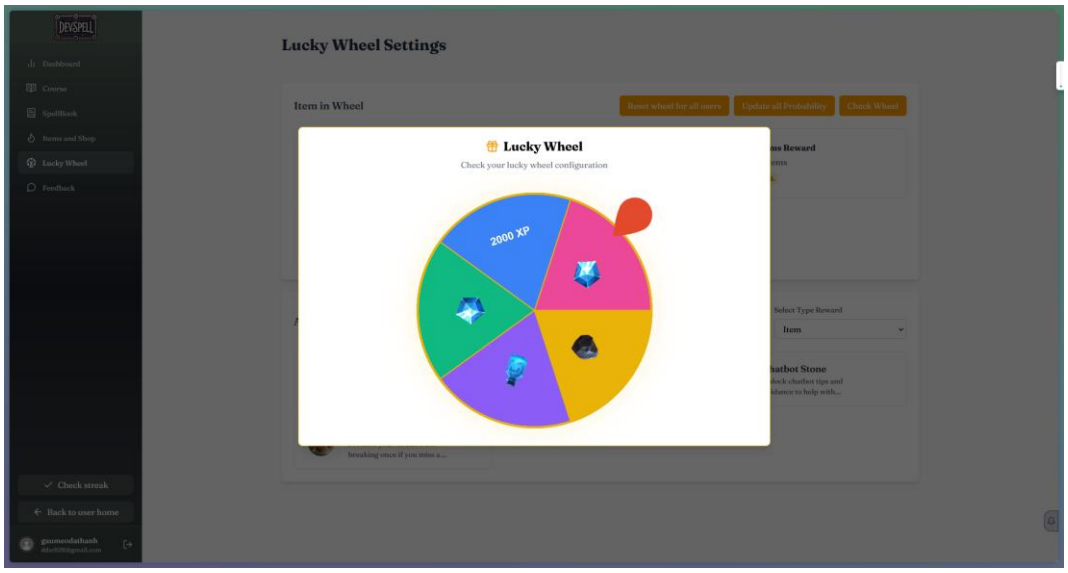

Figure 4.42 Lucky wheel

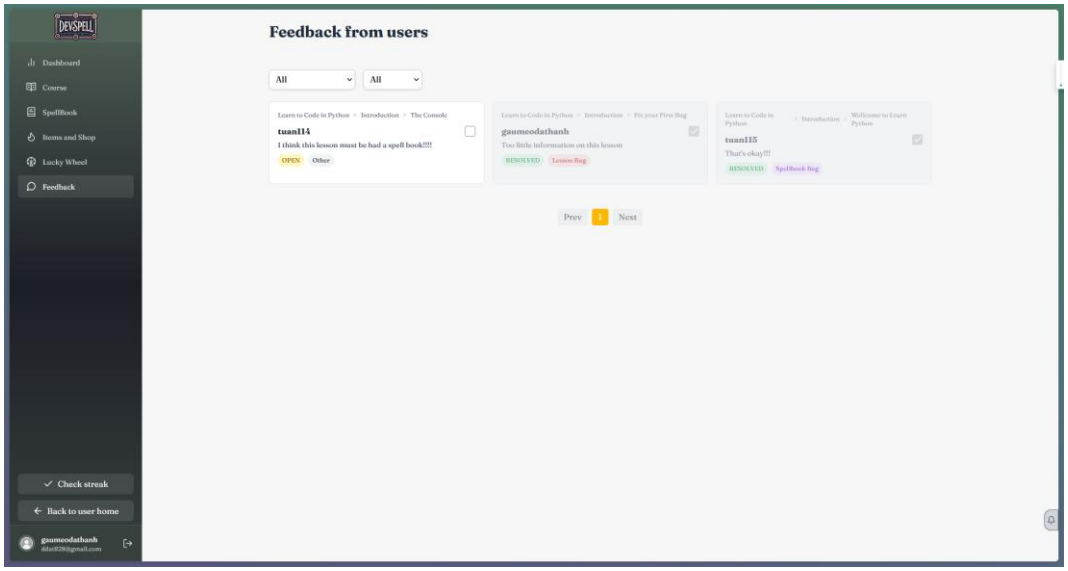

Figure 4.43 Admin feedback management page

CONCLUSION

1. Result

The completion of the DEVSPELL project has successfully achieved the primary objectives set out at the beginning of this thesis. The platform offers an engaging, interactive programming learning environment by combining traditional course structures with gamification elements such as levels, items, daily rewards, achievements, and leaderboards.

Through the development of the DEVSPELL project, I gained valuable knowledge and hands-on experience in multiple domains. On the AI front, I deepened my understanding of advanced concepts such as **function calling** and **vector similarity**, which are crucial for building intelligent and context-aware chatbot functionalities. This enhanced my ability to integrate AI-powered features that deliver personalized and dynamic support to users.

From the application development perspective, I experienced the complete software development lifecycle—from designing and implementing backend and frontend components, to **containerizing** the application with Docker, and setting up **continuous integration and deployment pipelines** using GitHub Actions. This end-to-end process improved my skills in building scalable, maintainable, and deployable web applications in a professional setting. Additionally, working with modern technologies like NestJS, ReactJS, Redis, and Supabase gave me practical insights into building robust systems that effectively combine performance, user experience, and operational reliability.

Overall, the project demonstrates that combining educational content with RPG-style engagement and AI assistance can significantly enhance learner motivation and improve the effectiveness of self-paced programming education.

2. Future work

Although DEVSPELL has met its initial goals, several areas remain for further development and enhancement:

- **Mobile Application**: Developing a mobile version of the platform would make learning more accessible and convenient for users on the go.
- **Multiplayer and Collaborative Features**: Adding team-based missions, code battles, or peer-to-peer learning could increase engagement and foster a learning community.
- **Personalized Learning Paths**: Implementing adaptive learning algorithms to tailor lesson recommendations based on user performance and interest.
- **Integration with External Platforms**: Connecting with GitHub, coding challenge platforms, or certification services would add practical value to the user's learning progress.
- **Admin Analytics and Reporting Tools**: More detailed data visualization for user performance, drop-off rates, and content effectiveness would support better system management and decision-making.
- **Add new types of exercises:** Introduce new exercise formats such as assessment through the learner's terminal. This will help users get familiar with real project environments and avoid being constrained by the platform interface.

These improvements can help evolve DEVSPELL from a standalone learning tool into a comprehensive, intelligent, and community-driven platform for future programmers.

REFERENCES

[1] D. Inc, "dockerdocs," [Online]. Available: https://docs.docker.com/.

[2] M. D. Hardt, "The OAuth 2.0 Authorization Framework," 13 10 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6749.html.

[3] Michael B. Jones, John Bradley, Nat Sakimura, "JSON Web Token (JWT)," 5 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7519.html.

[4] I. Hickson, "Server-sent events," [Online]. Available: https://en.wikipedia.org/wiki/Server-sent_events.

[5] R. Schwaber-Cohen, "Vector Similarity Explained," 30 6 2023. [Online]. Available: https://www.pinecone.io/learn/vector-similarity/.

[6] Google, " Embeddings," [Online]. Available: https://ai.google.dev/gemini-api/docs/embeddings.

[7] Google, "text-embedding-004," [Online]. Available: https://ai.google.dev/gemini-api/docs/models/gemini#text-embedding.

[8] Google, "Function calling with the Gemini API," Function calling, [Online]. Available: https://ai.google.dev/gemini-api/docs/function-calling.

[9] Tailwind, "Tailwind documents," [Online]. Available: https://tailwindcss.com/docs/installation/using-vite.

[10] React, "React documents," [Online]. Available: https://react.dev/reference/react.

[11] Github, "Github Action documents," [Online]. Available: https://docs.github.com/en/actions.

[12] R. Cloud, "Redis Cloude documents," [Online]. Available: https://redis.io/docs/latest/operate/rc/.

[13] Supabase, "Supabase documents," [Online]. Available: https://supabase.com/docs.

[14] Amazon, "S3 storage," [Online]. Available: https://aws.amazon.com/s3/.

[15] engineer-man, "piston api," [Online]. Available: https://piston.readthedocs.io/en/latest/api-v2/.

[16] "Brevo," [Online]. Available: https://www.brevo.com/.

[17] NestJS, "NestJS documents," [Online]. Available: https://docs.nestjs.com/.