

**CEBU INSTITUTE OF TECHNOLOGY  
UNIVERSITY**

**COLLEGE OF COMPUTER STUDIES**

**Software Design Description**

*for*

*PronounceIT:: English Vocabulary/Pronunciation E-Learning System for  
Kindergarten*

**Signature**

<b>Date</b>	<b>Version</b>	<b>Description</b>	<b>Authors</b>
3/20/2025	1.00	Initial Draft	Kho, Antonio A. Alico, Christian Barry R. Laude, Raymund Christian A. Saceda, Rhandulf Q. Pobadora, Rommel John L.



## Preface

---



# Table of Contents

<b>1.</b>	<b>INTRODUCTION .....</b>	<b>6</b>
1.1.	PURPOSE .....	6
1.2.	SCOPE .....	6
1.3.	DEFINITIONS AND ACRONYMS .....	6
1.4.	REFERENCES .....	7
<b>2.</b>	<b>ARCHITECTURAL DESIGN .....</b>	<b>8</b>
<b>3.</b>	<b>DETAILED DESIGN .....</b>	<b>9</b>
	<i>Module 1 .....</i>	<i>9</i>
	<i>Module 2 .....</i>	<i>14</i>

# 1. Introduction

## 1.1. Purpose

The purpose of this Software Design Description (SDD) is to provide a comprehensive architectural and design specification for the PronounceIT software, an AI-driven English Vocabulary and Pronunciation E-Learning System tailored for kindergarten students. This document translates the requirements outlined in the Software Requirements Specification (SRS) into detailed technical specifications, design patterns, and implementation guidelines. It will serve as the primary reference for developers, software architects, and technical stakeholders to ensure the successful development of a system that fulfills all specified requirements.

This SDD presents the technical approach and design decisions that will be implemented to deliver PronounceIT functionality. It includes detailed descriptions of system components, data structures, interfaces, and algorithms necessary to meet the requirements defined in the SRS, particularly focusing on the AI-driven voice recognition technology that provides immediate feedback on pronunciation attempts.

## 1.2. Scope

This Software Design Description covers the detailed design of PronounceIT, a web and mobile application designed to assist kindergarten students in improving their pronunciation skills and expanding their English vocabulary. The document describes the architectural design, component specifications, and implementation details for all system features including:

- AI voice recognition subsystem for evaluating pronunciation accuracy
- Audio playback system for correct pronunciation examples
- Visual vocabulary presentation with supporting imagery
- Gamification elements to enhance learner engagement
- User profile and progress tracking functionality
- Administrative interfaces for content management

The design addresses both the user-facing components and the backend systems necessary to support the application's functionality across web and mobile platforms. This document provides implementation guidance while allowing for the flexibility needed during the development process.

The design does not include:

- Implementation details for languages other than English
- Designs for advanced grammar or sentence structure training features
- Integration designs with specific classroom management systems

## 1.3. Definitions and Acronyms

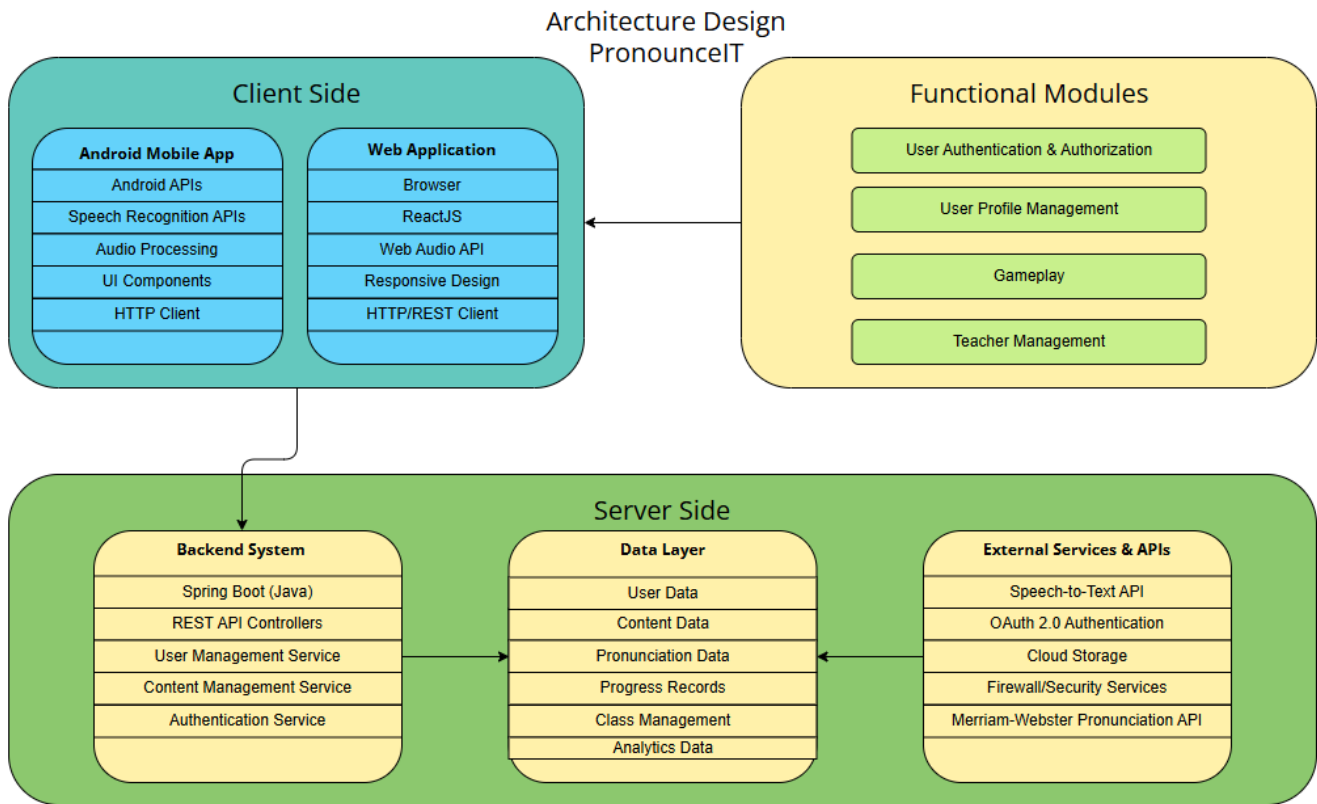
PronounceIT	The name of the web and mobile application.
Admin	An administrator with the ability to manage users, content, category and lessons.
SRS	Software Requirements Specification
SDD	Software Design Description
AI	Artificial Intelligence, used for voice recognition and pronunciation feedback.

API	Application Programming Interface
UI	User Interface
UX	User Experience
REST	Representational State Transfer
JSON	JavaScript Object Notation
COPPA	Children's Online Privacy Protection Act, a data protection regulation ensuring child safety online.

## 1.4. References

Component	Name	Description
Problem Statement	<a href="#">PronounceIT Problem Statement</a>	This document defines the core issue that PronounceIT aims to solve, outlining the challenges related to pronunciation learning and how the software addresses them.
Software Project Proposal	<a href="#">PronounceIT Software Project Proposal</a>	This proposal provides an overview of the PronounceIT project, detailing its objectives, features, target audience, and the technologies used for development.
Software Requirements Specifications	<a href="#">PronounceIT Software Requirements Specifications</a>	This document details the functional and non-functional requirements that form the basis for this design document.

## 2. Architectural Design



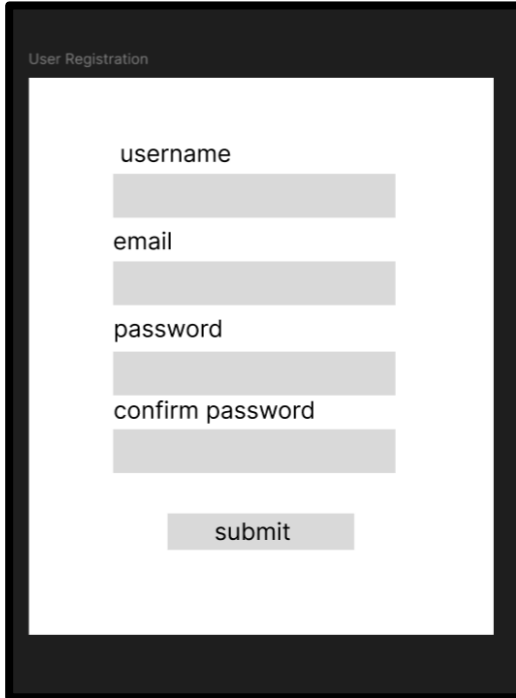


## 3. Detailed Design

### Module 1: User Authentication and Authorization

#### 1.1 User Sign-in

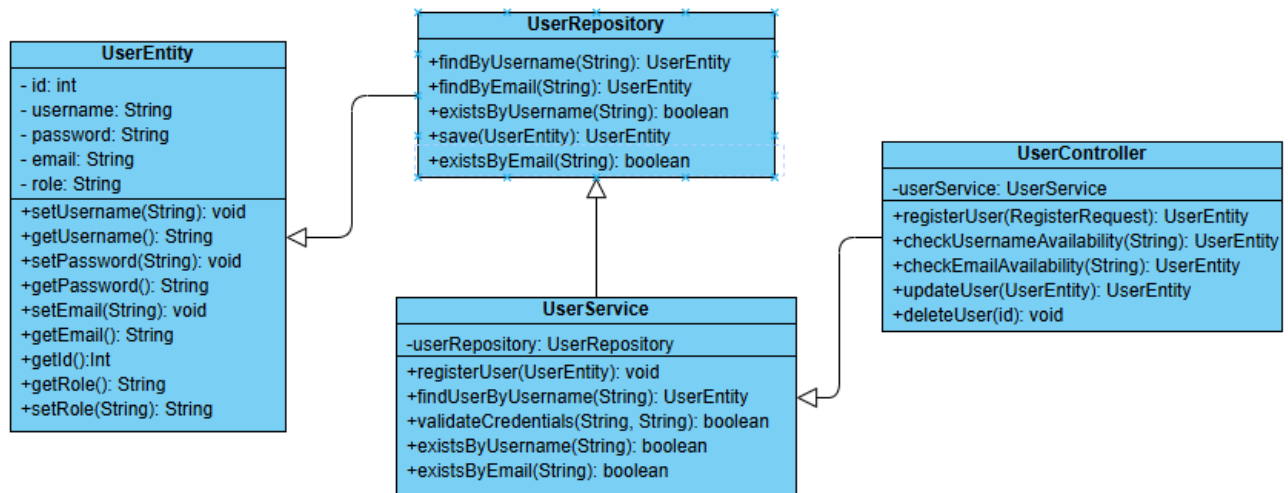
- User Interface Design

A UI mockup of a 'User Registration' form. The form is titled 'User Registration' in a small font at the top left. It contains four text input fields stacked vertically, labeled 'username', 'email', 'password', and 'confirm password'. Below these fields is a single 'submit' button. The entire form is enclosed in a dark gray border.

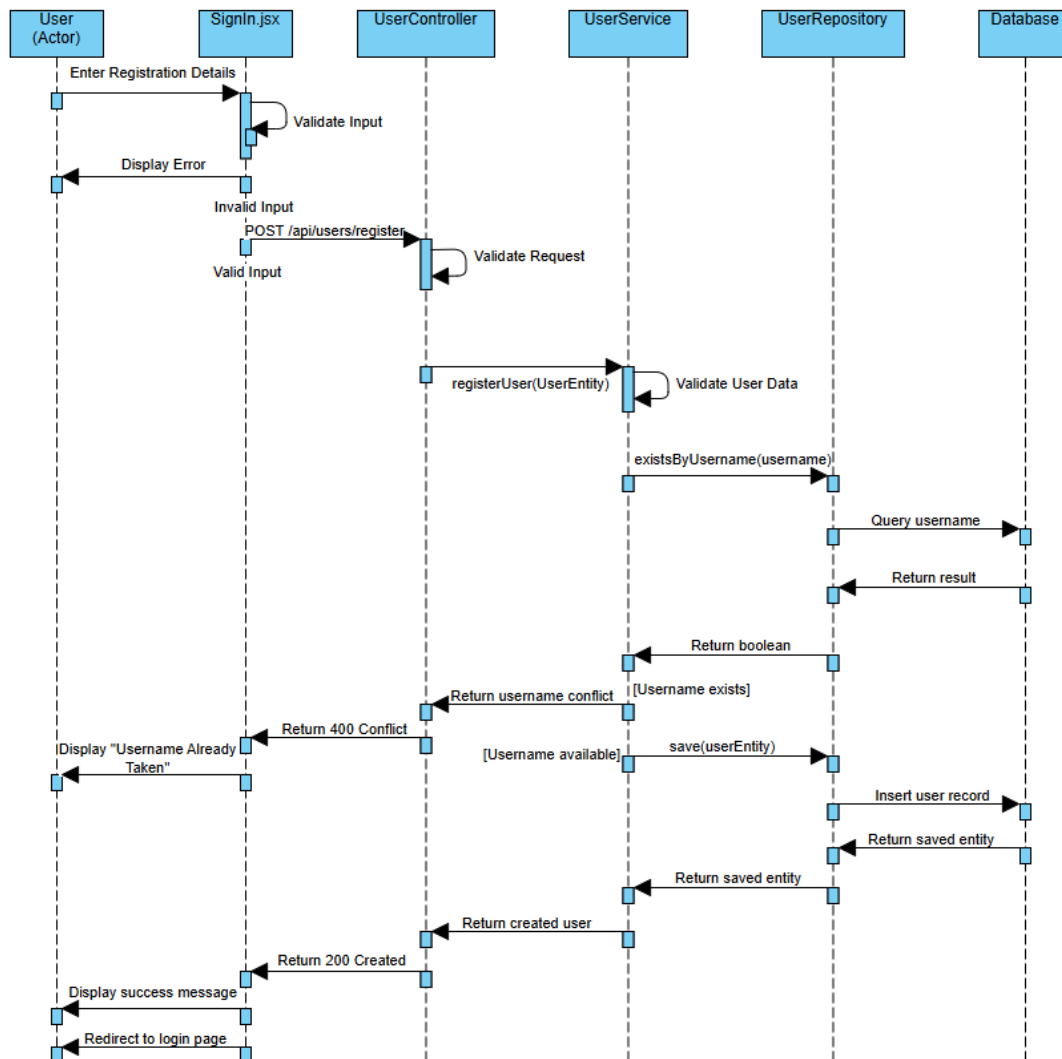
- Front-end component(s)
  - SignIn.jsx
    - **Description and Purpose:** Handles user registration by capturing user input, validating fields, and sending registration requests to the backend.
    - **Component type or format:** React functional component with Material UI for styling.
- Back-end component(s)
  - UserEntity.java
    - **Description and Purpose:** Represents the user data model in the system.
    - **Component type or format:** Spring Boot JPA Entity.
  - UserRepository.java
    - **Description and purpose:** Handles data persistence and interaction with the SQL database.
    - **Component type or format:** Spring Data JPA Repository.
  - UserService.java
    - **Description and purpose:** Implements business logic for user sign-up, including

validation and duplicate checks.

- **Component type or format:** Spring Boot Service Component.
- UserController.java
  - **Description and Purpose:** Handles API requests for user registration, validates input, and stores user data in the database.
  - **Component type or format:** Spring Boot REST Controller.
- Object-Oriented Components
  - Class Diagram

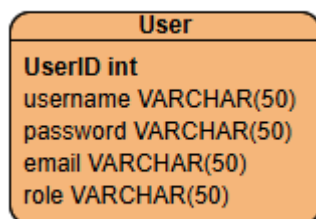


○ Sequence Diagram



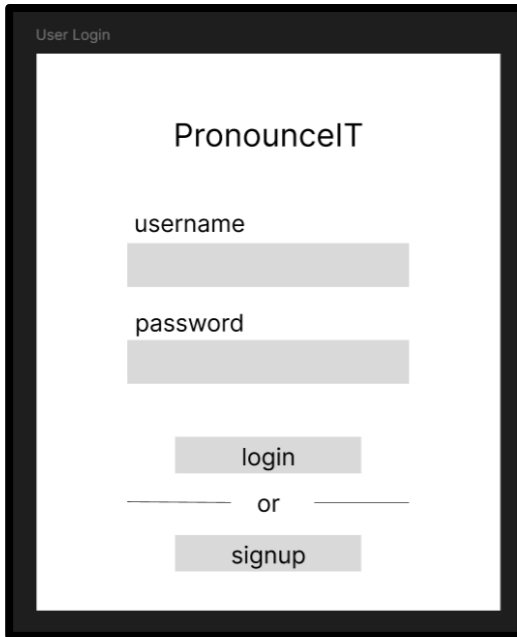
• Data Design

○ ERD or schema



## 1.2 Log-in

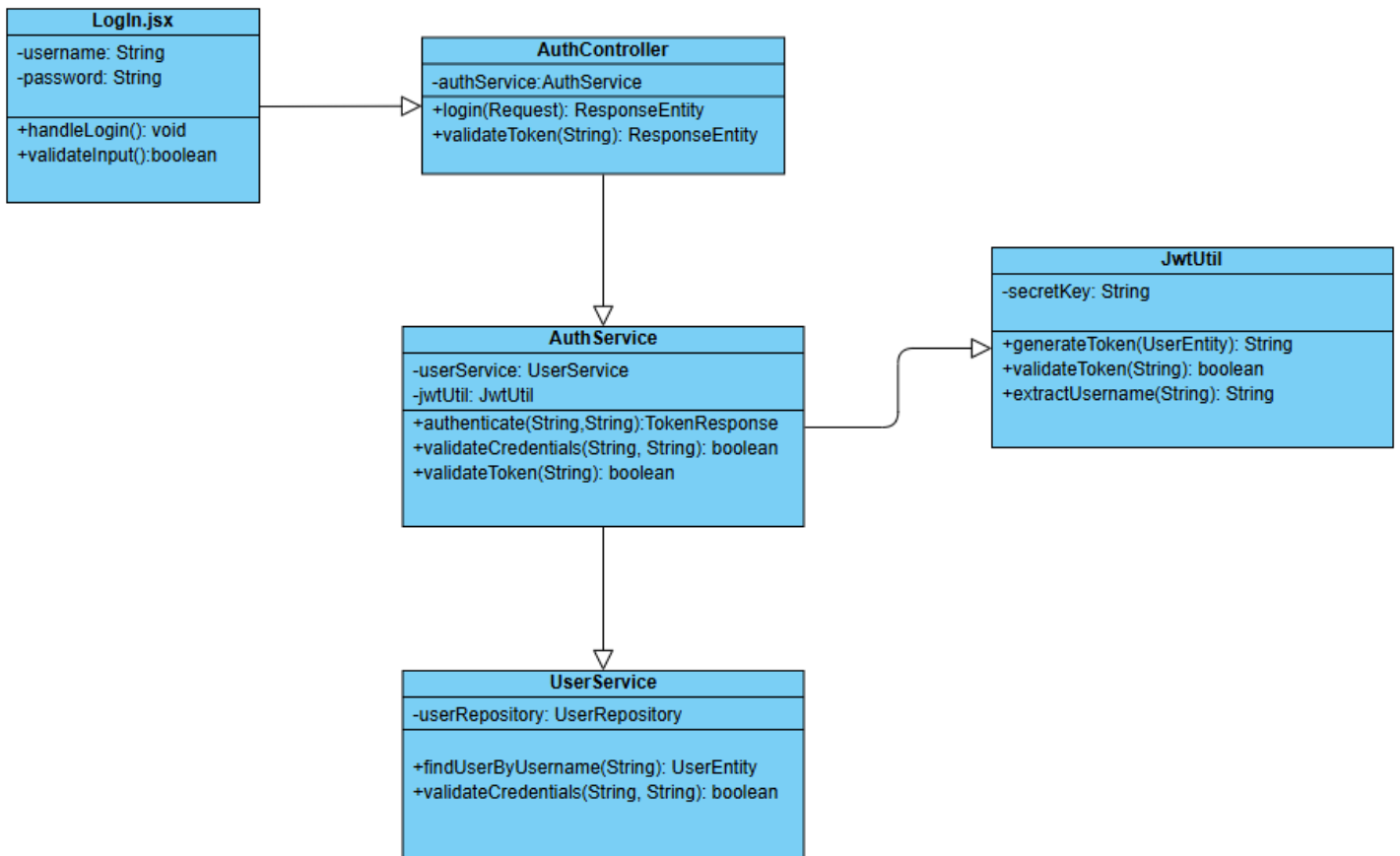
- User Interface Design



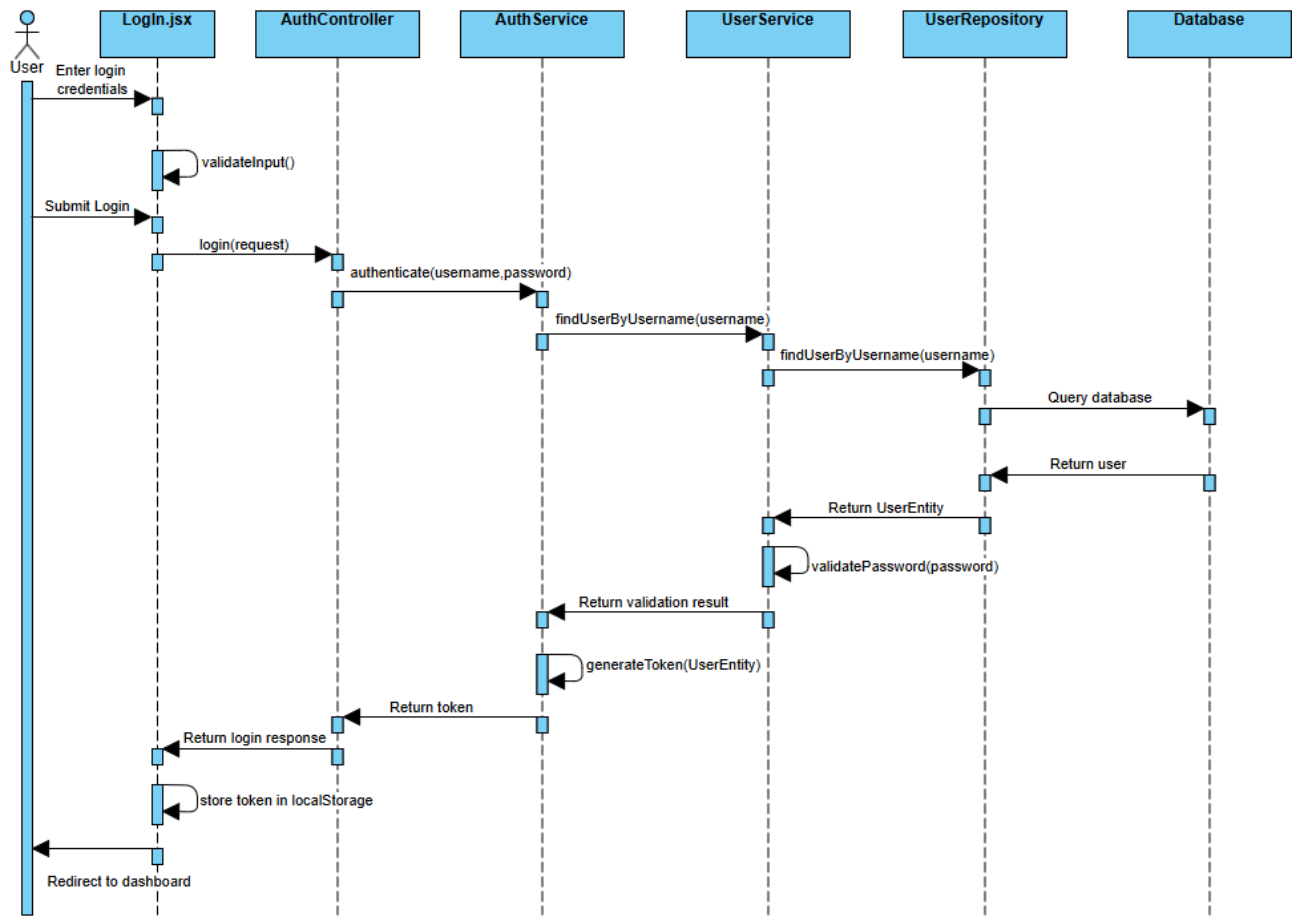
The image shows a user login interface for 'PronounceIT'. It features a title 'PronounceIT' at the top. Below the title are two input fields: 'username' and 'password'. Each input field is represented by a light gray rectangular box. Below the password field is a 'login' button, also a light gray rectangle. Below the 'login' button is the text 'or' centered between two horizontal lines. Below that is a 'signup' button, another light gray rectangle. The entire form is enclosed in a black border. In the top-left corner of the black border, the text 'User Login' is visible.

- Front-end component(s)
  - LogIn.jsx
    - **Description and purpose:** Captures user credentials, validates input, and sends authentication requests to the backend.
    - **Component type or format:** React Functional Component with Material UI.
- Back-end component(s)
  - AuthService.java
    - **Description and purpose:** Implements authentication logic, including credential verification and token generation.
    - **Component type or format:** Spring Boot Service Component
  - AuthController.java
    - **Description and purpose:** Handles authentication requests, validates credentials, and issues JWT tokens.
    - **Component type or format:** Spring Boot REST Controller
  - JwtUtil.java
    - **Description and purpose:** Generates and validates JWT tokens for user authentication.
    - **Component type or format:** Utility class.
- Object-Oriented Components

○ Class Diagram



○ Sequence Diagram



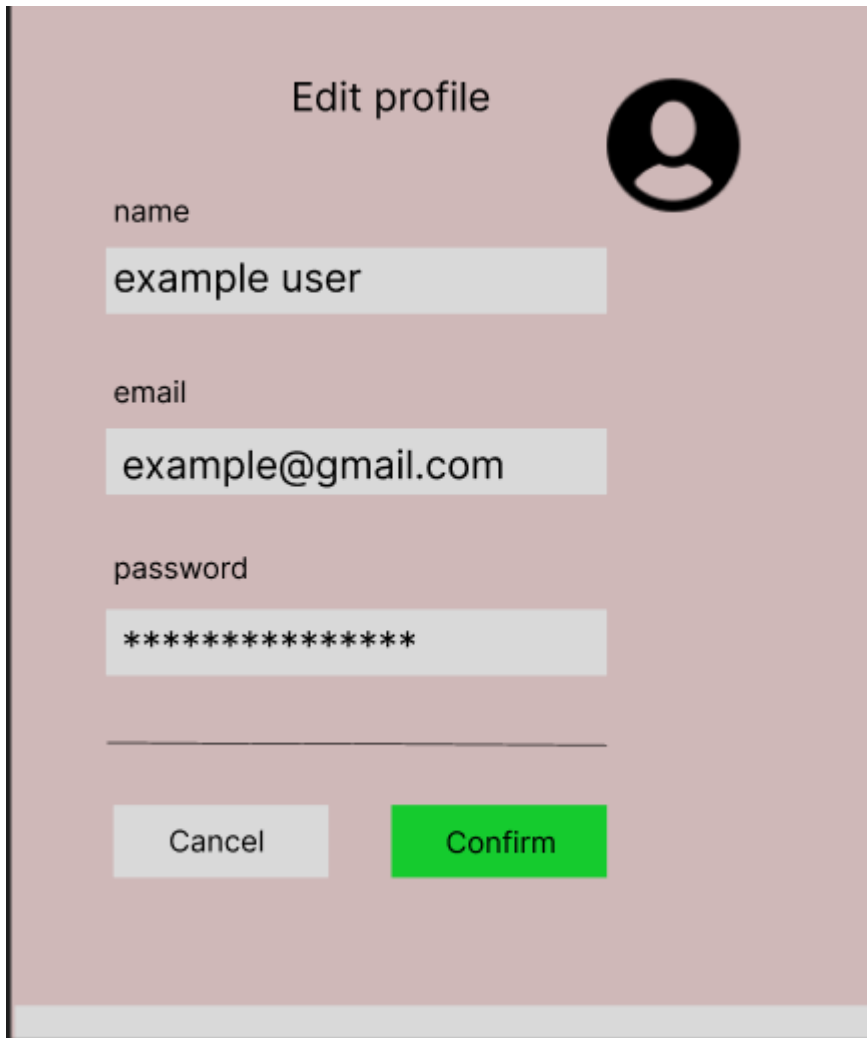
- Data Design
  - ERD or schema



**Module 2: User Profile Management**

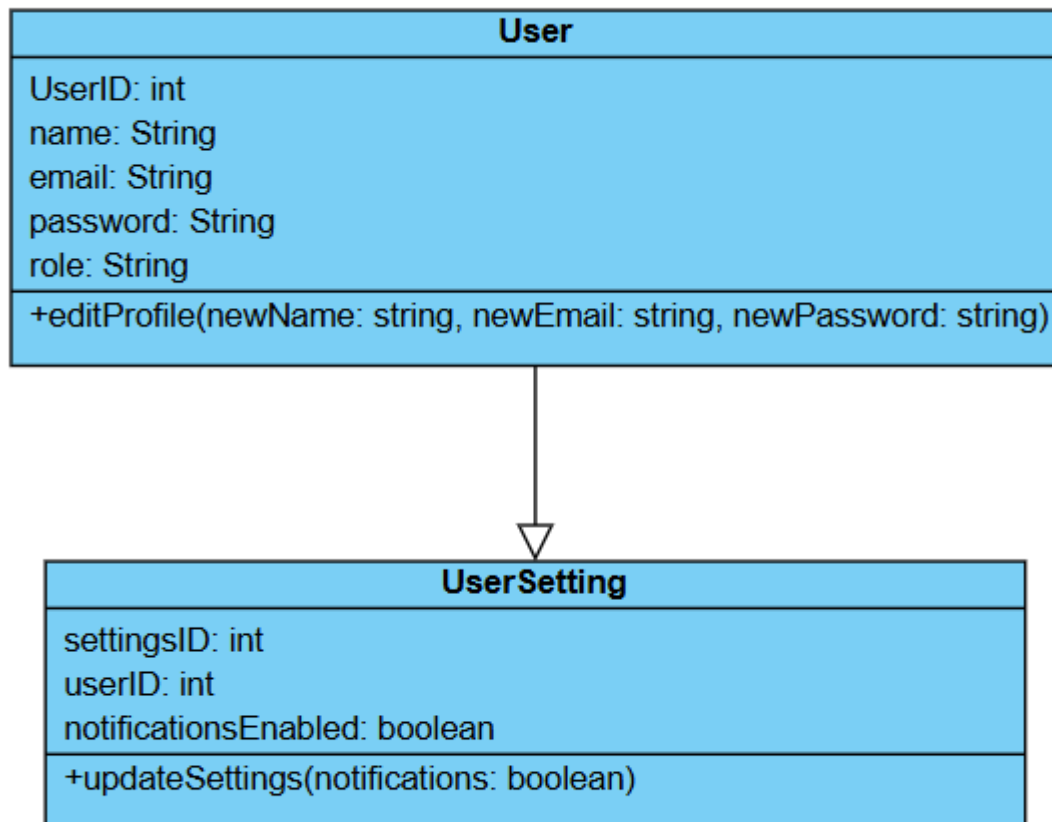
**2.1 Edit User Profile**

- User Interface Design

A screenshot of a web form titled "Edit profile". The form has a light pink background. At the top right is a circular profile picture placeholder. Below the title are three input fields: "name" with the text "example user", "email" with the text "example@gmail.com", and "password" with masked text "\*\*\*\*\*". At the bottom are two buttons: "Cancel" (light gray) and "Confirm" (green).

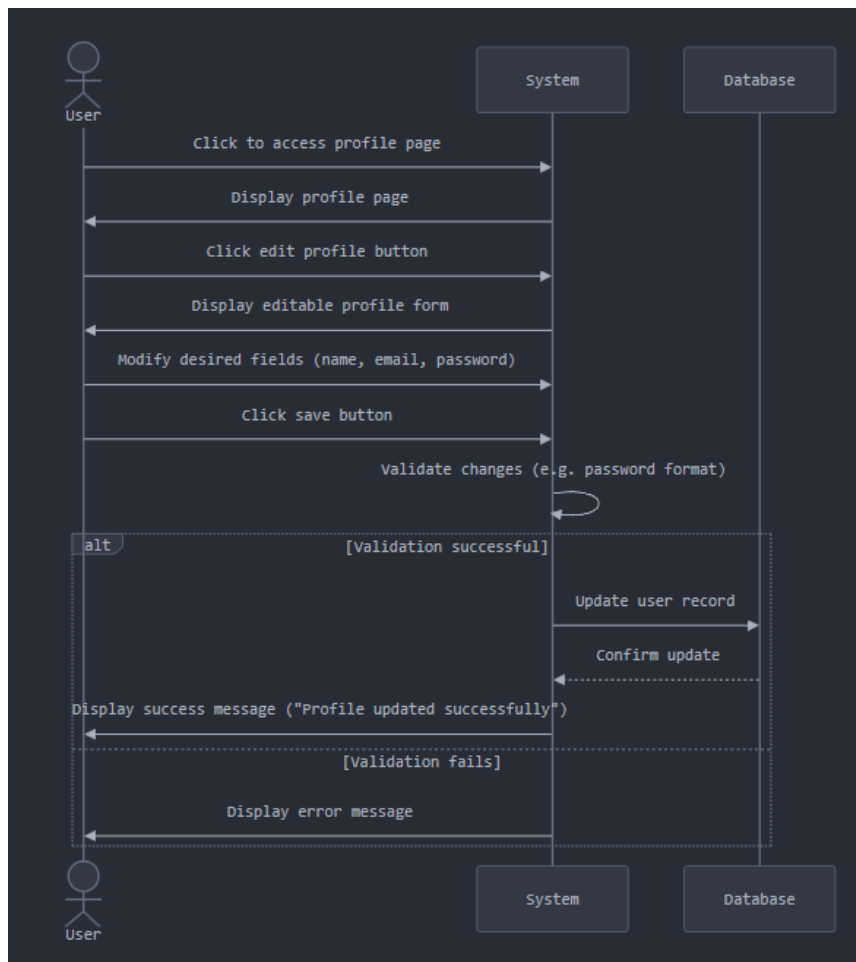
- Front-end component(s)
  - Edit User Profile Interface
    - **Description and Purpose:** Allows users to modify their personal information, including name, email, and password.
    - **Component Type or Format:** Web-based interface (React, JavaScript, CSS), Mobile-based interface (Android Studio, XML, Kotlin)
- Back-end component(s)
  - User Profile Management System
    - **Description and Purpose:** Handles profile updates and ensures data validation and security.
    - **Component Type or Format:** RESTful API service (Spring Boot), Database (MySQL), Server-side scripts (Java, JPA, Hibernate)
- Object-Oriented Components

- Class Diagram

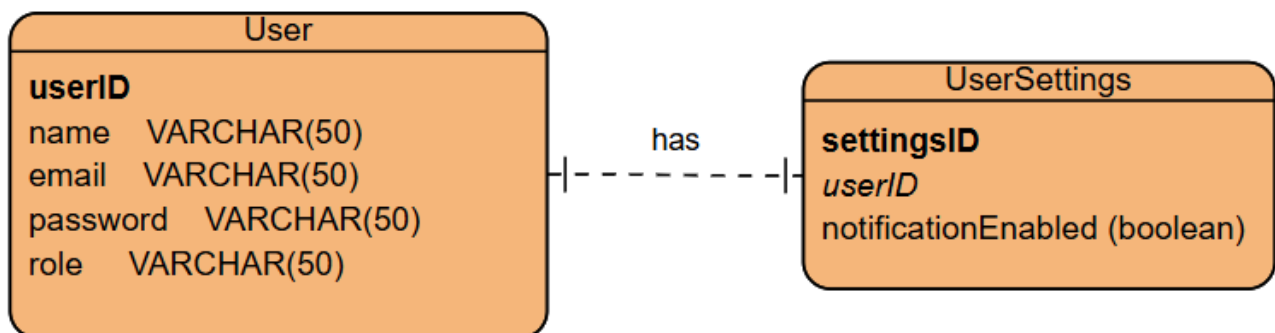


- Sequence Diagram



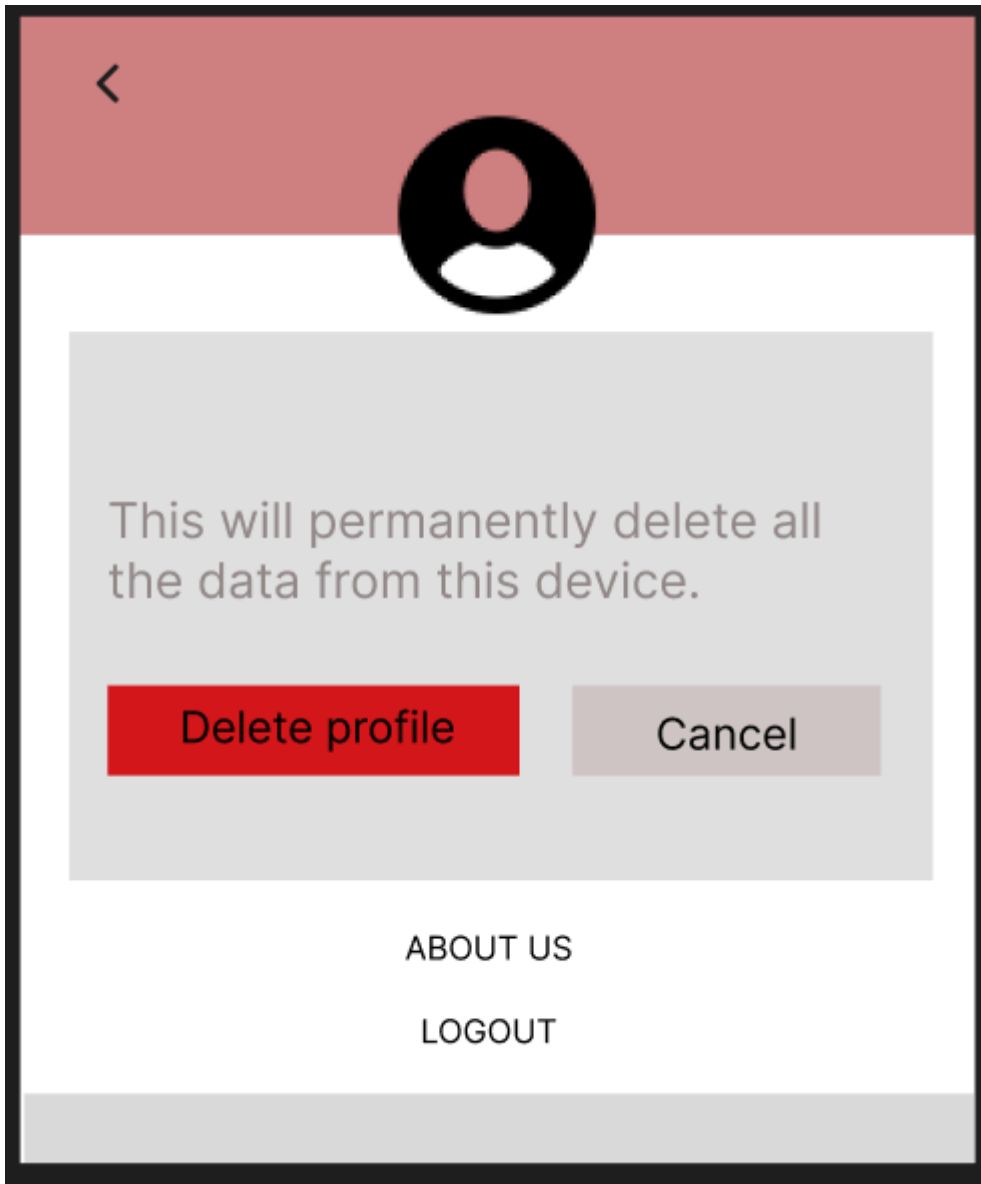


- Data Design
  - ERD or schema



## 2.2 Delete User Profile

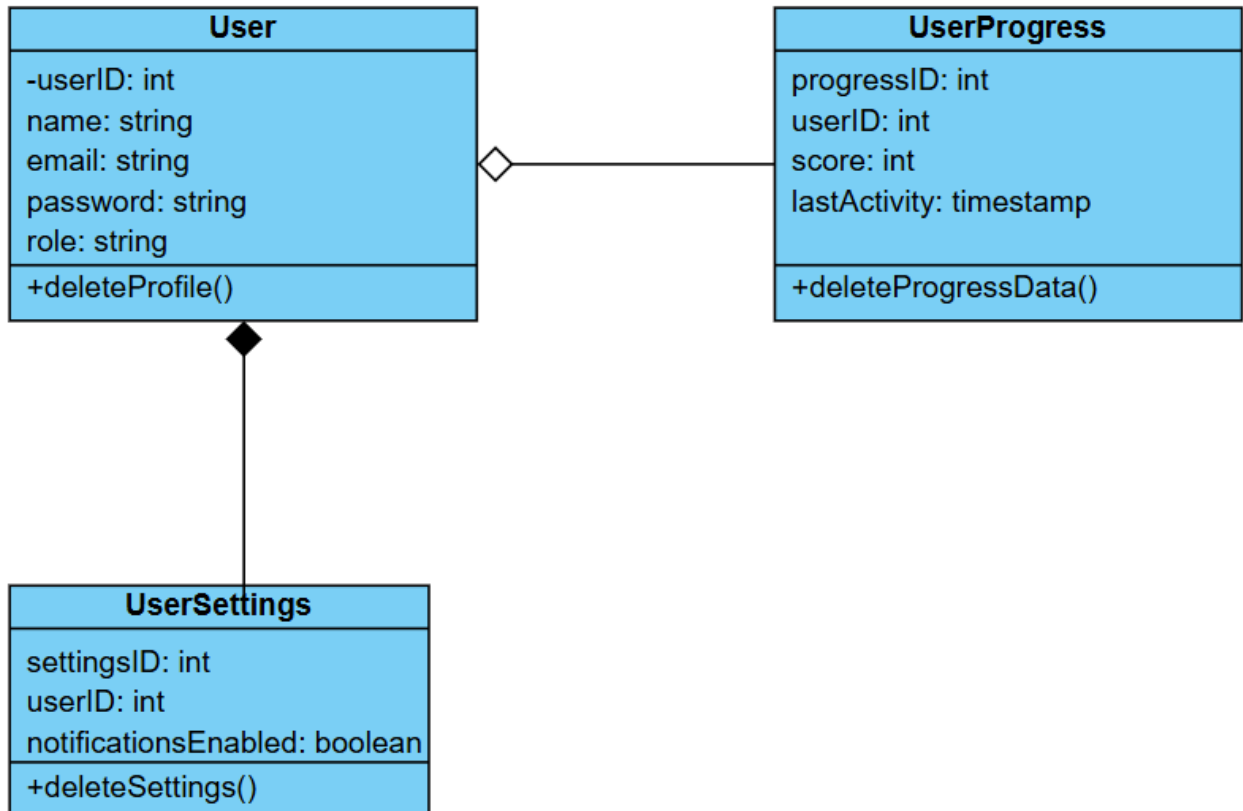
- User Interface Design



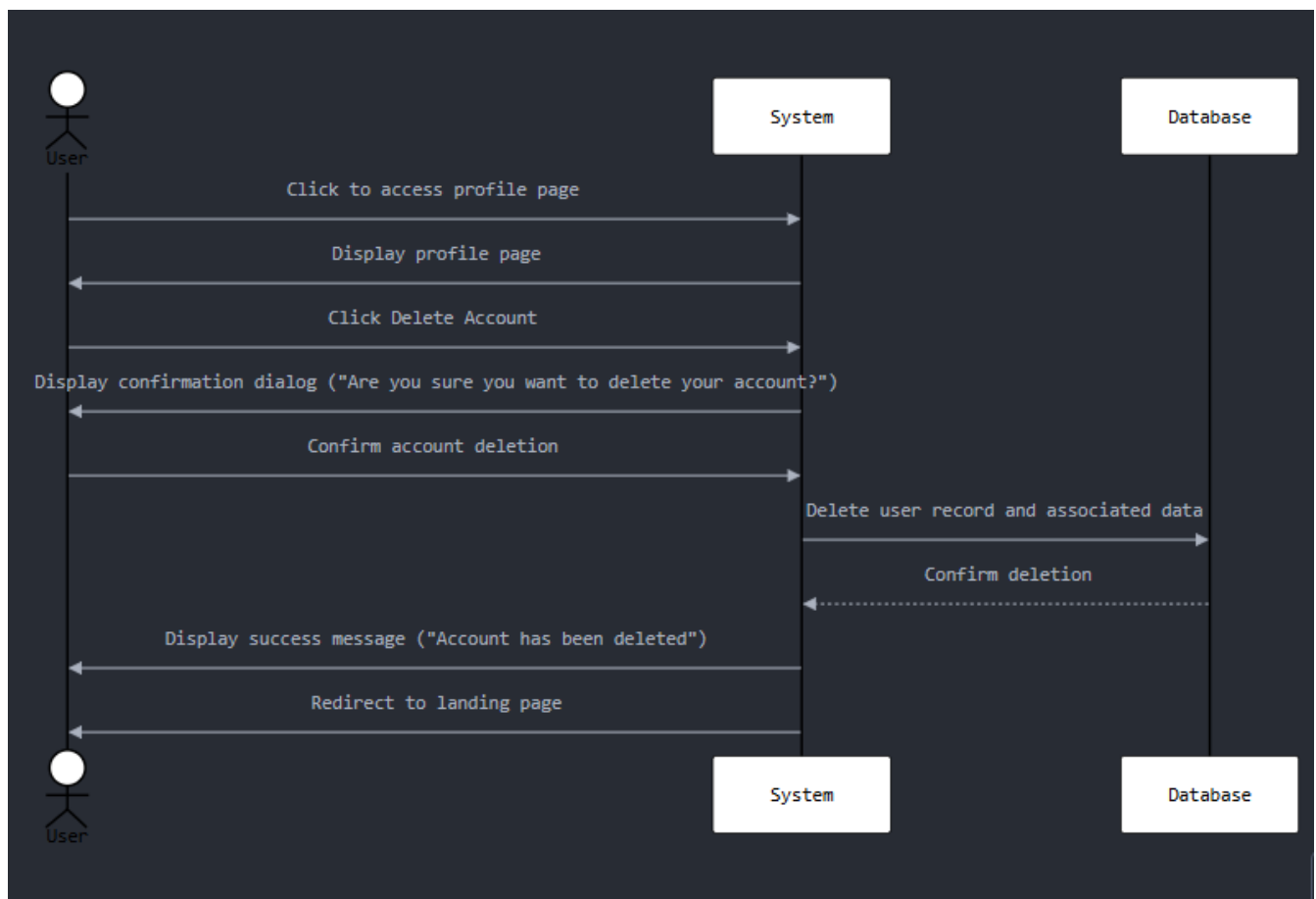
- Front-end component(s)
  - Delete User Profile Interface
    - Description and Purpose: Allows users to permanently delete their accounts, with a confirmation prompt to prevent accidental deletion.
    - Component Type or Format: Web-based interface (React, JavaScript, CSS), Mobile-based interface (Android Studio, XML, Kotlin)
- Back-end component(s)
  - User Profile Management System
    - Description and Purpose: Handles user profile deletion and removes associated

data.

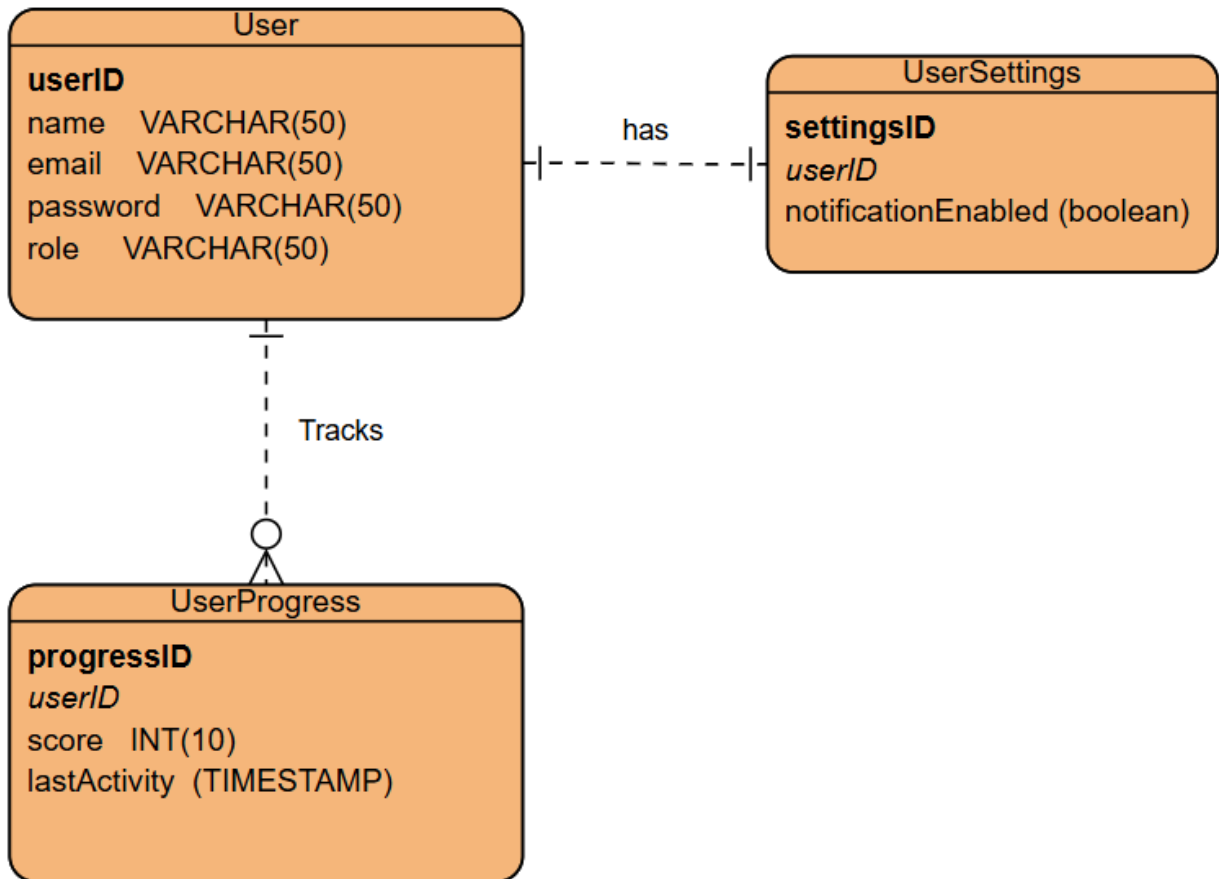
- Component Type or Format: RESTful API service (Spring Boot), Database (MySQL), Server-side scripts (Java, JPA, Hibernate)
- Object-Oriented Components
  - Class Diagram



- Sequence Diagram

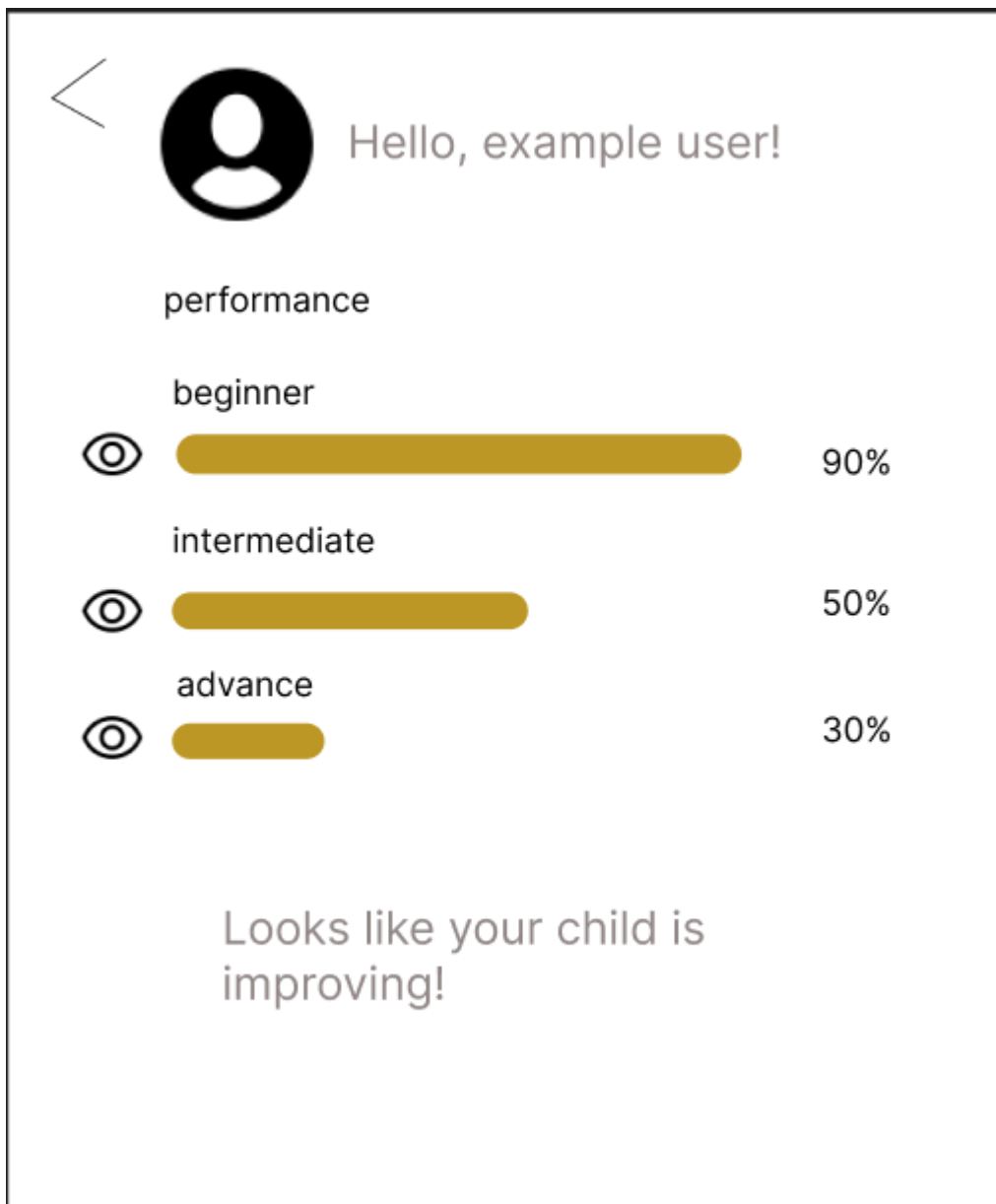


- Data Design
  - ERD or schema



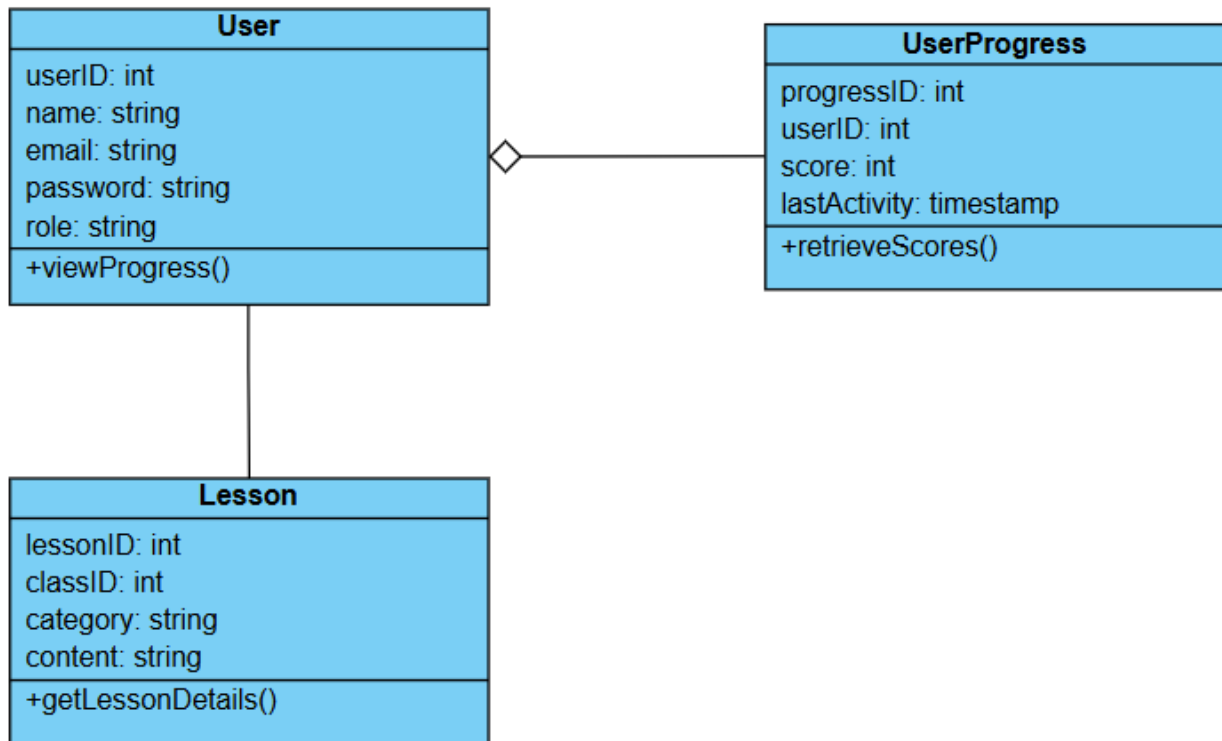
## 2.3 View User Progress and Scores

- User Interface Design

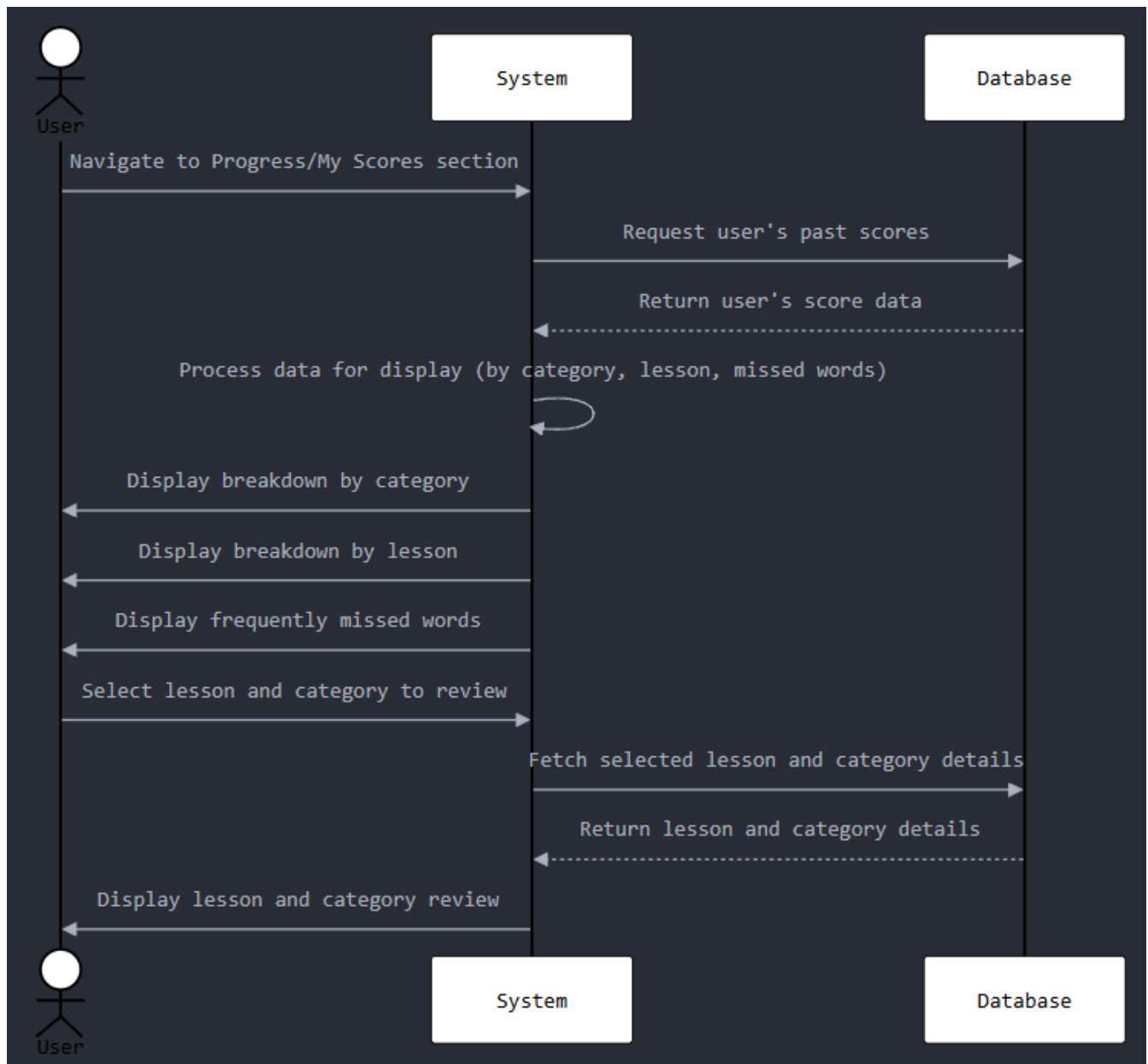


- Front-end component(s)
  - View User Progress & Scores Interface
    - Description and Purpose: Displays user progress and scores in a structured format with visual representations like tables and charts.
    - Component Type or Format: Web-based interface (React, JavaScript, CSS), Mobile-based interface (Android Studio, XML, Kotlin)
- Back-end component(s)
  - User Progress Tracking System
    - Description and Purpose: Manages and retrieves user scores and progress, ensuring accurate reporting.

- **Component Type or Format:** RESTful API service (Spring Boot), Database (MySQL), Server-side scripts (Java, JPA, Hibernate)
- Object-Oriented Components
  - Class Diagram

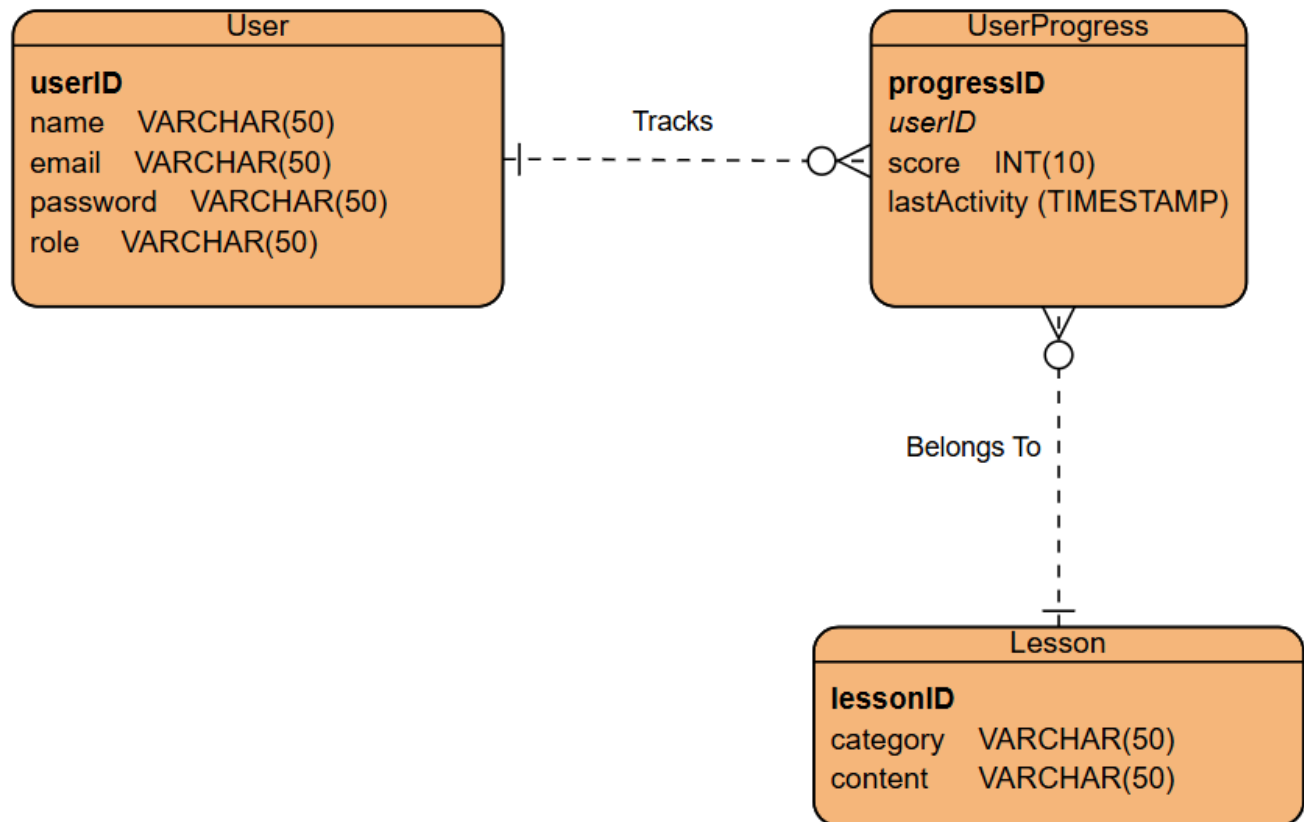


- Sequence Diagram



- Data Design
  - ERD or schema



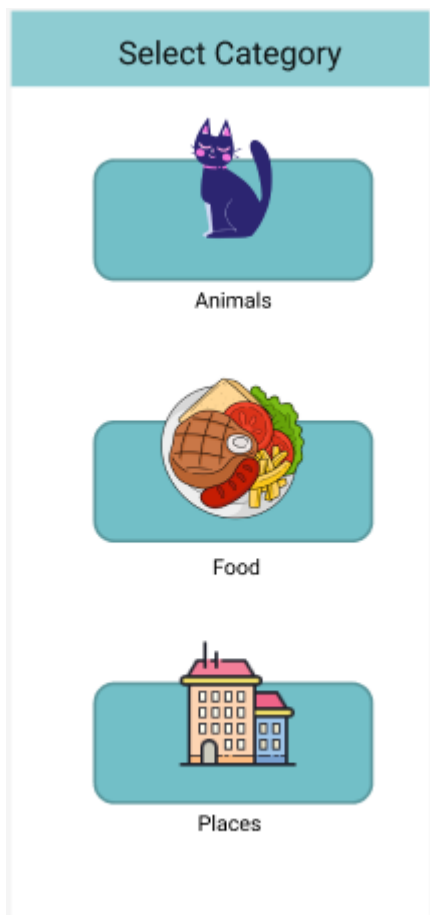


## Module 3: Gameplay

---

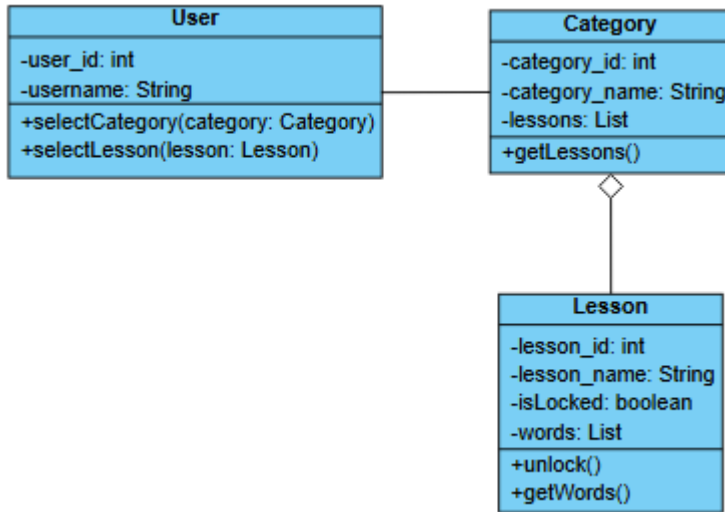
### 3.1 Select Category and Lesson

- User Interface Design

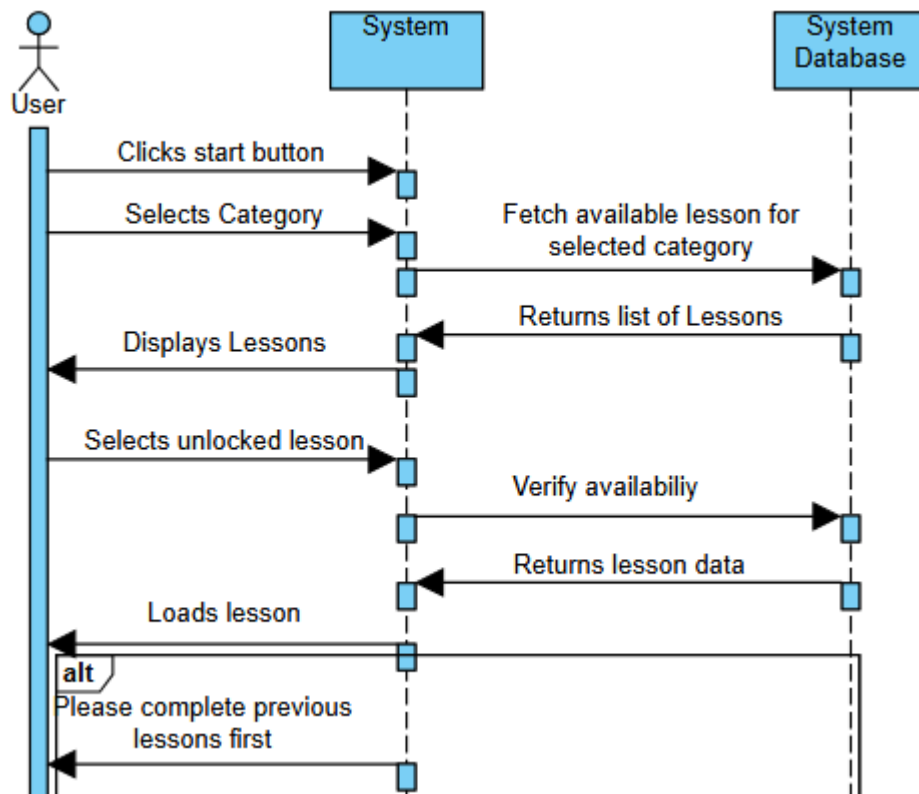


- Front-end component(s)
  - Category Selection UI
    - **Description and Purpose:** Allows users to select a category from a predefined list. Displays available categories with interactive elements.
    - **Component Type or Format:** HTML, CSS, JavaScript (React)
  - Lesson (Letter) Selection UI
    - **Description and Purpose:** Allows users to select a lesson (represented as a letter) within a chosen category. Displays available and locked lessons with progress indicators.
    - **Component Type or Format:** HTML, CSS, JavaScript (React)
- Back-end component(s)
  - Category Handler
    - **Description and Purpose:** Fetches category data from the database and provides structured responses to the front end.
    - **Component Type or Format:** Spring Boot (Java), REST API, PostgreSQL

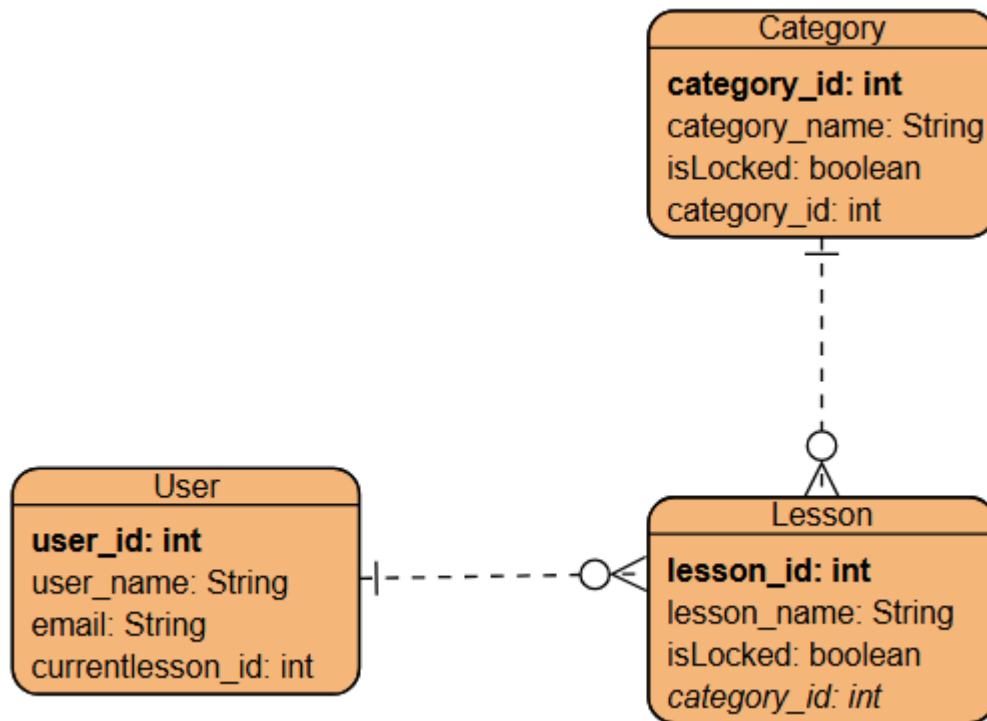
- Lesson Handler
  - **Description and Purpose:** Fetches lesson data from the database, verifies lesson availability, and updates user progress.
- **Component Type or Format:** Spring Boot (Java), REST API, PostgreSQL
  - Object-Oriented Components
    - Class Diagram



- Sequence Diagram



- Data Design
  - ERD or schema

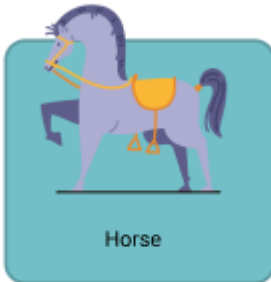


### 3.2 Core Gameplay Mechanics

- User Interface Design

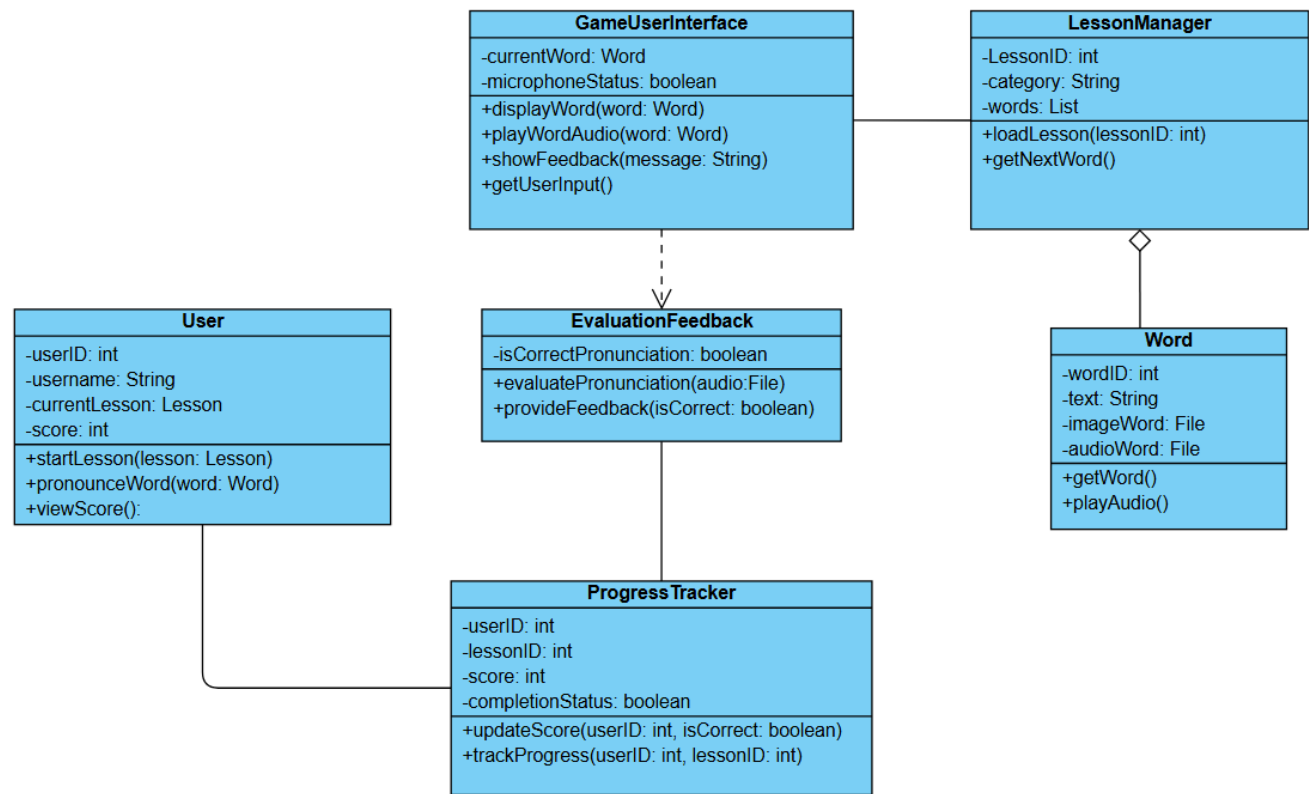
Lesson: H Words: 4/10

Score: 3/10



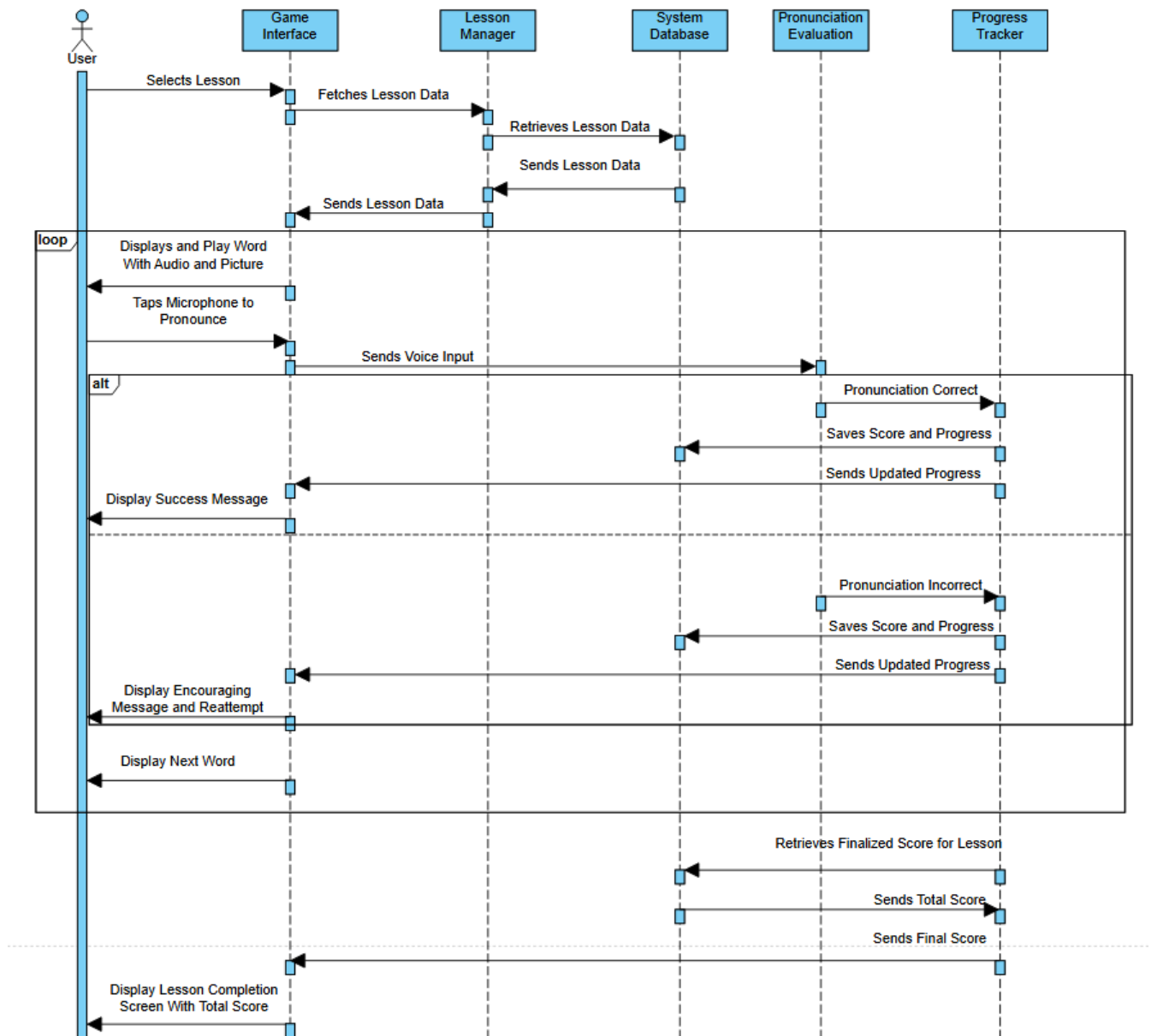
- Front-end component(s)
  - Word Display
    - **Description and Purpose:** Displays the word to be pronounced by the user.
    - **Component Type or Format:** Android Studio, Kotlin
  - Picture Word
    - **Description and Purpose:** Displays an image corresponding to the word being pronounced to aid visual learning.
    - **Component Type or Format:** Android Studio, Kotlin
  - Progress Tracker UI
    - **Description and Purpose:** Shows the user's current score, word number, lesson, and category.
    - **Component Type or Format:** Android Studio, Kotlin
  - Message Popup
    - **Description and Purpose:** Displays encouraging messages based on pronunciation performance.
    - **Component Type or Format:** Android Studio, Kotlin, Modal Popup UI
  - Microphone UI

- **Description and Purpose:** Allows users to tap an icon to begin pronouncing the displayed word.
  - **Component Type or Format:** Android Studio, Kotlin
- Back-end component(s)
  - Progress Tracker
    - **Description and Purpose:** Tracks the user's progress throughout the lesson
    - **Component Type or Format:** Spring Boot (Java), PostgreSQL
  - Voice and Speech Recognition
    - **Description and Purpose:** Captures and processes user voice input for pronunciation assessment.
    - **Component Type or Format:** Spring Boot (Java), Google Cloud Speech-to-Text API
  - Pronunciation Evaluation
    - **Description and Purpose:** Analyzes the accuracy of the user's pronunciation and provides feedback.
    - **Component Type or Format:** Spring Boot (Java), AI-based Speech Processing API
  - Scoring System
    - **Description and Purpose:** Calculates and assigns scores based on pronunciation accuracy and lesson completion.
    - **Component Type or Format:** Spring Boot (Java), PostgreSQL
- Object-Oriented Components
  - Class Diagram

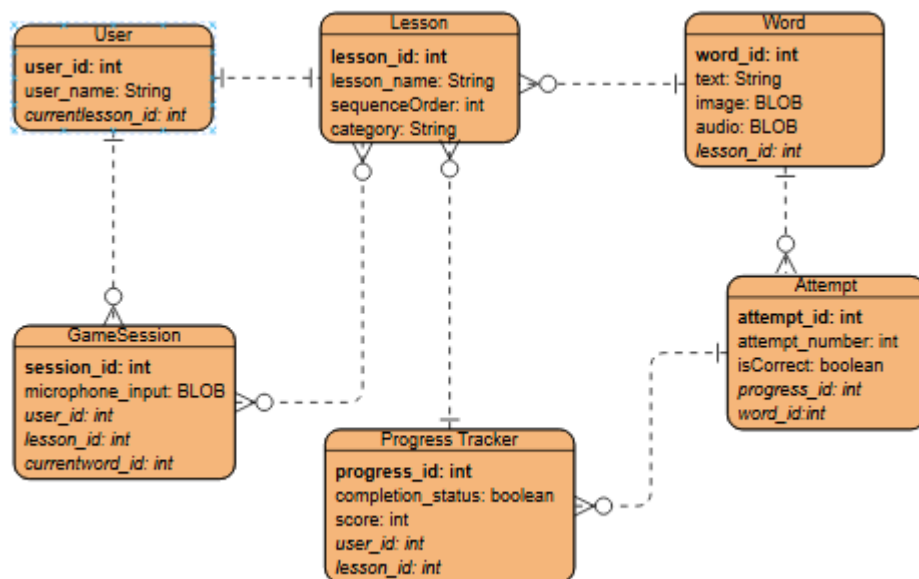


- Sequence Diagram





- Data Design
  - ERD or schema



### 3.3 User Retry Lesson

- User Interface Design

Lesson: H Words: 10/10

Score: 8/10

Lesson Completed



You Passed!



Retry Level



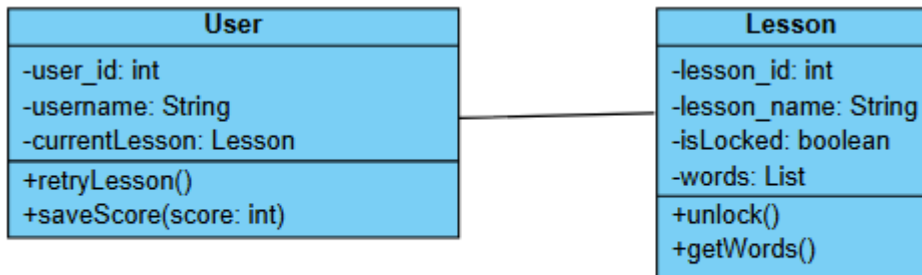
View Score Details



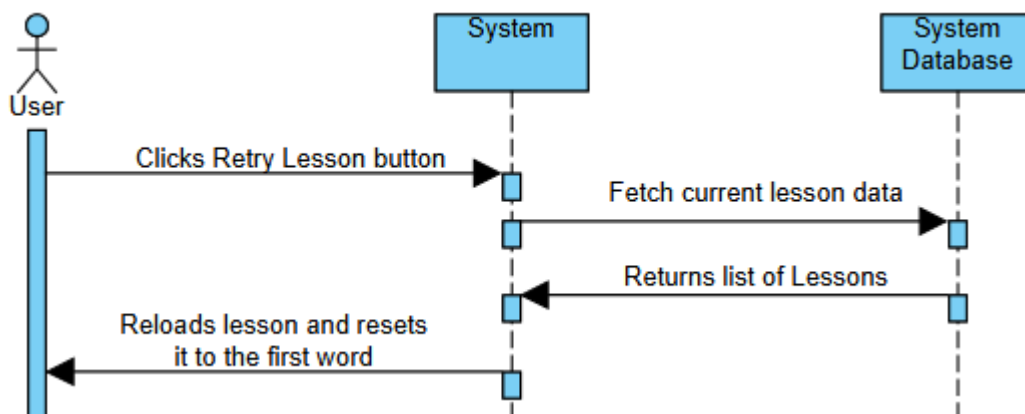
Next Level

- Front-end component(s)
  - Retry Button UI
    - **Description and Purpose:** Allows users to restart the lesson and attempt pronunciation exercises again.
    - **Component Type or Format:** Android Studio, Kotlin
- Back-end component(s)
  - Lesson Restart Handler
    - **Description and Purpose:** Allows users to replay a completed lesson
    - **Component Type or Format:** Spring Boot (Java), REST API, PostgreSQL
- Object-Oriented Components

- Class Diagram

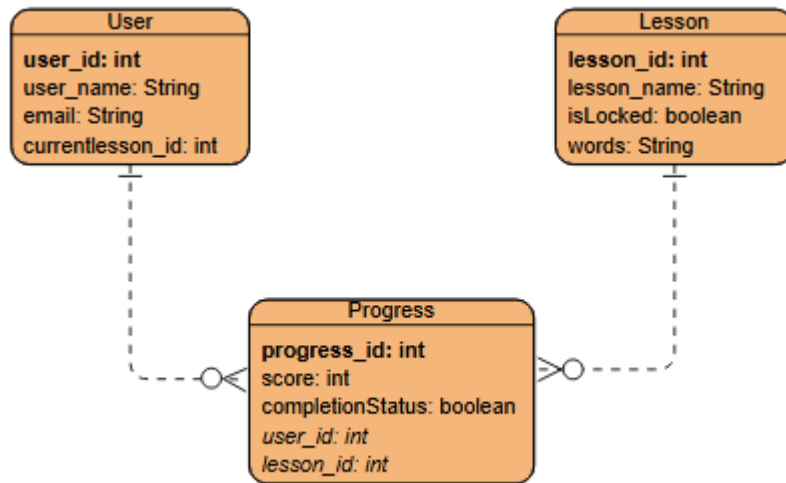


- Sequence Diagram



- Data Design

- ERD or schema



### 3.4 View Score Details

- User Interface Design

Lesson: H Words: 10/10

Score: 8/10

Word ✓

Word ✓

Word ✓

Word ✓

Word ✗

Word ✓

Word ✓

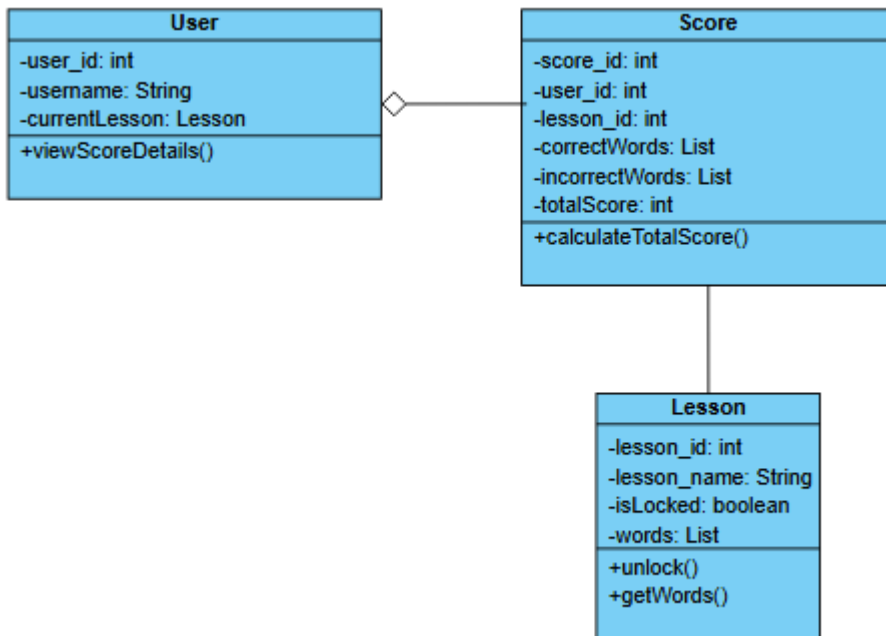
Word ✗

Word ✓

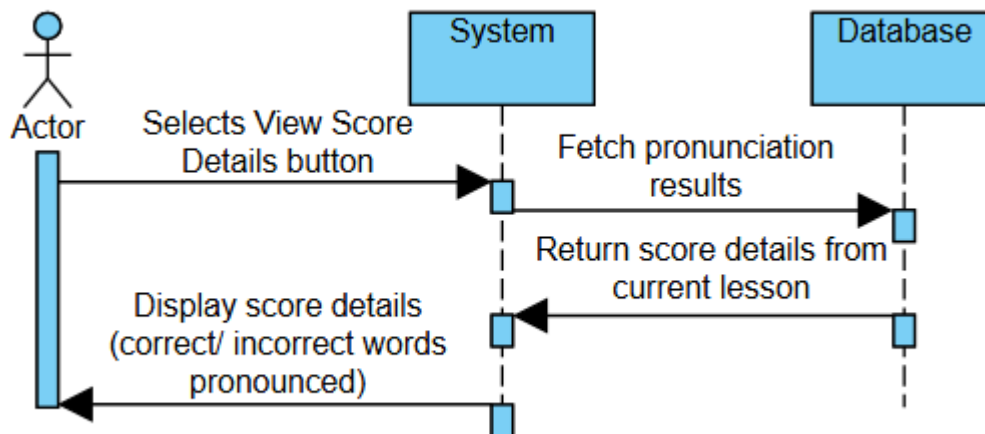
Word ✓

- Front-end component(s)

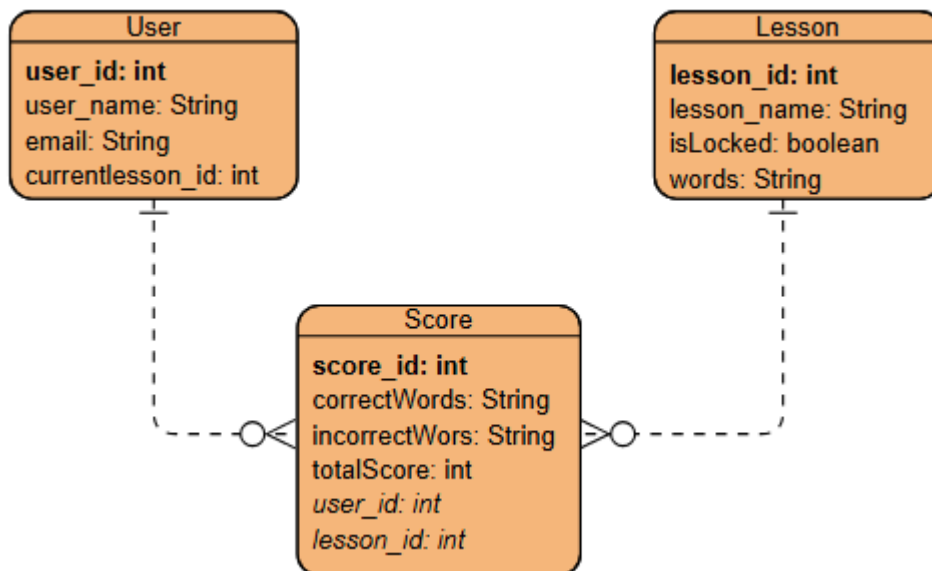
- Score Display UI
  - **Description and Purpose:** Shows detailed score breakdowns for completed lesson attempt, including which words were correctly and incorrectly pronounced.
  - **Component Type or Format:** Android Studio, Kotlin
- Back-end component(s)
  - Score Data Handler
    - **Description and Purpose:** Retrieves and processes user score details for a given lesson.
    - **Component Type or Format:** Spring Boot (Java), REST API, PostgreSQL
  - Object-Oriented Components
    - Class Diagram



- Sequence Diagram



- Data Design
  - ERD or schema



### 3.5 Next Lesson

- User Interface Design

Lesson: H Words: 10/10

Score: 8/10

Lesson Completed



You Passed!



Retry Level

10

View Score Details



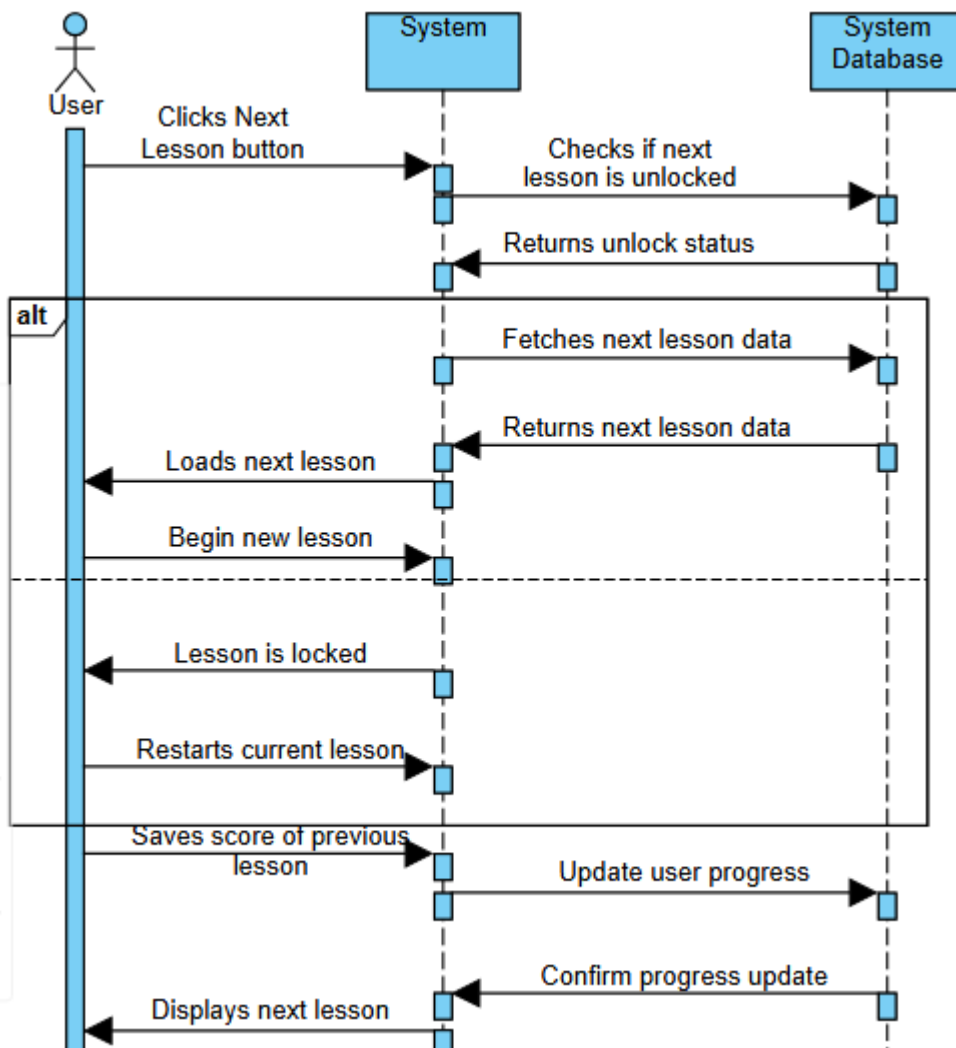
Next Level

- Front-end component(s)
  - Unlock Requirement Popup
    - **Description and Purpose:** Displays a message if the user fails to meet the minimum score to unlock the next lesson.
    - **Component Type or Format:** Android Studio, Modal Popup UI
  - Next Lesson UI
    - **Description and Purpose:** Allows users to navigate to the next available lesson after completing the current one.
    - **Component Type or Format:** Android Studio, Kotlin
- Back-end component(s)
  - Lesson Progress Handler
    - **Description and Purpose:** Determines if the next lesson is available based on the evaluated score from the current lesson.
    - **Component Type or Format:** Spring Boot (Java), REST API, PostgreSQL
- Object-Oriented Components

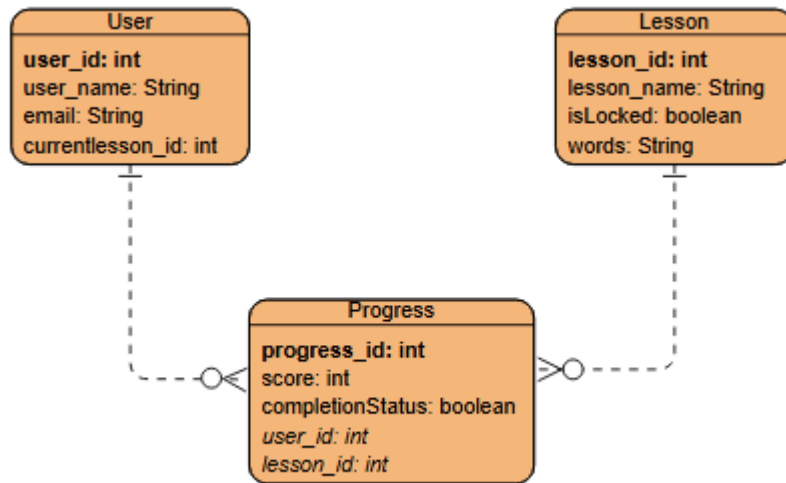


- Class Diagram

- Sequence Diagram



- Data Design
  - ERD or schema



## Module 4: Teacher Management

### 4.1: Teacher Adds Lesson

- User Interface Design

## Add Lesson

### Create New Lesson

Lesson Title

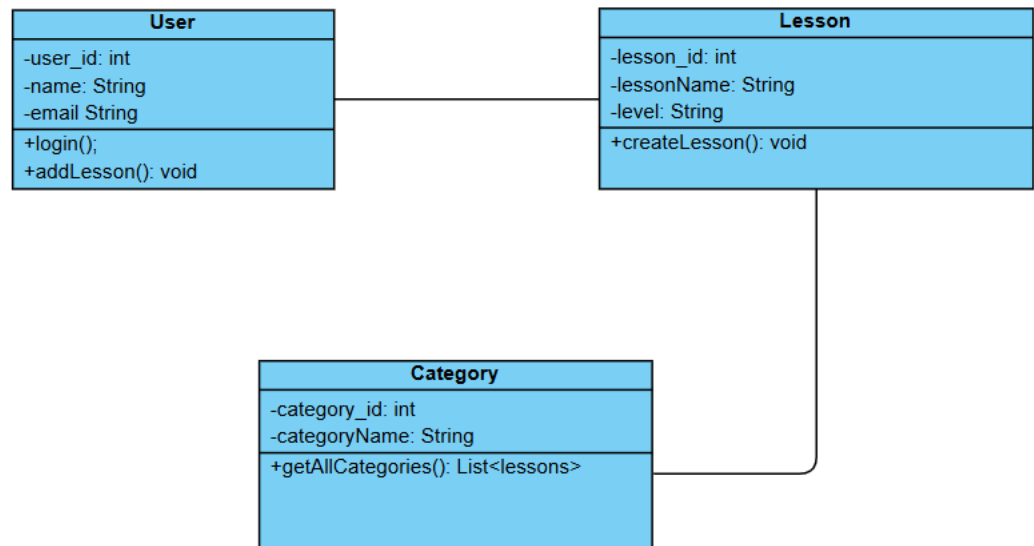
Lesson Description

Submit

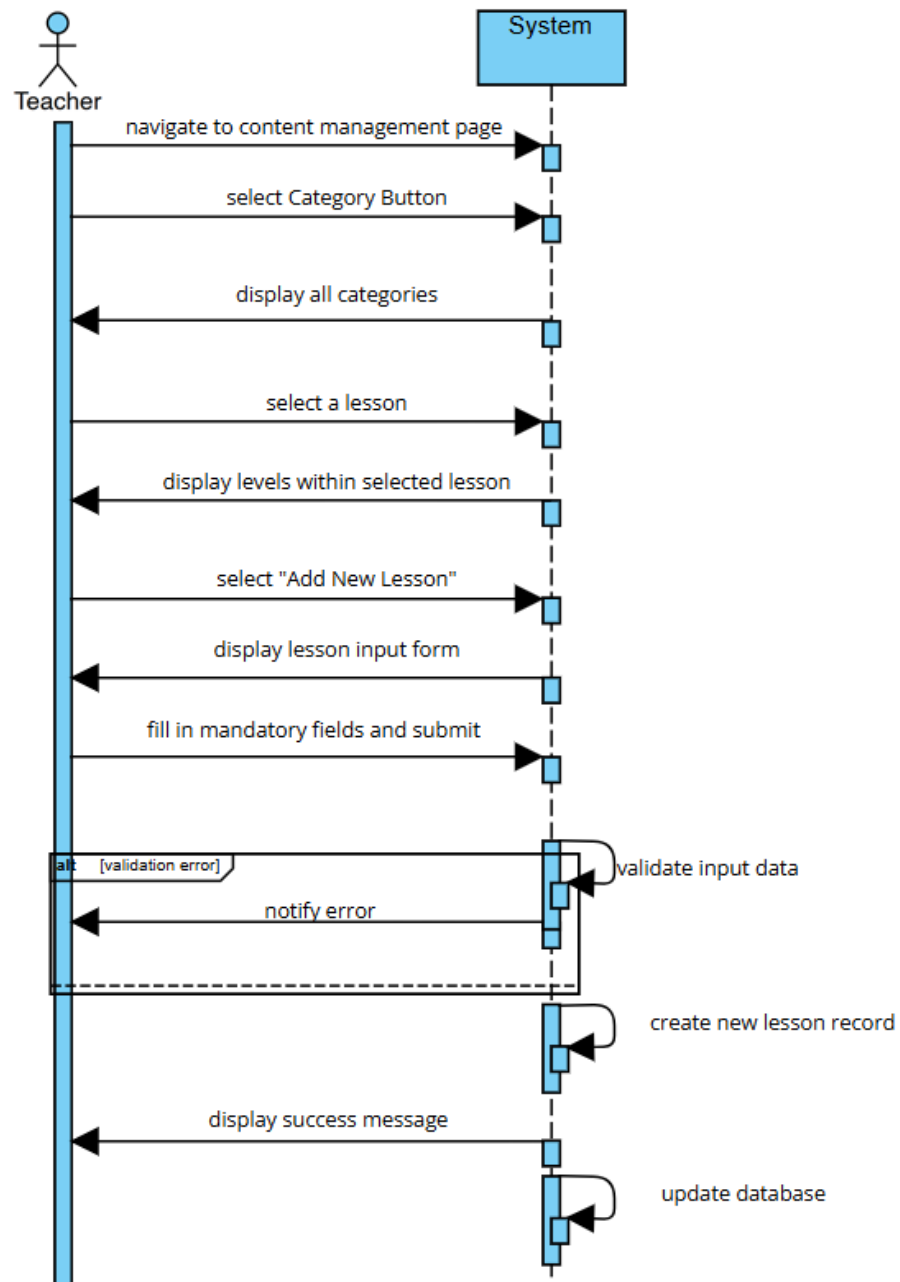
Cancel

- Front-end component(s)
  - **Description and purpose:** Allows the teacher to add a new lesson by inputting details such as lesson name and category.

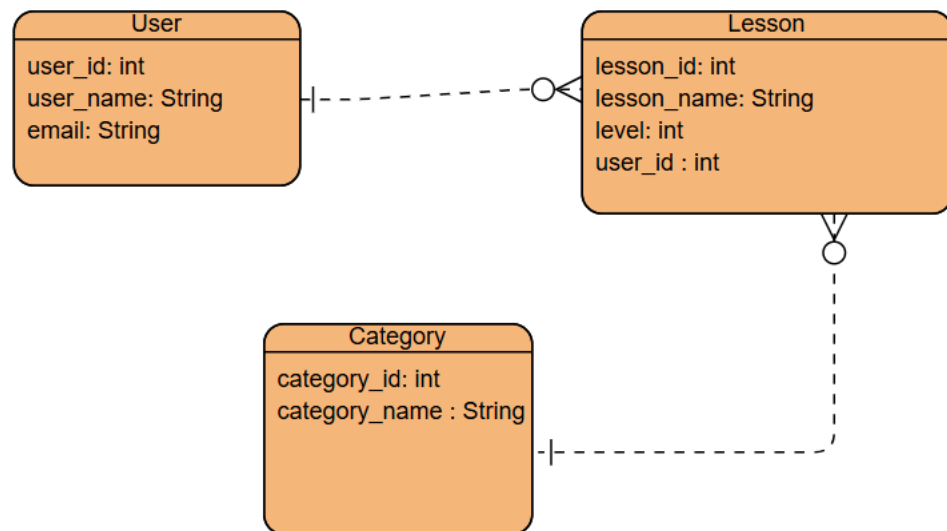
- **Component type or format:** form with text fields, dropdowns, and a submit button.
- Back-end component(s)
  - **Description and purpose:** Handles form submission, validation, and database insertion for a new lesson.
  - **Component type or format:**
    - **API Endpoints:** POST /lesson/add
    - **Database Model:** Lesson (id, name, category\_id, created\_at, updated\_at)
    - **Validation:** Ensures lesson name is unique and required fields are filled
- Object-Oriented Components
  - Class Diagram



- Sequence Diagram



- Data Design
  - ERD or schema



## 4.2: Teacher Deletes Lesson

- User Interface Design

# Delete Lesson

## Lessons

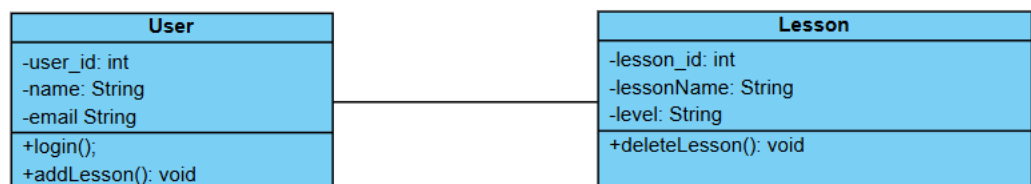
Lesson 1

Delete

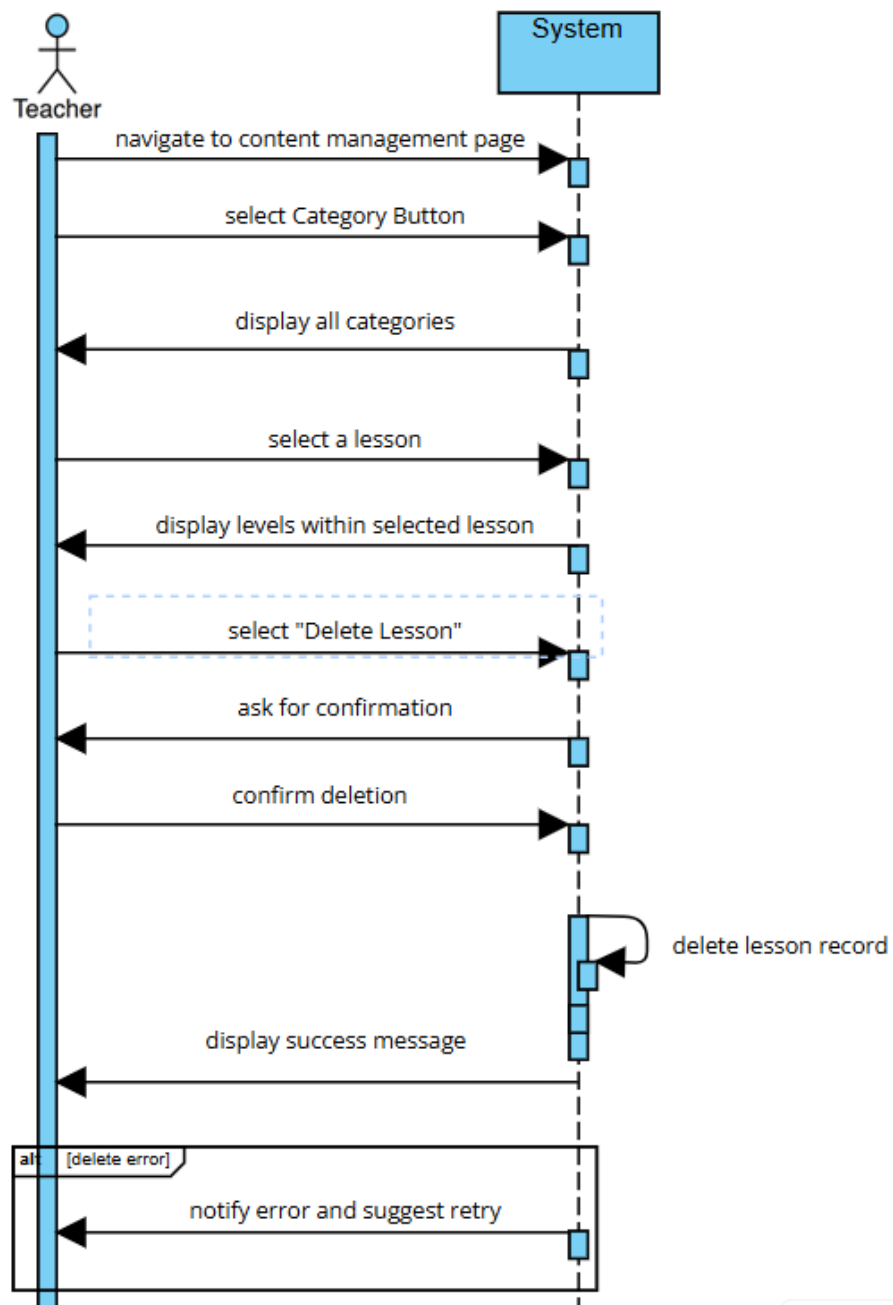
Lesson 2

Delete

- Front-end component(s)
  - **Description and purpose:** Allows the teacher to delete an existing lesson.
  - **Component type or format:**
    - **UI Elements:** Lesson list with delete button, confirmation modal
    - **Form Elements:** Delete confirmation dialog
- Back-end component(s)
  - **Description and purpose:** Handles lesson deletion from the database.
  - **Component type or format:** API endpoint: DELETE /lessons/{id}
- Object-Oriented Components
  - Class Diagram

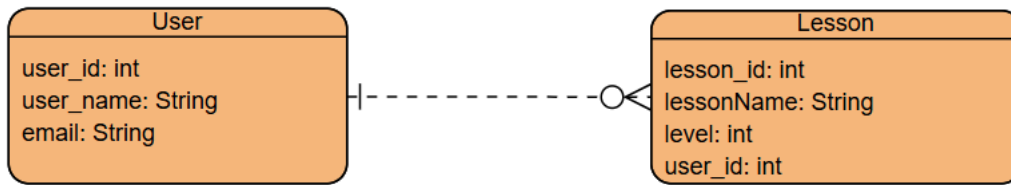


- Sequence Diagram



- Data Design

- ERD or schema



### 4.3: Teacher Adds Content

- User Interface Design

## Add Content

### Create New Content

Choose File

No file chosen

Submit

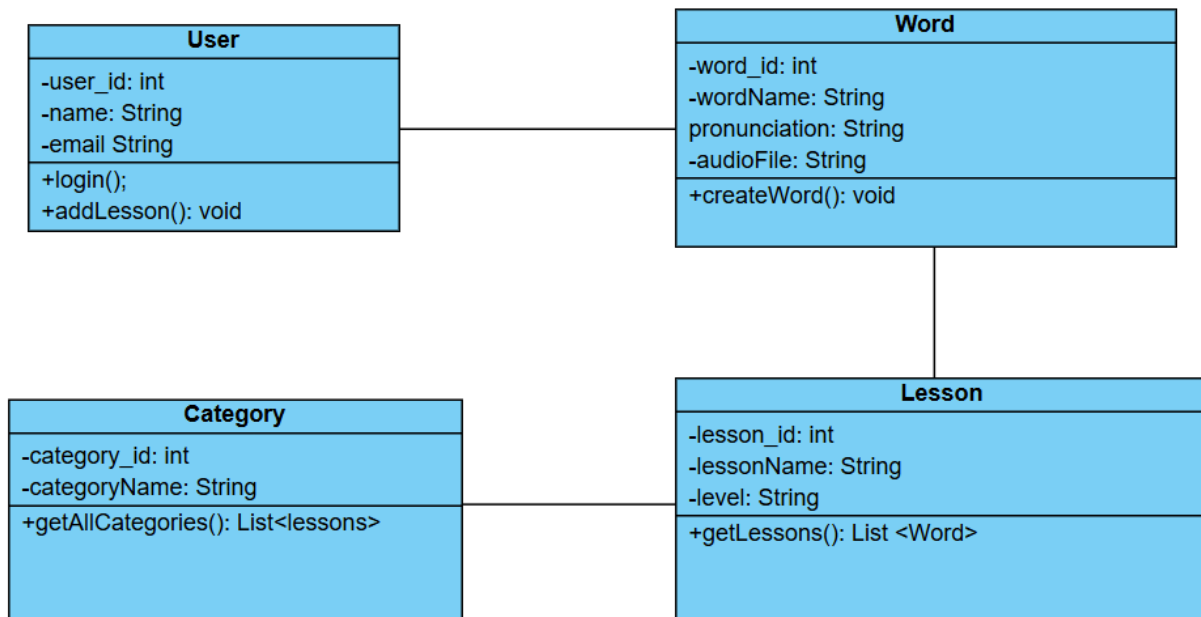
Cancel

- Front-end component(s)
  - **Description and purpose:**
    - A form-based interface that allows teachers to add content (e.g., words, pronunciation guides, and audio files) to lessons.

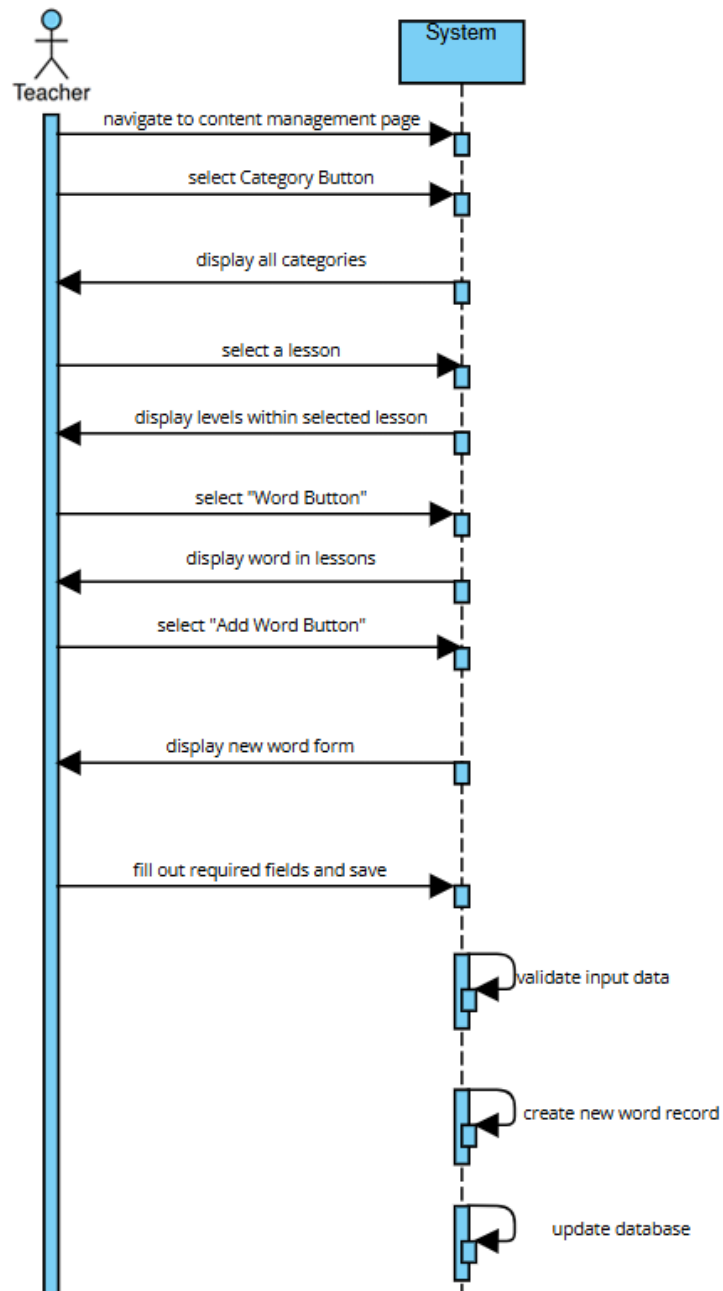


- Fields include **word/phrase, description, pronunciation guide, and audio file upload** (optional).
- A dropdown to select the lesson to which the content belongs.
- **Component type or format:**
  - Input fields: text fields for words and pronunciation guides.
  - File upload: Supports MP3/OGG/WAV for audio pronunciation.
  - Dropdown menu: To select the lesson.
  - Buttons: "Submit" (adds content) and "Cancel."
  - Framework Used: React.js (for modern SPAs) or basic HTML/CSS/JS for simpler implementations.
- Back-end component(s)
  - **Description and purpose:**
    - Handles new content creation and stores it in the database.
    - Validates input (e.g., ensuring no duplicate words in the same lesson).
    - Processes audio files and links them to the corresponding word or phrase.
  - **Component type or format:**
    - API Endpoint: POST /content/add
    - Backend Language: React.js, Spring Boot.
    - Database Interaction: MySQL/PostgreSQL
    - Middleware: For audio file upload (e.g., Multer for Node.js or Django File Storage).
    - Validation Logic: Ensures valid file format, non-empty fields, and correct lesson association.
- Object-Oriented Components

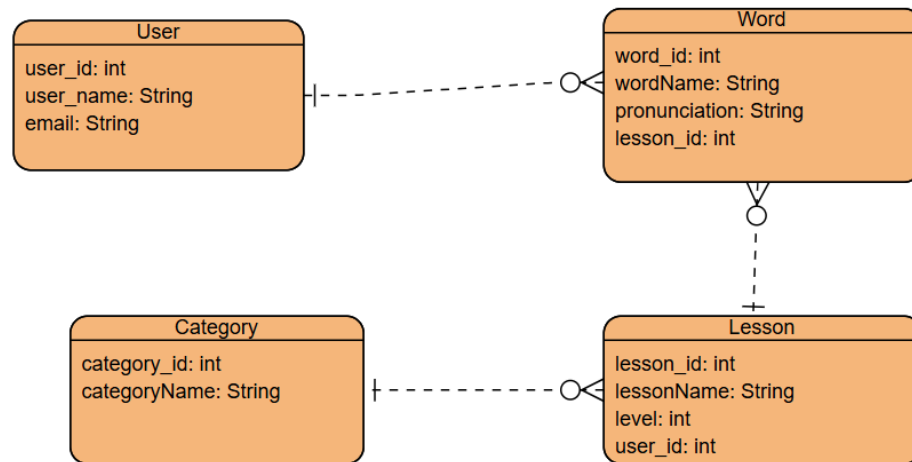
- Class Diagram



- Sequence Diagram



- Data Design
  - ERD or schema



#### 4.4: Teacher Edits Content

- User Interface Design

**Edit Content**

**Edit Existing Content**

Existing Content Title

Existing content description.

Choose File

No file chosen

Update

Cancel

- Front-end component(s)
  - Description and purpose:**
    - A user-friendly form that allows teachers to edit an existing word/phrase, pronunciation guide, and upload audio files.
    - The form is pre-populated with the existing content for easy modification.
    - Provides an option to replace or remove audio files.

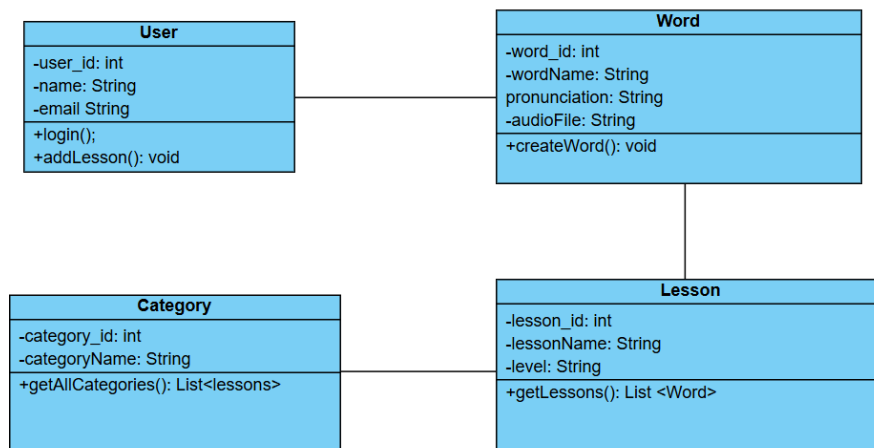
- **Component type:** Editable form interface

**format:**

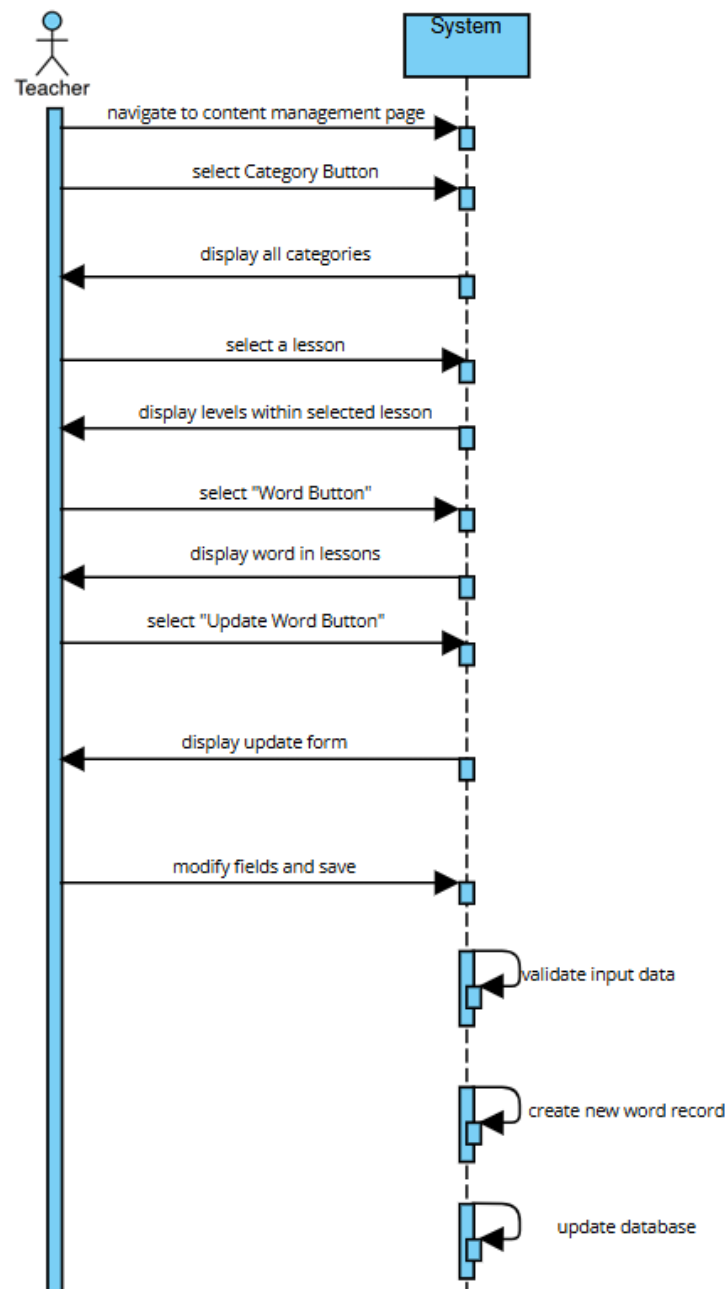
- **Input fields:** text fields for word/phrase and pronunciation guide.
- **File upload:** Supports MP3 for uploading new pronunciation audio.
- **Dropdown menu:** To change the associated lesson if needed.
- **Buttons:** "Save Changes" (updates content), "Remove Audio" (optional), and "Cancel."
- **Framework Used:** React.js or simple HTML/CSS/JS.
- Back-end component(s)
  - **Description and purpose:**
    - Manages content updates by modifying the word/phrase details in the database.
    - Allows updating or replacing the associated audio file.
    - Ensures validation (e.g., preventing duplicate entries).
  - **Component type:** API Endpoint

**or format:**

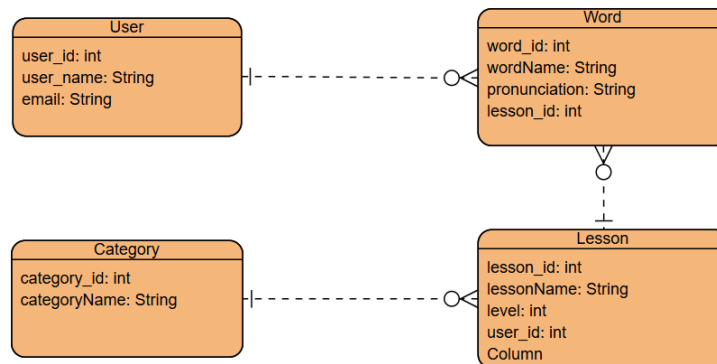
- **API Endpoint:** PUT /content/edit/{content\_id}
- Backend Language: React.js
- Database Interaction: MySQL/PostgreSQL/
- Middleware: Handles file uploads for replacing the audio file.
- Validation Logic: Ensures non-empty fields, valid file format, and proper lesson association.
- Object-Oriented Components
  - Class Diagram



- Sequence Diagram

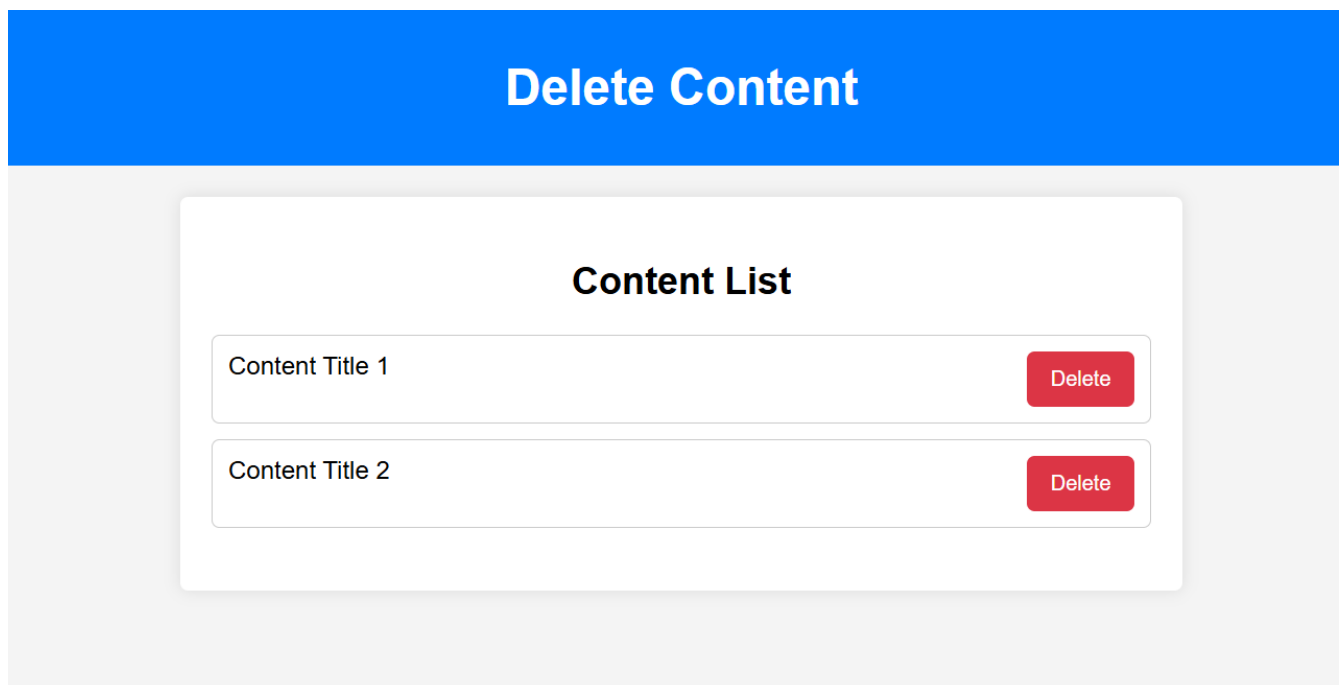


- Data Design
  - ERD or schema



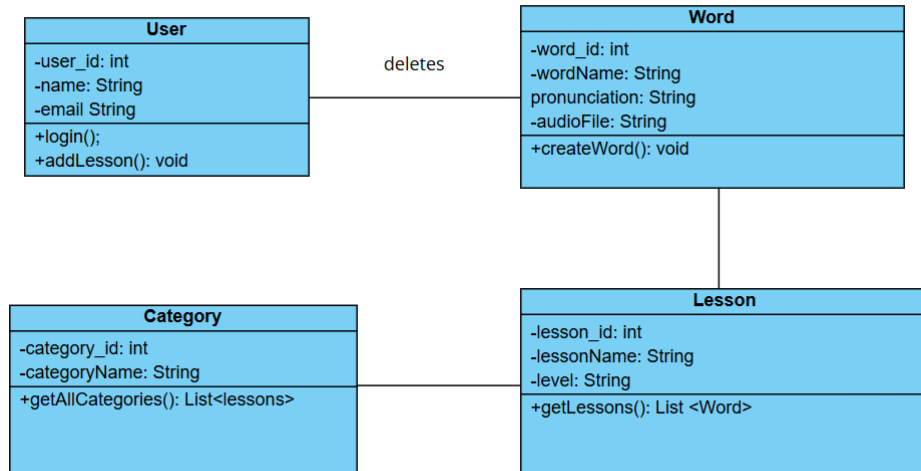
#### 4.5: Teacher Deletes Content

- User Interface Design



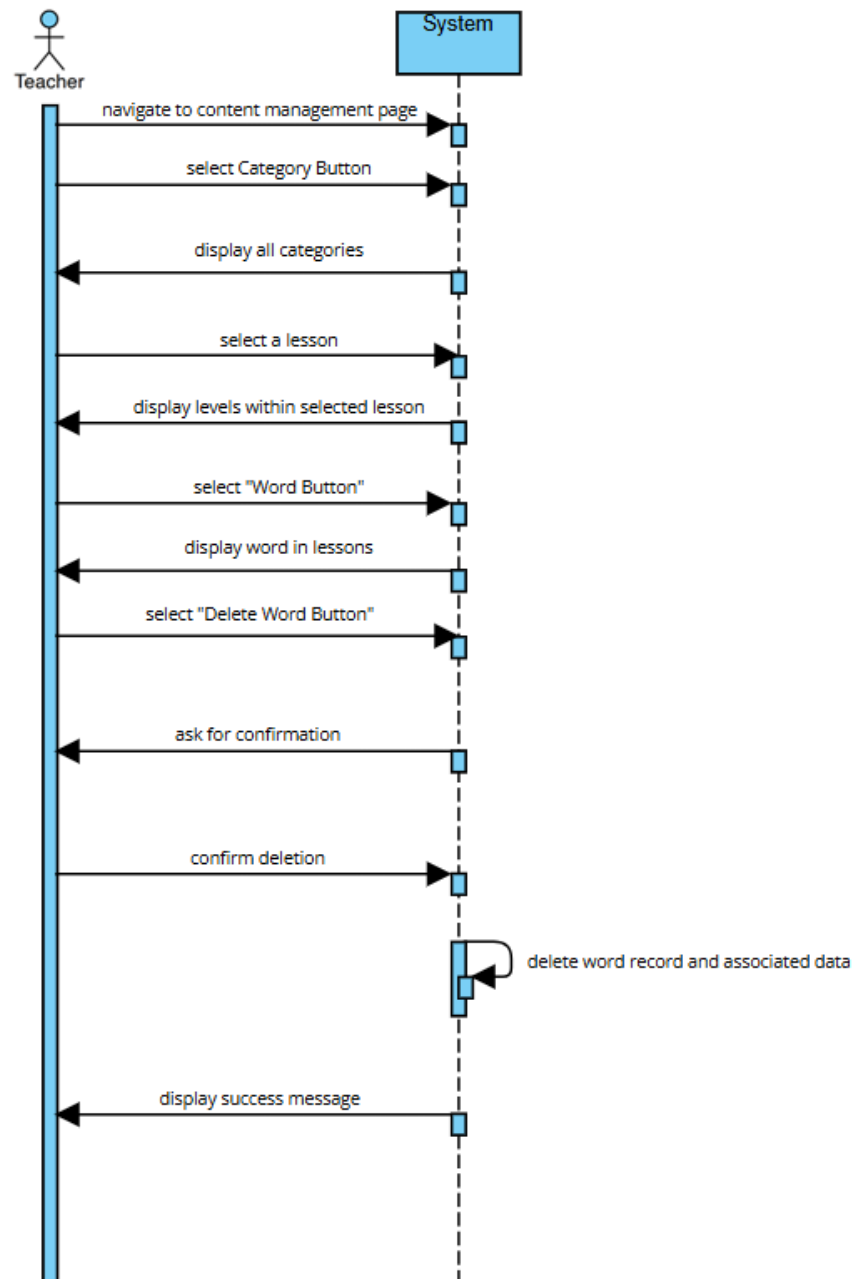
- Front-end component(s)
  - **Description and purpose:** Allows the teacher to delete a word along with its pronunciation guide and audio file by confirming the deletion.
  - **Component type or format:** Word list with delete buttons and a confirmation pop-up modal.

- Back-end component(s)
  - **Description and purpose:** Removes the word record from the database and deletes any associated files.
  - **Component type or format:** API request to delete the record and file from the server.
- Object-Oriented Components
  - Class Diagram

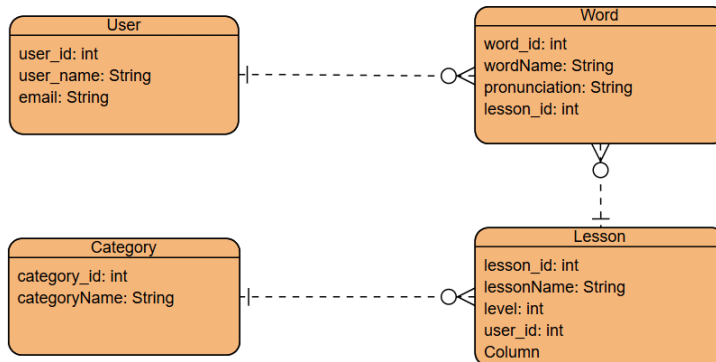


- Sequence Diagram





- Data Design
  - ERD or schema



#### 4.6: Teacher Creates Class

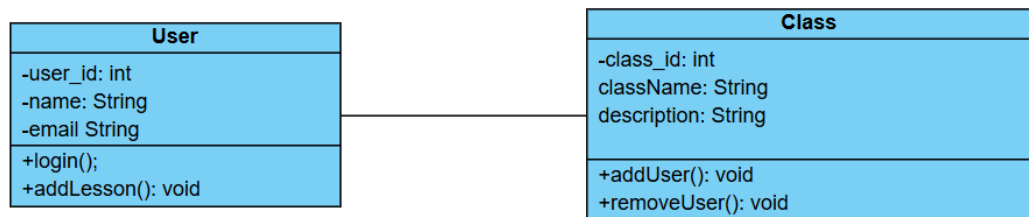
- User Interface Design

## Create Class

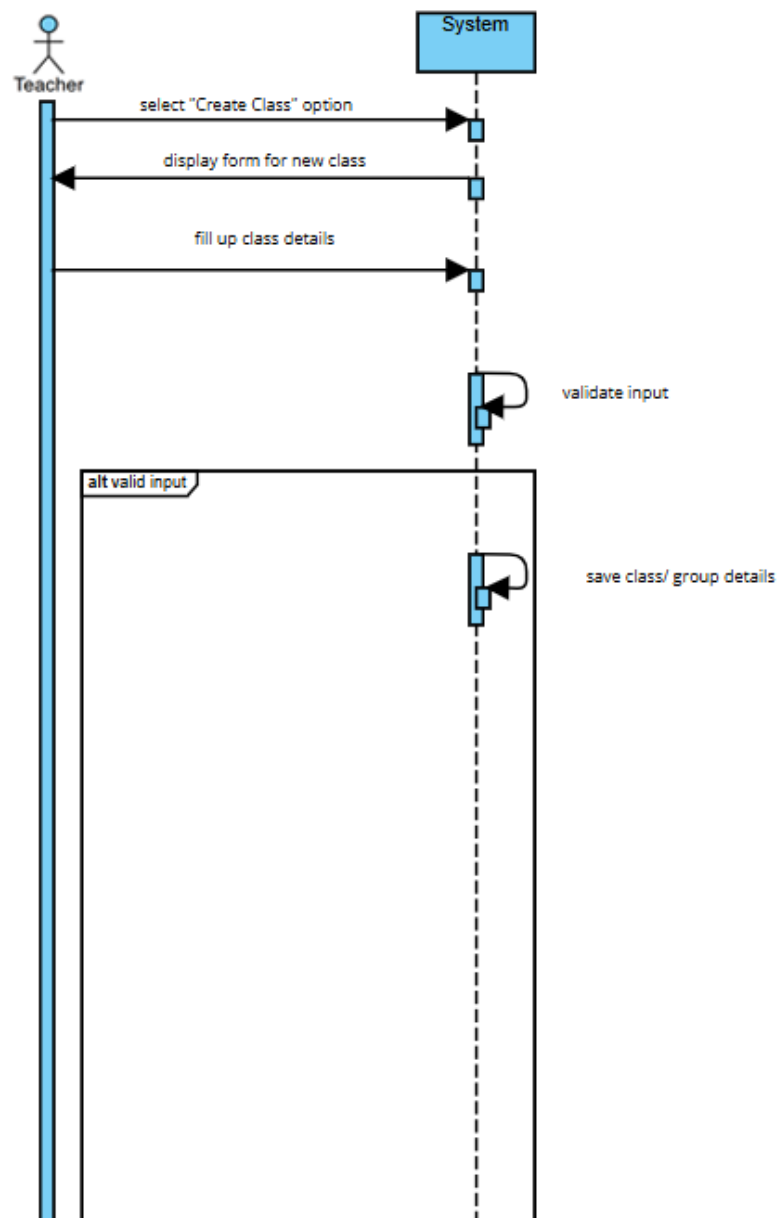
### New Class Details

- Front-end component(s)
  - **Description and purpose:** Allows the teacher to create a class by specifying details such as the class name and description.
  - **Component type or format:** form-based input with text fields and a submit button.
- Back-end component(s)

- **Description and purpose:** Stores the class details in the database and ensures it appears in the teacher's class list.
- **Component type or format:** API request to create a new class and update the database.
- Object-Oriented Components
  - Class Diagram



- Sequence Diagram



- Data Design
  - ERD or schema



#### 4.7: Teacher Assign User To Class

- User Interface Design

**Assign Student to Class**

**Student Name:**

Enter student name

**Select Class:**

Example ▼

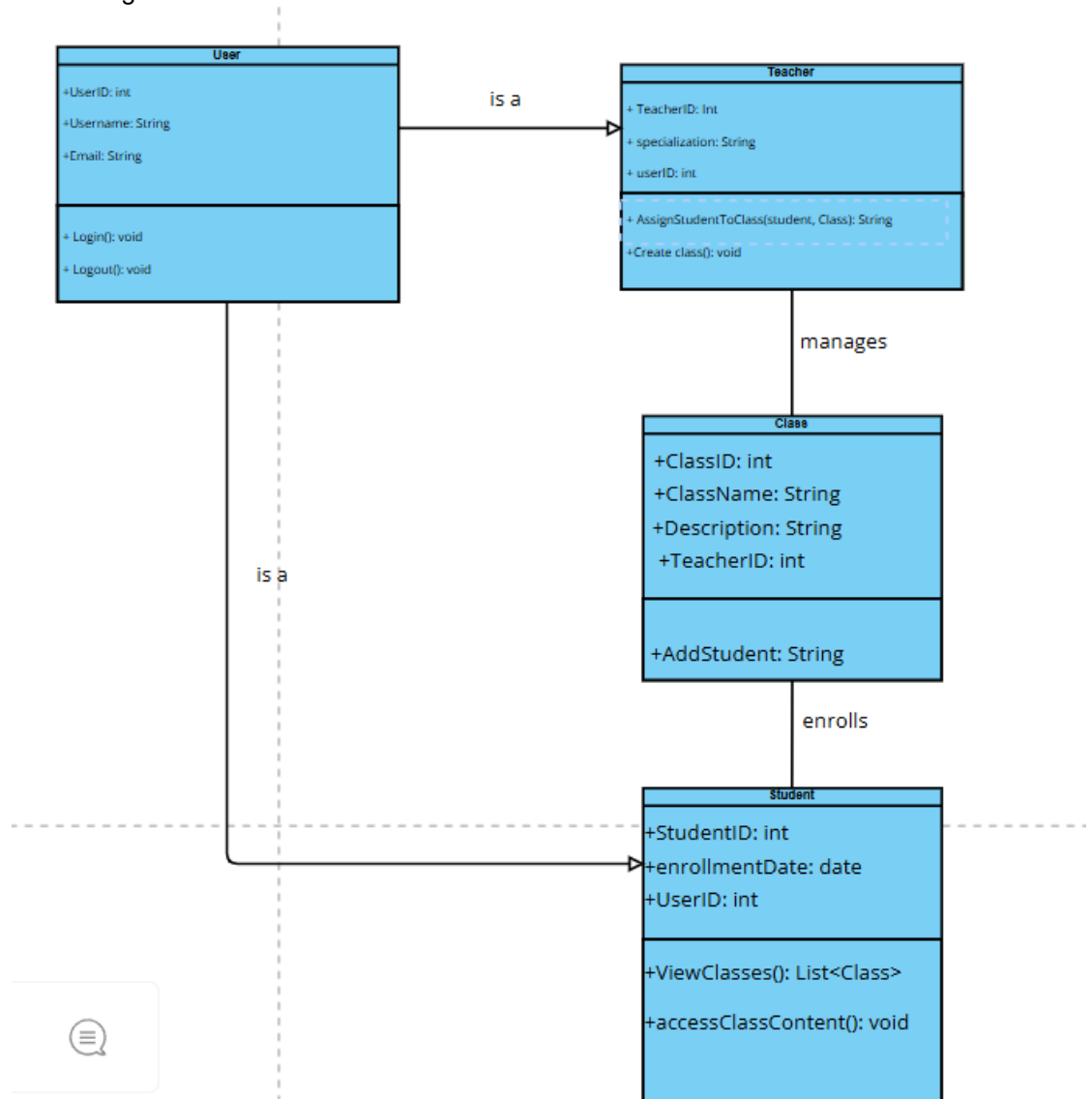
**Assign Student**

- Front-end component(s)
  - **Description and purpose:** Allows the teacher to assign students to a class by selecting a class and choosing users from a list.
  - **Component type or format:** dropdown or multi-select list for student selection, with an assign button.
- Back-end component(s)

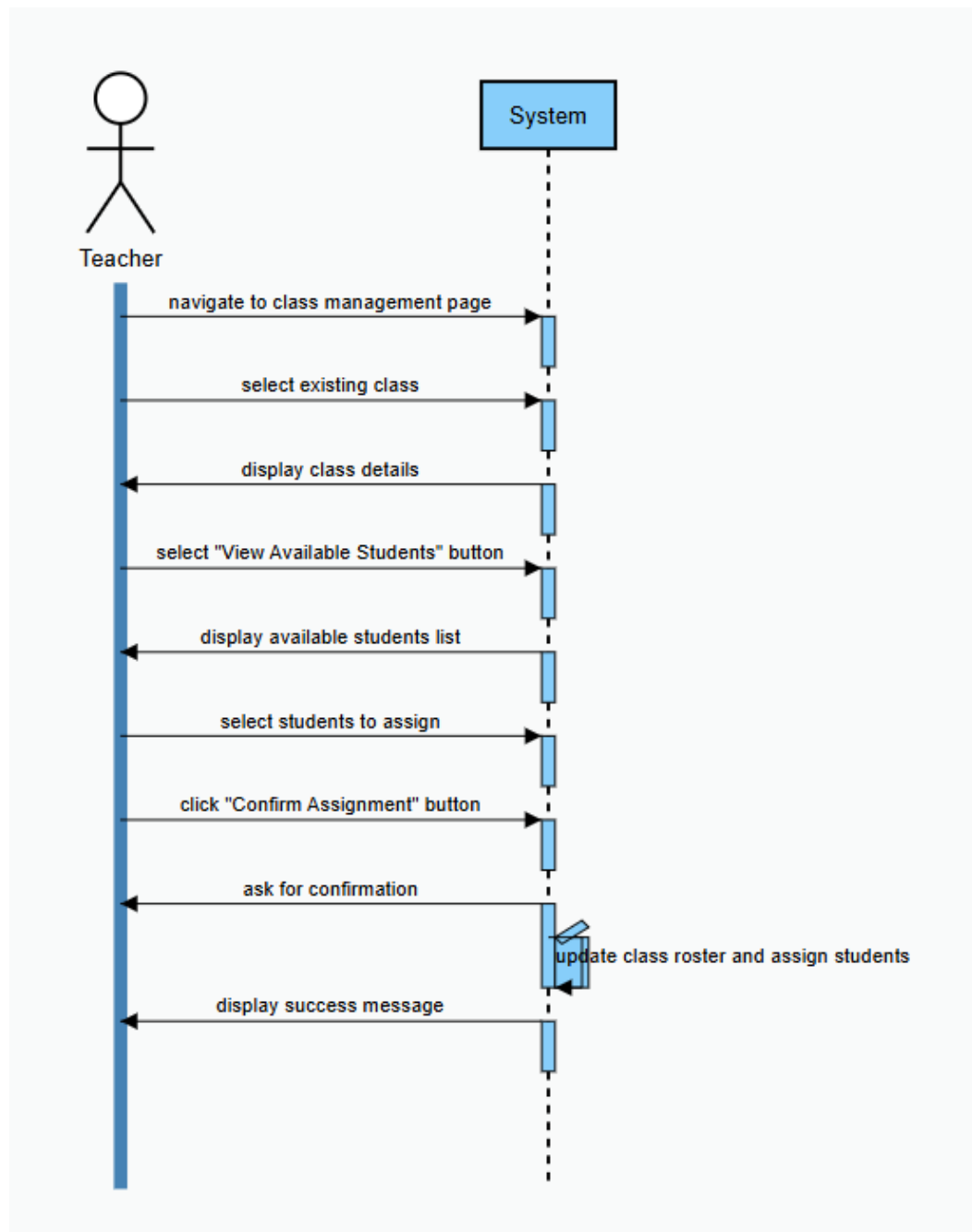
- **Description and purpose:** Link selected users to the chosen class and update the database accordingly.
- **Component type or format:** API request to update the user-class relationship in the database.

- Object-Oriented Components

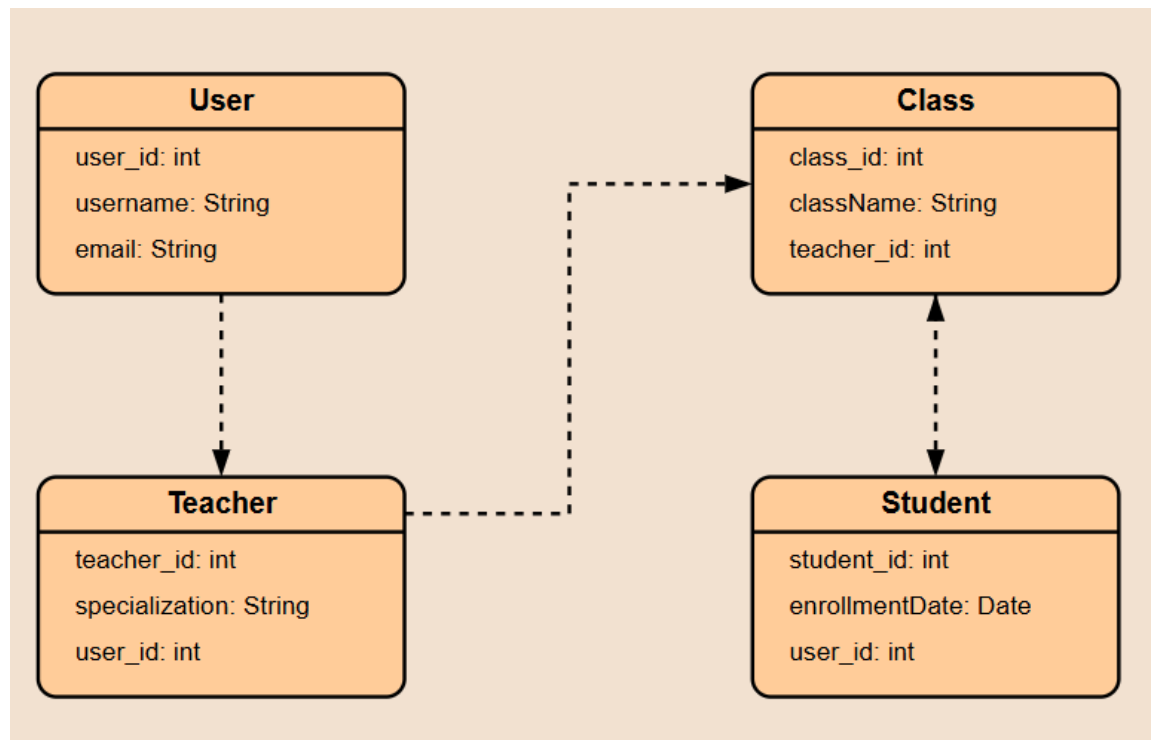
- Class Diagram



- Sequence Diagram



- Data Design
  - ERD or schema



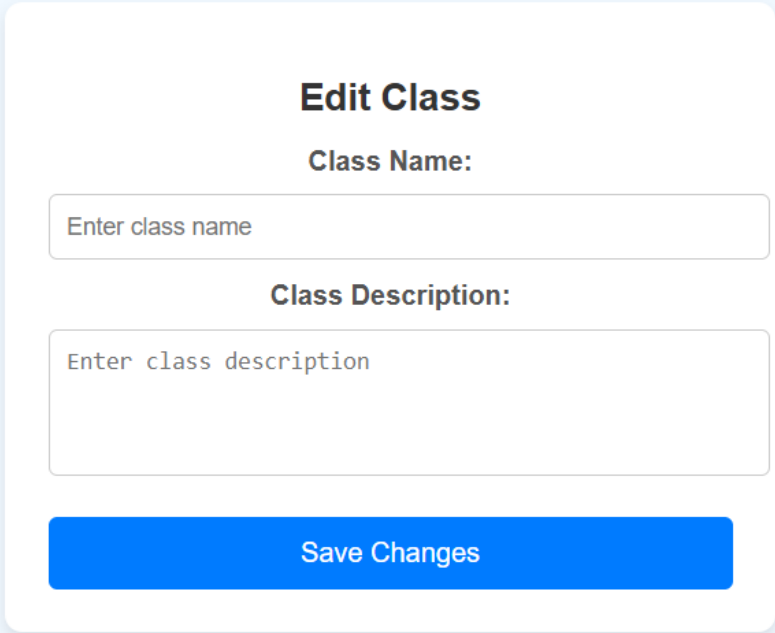
#### 4.8: Teacher Edits Class

- User Interface Design

### Manage Classes

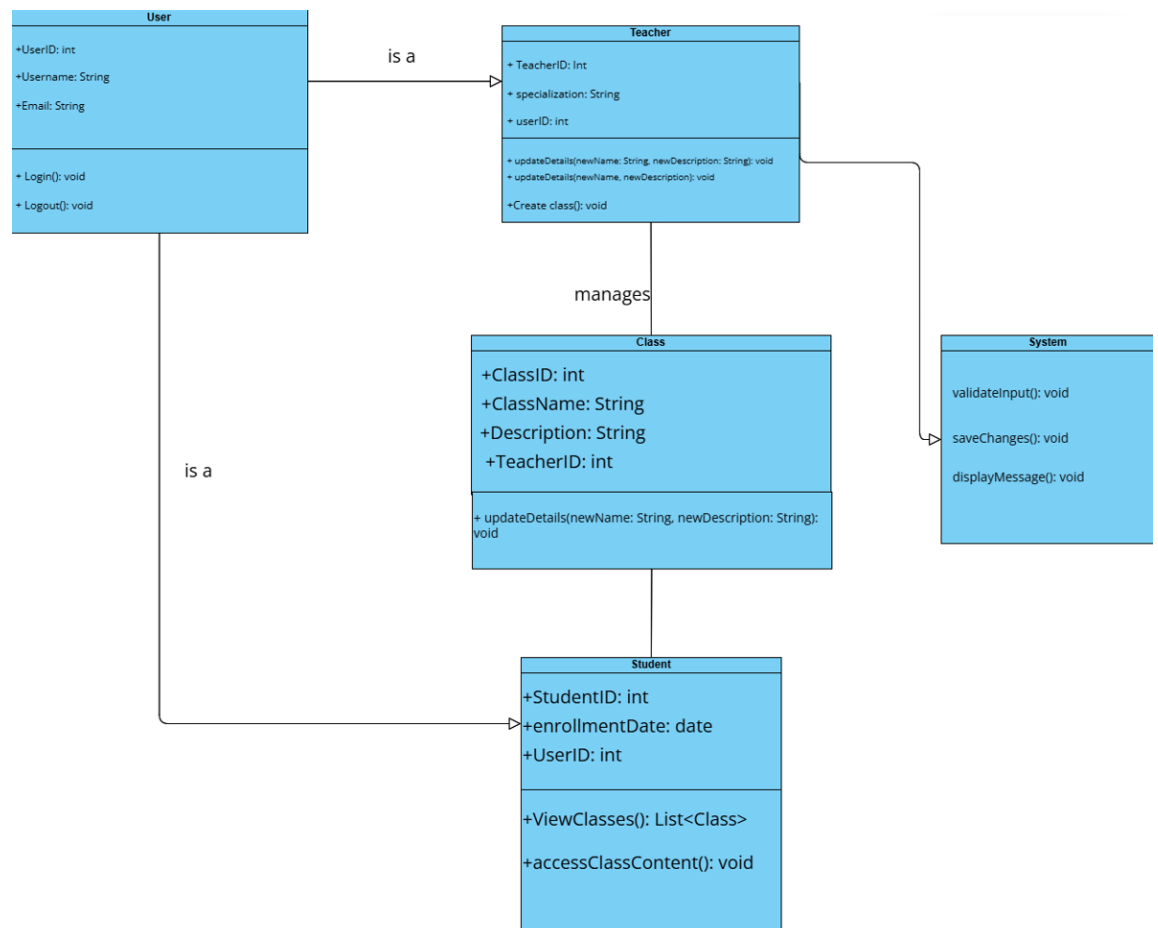
Example Class 1	Edit	Delete
Example Class 2	Edit	Delete
Example Class 3	Edit	Delete



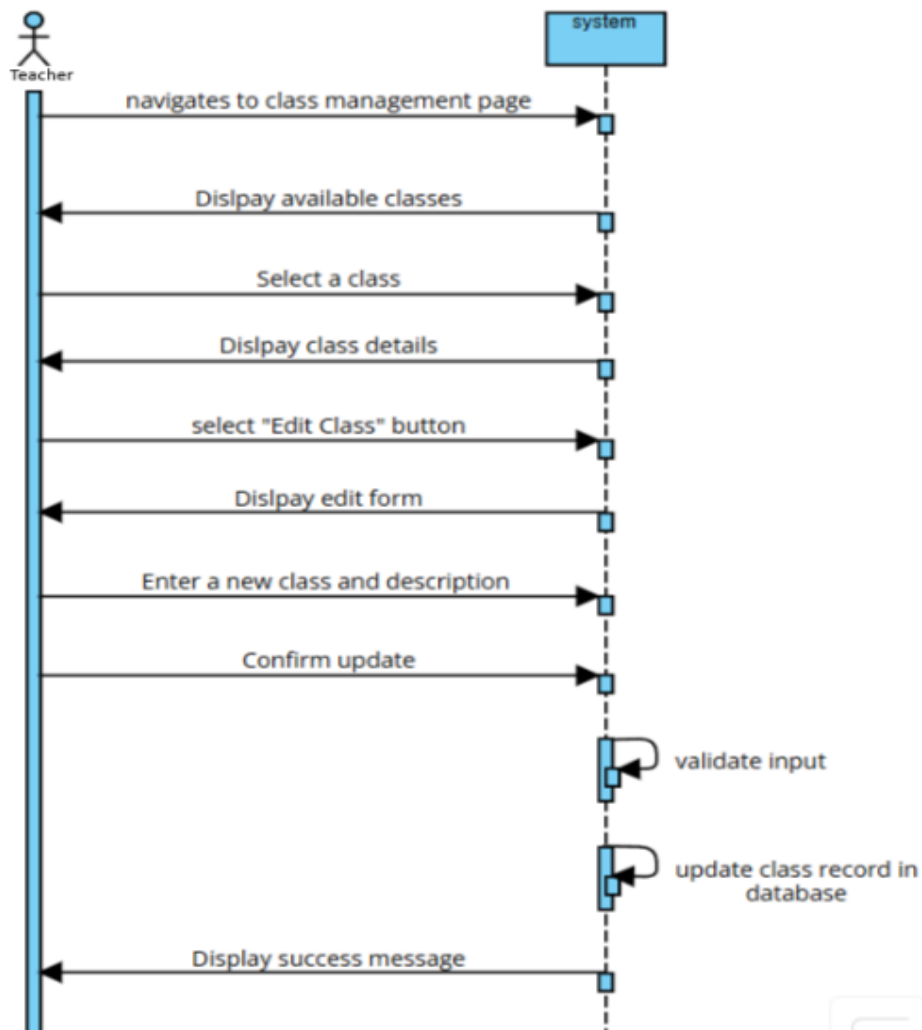


The image shows a web form titled "Edit Class". It is set against a light blue background. The form itself is white with rounded corners. At the top, the title "Edit Class" is centered in a bold, dark blue font. Below the title, the label "Class Name:" is centered. Underneath this label is a text input field with a light gray border and the placeholder text "Enter class name". Below the input field, the label "Class Description:" is centered. Underneath this label is a larger text area with a light gray border and the placeholder text "Enter class description". At the bottom of the form is a prominent blue button with the text "Save Changes" in white, centered.

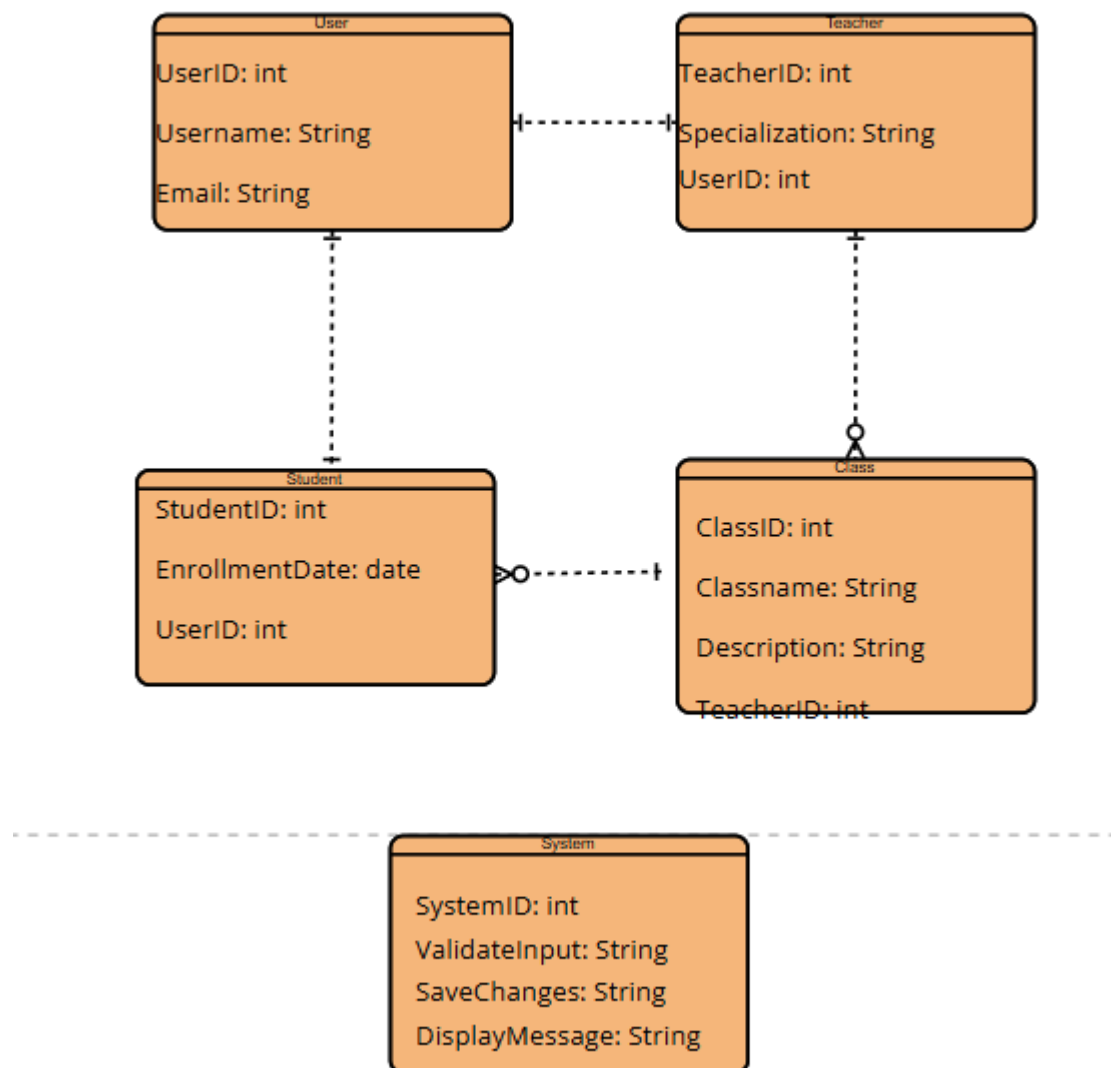
- Front-end component(s)
  - **Description and purpose:** Allows the teacher to modify class details such as the class name, description, or assigned students.
  - **Component type or format:** Edit form with prefilled values and update buttons.
- Back-end component(s)
  - **Description and purpose:** Updates class details in the database and ensures changes reflect in the teacher's dashboard.
  - **Component type or format:** API request to modify the class record in the database.
- Object-Oriented Components
  - Class Diagram



- Sequence Diagram

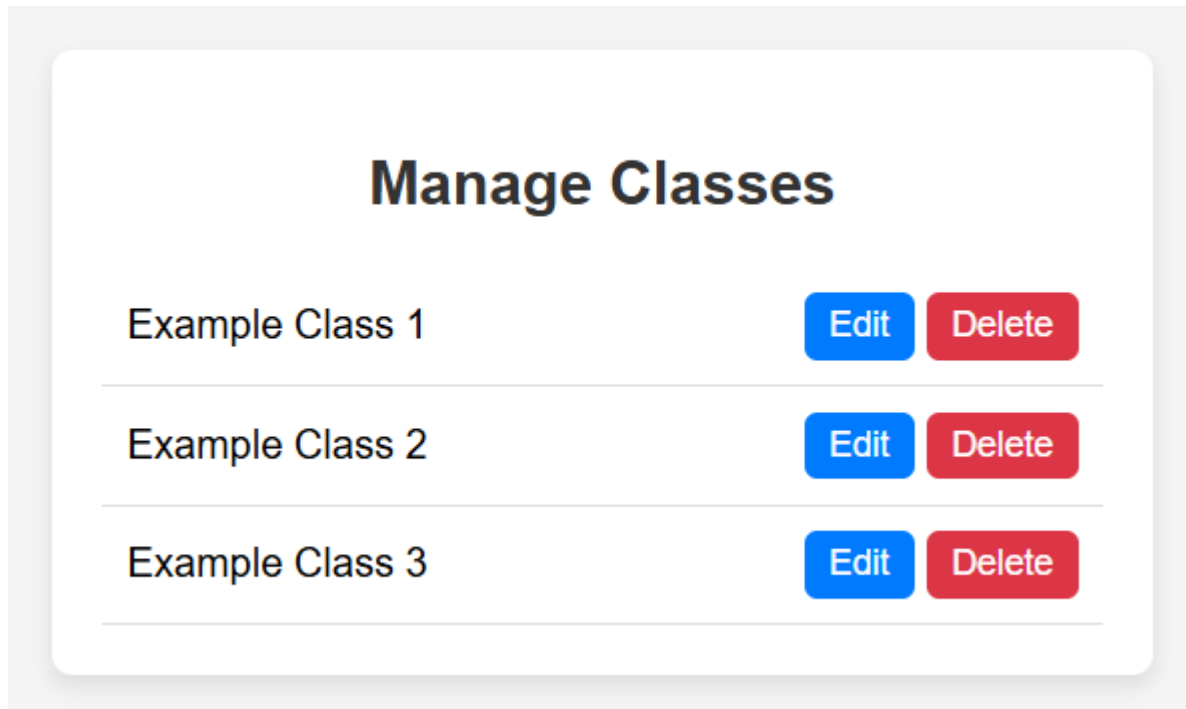


- Data Design
  - ERD or schema

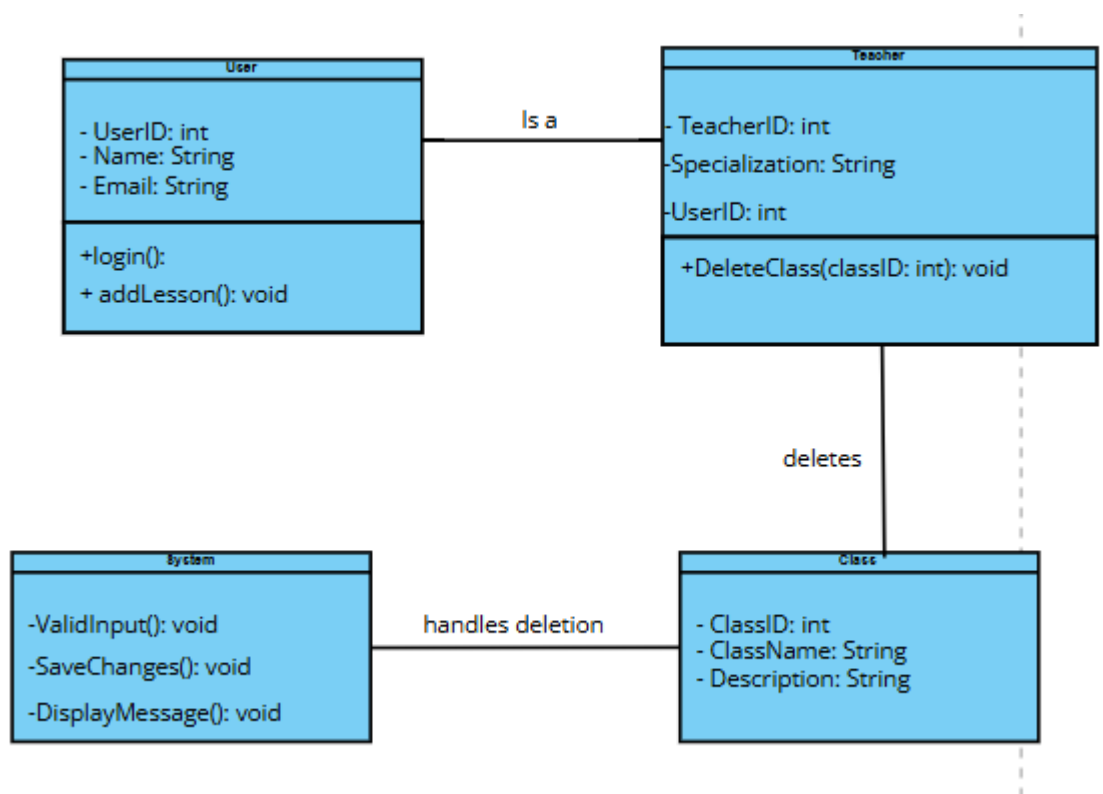


#### 4.9: Teacher Deletes Class

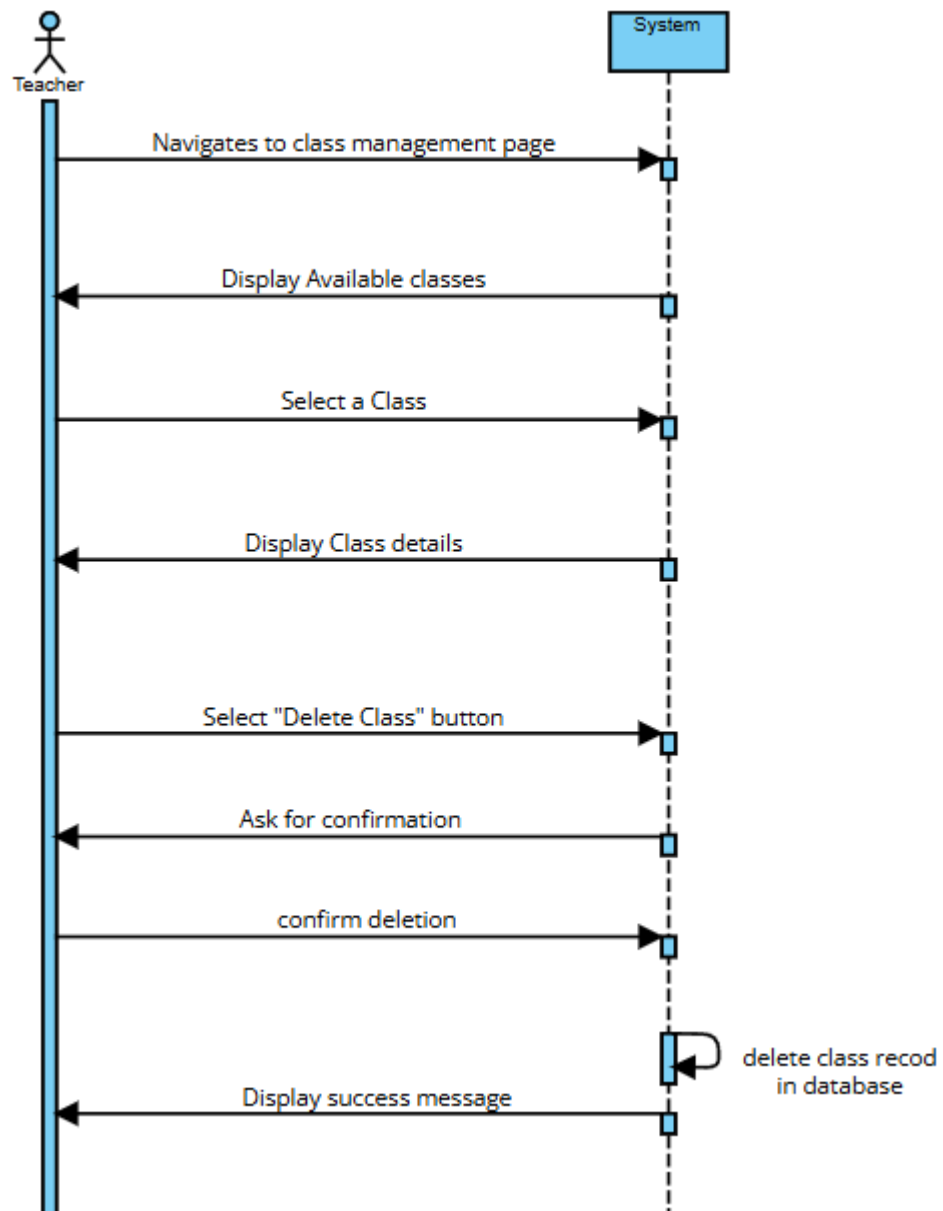
- User Interface Design



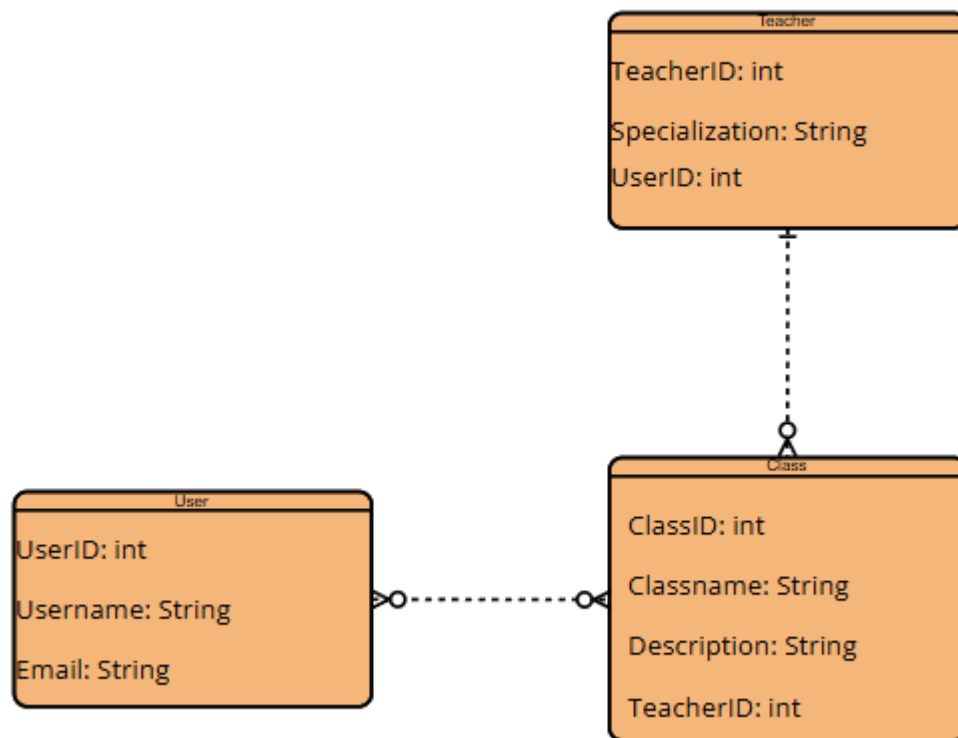
- Front-end component(s)
  - - **Description and purpose:** Allows the teacher to delete a class by selecting it and confirming the deletion.
    - **Component type or format:** Class list with delete buttons and a confirmation pop-up modal.
- Back-end component(s)
  - **Description and purpose:** Removes the class record from the database and unlinks any associated students.
  - **Component type or format:** API request to delete the class record and update related records.
- Object-Oriented Components
  - Class Diagram



- Sequence Diagram



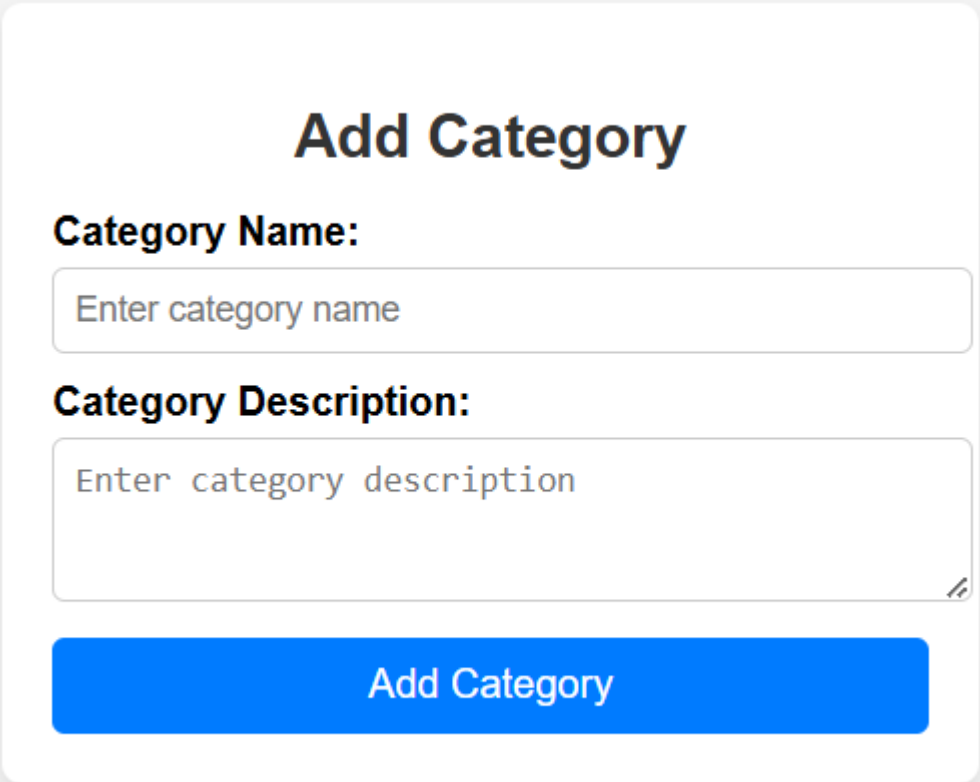
- Data Design
  - ERD or schema



#### 4.10: Teacher Adds Category

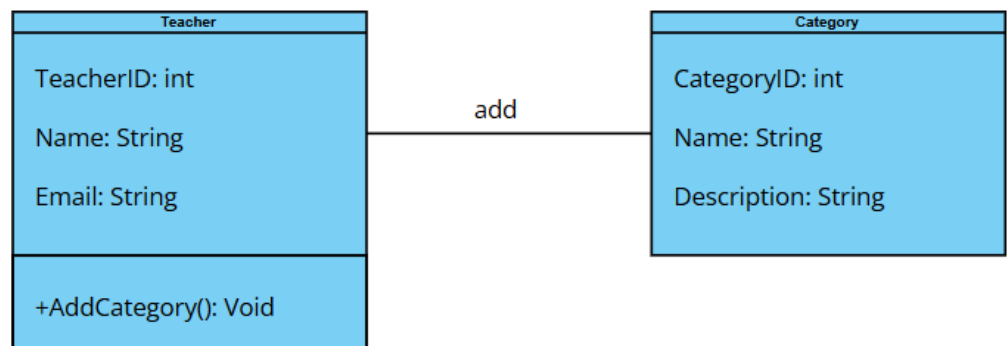
- User Interface Design



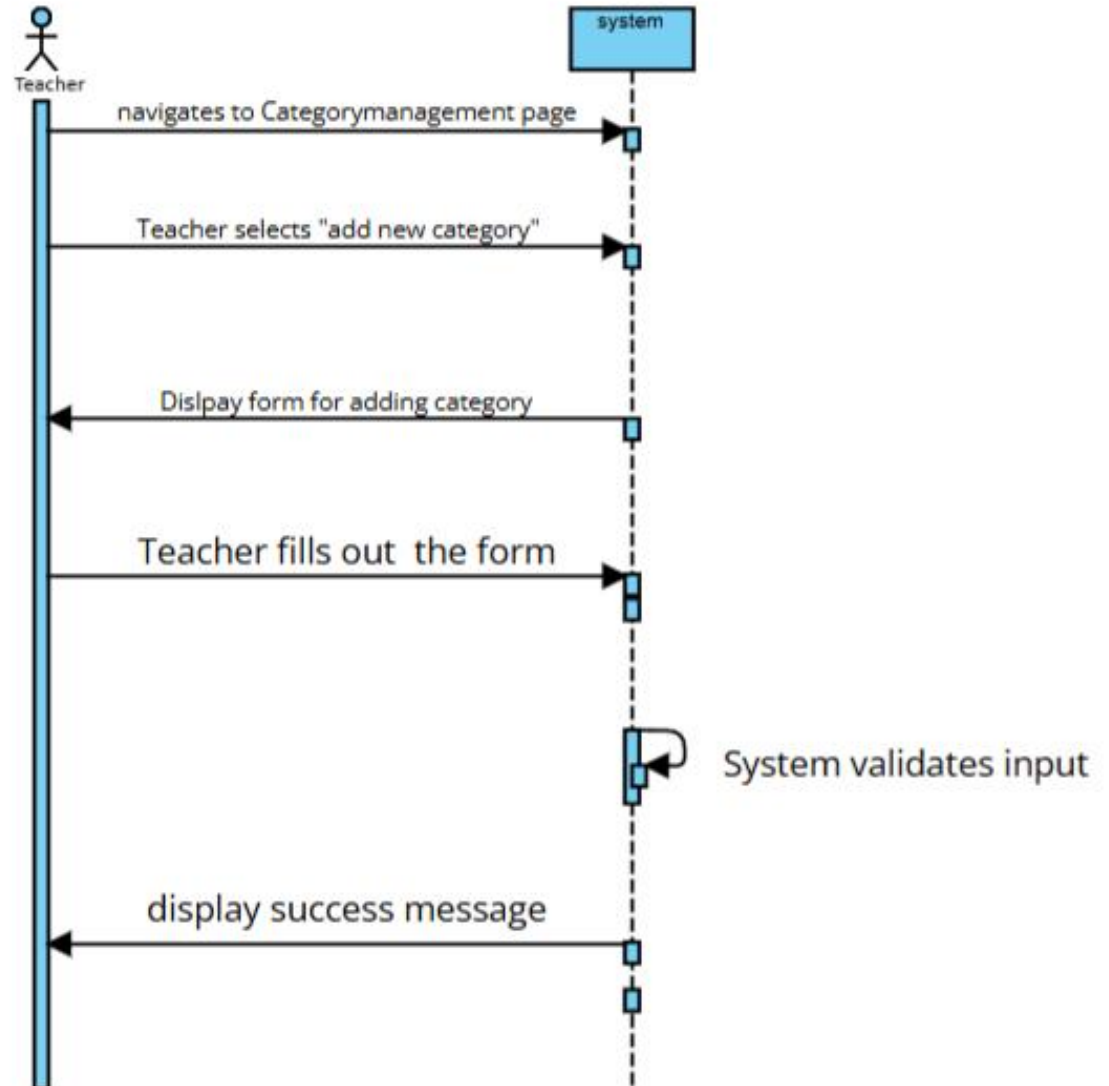


The image shows a web form titled "Add Category". It has a white background with rounded corners and a subtle shadow. The form contains two text input fields. The first field is labeled "Category Name:" and has a placeholder text "Enter category name". The second field is labeled "Category Description:" and has a placeholder text "Enter category description". Below the description field is a blue button with the text "Add Category".

- Front-end component(s)
  - **Description and purpose:** Allows the teacher to create a new vocabulary category by entering a name and submitting the form.
  - **Component type or format:** form-based input with a text field and submit button.
- Back-end component(s)
  - **Description and purpose:** validate and store the new category in the database for lesson and content organization.
  - **Component type or format:** API request to insert a new category into the database.
- Object-Oriented Components
  - Class Diagram



- Sequence Diagram

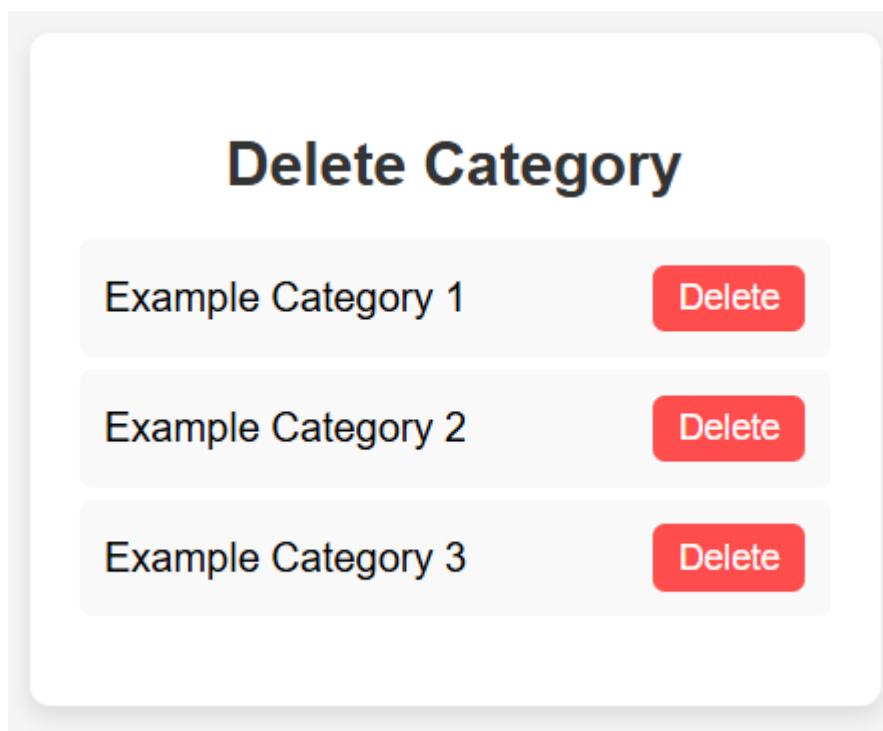


- Data Design
  - ERD or schema



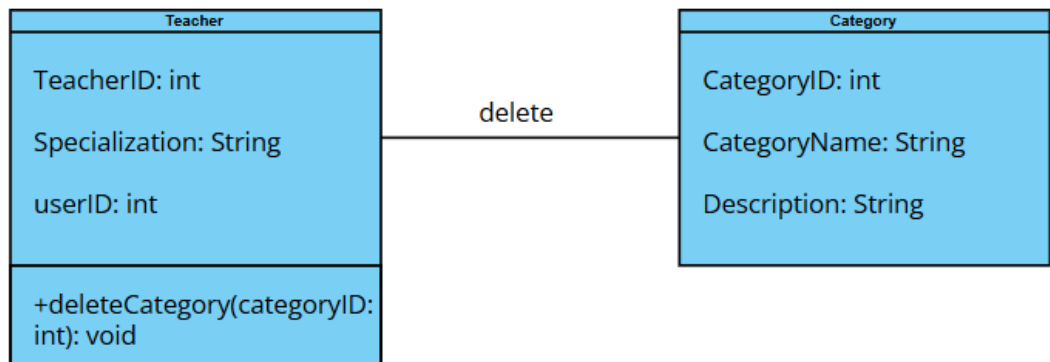
#### 4.11: Teacher Deletes Category

- User Interface Design

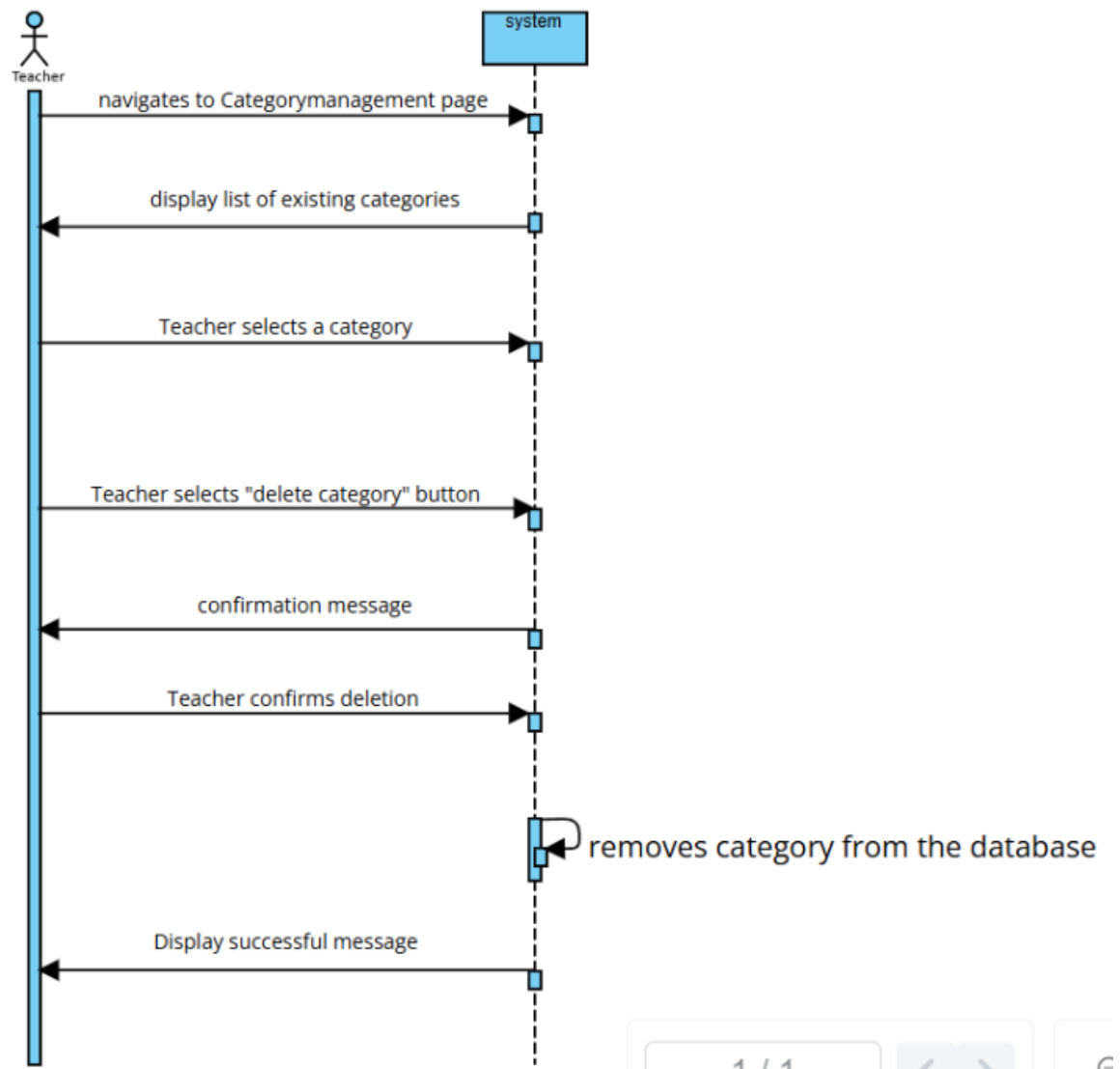


- Front-end component(s)
  - - **Description and purpose:** Allows the teacher to delete a category by selecting it from the category list and confirming the action.
    - **Component type or format:** category list with delete buttons and a confirmation pop-up modal.
- Back-end component(s)

- **Description and purpose:** Removes the category from the database and ensures that related records remain consistent.
- **Component type or format:** API request to delete the category record from the database.
- Object-Oriented Components
  - Class Diagram



- Sequence Diagram

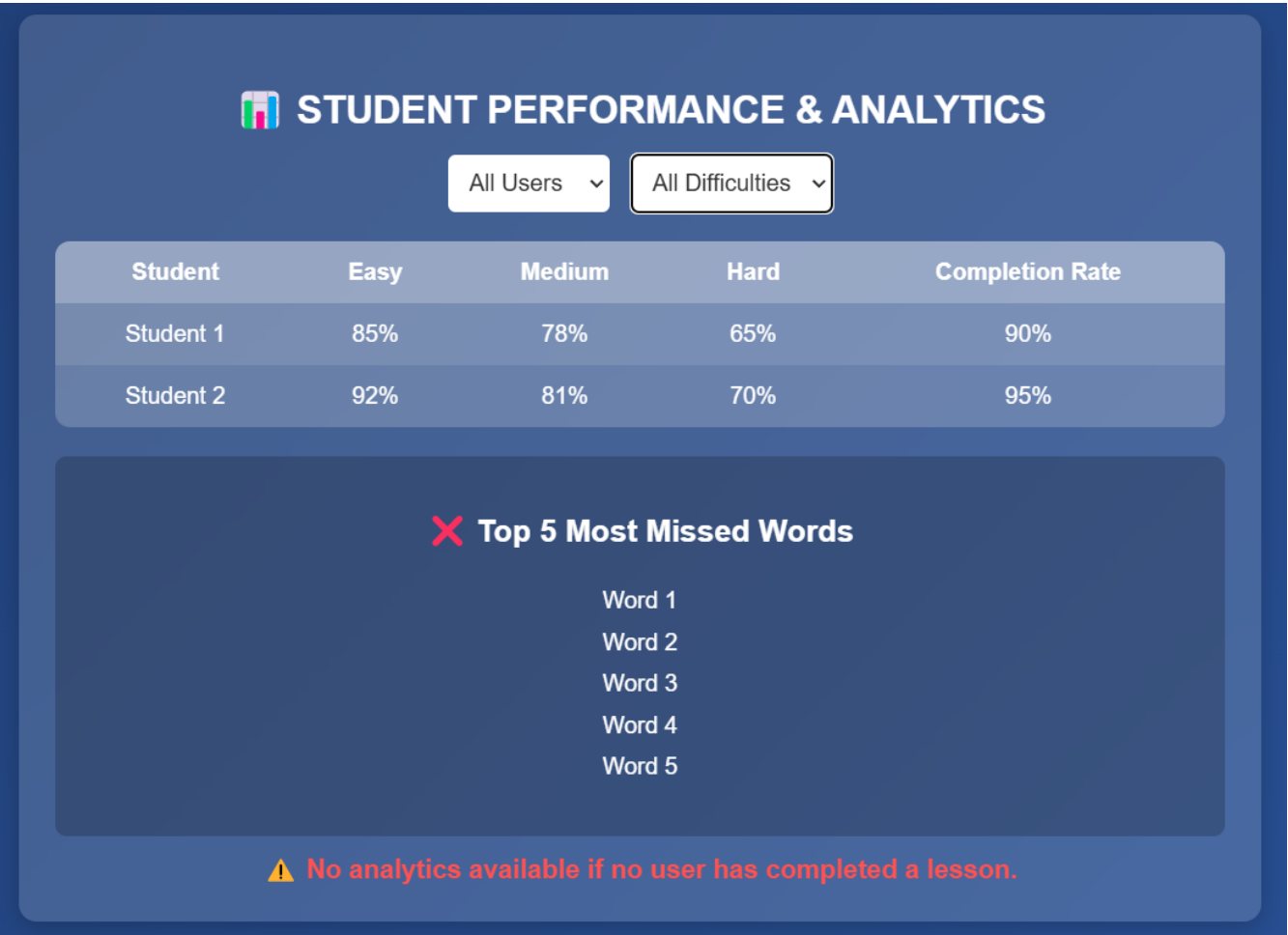


- Data Design
  - ERD or schema



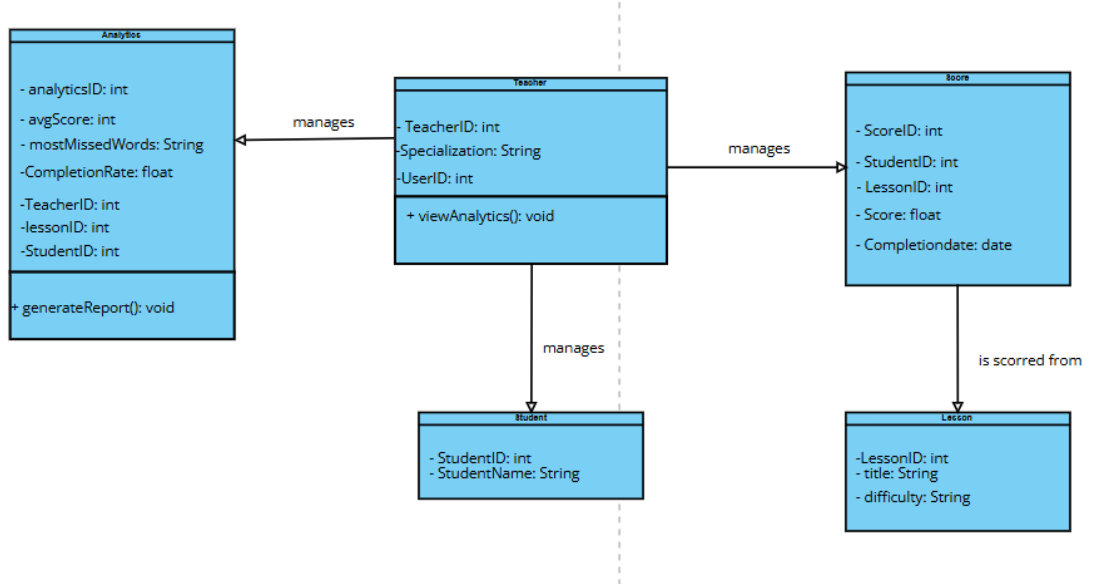
4.12: Teacher Tracks Scores and Data Analytics

- User Interface Design



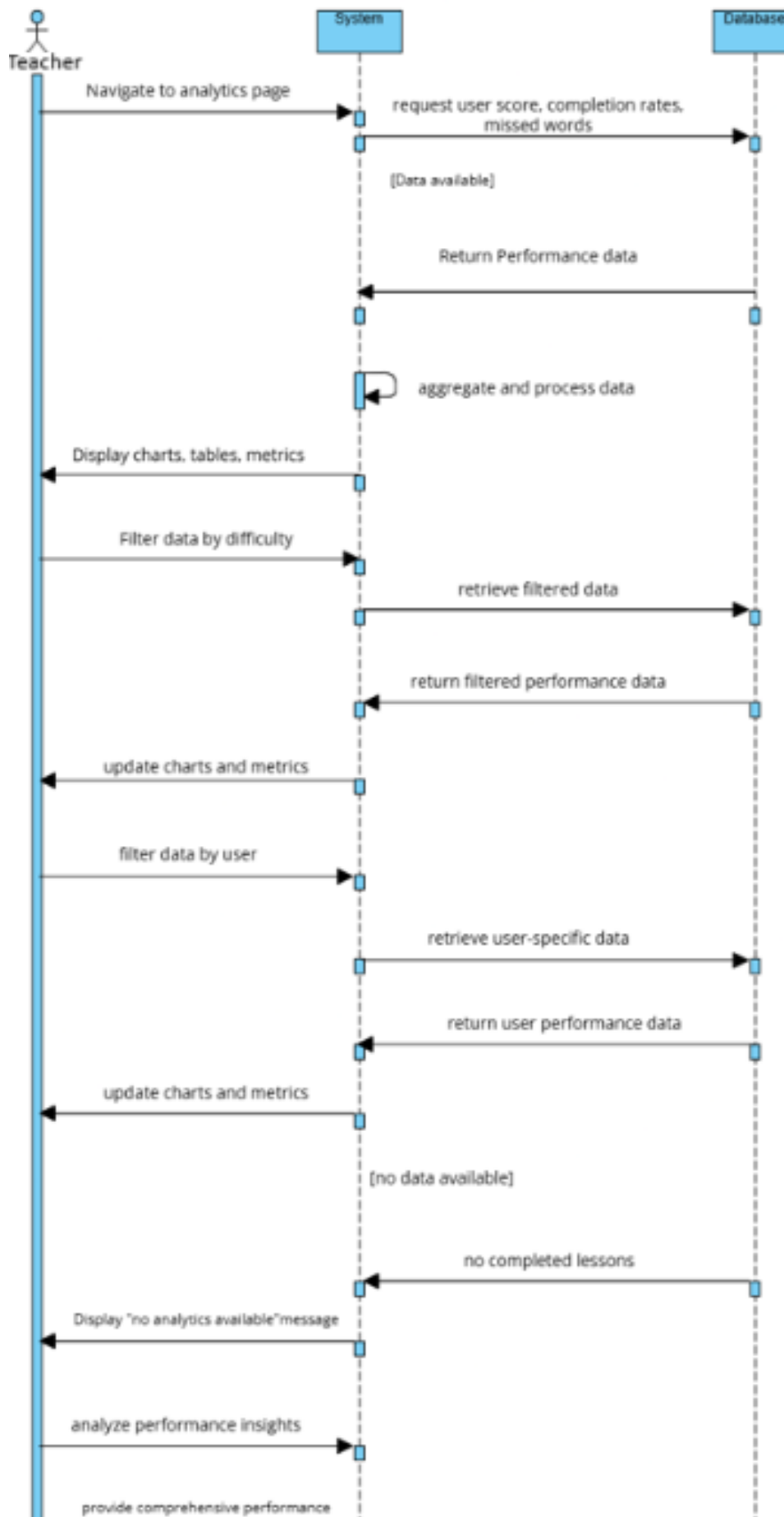
- Front-end component(s)
  - - **Description and purpose:** Displays a dashboard where the teacher can review student performance, including scores and commonly missed words.
    - **Component type or format:** graphic dashboard with charts, tables, and filters for data analysis.
- Back-end component(s)
  - **Description and purpose:** Retrieves and processes student performance data, generating reports and visual insights.
  - **Component type or format:** API that fetches analytics data from the database and formats it for visualization.
- Object-Oriented Components

○ Class Diagram



○ Sequence Diagram





- Data Design
  - ERD or schema

