1. Project Overview

1.1 Introduction

The "SmartTot" online quiz application represents a significant leap forward in educational technology, offering a dynamic, interactive platform specifically designed for children's learning through quizzes. This application is grounded in the principles of engaging educational experiences and promoting learning through a wide array of quiz categories tailored to various educational subjects and age groups.

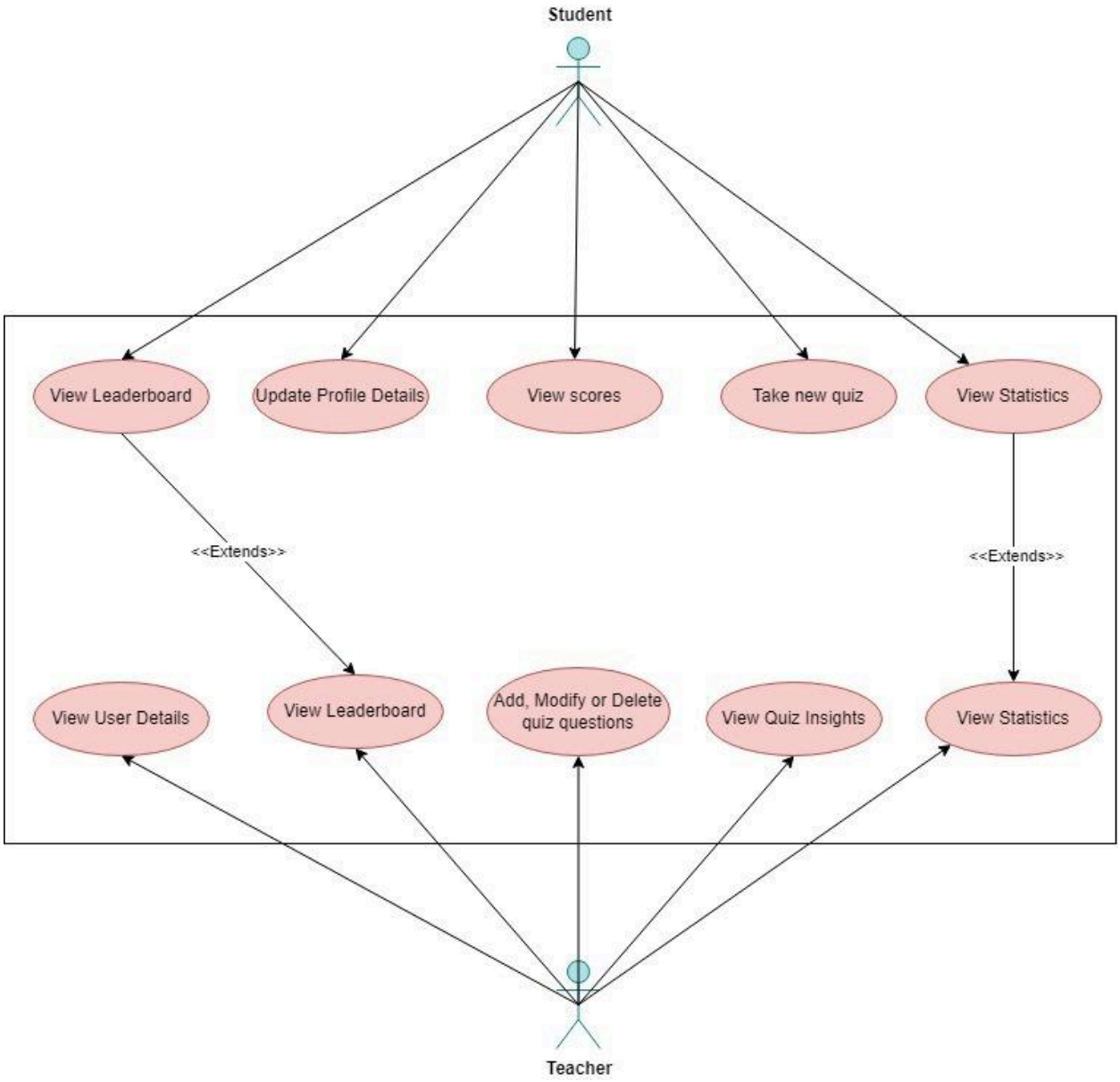
1.2 Objective

The primary objective of SmartTot is to facilitate an accessible, enriching educational experience that motivates and challenges young learners. By integrating technology with traditional learning methods, SmartTot aims to enhance the efficiency of learning assessments, provide immediate feedback, and foster a love for learning among children.

1.3 Target Audience

SmartTot is meticulously designed for children of school-going age, offering a safe, stimulating, and user-friendly environment for learning. Secondary users of the application include educators and parents who seek to monitor progress, administer quizzes, and contribute to the educational content.

Use Case Diagram:



User Stories:

1. **Add Questions**:

A teacher should be able to add questions to quizzes of different categories so that they can evaluate the knowledge of their students.

Pre-condition: Logged in as a teacher

Post-condition: The questions can be added to the categories

2. **Edit/Delete Questions:**

A teacher should be able to edit or delete the questions related to particular category.

Pre-condition: Logged in as a teacher

Post-condition: Quiz can be updated or deleted

3. **Take Insights:**

A teacher should be able to view the insights of the quiz.

Pre-condition: Logged in as a teacher

Post-condition: Access the insights of the quiz

4. **View students**

As an teacher or teacher, I want to view the students.

Pre-condition: Logged in as a teacher

Post-condition: Access the student details page

5. **View Leaderboard**

A teacher should be able to view the Leader board.

Pre-condition: Logged in as a teacher

Post-condition: Access the Leaderboard page

6. **Teacher Logout**

A teacher should be able to logout from the application any time after login.

Pre-condition: Logged in as a teacher

Post-condition: Logout the teacher

7. **Attempt Quiz**

A student should be able to attempt the quiz

Pre-condition: Logged in as a student.

Post-condition: Choose specific quiz under a subject in which you want to attempt

8. **View Score**
   A student should be able to view his/her score
   Pre-condition: Logged in as a student
   Post-condition: View the score of that particular quiz

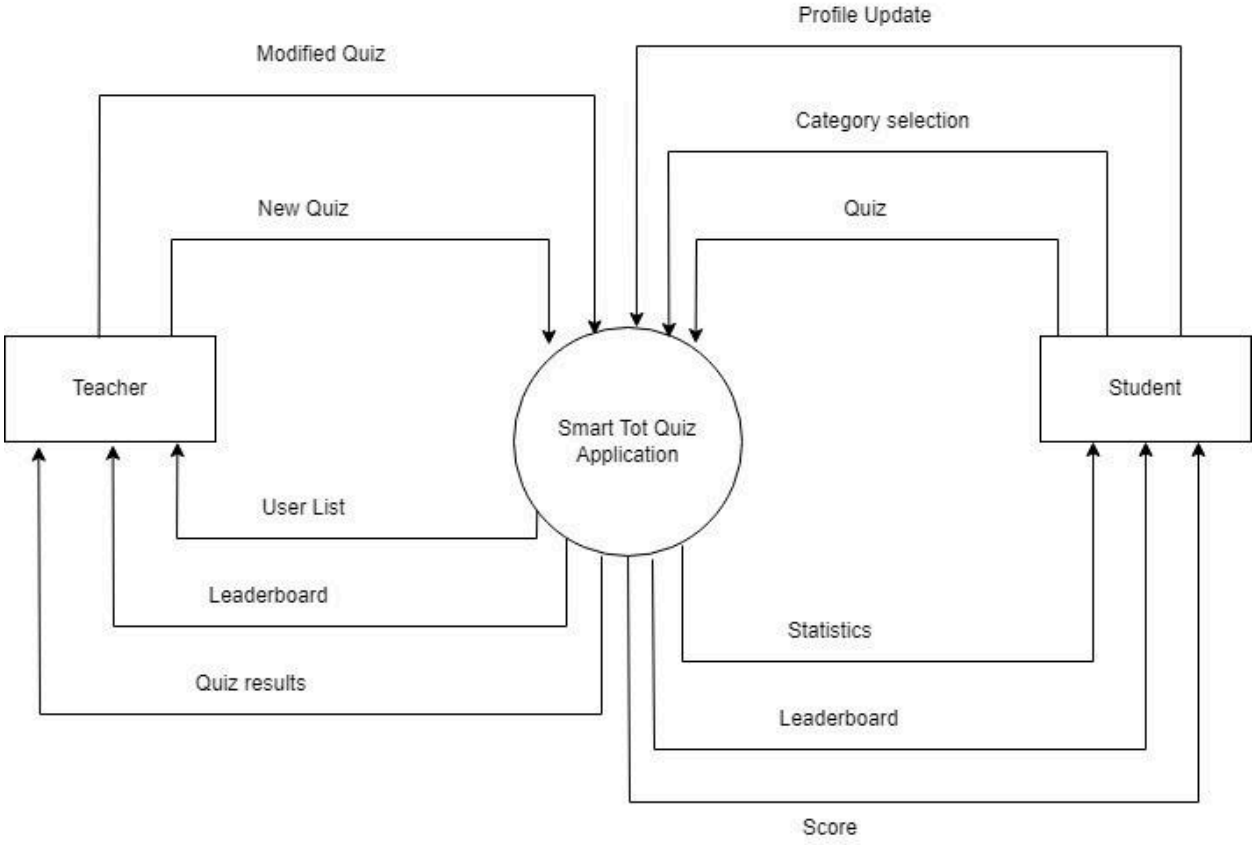
9. **View Leaderboard**
   A student should be able to view the Leader board.
   Pre-condition: Logged in as a teacher
   Post-condition: Access the Leaderboard page

10. **View statistics**
    A student should be able to view his/her statistics
    Pre-condition: Logged in as a student.
    Post-condition: Access the statistics page

11. **Update Profile**
    A student should be able to update their profile
    Pre-condition: Logged in as a student.
    Post-condition: Access the profile page

12. **User Logout**
    As a student, I want to logout from the application any time after login.
    Pre-condition: Logged in as a student
    Post-condition: Logout the student

Context Diagram:



2. Architectural Overview

2.1 Subsystem Architecture

SmartTot adopts a modular architecture that separates the user interface, business logic, and data storage layers to ensure scalability, maintainability, and ease of development.

- **Frontend**: Built with HTML 5.0, CSS W3, and JavaScript ES2015, the frontend is designed for cross-platform compatibility, ensuring a responsive and engaging user experience across devices. The incorporation of jQuery enhances dynamic content rendering and interaction, providing a seamless user experience.
- **Backend**: Utilizes JSON-server, a minimalist Node.js package, to simulate a RESTful API environment. This choice allows for rapid development and testing, with db.json serving as the mock database for storing and retrieving data.

2.2 Deployment Architecture

The application is designed for easy deployment, both for development purposes and scalable production use. During development, Visual Studio Code's Live Server extension and JSON server facilitate a live-reload environment for immediate feedback on changes.

The SmartTot quiz application embraces a deployment architecture that caters to both development agility and production robustness. This two-pronged approach ensures developers can iterate rapidly, while also guaranteeing that the application scales efficiently and remains stable in a live environment.

Development Deployment

During the development phase, the application is served using Visual Studio Code's Live Server extension, which offers a real-time development environment by automatically reloading the web page as changes are made to the code. This immediate feedback loop accelerates front-end development and allows for quick visual and functional checks.

The backend utilizes the JSON server, a full fake REST API with zero coding in less than 30 seconds, making it an excellent choice for front-end developers who need a quick back-end for prototyping and mocking. The server reads from a db.json file, allowing the application to simulate CRUD operations against a live data store without the overhead of setting up a database

server. This facilitates a smooth and rapid development process, enabling the developers to focus on building out features and testing functionality with a realistic data layer.

To set up this environment, developers need to install the JSON server as a Node.js package and run it so that it watches for any changes to the db.json file, reflecting updates in real time.

Production Deployment

For production, the application is packaged and deployed to a server environment that supports Node.js applications. The static files generated for the front end are served through a web server, like NGINX or Apache, configured to deliver the application's HTML, CSS, and JavaScript assets efficiently.

The JSON server is replaced with a more robust and persistent database solution suited for production use. This transition ensures that the application can handle the scale of production traffic, maintain data integrity, and provide continuity of service. The production database is set up to handle concurrency, backups, and data recovery, with security measures in place to protect sensitive information.

The application's server-side logic, which was prototyped using a JSON server, is migrated to a full-fledged Node.js server, such as Express.js, which serves the RESTful API endpoints. The Express.js server is configured to interface with the production database, manage user sessions, and handle business logic.

Load balancers are employed to distribute traffic across multiple server instances, providing high availability and redundancy. Continuous integration and continuous deployment (CI/CD) pipelines automate the testing and deployment process, ensuring that each release is stable and that the deployment is seamless.

This deployment architecture underscores the SmartTot application's ability to adapt to the needs of a growing user base while providing a stable, secure, and responsive platform for children's education.

2.3 Persistent Data Storage

The SmartTot quiz system leverages a lightweight and efficient data storage strategy by using a db.json file. This file serves as a mock database during the development phase and stands in for more complex database systems without the need for immediate setup or maintenance.

Data Structure: The db.json file encapsulates the necessary data structures, containing collections for quizzes, questions, user profiles, and performance statistics. Each collection is

accessed via a unique endpoint provided by the JSON server, simulating typical database operations such as create, read, update, and delete (CRUD).

Simplification and Portability: By adopting a JSON-based storage system, the initial project setup is significantly simplified. This format allows for easy data manipulation and quick modifications, providing a flexible and portable means of data storage. Developers can swiftly copy, back up, or share the database file, enabling a seamless migration and version control process.

Transition to Production: For the production environment, the structure laid out in db.json can be directly translated into schemas for a full-scale database. This ensures that the transition from development to production is smooth, maintaining the integrity and format of the stored data.

2.4 Global Control Flow

In SmartTot, the global control flow is meticulously architected to provide a cohesive and responsive user experience. The application employs AJAX (Asynchronous JavaScript and XML) to handle data interactions seamlessly between the client-side front end and the server-side back end.

Asynchronous Operations: AJAX techniques are used to make asynchronous calls to the JSON server, enabling the web page to update dynamically without the need to reload. This results in a more interactive experience for end-users, as they can take quizzes, view results, and interact with content without interruption or delay.

Functional Interactivity: The control flow is designed to support a wide array of functionalities, including:

> **Quiz Participation**: Students interact with the quiz interface, selecting answers and submitting their responses. The system fetches questions and records answers without page refreshes, promoting a fluid quiz-taking environment.
>
> **Results Feedback**: After quiz submission, the system rapidly calculates scores and provides detailed feedback, such as correct answers and explanations if required, enhancing the learning process.
>
> **Content Management**: Teachers can manage quiz content through the administrative interface. This includes adding new questions, editing existing ones, and organizing them into categories or quizzes.

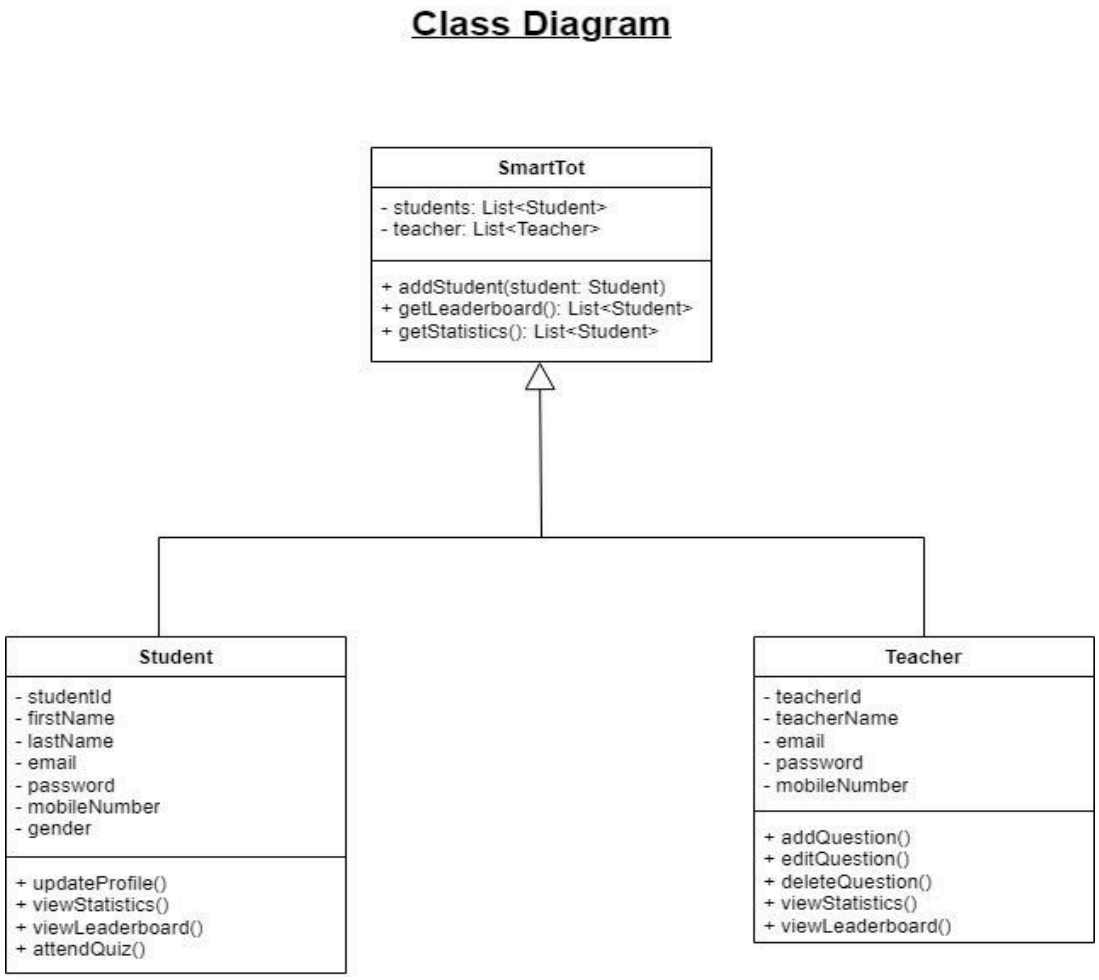
Data Synchronization: The JSON server plays a pivotal role in data synchronization, ensuring that all user interactions, from account creation to quiz completion, are immediately reflected in the db.json data store. The integrity of the user experience is maintained through this real-time data flow.

The Global Control Flow in SmartTot is a testament to the application's commitment to delivering a stable, efficient, and engaging educational tool that caters to the modern needs of both learners and educators.

3. Detailed System Design

3.1 Static View

Class Diagram:



The class diagram for the SmartTot online quiz system illustrates three main classes with their respective attributes and methods:

SmartTot:

- **addStudent(student: Student)**: Registers a new student in the system.
- **getLeaderboard():** Retrieves a list of students, likely sorted based on quiz scores or some performance metric, to display the leaderboard.
- **getStatistics()**: Gathers and returns statistics about quiz performances, possibly including average scores, number of quizzes taken, etc.

Student:

- **updateProfile()**: Allows a student to update personal information like name, password, or contact details.
- **viewStatistics():** Enables a student to view their performance statistics, such as scores and rankings.
- **viewLeaderboard()**: The student can access the leaderboard to see where they stand among peers.
- **attendQuiz()**: The primary method for a student to participate in a quiz.

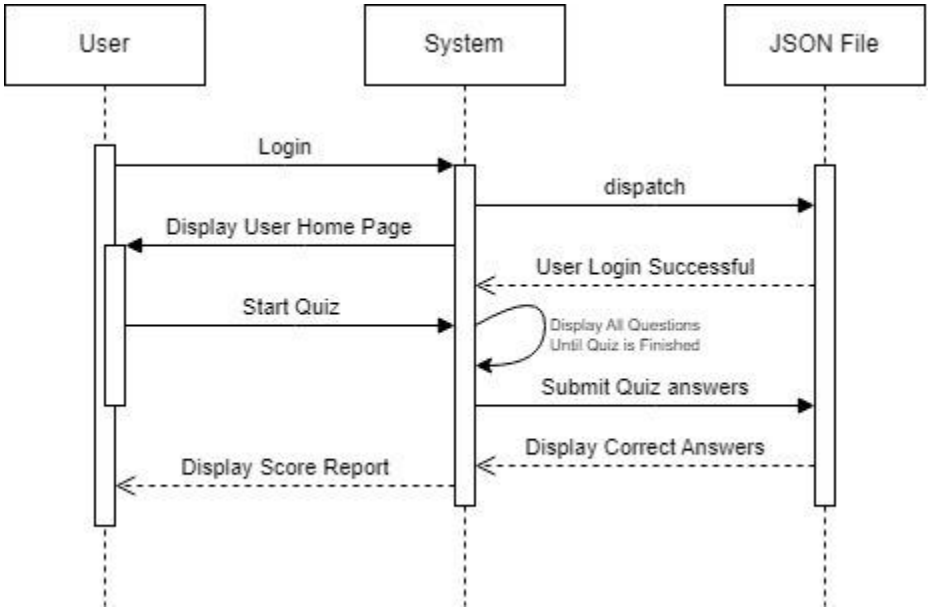
Teacher:

- **addQuestion():** Provides functionality for a teacher to add new questions to the quiz database.
- **editQuestion()**: Allows for modification of existing quiz questions.
- **deleteQuestion():** Enables the removal of questions from the quiz database.
- **viewStatistics():** Teachers can access performance statistics for analysis.
- **viewLeaderboard():** Allows teachers to review the leaderboard, possibly to gauge the overall performance of the class or for individual recognition.

Each method is designed to facilitate interaction with the system's data, either by performing operations like adding, editing, and deleting quiz content or by providing users with information such as statistics and leaderboards. The relationship between SmartTot and the Student and Teacher classes suggests that the system keeps track of multiple users and their roles within the platform, which is essential for maintaining the functionality of a multi-user quiz system.

3.2 Dynamic View

Sequence Diagram:



The sequence diagram illustrates the interaction between a user and the SmartTot system, focusing on the login and quiz-taking process:

Login: The user initiates the process by logging in.
Display User Home Page: After successful login, the system displays the user's home page.
Start Quiz: The user starts the quiz from the home page.
User Login Successful (dispatched to JSON File): The system validates the login credentials with the JSON File.
Display All Questions Until Quiz is Finished: The system sequentially presents questions to the user until the quiz is completed.
Submit Quiz Answers: The user submits their answers, which the system records.
Display Correct Answers: After submission, the system displays the correct answers to the user for review.
Display Score Report: Finally, the user receives a score report from the system.

This flow reflects a typical user's journey through an online quiz, from authentication to completion and review of results, managed by the SmartTot system's backend.

Github link: https://github.com/msajjauncc/SSDI-Project-Group-10