Quiz Odyssey

Design Specification

Group: 26

Eren Taşdemir 150200035

Berkant Bakışlı 150200069

Karol Jan Charchut 912310003

Pijus Jonas Navasaitis 912310007

Egi Gjineci 150190910

Table of Contents

1. Introduction

1.1. Goal

The primary goal of this design specification document is to provide a comprehensive blueprint for the development of our mobile trivia-style quiz application, Quiz Odyssey. This document aims to outline the system architecture, and detailed design elements that will guide the development team in creating an engaging, user-friendly, and scalable quiz platform. The application will offer a diverse range of quiz categories, daily challenges, and a competitive leaderboard to enhance the user experience and encourage continuous learning and interaction.

1.2. Content

This document provides the design specifications of *Quiz Odyssey* application that lets users play trivia quizzes on various topics and compete with other users online asynchronously through leaderboards. It includes the system architecture and the low-level design of the application, such as the structure and components, the classes and methods, the interactions and data flow. The organization of the document can be seen in the Table of Contents.

2. System Architecture Details

2.1. System Architecture

Our system architecture is a **variation of a client-server architecture** featuring **a REST API and real-time communication features using WebSockets**. The client-server architecture is evident in the separation of the mobile application (client) and the backend server, which handle different aspects of the application's functionality. The real-time component built with WebSockets comes into play specifically during the quiz-taking phase when immediate communication is required for presenting questions and receiving responses in real-time. This mix of RESTful APIs and WebSockets allows for efficient handling of our application's standard and real-time interactions.

Client-Side (Mobile Application - React Native):

- User Interface (UI): The React Native framework is responsible for rendering the user interface of our mobile application.
- User Registration: Handles user authentication and registration. This involves sending requests to the server for user creation and authentication.
- Quiz Module: This part of the application manages the presentation of quizzes to the user and captures their responses. It communicates with the server using both RESTful APIs and WebSockets, depending on the real-time nature of the quiz.

Server-Side (Backend - Python, Flask, SocketIO):

- Flask REST API: Handles user registration, authentication, and other non-real-time functionalities. This includes operations like fetching quiz categories, retrieving user information, etc.

- SocketIO for Real-Time Quiz: Manages real-time communication during quizzes. When a quiz starts, the server can push questions to connected clients over WebSockets, and it can receive and validate real-time responses from users.

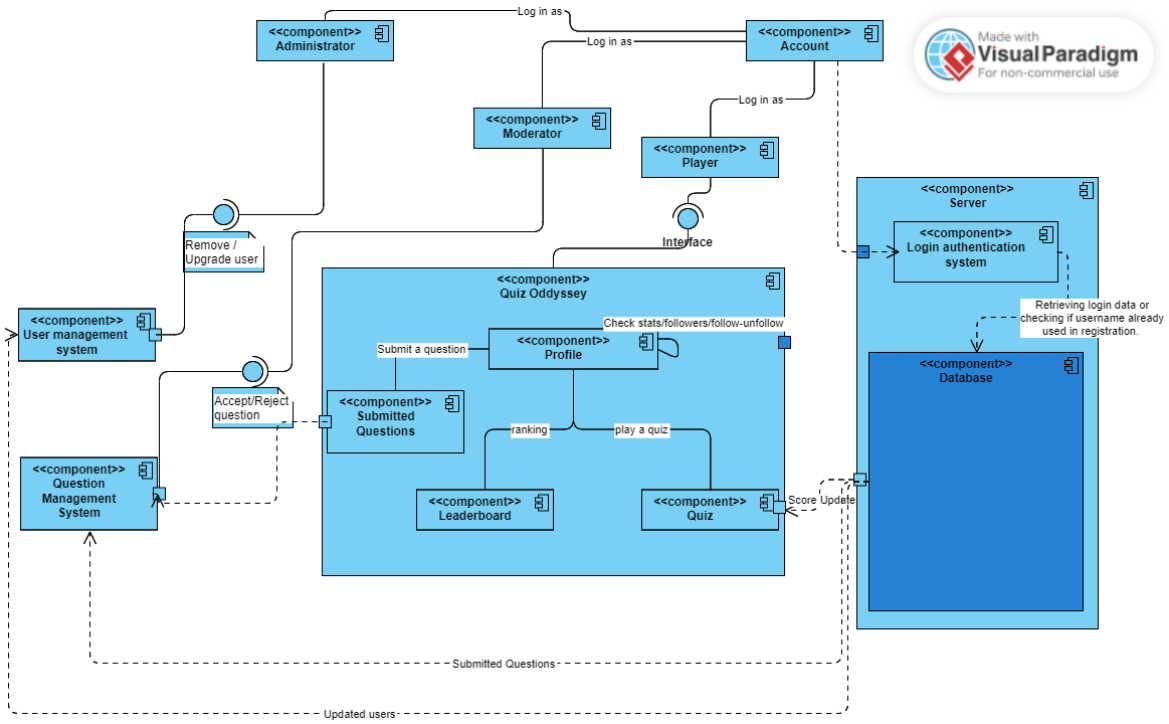
Backend Components:

- User Management: Manages user accounts, authentication, and authorization. Flask handles the RESTful API requests related to user registration, login, and profile management.
- Quiz Management: Manages the quizzes, including fetching questions, handling quiz submissions, and calculating scores. This involves both RESTful API endpoints for non-real-time interactions and WebSockets for real-time quiz sessions.
- WebSocket Server (SocketIO): Manages real-time communication during quizzes. It handles the WebSocket connections from clients and facilitates the exchange of real-time data.

Database:

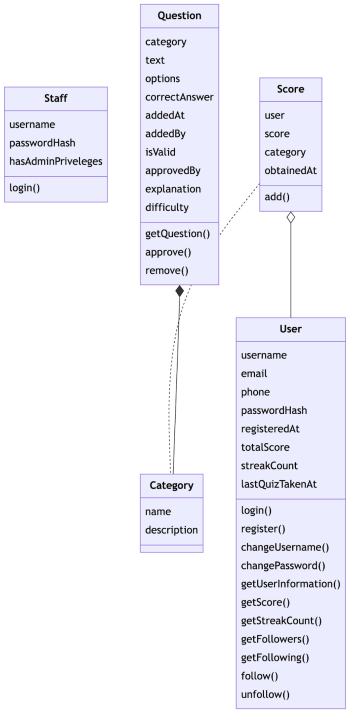
- Holds data such as user profiles, quiz questions, categories, and leaderboards. The backend (Flask) interacts with the database to fetch and store relevant information.

2.2. Component Diagram



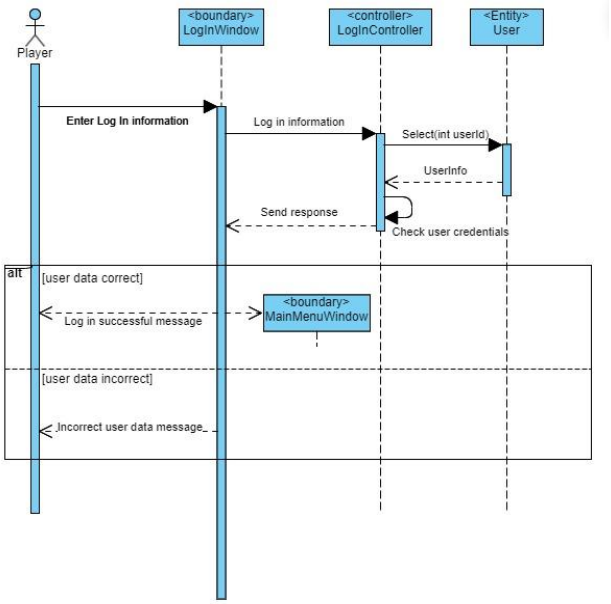
3. Low-Level Design

3.1. Class Diagrams

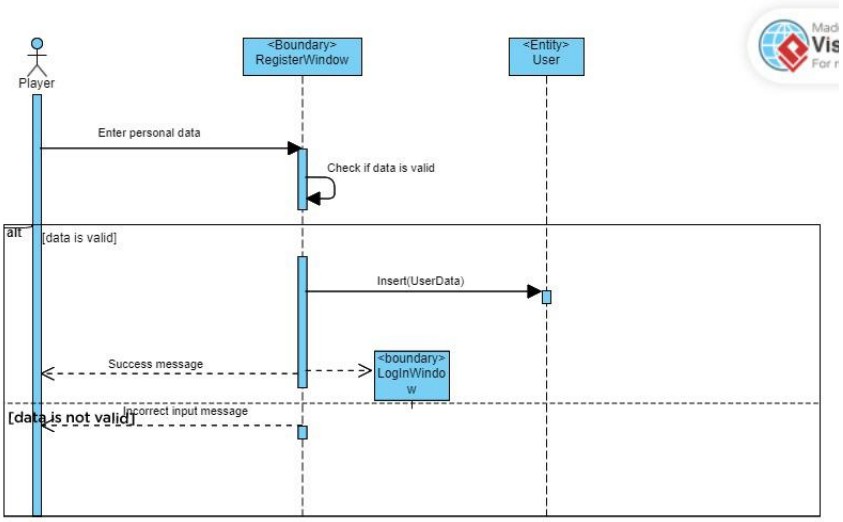


Our system doesn't utilize classes directly, so here the database models are included as classes and their respective API endpoints are shown as class methods.

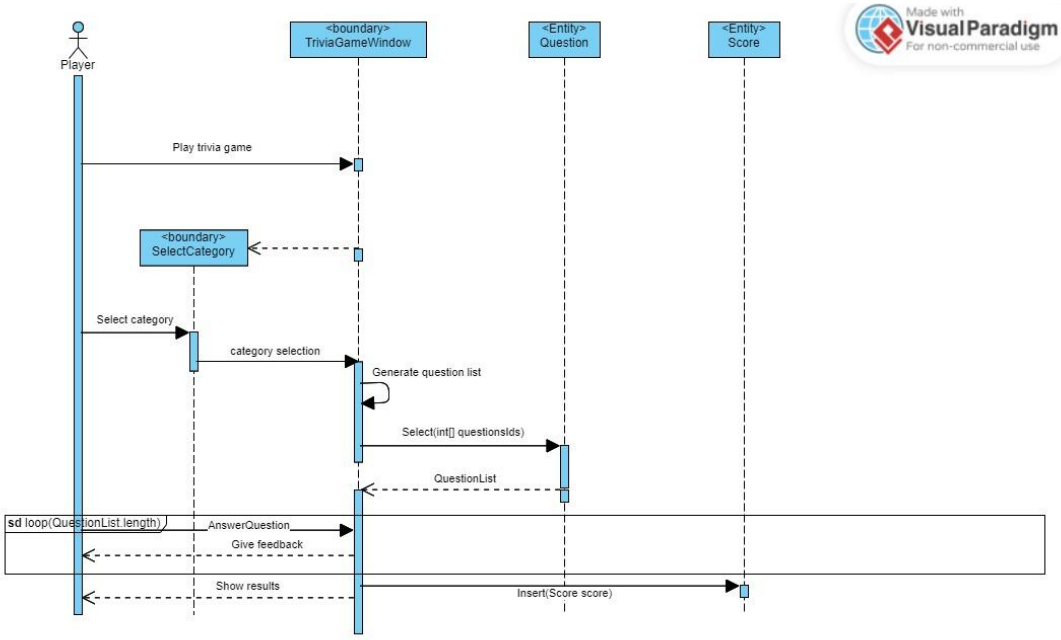
3.2. Sequence or Collaboration Diagrams



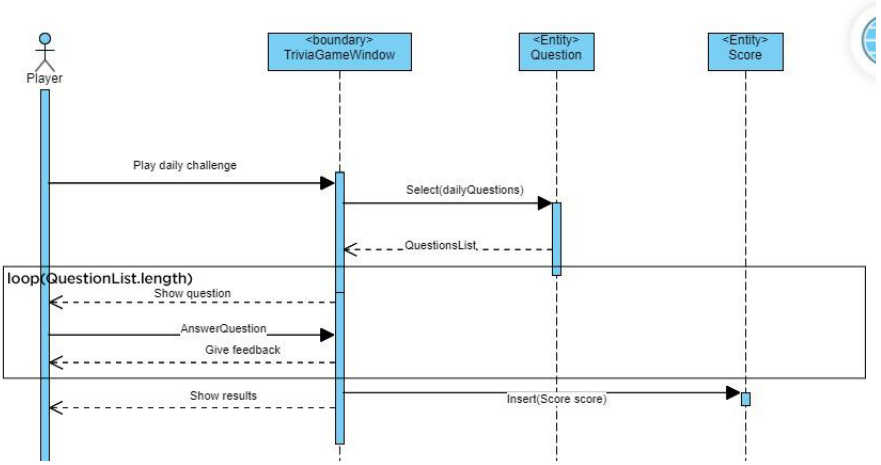
Login sequence diagram



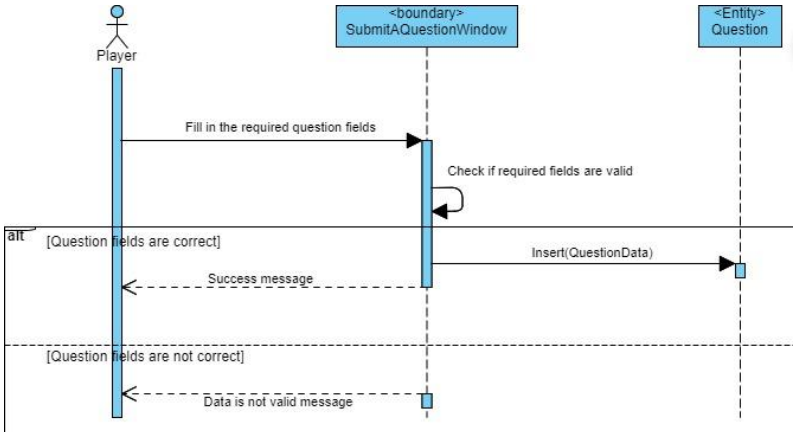
Register sequence diagram



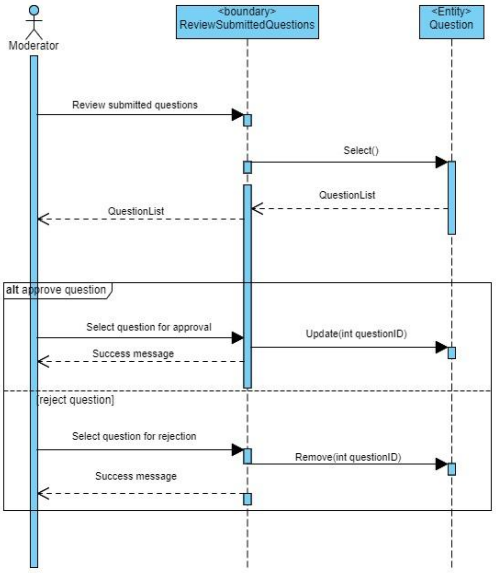
Play trivia game sequence diagram



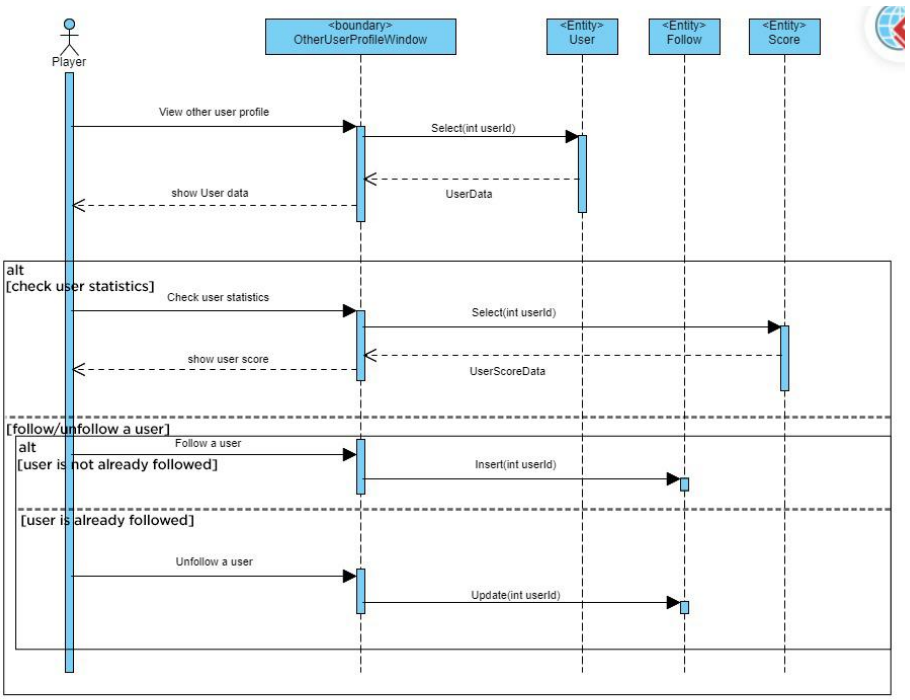
Play daily challenge sequence diagram



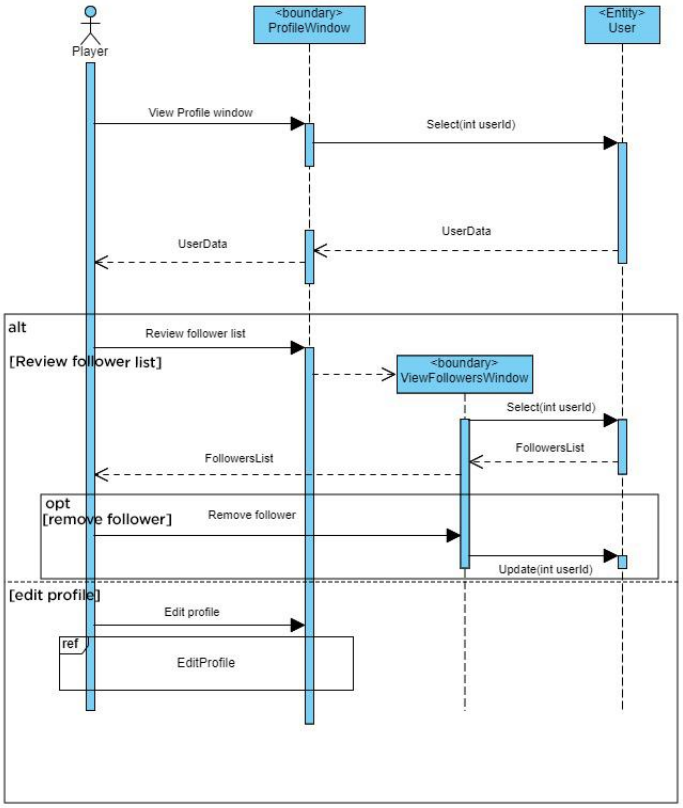
Submit a question sequence diagram



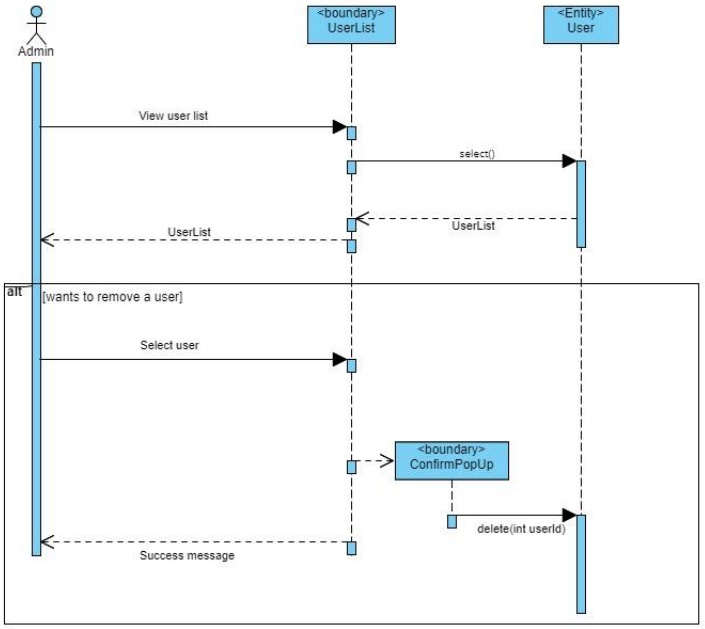
Review submitted question sequence diagram



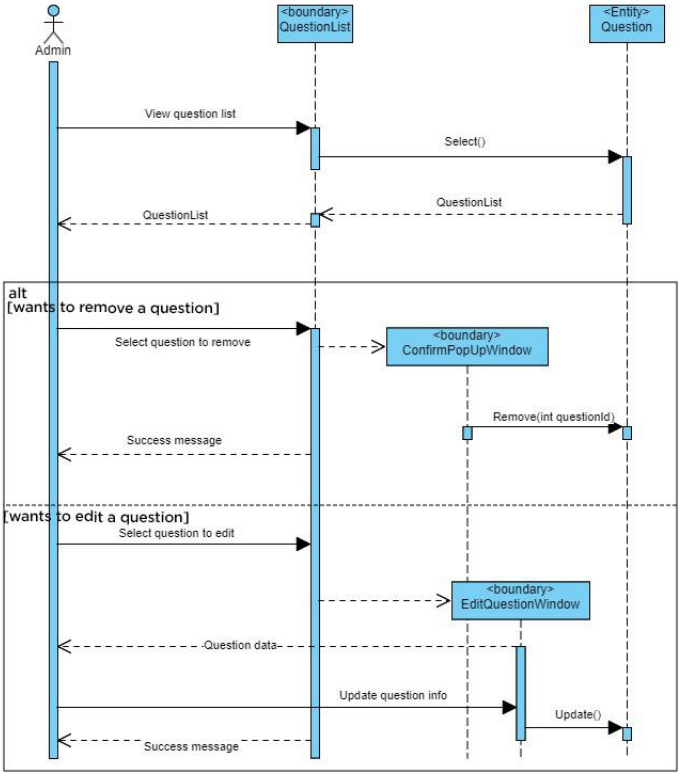
Check other user profile sequence diagram



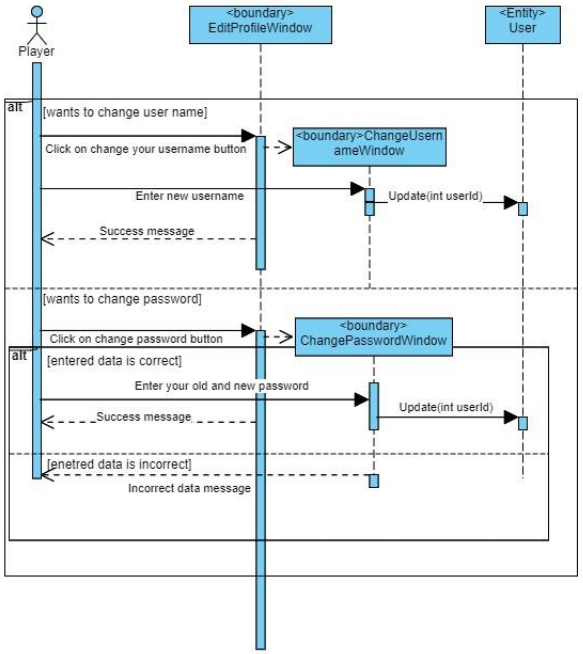
Check your own profile sequence diagram



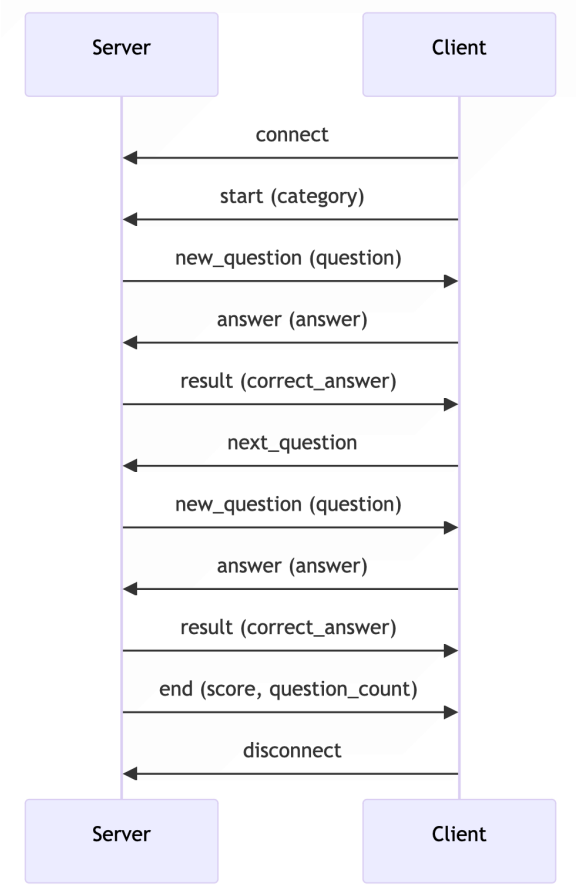
View user list sequence diagram



View question list sequence diagram



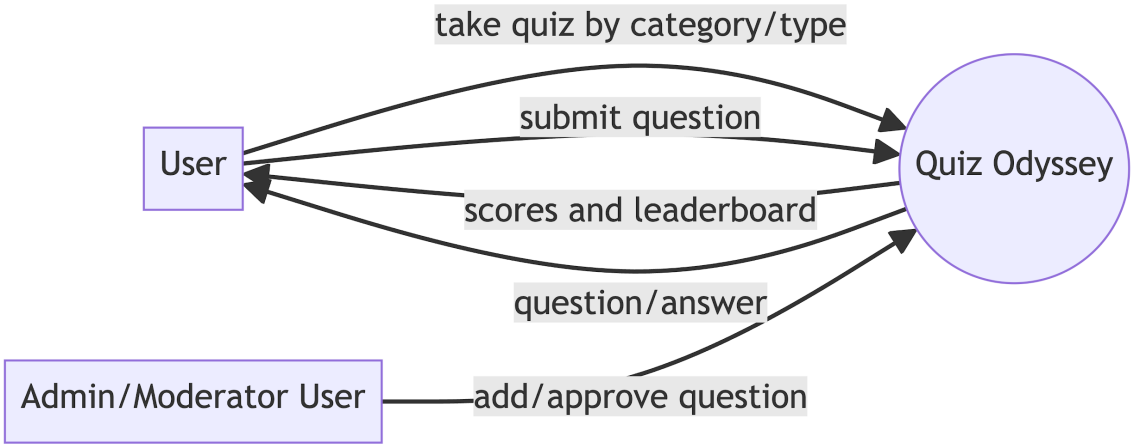
Edit Profile sequence diagram



Quizzing system event-response diagram

3.3. Data Flow Diagrams

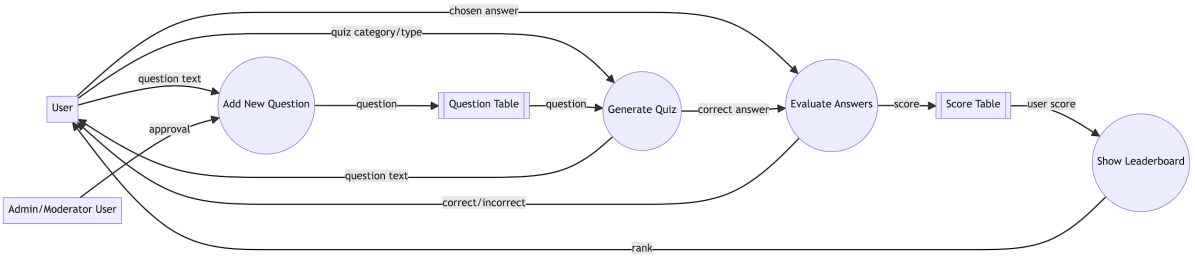
3.3.1. Level 0 (Context Level)



As seen from the above Level 0 Data Flow Diagram, users and admins of our application can do the following:

- Users can submit questions for review
    - Admins can add and approve these questions
- Users can take quizzes by choosing a category, or opting for a daily challenge.
- Users can see questions and correctness of their answers to each question.
- Users can see their scores and the respective leaderboard.

3.3.2. Level 1



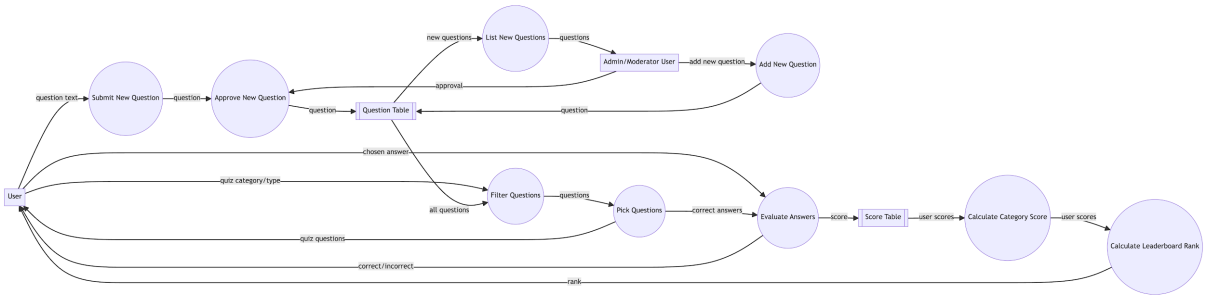
Our system has four major processes in the quizzing mechanism, when broken down to Level 1. About the details of these processes:

- The user can send requests to these processes to get a response.
- The Add New Question process gets the question from a regular user and an approval from the Admin or Moderator, and updates the question table respectively.
- To play a quiz, the user sends a request to the Generate Quiz process, and the process pulls questions from the database. Afterwards, the questions are displayed to the user.
- When the user answers a question, the answer is sent to the Evaluate Answers process, and the correct answer is pulled from the generated quiz. The answer's

correctness is shown to the user, and the Score table is updated with the calculated score.

- At the end of a quiz or on request, the Show Leaderboard process pulls the user scores from the Score table and displays that to the user.

3.3.3. Level 2



In the Level 2 Diagram, every process except for the evaluation of answers was broken down further. Their details are as follows:

- The addition of a question by a user goes to approval for an Admin, and the admin can list the new questions and add questions without the need for approval.
- When the user chooses a category or a type of quiz, firstly questions matching the filter are sent to the question picking process, and from there the questions and their answers are passed to the user and the evaluation process respectively.
- The calculation of the leaderboard rank first happens through the calculation of category scores, ultimately to be converted into a rank to be displayed to the user.