# MAD 2 Project Report: Quiz Master

March 25, 2025

## Author

**Name**: Medha Sen
**Roll number**: 21f1000057
**Student email**: 21f1000057@ds.study.iitm.ac.in
**Author Background**: I am a final year Integrated BS-MS student at Indian Association for the Cultivation of Science, pursuing a major in Computer Science and a minor in Mathematics.

## Description

The Quiz Master - V2 project is a multi-user quiz application that allows an admin (quiz master) to create and manage subjects, chapters, quizzes, and questions. At the same time, users can register, attempt quizzes, and view their scores.

## Frameworks and libraries used

- SQLite: Used as the primary database for storing application data.
- Flask: Web framework for backend logic and routing
- VueJS: Powers the frontend, enabling a dynamic user interface.
- Bootstrap: Used for HTML generation and styling, ensuring a responsive and visually consistent design.
- Redis: Acts as a caching layer to improve application performance and reduce database load.
- Celery: Manages asynchronous task execution, enabling background processing for time-consuming operations like batch jobs and scheduled tasks.
- APScheduler: Handles scheduled tasks and periodic jobs, allowing the execution of recurring background tasks within the application.
- Chart.js – JavaScript library for interactive data visualization.

## Project Details

### Database Schema

All tables have an id as the primary key.

1. User Table
   - id (Integer, PK) – Unique user identifier.
   - username (String(100), Unique, Not Null) – User's email (login ID).
   - password (String(100), Not Null) – Hashed for security.
   - full_name (String(100), Not Null) – User's full name.
   - qualification (String(100), Nullable) – Educational qualification.
   - dob (Date, Nullable) – Date of birth.
   - active (Boolean, Default=True) – Account status (required by Flask-Security).
   - fs_uniquifier (String(64), Unique, Not Null) – Unique identifier for Flask-Security.

   **Constraints and Design Choices**: The admin is the superuser of the app and requires no registration. This account pre-exists in the database when the application is initialized.

2. Role Table
   - id (Integer, PK) – Unique role identifier.
   - name (String(80), Unique, Not Null) – Role name (e.g., Admin, User).
   - description (String(255), Nullable) – Role details.

   **Constraints and Design Choices**: Unique role names ensure clear differentiation. Allows flexible role-based access control.

3. Subject Table
   - id (Integer, PK) – Unique subject identifier.
   - name (String(100), Unique, Not Null) – Subject name.
   - description (Text, Nullable) – Subject details

   **Constraints and Design Choices**: Unique constraint on name to avoid duplicate subjects. Text field for description to allow detailed subject information.

4. Chapter Table
   - id (Integer, PK) – Unique chapter identifier.
   - name (String(100), Not Null) – Chapter name.
   - description (Text, Nullable) – Chapter details.
   - subject_id (Integer, FK $\rightarrow$ Subject.id, Not Null) – Links to a subject.

   **Constraints and Design Choices**: Foreign key subject_id ensures a chapter belongs to one subject. Allows multiple chapters under a subject without duplicates.

5. Quiz Table
   - id (Integer, PK) – Unique quiz identifier.
   - chapter_id (Integer, FK $\rightarrow$ Chapter.id, Not Null) – Associated chapter.
   - date_of_quiz (Date, Nullable) – Quiz date.
   - time_duration (String(10), Nullable) – Duration (HH:MM).
   - remarks (Text, Nullable) – Additional comments.

   **Constraints and Design Choices**: A quiz is tied to a chapter for better subject-wise organization.Time_duration is stored as a string for flexibility in formatting (HH:MM).

6. Questions Table

- id (Integer, PK) – Unique question identifier.
- quiz_id (Integer, FK → Quiz.id, Not Null) – Associated quiz.
- question_title (String(100), Not Null) – Short title.
- question_statement (Text, Not Null) – Full question..
- option1, option2, option3, option4 (String(300), Not Null) – MCQ options.
- correct_option (Integer, Not Null) – Correct answer (1-4).

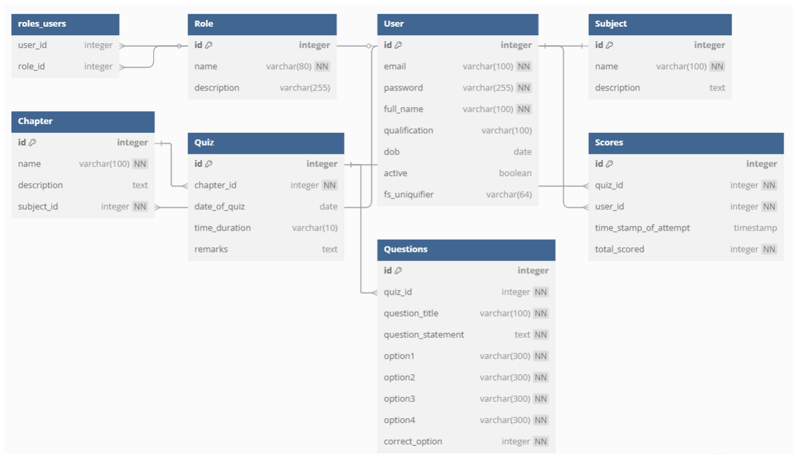
**Constraints and Design Choices**: Foreign key quiz_id ensures a question belongs to a quiz. MCQ format with four fixed options ensures standardization. Integer correct_option (1-4) ensures structured answer validation.

7. Scores Table

- id (Integer, PK) – Unique quiz attempt identifier.
- quiz_id (Integer, FK → Quiz.id, Not Null) – Associated quiz.
- user_id (Integer, FK → User.id, Not Null) – User attempting the quiz.
- time_stamp_of_attempt (DateTime, Not Null) – Attempt time.
- total_scored (Integer, Not Null) – User's score.

**Constraints and Design Choices**: Tracks performance by linking a quiz attempt to a user. DateTime field ensures precise tracking of attempts. Foreign keys (quiz_id, user_id) establish relationships for analytics.

## Entity Relationship diagram



## API design

The API endpoints ensure access to quiz-related data, structured around subjects, chapters, quizzes, and user scores. The full list of API endpints are as listed in the YAML file attached.

## Architecture and Features

### Architecture

The project follows a modular Flask structure for clarity and maintainability.

- The folder *application* has several subfolders: exports, instance, static, templates
- models.py defines the database schema.
- app.py initializes the Flask application, configures authentication, database, and API routes, and ensures required roles and users exist. It also includes utility functions for generating quiz-related performance charts.
- resources.py defines RESTful API endpoints for managing subjects, chapters, quizzes, questions, scores, and statistics using Flask-RESTful. It includes caching, rate limiting, and CRUD operations for efficient data retrieval and updates.
- routes.py defines Flask routes for user authentication, profile management, and role-based access control (RBAC) using Flask-Security. It includes caching, rate limiting, and admin/user-specific endpoints.
- tasks.py is responsible for scheduling tasks, sending email notifications, and generating performance reports.
- config.py defines application configuration settings, including database connection, security options, and background task management using Celery.
- In *static/script.js* the routing configuration mapping URLs to specific components such as login, registration, quiz management, and dashboards for users and admins is defined.
- *static/components* contains all the vue js files that define and render the frontend UI components of the application.
- The *templates/index.html* file serves as the main entry point for the frontend, loading Vue.js, Bootstrap, and other dependencies while providing a root for rendering the Vue components.
- The *static* folder also contains subfolders named *exports* (for storing exported csv and monthly reports) and *images* for storing images used in app frontend design.
- Instance folder stores the SQLite database.
- .env is the virtual env folder containing all dependencies and required libraries.

## Features

The Admin is the superuser of the application and does not require registration. The admin account is pre-created in the database upon initialization. Upon launching the app, users are directed to the Home Page, where they can:

- User Login: Requires a valid username and password. Users must be registered in the database or can sign up.
- Admin Login: Requires a pre-existing username and password. Admin registration is not allowed. Incorrect credentials prompt an error message
- User Registration: Requires full name, email, qualification, date of birth, and password. The username must be unique.

**User Dashboard and Quiz Features**

Upon login, users are directed to their dashboard, where they can browse and attempt quizzes with set expiry dates and time limits. If a quiz is not manually submitted, it auto-submits upon timeout, redirecting users to the Scores Page. Users can view their latest scores via the "View Scores" option on quiz cards and download their quiz attempt history as a CSV file. A search bar allows filtering quizzes by chapter, subject, or keywords, while the *Edit User* option (an additional feature added by me) enables profile updates. The Scores Page features a Quiz Leaderboard ranking the top three scorers, personal performance insights—including total attempts, highest and average scores, and attempt history—and a Performance Overview with subject-wise score comparisons and a pie chart of the most attempted subjects.

### Admin Dashboard and Management Controls

Admins can manage subjects, chapters, quizzes, and questions through the Admin Dashboard, with search functionality for quick access. They can create and modify quizzes, set durations, and edit multiple-choice questions. The Scores Summary Dashboard provides insights into user performance, featuring a Quiz Leaderboard, quiz attempt statistics, and a subject attempt distribution chart. Admins can search for users to view detailed performance analysis, export data as a CSV, and access a Quiz Summary Table with key details of all quizzes.

### Automated Notifications and Reports

Users receive automated monthly performance PDF reports via email on the 1st of each month, along with daily quiz reminders sent at 9 AM.

## Video Link

MAD 2 Video presentation