Subject: Modern Application development 1

Name: M.Sharukesh

Student ID: 23f2001918

Quiz Master v1 – Multi-User Quiz Application

Objective:
The Quiz Master application is designed as an exam preparation platform that supports two types of users: Admin (Quiz Master) and the Quiz User. The admin manages content by creating subjects, chapters, quizzes which has questions while setting time duration and start time for quiz access. Quiz users can register, log in attempt quizzes within the given time-frame and review their past performance.

Problem Statement:
Users need a platform to prepare for exams by practicing with multiple-choice questions (MCQs) across various subjects and chapters. The platform must allow administrators to create and manage content and control quiz availability. Users should receive feedback after attempting quizzes and have access to their quiz history for their performance review.

Approach:
The project was implemented iteratively:

- **Database Design:** Relational database schema was designed for the following entities subjects, chapters, quizzes, questions, scores, and user details.
- **Authentication & Role Management:** Flask-Login was integrated to handle separate access for admin and quiz users.
- **Quiz logic:** The quiz functionality includes a timer, answer evaluation, and feedback generation.
- **Reporting:** Features for search, quiz history, and performance reporting were added to enhance user experience.

Implementation Details

Back end:
The back end is built with Flask and uses Flask-SQL Alchemy as the ORM to interact with an SQLite database.

Backend Functionalities

- **User Authentication:** Handles user login and user registration
- **Admin Dashboard:** Admin users can manage subjects, chapters, quizzes, and questions. The admin dashboard displays Count for the number of subjects, chapters, quizzes, question.
- **Quiz Attempt:** Users can attempt quizzes only within the timeframe defined by the admin. There is a visual countdown, upon completion of the countdown the exam auto submits.
- **Quiz Feedback and History:** After completion of the quiz the user can see the score of the quiz. User is also able to see the past quiz attempts.

Front end:
HTML, CSS, and JavaScript were used for the user interface. Templating is handled with Jinja2.

Front end Functionalities

- **Login/Sign up Pages:** For user login and sign up
- **Admin Dashboard:** Has a side navbar for navigation and forms to accomplish all the CRUD operations on the subjects, chapters, quizzes, question
- **Quiz Pages:** It has a timer, questions and options with radio buttons

Frameworks and Libraries Used

- **Flask:** Used to handle web requests and routing, while maintaining user type access with password validation as security in terms of per user type (admin, default user)
- **Flask-SQL Alchemy:** Provides ORM capabilities for interacting with the SQLite database in an object-oriented manner

- **Flask-Login:** Manages user sessions, ensuring valid authentication and role-based access control
- **Flask-Bcrypt:** Handles secure password hashing to protect user credentials
- **Jinja2:** The templating HTML pages used to display dynamic content
- **SQLite:** A lightweight, file-based database chosen for ease of use and setup
- **HTML/CSS/JavaScript:** Used to build the front end, ensuring a responsive and user-friendly interface

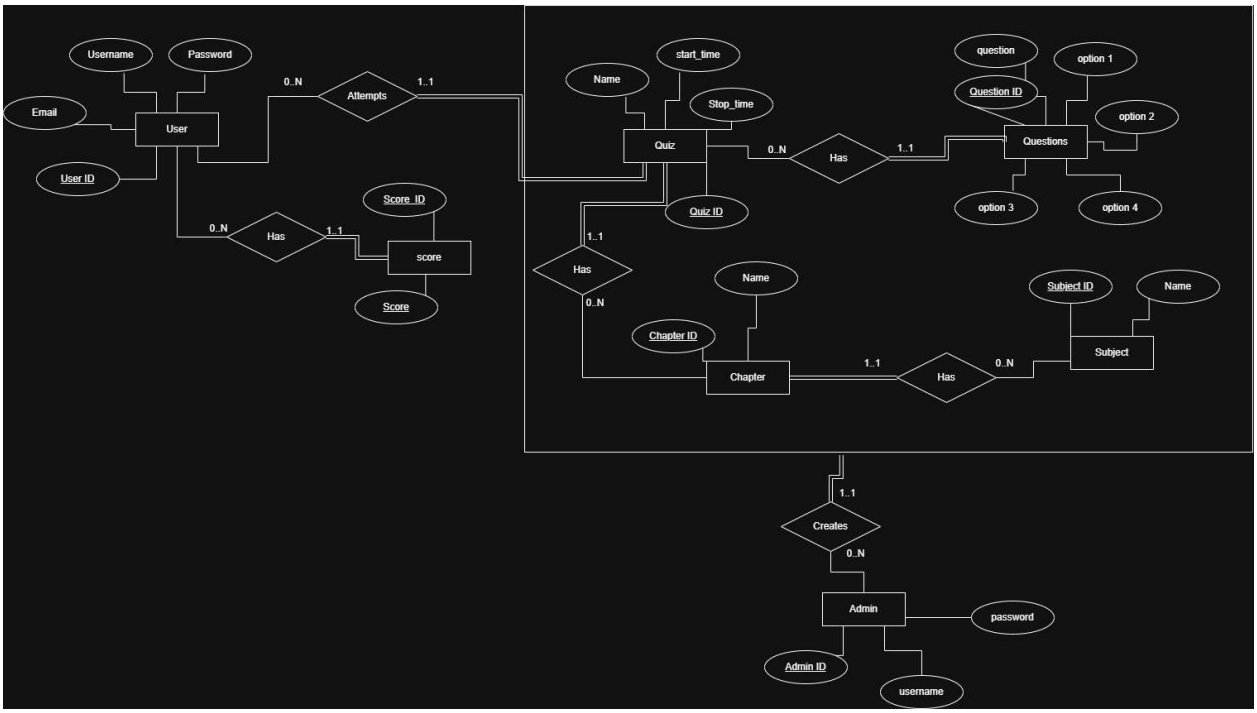
Design Decisions

- **Database Schema:**
  The database schema was designed with clear one-to-many relationships:
  - **Subject → Chapter:** One subject can have multiple chapters
  - **Chapter → Quiz:** Each chapter can contain multiple quizzes
  - **Quiz → Question:** A quiz comprises many questions
  - **Score:** Stores each quiz attempt's score, linking users and quizzes

- **Role-Based Functionality:**
  Separate models and authentication logic ensure that only admin users can manage content.

- **Time frame Enforcement:**
  Quizzes have defined start and end time, ensuring that users can only attempt quizzes within the allowed window. Start and end time of each quiz is saved.

ER Diagram



Explanation:

- **subject:**
  Contains the subjects, each of which can have multiple chapters.
- **chapter:**
  Belongs to a subject and contains quizzes.
- **quiz:**
  Belongs to a chapter, includes a start_time and end_time to enforce the availability window. A quiz contains multiple questions.
- **question:**
  Belongs to a quiz and contains the text, options, and correct answer.
- **score:**
  Records each quiz attempt, linking a user to a quiz and storing the score achieved.

- **User & Admin:**
  Represent quiz users and administrators, ensuring role-based access to features.

Challenges and Lessons Learned

- **Role Management:**
  Differentiating between admin and regular users was a key challenge. The project required careful management of user sessions and explicit role-checking to ensure that sensitive functions (like content management) specific to admin user.

- **Time frame Enforcement:**
  Implementing a timer and enforcing quiz attempt windows required precise use of date-time operations, ensuring that the quiz can only be attempted within the allotted time frame.

- **Database Design:**
  Designing a normalized schema with clear relationships was crucial for maintaining data integrity. The iterative design process and ER diagram helped in visualizing and implementing the relationships.

Drive link for the presentation video

- https://drive.google.com/file/d/17wClg8QuL85R-sGC0objH6maHdfVgioy/view?usp=sharing