Quiz Management System

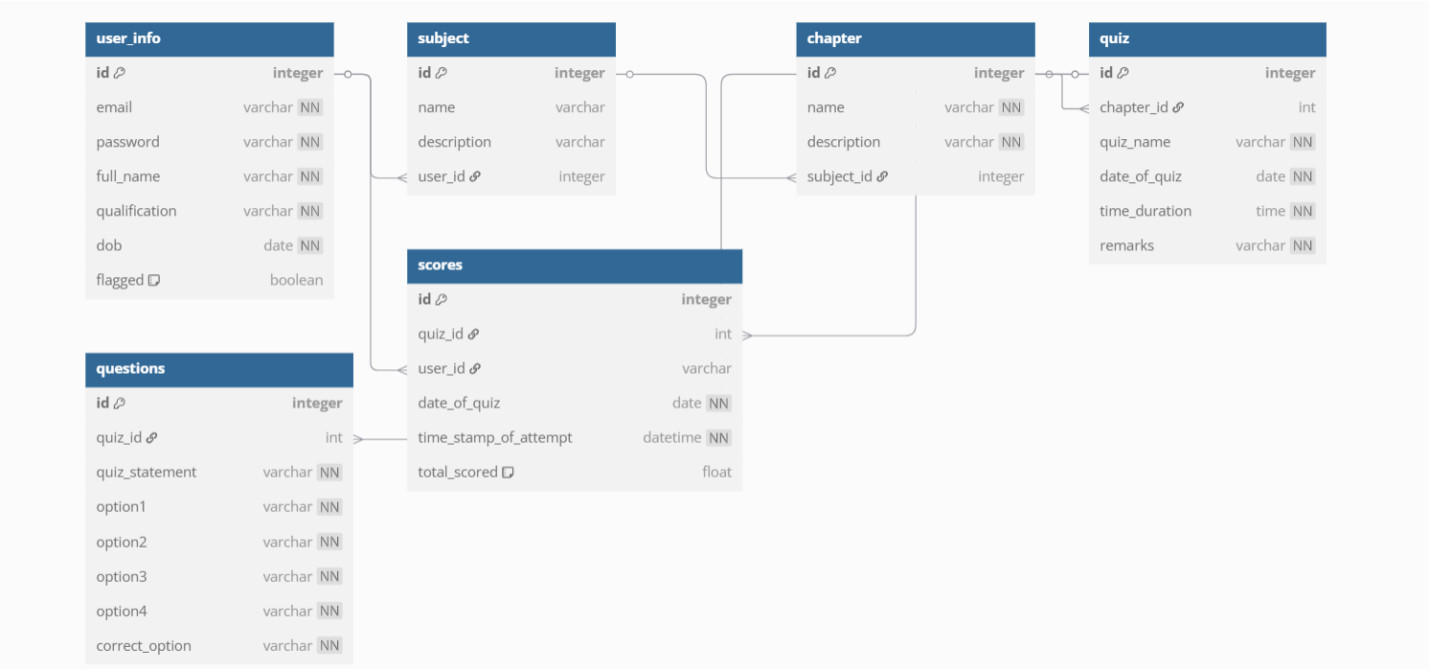
Description

The **Quiz Master Application** allows admins to create, edit, and delete quizzes under various chapters and subjects while managing user access by flagging/unflagging users. Admins can also search and view subject-wise top scores. Users have a personalized dashboard to search for and attend quizzes, track progress, and view their summary along with scores.

Frameworks and libraries used

- **Flask framework** was used for developing the application.
- **Jinja2** was used for templating and HTML generation.
- **Bootstrap 5** was used for styling and designing purpose.
- **Flask-SQLAlchemy** and **sqlite3** were used for database operations.
- **Flask-RESTful** was used for implementing the RESTful APIs.
- **Matplotlib** is used for generating the insights on subject wise top scores by various users with the help of histogram plots.

DB Schema Design



User_Info table: id is PRIMARY KEY AUTO INCREMENTED. All the attributes are not null. This table uses backref relationship with subject table.

Subject table: id is PRIMARY KEY AUTO INCREMENTED. user_id is FOREIGN KEY. All the attributes are not null. This table uses backref relationship with chapters table.

Chapter table: id is PRIMARY KEY AUTO INCREMENTED. subject_id is FOREIGN KEY. All the attributes are not null. This table uses backref relationship with quiz table.

Quiz table: id is PRIMARY KEY AUTO INCREMENTED. chapter_id is FOREIGN KEY. All the attributes are not null. This table uses backref relationship with questions table.

Questions table: id is PRIMARY KEY AUTO INCREMENTED. quiz_id is FOREIGN KEY. All the attributes are not null.

Scores table: id is PRIMARY KEY AUTO INCREMENTED. quiz_id and user_id are FOREIGN KEYS. All the attributes are not null. This table uses backref relationship with Quiz and User_Info tables.

API Design

API has been created using the resource class from Flask-Restful. They can handle *GET/POST/PUT/DELETE* or *CRUD* requests and their response is in *JSON* format.

- **UserApi**-Resource for CRUD on **User_Info** Model

Endpoints are

- **/api/users/<int:user_id>** - for creating, reading, updating and deleting the user_info

Architecture and Features

The root folder consists of app.py, api.py and 5 more folders named controllers (which consists of routes.py), instance (consists of the database file), models (consists of different models), static (consists of css files, images folder, and the stats image folder) and templates (which has all the html files).