PROJECT DETAILS

Problem Statement:

A Quiz Management System that allows users to create, manage, and participate in quizzes. The system ensures smooth administration and evaluation of quizzes, making it an efficient tool for both educators and students.

How I approached the Problem Statement:

To address this problem, a Flask-based web application was developed with a well-structured database and a modular architecture to ensure scalability and ease of use. The development process began with a thorough requirement analysis to understand the needs of different types of users, including Admins. This helped in identifying the essential features required to provide a seamless experience.

Once the requirements were clear, the database design phase focused on defining entities and their relationships. Ensuring proper data normalization and efficient retrieval mechanisms was a priority. With the database structure in place, the backend was implemented using Flask to handle authentication, quiz management, and user interactions, while SQLAlchemy was chosen for ORM-based database handling to streamline queries and data manipulation.

Parallelly, the frontend was developed using Jinja2 templates along with Bootstrap to create a responsive and user-friendly interface. This ensured that the system was not only functional but also accessible and easy to navigate. After development, the project underwent extensive testing to identify and resolve any issues. The final step was deploying the system, making it available for real-time usage.

The system consists of several major components. User management enables users to register and log in with different roles, such as Participants. Quiz management allows quiz master (Admin) to create, edit, and delete quizzes, while the participation module ensures that users can attempt quizzes and view their scores. SQLAlchemy is used to handle database operations efficiently.

The Flask application is initialized in app.py, where a Flask instance is created, a SQLite database (Quiz.sqlite3) is set up, and database models are defined in backend.models. Routes are managed using a separate backend.controllers module to maintain a clean code structure.

FRAMEWORKS AND LIBRARIES USED

- **HTML, CSS, Bootstrap** (Frontend)
- **Flask** (Web framework for Python)
- **SQLAlchemy** (ORM for database management)
- **Jinja2** (Templating engine for HTML rendering)
- **Bootstrap** (Front-end framework for UI design)
- **OS, Datetime** (Utilities & Modules)

Database Schema Design:

```
QUIZ MASTER/
│── __pycache__/
│── requirments.txt
│── .gitignore
│── extensions.py
│── main.py
│── model.py
│── readme.md
│── routes.py
│── instance/
│── static/
│── templates/
│   │── admin/
│   │── user/
```

Key Features

- Admin & User Login: Admin can log in using credentials from dbmanage.py, while users must sign up.
- Quiz Management: Admin can create, edit, and delete subjects, chapters, quizzes, and questions.
- Search & Analysis: Admin can search for users, subjects and quizzes and view performance using summary charts.
- User Features: Users can attempt quizzes, track scores, and review past attempts.

Video Link:
https://drive.google.com/file/d/1dGD6qMY7sdFFnAnVgzWMm7xbFXQerKAi/view?usp=sharing