

PROJECT REPORT

AUTHOR:-

NAME: MOHAMMED SHARUKH B

ROLLNO: 23F2002020

EMAIL: 23f2002020@ds.study.iitm.ac.in

I am currently pursuing a Bachelor's in Engineering (BE) at Sathyabama University, specializing in Computer Science and Engineering, while concurrently undertaking an online Bachelor's degree in Data Science from IIT Madras. This dual academic trajectory allows me to indulge in my passion for both engineering and data science, fostering a robust interdisciplinary skill set. My commitment to continuous learning drives my ambition to stay at the forefront of technological advancements, positioning me to address complex, data-driven challenges in an ever-evolving landscape.

PROJECT DESCRIPTION:-




This project is a web-based quiz management system that can be used by administrators and users alike. While users can attempt quizzes, view their scores, and monitor their progress, administrators have the ability to create, manage, and remove quizzes, chapters, and subjects. Performance analytics, quiz deadlines, and user authentication are additional features of this project.

TECHNOLOGIES USED:-

- **Flask:** A lightweight web framework for Python used to build the web application.
- **Flask-RESTful:** An extension for Flask that simplifies the creation of REST APIs, used for managing quizzes via API endpoints.
- **SQLAlchemy:** An ORM (Object-Relational Mapping) tool for database interactions, used to manage database models and queries.
- **Flask-Migrate:** An extension for handling database migrations, ensuring schema changes are managed smoothly.
- **Bootstrap:** A front-end framework for responsive and styled web pages.
- **Matplotlib:** A plotting library used to generate performance charts (e.g., bar charts, pie charts) for user analytics.
- **Jinja2:** A templating engine for rendering dynamic HTML content.

ARCHITECTURE AND FEATURES:-

Controllers : This is where all the backend logic is handled:

-  **controller.py:** Handles web routes, user authentication, quiz interactions, and dashboard functionalities
-  **api_controller.py:** Provides RESTful APIs for quizzes, including creating, updating, deleting, and retrieving quizzes
-  **model.py:** Specifies database models for users, subjects, chapters, quizzes, questions, quiz results, and user answers

Templates: HTML templates (e.g., user_layout.html, admin_layout.html, search.html) are placed in the templates folder and used to render front-end views.

Static Files: CSS and JavaScript files (e.g., style.css) are kept in the static folder for styling and client-side functionality.

User Authentication: Users can log in, sign up, and log out. Admins have additional privileges like creating quizzes and managing content.

Common features:

Quiz Management: Admins can create, edit, and delete quizzes, set deadlines, and add questions. Users can attempt quizzes before the deadline.

Performance Analytics: Users can view their quiz scores, subject-wise performance, and overall quiz completion rates through visual charts (bar charts and pie charts). Admin can also view the individual performance of user as well as collective performance.

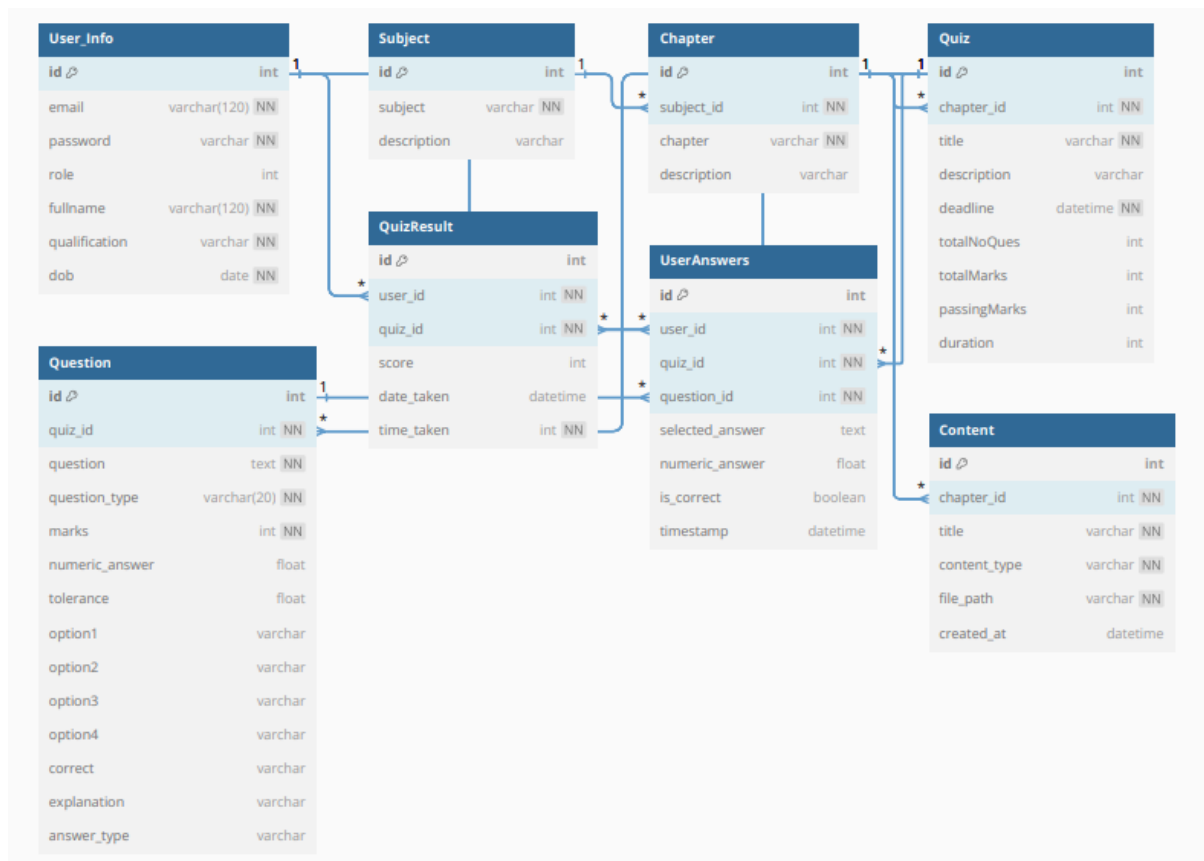
Search Functionality: Admins can search for subjects, chapters, quizzes, and users.

REST API: The api_controller.py provides RESTful endpoints for managing quizzes, allowing for integration with other systems.

Additional feature:

Content Management: Admins can upload and manage content (videos, PDFs, PPTs, etc.) for each chapter.

DATABASE SCHEMA DESIGN:-



This database schema is designed to maintain data in a organized way and to prevent repetition. It adheres to norms to ensure accuracy, effortless expansion and quick searches. Foreign keys link related data, keeping records connected. In order to efficient monitoring of quiz and learning progress while preserving data integrity, timestamps are used to track user actions.

API DESIGN:-

QuizAPI (/api/quizzes)

- GET: Fetch all quizzes.
- POST: Create a new quiz.
- PUT: Update an existing quiz by ID.
- DELETE: Delete a quiz by ID.

QuizSearchAPI (/api/quizzes/search/<id>)

- GET: Fetch a quiz by its ID.

The API provides a comprehensive set of endpoints for managing quizzes, including listing, creating, updating, and deleting quizzes. It also allows for searching for a specific quiz by its ID. The implementation leverages Flask-RESTful for routing and resource management, SQLAlchemy for database interaction, and includes basic validation and error handling to ensure data integrity.

VIDEO LINK:-

<https://drive.google.com/file/d/1Z8qE4868YB0cm3fZNEXTTx6J-AIEvr2k/view?usp=sharing>