



# PROJECT REPORT:

## Salt And Prepper

### Author

Krishiv Khambhayata

Roll No.: 23f3003998

23f3003998@ds.study.iitm.ac.in

Currently at the time of writing, I'm pursuing a Diploma in Programming from Indian Institute Of Technology, Madras and Bachelor of Engineering from SAL Engineering And Technical Institute.

### Description

This project is directed for two core responsibilities: Administrator and User. The Webapp acts as a GUI for the Admin to conduct CRUD operations to form/edit Subjects and Quizzes within the Subjects while Users are provided with a medium to attempt the Quizzes and record their scores.

### Technologies used

Languages: Python (for backend and operations)

HTML, CSS, JS (for frontend templating and adding functionalities)

Frameworks: Flask (for writing API end-points)

Libraries: Flask-SQLAlchemy (for Object-Relational Mapping)

Flask (for handling sessions, redirects, flashing feedbacks and fetching data from frontend)

Werkzeug.security (for hashing passwords at signup and verifying hashes at login)

Bootstrap (for reusable UI components)

Charts JS (for rendering charts displayed in the Summary tabs)

Jinja2 (for rendering databases dynamically within the HTML templates)

Tools and Technologies: Visual Studio Code (Integrated Development Environment)

SQLite3 (database storage)

Git (version control)

### DB Schema Design

User class

- a. id: integer, primary key
- b. email: string, unique, cannot have null value
- c. username: string, unique, cannot have null value
- d. password: string, cannot have null value
- e. full\_name: string, cannot have null value



- f. qualification: string, cannot have null value
- g. dob: date, cannot have null value
- h. role: string, cannot have null value, defaults to 'user'
- i. datetime: datetime, defaults to current UTC time

Subjects class

- a. id: integer, primary key
- b. name: string, unique, cannot have null value
- c. description: string, cannot have null value

Quizzes class (one to many relationship with Subjects)

- a. id: integer, primary key
- b. subject\_id: integer, foreign key to subjects.id, cannot have null value
- c. name: string, cannot have null value
- d. no\_of\_questions: integer, cannot have null value
- e. duration: integer, cannot have null value
- f. validity: datetime, cannot have null value
- g. datetime: datetime, defaults to current UTC time

Questions class (one to many relationship with Quizzes)

- a. id: integer, primary key
- b. quiz\_id: integer, foreign key to quizzes.id, cannot have null value
- c. question\_title: string, cannot have null value
- d. question\_statement: string, cannot have null value
- e. option1: string, cannot have null value
- f. option2: string, cannot have null value
- g. option3: string, cannot have null value
- h. option4: string, cannot have null value
- i. correct\_option: integer, cannot have null value

Scores class (one to many relationship with Users and Quizzes)

- a. id: integer, primary key
- b. user\_id: integer, foreign key to user.id, cannot have null value
- c. quiz\_id: integer, foreign key to quizzes.id, cannot have null value
- d. score\_percentage: integer, cannot have null value
- e. attempted\_answers: string, cannot have null value
- f. correct\_answers: string, cannot have null value
- g. start\_time: datetime, defaults to current UTC time
- h. end\_time: datetime, defaults to current UTC time



## API Design

### Users

1. API endpoint for Dashboard to display Available Quizzes with validity > current DateTime.
2. API endpoint to start a quiz.
3. API endpoint to render the started quiz, accept responses and POST them in Scores.db.
4. API endpoint for Scores to display past attempts and performances.
5. API endpoint for Summary to graphically display User Activity.
6. API endpoint to clear attempted selections from the session while taking a Quiz.

### Admin

1. API endpoint for Admin Dashboard to display Quizzes sorted by Subjects.
2. API endpoint to POST a new Subject to the Subjects.db.
3. API endpoint for Quizzes tab to display Questions sorted by Quizzes along with Quiz details.
4. API endpoint to POST a new Quiz to the Quizzes.db.
5. API endpoint to POST a new Question to the Questions.db.
6. API endpoint to UPDATE a question from the Questions.db.
7. API endpoint to UPDATE a subject from the Subjects.db.
8. API endpoint to UPDATE a quiz from Quizzes.db.
9. API endpoint to DELETE a subject from the Subjects.db.
10. API endpoint to DELETE a question from the Questions.db.
11. API endpoint to DELETE a quiz from Quizzes.db.
12. API endpoint for Admin Summary to graphically display Platform Activity.
13. API endpoint to add Admin to the Users.db.

### General

1. API endpoint for Login to verify POSTED credentials using check\_password\_hash function.
2. API endpoint for Signup to POST a new user record into the Users.db.
3. API endpoint for Logout to clear session variables and redirect to Signup webpage.

## Architecture and Features

### Architecture

The Root Project Directory contains the following:

1. app.py
  - a. This file contains all the necessary import statements such as flask, model classes/views, werkzeug.security, dotenv, os, json, datetime and SQLAlchemy.
  - b. This file also contains all the controllers (API endpoints/routes).
2. /models
  - a. This directory contains the models.py file.
  - b. models.py contains the schema of all the databases we'll require for our project.
3. add\_admin.py
  - a. This is a script meant to add the first and only record of 'superadmin' role into Users database.



4. templates/
  - a. This contains all HTML5 files that form the webpages of our web application.
  - b. This directory includes contains admin\_dashboard.html, admin\_summary.html, quizzes.html, add\_admin.html, dashboard.html, scores.html, summary.html, login.html and signup.html.
5. instance/
  - a. This directory is generated when db.create\_all() is executed for the first time. It generates all the databases whose schemas are present in models.py file.
  - b. Once the instance directory is created, it will include project\_db.sqlite3 which contains all of our databases.
  - c. The databases include the following classes: User, Subjects, Quizzes, Scores and Questions.
6. static/
  - a. The static directory includes an images directory to store all image/svg files we wish to display on the webpages.
  - b. The static directory also contains the styles.css file meant to apply styling to the HTML5 templates on top of the bootstrap library.
7. .env
  - a. This is used to store all the sensitive environment variables.
8. .gitignore
  - a. This includes files that we do not wish to push to GitHub.
9. requirements.txt
  - a. requirements.txt includes all the frameworks/libraries that're required to run the app.py file.
  - b. It stores the names and versions of all project dependencies.
10. ReadMe.md
  - a. This is a file written in markdown providing a brief description of the Project.

## Features


1. The following webpages are available before the session stores any user data:
  - a. User Signup: This webpage provides the feature to add your record to the Users database.
  - b. User Login: This webpage provides the feature of verifying your credentials from the Users database. If the credentials positively match, user data is stored in the session to dynamically render the rest of the webpages.
  - c. Admin Signup/Login: Admins CANNOT signup. A 'superadmin' role is added to the Users database when the script is executed for the first time. Only superadmin has the access to add 'admin' role users to the Users database.
2. Features provided to Users:
  - a. Users can view the available quizzes on their Dashboard.
  - b. Users have the option to view details of any quiz visible to them on the Dashboard.



- c. Users can start a quiz that leads them to the quiz.html webpage. This webpage dynamically renders the questions of the respective quiz. The Quiz page displays details of the Quiz. The sidebar on the left provides Question Titles of all the Questions belonging to the Quiz. The right panel displays the Question and its Options. Users can select the options and submit the Quiz. The performance is evaluated and the User is redirected to the Scores page.
  - d. Users can access the Scores page to see their past performances including attempted options vs correct options and calculated percentages.
  - e. Users can access the Summary page where they can graphically visualize their User Activity.
3. Features provided to Administrators:
- a. Admins can view Quizzes on their Dashboard grouped by Subjects.
  - b. Admins can perform CRUD operations such as CREATE a new Subject and UPDATE or DELETE existing Subjects.
  - c. Admin can access the Quizzes page where they can view Questions grouped by Quizzes.
  - d. Admins can perform CRUD operations such as CREATE a new Quiz.
  - e. Admins can perform CRUD operations such as CREATE a new Question under a Quiz and UPDATE or DELETE an existing Question.
  - f. Admins can access the Summary page where they can graphically visualize the Platform Activity.
4. Features exclusive to Super Administrator:
- a. There is only one 'superadmin' role in the Users database, created when the script is executed for the first time.
  - b. Only a Superadmin can view the Give Access page to add a record to the Users database with the 'admin' role.
5. The web application also consists of a Logout button that clears all session variables. It will redirect the user to the Signup page if attempts are made to access any of the routes without having logged in (and having verified their 'role') first.

## Video Demonstration

[https://drive.google.com/drive/folders/120ESURQwZ1djniUnRX1Jull0\\_GWbOAii?usp=sharing](https://drive.google.com/drive/folders/120ESURQwZ1djniUnRX1Jull0_GWbOAii?usp=sharing)

Or click:  Salt & Prepper