**Author** Pankaj Mohan Sahu
Roll Number: 21f2001203
Email: 21f2001203@ds.study.iitm.ac.in

A dedicated student passionate about backend systems, data modeling, and building user-centric web applications. Enthusiastic about applying software development principles in real-world projects.

**Description** Quiz Master - V2 is a multi-user exam preparation web application that enables an admin (Quiz Master) to manage subjects, chapters, and quizzes, while users can register, log in, attempt quizzes, and view performance insights. The project emphasizes async task handling, role-based access, performance, and caching.

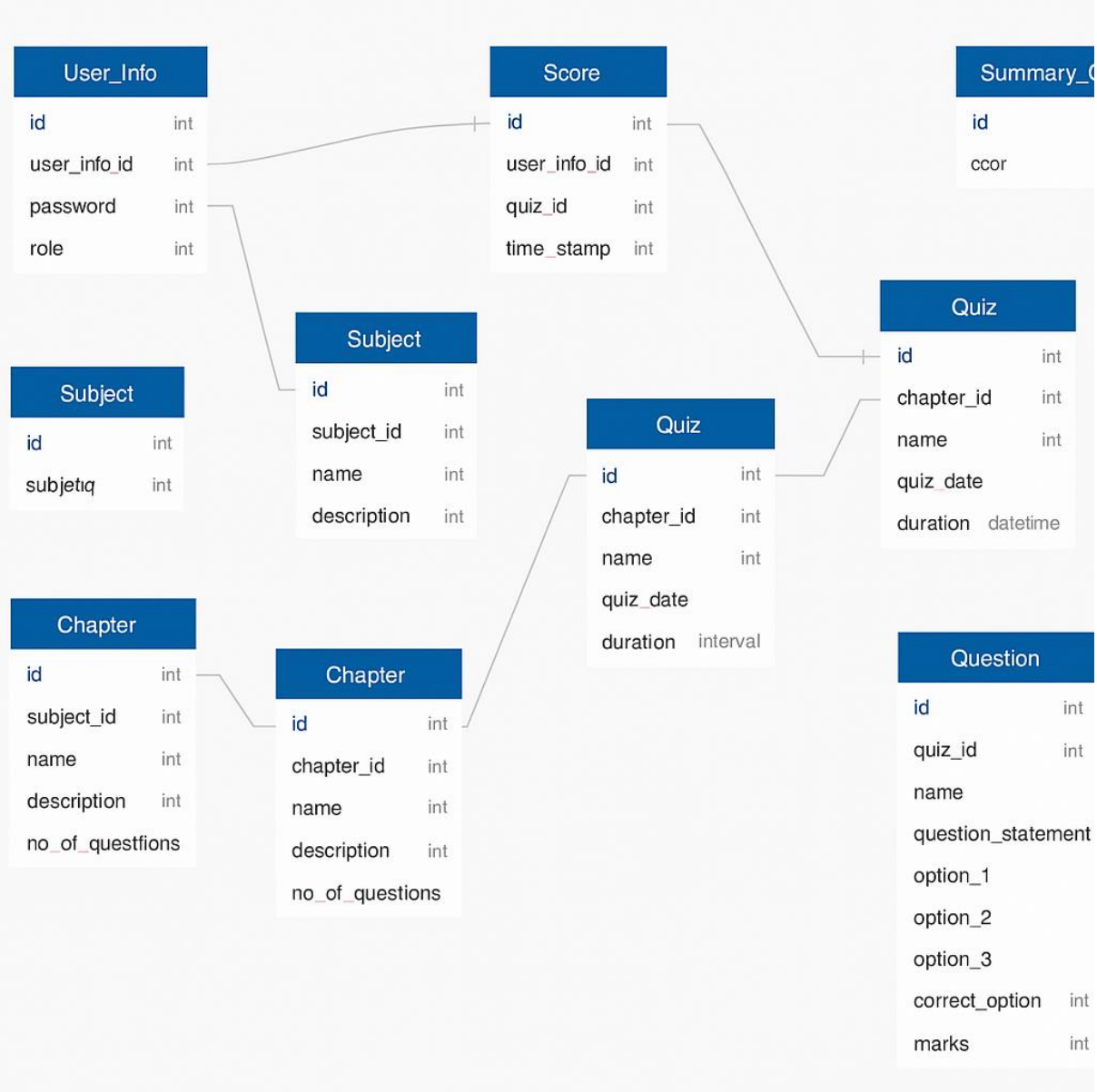
**Technologies Used**

- Flask (API framework)
- Flask-Security (User Authentication & Role Management)
- Flask-Mail (Email support)
- Flask-Caching with Redis (Caching)
- SQLite (Database)
- Celery (Async task queue)
- Redis (Message broker and caching backend)
- Vue.js (Frontend framework)
- Bootstrap (Styling and layout)
- Jinja2 (HTML email templates)
- Chart.js (Frontend chart rendering)

**DB Schema Design**

- **User:** id, email, username, password, fs_uniquifier, active (related to Role via UsersRoles)
- **User_Info:** id, email, password, role, full_name, qualification, date_of_birth (related to Score, Summary_Charts)
- **Role / UsersRoles:** Role-based access implementation
- **Subject:** id, name, description (related to Chapters)
- **Chapter:** id, name, description, no_of_questions, subject_id (related to Quizzes)
- **Quiz:** id, name, chapter_id, quiz_date, duration_time (related to Questions and Scores)
- **Question:** id, quiz_id, question_statement, options, correct_option, marks
- **Score:** id, user_id, quiz_id, total_score, time_stamp_of_attempt

This schema is normalized and supports flexibility in tracking multiple quiz attempts and organizing content hierarchically.




**API Design** RESTful API endpoints were created for managing subjects, chapters, quizzes, questions, quiz attempts, score tracking, and performance summaries. Endpoints are protected via role-based access control using `@auth_required` and `@roles_required` decorators. The APIs are integrated with the frontend using Vue.js `fetch` calls.

**Architecture and Features** The application is modular:

- `models.py` contains ORM mappings
- `routes.py` handles Flask routes for user registration, login, and admin-triggered batch jobs
- `resources.py` contains all RESTful API resources
- `tasks.py` defines background jobs using Celery
- `app.py` bootstraps the app, Flask extensions, and sets periodic Celery tasks
- `Admin.js` and other Vue components power the frontend with dynamic dashboards

**Key Features Implemented:**

- Admin and user login with Flask-Security
- Admin dashboard to manage subjects, chapters, quizzes
- Quiz attempt recording with scores
- Summary statistics and charts per user and global (admin)
- CSV export (async) for user-specific or all users' performance (admin only)
- Daily and monthly reminder reports (mail + gchat)
- Redis-based caching for performance-sensitive endpoints with expiry

**Video**
https://drive.google.com/file/d/1VMafWwAhRk0MKcuWjIRsGwJRYDSJW2MP/view?usp=sharing

---