Author

Name	**Jishnu Jayaprakash**
Roll No	**21F1006194**
Email	21f1006194@ds.study.iitm.ac.in

I'm Jishnu Jayaprakash, an ECE graduate from College of Engineering Chengannur, software developer and CTO at Skillecta.ai. I build scalable projects and drive product development end-to-end.

Description

Problem Statement
Build a Flask + Vue web app where an admin creates subjects, chapters, quizzes (MCQs), registered users take timed quizzes, view past scores and performance charts. Scheduled jobs send daily reminders and monthly activity reports; users can export quiz data as CSV.

AI/LLM Usage
• AI assistance used for CSS styling, code-completion and some debugging.(~10%)
• No LLM in core logic.

Technologies Used

Technology	Purpose
Flask	Core backend framework
Flask-SQLAlchemy	ORM for models and migrations
Flask-Migrate	Alembic-powered DB schema management
Flask-JWT-Extended	JWT-based auth for role-protected APIs
Flask-Mail	Sending reminder and report emails
Flask-Caching	Cache service functions

Flask-sse	Enables server-sent events to notify users in real-time when their background CSV export completes.
Celery	Async tasks & periodic jobs (daily reminders, monthly reports, CSV exports)
Redis	Celery broker & Flask cache backend
SQLite	Lightweight relational datastore
Vue.js	Frontend SPA framework
Pinia	Frontend state management
Vue-Router	SPA navigation
Chart.js	Rendering performance and summary charts
Vue-Quill	WYSIWYG editor for rich question input
Docker & Docker Compose	Containerize and orchestrate backend and frontend services for consistent, easy demo.

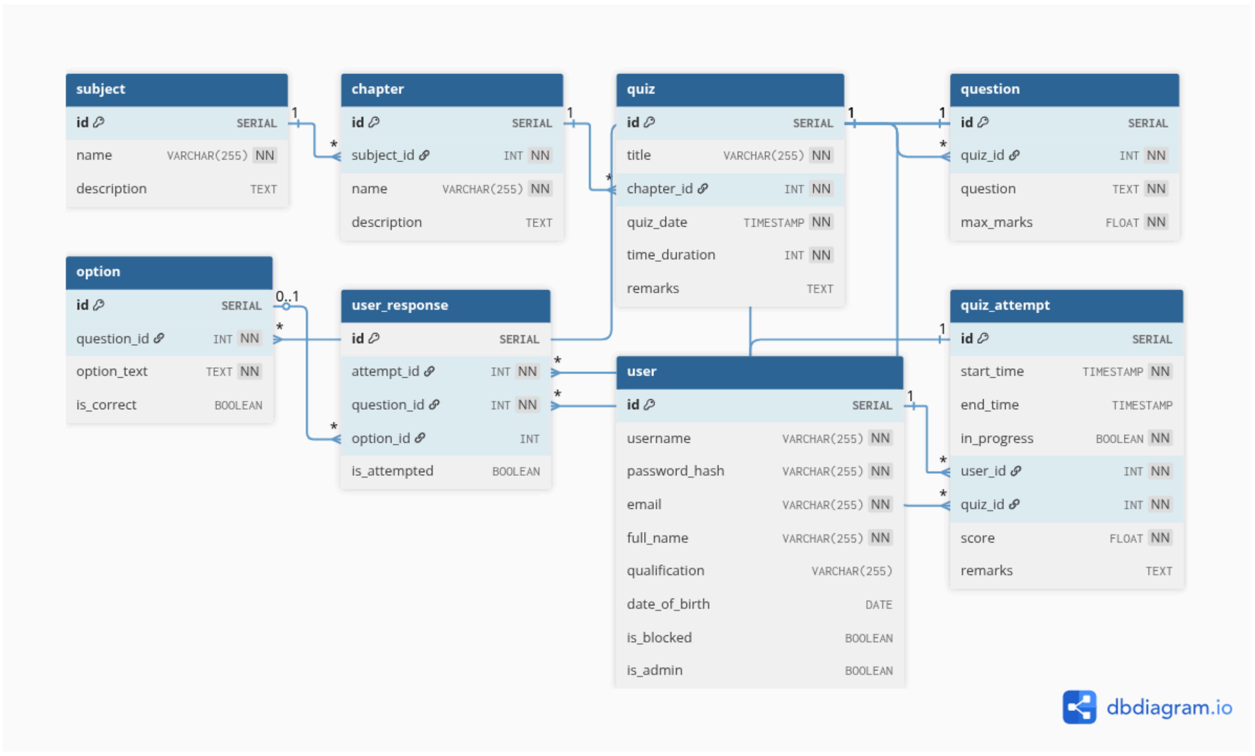
API Design

- **Auth**: `POST /api/register`, `POST /api/login`
- **Admin CRUD** (prefixed `/admin/api`):
  - **Subjects**: `GET/POST /subjects`, `GET/PUT/DELETE /subjects/{id}`
  - **Chapters**: `GET/POST /subjects/{id}/chapters`, `GET/PUT/DELETE /chapters/{id}`
  - **Quizzes**: `GET /quizzes`, `GET/POST /chapters/{id}/quiz`, `GET/PUT/DELETE /quiz/{id, GET /quiz_summary`
  - **Questions**: `GET/POST /quiz/{id}/questions`, `GET/PUT/DELETE /questions/{id}`
  - **Users**: `GET /all_users, PUT /block_user/{user_id}, PUT /unblock_user/{user_id}`
- **User APIs** (prefixed `/user/api`):
  - `GET /profileinfo`, `/quizzes, /quiz/{id}/questions`
  - `GET /subjects, /quiz/{quiz_id}/result, /quiz/history`
  - `GET /quiz/history/download, /quiz/{quiz_id}/leaderboard`
  - `POST/PUT /quiz/{id}` , `GET /download/{filename}`
  - `POST/GET /quiz/{id}/attempt` (answer & fetch responses)

NOTE: *All endpoints secured via JWT. POSTMAN Collection included in the project repo.*

DB Schema Design

- **user** (`id`, `username`, `password_hash`, `email`, `full_name`, `qualification`, `date_of_birth`, `is_blocked`, `is_admin`)
- **subject** (`id`, `name` , `description`)
- **chapter** (`id`, `subject_id` , `name`, `description`)
- **quiz** (`id` , `chapter_id` , `title`, `quiz_date`, `time_duration`, `remarks`)
- **question** (`id` , `quiz_id` , `question`, `max_marks`)
- **option** (`id` , `question_id` , `option_text`, `is_correct`)
- **quiz_attempt** (`id` , `user_id` , `quiz_id` , `start_time`, `end_time`, `in_progress`, `score`, `remarks`)
- **user_response** (`id` , `attempt_id` , `question_id` , `option_id` , `is_attempted`)



Architecture & Features

Project Layout

- **Backend** (`/backend/app`):
    - `models/`: SQLAlchemy models for all entities
    - `routes/`: Flask-RESTful resources (`admin.py`, `user.py`, `public.py`, etc.)
    - `services/`: Business logic layer split into `admin`, `catalog`, `user` services
    - `tasks/`: Celery tasks for scheduled and user-triggered jobs
    - `utils/`: Commons, constants and mail helper
    - `config.py` & `celery_app.py` for environment and Celery setup
- **Frontend** (`/frontend/src`):
    - `views/`: Page components (AdminDashboard, UserDashboard, AttemptQuiz…)
    - `components/`: Reusable UI pieces & modals
    - `stores/`: Pinia auth and state stores
    - `api.js`: Axios instances with JWT interceptor
    - `router/index.js`: Route definitions

Features Implemented

- **Core**: Role-based JWT auth; full CRUD for subjects/chapters/quizzes/questions; timed quiz .
- **Reporting**: Daily email reminders; monthly HTML reports via Celery + Flask-Mail.
- **Exports**: CSV exports for users (async jobs with Redis/Celery + SSE push to frontend).
- **Performance**: Flask-Cachingon some service functions.

- **Extras**: WYSIWYG question input (Vue-Quill), advanced search functionality, Analytics and Leaderboard

Video Presentation Link:
https://drive.google.com/file/d/1nBGLb3PBvTLOJggA4LSb8WAe7s2rETYK/view?usp=sharing