Quiz Master V2 - Project Report

Modern Application Development II

Student Details
Student Name: Abhist Jain
Roll Number: 21f3000201
Course: Modern Application Development II
Project: Quiz Master V2

Project Overview:

Problem Statement
Quiz Master V2 is a comprehensive multi-user quiz management application designed as an exam preparation platform for multiple courses. The application serves two primary user roles: administrators (quiz masters) who manage content and users who attempt quizzes.

Approach to Problem Statement:

The project was approached using a modern full-stack architecture with clear separation of concerns:

1. Backend-First Development: Started with designing the database schema and REST API endpoints
2. Role-Based Architecture: Implemented distinct admin and user workflows with appropriate access controls
3. Scalable Design: Used Redis caching and Celery background jobs for performance optimization
4. Responsive Frontend: Built a Vue.js SPA with Bootstrap for cross-device compatibility
5. Automated Background Jobs: Implemented scheduled reminders and report

generation

Technology Stack & Frameworks:

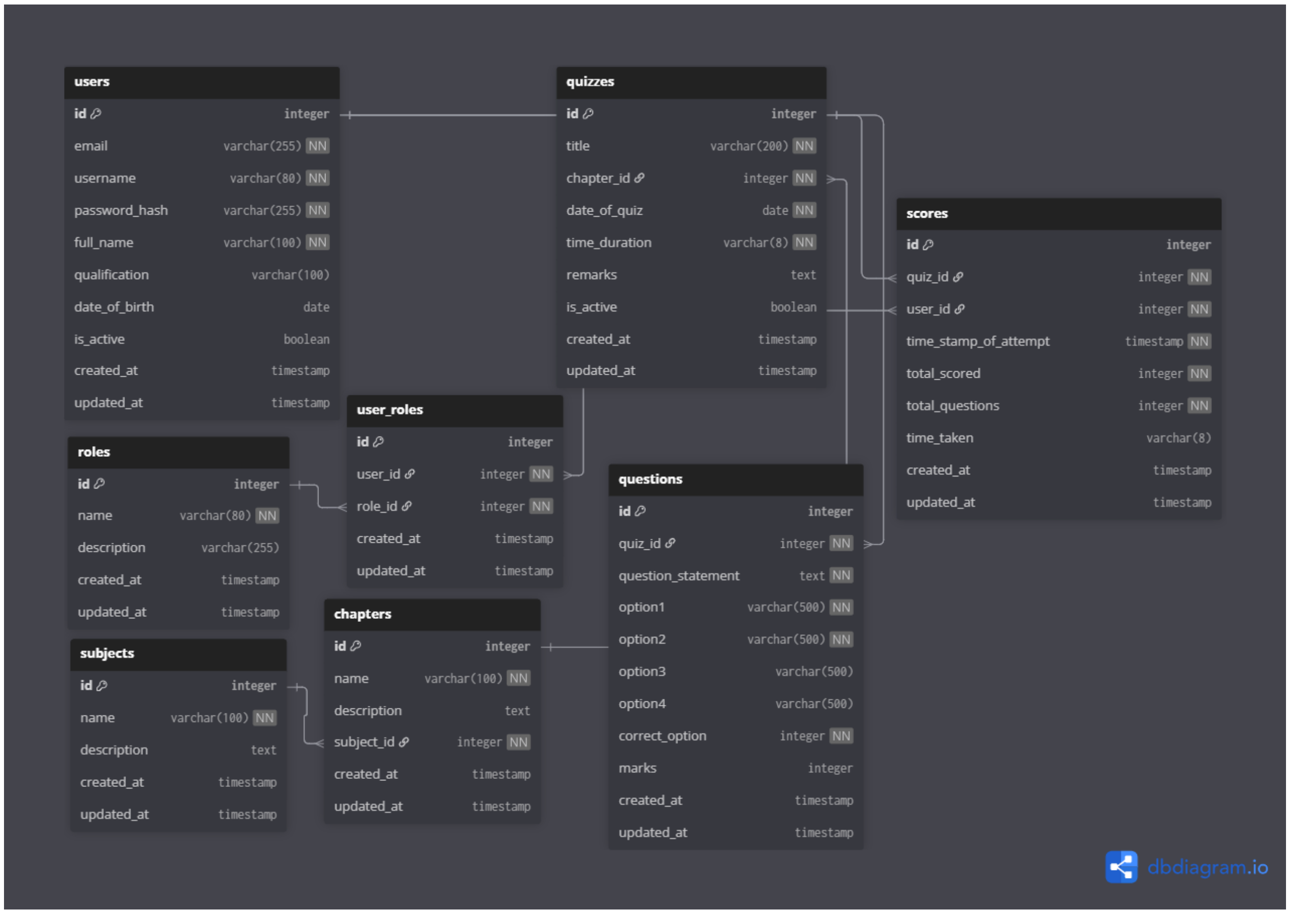
Mandatory Frameworks (As Required)
- Database: SQLite with SQLAlchemy ORM
- Backend API: Flask with Flask-RESTful
- Frontend UI: Vue.js 3 with Vue CLI
- Styling: Bootstrap 5 (No other CSS frameworks used)
- Caching: Redis for performance optimization
- Background Jobs: Redis + Celery for batch processing

Additional Libraries & Tools
- Authentication: Flask-JWT-Extended for token-based auth
- Charts: Chart.js for dashboard visualizations
- Email: Flask-Mail for notifications
- HTTP Client: Axios for API communication
- Development: Vite for frontend build tooling

Database Schema & ER Diagram:

Entity Relationship Overview

The database follows a hierarchical structure: Subject → Chapter → Quiz → Questions



users	
id	integer
email	varchar(255) NN
username	varchar(80) NN
password_hash	varchar(255) NN
full_name	varchar(100) NN
qualification	varchar(100)
date_of_birth	date
is_active	boolean
created_at	timestamp
updated_at	timestamp

quizzes	
id	integer
title	varchar(200) NN
chapter_id	integer NN
date_of_quiz	date NN
time_duration	varchar(8) NN
remarks	text
is_active	boolean
created_at	timestamp
updated_at	timestamp

scores	
id	integer
quiz_id	integer NN
user_id	integer NN
time_stamp_of_attempt	timestamp NN
total_scored	integer NN
total_questions	integer NN
time_taken	varchar(8)
created_at	timestamp
updated_at	timestamp

user_roles	
id	integer
user_id	integer NN
role_id	integer NN
created_at	timestamp
updated_at	timestamp

roles	
id	integer
name	varchar(80) NN
description	varchar(255)
created_at	timestamp
updated_at	timestamp

questions	
id	integer
quiz_id	integer NN
question_statement	text NN
option1	varchar(500) NN
option2	varchar(500) NN
option3	varchar(500)
option4	varchar(500)
correct_option	integer NN
marks	integer
created_at	timestamp
updated_at	timestamp

subjects	
id	integer
name	varchar(100) NN
description	text
created_at	timestamp
updated_at	timestamp

chapters	
id	integer
name	varchar(100) NN
description	text
subject_id	integer NN
created_at	timestamp
updated_at	timestamp

Core Entities

1. Users – Student/admin accounts with role-based access
2. Roles – Admin/user role definitions
3. Subjects – Top-level academic subjects
4. Chapters – Subdivisions within subjects
5. Quizzes – Timed assessments within chapters
6. Questions – MCQ questions within quizzes
7. Scores – User quiz attempt results

Key Relationships

- Users ↔ Roles (Many-to-Many via user_roles junction table)
- Subjects → Chapters (One-to-Many)
- Chapters → Quizzes (One-to-Many)
- Quizzes → Questions (One-to-Many)

- Users → Scores (One-to-Many)
- Quizzes → Scores (One-to-Many)

API Resource Endpoints:

Authentication Endpoints

POST	/api/auth/login	- User/Admin login
POST	/api/auth/register	- User registration
POST	/api/auth/refresh	- Token refresh
POST	/api/auth/logout	- User logout
GET	/api/auth/profile	- Get user profile

Subject Management (Admin)

GET	/api/subjects	- Get all subjects
GET	/api/subjects/<id>	- Get specific subject
POST	/api/subjects	- Create new subject
PUT	/api/subjects/<id>	- Update subject
DELETE	/api/subjects/<id>	- Delete subject

Chapter Management (Admin)

GET	/api/chapters	- Get all chapters
GET	/api/chapters/<id>	- Get specific chapter
POST	/api/chapters	- Create new chapter
PUT	/api/chapters/<id>	- Update chapter
DELETE	/api/chapters/<id>	- Delete chapter

Quiz Management

GET	/api/quizzes	- Get all quizzes
GET	/api/quizzes/<id>	- Get specific quiz
POST	/api/quizzes	- Create new quiz (Admin)
PUT	/api/quizzes/<id>	- Update quiz (Admin)
DELETE	/api/quizzes/<id>	- Delete quiz (Admin)

Question Management (Admin)

GET	/api/questions	- Get all questions
GET	/api/questions/<id>	- Get specific question
POST	/api/questions	- Create new question
PUT	/api/questions/<id>	- Update question
DELETE	/api/questions/<id>	- Delete question

Quiz Attempts & Scoring

GET	/api/scores	- Get user scores
POST	/api/quiz-attempt/<id>	- Submit quiz attempt
GET	/api/scores/<id>	- Get specific score

User Management (Admin)

GET	/api/users	- Get all users
GET	/api/users/<id>	- Get specific user
PUT	/api/users/<id>	- Update user
DELETE	/api/users/<id>	- Delete user

Dashboard & Analytics

GET	/api/dashboard	- Get dashboard data (role-specific)
GET	/api/search	- Global search (Admin)

Background Jobs & Export

POST	/api/jobs/export/user	- Trigger user CSV export
POST	/api/jobs/export/admin	- Trigger admin CSV export
GET	/api/jobs/status/<id>	- Check job status
GET	/api/jobs/download/<file>	- Download generated CSV

Core Features Implemented

Admin Features

- Pre-seeded admin account (admin@quizmaster.com)
- Subject/Chapter/Quiz/Question CRUD operations
- User management and monitoring
- Global search across all entities
- Admin dashboard with comprehensive analytics
- CSV export of all user data

User Features

- User registration and authentication
- Quiz browsing and filtering
- Timer-based quiz attempts

- Score tracking and history
- User dashboard with performance analytics
- Personal CSV export of quiz attempts

Background Jobs (Celery)

- Daily reminders: Automated email notifications to inactive users
- Monthly reports: HTML/PDF activity reports sent via email
- CSV export: Asynchronous data export with job status tracking

Performance & Caching

- Redis caching for frequently accessed data
- Cache expiry policies (5–15 minutes)
- Optimized database queries with eager loading
- API response caching for dashboard data

Video Link: Link