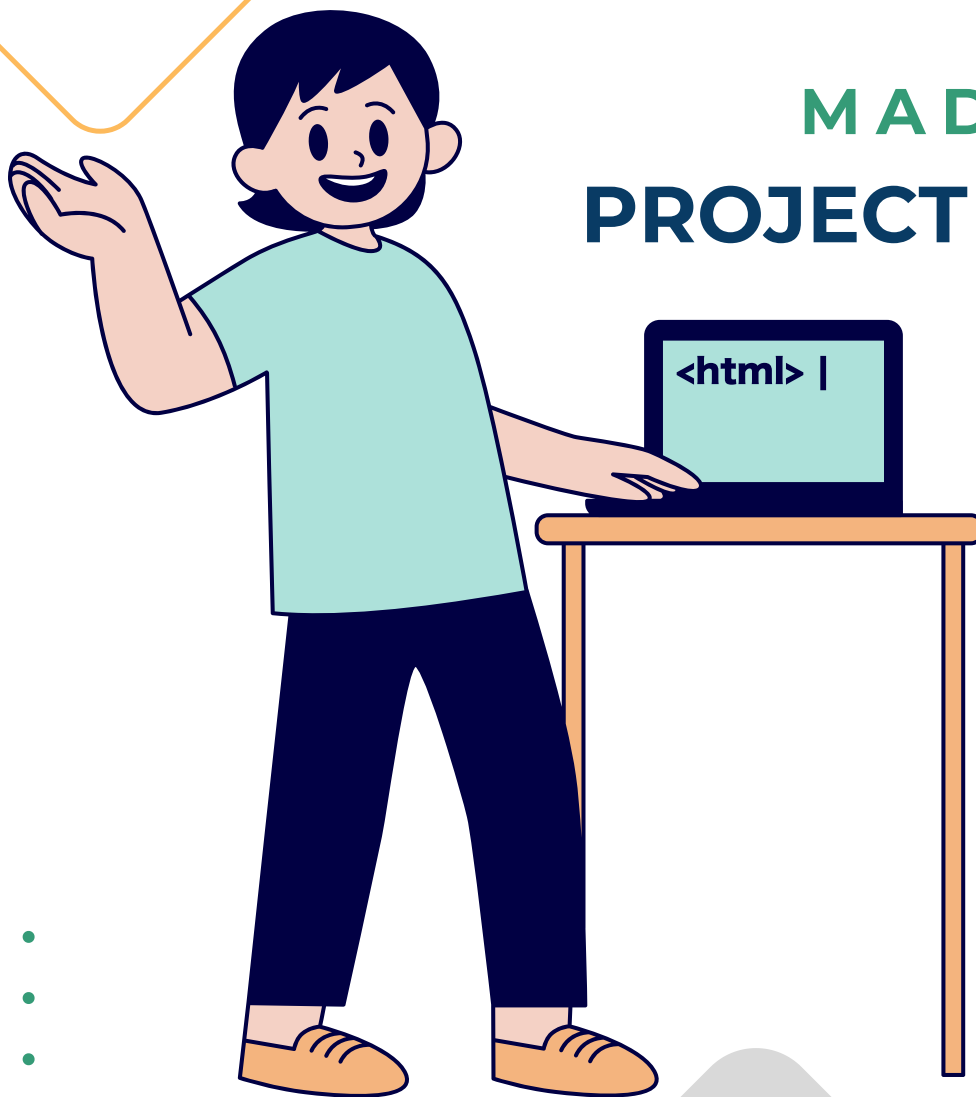




StudyBug



MAD-I PROJECT REPORT



Prepared by :
Shrestha Srivastava








About Me

New Tab

+

< > 🔍 Shrestha Srivastava ↻ ≡

Name	Shrestha Srivastava
Roll Number	23F3000168
Student Mail	23F3000168@ds.study.iitm.ac.in
  	@Srivastava-Shrestha

Project Details

☐ StudyBug : Quiz Master ✕

StudyBug is a feature-rich, multi-user quiz platform designed to help students efficiently prepare for exams across various subjects. The application ensures a structured learning environment where users can enroll in subjects, review chapters, and attempt quizzes while an admin manages the content. This system not only provides quiz-based learning but also ensures real-time evaluation, user progress tracking, and an intuitive user experience. With a well-organized interface, StudyBug allows seamless interaction between students and the quiz system to enhance learning outcomes.



TECHNOLOGIES USED

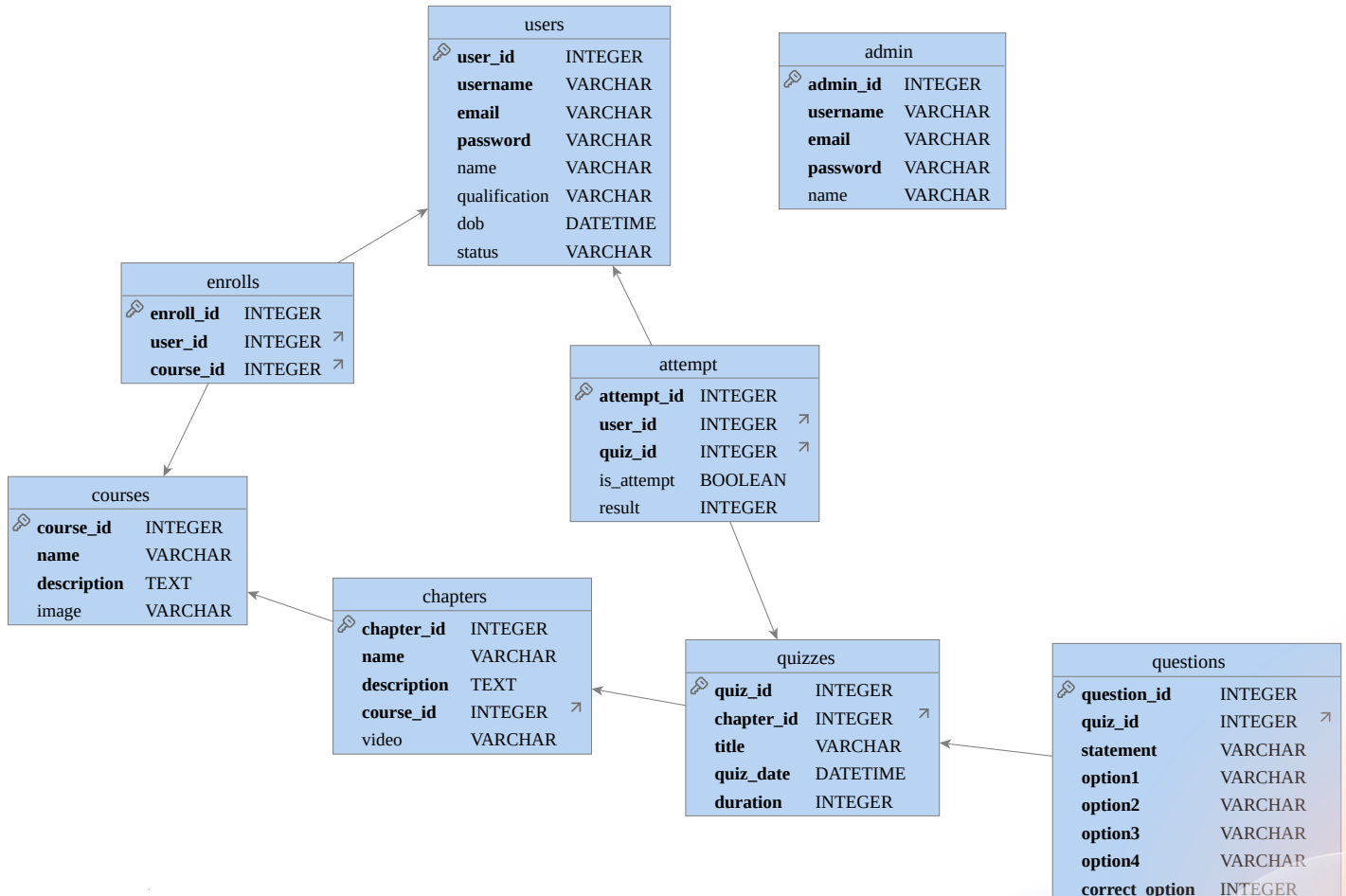
- **Flask:** A lightweight Python web framework.
- **Jinja2:** A templating engine for generating HTML.
- **Flask-SQLAlchemy:** Adds SQLAlchemy support for database management in Flask.
- **Flask-RESTful:** Simplifies building REST APIs in Flask.
- **Requests:** A library for making HTTP requests in Python.
- **SQLite:** A lightweight, disk-based database.
- **HTML:** Markup language for creating web pages.
- **CSS:** Stylesheet language for designing web pages.
- **JavaScript:** Programming language for interactive web content.
- **Bootstrap:** Framework for responsive, mobile-first web design.
- **Chart.js:** Library for creating interactive charts.



DB Schema



The StudyBug database schema is designed to manage a structured quiz platform efficiently. The Users table stores registered user details, allowing them to enroll in courses and take quizzes, while the Admin is a predefined entity with full control over the system. The Courses table holds subject details, each connected to multiple Chapters that break down topics for better learning. The Quizzes table links to chapters and includes details like duration and availability, while the Questions table stores MCQs with correct answers. The Enrollments table tracks users joining courses, and the Attempts table records quiz submissions, scores, and timestamps. This schema ensures well-organized content management, structured learning, and accurate progress tracking, making StudyBug a scalable and efficient exam preparation system.





Directory Structure



```
StudyBug
├── api
│   ├── chapter.py
│   ├── course.py
│   ├── __init__.py
│   ├── question.py
│   └── quiz.py
├── config.py
├── __init__.py
├── models
│   ├── admin.py
│   ├── attempt.py
│   ├── chapter.py
│   ├── course.py
│   ├── enroll.py
│   ├── __init__.py
│   ├── question.py
│   ├── quiz.py
│   └── user.py
├── routes
│   ├── auth.py
│   ├── dashboard.py
│   ├── home.py
│   └── __init__.py
├── static
│   ├── css
│   ├── font
│   ├── image
│   └── js
├── templates
│   ├── 404.html
│   ├── admin_base.html
│   ├── admin_chapter.html
│   ├── admin_course.html
│   ├── admin_dashboard.html
│   ├── admin_questions.html
│   ├── admin_quiz.html
│   ├── admin_user.html
│   ├── home.html
│   ├── login.html
│   ├── signup.html
│   ├── user_attempt.html
│   ├── user_base.html
│   ├── user_course_detail.html
│   ├── user_course.html
│   ├── user_dashboard.html
│   ├── user_feedback.html
│   ├── user_quiz.html
│   └── user_result.html
├── utils.py
├── instance
│   └── StudyBug.db
├── README.md
├── requirements.txt
├── run.py
└── Project Report.pdf
```



API Resource



1. Course API

Endpoint: /api/course, /api/course/<int:id>

- **GET:** Retrieve all courses or a specific course by ID.
- **POST:** Create a new course.
- **PUT:** Update an existing course by ID.
- **DELETE:** Delete a course by ID.

2. Chapter API

Endpoint: /api/chapter, /api/chapter/<int:chapter_id>

- **GET:** Retrieve all chapters or a specific chapter by ID.
- **POST:** Create a new chapter.
- **PUT:** Update an existing chapter by ID.
- **DELETE:** Delete a chapter by ID.

3. Quiz API

Endpoint: /api/quiz, /api/quiz/<int:quiz_id>

- **GET:** Retrieve all quizzes or a specific quiz by ID.
- **POST:** Create a new quiz.
- **PUT:** Update an existing quiz by ID.
- **DELETE:** Delete a quiz by ID.

4. Question API

Endpoint: /api/question, /api/question/<int:question_id>

- **GET:** Retrieve all questions or a specific question by ID.
- **POST:** Create a new question.
- **PUT:** Update an existing question by ID.
- **DELETE:** Delete a question by ID.



Presentation Video



Youtube : <https://youtu.be/lq1KsgAVZAI>

Google Drive :

<https://drive.google.com/file/d/1wWr6dtRciizkVFLZuuna4XKoM1kQCZqZ/view?usp=sharing>



**Thank You
For Your Attention!**

Shrestha Srivastava

29 March 2025 →