Back End Engineering-II (23CS008)

Project Report

Semester- 5 (Batch2023)

Student Learning Platform

Supervised By:
Dr. Muskan Chawla
(Assistant Professor)

Submitted By:
Ramanjot, 2310992189 (G-25)
Shefali, 2310992208 (G-25)
Shiwangi, 2310992210 (G-25)
Teesha, 2310991069 (G-25)
Vanshika, 2310990912 (G-25)

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

Abstract

The way in which education is provided is changing fast due to online learning, creating a demand for effective, secure, and scalable online platforms. Edemy is a backend service that I developed for this e-learning program to provide user management, course management, and administrative management capabilities.

The backend services create RESTful APIs for user authentication, enrollment, and course features, as well as an admin dashboard built in EJS that displays real-time statistics and allows admins to run the platform efficiently.

The system was developed with Node.js/Express.js, MongoDB Atlas, and Mongoose. This ensures incredibly scalable and performant services. Using modular routing, middleware-based request validation, and a cloud-based database allows this project to address the needs that online education platforms face in a reliable, secure, and extensible way.

Table of Contents

1. Introduction

1.1 Background

The growth of the internet and digital technologies has revolutionized the education industry and contributed to the speedy acceptance of e-learning platforms worldwide, with more students and instructors having an array of choices to suit their flexible learning, accessibility, and personalized choices. However, with this increased demand ultimately comes great responsibility, leading to a commensurate requirement for sound backend systems that provide features that allow the management of multiple users, courses, and data security.

Traditional learning management systems usually face challenges related to scalability, security, and administrative control. In response to these challenges, many learning management systems are built with common modern web technologies, such as Node.js, Express.js, MongoDB Atlas, and Mongoose. These common technologies deliver high performance, flexibility, and cloud-level reliability, thus making them a viable technology stack for building scalable e-learning platforms.

In this context, the Edemy backend project is proposed as a secure, modular, and efficient backend solution for managing users, courses, and administrative duties in an online learning environment.

1.2 Objectives

The main objectives of the **Edemy Backend Project** are:

1. **To develop a secure backend system** for managing users, courses, and enrollments in an e-learning platform.
2. **To implement RESTful APIs** that enable seamless integration with frontend clients (web or mobile applications).
3. **To design role-based access control** (student, instructor, admin) ensuring appropriate privileges and data protection.
4. **To build an admin dashboard** using EJS templates for real-time monitoring of users, courses, and enrolment statistics.
5. **To apply middleware** for authentication, error handling, and request validation, ensuring smooth API workflows.

1.3 Significance

The significance of Edemy lies in its contribution to solving real-world challenges in the e-learning ecosystem:

1. Enhanced Learning Experience: By ensuring seamless backend services, students and instructors can interact with the platform efficiently.
2. Scalability for Future Growth: The cloud-hosted database and modular design allow the system to handle increasing users and courses without performance degradation.
3. Administrative Control: The admin dashboard provides real-time insights and management tools, empowering administrators to maintain quality and monitor usage effectively.
4. Security and Reliability: Secure authentication and encrypted data storage protect sensitive information, which is essential for modern online platforms.

2. Problem Definition and Requirements

2.1 Problem Statement

With the rapid growth of online education, there is an increasing demand for robust and scalable e-learning platforms that can efficiently manage large volumes of users, courses, and administrative operations. Existing systems often struggle with secure user authentication, efficient course management, and providing administrators with real-time insights into platform activities.

Educational institutions and startups face challenges such as:

- Managing dynamic course content and multiple user roles (students, instructors, administrators).
- Ensuring secure access control to protect sensitive user and course data.
- Providing an intuitive administrative interface for monitoring platform usage, statistics, and managing resources.
- Maintaining a scalable and reliable backend that can handle growing numbers of users and courses without performance issues.

2.2 Requirements

There is a need for a modular, secure, and scalable backend solution that offers:

1. RESTful APIs for seamless integration with various front-end clients (web/mobile).
2. User and course management features with role-based access.
3. A dynamic admin dashboard for real-time monitoring and resource control.
4. A cloud-hosted, scalable database to ensure reliability and availability.

The Edemy backend project is proposed to address these challenges by providing an efficient e-learning backend system built with Node.js, Express.js, MongoDB Atlas, Mongoose, and EJS.

3. Proposed Design/Methodology

3.1 Overall Workflow

The overall workflow of the system can be summarized as follows:

1. Client Interaction: The frontend (web or mobile) sends requests to the backend APIs.
2. Routing: Express.js routes process these requests and forward them to the respective controllers.
3. Controllers: Controllers contain the business logic to handle user, course, or admin operations.
4. Database Access: Controllers interact with MongoDB Atlas through Mongoose models for CRUD operations.
5. Middleware: Middleware functions are applied for authentication, authorization, logging, and error handling.
6. Responses: The processed data is returned as JSON (for APIs) or rendered views (for the admin dashboard via EJS).

3.2 Module Design

The system is divided into key modules to ensure separation of concerns and modularity:

- User Module
  - Handles registration, login, authentication, and profile management.
  - Implements role-based access control (student, instructor, admin).
- Course Module
  - Manages course creation, updating, deletion, and retrieval.
  - Supports enrollment functionality for students.
- Enrollment Module
  - Maintains mapping between users and courses.
  - Tracks enrollment date and progress.
- Admin Module
  - Provides an EJS-based dashboard for monitoring platform statistics.
  - Enables administrators to manage users and courses.
- Middleware Module
  - Includes authentication middleware , error handling, and validation middleware.

3.3 Error Handling & Validation

Error handling and validation are crucial for ensuring a secure and user-friendly experience:

- Input Validation
    - Implemented using middleware (e.g., Joi or custom validators).
    - Ensures fields like email, password, and course details meet required formats.
- Authentication & Authorization Errors
    - Invalid or expired JWT tokens trigger 401 Unauthorized errors.
    - Role-based middleware prevents unauthorized access to admin/instructor routes.
- Database Errors
    - Handled gracefully using try-catch blocks.
    - Duplicate key errors (e.g., duplicate email) are caught and returned with meaningful messages.
- Global Error Handler
    - A centralized Express middleware handles uncaught errors.
    - Ensures consistent JSON error responses across the system.

3.4 Extensibility

The system is designed with extensibility in mind, allowing future enhancements without major architectural changes:

1. Additional Roles: New user roles (e.g., “Moderator” or “Content Reviewer”) can be easily added with role-based middleware.
2. New Features: Modules such as quizzes, assignments, or certification can be integrated into the existing structure.
3. API Expansion: Additional endpoints can be added without disrupting current functionality due to the modular route design.
4. Third-Party Integrations: Services like payment gateways, video hosting, or analytics can be integrated into the system.

4. Results

The project implementation produced the following key outcomes:

- User and Course APIs: CRUD operations for both entities were implemented and tested successfully.
- Authentication: Secure login and token-based session management were established.
- Admin Dashboard: Real-time data visualization enabled efficient platform monitoring.
- Database Scalability: MongoDB Atlas provided fault tolerance and horizontal scalability.
- System Maintainability: Modular architecture allowed separation of concerns, making it easy to extend.
- Performance: APIs responded within milliseconds for typical queries due to optimized indexing and Mongoose queries.

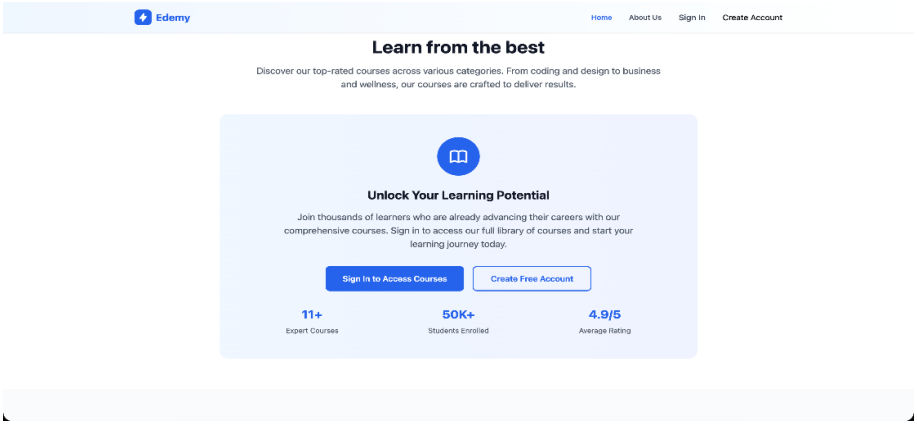

FIG 4.1

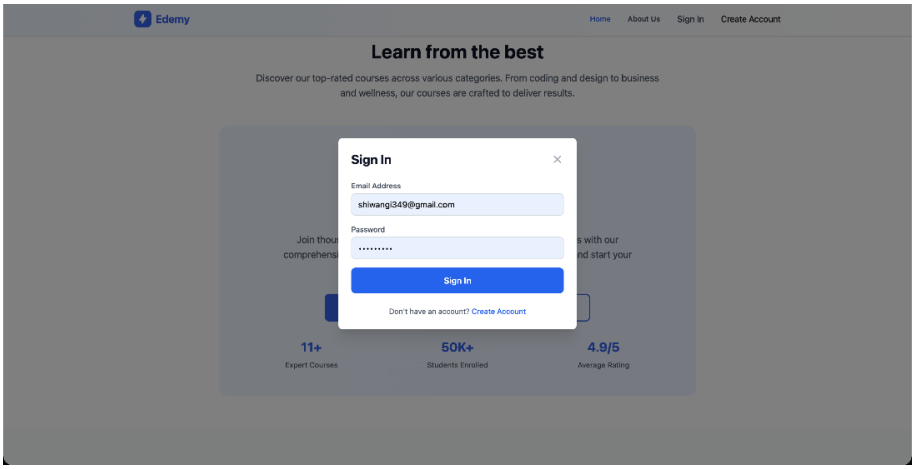

FIG 4.2

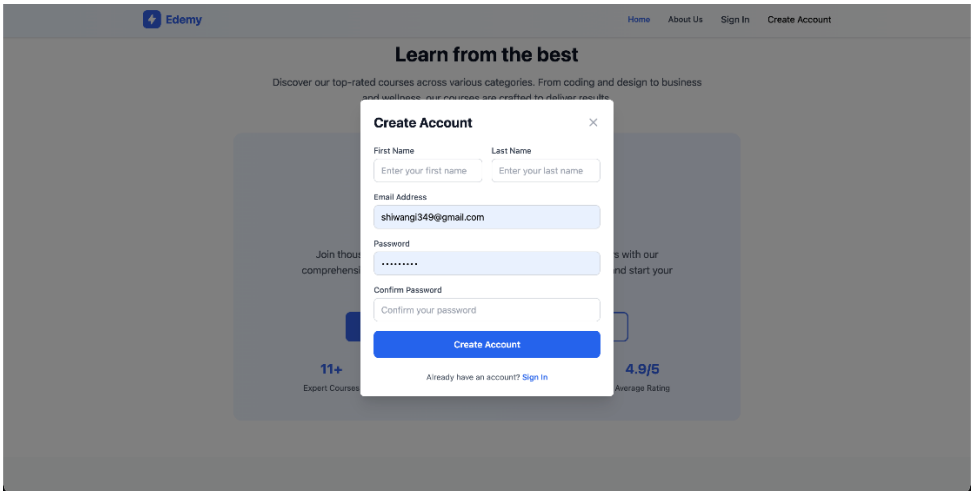

FIG 4.3

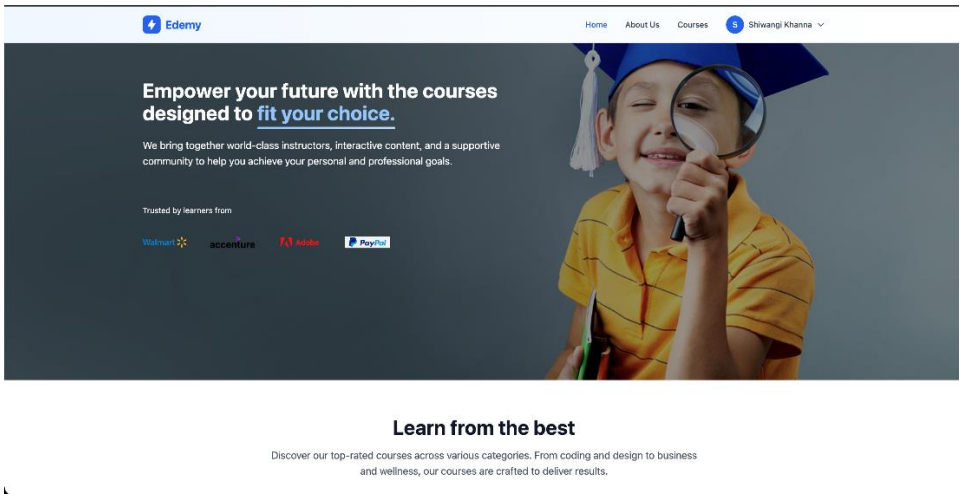

FIG 4.4

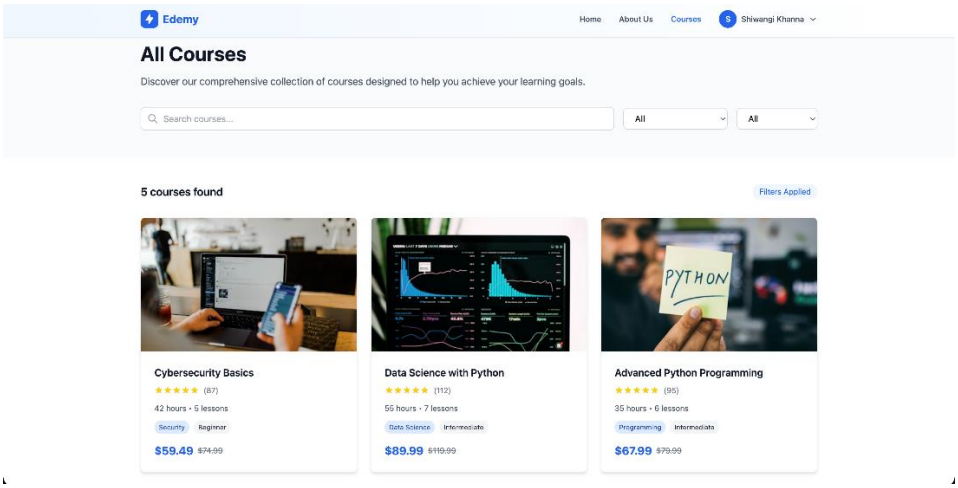

FIG 4.5

FIG 4.6

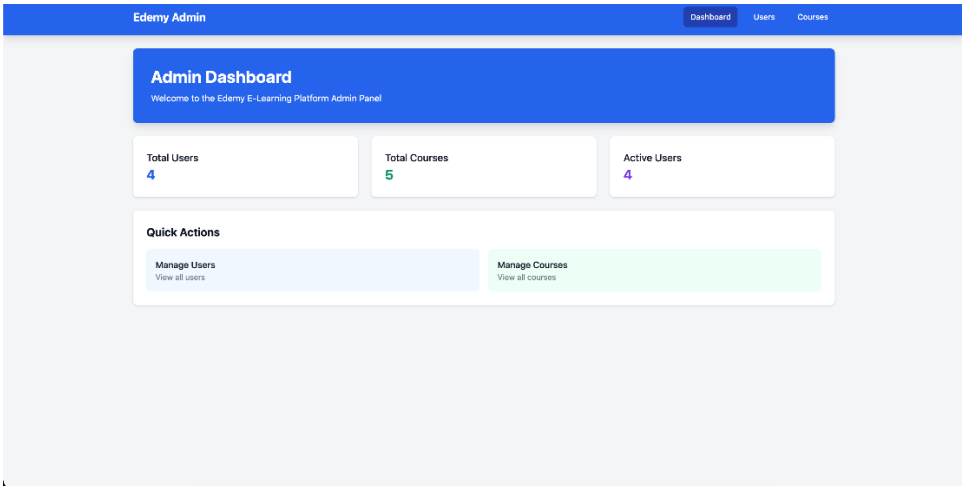

FIG 4.7

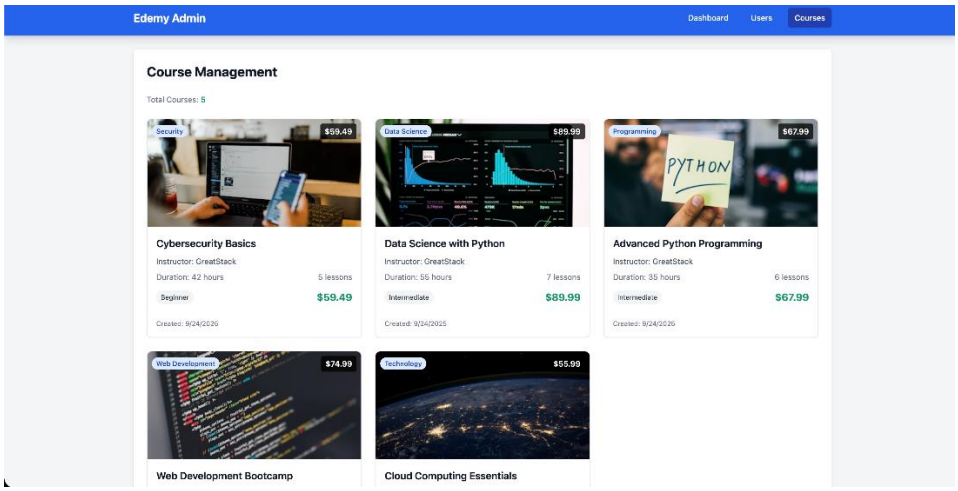

FIG 4.8

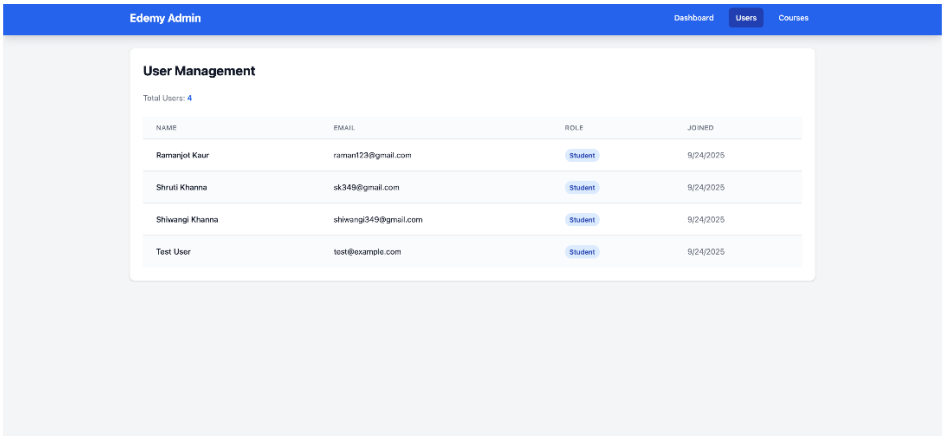

FIG 4.9

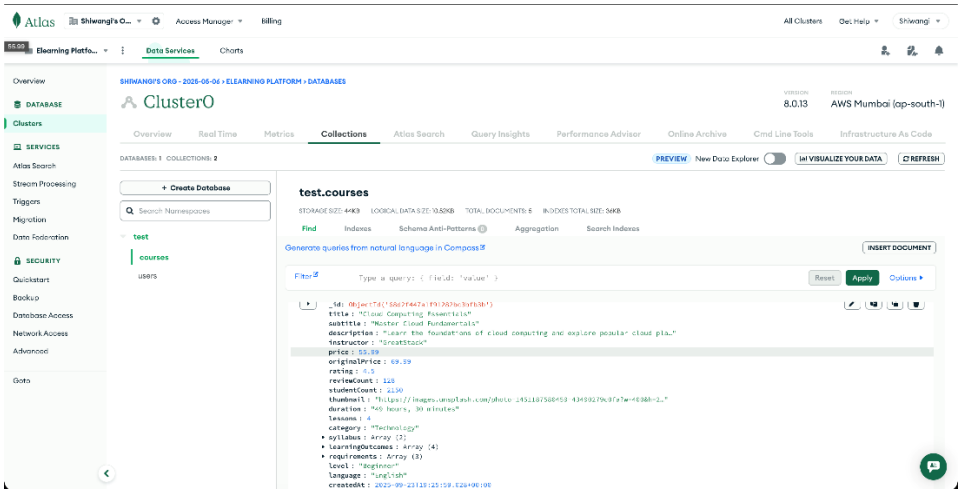

FIG 4.10

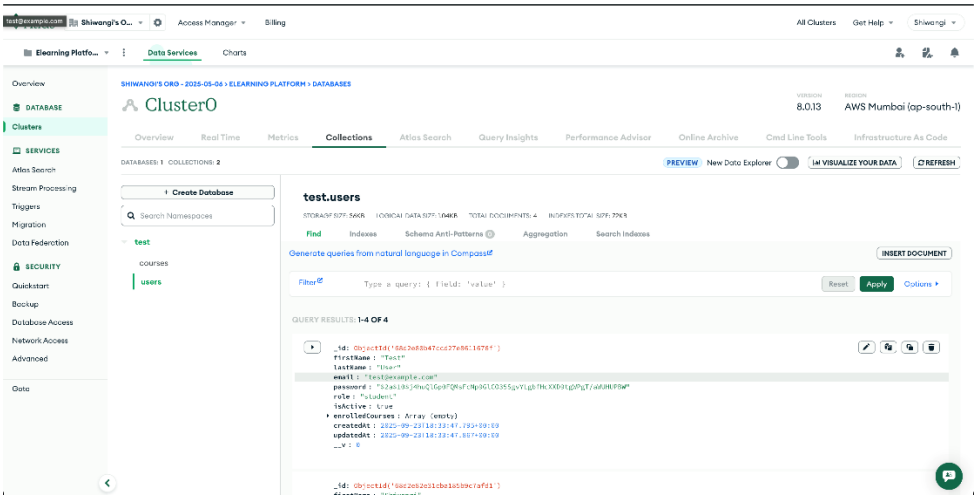

FIG 4.11

5. References

1. Node.js Documentation – https://nodejs.org/docs
2. Express.js Documentation – https://expressjs.com
3. MongoDB Atlas Documentation – https://www.mongodb.com/docs/atlas
4. EJS Documentation – https://ejs.co
5. GeeksforGeeks – Tutorials on Node.js and Express.js