

# CHAPTER 1

## INTRODUCTION

---

### **1.1 Client/Organization Profile:**

**Organization name:** Zeal Institute of Business Administration, Computer Applications and Research

**Location:** Survey No-39, Dhayari Narhe Rd, Narhe, Pune, Maharashtra 411041, India

**Pin-code:** 413001

**Established year:** Year 1996

**About Organization:** The Zeal Institute of Business Administration, Computer Application and Research is Top Management Institute Approved by AICTE, New Delhi, Recognized by DTE Govt. of Maharashtra & Affiliated to Savitribai Phule Pune University. It conducts MBA (300 intake) & MCA (120 intake) programs. The Institute is run by Zeal Education Society, which was established in 1996 and has set up multiple center of educational excellence. We have completed 20 years of presence in academic sector and will continue to redefine excellence in future to remain the Top Management Institute. The Institute strongly believes that world-class education is the stepping-stone to progress. With a longstanding commitment towards quality teaching and learning, the Institute has nurtured values that go into the making of successful careers. We are redefining excellence through relentless efforts and innovative teaching learning by qualified, experienced and dedicated faculties contributing to outstanding academics. At Zeal, we supplement the routine teaching, with training and guidance from Industry experts and renowned academicians from institutions of high repute and Top Management Institute. Institute provides the extra edge through continuous interaction with entrepreneurs and practicing managers. Many leading industries are located in vicinity to boost the Industry – Academia interaction and placements. Amenities like Culture Center, Music Studio, Dance Studio, TED Studio etc. have been created for students to exhibit their talent.

## **1.2 Abstract**

The E-learning Platform is a comprehensive software solution designed to streamline and automate the processes involved in online education and learning management. It aims to enhance the efficiency, accessibility, and effectiveness of educational content delivery while providing a robust platform for both instructors and students. The system offers a range of features to facilitate seamless learning experiences and effective management of courses, assessments, and student progress. It allows users to create and maintain a centralized database of courses, learning materials, and user profiles, enabling easy access to relevant information for educational decision-making. In the course management module, users can create and organize educational content, manage course enrollments, and track student progress. The system provides functionalities for instructor management, including maintaining instructor profiles, evaluating teaching performance, and managing course materials. It also supports content management by enabling users to upload various types of learning materials, track content usage, and automatically generate progress reports based on student engagement.

The assessment module of the system enables instructors to manage the evaluation process from quiz creation to grading. It allows for the creation and management of assessment questions, tracking of student submissions, and generation of performance reports. Additionally, it provides analytics capabilities to analyse learning outcomes, student trends, and course effectiveness. Furthermore, the system incorporates user management features, such as role-based access control, profile management, and communication tools, to ensure secure and efficient platform operation. The platform also includes payment processing capabilities for course subscriptions and premium content access, integrated with secure payment gateways. The system is built with scalability and security in mind, utilizing modern technologies and following best practices in software development. It includes comprehensive testing strategies to ensure reliability and performance, with features for both automated and manual testing processes. This E-learning Platform represents a complete solution for educational institutions and organizations looking to deliver high-quality online education while maintaining robust management and monitoring capabilities.

### **1.3 Existing system and need for system**

The E-learning Platform represents a comprehensive solution in the modern educational landscape, addressing the growing demand for flexible and accessible learning opportunities. This sophisticated web-based learning management system serves as a bridge between educators and learners, providing a robust environment for course creation, delivery, and management. Built with a modern tech stack utilizing React.js for the frontend interface and Spring Boot for the backend services, while MongoDB serves as the primary database, the platform is designed to meet the complex needs of contemporary online education. At its core, the platform supports multiple user roles, including students, instructors, administrators, and super-administrators, each with specific permissions and capabilities. Students can browse and enroll in courses, track their learning progress, and receive certificates upon completion. Instructors have the ability to create and manage courses, upload learning materials, and monitor student progress. Administrators oversee the platform's operations, manage user accounts, and handle course approvals, while super-administrators have additional privileges for managing sales reports and user roles.

This multi-tiered approach ensures efficient platform management and quality control. The system incorporates several key features essential for modern e-learning, including a robust payment processing system integrated with Razorpay, a comprehensive progress tracking mechanism, and an automated certificate generation system. The platform also includes advanced analytics and reporting capabilities, allowing administrators to monitor course performance, track sales, and analyse user engagement. The user interface is designed to be intuitive and responsive, ensuring a seamless learning experience across different devices. The development of this E-learning Platform addresses several critical needs in today's educational landscape. In an era where traditional classroom-based learning is not always possible or practical, the platform provides a centralized system where educators can create and deliver courses, and students can access learning materials at their convenience. This flexibility is particularly important in a world that increasingly demands accessible and high-quality educational resources. From a business perspective, the platform serves as a sustainable model for educational content delivery and monetization. It enables course creators to reach a wider audience and generate revenue through course sales, while providing administrators with tools to track sales performance and manage

the platform's financial aspects. The integrated payment system ensures secure and efficient transaction processing, while the analytics features provide valuable insights into course performance and user engagement. The system's administrative capabilities address the need for effective platform management and quality control. Administrators can monitor user activities, manage course content, and ensure compliance with educational standards.

The role-based access control system ensures that sensitive operations are restricted to authorized personnel, while the reporting features enable data-driven decision-making for platform improvement and growth. From a technical standpoint, the platform addresses the need for a secure, scalable, and reliable learning environment. The implementation of robust authentication mechanisms, secure payment processing, and efficient content delivery systems ensures that the platform can handle a large number of users while maintaining performance and security. The system's architecture is designed to be scalable, allowing for future expansion and the addition of new features as needed. The platform's comprehensive testing strategy and quality assurance measures address the need for reliability and user satisfaction. Through systematic testing of all components and features, the system ensures that it meets the highest standards of quality and performance. This focus on quality is essential for maintaining user trust and ensuring a positive learning experience.

## **1.4 Scope of System**

The scope of the E-learning Platform encompasses a comprehensive range of features and functionalities designed to create an effective online learning environment. The platform's scope is defined by its core components, user roles, and system capabilities, all working together to deliver a complete educational experience. The functional scope of the platform includes a robust user management system that handles user registration, authentication, and role-based access control for students, instructors, administrators, and super-administrators. The course management system provides comprehensive capabilities for course creation, content management, and enrollment control, while the learning content delivery system supports various content types and interactive learning materials.

The assessment and evaluation system includes features for quiz creation, automated grading, and certificate generation, ensuring a complete learning cycle for students. The platform's technical scope covers both frontend and backend development aspects. The frontend development focuses on creating a responsive and user-friendly interface with interactive learning components and real-time updates. The backend development implements a RESTful API architecture, database management, and secure user authentication. The system includes robust security features such as data encryption, role-based access control, and secure payment processing, ensuring the safety and integrity of user data and transactions. The platform's integration capabilities extend to various external services, including payment gateways, email services, cloud storage, and analytics tools. These integrations enhance the platform's functionality and provide a seamless experience for users.

The system's technical limitations are carefully defined to ensure optimal performance, including maximum file sizes for content uploads, concurrent user capacity, and storage limitations. The platform's future scope includes planned enhancements such as mobile application development, advanced analytics features, and AI-powered learning recommendations. The system's architecture is designed to be scalable, allowing for infrastructure scaling, performance optimization, and load balancing as the platform grows. The database and content delivery network are also structured to support future expansion and increased user load.

## **1.5 Operating Environment – H/w & S/w**

- **Hardware Requirement**

- A PC with Windows/Linux OS
- Processor with 1.7 to 2.4GHz speed
- Minimum of 4gb RAM
- Integrated or 2gb Graphic card

- **Software Specification:**

- IDE: VScode
- Client-Side Technologies: HTML, CSS, React
- Server-Side Technologies: Node.js, Java Script, Express
- Data Base: MongoDB

## **1.6 Detail Description of Technology Used**

### **MongoDB**

MongoDB is a NoSQL database that is used to store and manage data in the form of documents rather than traditional rows and columns. It stores data in BSON format, which is a binary representation of JSON, allowing complex and nested data to be stored efficiently. In contrast to relational databases, MongoDB is schema-less, meaning it does not require a fixed structure, which makes it very flexible and adaptable to changes. This is especially useful in modern web applications where data models can evolve over time. In this e-learning platform, MongoDB is used to store various types of data such as user profiles, course content, quiz records, and student progress.

The flexibility of MongoDB allows the platform to handle different types of information, from text and numbers to multimedia references. It also supports indexing and advanced query features, which help in retrieving data quickly. MongoDB is highly scalable and supports features like replication and sharding, which make it suitable for large-scale applications. Integration with Node.js using Mongoose (an Object Data Modeling library) allows easy communication between the server and the database. Overall, MongoDB is a reliable and powerful backend database choice for this kind of dynamic and content-rich web application.

In this project, MongoDB is used to store:

- User information (students, instructors, admins)
- Course content (titles, videos, PDFs)
- Quiz questions and results
- Student activity and progress tracking

MongoDB allows quick data retrieval and modification without requiring a fixed schema, which is helpful in applications where data structures may evolve over time.

## **Express.js**

Express.js is a lightweight and efficient web application framework built on top of Node.js. It is used to handle the server-side logic of web applications and APIs. In this project, Express serves as the central part of the backend that processes client requests, interacts with the database, and sends appropriate responses. It allows developers to define routes for different operations like user registration, login, course enrollment, and quiz submission. Express makes it easier to handle HTTP methods such as GET, POST, PUT, and DELETE, which are commonly used in web development. It supports middleware, which can be used to manage tasks like authentication, data validation, logging, and error handling.

The structure of Express is minimal but very flexible, allowing developers to organize their code in a clean and modular way. Since it works seamlessly with Node.js, Express can handle multiple requests at once, making it suitable for real-time applications with many users. In the e-learning platform, Express connects the frontend React components with MongoDB and ensures that data is passed securely and efficiently between the user and the database. Express also allows easy integration of third-party packages, which makes development faster and more scalable.

In this project, Express handles:

- Routing HTTP requests (GET, POST, PUT, DELETE)
- Middleware for logging, authentication, and validation
- Communication between frontend (React) and database (MongoDB)
- Error handling and response management

Express acts as the central controller of the backend, making sure each request is processed and responded to correctly. It's fast, reliable, and commonly used in full-stack JavaScript development.



## React.js

React.js is a powerful JavaScript library developed by Facebook for building user interfaces, especially single-page applications. It allows developers to create reusable components that manage their own state, which leads to more efficient and maintainable code. In the e-learning platform, React is used to develop the entire frontend interface, including the homepage, login and registration forms, course list pages, and student dashboards. React uses a virtual DOM, which compares changes before updating the actual DOM, making the application faster and more responsive. This improves performance, especially when dealing with frequent updates like quiz interactions or live progress tracking. JSX, a syntax extension in React, allows HTML to be written inside JavaScript, which makes the code more readable and easier to manage.

React also supports hooks, which help manage state and lifecycle methods in a cleaner way. The component-based structure of React means that each part of the platform, such as a course card or a quiz form, can be built and reused independently. This speeds up development and reduces bugs. React Router is used to manage navigation between different pages without reloading the entire application. Overall, React provides a smooth, modern, and user-friendly experience for students and instructors using the platform.

In this e-learning platform, React is responsible for:

- Building the user interface (course pages, dashboards, forms)
- Managing state and rendering dynamic content
- Handling user interactions like quiz submissions or course enrollment
- Communicating with backend APIs using Axios or Fetch

React uses a **Virtual DOM** to update only the parts of the UI that change, improving speed and user experience. It supports modern development concepts like hooks, context API, and client-side routing using React Router.

## **HTML (Hyper Text Markup Language)**

HTML is the standard markup language used to create the basic structure and content of web pages. It provides the foundation upon which all other technologies like CSS and JavaScript operate. In this project, HTML is used within React components (using JSX) to define the layout of the interface, including headings, buttons, text fields, images, and links. Even though React dynamically generates the interface, HTML still plays a critical role in determining how the content is organized and displayed. Each part of a page, such as a course listing or a quiz question, is built using HTML elements like `<div>`, `<h1>`, `<p>`, `<form>`, and `<input>`.

These elements tell the browser how to display the content and how users can interact with it. HTML is also essential for accessibility, helping screen readers and other tools understand the structure of the page. Semantic HTML elements like `<header>`, `<nav>`, `<main>`, and `<footer>` help improve both user experience and SEO. Forms built with HTML are used for user registration, login, and submitting quizzes. Though basic in appearance, HTML is a crucial part of the frontend that ensures the user interface is structured properly and functions as expected.

**Use in Project:** In this e-learning platform, HTML forms the foundation of the entire frontend interface. Every visual element seen on the platform—such as buttons, input fields, forms, text areas, and layout containers—is structured using HTML. It provides the skeleton upon which styles and functionality are applied through CSS and JavaScript. When working with React, HTML is written using JSX, which allows HTML-like syntax to be embedded within JavaScript code. This makes it easier to manage the structure of each component. For example, a course card displaying the course title, description, and enroll button is built using basic HTML elements like `<div>`, `<h2>`, `<p>`, and `<button>`. In forms like login, registration, and quiz submissions, HTML elements such as `<form>`, `<input>`, and `<label>` are used to capture user data. Semantic HTML tags such as `<header>`, `<nav>`, `<main>`, and `<footer>` help organize content in a meaningful way, which improves accessibility and search engine visibility. Overall, HTML ensures that the platform is well-structured, accessible, and easily navigable for users across all devices and browsers.

## **CSS (Cascading Style Sheets)**

CSS is a stylesheet language used to control the look and feel of web pages created with HTML. While HTML provides the structure, CSS is responsible for the design, including colours, fonts, layout, and animations. In the e-learning platform, CSS is used to style the React components, making the interface visually appealing and easy to use. Modern CSS techniques like Flexbox and Grid are used to create responsive layouts that work well on desktops, tablets, and smartphones. CSS ensures that elements are properly spaced, aligned, and sized to provide a clean and organized user experience. In addition to layout, CSS is also used for aesthetic improvements such as hover effects, transitions, buttons, and background images.

Media queries help adjust styles based on screen size, which is especially important in education platforms where students may access content from various devices. CSS can be written directly in components using inline styles, as separate .css files, or with styling libraries such as Tailwind CSS or Styled Components. Regardless of the approach, CSS plays a key role in improving usability and ensuring that the application is both functional and visually polished.

**Use in Project:** In this e-learning platform, CSS is used to style the frontend components created with React. It defines the overall layout of course cards, navigation bars, quiz pages, and dashboards. CSS is responsible for ensuring a responsive design so that users can access the platform smoothly from desktops, tablets, or mobile phones. Flexbox and Grid are used to create structured layouts, while media queries help adjust styles based on screen size. Animations and hover effects are applied to enhance the user experience and provide visual feedback for actions like button clicks or navigation. Consistent use of colour schemes and fonts gives the platform a unified and professional appearance.

## Node.js

Node.js is a JavaScript runtime built on Chrome's V8 engine that allows JavaScript to run outside the browser. It enables developers to build the server-side of web applications using the same language they use on the frontend. Node.js is event-driven and non-blocking, meaning it can handle many requests simultaneously without getting stuck waiting for one task to finish. This makes it ideal for high-performance web applications like an e-learning platform, where multiple users might be accessing courses or submitting quizzes at the same time. In this project, Node.js serves as the base environment for the server, running the backend logic and controlling the flow of data between the frontend and the database.

It executes the Express.js server, handles API requests, manages user sessions, and connects to MongoDB. The package manager that comes with Node.js, called npm (Node Package Manager), allows easy installation of libraries and tools needed during development. Node's ability to manage asynchronous tasks efficiently improves the speed and scalability of the system. It also makes development faster by allowing a single language—JavaScript—to be used across the entire application.

**Use in Project:** In this e-learning platform, Node.js serves as the foundation for the backend server. It runs the Express.js application and handles all server-side operations such as user authentication, API processing, and communication with the MongoDB database. Node.js enables asynchronous processing, allowing the server to handle multiple requests at once without slowing down. This is particularly useful in situations where many users are interacting with the platform simultaneously—such as submitting quizzes, updating progress, or retrieving course data. Node.js also provides access to npm (Node Package Manager), which is used to install and manage dependencies needed for backend functionalities like encryption, validation, and routing.

## **JavaScript**

JavaScript is a high-level, interpreted programming language that powers the dynamic behaviour of web applications. It is the backbone of modern web development and is used on both the client-side and server-side in this project. On the frontend, JavaScript is used within React to create interactive components, handle events like clicks or form submissions, and manage application state. It allows content to update without reloading the page, which is crucial for features like quizzes, dashboards, and real-time feedback.

On the backend, JavaScript is used through Node.js and Express.js to handle routing, data processing, user authentication, and database interactions. JavaScript is known for its flexibility, supporting object-oriented, procedural, and functional programming styles. It has a vast ecosystem, and its compatibility with frameworks, libraries, and APIs makes it a central language in web development. In the e-learning platform, JavaScript enables a seamless user experience by making the system responsive, fast, and interactive from both the frontend and backend.

**Use in Project:** In this e-learning platform, JavaScript is the glue that connects the entire system. It is used in React to manage user interactions, control application state, and dynamically render content. On the backend, JavaScript is used with Node.js and Express to handle data processing, API routing, and server logic. JavaScript enables the platform to respond instantly to user actions, such as clicking a button, submitting a form, or switching between pages. It also handles form validations, user inputs, and real-time feedback during quiz attempts. Thanks to JavaScript's versatility and its ability to run the full stack, the platform maintains consistency and performance across all components, resulting in a smooth and interactive learning experience.

## CHAPTER 2

# PROPOSED SYSTEM

---

---

### **2.1 Study of Similar Systems**

In studying similar systems to our E-learning Platform, we can examine existing Learning Management Systems (LMS) and Educational Technology (EdTech) platforms that have established themselves in the market. These systems provide valuable insights into best practices, features, and implementation strategies for online education platforms. Learning Management Systems (LMS): LMS platforms are comprehensive systems designed to deliver, track, and manage educational content and training programs. They integrate various educational functions such as course management, student tracking, assessment tools, and communication features.

Popular LMS platforms include Moodle, Canvas, Blackboard, and Google Classroom. These systems typically include modules for course creation and management, student enrollment, progress tracking, and assessment tools. They also provide features for content delivery, discussion forums, and grading systems. The integration of these components creates a cohesive learning environment that supports both synchronous and asynchronous learning. Educational Technology (EdTech) Platforms: EdTech platforms focus primarily on delivering educational content and managing the learning experience. They provide tools for content creation, student engagement, and learning analytics. These platforms often include features for video lectures, interactive assessments, and progress tracking. Popular EdTech platforms include Coursera, Udemy, edX, and Khan Academy. These systems typically offer modules for course creation, student enrollment, payment processing, and certificate generation. They may also provide integrations with other educational tools and services to enhance the learning experience. When studying these similar systems, we can analyse their key features, implementation processes, and best practices. For instance, Moodle's open-source nature and extensive customization options demonstrate the importance of flexibility in LMS design. Canvas's user-friendly interface and mobile responsiveness highlight the significance of accessibility in modern educational platforms. Coursera's

partnership model with universities and its certificate programs show the value of credentialing in online education. The study of these systems reveals several important aspects for our E-learning Platform:

1. Content Management: Similar systems emphasize the importance of robust content management capabilities, including support for various media types, organized course structures, and easy content updates.
2. User Experience: Successful platforms prioritize intuitive navigation, responsive design, and accessibility features to ensure a positive learning experience for all users.
3. Assessment and Progress Tracking: Effective systems incorporate comprehensive tools for creating assessments, tracking student progress, and providing feedback.
4. Monetization Strategies: Platforms like Udemy and Coursera demonstrate successful approaches to course pricing, subscription models, and payment processing.
5. Analytics and Reporting: Leading systems provide detailed analytics on student engagement, course performance, and learning outcomes.
6. Integration Capabilities: Modern platforms offer APIs and integration options to connect with other educational tools and services.

## **2.2 Feasibility Study**

### **1. Technical Feasibility**

The platform uses a modern, proven, and scalable tech stack that ensures a smooth development process and long-term maintainability. All selected technologies are open-source, lightweight, and widely supported.

- **Frontend:** Developed using React.js (with Vite) for fast and dynamic UI rendering. Features like routing (React Router), notifications (React Hot Toast), and visual analytics (Chart.js) enhance the user experience.
- **Backend:** Built using Node.js and Express.js, which efficiently manage server-side logic and RESTful APIs. The system handles user management, payments, file uploads, and real-time data handling.
- **Database:** Uses MongoDB, a NoSQL database well-suited for flexible and complex data structures. It stores users, courses, lectures, payments, certificates, and user progress efficiently.
- **Other Key Technologies:**
  - JWT for secure user authentication.
  - Bcrypt for password hashing.
  - Razorpay for handling payments securely.
  - PDFKit for dynamic certificate generation.
  - Nodemailer for sending OTPs and emails.



## 2. Economic Feasibility

The project is cost-effective and practical for institutions with limited budgets.

- No software licensing fees (all core tech is open-source).
- No requirement for high-end infrastructure — basic internet and personal computers are sufficient.
- Minimal hosting costs; the frontend can even run on GitHub Pages (free).
- Reusable components and modular code reduce development and maintenance time.
- Requires minimal ongoing expenses — ideal for student projects or small educational institutions.

## 3. Operational Feasibility

From the end-user perspective (students and teachers), the platform is simple, intuitive, and requires almost no technical background to use. It's designed for quick adoption with minimal support.

- Clean, user-friendly interface for both students and faculty.
- Role-based access ensures admins and users only see relevant features.
- Students can access the platform from any device and resume learning anytime.
- Faculty can create/update courses and lectures without any coding.
- Progress tracking, email alerts, and certificates are all automated.
- Scalable to support hundreds of users without system degradation.

## **2.3 Objectives of the System**

- To deliver structured online technical education by providing a centralized platform for publishing and accessing course content in programming, software testing, AI, and data science.
- To enable easy content upload and management by allowing instructors and admins to upload, organize, and manage video lectures through a simple interface.
- To provide role-based access control by assigning users roles like Student, Admin, or Super Admin, each with specific permissions to securely manage their responsibilities.
- To ensure seamless content delivery by enabling students to access study materials from any device via a responsive and user-friendly interface.
- To maintain a list of all registered users by offering admins a centralized module to view and manage user roles, activity, and account status.
- To support institutional representation by including a dedicated College About page that showcases institutional background, departments, vision, and contact information.
- To simplify administrative control by giving admins tools to manage users, approve or reject content, and monitor system activity from a dedicated admin dashboard.
- To promote scalability and ease of use by building a system that supports a growing number of users and courses with a clean, intuitive design.
- To integrate secure online payments using Razorpay so that students can pay for premium content and admins can track transactions.
- To generate completion certificates automatically for students who finish courses, allowing them to download and verify their achievements.
- To provide detailed reports and analytics for admins and instructors to track course popularity, revenue data, and system performance.

## **2.4 Users of System**

### **Super Admin**

- Has complete control over the entire platform
- Can manage all users, courses, and content
- Has access to all reports and analytics
- Can override any permission restrictions
- Manages system configuration

### **Admin**

- Manages courses and content
- Handles user management
- Views and generates reports
- Manages certificates
- Controls course access
- Creates and manages courses
- Uploads and manages lectures
- Tracks student progress
- Generates course reports
- Manages course content

### **Student**

- Enrolls in courses
- Accesses course content
- Takes assessments
- Views progress
- Downloads certificates
- Manages profile

## CHAPTER 3

### ANALYSIS AND DESIGN

---

#### 3.1 TABLE SPECIFICATIONS (DATABASE)

##### 3.1.1 Functional Requirements

- **User Registration and Authentication:** Users should be able to register on the platform using a unique email and a secure password. An email verification step (OTP-based) must be included to confirm account validity. Authentication must be implemented using JWT (JSON Web Token) to maintain secure login sessions and role-based access.
- **Role-Based Access Control**

The system should support three roles: Student, Admin, and Super Admin.

  - Students can browse and enroll in courses, view lectures, track their progress, and download certificates.
  - Admins can upload and manage course content, view student progress, and moderate feedback.
  - Super Admins can manage all users, content, platform settings, and oversee the entire system.
- **Course Content Management:** Admins and Super Admins should be able to create, edit, update, and delete courses. This includes uploading lecture videos, PDFs, resource links, and categorizing content by subject (e.g., AI, Data Science). The platform must allow dynamic updates without affecting existing student progress.
- **Student Progress Tracking:** The platform must track each student's lecture completion status, time spent on courses, and quiz results. Visual indicators (like progress bars) should help students monitor their learning. Admins should be able to view analytics to evaluate course effectiveness and student performance.
- **Certificate Generation:** Upon course completion, students should automatically receive downloadable PDF certificates containing their name, course name, and completion date, generated and delivered through the system.

- **Admin Dashboard:** Super Admins and Admins should have access to a dashboard showing metrics like the number of users, active courses, feedback received, student performance data, and total revenue. Admins should also be able to upload new content, respond to issues, and manage user roles.
- **User Listing and Management:** Super Admins should have access to a complete list of all registered users, including role and account status. They should be able to activate/deactivate accounts, assign or revoke roles, and manage user access. Admins can view lists of students assigned to their courses.
- **Payment Integration:** The platform must support payment functionality via Razorpay. Students should be able to make secure payments before enrolling in paid courses. Payment status should be tracked, and successful transactions should allow instant access to course material.
- **College About Page:** A public-facing “College About” section should provide information about the institution—its background, departments, vision, and contact details—editable by the Super Admin from the dashboard.
- **Reports and Analytics:** The system should generate reports for Admins and Super Admins regarding user activity, course popularity, payments made, quiz scores, and overall platform performance for strategic decisions.

### 3.1.2 Non-Functional Requirements

- **Performance**

The system should efficiently handle multiple concurrent users, including students streaming video lectures and admins managing content. Page loading times, dashboard access, course interactions, and search queries must remain fast and responsive even under high traffic.

- **Security**

All user data, including login credentials, course progress, and payment details, must be securely stored and protected from unauthorized access. Implement best practices such as HTTPS, JWT-based session management, bcrypt password hashing, and secure API endpoints to ensure data confidentiality and integrity.

- **Reliability**

The platform should maintain high availability with minimal downtime, ensuring students and instructors have consistent access to learning materials. Regular backups and disaster recovery strategies must be in place to prevent data loss in case of failure.

- **Scalability**

The system must support increasing numbers of users, courses, and lectures over time without degrading performance. This includes the ability to handle growth in registered students, video uploads, and transactions by adopting modular architecture and scalable backend services.

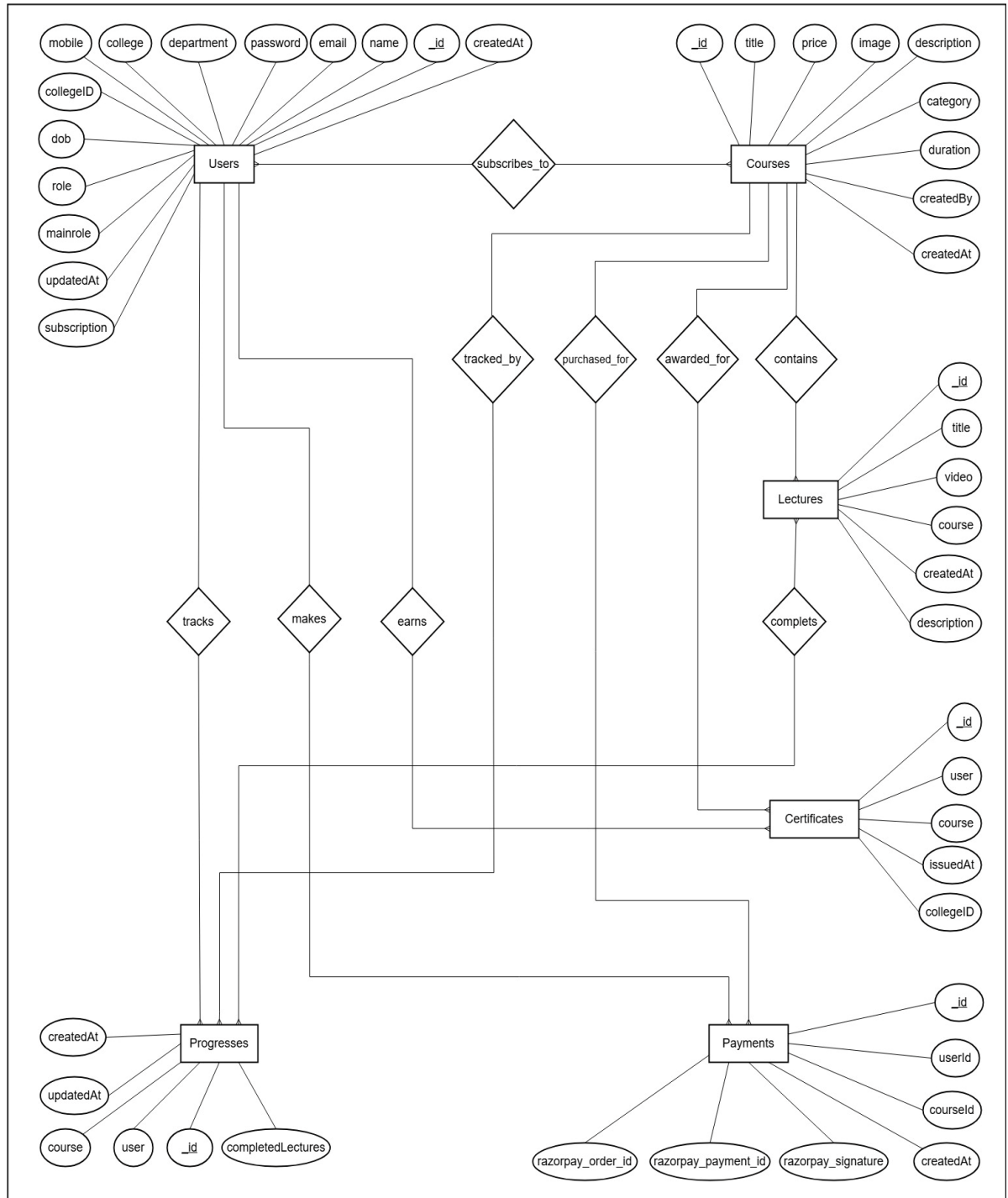
- **Usability**

The platform should have a clean, intuitive, and user-friendly interface for all roles—Students, Admins, and Super Admins. It must be easy to navigate, with clear instructions, tooltips, and responsive layouts that work across all devices to support self-paced learning.

- **Compatibility**

The system must be compatible with all major modern browsers (Chrome, Firefox, Edge, Safari) and function properly on desktops, tablets, and mobile devices. Consistent user experience across platforms should be ensured through responsive design and cross-platform testing.

### 3.2 ER Diagram



### 3.3 Table Structure

#### 1. Users:

<b>Table Name</b>		Users		
<b>Primary Key</b>		_id		
<b>Foreign Key</b>		-		
<b>Description</b>		This table contains user details.		
<b>Sr No.</b>	<b>Field Name</b>	<b>Data Type</b>	<b>Constraints</b>	<b>Description</b>
1	_id	ObjectId	Primary Key	Unique identifier for the user
2	name	string	Not Null	User's full name
3	email	string	Not Null	User's email address used for login
4	password	string	Not Null	User's encrypted password
5	department	string	Not Null	User's department name
6	college	string	Not Null	User's college name (from predefined list)
7	mobile	string	Not Null	User's mobile phone number
8	collegeID	string	Not Null	User's college identification number
9	dob	date	Not Null	User's date of birth
10	role	string	Not Null	User's system role for permissions
11	mainrole	string	Not Null	User's primary role in the system
12	subscription	array	Not Null	Array of Course IDs the user is subscribed
13	createdAt	date	Not Null	Timestamp when user account was created
14	updatedAt	date	Not Null	Timestamp when user open account



**2. Courses:**

<b>Table Name</b>		courses		
<b>Primary Key</b>		_id		
<b>Foreign Key</b>		-		
<b>Description</b>		This table contains details about course details.		
<b>Sr No.</b>	<b>Field Name</b>	<b>Data Type</b>	<b>Constraints</b>	<b>Description</b>
1	_id	ObjectId	Primary Key, Unique	Unique identifier for the course
2	title	string	Not Null	Title of the course
3	description	string	Not Null	Detailed description of the course content
4	image	string	Not Null	URL or path to the course thumbnail image
5	price	number	Not Null	Cost of the course
6	duration	number	Not Null	Duration of the course (in hours/days)
7	category	string	Not Null	Category the course belongs to
8	createdBy	string	Not Null	Identifier of the course creator
9	createdAt	date	Not Null	Timestamp when course was created

**3. Lectures:**

<b>Table Name</b>		lectures		
<b>Primary Key</b>		_id		
<b>Foreign Key</b>		course		
<b>Description</b>		This table details about course lectures.		
<b>Sr No.</b>	<b>Field Name</b>	<b>Data Type</b>	<b>Constraints</b>	<b>Description</b>
1	_id	ObjectId	Primary Key, Unique	Unique identifier for the lecture
2	title	string	Not Null	Title of the lecture
3	description	string	Not Null	Detailed description of the lecture content
4	video	string	Not Null	URL or path to the lecture video
5	course	ObjectId	Foreign Key	Course to which the lecture belongs
6	createdAt	date	Not Null	Timestamp when lecture was created

**4. Progresses:**

<b>Table Name</b>		progresses		
<b>Primary Key</b>		_id		
<b>Foreign Key</b>		Course, user		
<b>Description</b>		This table contains details about user progress.		
<b>Sr No.</b>	<b>Field Name</b>	<b>Data Type</b>	<b>Constraints</b>	<b>Description</b>
1	_id	ObjectId	Primary Key	Unique identifier for the progress record
2	course	ObjectId	Foreign Key, References COURSES._id	Course for which progress is tracked
3	completedLectures	array	Not Null	Array of completed Lecture IDs
4	user	ObjectId	Foreign Key, References USER._id	User whose progress is being tracked
5	createdAt	date	Not Null	Timestamp when progress tracking started
6	updatedAt	date	Not Null	Timestamp when progress was last updated

**5. Payments:**

<b>Table Name</b>		payments		
<b>Primary Key</b>		_id		
<b>Foreign Key</b>		userId, courseId		
<b>Description</b>		This table contains details about user payments.		
<b>Sr No.</b>	<b>Field Name</b>	<b>Data Type</b>	<b>Constraints</b>	<b>Description</b>
1	_id	ObjectId	Primary Key, Unique	Unique identifier for the payment
2	userId	ObjectId	Foreign Key, References USER._id	User who made the payment
3	courseId	ObjectId	Foreign Key, References COURSES._id	Course for which payment was made
4	razorpay_order_id	string	Not Null	Order ID from Razorpay payment gateway
5	razorpay_payment_id	string	Not Null	Payment ID from Razorpay payment gateway
6	razorpay_signature	string	Not Null	Payment signature from Razorpay for verification
7	createdAt	date	Not Null	Timestamp when payment was made

**6. Certificates:**

<b>Table Name</b>		certificates		
<b>Primary Key</b>		_id		
<b>Foreign Key</b>		userId, courseId		
<b>Description</b>		This table contains details about user certificates.		
<b>Sr No.</b>	<b>Field Name</b>	<b>Data Type</b>	<b>Constraints</b>	<b>Description</b>
1	_id	ObjectId	Primary Key, Unique	Unique identifier for the certificate
2	user	ObjectId	Foreign Key, References USER._id	User who earned the certificate
3	course	ObjectId	Foreign Key, References COURSES._id	Course for which certificate was issued
4	issuedAt	date	Not Null	Timestamp when certificate was issued
5	collegeID	string	Not Null	College ID of the user receiving the certificate

### 3.3.2 Data Dictionary

Sr No.	Table Name	Field Name	Data Type	Constraints	Description
1	Users	_id	ObjectId	Primary Key, Unique	Unique identifier for the user
2	Users	name	string	Not Null	User's full name
3	Users	email	string	Not Null	User's email address used for login
4	Users	password	string	Not Null	User's encrypted password
5	Users	department	string	Not Null	User's department name
6	Users	college	string	Not Null	User's college name (from predefined list)
7	Users	mobile	string	Not Null	User's mobile phone number
8	Users	collegeID	string	Not Null	User's college identification number
9	Users	dob	date	Not Null	User's date of birth
10	Users	role	string	Not Null	User's system role for permissions
11	Users	mainrole	string	Not Null	User's primary role in the system
12	Users	subscription	array	Not Null	Array of Course IDs the user is subscribed to

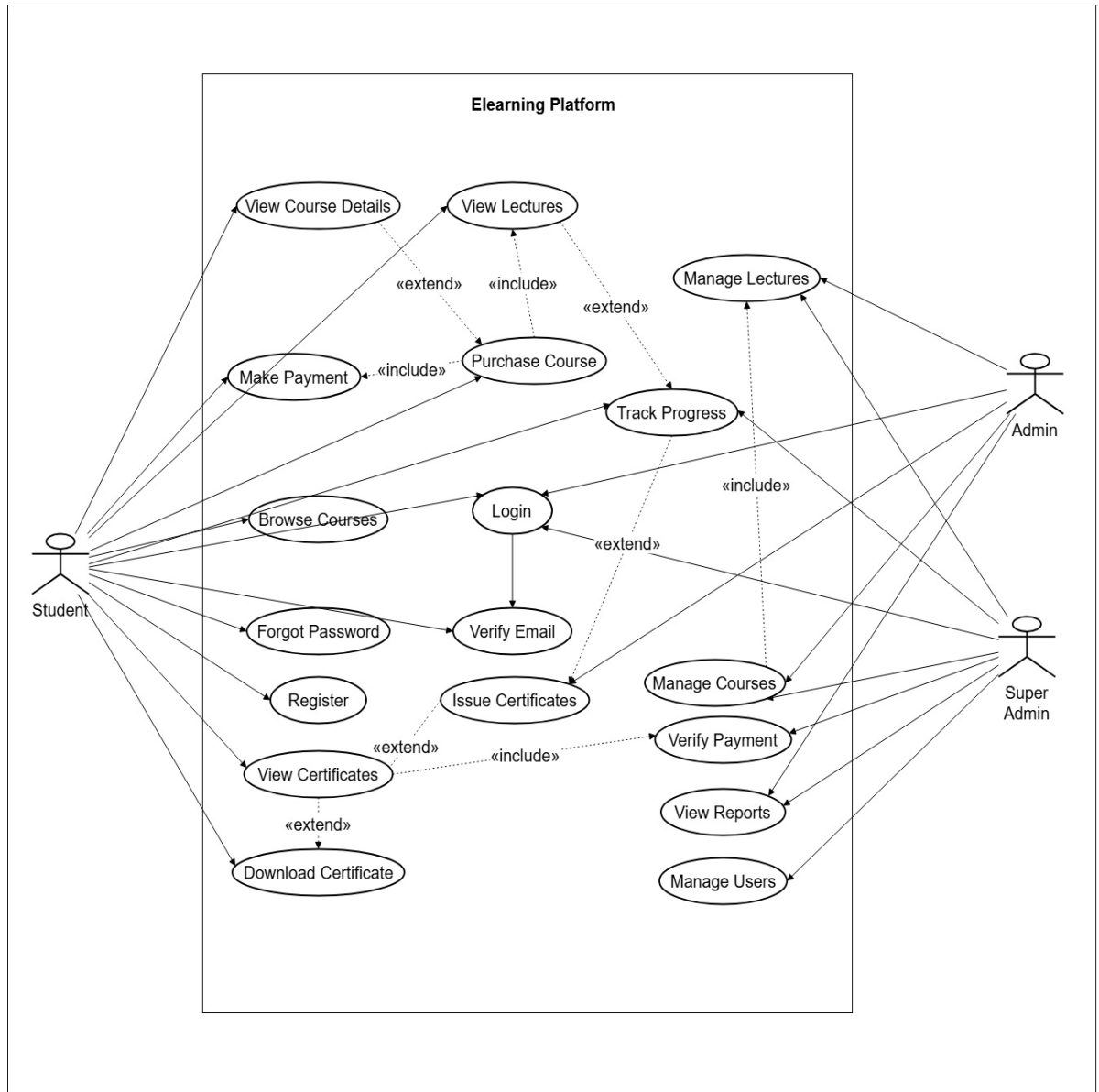
13	Users	createdAt	date	Not Null	Timestamp when user account was created
14	Users	updatedAt	date	Not Null	Timestamp when user account was last updated
15	Courses	_id	ObjectId	Primary Key, Unique	Unique identifier for the course
16	Courses	title	string	Not Null	Title of the course
17	Courses	description	string	Not Null	Detailed description of the course content
18	Courses	image	string	Not Null	URL or path to the course thumbnail image
19	Courses	price	number	Not Null	Cost of the course
20	Courses	duration	number	Not Null	Duration of the course (in hours/days)
21	Courses	category	string	Not Null	Category the course belongs to
22	Courses	createdBy	string	Not Null	Identifier of the course creator
23	Courses	createdAt	date	Not Null	Timestamp when course was created
24	Lectures	_id	ObjectId	Primary Key, Unique	Unique identifier for the lecture
25	Lectures	title	string	Not Null	Title of the lecture
26	Lectures	description	string	Not Null	Detailed description of the lecture content

27	Lectures	video	string	Not Null	URL or path to the lecture video
28	Lectures	course	ObjectId	Foreign Key, References COURSES._id	Course to which the lecture belongs
29	Lectures	createdAt	date	Not Null	Timestamp when lecture was created
30	Progresses	_id	ObjectId	Primary Key	Unique identifier for the progress record
31	Progresses	course	ObjectId	Foreign Key, References COURSES._id	Course for which progress is tracked
32	Progresses	completedLectures	array	Not Null	Array of completed Lecture IDs
33	Progresses	user	ObjectId	Foreign Key, References USER._id	User whose progress is being tracked
34	Progresses	createdAt	date	Not Null	Timestamp when progress tracking started
35	Progresses	updatedAt	date	Not Null	Timestamp when progress was last updated
36	Payments	_id	ObjectId	Primary Key, Unique	Unique identifier for the payment
37	Payments	userId	ObjectId	Foreign Key, References USER._id	User who made the payment
38	Payments	courseId	ObjectId	Foreign Key, References COURSES._id	Course for which payment was made

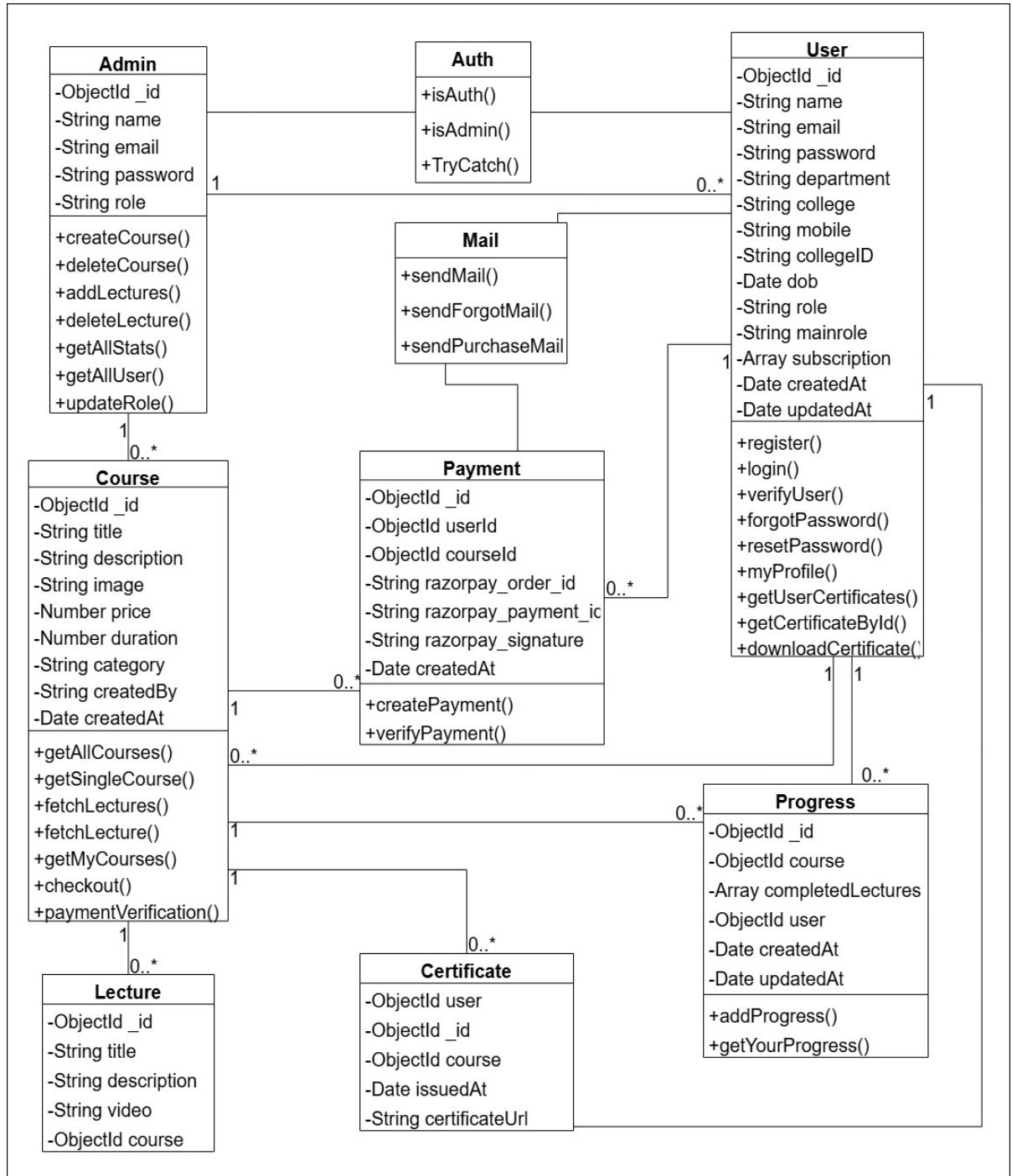


39	Payments	razorpay_order_id	string	Not Null	Order ID from Razorpay payment gateway
40	Payments	razorpay_payment_id	string	Not Null	Payment ID from Razorpay payment gateway
41	Payments	razorpay_signature	string	Not Null	Payment signature from Razorpay for verification
42	Payments	createdAt	date	Not Null	Timestamp when payment was made
43	Certificates	_id	ObjectId	Primary Key, Unique	Unique identifier for the certificate
44	Certificates	user	ObjectId	Foreign Key, References USER._id	User who earned the certificate
45	Certificates	course	ObjectId	Foreign Key, References COURSES._id	Course for which certificate was issued
46	Certificates	issuedAt	date	Not Null	Timestamp when certificate was issued
47	Certificates	collegeID	string	Not Null	College ID of the user receiving the certificate

### 3.4 Use case Diagram

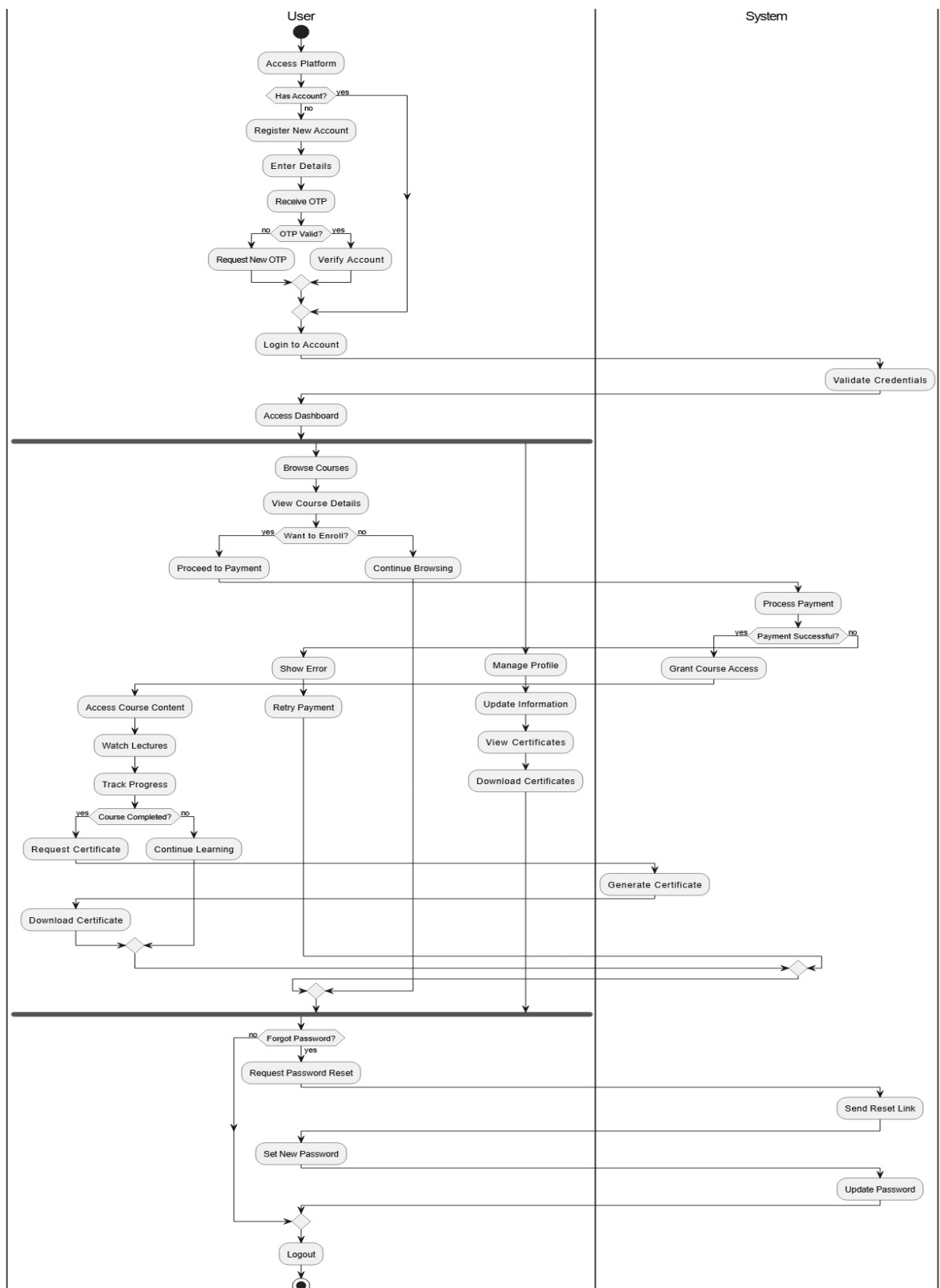


### 3.5 Class Diagram

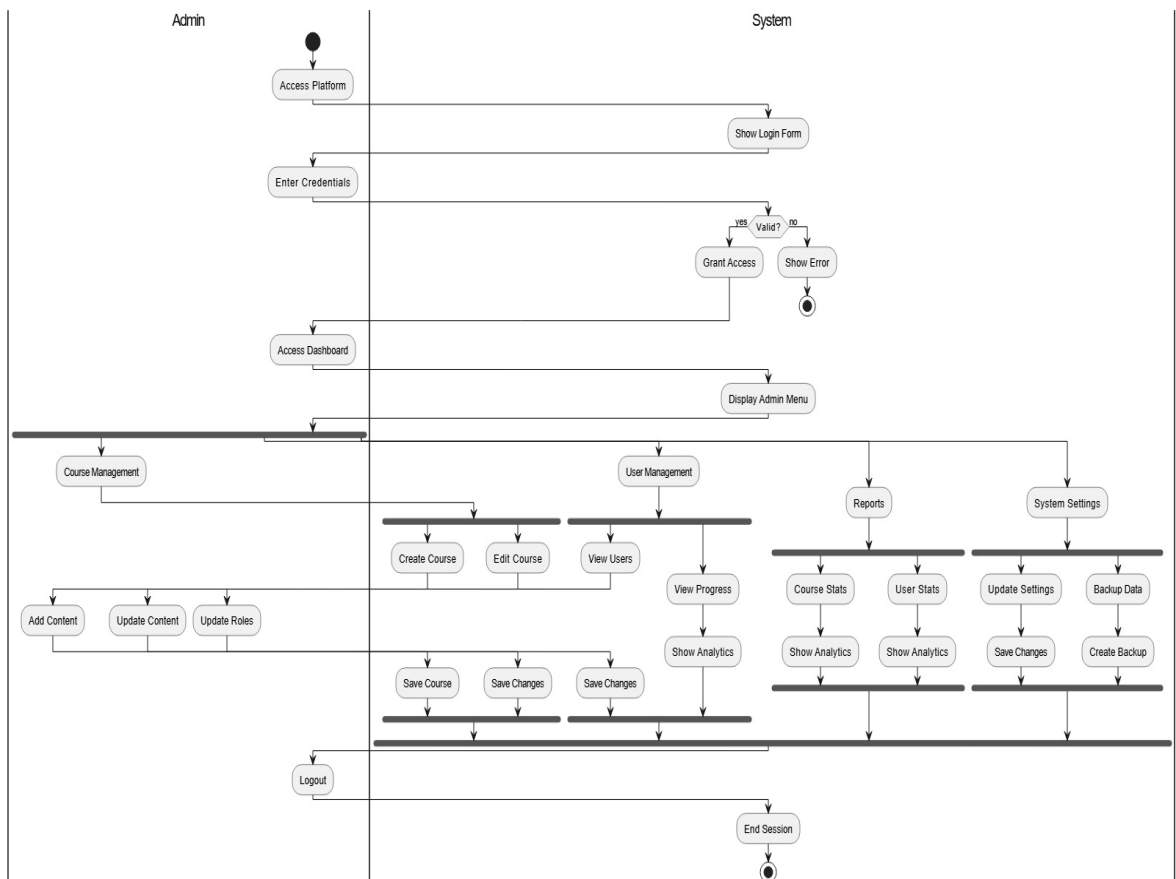


### 3.6 Activity Diagram

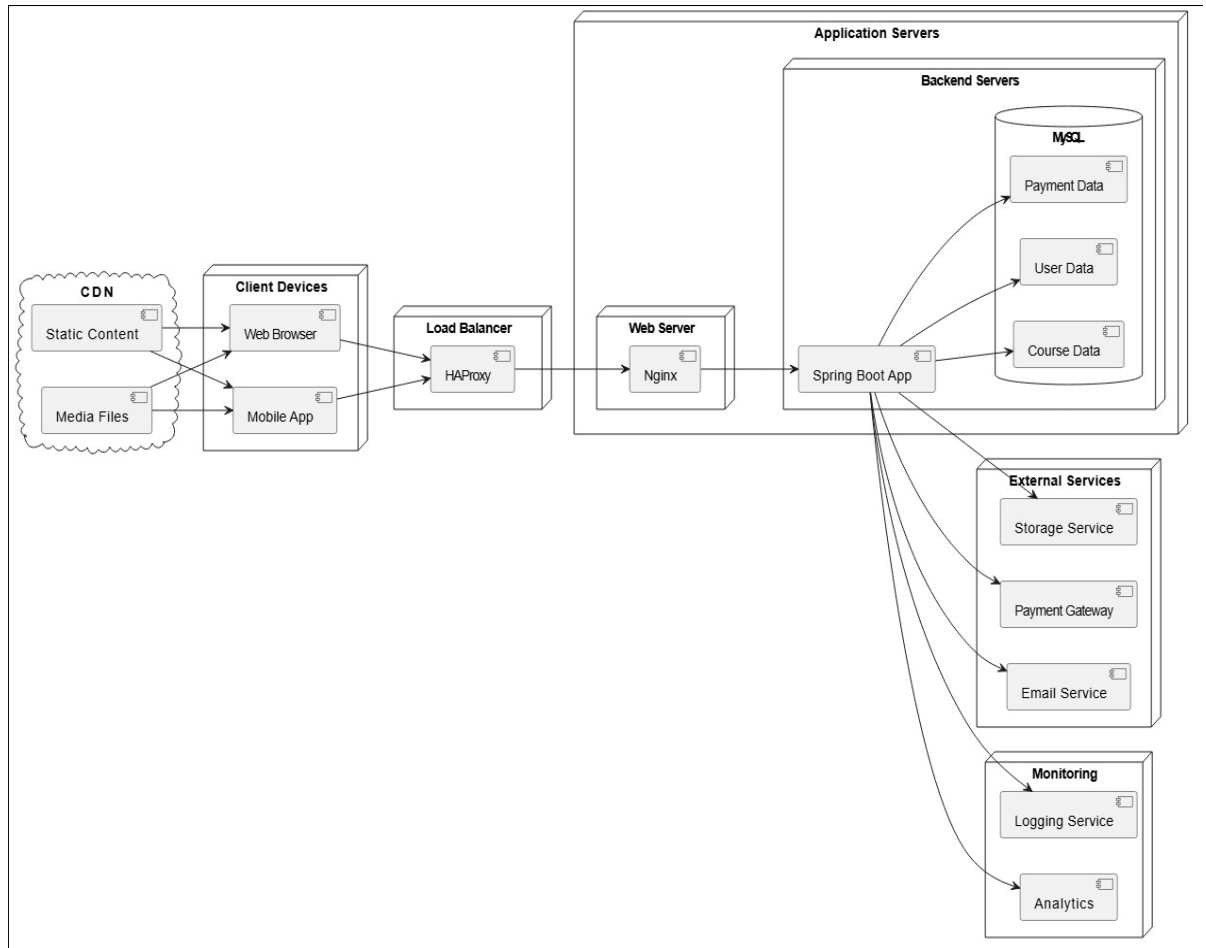
#### User



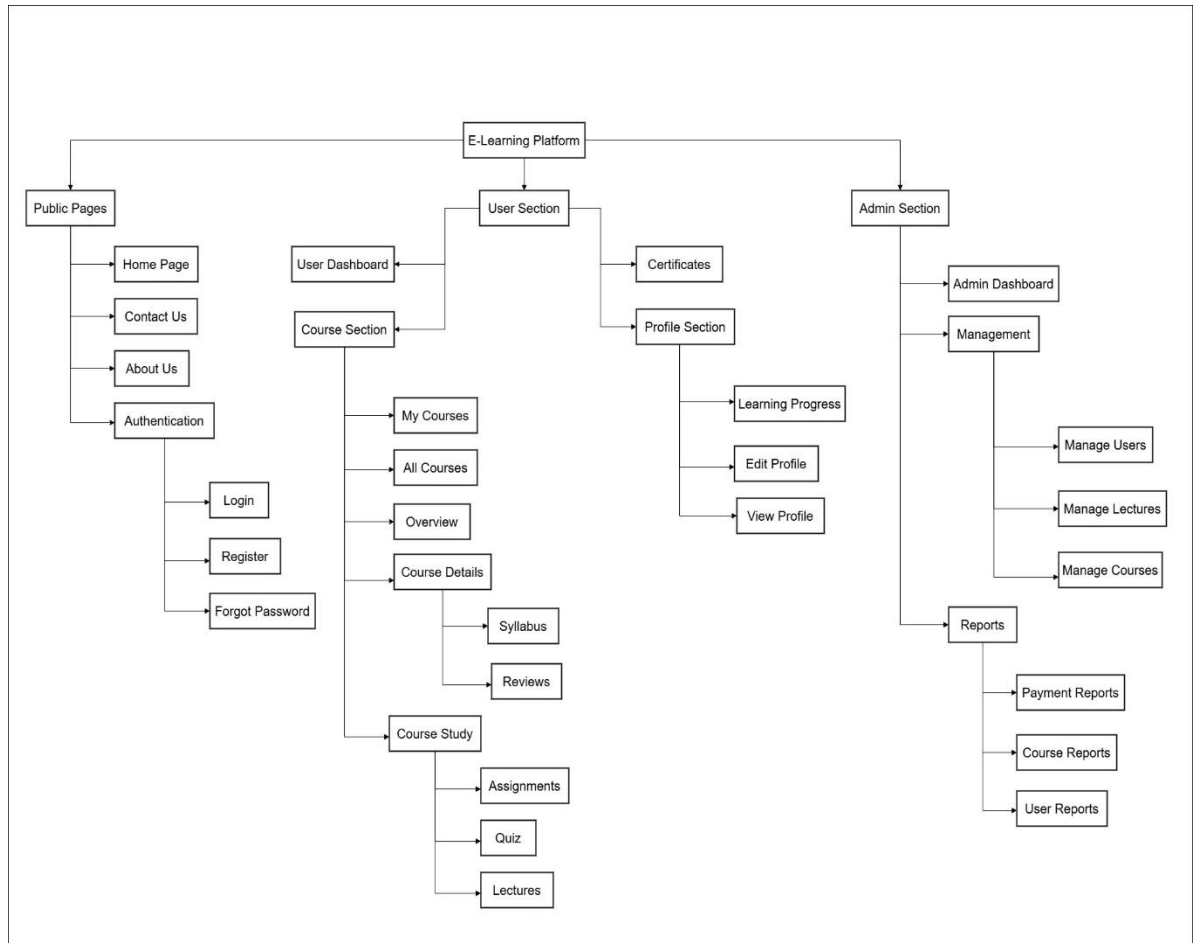
## Admin



### 3.7 Deployment Diagram



### 3.8 Module Hierarchy Diagram



### 3.9 Sample Input and Output Screens

**ZEAL EDUCATION SOCIETY**

Home Courses About Account

Home Courses Add Courses Users Sales Lectures Certificates Exit

#### Lectures Report

Total Lectures	Total Courses	Average Lectures per Course	Courses with Most Lectures
1	9	0.1	Android App

**All Lectures**

Sr. No.	Course	Lecture Title
1	Android App Development (Kotlin)	ewf
Total		1

**Course Breakdown**

**ZEAL EDUCATION SOCIETY**

Home Courses About Account

Home Courses Add Courses Users Sales Lectures Certificates Exit

#### Sales Reports

Total Sales	Total Transactions	Average Transaction
₹23,256	34	₹684


**Sales Report**

Filter by Time Period: Monthly

##### Course Sales Summary

Course Name	Units Sold	Price (₹)	Sales Amount (₹)
Data Structures and Algorithms	14	₹599	₹8,386
Mathematical Optimization Techniques	1	₹499	₹499
Information Security and Risk Management	3	₹899	₹2,697
Java Programming and Applications	2	₹899	₹1,798
Python Programming And Fundamentals	3	₹439	₹1,317
Full Stack JavaScript Development	1	₹550	₹550





[Home](#)
[Courses](#)
[About](#)
[Account](#)

[Home](#)
[Courses](#)
[Add Courses](#)
[Users](#)
[Sales](#)
[Lectures](#)
[Certificates](#)
[Exit](#)

### Certificate Reports


Sr. No.	User Name	Course Name	ZPRN	Issue Date	Certificate ID
1	Prathmesh Bairaj Gaddam	Data Structures and Algorithms	N/A	4/26/2025	680cc5c833abc41773a9af3a
2	Sameer Vishwnath Dagale	Data Structures and Algorithms	N/A	4/26/2025	680cd0ee169d77cfc2cc139
3	Prathmesh Bairaj Gaddam	Android App Development (Kotlin)	N/A	4/27/2025	680d39ee916e6498b834e7d3
4	Aditya Pramod Dayma	Data Structures and Algorithms	N/A	4/27/2025	680df7ef4c0678ada890bd86
5	Akash Anil Khot	Data Structures and Algorithms	523C10027	4/28/2025	680efad8f1cdf03a6f51feb2
6	Rushi Ramesh Kumbhar	Data Structures and Algorithms	46787654	5/1/2025	6812e103d8ee2158d351f868
7	Sai Pramod Jawake	Data Structures and Algorithms	523C10005	5/2/2025	6813c6e4ee2ef11cf44ce68d
8	Sai Pramod Jawake	Android App Development (Kotlin)	523C10005	5/2/2025	6813c903ee2ef11cf44ce854


[Home](#)
[Courses](#)
[About](#)
[Account](#)

[Home](#)
[Courses](#)
[Add Courses](#)
[Users](#)
[Sales](#)
[Lectures](#)
[Certificates](#)
[Exit](#)

### All Users

Sr. No.	Name	Email ID	User Role	Update Role
1	Athar Patel	patelathar12@gmail.com	admin	<a href="#">Update Role</a>
2	Aditya Pramod Dayma	adityadayma2003@gmail.com	user	<a href="#">Update Role</a>
3	Abhishek Dattatray Shirke	abhishekshirke5224@gmail.com	user	<a href="#">Update Role</a>
4	Padmashri Tanaji Shinde	padmashrishinde270@gmail.com	user	<a href="#">Update Role</a>
5	Nikhil Kisan Parhad	dragonemperor779@gmail.com	user	<a href="#">Update Role</a>
6	Aryan Sandip Pote	aryanpote26@gmail.com	user	<a href="#">Update Role</a>
7	Prathmesh Bairaj Gaddam	prathmeshgaddam11pg@gmail.com	user	<a href="#">Update Role</a>
8	Sanjana Pravin Mane	sanjanamane33@gmail.com	user	<a href="#">Update Role</a>
9	Sameer Vishwnath Dagale	samsteelsingh@gmail.com	user	<a href="#">Update Role</a>

ZEAL EDUCATION SOCIETY

Home Courses About Account

Lectures completed - 0 out of 0  
0 %

Select a Lecture

Close

Add Lecture

Title  
Merging With DSA


Description  
The goal is to combine them into a single array th


Choose File Merge\_Two\_So...aKD(360p) mp4

All Power is Within You  
You Can Do Anything  
And Everything  
0:00 / 12:40

Add

Lectures to be upload



ZEAL EDUCATION SOCIETY

Home Courses About Account

Home Courses Add Courses Users Sales Lectures Certificates Exit

Add Course

Data Structures and Algorithms

A Data Structure is a way of organizing and storing data so l


299

Ms. Shradha Khapra

Data Science

2

Choose File dsa-by-sandeep-jain.png



Add

## CHAPTER 4

### LOADING

---

#### 4.1 Code Snippets

##### **AdminDashboard.jsx**

```
import React, { useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";
import Layout from "../Utils/Layout";
import axios from "axios";
import { server } from "../../main";
import "../dashboard.css";

const AdminDashboard = ({ user }) => {
  const navigate = useNavigate();
  const [stats, setStats] = useState({});

  useEffect(() => {
    if (!user || user.role !== "admin") {
      navigate("/");
    } else {
      fetchStats();
    }
  }, [user, navigate]);

  async function fetchStats() {
    try {
      const { data } = await axios.get(`${server}/api/stats`, {
```

```

        headers: { token: localStorage.getItem("token") },
    });

    setStats(data.stats);

  } catch (error) {

    console.error("Error fetching stats:", error);

  }

}

const dashboardItems = [

  { label: "Total Courses", value: stats.totalCoures, link: "/admin/course", img:
"book.png" },

  { label: "Total Lectures", value: stats.totalLectures, link: "/admin/course", img:
"lectures.png" },

  { label: "Total Users", value: stats.totalUsers, link: "/admin/users", img:
"users.png" },

  { label: "New Courses", value: "", link: "/admin/addcourses", img:
"add%20course.png", buttonText: "Add Courses" },

  { label: "Update Lectures", value: "", link: "/admin/course", img:
"update%20lecture.png", buttonText: "Edit Lectures" },

  { label: "Update User Role", value: "", link: "/admin/users", img:
"update%20role.png", buttonText: "Edit Role" },

];

return (

  <Layout>

    <div className="pd">

      <div className="student-dashboard1">

        <h2 style={{ fontSize: "2rem", marginBottom:"30px" }}>Admin
Dashboard</h2>

        <div className="dashboard-content1">

```

```

    {dashboardItems.map(({ label, value, link, img, buttonText = "View" }, index)
=> (
    <div className="box" key={index}>
      <img src={`https://saisj2002.github.io/Images/${img}`} alt={label} />.
      <div>
        <p>{label}: <b>{value}</b></p>
        <button onClick={() => navigate(link)} className="common-btn">
          {buttonText}
        </button>
      </div>
    </div>
  )))
</div>
</div>
</div>
</Layout>
);
};

```

```
export default AdminDashboard;
```

### **AdminDashboard.css**

```

.dash {
  margin-left: 100px;
  padding: 2rem;
  background-color: #f1f5f9;
  min-height: 100vh;
}

```

```
.main-content {  
  background-color: #ffffff;  
  border-radius: 20px;  
  padding: 2rem;  
  box-shadow: 0 8px 24px rgba(0, 0, 0, 0.1);  
  max-width: 1200px;  
  margin: 0 auto;  
  
  .card-grid {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(260px, 1fr));  
    gap: 1.5rem;  
  }  
  
  .box {  
    background: linear-gradient(135deg, #3b82f6, #2563eb);  
    padding: 1.5rem;  
    border-radius: 16px;  
    color: #ffffff;  
    height: 180px;  
    box-shadow: 0 6px 14px rgba(0, 0, 0, 0.1);  
    transition: transform 0.3s ease, box-shadow 0.3s ease;  
    display: flex;  
    flex-direction: column;  
    justify-content: space-between;  
  
    &:hover {  
      transform: translateY(-6px);  
    }  
  }  
}
```

```
        box-shadow: 0 12px 24px rgba(0, 0, 0, 0.15);  
    }  
  
    h3 {  
        font-size: 1.2rem;  
        font-weight: 600;  
        margin-bottom: 0.5rem;  
    }  
  
    p {  
        font-size: 1rem;  
        opacity: 0.9;  
    }  
}  
}
```

## CHAPTER 5

### TESTING

---

#### **5.1 Test Strategy**

##### **Project Overview**

This document defines the test strategy for the E-learning Platform application. The platform is designed to provide an interactive learning environment where users can access courses, track progress, participate in assessments, and engage in discussions. The system supports both student and instructor roles, with administrative capabilities for course management, user management, and analytics. The purpose of this test strategy is to outline the testing approach, scope, objectives, levels, methods, and deliverables to ensure software quality and reliability before release.

##### **Objectives**

- To verify that each individual component of the application functions as per the specified requirements
- To ensure that all integrated modules interact correctly with each other
- To validate that the complete system meets the functional and non-functional business requirements
- To confirm that the application is acceptable to end-users and stakeholders
- To maintain application stability by conducting regression testing after updates and bug fixes
- To identify and rectify defects prior to deployment in the production environment



### Scope of Testing

This strategy applies to all modules and functionalities developed for the E-learning Platform. It covers the following key areas:

- User registration, authentication, and profile management
- Course management including creation, update, and removal of courses
- Learning content delivery and progress tracking
- Assessment and quiz management
- Discussion forum and communication features
- Payment processing and subscription management
- Analytics and reporting functionalities
- Role-based security and access control

### Types of Testing

Type of Testing	Purpose
Unit Testing	To verify the correctness of individual functions, classes, and methods
Integration Testing	To ensure that modules and services interact correctly when integrated
System Testing	To validate that the complete and integrated system works as intended
User Acceptance Testing (UAT)	To confirm that the system meets the expectations and requirements of end-users
Performance Testing	To verify system performance under various load conditions
Security Testing	To verify secure handling of sensitive information and proper enforcement of role-based access

## Test Levels

### 1. Unit Testing

- Individual services, repositories, and utility classes will be tested in isolation
- Performed by the development team using JUnit and Mockito
- Focus on business logic and data processing

### 2. Integration Testing

- Test interactions between components such as:
  - User authentication and authorization
  - Course enrollment and progress tracking
  - Assessment submission and grading
  - Payment processing and subscription management

### 3. System Testing

- End-to-end workflows in a controlled environment
- Coverage of all use cases and business rules
- Performance and load testing scenarios

### 4. User Acceptance Testing (UAT)

- Testing by end-users and stakeholders
- Validation of business requirements
- Usability and user experience assessment

## Test Approach

- Manual testing for UI, workflow validations, and acceptance testing
- Automated unit testing for backend services using JUnit and Mockito
- Automated integration testing for API endpoints
- Performance testing using JMeter

- Regression testing after each build release
- Systematic defect logging and tracking

### **Entry and Exit Criteria**

#### **Entry Criteria:**

- Completion of code development for the module or feature
- Availability of unit test reports with satisfactory results
- Deployment of build in the test environment
- Approval of test plans and test cases by the QA team

#### **Exit Criteria:**

- Successful execution of all planned test cases
- Resolution of all critical and major defects
- Acceptance of test summary and defect reports by project stakeholders
- Formal approval for deployment to the production environment

### **Risks and Mitigation**

<b>Risk</b>	<b>Mitigation Strategy</b>
Insufficient time for complete testing coverage	Prioritize testing of high-impact and critical business functions
External API dependency failures during testing	Implement mock services or controlled test environments
Changing requirements during acceptance testing	Maintain close communication with stakeholders and manage scope changes through formal processes
Performance issues under load	Early performance testing and optimization

## **5.2 Unit Test Plan**

### **Objective**

The objective of the Unit Test Plan is:

- To verify the correctness and functionality of individual components within the E-learning Platform
- To ensure that each software component behaves as expected in isolation
- To detect and resolve defects in the early stages of development
- To validate internal logic, data processing, and return values

### **Scope**

This Unit Test Plan covers unit-level testing for all key services, repositories, and controllers:

- User Service
- Course Service
- Assessment Service
- Progress Tracking Service
- Payment Service
- Analytics Service
- Utility Classes
- Repository Layer (mocked during testing)

### **Test Environment**

- Local development environment configured with:
  - Java Development Kit (JDK 17)
  - Spring Boot 3.x framework
  - Maven build tool
  - JUnit 5 and Mockito dependencies

**Test Cases Design**

Unit test cases will cover:

- Normal (positive) scenarios with valid inputs
- Boundary value conditions
- Exception handling with invalid inputs
- Null or empty input conditions
- Method return values and state changes
- Interaction with dependent components

**Entry and Exit Criteria****Entry Criteria:**

- Completion of module coding
- Availability of functional specifications
- Required test tools and frameworks configured

**Exit Criteria:**

- Successful execution of all unit test cases
- Resolution of all identified defects
- Achievement of at least 90% unit test coverage
- Unit test report reviewed and approved

**Deliverables**

- Unit Test Cases (JUnit test class files)
- Unit Test Execution Report
- Mock data and configuration files
- Defect Logs

### **5.3 Acceptance Test Plan**

#### **Objective**

The objective of the Acceptance Test Plan is:

- To ensure the E-learning Platform meets predefined requirements
- To verify complete system functionality from end-user perspective
- To validate business needs and operational processes
- To confirm system readiness for production deployment

#### **Scope**

This Acceptance Test Plan covers:

- End-to-end workflows and system functionalities
- Integrated application modules
- Functional and non-functional requirements
- UI and usability testing
- Compatibility and error handling

#### **Test Approach**

- Black-box testing approach
- Test cases derived from SRS document
- Manual testing and tool-supported validations
- Realistic, production-like scenarios

#### **Test Environment**

- Staging environment mirroring production setup
- PostgreSQL database with sample data
- Browser-based access for front-end testing
- Email configurations for notifications

**Test Items**

Module	Key Functions to Validate
User Management	Registration, login, profile management, role-based access
Course Management	Course creation, enrollment, content delivery
Assessment System	Quiz creation, submission, grading, feedback
Progress Tracking	Learning progress, completion status, certificates
Payment System	Subscription management, payment processing
Analytics	User engagement, course performance, revenue reports

**Test Data Requirements**

- Valid user credentials
- Sample courses and learning materials
- Test payment information
- Test email addresses

**Entry and Exit Criteria****Entry Criteria:**

- Completion of system integration and testing
- Deployment of stable build in staging
- Availability of test cases and data

**Exit Criteria:**

- Successful execution of all test cases
- No critical or major defects
- Documentation of test results