SE Project 3 Report (1)

👉 **E-Learning Platform (Team 8)**

👉 https://github.com/deafSpy/E-Learning

Hello and welcome to our project! We, Team 8, present a fully functional E-Learning App. Our motivation was to try and build a Software Engineering course within the Software Engineering course - *Inception!*

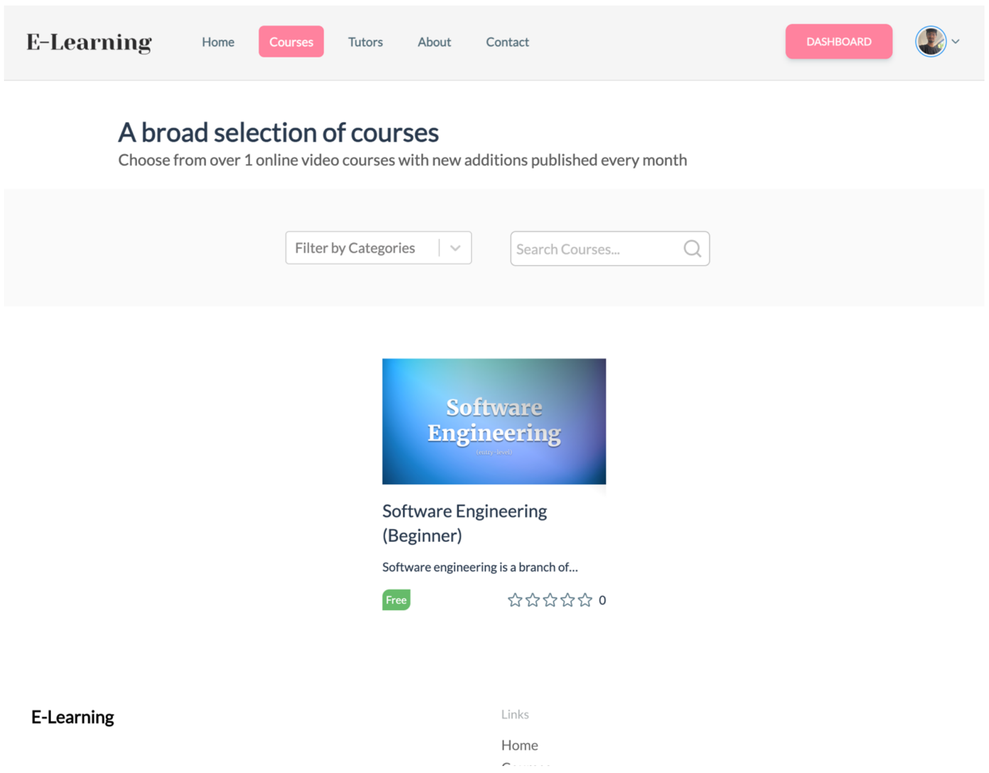
Task-1

Functional Requirements

1. **User Authentication:**
    - **Encryption and Secure Storage:**
    - Encrypting passwords ensures that even if the database is compromised, user credentials remain protected. Securely storing user sessions prevents session hijacking and unauthorised access to user accounts. We use Redis caching and JWT tokens that have a fixed duration of time before expiring. To obtain a new access token, that session is assigned a refresh token which can request for a new access token. These are encrypted by access and refresh secret keys.
    - **Role-Based Access Control (RBAC):** Implementing RBAC allows for granular control over user permissions, ensuring that instructors have the necessary privileges to create and manage courses, while students can only access enrolled courses.
    - **Password Reset Functionality:** Password reset functionality enhances user experience by providing a convenient way for users to regain access to their accounts if they forget their passwords, reducing support requests and improving user satisfaction.
    - **Enrollment Mechanism:** A seamless enrollment process allows students to easily discover and enroll in courses of interest, driving user acquisition and retention.
    - **Content Accessibility:** Uploading and storing course materials securely ensures that students can access learning resources anytime, anywhere, enhancing the flexibility and convenience of the learning experience.

2. **Dashboards:**
    - **Course List Dashboard:** Viewing List of courses for an anonymous user or those of the logged-in user.

      (anonymous)



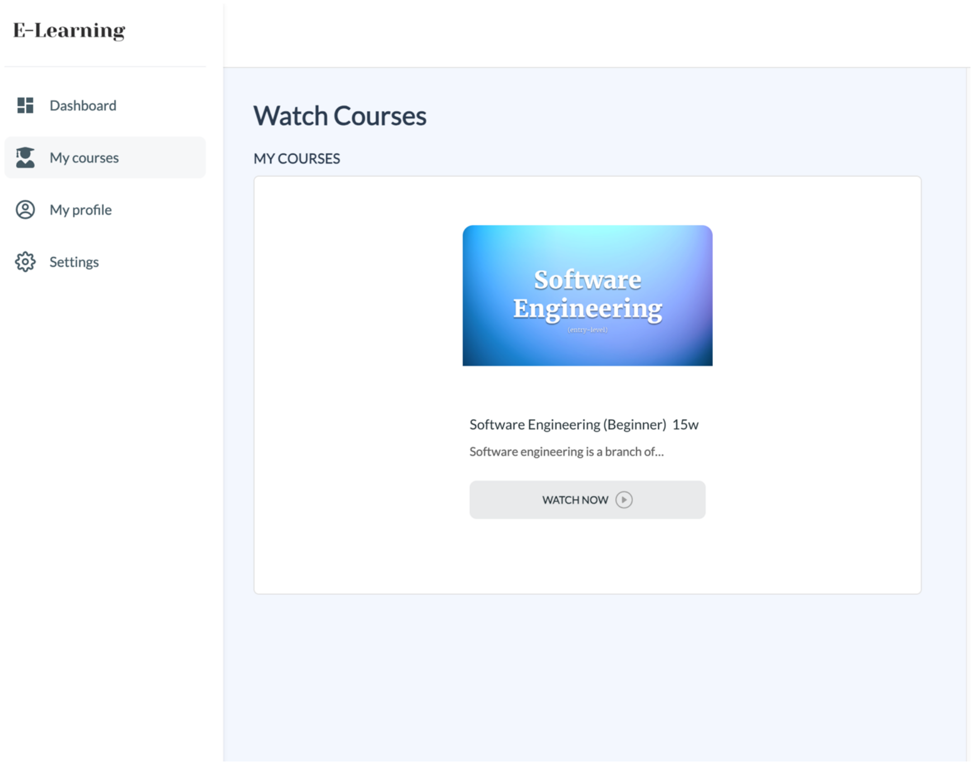
A broad selection of courses

Choose from over 1 online video courses with new additions published every month

Software Engineering (Beginner)

Software engineering is a branch of...

(Course Display 1 - User view)



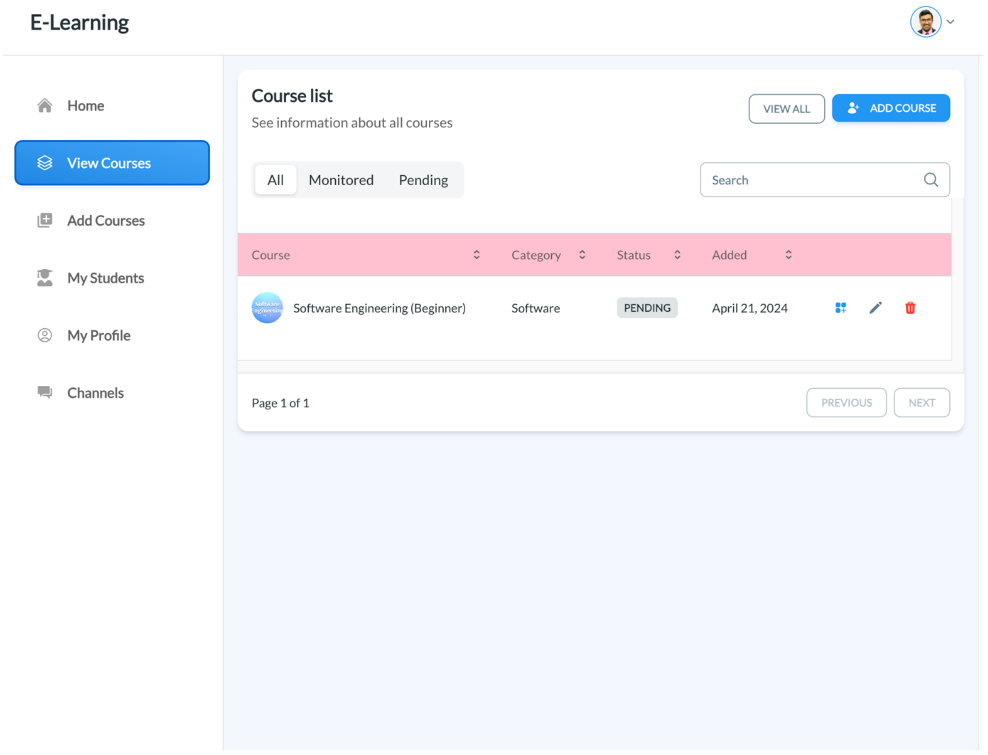
Watch Courses

MY COURSES

Software Engineering (Beginner) 15w

Software engineering is a branch of...

(Course Display 2 - User view)



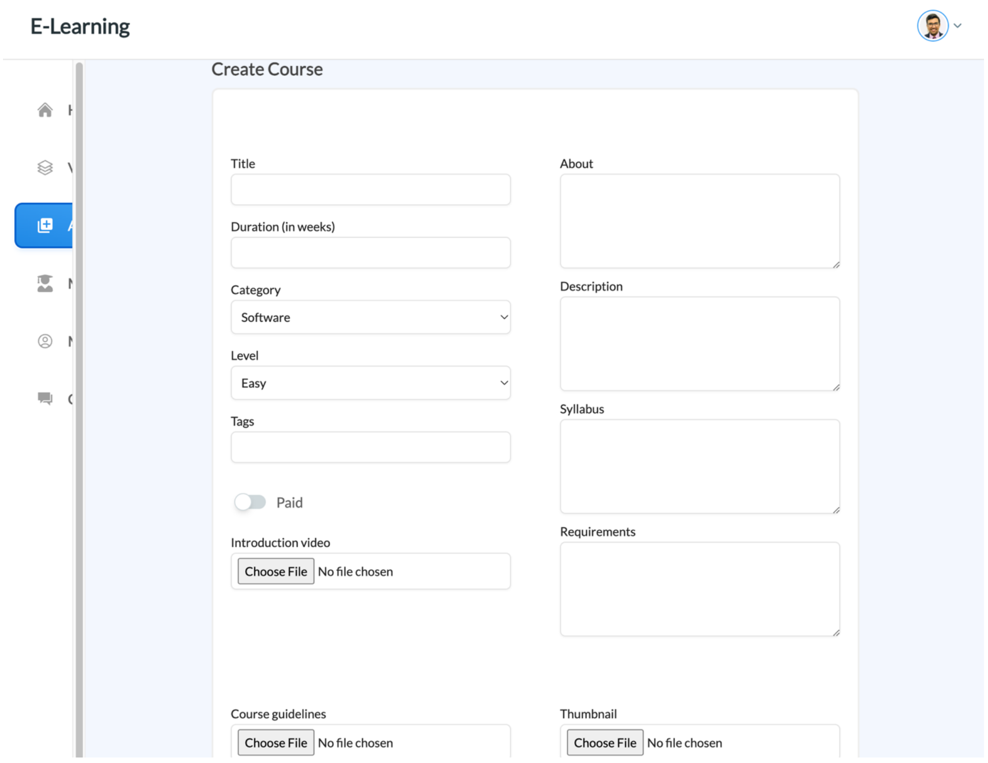
Course list

See information about all courses

Course	Category	Status	Added
Software Engineering (Beginner)	Software	PENDING	April 21, 2024

Page 1 of 1

(Course management - Instructor view)

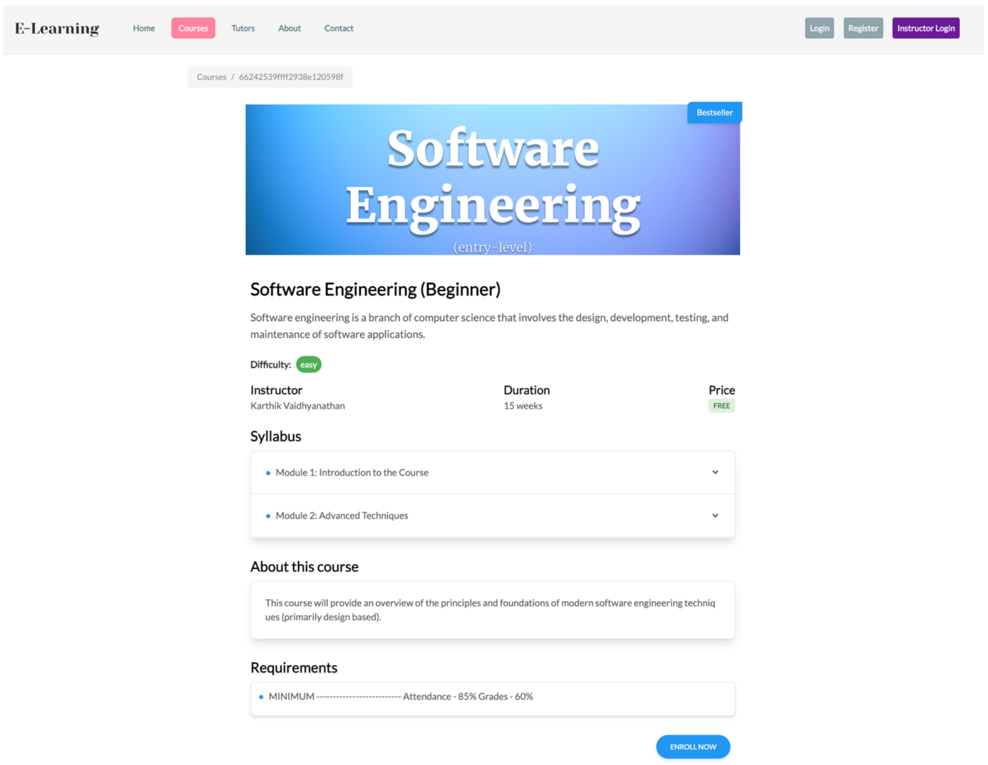


Create Course

Title	About
Duration (in weeks)	Description
Category: Software	Syllabus
Level: Easy	Requirements
Tags	
Paid	
Introduction video: Choose File No file chosen	
Course guidelines: Choose File No file chosen	Thumbnail: Choose File No file chosen

(Creating course - Instructor view)

- **Course Dashboard:** Displaying the details of the course and specific details for students including guidelines and introductory video



Software Engineering (Beginner)

Software engineering is a branch of computer science that involves the design, development, testing, and maintenance of software applications.

Difficulty: easy

Instructor	Duration	Price
Karthik Vaidhyanathan	15 weeks	FREE

Syllabus

- Module 1: Introduction to the Course
- Module 2: Advanced Techniques

About this course

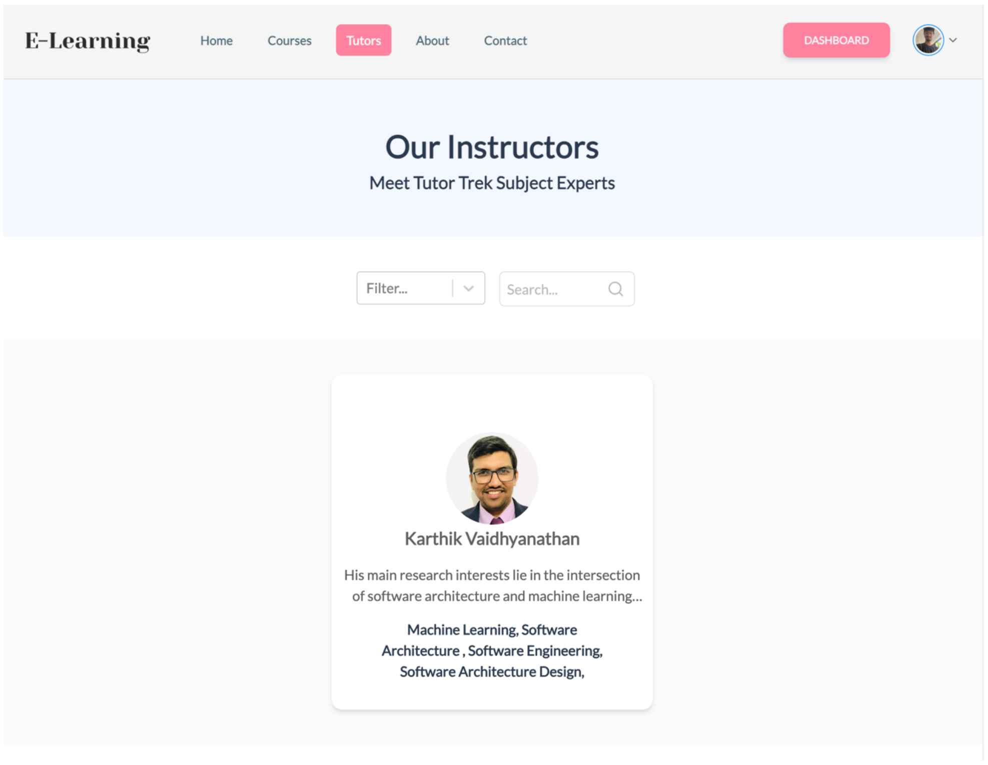
This course will provide an overview of the principles and foundations of modern software engineering techniq ues (primarily design based).

Requirements

- MINIMUM ------------------------- Attendance - 85% Grades - 60%

(Course Display - User view)

- **Instructor Dashboard:** Displaying the instructors/educators who are part of the E-learning system



Our Instructors

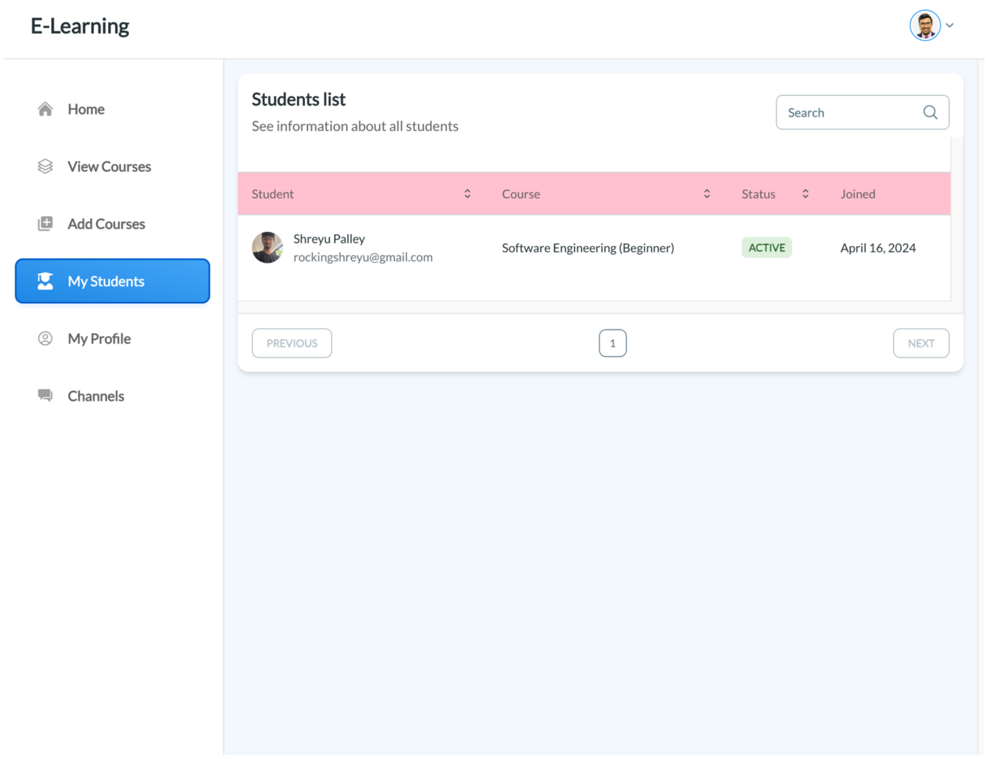
Meet Tutor Trek Subject Experts

Karthik Vaidhyanathan

His main research interests lie in the intersection of software architecture and machine learning...

Machine Learning, Software Architecture , Software Engineering, Software Architecture Design,

(Instructor Display - User view)

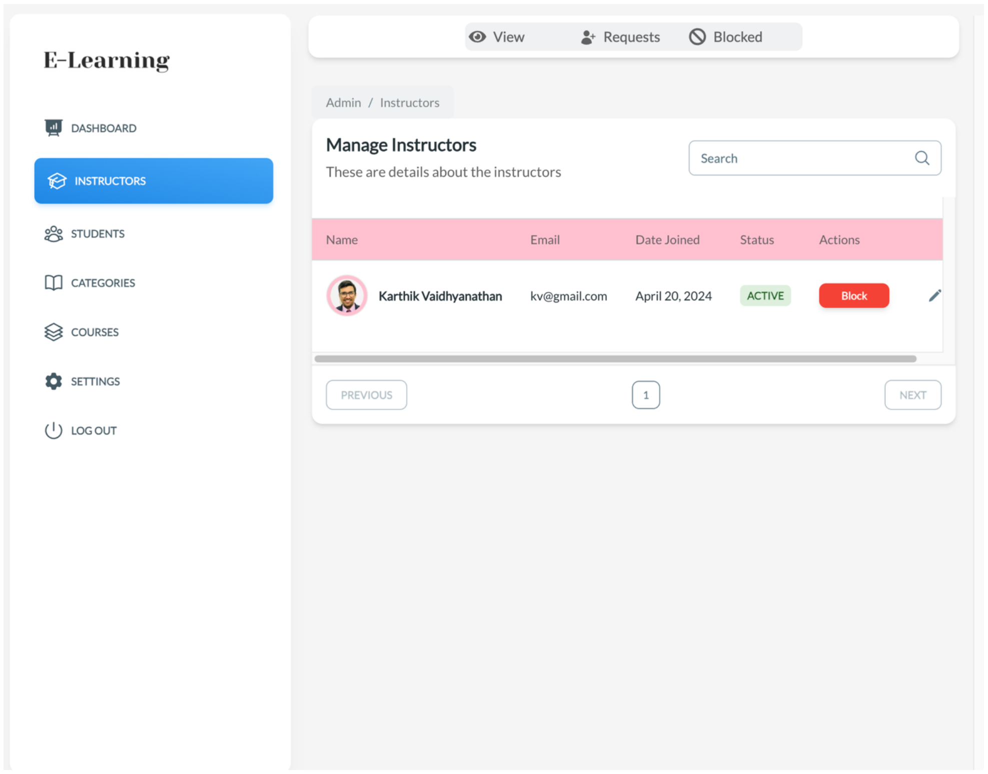


Students list

See information about all students

Student	Course	Status	Joined
Shreyu Palley rockingshreyu@gmail.com	Software Engineering (Beginner)	ACTIVE	April 16, 2024

(Student management view - Instructor view)

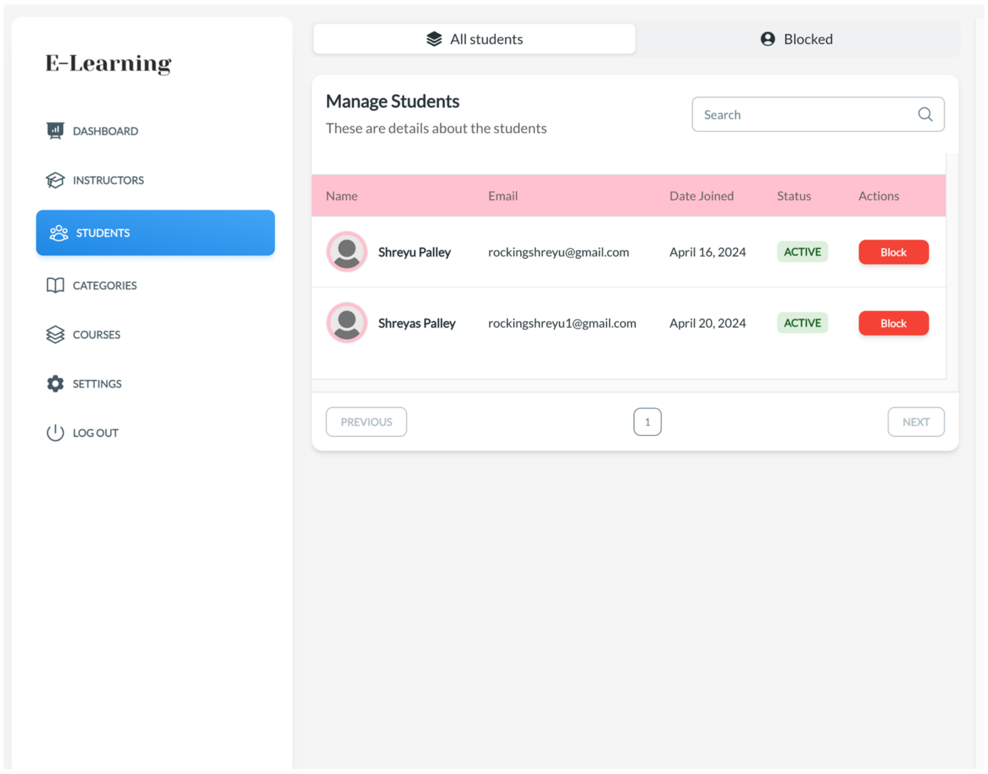


Manage Instructors

These are details about the instructors

Name	Email	Date Joined	Status	Actions
Karthik Vaidhyanathan	kv@gmail.com	April 20, 2024	ACTIVE	Block

(Instructor management - Admin view)



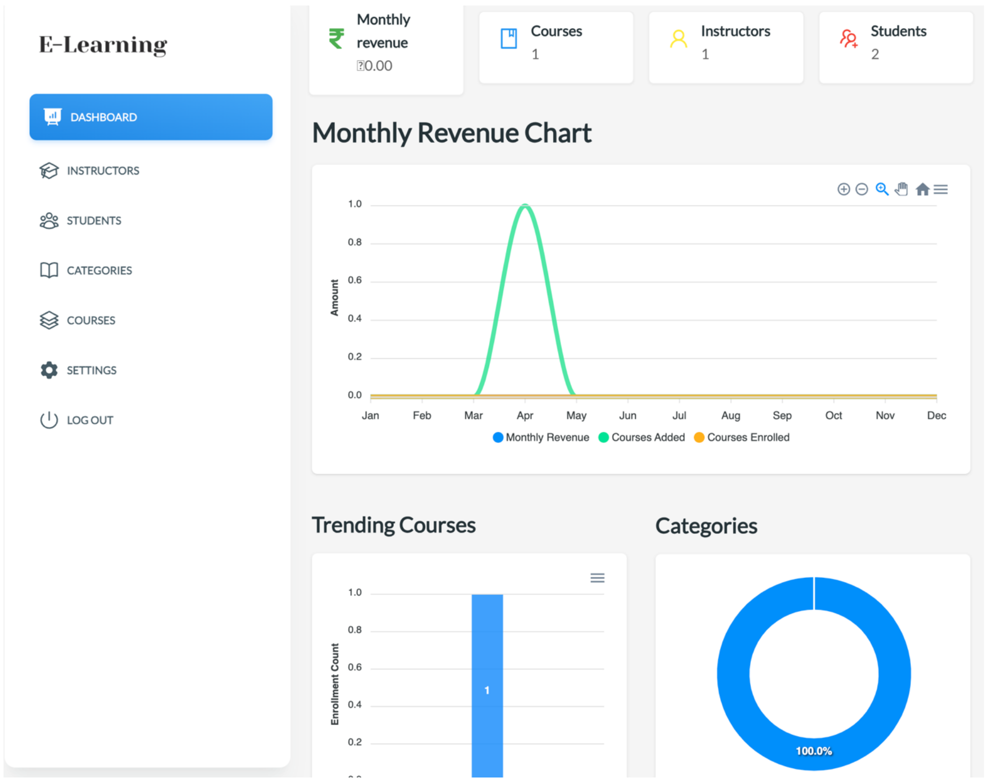
Manage Students

These are details about the students

Name	Email	Date Joined	Status	Actions
Shreyu Palley	rockingshreyu@gmail.com	April 16, 2024	ACTIVE	Block
Shreyas Palley	rockingshreyu1@gmail.com	April 20, 2024	ACTIVE	Block

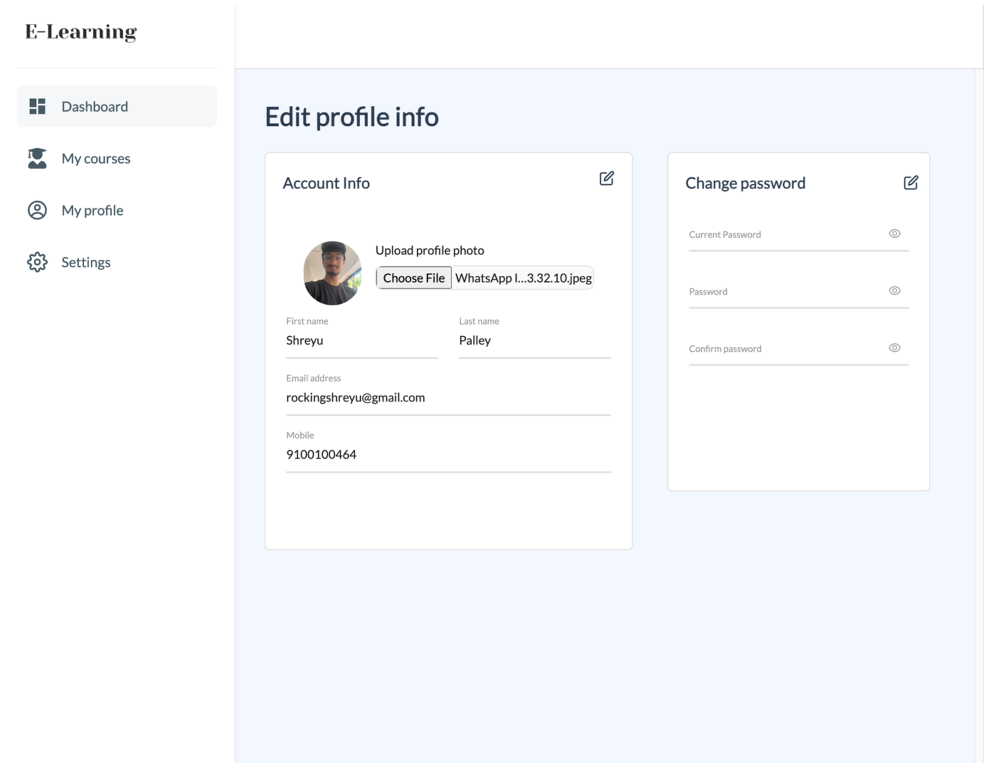
(Student management - Admin view)

- **Data Visualization:** Interactive dashboards with visualisations such as progress bars, charts, and graphs provide users with intuitive insights into their learning journey, making it easier to track progress, identify areas for improvement, and stay motivated.



(E-Learning Statistics - Admin view)

- **Real-Time Updates:** Providing real-time updates on course progress and grades keeps users engaged and informed, fostering a sense of accomplishment and facilitating timely feedback from instructors.
- **Profile/Personalization:** Tailoring dashboard views based on user roles, information, and preferences enhances usability and relevance, ensuring that users have access to the most pertinent information at their fingertips.



Edit profile info

Account Info

Upload profile photo

Choose File WhatsApp I...3.32.10.jpeg

First name: Shreyu

Last name: Palley

Email address: rockingshreyu@gmail.com

Mobile: 9100100464

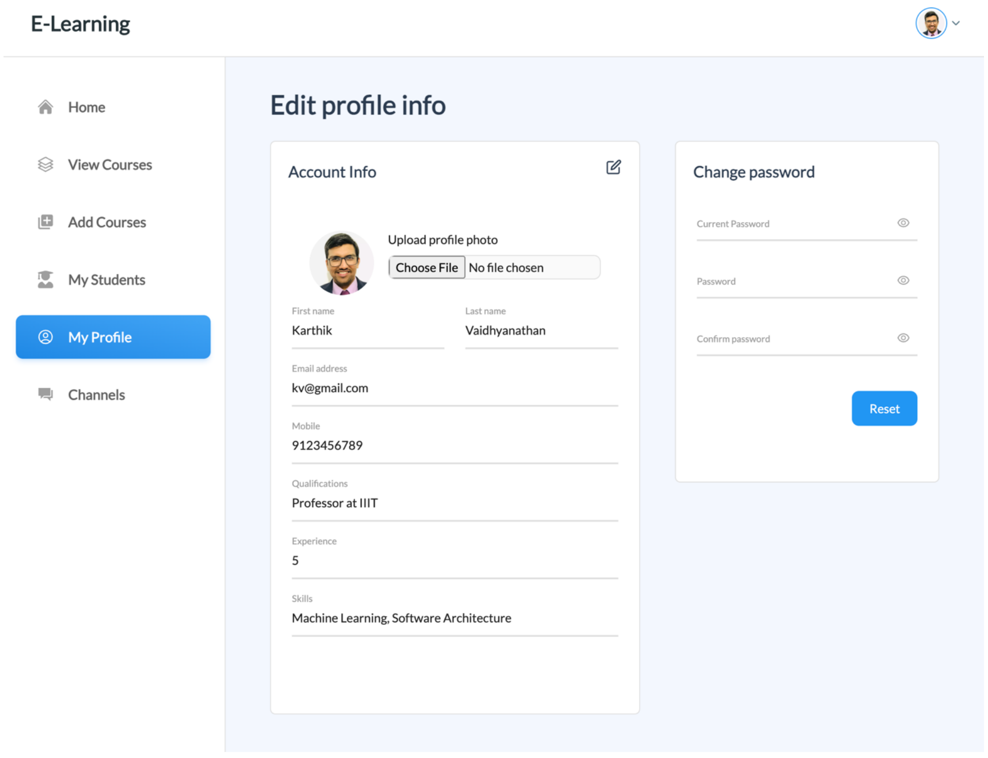
Change password

Current Password

Password

Confirm password

(Student Profile page)



Edit profile info

Account Info

Upload profile photo

Choose File No file chosen

First name: Karthik

Last name: Vaidhyanathan

Email address: kv@gmail.com

Mobile: 9123456789

Qualifications: Professor at IIIT

Experience: 5

Skills: Machine Learning, Software Architecture

Change password

Current Password

Password

Confirm password

Reset

(Instructor Profile page)

3. **Payment Processing:**
    - **Seamless Integration:** Integrating with a reputable payment gateway like Stripe streamlines the payment process, reducing friction and increasing conversion rates.
    - **Secure Transactions:** Secure payment processing with encryption and compliance with industry standards (e.g., PCI DSS) instills confidence in users and protects sensitive financial information from unauthorised access or fraud.

4. **Course Management:**

    Course Enrollment:
    - Students are able to browse through available courses, view detailed course descriptions, and enroll in courses of interest.
    - Enrolled courses are listed in the student's dashboard for easy access.
    - Enrollment status are updated in real-time to reflect changes in course availability or enrollment capacity.

    Course Collaboration:
    - Instructors have the option to invite guest lecturers or co-instructors to collaborate on course content creation and delivery.
    - Collaboration features may include shared editing capabilities, comments, and version control to facilitate seamless collaboration among course contributors.

    Feedback and Reviews:
    - Students have the ability to provide feedback and ratings for courses they have completed.
    - Instructors are able to view and respond to student feedback, fostering a culture of continuous improvement and accountability.
    - Aggregate course ratings and reviews are displayed to help prospective students make informed enrollment decisions.

    Course Progress Tracking:
    - Both instructors and students are able to track course progress, including completion status, assessment scores, and time spent on each module or lesson.
    - Progress tracking features support granular insights, such as quiz scores, assignment submissions, and discussion forum participation.

5. **Cloud Storage:**

    File Sharing and Collaboration:
    - Instructors are able to share course materials with other users or groups, facilitating collaborative learning and content sharing.
    - Shared files support fine-grained access controls to specify permissions and restrictions for different collaborators.

    Integration with Content Creation Tools:
    - The cloud storage service seamlessly integrate with popular content creation tools (e.g., Google Docs, Microsoft Office 365) to enable direct editing and synchronisation of course materials.

- Integration features may include file import/export capabilities, version history synchronisation, and real-time collaboration features.

Backup and Recovery:

- The cloud storage service provide robust backup and recovery mechanisms to protect against data loss due to accidental deletion, corruption, or system failures.
- Automated backup schedules, data redundancy, and point-in-time recovery options are available to ensure data resilience and continuity of service.

6. **Caching (Redis):**

Session Management:

- Redis are used for session management to store user session data securely and efficiently.
- Session data include user authentication tokens, user preferences, and temporary session state information.

Content Delivery Optimization:

- Redis caching are leveraged to cache static assets, such as HTML, CSS, and JavaScript files, to minimise server load and accelerate content delivery to users.
- Cached assets are served with HTTP cache headers to control caching behaviour and cache expiration policies effectively.

Search Indexing and Query Optimization:

- Redis can be used to cache search query results and frequently accessed database queries to improve search performance and reduce database load.
- Cached query results are indexed and searchable to provide fast and efficient data retrieval for search operations.

7. **Responsive Design:**

Cross-Device Compatibility:

- The user interface adapt seamlessly to various screen sizes and resolutions, ensuring consistent user experience across desktops, laptops, tablets, and smartphones.
- Navigation menus, layouts, and content are optimised for touch interaction on mobile devices.

8. **Email Service:**

Email Confirmation:

- After instructor sign up, admin can decide to reject or accept and instructor will subsequently receive an email reporting this decision

Forget Password:

- If one forgets their password, they can reset it by receiving an email and following the according steps

Non-Functional Requirements

1. **Scalability:**

- **Elastic Infrastructure**: Leveraging cloud-based services like AWS enables the platform to scale resources dynamically in response to fluctuating demand, ensuring consistent performance and availability during peak usage periods.
- **Horizontal Scaling:** Distributing workload across multiple servers or containers allows the platform to accommodate growth without sacrificing performance or incurring downtime, supporting scalability at both the application and infrastructure levels.
- The cloud storage infrastructure are designed for scalability and elasticity to accommodate fluctuations in storage demands and user activity. Auto-scaling configurations, dynamic storage provisioning, and load balancing mechanisms are implemented to ensure optimal performance and resource utilisation.
- Redis support concurrent access and scalability to handle high concurrency scenarios and scale horizontally as the application grows. Using Redis, distributed caching architectures, clustering, and sharding strategies are employed to distribute cache load and handle increasing throughput and data volume effectively.

2. **Performance/Performance Optimization:**

- **Caching Mechanism**: Implementing Redis caching reduces database load and latency by storing frequently accessed data in memory, resulting in faster response times and improved overall performance.
- **Optimised Frontend**: Employing techniques such as code splitting, lazy loading, and minification optimises frontend assets for faster rendering and smoother user interactions, enhancing perceived performance and usability.
- The course management system prioritise performance optimization techniques, such as lazy loading of course content and asynchronous data fetching, to minimise page load times and enhance user experience. Backend processes, such as course creation and enrollment updates, are optimised for efficient data processing and transaction handling to prevent bottlenecks during peak usage periods.

3. **Usability:**

- Responsive design prioritise usability and accessibility, making it easy for users to navigate, interact with, and consume content regardless of the device they're using.
- Performance considerations, such as page load times and rendering speed, are optimised to deliver a smooth and responsive user experience across devices and network conditions.

4. **Security**:

- **End-to-End Encryption:** Encrypting data both in transit and at rest ensures confidentiality and integrity, safeguarding sensitive information from interception or unauthorised access.
- **Access Controls:** Enforcing access controls at the application and database levels prevents unauthorised users from accessing restricted resources, mitigating the risk of data breaches and unauthorised modifications.
- **Security Audits and Compliance:** Regular security audits and adherence to industry standards and regulations (e.g., GDPR, HIPAA) demonstrate a commitment to security best practices and instil trust in users regarding the protection of their personal data.

5. **Accessibility:**

- The course management interface adhere to accessibility standards (e.g., WCAG) to ensure inclusivity and provide equal access to users with disabilities.

- Features such as keyboard navigation, screen reader compatibility, and alternative text for multimedia content are implemented to accommodate diverse user needs.

6. **Data Consistency and Durability:**
   - Redis ensure data consistency and durability through persistence mechanisms (e.g., snapshotting, append-only file mode) to prevent data loss in case of server failures or restarts.
   - Consistency guarantees are configurable based on application requirements, balancing performance with data durability considerations.

7. **Compliance and Data Governance:**
   - The cloud storage solution comply with industry regulations and data protection standards (e.g., GDPR, HIPAA) to safeguard user data and maintain legal compliance.
   - Data governance features, such as access controls, audit trails, and data encryption, are implemented to ensure data privacy, security, and regulatory compliance.

Subsystem Overview

Subsystem	**Role**	**Functionality**
User Management	Handles user authentication, registration, login, and profile management functionalities.	Allows users to register and log in securely, manage their profiles, update personal information, upload profile pictures, change preferences, reset passwords, and manage account settings.
Course Management	Facilitates course creation, management, exploration, and enrollment functionalities.	Provides instructors with tools to create, edit, and manage courses, including uploading course content, setting pricing, and scheduling. Allows students to explore course catalog, search, filter, sort, view course details, enroll/unenroll in courses, and save courses to Wishlist or bookmarks.
Learning Progress	Tracks and monitors learners' progress within courses.	Tracks completion status of lectures, quizzes, and assignments. Provides bookmarking or saving progress features. Offers personalized learning paths or recommendations based on learners' interests and progress. Issues badges, achievements, or certificates upon course completion.
Communication and Collaboration	Facilitates communication and interaction among learners, instructors, and peers.	Provides discussion forums or community spaces for learners to interact and engage. Offers direct messaging or chat functionality for learner-to-learner or learner-to-instructor communication. Implements Q&A features for submitting questions and receiving answers from instructors or fellow learners.
Content Creation and Management	Supports instructors in creating and organizing course content.	Offers content creation tools for developing lectures, quizzes, assignments, and other learning materials. Facilitates content organization and structuring within courses or modules. Supports multimedia content uploads,

Subsystem	Role	Functionality
		including videos, documents, and supplementary materials.
Instructor Management	Manages instructor profiles and course-related activities.	Enables instructors to create and manage their profiles, including adding bio, expertise, and social links. Provides tools for course creation, editing, and updating. Offers course analytics and insights on learner engagement and performance.
Admin Dashboard and Management	Administers platform settings, user accounts, and course content.	Provides an admin dashboard for managing platform settings, user accounts, and course content. Includes user management features for role assignment, permissions, and account moderation. Offers user analytics and reporting functionalities.
Payment Subsystem	Handles payment processing and transaction management.	Manages payment transactions for course enrollments, purchases, and subscription plans. Provides secure payment gateways, processes transactions, and ensures compliance with payment regulations. Offers features for refunds, billing management, and subscription renewals.
Storage Subsystem	Manages storage and retrieval of digital assets, course materials, and user-generated content.	Provides scalable and reliable storage solutions for storing course videos, documents, images, and user-uploaded content. Ensures data durability, accessibility, and efficient retrieval. Implements storage optimization techniques and supports integration with content delivery networks (CDNs).

Task-2

Stakeholders

*(following **IEEE 42010**)*

1. **Students/Learners:**
    - **Concerns:** Access to quality education, user-friendly interface, course variety, progress tracking, communication with instructors and peers.
    - **Viewpoint:** User Experience (UX) Viewpoint
        - **Views:**
            - **User Interface Design:** Ensuring an intuitive and easy-to-navigate interface for course exploration, enrollment, and progress tracking.
            - **Learning Experience:** Providing interactive and engaging learning materials, progress tracking features, and communication tools for collaboration and support.
2. **Instructors/Course Creators:**
    - **Concerns:** Course creation and management tools, engagement with learners, feedback mechanisms, analytics on course performance.
    - **Viewpoint:** Course Management Viewpoint

        - **Views:**
            - **Course Creation Tools:** Providing easy-to-use content creation and organization tools for creating lectures, quizzes, and assignments.
            - **Engagement and Interaction:** Facilitating communication channels, discussion forums, and feedback systems for interaction with learners.
            - **Analytics and Insights:** Offering analytics dashboards with insights into learner engagement, course performance, and feedback.

3. **Administrators/Platform Owners:**
    - **Concerns:** Platform security, user management, content moderation, revenue generation, platform analytics.
    - **Viewpoint:** System Management and Governance Viewpoint
        - **Views:**
            - **Security Measures:** Implementing robust user authentication, access controls, and data encryption to ensure platform security.
            - **User Management:** Admin dashboard for managing user accounts, roles, permissions, and moderation of user-generated content.
            - **Revenue Generation:** Integration with payment processing systems, revenue tracking, and reporting functionalities.
            - **Analytics and Reporting:** Platform analytics dashboards for tracking usage metrics, learner engagement, revenue generation, and content moderation.
4. **Content Providers/Publishers:**
    - **Concerns:** Content licensing, revenue sharing, content quality, content moderation.
    - **Viewpoint:** Content Management Viewpoint
        - **Views:**
            - **Licensing and Revenue Sharing:** Establishing agreements and mechanisms for licensing content and sharing revenue with content providers.
            - **Quality Control:** Implementing content moderation workflows, review processes, and quality assurance measures to ensure the quality and integrity of published content.
5. **Regulatory Bodies/Government Agencies:**
    - **Concerns:** Compliance with regulations, data privacy, accessibility standards, educational standards.
    - **Viewpoint:** Regulatory Compliance Viewpoint
        - **Views:**
            - **Compliance Measures:** Ensuring adherence to data privacy regulations (e.g., GDPR, COPPA), accessibility standards (e.g., WCAG), and educational standards set by regulatory bodies.
            - **Audit Trails and Reporting:** Implementing audit trails, logging mechanisms, and reporting functionalities to demonstrate compliance with regulations and standards.

Architecture Decision Records

1. Choice of Technology Stack (MERN)

Context: We needed to select a suitable technology stack for building the e-learning platform.

Decision: We decided to use the MERN stack, consisting of MongoDB, Express.js, React.js, and Node.js.

Rationale:

1. **Full-stack TypeScript:** Using a unified typescript stack simplifies development, reduces context switching, and promotes code reuse between frontend and backend components.
2. **Rich Ecosystem:** The MERN stack offers a rich ecosystem of libraries, frameworks, and tools for rapid development and scalability.
3. **Flexibility and Scalability:** Each component of the stack is highly flexible and scalable, allowing us to adapt to evolving requirements and handle increasing user traffic.
4. **Community Support:** The MERN stack has a large and active developer community, providing ample resources, documentation, and community-driven support for troubleshooting and best practices.

2. Integration with AWS for Cloud Storage

Context: We needed to choose a reliable cloud storage solution for storing course materials and other user-generated content.

Decision: We opted to integrate with Amazon Web Services (AWS) for cloud storage solutions.

Rationale:

1. **Scalability and Reliability:** AWS offers scalable and reliable cloud storage services, such as Amazon S3, suitable for storing large volumes of multimedia content.
2. **Integration Capabilities:** AWS provides robust APIs and SDKs for seamless integration with our existing architecture and development workflows.
3. **Security Features:** AWS offers comprehensive security features, including data encryption, access controls, and compliance certifications, ensuring the confidentiality and integrity of user data.
4. **Cost-Effectiveness:** AWS offers flexible pricing options and pay-as-you-go billing models, allowing us to scale storage resources according to demand and optimize costs over time.

3. Implementation of Redis for Caching

Context: We needed to improve the performance and scalability of the platform by implementing a caching solution for frequently accessed data.

Decision: We chose to implement Redis as our caching solution.

Rationale:

1. **In-Memory Data Store:** Redis is an in-memory data store, providing fast read and write operations ideal for caching frequently accessed data.
2. **Key-Value Store:** Redis offers a simple yet powerful key-value store, allowing us to store and retrieve data efficiently with low latency.
3. **Data Structures and Features:** Redis supports various data structures and features, such as lists, sets, and pub/sub messaging, enabling flexible caching strategies and use cases.
4. **Integration with Existing Infrastructure:** Redis integrates seamlessly with our existing technology stack and deployment architecture, allowing for easy implementation and management within our environment.

4. Responsive Design for Cross-Device Compatibility

Context: We needed to ensure optimal user experience across devices, including desktops, laptops, tablets, and smartphones.

Decision: We committed to implementing responsive design principles throughout the development of the platform.

Rationale:

1. **User Accessibility:** Responsive design ensures that the platform is accessible to users across various devices and screen sizes, accommodating diverse user preferences and accessibility needs.
2. **Improved Engagement:** A responsive design enhances user engagement by providing a consistent and seamless experience regardless of the device used, reducing bounce rates, and increasing user satisfaction.
3. **SEO Benefits:** Responsive design is favoured by search engines, contributing to better search rankings and visibility, which is crucial for attracting and retaining users.
4. **Future-Proofing:** With the proliferation of mobile devices and the trend towards multi-device usage, responsive design future-proofs the platform by adapting to evolving user behaviours and technological advancements.

Task-3

Architectural Tactics

1. **Horizontal Scaling:** This tactic involves adding more instances of servers or services to handle increasing load. In the E-learning system, horizontal scaling can be achieved by deploying multiple instances of serverless functions on AWS Lambda to handle user requests. This addresses the non-functional requirement of scalability by ensuring that the system can accommodate a growing number of users without degradation in performance.
2. **Caching with Redis:** Utilizing Redis as a caching layer can improve the performance of the system by storing frequently accessed data in memory. In the E-learning system, Redis can cache frequently requested user data, course information, or session tokens. This tactic enhances responsiveness and reduces latency, contributing to a better user experience.
3. **JWT Token Authentication:** Implementing JWT token-based authentication enhances security and simplifies user authentication across different services. JWT tokens are issued upon successful authentication and contain user information and access permissions. This tactic ensures secure access to resources and protects against unauthorized access, contributing to the system's security requirements.
4. **Content Delivery Network (CDN):** Leveraging AWS CloudFront or similar CDN services improves content delivery speed by caching static assets closer to the user's location. In the E-learning system, CDN can cache images, videos, and other static resources, reducing latency and enhancing user experience, especially for users accessing the platform from different geographic locations.
5. Rate Limiting (Performance Optimization): Rate limiting restricts the number of requests a client can make within a specified time interval to prevent abuse, protect against denial-of-service attacks, and optimize resource utilization. In the E-learning system, implementing rate limiting mechanisms at the API gateway or server level helps maintain system stability, prevent server overload, and ensure fair access to resources for all users.
6. Asynchronous Processing (Performance Optimization): Asynchronous processing involves decoupling time-consuming tasks from the main request-response cycle, allowing them to be executed asynchronously in the background. In the E-learning system, asynchronous processing can be used for tasks such as file uploads, email notifications, or batch data processing, improving responsiveness and user experience.
7. Fault Tolerance (Reliability): Fault tolerance ensures that the system remains operational and resilient to failures, even in the presence of hardware or software faults. Techniques such as redundant components, failover mechanisms, and graceful degradation help mitigate the impact of failures and maintain service availability. In the E-learning system, implementing fault tolerance mechanisms at various levels, such as load balancers, databases, and application servers, enhances reliability and uptime.

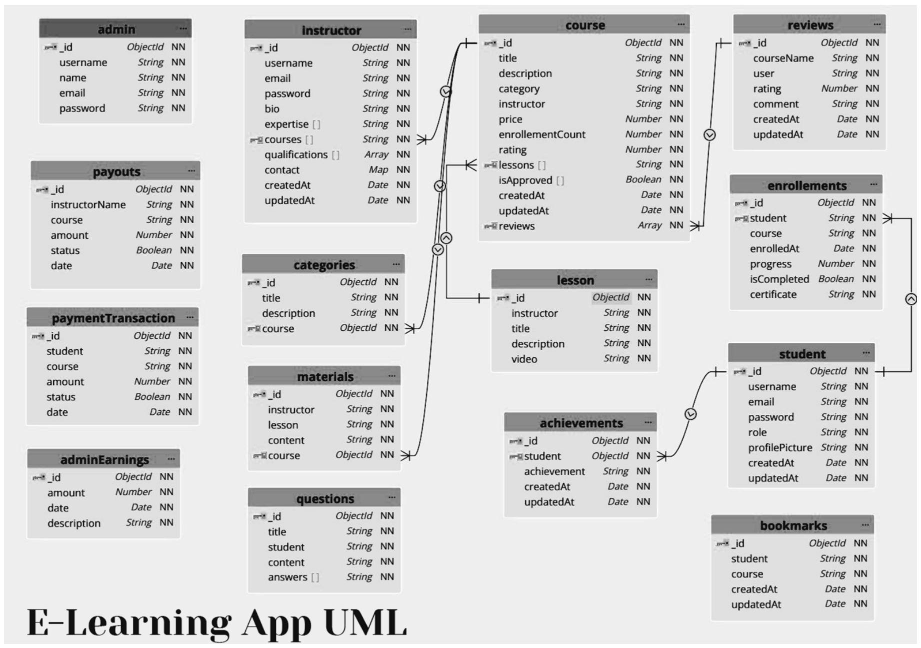
Implementation Patterns

1. **Client-Server (with MERN):** In the MERN stack, the client-server pattern is central to its architecture. The client side, constructed with React.js, serves as the user interface, interacting with users via web browsers. It communicates with the server, built with Node.js and Express.js, through HTTP requests. The server, responsible for handling business logic and data processing, interacts with the MongoDB database and serves data back to the client. This separation of concerns facilitates independent development of frontend and backend functionalities, enhances scalability, and ensures robust security measures are implemented on the server side.
2. **Model-View-Controller (MVC):** The MVC pattern separates the application into three interconnected components: Model (data storage and manipulation), View (user interface), and Controller (handles user input and updates the model). In the E-learning system, MVC helps organize code and ensures separation of concerns, making the application easier to maintain and scale.
3. **Repository Pattern:** The Repository pattern abstracts data access and manipulation logic from the rest of the application. It provides a centralized interface for accessing data stored in databases or external services. In the E-learning system, the Repository pattern facilitates data management operations, improves testability, and enables switching between different data sources seamlessly.
4. **Serverless Architecture:** Embracing serverless architecture using AWS Lambda enables deploying and running code without managing servers. Functions are executed in response to events, allowing efficient resource utilization and auto-scaling. In the E-learning system, serverless architecture reduces operational overhead, optimizes costs, and ensures scalability, as server resources are allocated dynamically based on demand.
5. **Microservices:** Adopting a microservices architecture decomposes the system into smaller, independently deployable services, each responsible for a specific business function. Microservices promote modularity, flexibility, and scalability, facilitating continuous delivery and enabling teams to work autonomously on different services. In the E-learning system, microservices enable agile development, improve fault isolation, and support scaling individual components based on usage patterns.
6. **Retry Pattern:** The retry pattern automatically retries failed operations with the expectation that they may succeed after subsequent attempts. In the e-learning app, retriesare employed for network requests, database transactions, or external service calls to improve reliability and robustness. We also use it initially when the server connects the MongoDB database, AWS s3 cloud storage, AWS Cloudfront CDN, Google OAuth, Redis client, and AWS Lambda where they retry connecting to the network.
7. **Chain of Responsibilities:** The Chain of Responsibility pattern handles requests by passing them through a series of handlers, each capable of processing or delegating the request to the next handler. This allows for flexible request handling without tightly coupling objects. In the e-learning app, this pattern manages user interactions like content viewing, assignment submissions, and support requests, distributing responsibility across specialized handlers for better modularity and scalability. In context to the API, there are multiple layers such as Model, View, Controller, API where type is reinforced through typescript interfaces and static/strict typing
8. **Facade Pattern:** The Facade pattern simplifies the complex interactions between various subsystems by providing a unified interface. In the e-learning system, the Facade pattern is used to create a simplified interface for learners, instructors, and administrators to interact with different modules such as user management, course content, and communication features.
9. **Command Pattern:** The Command pattern encapsulates a request as an object, allowing for parameterization of clients with queues, requests, and operations. In the e-learning system, this pattern is employed to implement features like queuing of asynchronous requests.
10. **Observer Pattern:** The Observer pattern defines a one-to-many dependency between objects, where changes in one object trigger updates in dependent objects. In the e-learning system, the Observer pattern notifies users or

instructors about updates, announcements, or changes in course content, ensuring timely communication and engagement.

11. **Template Method Pattern:** The Template Method pattern defines the skeleton of an algorithm in a superclass but allows subclasses to override specific steps of the algorithm without changing its structure. In the e-learning system, this pattern is utilized with typescript to define a template for course creation or lesson planning, with customizable sections for adding content, quizzes, assignments, and discussions tailored to the specific needs of each course.
12. **Filter Pattern:** The Filter pattern is a structural design pattern enabling the selection of objects from a collection based on specified criteria, without exposing the underlying implementation of the collection. In the context of an e-learning system, the Filter pattern filters courses based on attributes like category, difficulty level, instructor, or rating.

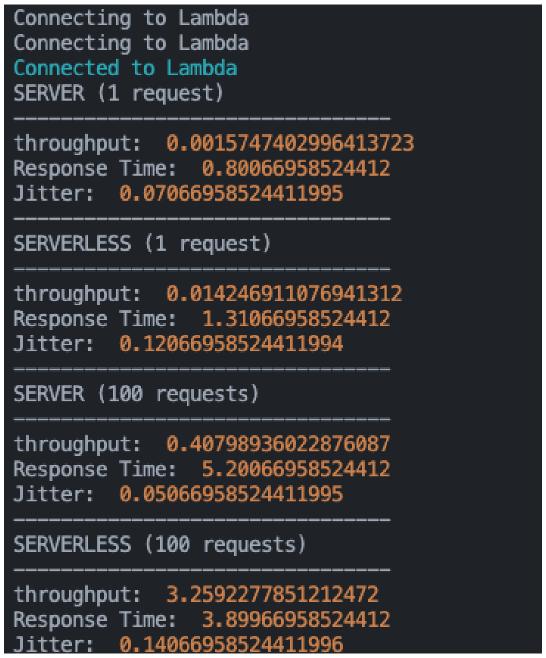
UML



admin: _id ObjectId NN, username String NN, name String NN, email String NN, password String NN
payouts: _id ObjectId NN, instructorName String NN, course String NN, amount Number NN, status Boolean NN, date Date NN
paymentTransaction: _id ObjectId NN, student String NN, course String NN, amount Number NN, status Boolean NN, date Date NN
adminEarnings: _id ObjectId NN, amount Number NN, date Date NN, description String NN
instructor: _id ObjectId NN, username String NN, email String NN, password String NN, bio String NN, expertise [] String NN, courses [] String NN, qualifications [] Array NN, contact Map NN, createdAt Date NN, updatedAt Date NN
categories: _id ObjectId NN, title String NN, description String NN, course ObjectId NN
materials: _id ObjectId NN, instructor String NN, lesson String NN, content String NN, course ObjectId NN
questions: _id ObjectId NN, title String NN, student String NN, content String NN, answers [] String NN
course: _id ObjectId NN, title String NN, description String NN, category String NN, instructor String NN, price Number NN, enrollementCount Number NN, rating Number NN, lessons [] String NN, isApproved [] Boolean NN, createdAt Date NN, updatedAt Date NN, reviews Array NN
lesson: _id ObjectId NN, instructor String NN, title String NN, description String NN, video String NN
achievements: _id ObjectId NN, student ObjectId NN, achievement String NN, createdAt Date NN, updatedAt Date NN
reviews: _id ObjectId NN, courseName String NN, user String NN, rating Number NN, comment String NN, createdAt Date NN, updatedAt Date NN
enrollements: _id ObjectId NN, student String NN, course String NN, enrolledAt Date NN, progress Number NN, isCompleted Boolean NN, certificate String NN
student: _id ObjectId NN, username String NN, email String NN, password String NN, role String NN, profilePicture String NN, createdAt Date NN, updatedAt Date NN
bookmarks: _id ObjectId NN, student String NN, course String NN, createdAt Date NN, updatedAt Date NN
E-Learning App UML

Task-4

Architecture Analysis



```
Connecting to Lambda
Connecting to Lambda
Connected to Lambda
SERVER (1 request)
--------------------------------
throughput:  0.0015747402996413723
Response Time:  0.80066958524412
Jitter:  0.07066958524411995
--------------------------------
SERVERLESS (1 request)
--------------------------------
throughput:  0.014246911076941312
Response Time:  1.31066958524412
Jitter:  0.12066958524411994
--------------------------------
SERVER (100 requests)
--------------------------------
throughput:  0.40798936022876087
Response Time:  5.20066958524412
Jitter:  0.05066958524411995
--------------------------------
SERVERLESS (100 requests)
--------------------------------
throughput:  3.2592277851212472
Response Time:  3.89966958524412
Jitter:  0.14066958524411996
```

(1 REQUEST)	Throughput	Response Time	Jitter
SERVER	0.0016	0.8007	0.0707
SERVERLESS	0.0142	1.3107	0.1207

(100 REQUESTS)	Throughput	Response Time	Jitter
SERVER	0.4080	5.2007	0.0507
SERVERLESS	3.2592	3.8997	0.1407

The trade-offs involved in the comparison between connecting to Lambda and using a traditional server architecture depend on various factors such as throughput, response time, and jitter. When connecting to Lambda, the throughput for a single request is lower compared to a traditional server, indicating that Lambda may have longer processing times for individual requests. However, Lambda demonstrates higher throughput for a larger number of requests (100 requests), suggesting better scalability and ability to handle higher loads efficiently.

On the other hand, the response time for a single request is slightly higher for Lambda compared to a traditional server, indicating that there may be additional overhead or latency associated with invoking Lambda functions. However, the response time for 100 requests is lower for Lambda, suggesting that it can handle larger workloads more effectively, resulting in shorter overall response times when scaled.

Additionally, jitter, which represents variations in response times, is slightly higher for Lambda compared to a traditional server for both single and multiple requests. This suggests that there may be more variability in the performance of Lambda functions, potentially due to factors such as cold starts or resource allocation.

Overall, the trade-offs involve considerations of performance, scalability, and consistency. While Lambda may offer better scalability and cost efficiency for handling large workloads, it may incur slightly higher response times and variability in performance compared to traditional server architectures. The choice between the two approaches depends on the specific requirements and priorities of the application, balancing factors such as cost, performance, and scalability.

Additional Code Metrics:

Personal Report

Design Decisions & Architecture:

We built the App using MERN for collaboration and maintenance as well as all having experience with it and MVC for easier maintenance, better scalability, more flexibility, and higher testability.

Learning, Challenges & Lessons:

This project was a great learning experience for all of us as we explored and tried out many AWS services, Redis, JWT, and best practices. We had already some experience with MERN, tailwind but knew only javascript so learning typescript was an added, but welcome challenge. We used typescript as it is easier to collaborate with, makes less mistakes/errors, becomes more maintainable, allows static typing (many usecases in our code). It was also easy to transition from learning software engineering in Java to Typescript over javascript with the introduction of interfaces

We had a lot of fun with the idea of making a Software Engineering Online Teaching App within the Software Engineering project as well as making Karthik Vaidhyanathan sir as an instructor within the course.

There were many challenges in getting the APIs to work and for each other to collaborate on the code as we ran into many simple bugs and errors.

Overall, we were glad to design & complete this project.