CS6.401 Software Engineering Spring 2024

Project- 3 Report

Team 23

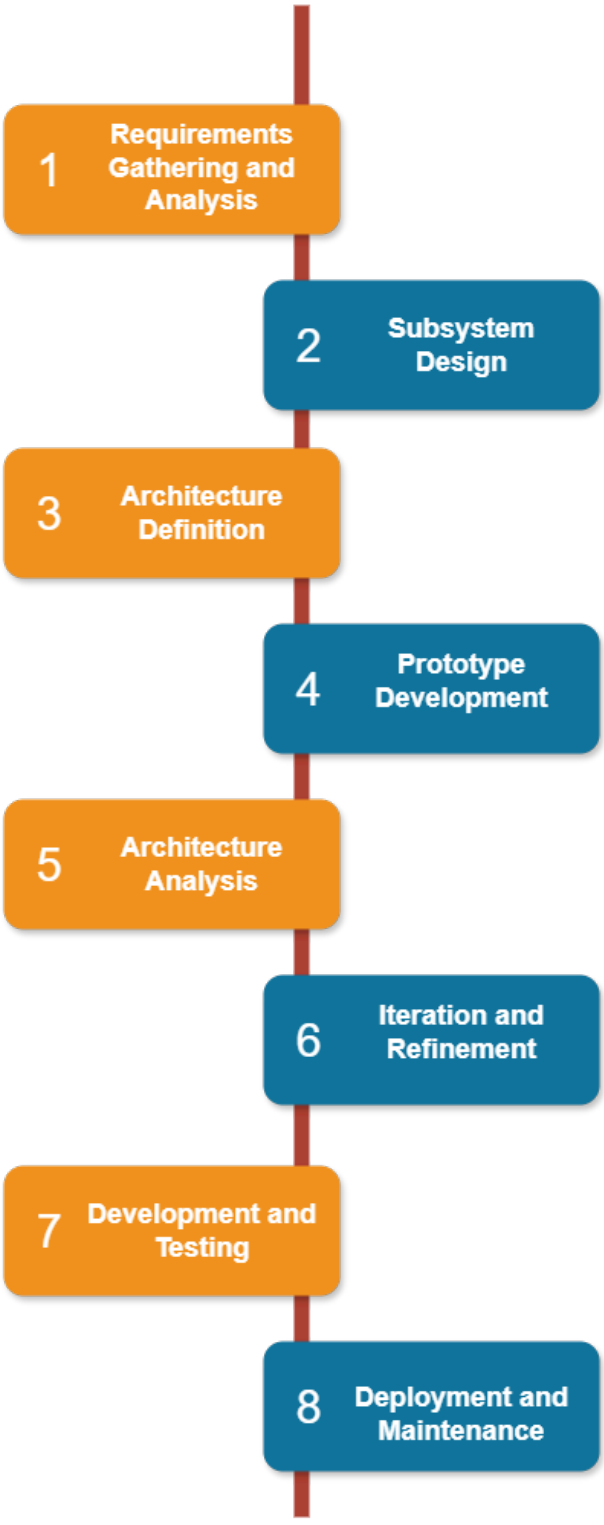
Team Members:

Name	Roll Number
Karan Bhatt	2022202003
Madhusree Bera	2022202007
Vedashree Ranade	2022201073
Yash Maheshwari	2022201074
Piyush Rana	2022202012

Repository Link: https://github.com/kbbhatt04/se_project_3

Table of Contents

Project Process



Task 1: Requirements and Subsystems

1.1.1 Functional Requirements

R1. User Registration and Authentication: Users should be able to create accounts, log in, and manage their profiles securely.
R2. Course Management: Instructors should be able to add, update, and delete course content. They should also have tools for organizing course materials, including videos, quizzes, assignments, etc.
R3. Course Exploration: Learners should be able to browse, search, filter, and compare courses based on categories, keywords, ratings, etc. provided by EduMerge as well as partner platforms
R4. Course Enrolment: Learners should be able to enroll into both paid and free courses of EduMerge and partner platforms
R5. Track learning progress: Learners should be able to track the progress of their learning in the courses they have enrolled into.
R6. Notifications and Communication: The platform should support notifications via email or in-app notifications to keep users informed about updates in enrolled courses, new courses added, etc.
R7. Payment Processing: The platform should support various payment methods for purchasing subscriptions, courses, including credit/debit cards, UPI, etc.
R8. Course Review and Rating Submission: Learners should be able to submit reviews and ratings for courses. This functionality should include a user-friendly interface for entering text-based reviews and selecting a rating score.
R9. Average Rating Calculation: Calculate and display the average rating for each course based on all submitted ratings. This average rating should be prominently displayed on the course page.
R10. Review Editing and Deletion: Learners should be able to edit or delete their own reviews after submission. However, the platform must maintain a revision history to track changes made to reviews.
R11. Review Verification: The reviews given by learners who have completed the course will be indicated clearly in the Review section.
R12. Browse Review: Users should be able to easily browse reviews and ratings to find the best courses for learning.

R13. Subscription Plan: Learners should be able to opt for various subscription plans which may include options for monthly, quarterly, or annual subscriptions, each offering different benefits such as access to paid courses of different platforms, discounts on courses, etc.
R14. Device Limitation Setting: Admin should be able to set a maximum number of devices allowed per user account. This setting can be configurable and adjustable based on platform policies and user requirements.

1.1.2 Non-functional Requirements

Performance

- The platform should ensure that users experience quick responses when interacting with the platform, such as loading pages, browsing or searching for content, playing lecture videos, etc.
- The platform should support a large number of users accessing the platform simultaneously, ensuring smooth performance even during peak usage times.
- The platform should grow seamlessly as the user base expands, allowing for increased capacity without sacrificing performance.

Security

- The platform should protect sensitive user data by encrypting information such as passwords, payment details, and personal information.
- The platform should ensure that only authorized users can access specific features and data, using secure authentication mechanisms and role-based access control.

Availability

- The platform shall maintain a high level of availability, ensuring that users can access the platform and its services reliably and without significant interruptions.
- Availability shall be maintained at a level of at least 99.99% uptime over any given 12-month period.
- The platform shall be hosted on redundant infrastructure spread across multiple data centers or regions to minimize the impact of localized failures and ensure continuous service availability.
- Load balancing mechanisms shall be employed to distribute incoming traffic evenly across redundant servers or instances, preventing overload on any single component and optimizing resource utilization.

Scalability

- The platform should be able to handle increased workload by upgrading hardware resources such as CPU, memory, and storage capacity. (Vertical Scaling)
- The platform should be able to scale out across multiple servers or cloud instances, distributing workload efficiently to accommodate growth. (Horizontal Scaling)

Usability

- The platform shall feature an intuitive user interface that allows users to easily navigate between different sections, discover courses, and access relevant features without encountering confusion or ambiguity.
- Robust search and discovery mechanisms shall be implemented to enable users to easily find relevant courses based on keywords, categories, ratings, and other filters, facilitating efficient exploration of available content.
- The platform shall be designed with responsive layouts and adaptive designs to ensure optimal viewing and interaction across various devices and screen sizes, including desktops, laptops, tablets, and smartphones.

Maintainability

- The platform should be designed with modular and reusable components to facilitate easier maintenance, updates, and enhancements.
- We must maintain clean, well-structured code following coding standards and best practices, making it easier to understand, debug, and extend.
- Proper design patterns should be used while developing so that the platform is readable, maintainable and extensible.

1.2 Subsystem Overview

1.2.1 Course Management Subsystem

The Course Management Subsystem is a central component of EduMerge, handling the creation, management, and life cycle of courses within the platform. It empowers instructors to create and deliver their knowledge through structured courses, enhancing the learning experience for users.

Key Functionalities:

- **Course Creation:** Instructors can add new courses by providing details like title, description, instructor name, difficulty level, and platform (EduMerge or external).
- **Course Organization with Categories:** Categorization allows instructors to classify their courses under relevant subject areas (e.g., Programming, Data Science, Design) improving discoverability for learners.
- **Content Management:** Beyond basic information, instructors can manage various course content types:
    - **Video Lectures:** Upload and store video files for lessons.
    - **Learning Materials:** Provide supplementary resources in Markdown format (documents, images, code snippets).
- **Course Editing and Updating:** Instructors can edit existing courses, including details, content sections, and materials.
- **Course Deletion:** A basic deletion function allows instructors to remove courses.

Course model:

```python
class Course(BaseModel):
    _id: str
    id: str
    title: str
    description: str
    instructor: str
    platform: str
    level: str
    url: list
    num_enrolled_students: int
    num_chapters: int
    is_paid: bool
    price: float
```

1.2.2 Course Exploration Subsystem

The Course Exploration Subsystem is the gateway for learners to discover and explore the vast educational resources available on EduMerge. This user-centric subsystem empowers learners to find the perfect course for their needs through powerful search functionalities and informative course details.

Key Functionalities:

- **Course Listing:** The subsystem provides a comprehensive list of all available courses within EduMerge, potentially including courses integrated from partner platforms.

- **Search and Filter:** Learners can leverage robust search features to find specific courses based on various criteria, such as:
    - **Keywords:** Search by course title, instructor name, or relevant subject keywords.
    - **Category:** Filter courses within specific subject areas (e.g., Programming, Data Science, Design).
    - **Level:** Filter based on difficulty level (e.g., Beginner, Intermediate, Advanced).
    - **Platform**: Search for courses hosted on EduMerge or integrated partner platforms.
    - **Price:** Filter based on course pricing (free, paid) or price range.

- **Course Details:** Clicking on a course listing will display comprehensive information, allowing learners to make informed decisions:
    - **Title and Description:** Understand the course content and objectives.
    - **Instructor:** Learn about the instructor's expertise and experience.
    - **Platform:** Identify the platform where the course resides (EduMerge or partner platform).
    - **Level and Learning Objectives:** Gauge the course difficulty and intended learning outcomes.
    - **Content Structure and Format:** Explore the course outline, video lectures, and supplementary materials.
    - **Ratings and Reviews:** Read reviews from other learners to get valuable insights into course quality and effectiveness.

- **Enroll Courses:** When a learner decides to enroll, the Course Exploration Subsystem facilitates course purchase by redirecting them to the next step according to the type of learner they are:
    - **Subscribed learners** are directly enrolled to the course when they select the enroll option
    - **Unsubscribed learners** are directed to the payment platform for secure transaction processing.

Interaction with Partner Integration Subsystem:

The Course Exploration Subsystem interacts with the Partner Integration Subsystem to seamlessly integrate with partner platforms like Udemy, Coursera, etc., to provide learners with a unified platform for exploring courses across diverse sources. This integration allows the subsystem to display details of partner platform courses, such as title, description, instructor, and platform source.

1.2.3 Learning Management Subsystem

The Learning Management Subsystem serves as the learner's command center for their educational journey within EduMerge. It offers a comprehensive suite of features to manage enrollment, track progress, and celebrate achievements.

Key Functionalities:

- **Enrollment Management:** Learners can seamlessly enroll in courses offered on EduMerge or integrated partner platforms. It securely connects them to the appropriate platform for course access.

- **Content Access:** Once enrolled, learners can access all course content, including:
    - **Video Lectures:** Stream or download video lectures for flexible learning.
    - **Learning Materials:** Access supplementary materials like documents, images, and code snippets.

- **Discussion Forum:** The subsystem fosters a collaborative learning environment by providing a discussion forum where learners can interact with instructors and peers, ask questions, and share insights.

- **Progress Tracking:** Learners can monitor their progress within each course. The subsystem tracks progress by the completed unit, assignments and quizzes. The learners can mark units or modules as complete as they proceed through the course.

  Progress Model

  ```
  class Progress(BaseModel):
      user_id: str
      course_id: str
      progress_details: dict
  ```

- **Certificate Issuance:** Upon successful course completion, the subsystem facilitates the issuance of certificates. This may involve:
    - **EduMerge-Hosted Courses:** Issuing certificates directly from the EduMerge platform.

- **Partner Platform Integration:** Retrieving certificates from partner platforms for courses completed on their platforms.

Interaction with Partner Integration Subsystem:

The Learning Management Subsystem interacts with the Partner Integration Subsystem to seamlessly integrate with partner platforms like Udemy, Coursera, etc., to manage learner enrollment and certificate issuance for courses hosted elsewhere. This ensures a smooth learning experience regardless of the course platform.

1.2.4 Course Review Subsystem

The EduMerge Course Review Subsystem fosters a transparent and community-driven learning environment. It allows verified learners to share their experiences with completed courses through reviews and ratings, helping others make informed decisions about their learning journeys. Additionally, it empowers course creators to manage the review landscape for their courses.

Key Functionalities:

- **Review Submission:** Verified learners who have completed a course can provide a rating on a scale of 1-5 and also submit reviews detailing their experiences. This might include feedback on the quality and clarity of the course material, insights into the instructor's teaching style and effectiveness, sharing thoughts on the course structure, value for money, and any other relevant aspects.

  Review Model:

```
class Review(BaseModel):
    user_id: str
    course_id: str
    rating: int
    review: str
```

- **Review Listing:** Learners can access a comprehensive list of all reviews for a specific course, providing diverse perspectives on the learning experience.

- **Average Rating:** The subsystem calculates and displays an average rating for each course based on submitted reviews. This provides a quick reference point for course quality.

- **Review Management for Course Creators:** Course creators can:
  - **View All Reviews:** Read through all submitted reviews for their courses.

    - **Flag Inappropriate Reviews:** Flag reviews that violate platform guidelines for further moderation.
    - **Moderate and Delete Reviews:** Implement a moderation process to review flagged reviews and remove any that violate platform guidelines.
    - **Respond to Reviews:** Optionally, course creators can respond to reviews, addressing learner feedback and fostering communication.

- **Verification for Reviews:** The Course Review Subsystem prioritizes a secure and respectful environment. To ensure this, the platform may require learners to complete a course before submitting a review, preventing reviews from those who haven't experienced the content.

1.2.5 Payment Subsystem

The EduMerge Payment Subsystem facilitates secure and seamless financial transactions for course purchases within the platform as well as payments for getting the subscription of the platform.

Key Functionalities:

- **Payment Processing:** The subsystem integrates with secure payment gateways to handle various payment methods commonly used by learners, such as credit cards, debit cards, and UPI.

  PaymentData Model:

```
class PaymentData(BaseModel):
    user_id: str
    course_id: str
    payment_method: str
```

- **Flexible Payment Options:** The subsystem supports a variety of payment models offered by instructors, including:
    - **One-time Payment:** Charge learners a single upfront fee for course access.
    - **Subscription-based Courses:** Facilitate recurring payments for courses with ongoing content or access.
    - **Free Courses:** Allow instructors to offer courses at no cost to learners.

- **Secure Transactions:** The payment subsystem prioritizes security by:
    - Partnering with reputable payment gateways that adhere to industry security standards (e.g., PCI DSS).
    - Utilizing secure data encryption methods to protect learner financial information.

1.2.6 Subscription Subsystem

The Subscription Subsystem provides learners with a cost-effective way to access a vast library of educational content through tiered subscription plans. This subsystem caters to learners who desire ongoing access to a broad range of courses, fostering continuous learning and exploration.

Key Functionalities:

- **Subscription Tiers:** EduMerge offers various subscription tiers, each providing access to a defined set of benefits.
    - **Course Access:** Subscription plans grant access to a curated selection of paid courses from EduMerge and potentially partner platforms (Udemy, Coursera, etc.). Higher tiers might offer access to a wider range of courses.
    - **Discounted Courses:** Subscribers might receive exclusive discounts on additional course purchases beyond those included in the subscription.
    - **Early Access:** In some cases, subscribers might receive early access to newly released courses before they become available for individual purchase.

- **Subscription Management:** Learners can easily:
    - **Subscribe:** Select and enroll in a subscription tier that aligns with their learning goals and budget.
    - **Manage Subscriptions:** View, update, or cancel their subscriptions at any time.
    - **Track Usage**: Monitor their access to courses included in their subscription plan.

- **Subscription Periods:** EduMerge offers flexible subscription periods to cater to learner preferences:
    - **Monthly Subscription:** Provides access for a single month, ideal for learners who prefer short-term commitments.
    - **Quarterly Subscription**: Offers access for three months, balancing affordability with extended access.
    - **Annual Subscription:** Grants access for a full year, providing the most cost-effective option for learners seeking long-term learning journeys.

Interaction with Partner Integration Subsystem:

The Subscription Subsystem interacts with the Partner Integration Subsystem to seamlessly integrate with partner platforms to provide access to their course libraries.

1.2.7 User Management Subsystem

The User Management Subsystem serves as the foundation for user interaction with the platform. It facilitates secure user registration, login, and authorization, ensuring a seamless experience for learners, instructors, partners, and administrators.

Key Functionalities:

- **User Registration:** The subsystem allows users to create accounts on the EduMerge platform. Different registration options cater to various user types:
    - **Learner:** Learners can register to enroll in courses, access learning materials, and participate in discussions.
    - **Instructor:** Instructors can register to create and manage courses, upload content, and track student progress.
    - **Partner:** Partners (Udemy, Coursera, etc.) can register to integrate their platforms with EduMerge for course discovery and subscription offerings.
    - **Admin:** Administrators can register to manage the platform, user accounts, system configuration, and access control.

- **User Login and Session Management:**
    - **Login:** Users can securely log in using their registered credentials (username or email and password).
    - **Session Management:** The subsystem manages user sessions, ensuring a secure connection throughout their activity on the platform. Sessions might have timeouts to prevent unauthorized access if left inactive.
    - **Multi-factor Authentication (MFA):** EduMerge might offer optional multi-factor authentication (MFA) for an additional layer of security. This could involve verification codes sent via SMS or authenticator apps upon login attempts.

- **Authorization and Role-based Access Control (RBAC):** The subsystem assigns specific permissions and access levels based on user roles (learner, instructor, partner, admin).
    - **Learner Permissions:** Learners can typically access course materials, complete assessments, and engage in discussions within their enrolled courses.
    - **Instructor Permissions:** Instructors might have additional permissions to manage their courses, create content, view student progress, and communicate with learners.
    - **Partner Permissions:** Partner accounts might have specific access for course integration, data exchange, and potential revenue-sharing functionalities.
    - **Admin Permissions:** Administrators have comprehensive access to manage the platform, user accounts, system settings, and handle security measures.

- **Devices Logged In Management:** This functionality can refer to tracking the devices a user has logged in to EduMerge from. It can serve for:
    - **Security:** Alerting users about login attempts from unrecognized devices, potentially indicating unauthorized access.
    - **Session Management:** Allowing users to remotely log out of sessions on forgotten devices to prevent unauthorized access.

1.2.8 Administration Subsystem

The Administration System empowers designated personnel with the tools to manage the platform's day-to-day operations, ensuring smooth functionality and a positive user experience. Administrators can leverage this system to perform a variety of critical tasks.

Key Functionalities:

- **User Management:** Administrators have comprehensive control over user accounts:
    - **User Creation and Deletion:** Add new users (learners, instructors, partners) or remove existing ones as needed.
    - **User Role Management:** Assign appropriate user roles (instructor, learner, partner) and manage associated permissions for each role.
    - **User Activity Monitoring:** Optionally monitor user activity logs to track platform usage and identify potential issues.

- **Content Management:** Administrators can oversee the vast repository of educational content on EduMerge:
    - **Course Management:** Manage the overall course lifecycle, including reviewing course proposals, approving new courses, and potentially taking down courses that violate platform guidelines.
    - **Content Review:** Review uploaded content for adherence to quality standards and platform policies.
    - **Content Organization:** Categorize and organize courses to ensure discoverability for learners.

- **Platform Configuration:** Administrators can fine-tune the platform's settings and functionality:
    - **System Settings:** Manage system-wide configurations like payment processing options, email notifications, and user interface customization.
    - **Integration Management:** Oversee integration with partner platforms for course discovery and subscription offerings.
    - **Security Management:** Implement security measures like user authentication protocols, data encryption, and system access controls.

- **Analytics and Reporting:** Administrators can gain valuable insights into platform usage and user behavior:
    - **Course Performance Analysis:** Analyze learner engagement, completion rates, and feedback for individual courses.
    - **User Activity Reports:** Generate reports on user enrollment trends, content access patterns, and overall platform activity.
    - **Subscription Analytics:** For the subscribed learners, monitor their behavior, analyze subscription trends, and identify areas for improvement.

1.2.9 Partner Integration Subsystem

The Partner Integration Subsystem bridges the gap between EduMerge and external learning platforms like Udemy or Coursera. It facilitates seamless communication and data exchange, enabling EduMerge to offer a broader range of courses and enhance the learning experience for users.

Key Functionalities:

- **Partner Portal:** The Partner Portal serves as a dedicated platform for potential and existing partners. It offers functionalities like:
    - **Apply as Partner:** Educational platforms can submit applications to integrate their courses with EduMerge.
    - **Approval Process:** EduMerge reviews partner applications based on predefined criteria, ensuring quality and alignment with platform goals.
    - **Service Level Agreements (SLAs):** Upon approval, EduMerge establishes SLAs with partners outlining terms of collaboration, data exchange protocols, and revenue sharing models (if applicable).
    - **API Access:** Approved partners receive access to EduMerge APIs, enabling them to provide their course data and functionalities.

- **Communication and Data Exchange:** The subsystem facilitates secure communication between EduMerge and partner platforms:
    - **Course Information Retrieval:** EduMerge retrieves key information about partner platform courses, including titles, descriptions, instructors, and learning objectives.
    - **Enrollment Facilitation:** EduMerge might integrate with partner platforms to allow learners to seamlessly enroll in courses offered on those platforms. This could involve:
        - **Single Sign-On (SSO):** Learners can leverage EduMerge credentials to access partner platform courses, eliminating the need for separate logins.
        - **Payment Processing:** For subscriptions or individual course purchases on partner platforms, EduMerge might handle payments centrally or redirect learners to the partner platform's payment gateway.

Interaction with Other Subsystems:

The Partner Integration Subsystem interacts with other core EduMerge functionalities:

- **Course Exploration Subsystem:** Retrieved course information from partners populates the course listings displayed on EduMerge.

- **Learning Management Subsystem:** For enrollment facilitation, the subsystem might integrate with partner systems to track learner progress and issue certificates for completed partner platform courses.

- **Subscription Subsystem:** The Partner Integration Subsystem can interact with the Subscription Subsystem to potentially offer partner platform courses within EduMerge subscriptions.

1.2.10 Web Application Subsystem

The Web Application Subsystem serves as the user interface (UI) that learners, instructors, and administrators interact with to access the platform's functionalities. This responsive web application ensures a seamless experience across various devices, from desktops and laptops to tablets and smartphones.

Key Functionalities:

- **Intuitive User Interface:** The subsystem delivers a user-friendly and visually appealing interface that caters to diverse user types:
    - **Learners:** Explore courses, enroll, access learning materials, participate in discussions, track progress, and potentially receive certificates.
    - **Instructors:** Create and manage courses, upload content, interact with learners, view progress reports, and manage course settings.
    - **Partners:** Manage course integration through the Partner Portal (separate interface).
    - **Administrators:** Manage users, content, platform settings, access reports, and oversee overall platform operations.

- **Responsive Design:** The web application adapts its layout and functionality to different screen sizes, ensuring optimal user experience on any device, be it a desktop computer, tablet, or mobile phone.

- **Accessibility:** The subsystem prioritizes accessibility by incorporating features like:
    - **Keyboard Navigation:** Users can navigate the application and access features using just a keyboard.
    - **Screen Reader Compatibility:** The UI is compatible with screen reader software for visually impaired users.
    - **High Contrast Mode:** An optional high contrast mode can be implemented to improve user interface readability for those with visual impairments.

Task 2: Architecture Framework

2.1 Stakeholder Identification

Learners:

Learners seek to explore, purchase, and review courses from various platforms in a centralized location.

- **Concerns:**
    - Difficulty finding the best course for their needs across multiple platforms.
    - Limited information or unreliable reviews on available courses.
    - Security and privacy concerns regarding personal data and financial transactions.
    - User-friendliness and smooth navigation of a platform integrating various interfaces.
- **Viewpoints and Views:**
    - Users desire a well-organized platform with comprehensive search functionalities and reliable course reviews to make informed decisions.
    - They value a secure and user-friendly experience with clear privacy policies.

Course creators:

Experts offer their knowledge through online courses and potentially reach a wider audience through platform integration.

- **Concerns:**
    - Reaching a wider audience beyond their own platform.
    - Fair revenue sharing agreements for integrated courses.
    - Preserving ownership and control over their course content.
    - Platform visibility and reputation impacting their course reach.
- **Viewpoints and Views:**
    - Creators would value seamless integration with minimal setup and a wider learner base through the platform.
    - Clear and fair revenue sharing models and intellectual property protection are essential.

Admins:

Platform administrators who manage the platform operations, user accounts, course listings, and ensure smooth functioning of the platform.

- **Concerns:**
    - Managing a large user base and course listings from various platforms.
    - Ensuring data security and compliance with regulations.
    - Maintaining system performance and scalability.

    - Content moderation and preventing plagiarism.
- **Viewpoints and Views:**
    - Admins need robust platform tools for user management, course listing integration, and data security.
    - Scalability is crucial to accommodate platform growth.

Platform Provider (Udemy, Coursera, etc.):

Partners with our platform to integrate their course offerings for wider user reach and potential revenue sharing.

- **Concerns:**
    - Maintaining control over their branding and user experience within the platform.
    - Sharing user data responsibly and ensuring fair revenue distribution.
    - Integration effort and technical compatibility.
- **Viewpoints and Views:**
    - Providers seek clear agreements regarding data access, branding, and user experience within the platform.
    - They expect seamless integration with minimal technical overhead.

Investor:

Individual or organization who provides financial backing and expects a return on investment through platform growth and revenue generation.

- **Concerns:**
    - Profitability of the platform and return on their investment.
    - Scalability and growth potential of the platform in the market.
    - Sustainability of the business model and competitive landscape.
- **Viewpoints and Views:**
    - Investors require a solid business plan outlining the platform's growth strategy, revenue streams, and competitive edge.
    - Demonstrating a sustainable model with clear revenue generation potential is crucial.

Developers/Designers/Maintainers:

Professionals responsible for building and maintaining the platform and ensuring the platform meets user requirements, and enhancing the platform's functionality and user experience.

- **Concerns:**
    - Designing a user interface that integrates various platforms and functionalities.
    - Maintaining technical standards and ensuring platform security.
    - Adapting to changing industry trends and user demands.

- **Viewpoints and Views:**
    - Developers need clear API documentation and support from platform providers to facilitate seamless integration.
    - They require security measures and best practices to maintain platform integrity and user trust.

Payment Gateways:

Service providers securely processing online course purchases and facilitating transactions between users and course creators/platforms.

- **Concerns:**
    - Security of transactions and fraud prevention.
    - Integration with the platform's payment system.
    - Transaction fees and revenue sharing agreements.

- **Viewpoints and Views:**
    - Payment gateways prefer secure and well-established platforms.
    - Clear integration guidelines and fair fees are essential for collaboration.

Regulators:

Authorities that ensure the platform adheres to data privacy, consumer protection, and other relevant regulations.

- **Concerns:**
    - Data privacy and user consent for data collection and usage.
    - Consumer protection regarding course quality and refunds.
    - Intellectual property rights and plagiarism issues.
- **Viewpoints and Views:**
    - The platform needs to prioritize data security and adhere to data privacy regulations.
    - Clear terms of service and dispute resolution mechanisms are crucial for consumer protection.

2.2 Major Design Decisions:

ADR-1: Selection of Microservices Architecture

Decision: Implement EduMerge as a microservices architecture.

We opted for a microservices architecture for EduMerge. This approach decomposes the platform into smaller, independent services responsible for specific functionalities (e.g., User Management, Course Management, Payment Processing). This enables independent scaling and deployment based on service needs, promoting better maintainability, fault isolation, and technological flexibility in adopting the most suitable tech stack for each service.

Rationale:

- **Scalability and Maintainability:** Breaking down the platform into smaller, independent services facilitates independent scaling and deployment based on specific functionalities (e.g., User Management, Course Management, Payment Processing). This allows for easier maintenance and future enhancements.
- **Fault Isolation:** If one microservice encounters an issue, it won't bring down the entire platform. Other services can continue functioning, minimizing downtime and improving platform resilience.
- **Technological Flexibility:** Microservices architecture allows for the use of different technologies for different services based on specific needs. This promotes flexibility in adopting the most suitable technology stack for each service.

ADR-2: Database Selection for Course Content

Decision: Implement a document database (MongoDB) to store course content.

For course content storage, we decided to implement a document database like MongoDB. Document databases excel at handling the unstructured and semi-structured nature of course content (text, video URLs, images). They also scale horizontally to accommodate large data volumes, and their ability to store course content along with metadata within a single document simplifies data retrieval and management.

Rationale:

- **Flexibility and Scalability:** Document databases excel at handling unstructured and semi-structured data like course content (text, video URLs, images). They can easily scale horizontally to accommodate large amounts of data.
- **Rich Content Management:** Document databases allow for storing course content along with metadata (e.g., instructor information, learning objectives) within a single document, simplifying data retrieval and manipulation.

ADR-3: Authentication and Authorization Strategy

Decision: Implement a token-based authentication system with role-based access control (RBAC).

To balance security and scalability, we will implement a token-based authentication system with role-based access control (RBAC). Token-based authentication offers a secure way to manage user sessions without storing sensitive credentials on the server and scales well for a large user base. RBAC assigns permissions based on user roles (learner, instructor, administrator), ensuring users can only access functionalities relevant to their role, further enhancing platform security.

Rationale:

- **Security:** Token-based authentication provides a secure way to manage user sessions without storing sensitive credentials like passwords on the server.
- **Scalability:** Tokens are stateless, reducing load on the authentication server and enabling horizontal scaling for handling a large user base.
- **Role-based Access Control (RBAC):** Assigning permissions based on user roles (learner, instructor, administrator) ensures users can only access functionalities relevant to their role, enhancing platform security.

ADR-4: Technology Stack for User Interface and Backend

Decision:

- **Frontend: Utilize a JavaScript framework like React or Angular for the EduMerge web application frontend.**

  We have opted for a technology stack that leverages both popular and efficient choices for the EduMerge web application. On the frontend, we will utilize a JavaScript framework like React or Angular. These frameworks are well-suited for building dynamic and responsive user interfaces, ensuring a smooth and engaging experience for all users. Their component-based architecture promotes code reusability and maintainability, which is crucial for a large-scale web application. Additionally, these frameworks boast vast developer communities with extensive learning resources, simplifying the development process and troubleshooting efforts.

- **Backend: Implement the EduMerge backend using Python and the FastAPI framework.**

  For the backend, we will implement EduMerge using Python and the FastAPI framework. Python is a popular backend language known for its readability, extensive libraries, and large developer community. FastAPI, built on top of Python, is a high-performance web framework ideal for creating modern APIs that seamlessly integrate with the frontend. This combination of Python and FastAPI offers a balance of readability, performance,

and a productive development environment, providing a solid foundation for building EduMerge.

Rationale:

- **Frontend:** Rich and Interactive UI: JavaScript frameworks like React or Angular excel at creating dynamic and responsive user interfaces, providing a smooth and engaging user experience for learners, instructors, and administrators. Their component-based architecture promotes code reusability and maintainability for a large-scale web application. Additionally, these frameworks benefit from vast developer communities with extensive learning resources, simplifying development and troubleshooting.
- **Backend:** Python and FastAPI: Python is a popular choice for backend development due to its readability, extensive libraries, and large developer community. FastAPI, a high-performance web framework built on top of Python, is ideal for building modern APIs that seamlessly integrate with the frontend. This combination offers efficiency, maintainability, and a productive development environment.

ADR-5: Use of API Gateway

Decision: Implement an API Gateway for EduMerge.

An API Gateway centralizes API access, improving security, maintainability, and scalability. It acts as a single entry point for requests, handling authentication, authorization, traffic management, and potentially caching. This decouples clients from backend services, simplifying future changes.

Rationale:

- **Single Point of Entry:** An API Gateway serves as a single entry point for all API requests coming from the frontend or external partner integrations. This centralizes authentication, authorization, rate limiting, and traffic management, simplifying client-side development and improving overall API security.
- **Improved Maintainability:** The API Gateway decouples the frontend and mobile clients from the backend services. This isolation simplifies future changes to backend services without requiring modifications on the client-side as long as the API Gateway maintains the same API interface.
- **Scalability and Performance:** API Gateways can be horizontally scaled to distribute incoming traffic across multiple backend instances, improving overall platform performance and scalability. Additionally, API Gateways can implement caching mechanisms to reduce the load on backend services for frequently accessed data.

ADR-6: Use of Content Delivery Network (CDN) for Content Delivery

Decision: Implement a Content Delivery Network (CDN) for efficient delivery of EduMerge content.
We've decided to implement a Content Delivery Network (CDN) to optimize EduMerge's content delivery. CDNs store static content (images, videos, JavaScript) on geographically distributed servers. This brings content closer to users, resulting in faster loading times and a smoother overall experience. Additionally, CDNs can handle traffic spikes and improve platform availability. By offloading static content, CDNs can also potentially reduce operational costs. We'll choose a CDN provider based on factors like global reach, performance, and pricing.

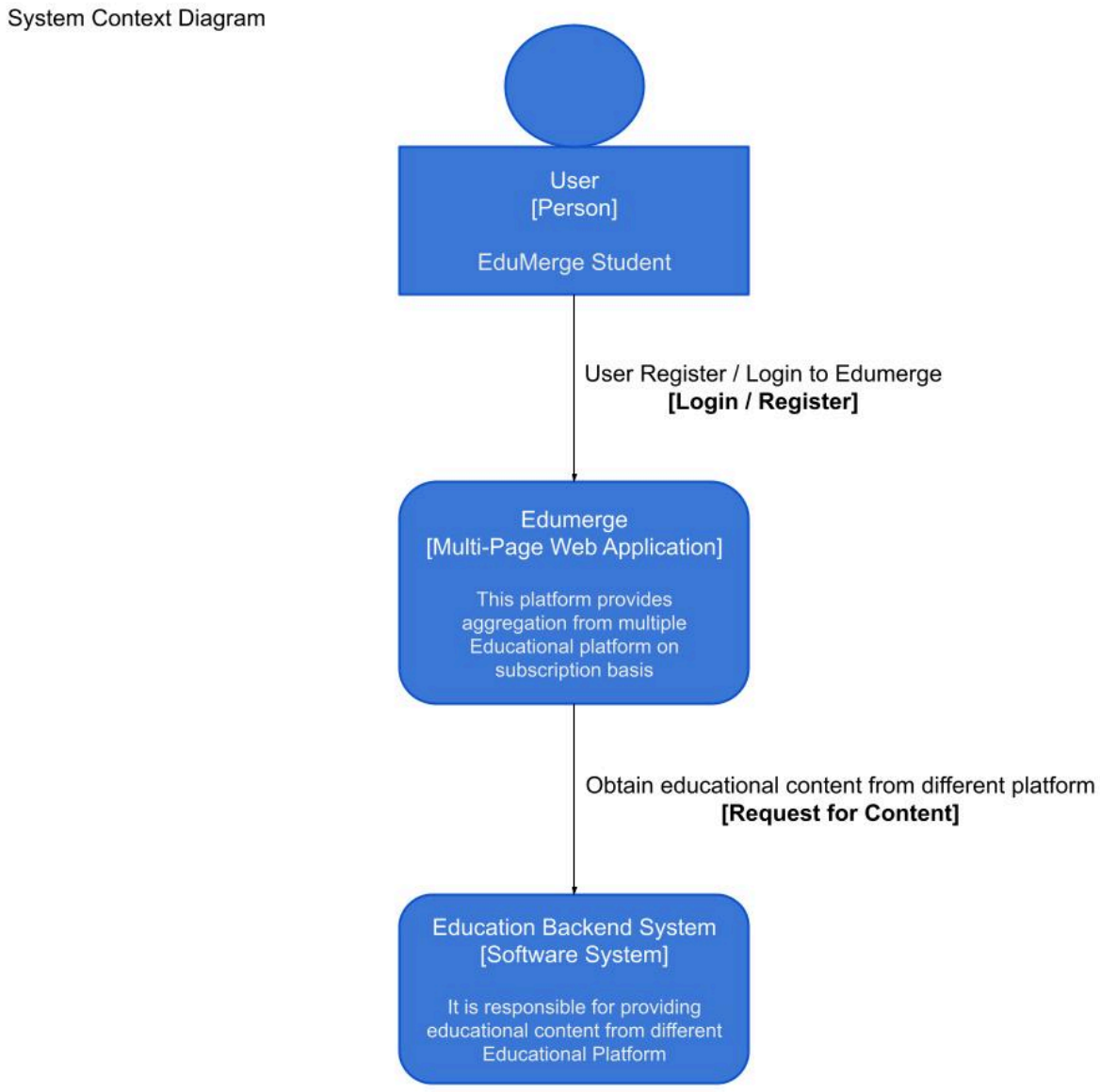
Rationale:

- **Improved Performance and User Experience:** A CDN caches static content (e.g., images, videos, JavaScript files) at geographically distributed edge servers. This reduces latency for users by serving content from the closest server location, resulting in faster loading times and a smoother user experience.
- **Scalability and Availability:** CDNs can handle surges in traffic effectively by distributing the load across multiple edge servers. This ensures platform availability even during peak usage periods.
- **Reduced Operational Costs:** Offloading static content delivery to a CDN frees up bandwidth on EduMerge's origin servers, potentially reducing operational costs associated with bandwidth usage.

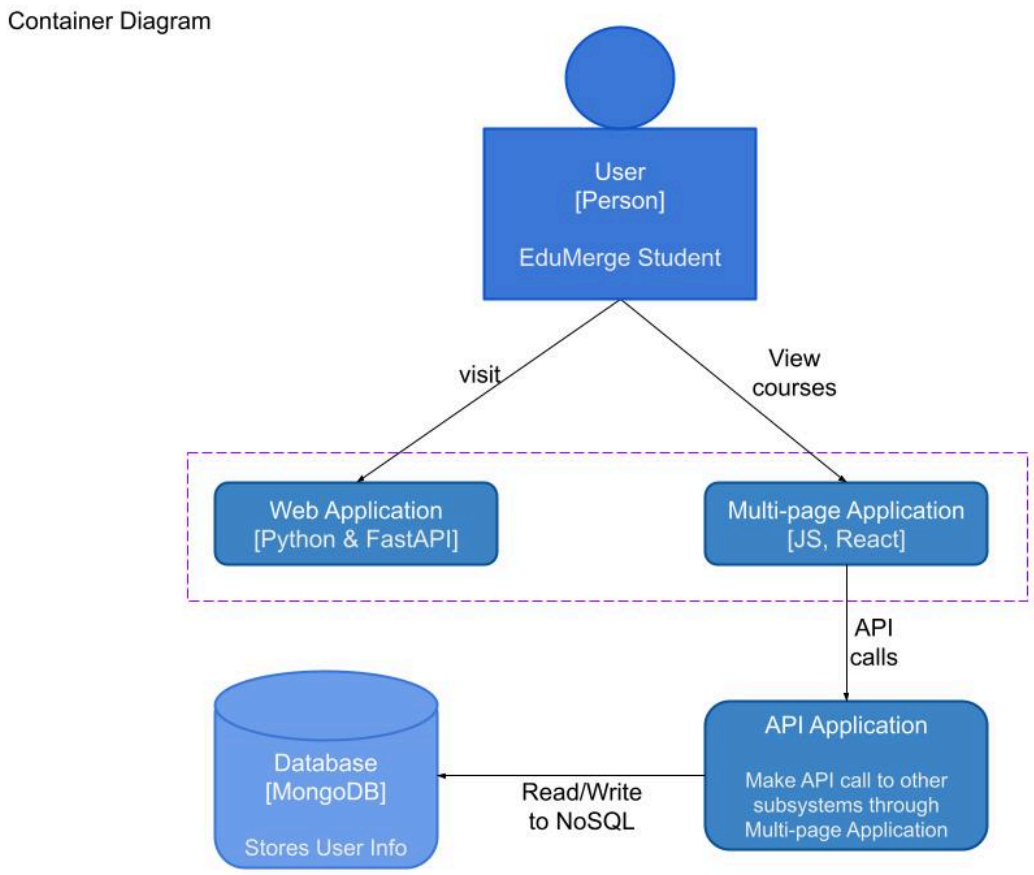
Architecture Diagram

C4 modeling technique is used for architecture

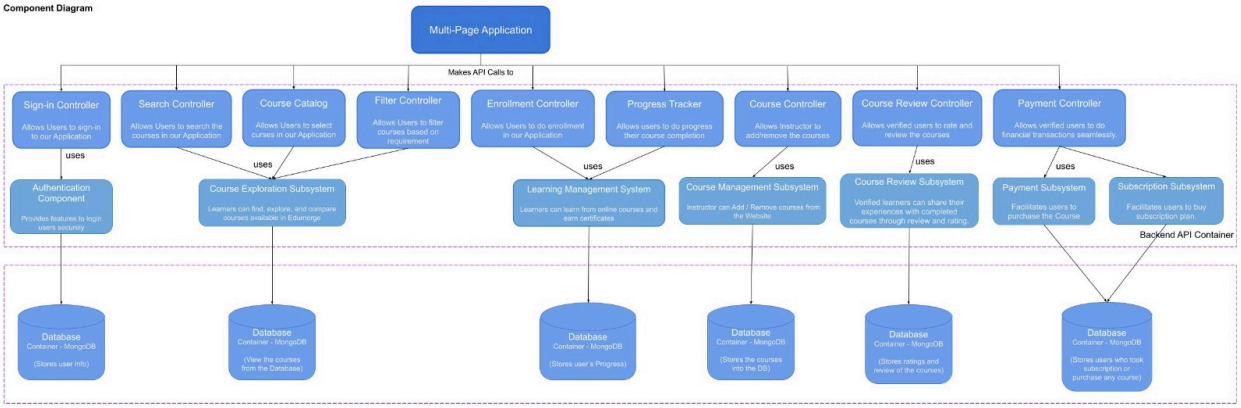
Context Diagram



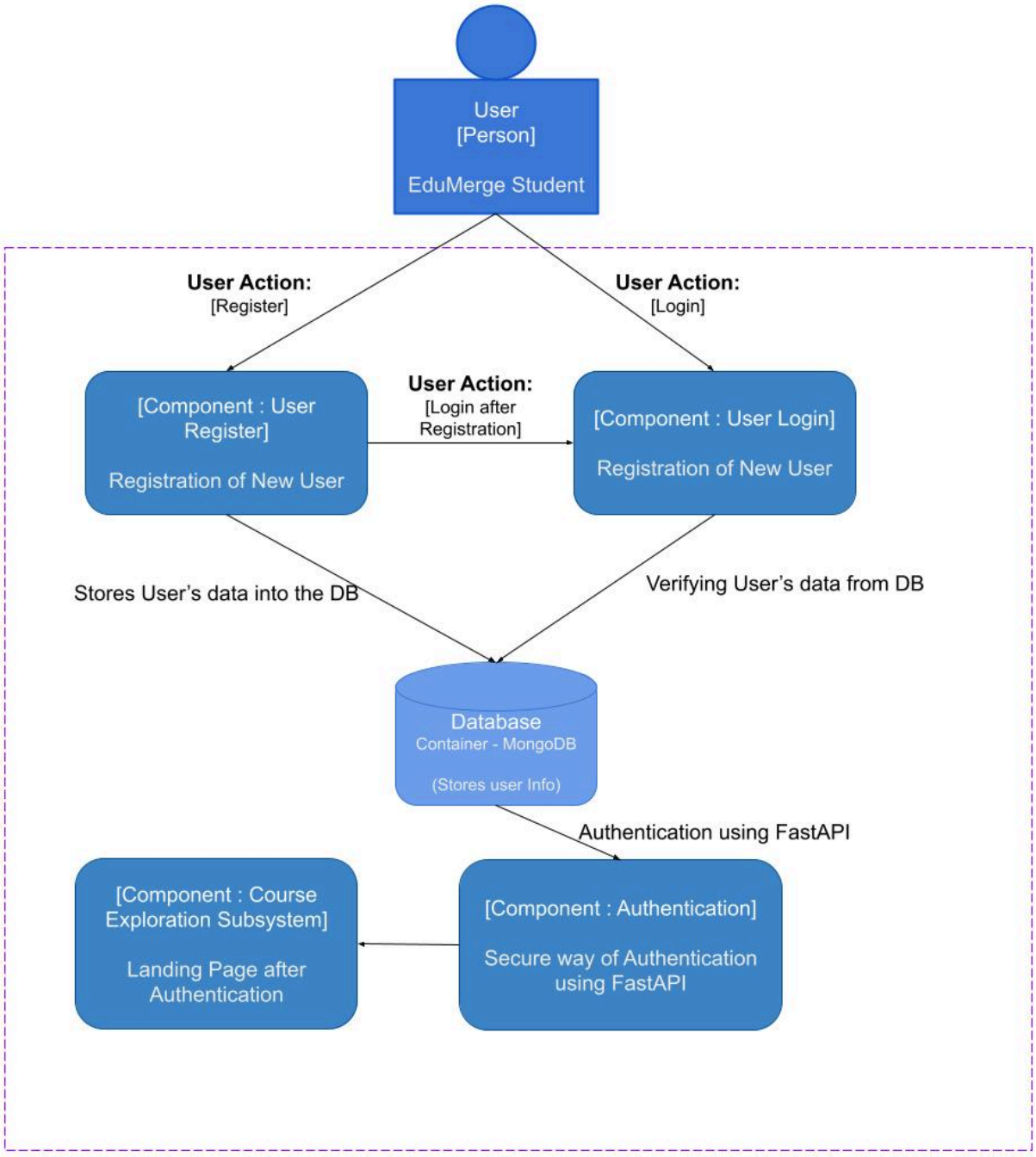
Container Diagram



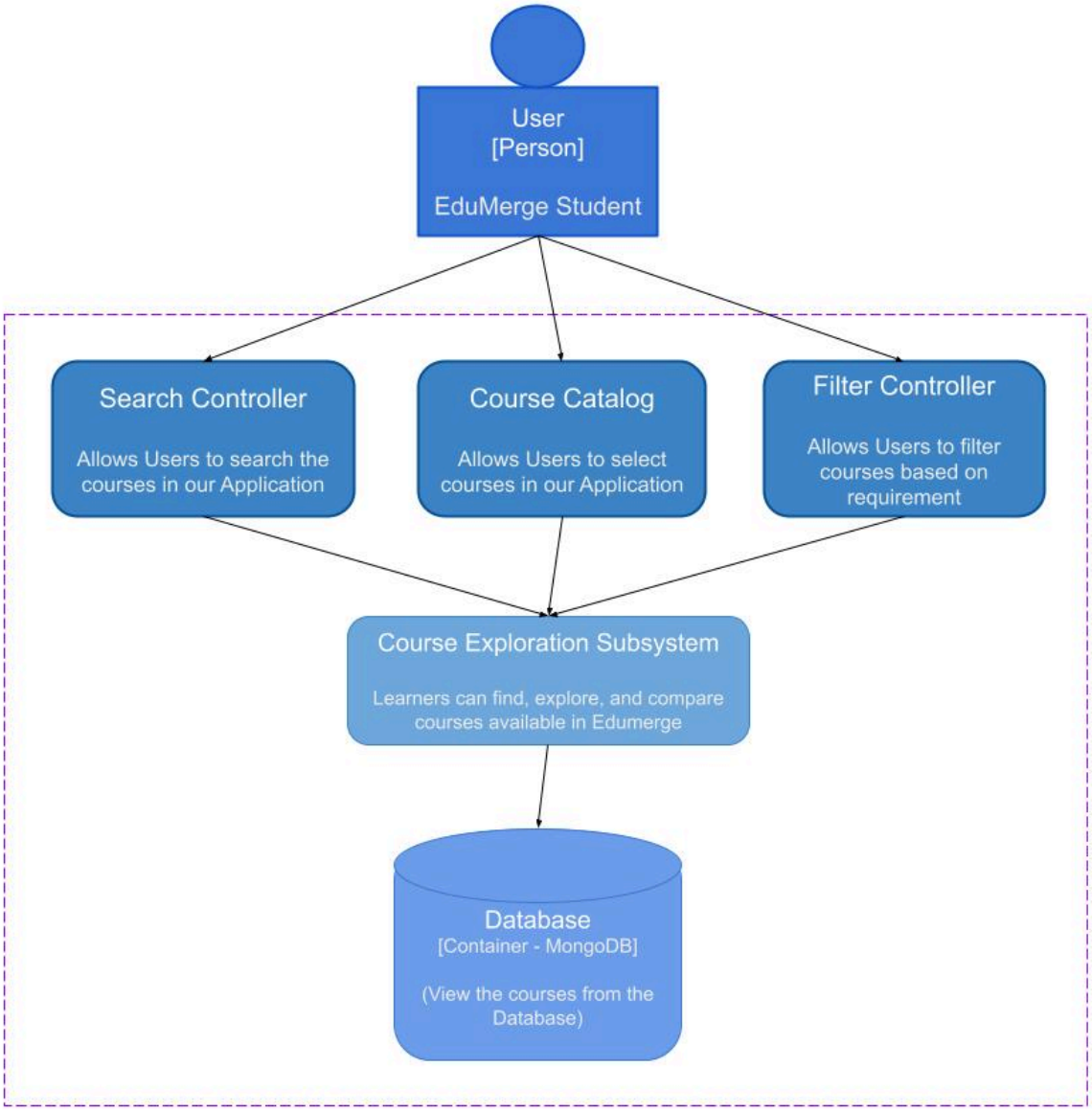
Component Diagram



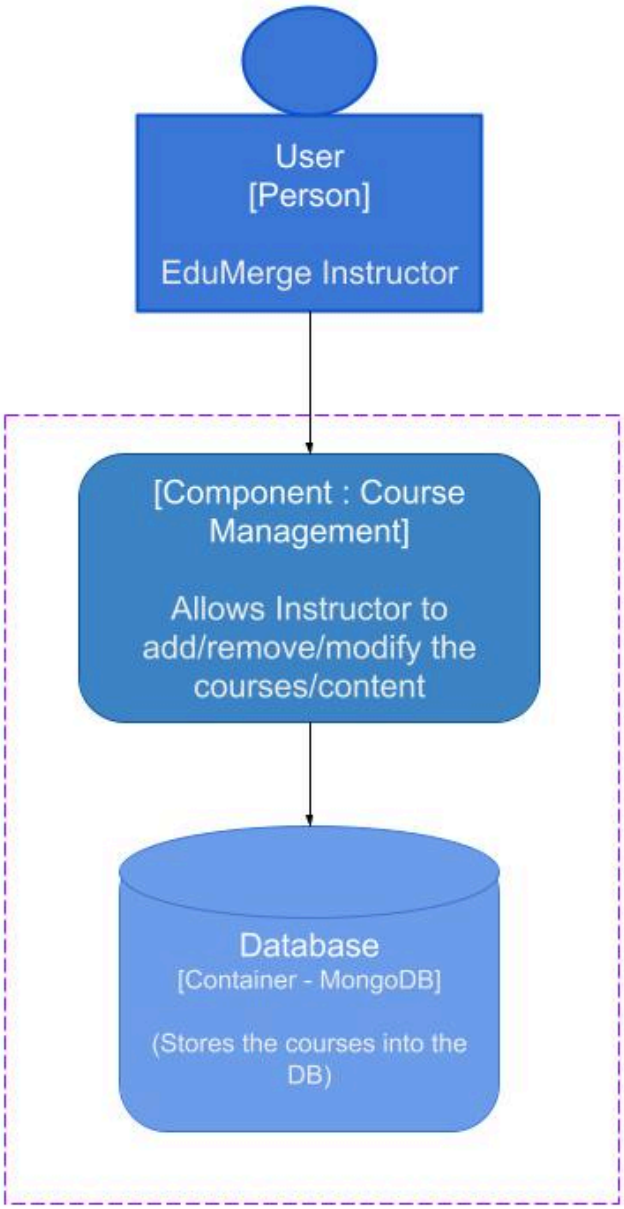
Component Diagram for User Management Subsystem



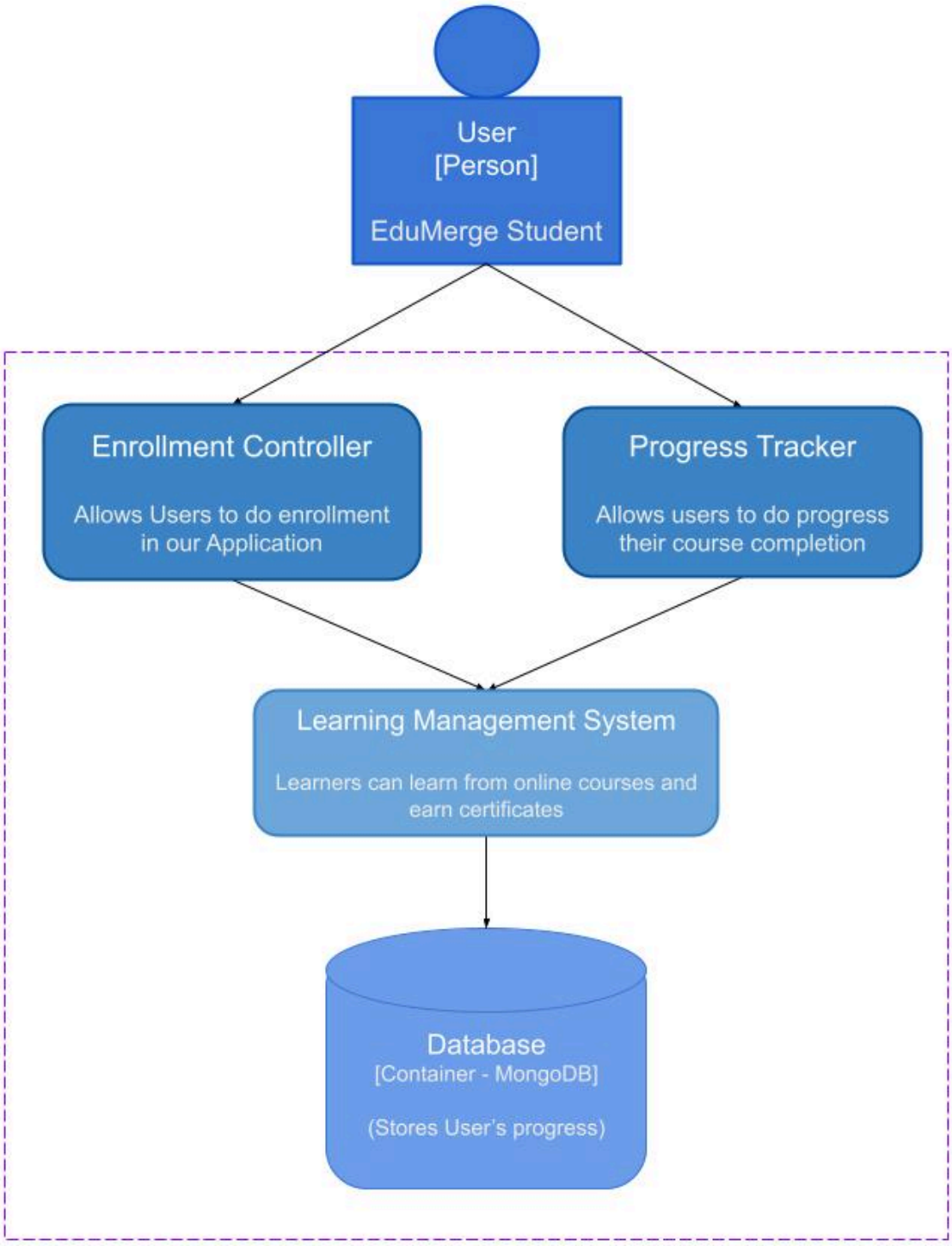
Component Diagram for Course Exploration Subsystem



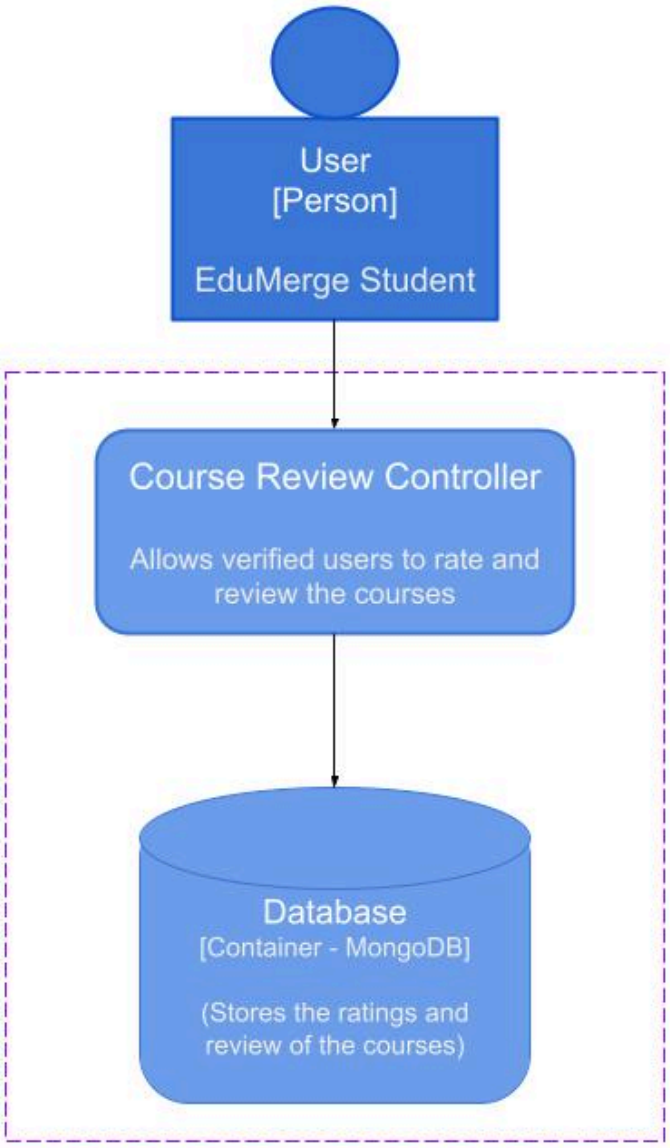
Component Diagram for Course Management Subsystem



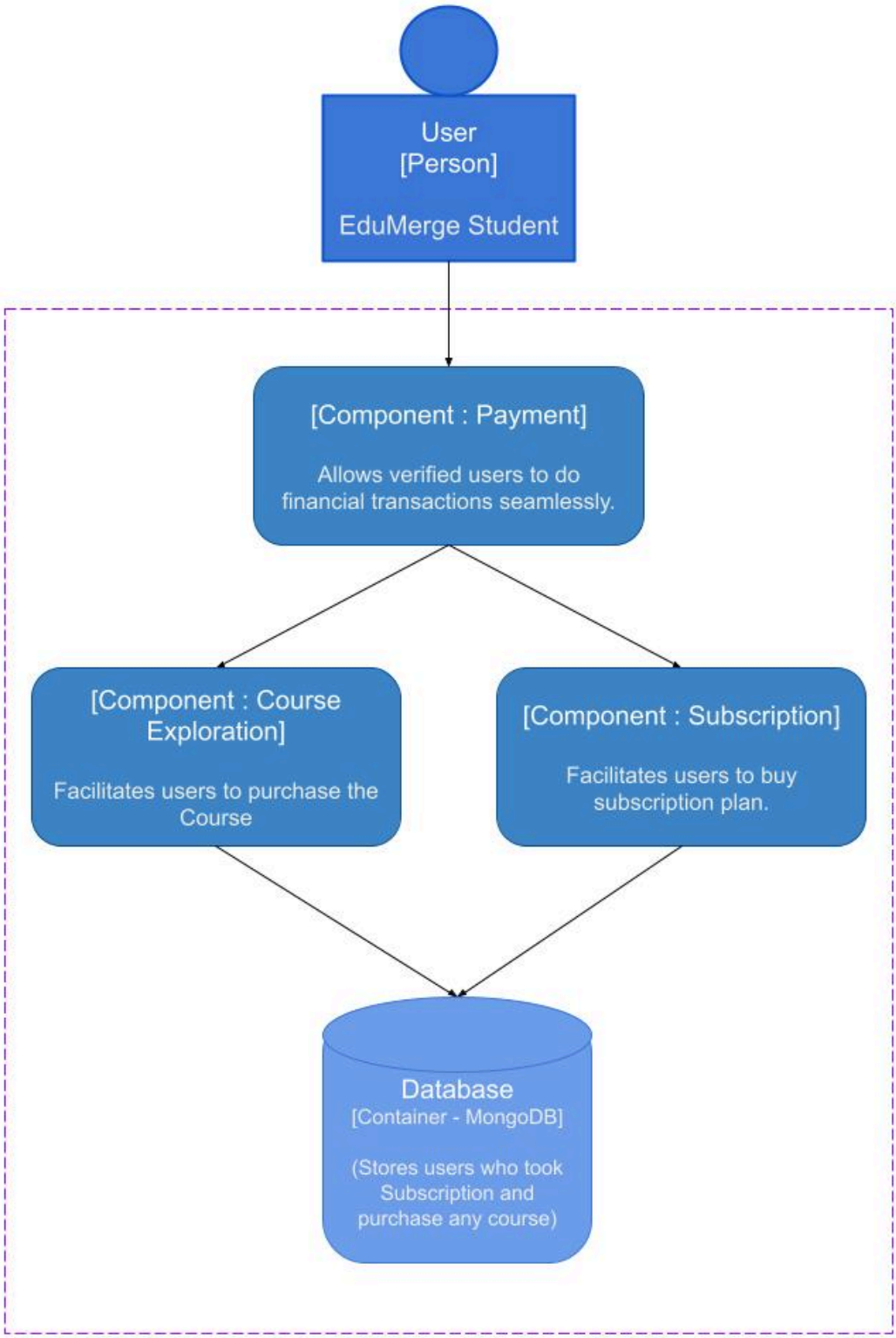
Component Diagram for Learning Management Subsystem



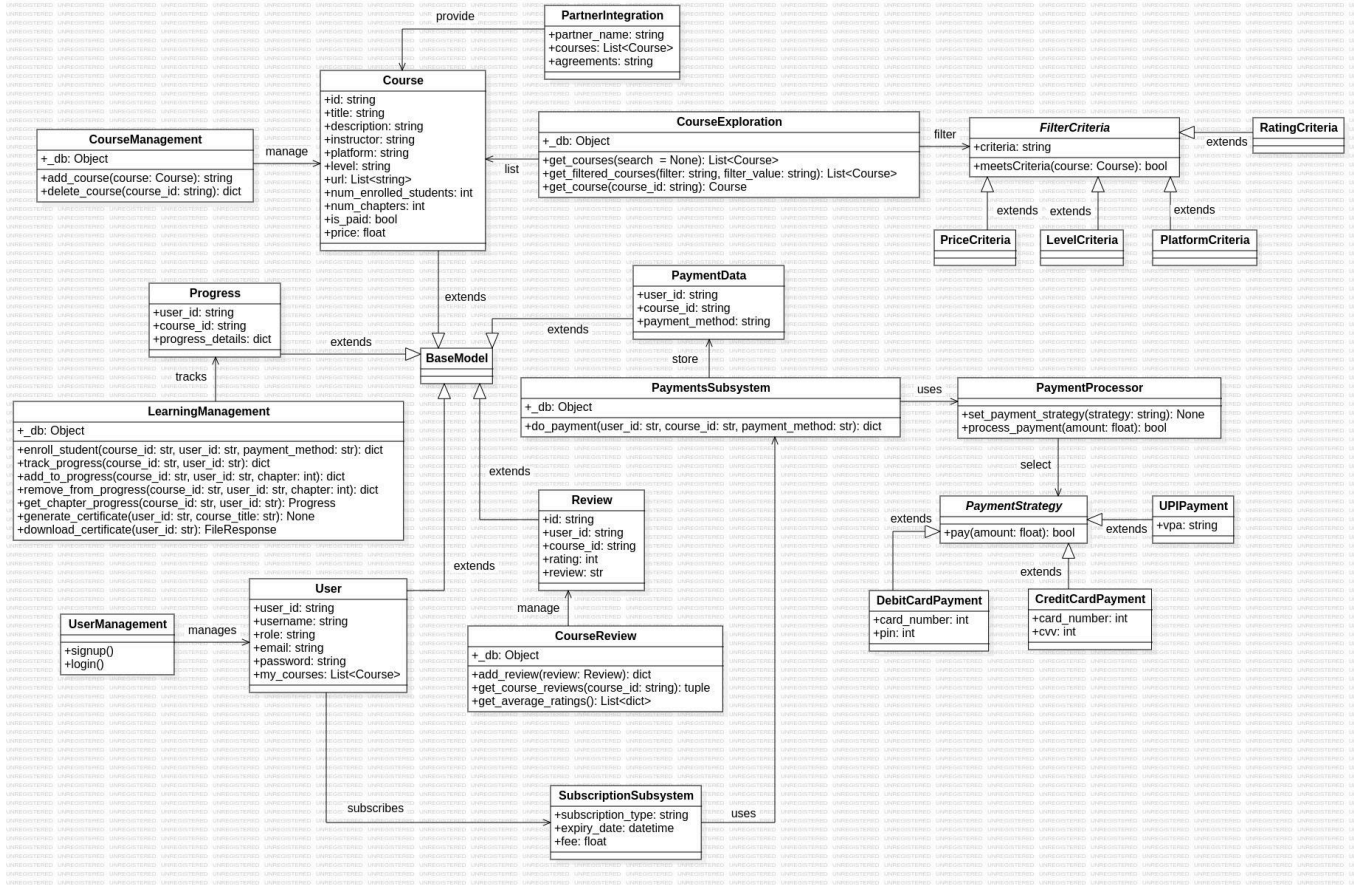
Component Diagram for Course Review Subsystem



Component Diagram for Payment Subsystem



Class Diagram



Task 3: Architectural Tactics and Patterns

3.1 Architectural Tactics

3.1.1 Architectural Tactics for High Availability in EduMerge

Fault Detection Tactics:

- **Ping/Echo & Heartbeat:** These tactics regularly check if components are alive and responsive. They can identify potential issues before they impact users by sending a ping or heartbeat message and expecting a response within a defined timeframe. This helps with early detection of failing components.

Fault Recovery Tactics:

- **Active Redundancy:** This tactic involves maintaining redundant components that can take over if a primary component fails. In EduMerge, this could involve having duplicate instances of critical services running concurrently, ensuring continuous operation if one instance encounters an issue.
- **Passive Redundancy:** Here, a backup component exists but is not actively processing requests until the primary component fails. In EduMerge, this could involve having a secondary database server ready to take over if the main database malfunctions.

Fault Prevention Tactics:

- **Process Monitor:** This tactic involves monitoring the health of processes and automatically restarting failed ones. In EduMerge, a process monitor could oversee critical services and ensure they are always running to avoid disruptions.

These architectural tactics can address non-functional requirements related to availability by:

- **Minimizing Downtime:** By implementing techniques for fault detection, recovery, and prevention, EduMerge aims to minimize the duration of outages or service disruptions.
- **Improving Fault Tolerance:** Redundancy mechanisms and process monitoring help EduMerge tolerate failures in certain components without the entire platform becoming unavailable.
- **Enhancing System Resilience:** The combination of tactics strengthens EduMerge's ability to recover from failures and maintain operational continuity.

3.1.2 Architectural Tactics for Performance in EduMerge

Reduce Resource Demand: This tactic focuses on optimizing the resource consumption of the EduMerge platform. This can involve techniques like:

- Using efficient algorithms and data structures to minimize processing power required for specific tasks.
- Minimizing unnecessary data transfers between components.
- Simplifying the user interface to reduce the load on client devices.

Increase Resource Availability: This tactic addresses performance by scaling up the resources available to the platform. This could involve:

- Adding more processing power (CPUs) to handle increased loads.
- Increasing memory capacity to improve data access speed.
- Upgrading storage solutions for faster data retrieval.

Resource Scheduling: This tactic optimizes how resources are allocated to different tasks. An efficient scheduler can ensure critical tasks receive the resources they need for smooth operation, even during peak usage periods.

Caching: By caching frequently accessed data, EduMerge can reduce the need to retrieve data from the main database or external sources for repeat requests. This can significantly improve response times for users.

These architectural tactics address non-functional requirements related to performance by:

- **Improving Scalability:** The tactics like reducing resource demand and increasing resource availability enable EduMerge to handle higher user volumes and data demands without performance degradation.
- **Enhancing Response Times:** Techniques like caching and resource optimization can significantly reduce response times for users, leading to a more responsive and fluid user experience.
- **Optimizing Resource Utilization:** By minimizing resource consumption and employing efficient scheduling, EduMerge can operate more efficiently and avoid resource bottlenecks.

3.1.3 Architectural Tactics for providing Security in EduMerge

- **Authentication:** This tactic ensures that only authorized users can access the EduMerge platform and specific resources within the platform. EduMerge can implement mechanisms like username/password login or token-based authentication to verify user identities.

- **Authorization:** This tactic controls what actions authorized users can perform within the platform. EduMerge can implement role-based access control (RBAC) to grant different permissions (e.g., view content, create courses) based on user roles (e.g., learner, instructor, administrator).

- **Data Encryption:** This tactic protects sensitive data (user information, course content) at rest and in transit by encrypting it with strong algorithms. Encryption scrambles the data so that even if intercepted, it remains unreadable without the decryption key.

- **Input Validation:** This tactic ensures that user input conforms to expected data types and formats. This helps prevent malicious users from injecting code or scripts (SQL injection, cross-site scripting) that could compromise platform security.

- **Limit Access:** This tactic restricts access to specific resources, functionalities, or even entire sections of the platform based on user roles or other defined criteria. For example, learner accounts might be limited from accessing administrative tools, or certain course materials could be restricted to paying subscribers.

- **Regular Security Updates:** This tactic involves keeping all EduMerge platform components (operating systems, software libraries) up-to-date with the latest security patches. These patches address known vulnerabilities that attackers might try to exploit.

These architectural tactics address non-functional requirements related to security by:

- **Confidentiality:** Encryption protects sensitive data from unauthorized access, ensuring only authorized users can view or modify it.
- **Integrity:** Input validation helps prevent unauthorized data modification, safeguarding the integrity of platform data.
- **Availability:** Security measures like authentication and authorization help prevent unauthorized access or disruptive actions that could impact platform availability.
- **Least Privilege:** The "Limit Access" tactic enforces the principle of least privilege, granting users only the minimum permissions necessary for their role, reducing the attack surface and potential damage from compromised accounts.

3.1.4 Architectural Tactics for providing Modifiability in EduMerge

- **Generalize Modules/Modularity:** This tactic breaks down the EduMerge platform into smaller, independent modules with well-defined interfaces. Each module encapsulates specific functionalities and interacts with other modules through standardized interfaces. This promotes loose coupling, where changes within a module have minimal impact on other modules. For example, a separate "User Management" module could handle user registration, login, and profile management tasks, with a clear interface for other modules (e.g., course management) to interact and retrieve user information.

- **Localize Changes:** This tactic groups related functionalities together and modifies them in a localized manner. This reduces the risk of unintended side effects in other parts of the system when making changes. In EduMerge, this could involve isolating course management functionalities within a specific module as mentioned previously.

- **Anticipate Expected Changes:** This tactic involves considering potential future modifications during the initial design phase. This can involve designing components that are flexible and adaptable to accommodate future needs. For EduMerge, this could involve designing the content management system to support different media types (text, video, audio) beyond basic text content, anticipating the potential addition of new content formats in the future.

- **Limit Possible Options:** This tactic aims to constrain the number of choices or configurations within the system. This can simplify future modifications by reducing the number of variations to consider when implementing changes. In EduMerge, this could

involve pre-defining a set of course content structures (e.g., lecture, video lesson, quiz) instead of allowing free-form course design. While this might limit some flexibility, it can make adding new features like progress tracking or automated grading easier to implement consistently across all courses.

These architectural tactics address non-functional requirements related to modifiability by:

- **Reduced Change Impact:** Generalize Modules/Modularity and Localizing Changes minimize the risk of unintended consequences when modifying the EduMerge platform. This makes it easier to introduce new features or update functionalities without causing disruptions in other parts of the system.
- **Improved Maintainability:** By anticipating expected changes, designing for flexibility (modularity), and grouping related functionalities, EduMerge can adapt to future requirements more easily. This reduces maintenance complexity and development effort in the long run.
- **Faster Time to Market:** When modifications are easier to implement due to these tactics, EduMerge can introduce new features or updates faster, improving its overall agility and responsiveness to evolving user needs.

3.1.5 Architectural Tactics for Usability in EduMerge

- **Design Time:** This tactic separates the user interface (UI) from the rest of the system. This allows for independent development and modification of the UI without affecting the core functionalities. This separation promotes a user-centric approach where the UI can be optimized for usability based on user feedback and testing, without requiring changes to the underlying system logic.

- **Support User Initiative:** This tactic empowers users to interact with the EduMerge platform in a way that aligns with their goals and preferences. This can involve features like:
    - Search functionality to easily find relevant courses, content, or users.
    - Customizable dashboards to organize learning materials and track progress.
    - Multiple course enrollment options to cater to different learning styles and paces.

These architectural tactics address non-functional requirements related to usability by:

- **Learnability:** Separating the UI from the rest of the system allows for a more intuitive and user-friendly interface design. Users can learn how to use the platform quickly by focusing on the clear visual cues and functionalities presented in the UI.
- **Efficiency:** Search functionality and customizable dashboards empower users to find information and complete tasks faster. Users can personalize their learning environment for optimal efficiency.
- **Satisfaction:** By addressing these usability factors, EduMerge can create a more user-friendly and enjoyable learning experience, increasing user satisfaction and platform adoption.

3.2 Implementation Patterns

Design Patterns used:

3.2.1 Adapter Pattern for Partner Platform Course Integration

Role: The adapter pattern allows EduMerge to integrate with various partner platforms that offer courses. It acts as a wrapper around the partner platform's API, translating their specific data format and communication protocols into a format that EduMerge understands.

Rationale: Partner platforms might have different ways of representing courses, user information, and enrollment data. The adapter pattern isolates platform-specific details and provides a consistent interface for EduMerge to interact with any partner platform seamlessly. This promotes loose coupling and simplifies the integration process for new partners.

3.2.2 Factory Pattern for Course Creation

Role: The factory pattern abstracts the process of creating different course types. EduMerge might offer various course formats (e.g., video lectures, interactive modules, self-paced learning). The factory pattern hides the logic for creating these different course types behind a single interface.

Rationale: The factory pattern centralizes the logic for course creation, making it easier to manage and maintain. It allows for adding new course types in the future without modifying existing code that interacts with the factory. This promotes code reusability and simplifies course management for administrators.

3.2.3 Factory Pattern for User Creation

Role: Similar to the course creation scenario, a user factory can be implemented. This factory would have different creator methods for each user role (learner, instructor, admin, partner). Each creator method would be responsible for creating a user object with the appropriate properties and potentially assigning them specific permissions based on their role.

Rationale: The factory pattern centralizes user creation logic, making it easier to manage and maintain. You can add new user roles in the future by simply adding new creator methods to the factory without modifying existing code. This promotes loose coupling and simplifies the user creation process.

3.2.4 Strategy Pattern for Payment

Role: The strategy pattern allows EduMerge to support multiple payment processing methods (e.g., credit cards, debit cards, UPI). The strategy pattern uses different concrete strategies (implementations) for each payment gateway, allowing the system to switch between them based on user preference or payment options available in different regions.

Rationale: The strategy pattern provides flexibility in handling payments. EduMerge can integrate with new payment gateways easily by adding new concrete strategies without modifying the core payment processing logic. This allows for future expansion and caters to a wider user base with diverse payment preferences.

3.2.5 Criteria Pattern for Filtering the courses

Role: The criteria pattern allows users to filter courses based on various criteria like subject, instructor, price, or rating. The criteria pattern builds flexible search queries by combining different filtering criteria objects.

Rationale: The criteria pattern enables powerful and expressive course search functionalities. Users can filter courses based on specific combinations of attributes, leading to a more personalized and efficient learning experience. This pattern also simplifies the implementation and maintenance of search functionalities within EduMerge.

3.2.6 Observer Pattern for Notifications

Role: The observer pattern establishes a communication channel for notifications in EduMerge. It defines two actors: subjects (services that trigger notifications) and observers (user roles like learners and instructors). Observers can subscribe to specific subjects to receive updates relevant to their interests.

Rationale: This pattern promotes loose coupling and simplifies development. Observers don't need to know how subjects work, and subjects don't need to manage individual user notification preferences. It also offers flexibility for users to control their notification streams and enhances scalability as EduMerge adds new features and notification types.

Task 4: Prototype Implementation and Analysis

4.1 Prototype Development

Repository Link: https://github.com/kbbhatt04/se_project_3

4.2 Architecture Analysis

Microservices vs. Monolithic Architecture in EduMerge

Monolithic:

/courses

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/courses	9037	0	530	790	1200	524.27	11	2368	353	170.4	0
	Aggregated	9037	0	530	790	1200	524.27	11	2368	353	170.4	0

/enroll

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/enroll?course_id=1&user_id=3	8918	0	510	740	930	477.97	12	3549	35	191	0
	Aggregated	8918	0	510	740	930	477.97	12	3549	35	191	0

/add_rating

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/add_review	9503	0	480	780	1100	466.01	25	3309	39	185	0
	Aggregated	9503	0	480	780	1100	466.01	25	3309	39	185	0

Microservices:

/courses

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/courses	7259	0	600	553.02	16	3493	353	137.3
	Aggregated	7259	0	600	553.02	16	3493	353	137.3

/enroll

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
POST	/enroll_student?course_id=1&user_id=3&payment_method=credit	7309	0	630	634.28	9	3965	35	162.8
	Aggregated	7309	0	630	634.28	9	3965	35	162.8

/add_rating

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
POST	/add_review	5852	0	640	681.98	17	3527	39	110.9
	Aggregated	5852	0	640	681.98	17	3527	39	110.9

Trade-offs

Increased Development and Operational Complexity:

- **Development:** Microservices require to design, build, and maintain multiple independent services. This involves managing separate codebases, APIs, and potentially different programming languages or frameworks for each service. Compared to a monolithic codebase, this requires more coordination during development.
- **Deployment:** Deploying changes in a microservices architecture involves managing deployments for multiple services. This requires robust CI/CD pipelines to automate testing and deployment processes across all services. Additionally, communication and coordination between development teams working on different services become more crucial to ensure smooth deployments without introducing regressions.
- **Operational Management:** Monitoring and troubleshooting issues in a distributed system with multiple services can be more complex. We need to monitor the health and performance of individual services as well as how they interact with each other. Additionally, distributed tracing tools might be required to track requests across different services and pinpoint the root cause of problems.

Potential Performance Overhead:

- **Communication Overhead:** In a monolithic application, components can directly access data and functionalities within the same codebase. In microservices, communication between services happens through APIs, which can introduce some network overhead compared to in-memory function calls. This overhead can be mitigated through techniques like:
    - **API Gateway:** An API gateway can act as a single entry point for all API calls, reducing the number of network hops required for requests.
    - **Caching:** Caching frequently accessed data within services or at an API gateway level can reduce the need for repeated calls to the backend and improve response times.
    - **Message Queues:** For asynchronous communication between services, message queues can be used to decouple services and potentially improve overall performance.

Mitigating Deployment Complexity with CI/CD:

- **Continuous Integration (CI):** CI involves automating the process of building, testing, and integrating code changes from developers into a central repository. This helps identify and fix issues early in the development cycle.
- **Continuous Delivery (CD):** CD automates the deployment process, allowing for frequent and reliable deployments of changes to production environments. This can be configured to handle deployments for individual microservices or coordinated deployments across multiple services.

Conclusion:

While microservices introduce additional complexity, the benefits for EduMerge are significant. The ability to scale specific services, deploy features faster, and improve fault tolerance are crucial for EduMerge's success. By adopting proper development practices, utilizing CI/CD pipelines, and implementing performance optimization techniques, EduMerge can manage the complexity of a microservices architecture and reap the long-term benefits for its platform and users.

Lessons learnt from the project

- The importance of clear and comprehensive requirements gathering early in the process. This helps ensure the final system meets user needs and stakeholders' expectations.
- Identifying stakeholders and their views and viewpoints and clear documentation of design choices are crucial for project success.
- ADRs provide a valuable record of reasoning behind key architectural decisions.
- Understanding the trade-offs between different tactics and patterns helps make informed choices that address specific non-functional requirements.
- Prototyping allows early validation of design choices and helps identify potential issues before full-scale development.
- Analyzing alternative architectures provides valuable insights and helps choose the best approach for the specific system needs.

Contributions

Name	**Roll Number**	**Contribution**
Karan Bhatt	2022202003	-API's: ● /course ● /course/{course_id} ● /add_review ● /enroll_student ● /get_chapter_progress ● /download_certificate -Sub-Systems: ● Course Exploration ● Course Review ● Learning Management -Architecture Analysis
Madhusree Bera	2022202007	-APIs ● /register_service ● /register_service_health_monitor ● /register_service_load_balancer ● /deregister_service/{service_id} ● /get_service ● /balance_load ● /start_health_monitor -Sub-Systems: ● Service Registry ● Load Balancer ● Health Monitor ● Web Application (UI)

		-Features: ● Email Notification feature ● Health Monitor feature ● Logging feature
Vedashree Ranade	2022201073	-API's: ● /courses/filter ● /reviews/average_ratings ● /get_reviews/{course_id} ● /track_progress ● /add_to_progress ● /remove_from_progress -Sub-Systems: ● Course Exploration ● Course Review ● Learning Management -Monolithic Architecture Subsystem
Yash Maheshwari	2022201074	-API's: ● /payment ● /add_course ● /delete_course -Sub-Systems: ● Course Management ● Payments -Diagrams
Piyush Rana	2022202012	-API's: ● /signup ● /login ● /logout ● /validate ● /protected -Sub-Systems: ● Authentication ● Payments ● Web Application (UI) -Integration of the subsystems