ONLINE COURSE REGISTRATION SYSTEM

By TEAM : TD11

OVERVIEW

PROBLEM STATEMENT:

Description:

Many educational institutions struggle with outdated and inef cient course registration systems, leading to frustration among students, faculty, and administrative staff. These systems often lack user-friendly interfaces, causing confusion and errors during the registration process. Additionally, they may lack real-time updates, making it dif cult for students to access accurate course information and availability. Moreover, the absence of personalized course suggestions based on students' interests and academic requirements further exacerbates the problem.

Problem:

The current online course registration system suffers from several shortcomings:

Lack of user-friendliness: The interface is complex and unintuitive, leading to confusion and errors during the registration process.

Limited accessibility: The system may not be accessible from all devices, hindering students' ability to register for courses remotely.

Inaccurate information: The system may not provide real-time updates on course availability, leading to students enrolling in lled or canceled courses.

Poor scalability: As student populations grow, the system may struggle to handle increased traf c, resulting in slowdowns or crashes during peak registration periods.

Absence of personalized course suggestions: The system does not offer tailored course recommendations based on students' academic interests, goals, and requirements.

Objective:

The objective is to develop an ef cient and user-friendly online course registration system that addresses the aforementioned shortcomings. This system should:

- Provide a seamless and intuitive user interface for students, faculty, and administrative staff.
- Ensure accessibility across multiple devices and platforms to facilitate remote registration.
- Offer real-time updates on course availability and scheduling to prevent enrollment errors.
- Scale effectively to accommodate growing student populations and peak registration periods.
- Implement a personalized course suggestion feature that recommends relevant courses based on students' interests, academic history, and degree requirements.

This project aims to enhance the overall registration experience for all stakeholders by providing a comprehensive and ef cient course registration system with personalized course suggestions, thereby improving student satisfaction and academic outcomes within the educational institution.

PROPOSED SOLUTION:

The proposed solution involves the development and implementation of an integrated online course registration system that incorporates advanced features such as user-friendly interface design, real-time updates, scalability optimization, and personalized course recommendation functionality. This solution aims to streamline the course registration process for students, faculty, and administrative sta while addressing the challenges identi ed in the problem statement.

How it Addresses the Problem:

User-Friendly Interface Design:

The new system will feature a redesigned user interface with intuitive navigation, clear instructions, and visually appealing design elements. This addresses the problem of user-friendliness by making it easier for students, faculty, and sta to navigate the registration process without encountering confusion or errors.

Real-Time Updates and Automation:

By integrating real-time data synchronization mechanisms with the institution's course management systems, the new system ensures that course availability information is accurate and up-to-date. Automated noti cations will alert students and faculty about changes in course status, enrollment deadlines, and academic calendar updates, mitigating the problem of inaccurate information and reducing manual administrative tasks.

Enhanced Accessibility:

The system will utilize responsive web design techniques and cloud-based infrastructure to ensure accessibility from various devices, including smartphones, tablets, and desktop computers. This addresses accessibility issues, enabling students to register for courses remotely with ease, even during peak registration periods.

Scalability Planning and Optimization:

Through capacity planning exercises and infrastructure scaling strategies, the new system will be optimized to handle increased tra c during peak registration periods without compromising performance. This ensures system scalability and reliability, addressing concerns about system slowdowns or crashes during high-demand periods.

Personalized Course Recommendation Engine:

A machine learning-based recommendation engine will analyze students' academic histories, interests, and career aspirations to generate personalized course recommendations. These recommendations will be validated by academic advisors and faculty members, providing students with tailored course suggestions that align with their goals and interests.

Benefits:

Improved User Experience:

The user-friendly interface and streamlined registration process enhance the overall user experience for students, faculty, and administrative sta , reducing frustration and errors.

Accurate and Up-to-Date Information:

Real-time updates and automated noti cations ensure that students have access to accurate course availability information, reducing the likelihood of enrolling in lled or canceled courses.

Enhanced Accessibility and Flexibility:

Responsive design and cloud-based infrastructure make the system accessible from various devices, enabling students to register for courses remotely with ease.

Scalability and Reliability:

Scalability optimization measures ensure that the system can handle increased trac during peak registration periods without experiencing performance issues, enhancing system reliability.

Personalized Learning Experience:

The personalized course recommendation engine provides students with tailored course suggestions that align with their academic interests and career goals, promoting a more personalized learning experience and improving student satisfaction and retention rates.

BUSINESS ARCHITECTURE:

Overview

- The business architecture diagram outlines the structure and components involved in the Elective Course Registration System. It showcases the interactions between different stakeholders, the system's core functionalities, and the underlying infrastructure.

Stakeholders

- *Admin:* Manages student registrations, course offerings, instructor assignments, registration activity monitoring, and report generation.
- *Student:* Views available courses, attends course information quizzes, receives course recommendations, registers for courses, and reviews/rates courses.
- *Instructor:* Views assigned courses, student enrollment lists, provides course recommendations, updates course information, and communicates with enrolled students.

KeyComponents

- *Login/Logout:* Central authentication mechanism for Admin, Student, and Instructor access.
- *Cache Management:* Utilizes Azure Redis Cache to enhance performance by providing low-latency access to frequently accessed data.
- *Web Hosting*: The web application is hosted on Azure cloud using Azure App Services.

Activities

- *Admin Activities:* Include managing student registrations, course offerings, instructor assignments, monitoring registration activity, and generating reports.
- *Student Activities*: Encompass viewing available courses, attending course information quizzes, receiving recommendations, registering for courses, and reviewing/rating courses.
- *Instructor Activities:* Cover viewing assigned courses, student enrollment lists, providing recommendations, updating course information, and communicating with students.
- Backend Overview

- *Database:* Managed by administrators with direct access for control and monitoring. Handles CRUD operations for courses, registrations, reviews, communications, and recommendations.
- *Web Server:* Interfaces for students and instructors, routing their requests to the database and rendering information.
- *Azure Cloud:* Provides scalable hosting, caching, security, and monitoring services.

PROPOSED SYSTEM:

1.ConceptualDesign

- Outlines a new software solution tailored to meet speci c organizational needs or challenges.
- Represents an evolution of the current system, incorporating enhancements or new features.

2.RoadmapforDevelopmentTeams

- Provides a detailed blueprint for key features, functionalities, and architecture.
- Guides development teams through the software creation process.

3.RequirementsAnalysis

- Identi es user needs and business objectives.
- Ensures the proposed system aligns with organizational goals.

4.SystemDesign

- De nes the software's structure and components.
- Establishes the foundational architecture for development.

5.Prototyping

- Allows stakeholders to visualize the system and provide feedback.
- Facilitates early identi cation of potential issues and user requirements.

6.FeasibilityStudies

- Assesses the system's technical, economic, and operational viability.
- Ensures the proposed solution can be implemented within existing constraints.

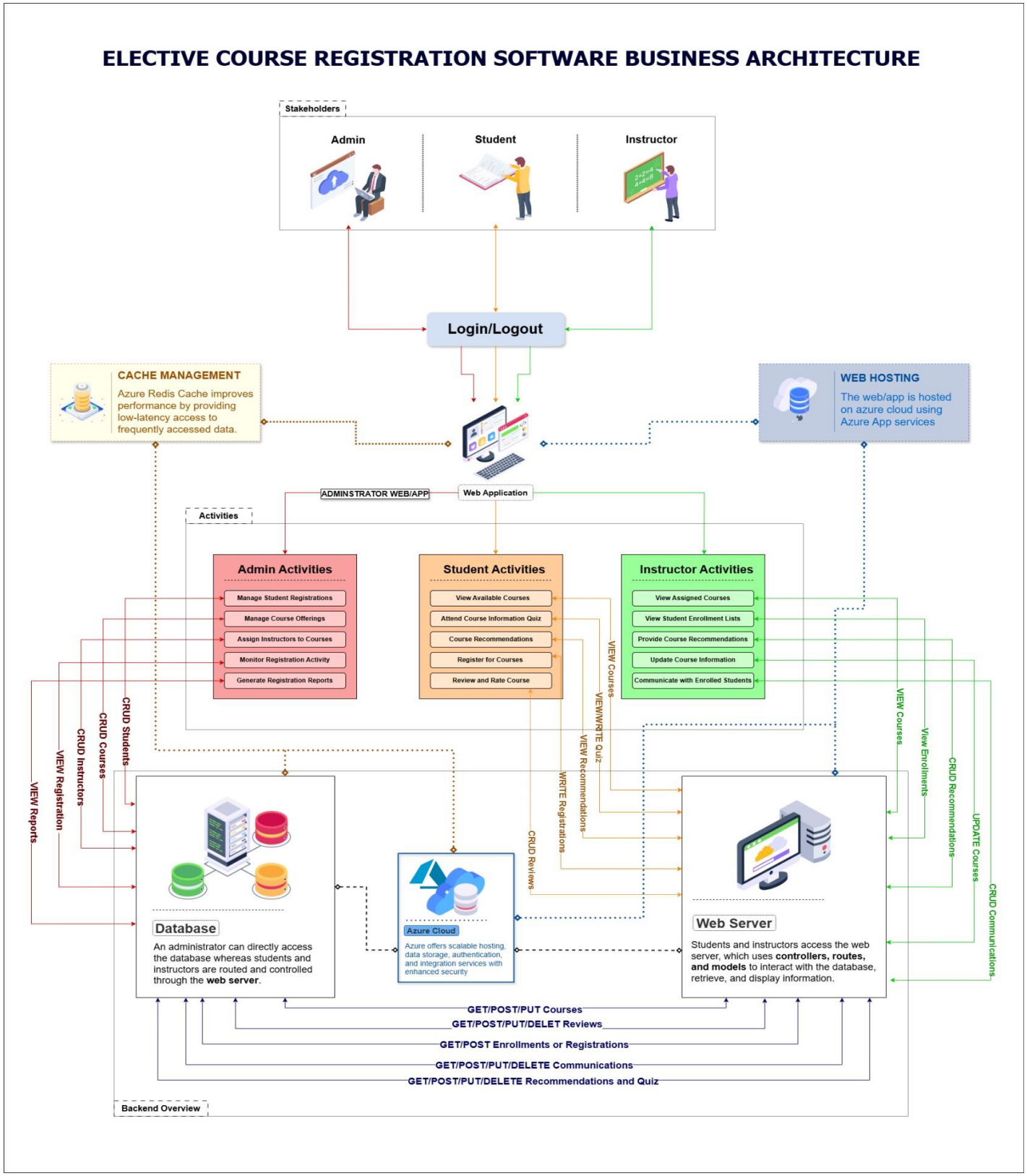
7.RiskManagement

- Identi es and mitigates potential obstacles.
- Develops strategies to address and reduce risks throughout the project lifecycle.

8.AlignmentwithOrganizationalGoals

- Ensures the proposed system supports and enhances business objectives.
- Streamlines development processes for ef ciency and effectiveness.

BUSINESS ARCHITECTURE DIAGRAM:



USER STORIES AND AGILE SCRUM :

Product Backlog:

Sprint 1:

Duration: 2 weeks

Sprint Goal: Implement basic course registration and view registration history functionality.

UserStories:

- Student Course Registration
- View Registration History

Tasks:

- Design database schema for course registrations and history.
- Create UI forms for course registration submission and registration history display.
- Implement backend logic for course registration submission and storage.
- Implement backend logic for fetching and displaying registration history.

Sprint Review: Demonstrate functionality for course registration submission and registration history display.

Sprint 2:

Duration: 2 weeks

Sprint Goal: Implement course approval by faculty and emergency course changes handling for administrators.

UserStories:

- Faculty Course Approval
- Administrator Emergency Course Changes

Tasks:

- Design database schema for pending course approvals and emergency course changes.
- Create UI for faculty to review pending course registrations.
- Implement backend logic for faculty to approve, reject, or request additional information for course registrations.
- Create UI for administrators to manage emergency course changes.
- Implement backend logic for administrators to handle emergency course changes. ● Set up noti cations for administrators on new emergency course changes.

Sprint Review: Demonstrate functionality for course approval by faculty and emergency course handling by administrators.

Sprint 3:
Duration: 2 weeks
Sprint Goal: Implement course management and elective selection quiz for students.

UserStories:

- Course Management
- Elective Selection Quiz

Tasks:

- Design database schema for courses and elective quiz results.
- Create UI for administrators to view and manage courses.
- Implement backend logic for administrators to update course information.
- Create UI for students to access and take the elective selection quiz.
- Implement backend logic for processing quiz results and suggesting electives.
- Log changes made to course information for audit purposes.
- Set up permissions for who can view or modify course information.

Sprint Review: Demonstrate functionality for course management and elective selection quiz.

Sprint 4:
Duration: 2 weeks
Sprint Goal: Enhance quiz functionality and integrate it with the course registration system.

UserStories:

- Enhanced Quiz Functionality for Elective Selection

Tasks:

- Improve the elective quiz algorithm to provide better elective suggestions.
- Create UI enhancements for the elective selection quiz to improve usability.
- Implement backend logic to integrate quiz results with the course registration system.
- Provide a summary of quiz results and elective suggestions to students.
- Set up notifications to remind students to complete the elective selection quiz.

Sprint Review: Demonstrate enhanced quiz functionality and integration with the course registration system.

Sprint Breakdown and Agile Alignment:

This breakdown aligns with Agile principles by focusing on iterative development, delivering working functionality in short sprints, and incorporating feedback at each sprint review. Adjustments can be made based on team capacity and project priorities.

USER STORIES:

As a Student:

1. *Browsing and Searching Courses:*
   - As a student, I want to browse a list of available elective courses so that I can explore my options for the upcoming semester.
   - As a student, I want to search for courses by keywords, course codes, or instructor names so that I can quickly find specific electives.

2. *Reading Course Descriptions:*
   - As a student, I want to read short descriptions of each elective course so that I can understand the course objectives and content.

3. Favoriting and Comparing Courses:
   - As a student, I want to mark courses as favorites so that I can easily refer back to them later.
   - As a student, I want to compare up to three courses side-by-side so that I can make an informed decision on which elective to choose.

4. Checking Schedule Compatibility:
   - As a student, I want to see how my selected courses fit into my existing schedule so that I can avoid time conflicts.

5. Enrolling in Courses:
   - As a student, I want to easily enroll in my selected courses through the application so that I can secure my spot without additional steps.

As a College Administrator

1. *Managing Course Listings:*
   - As a college administrator, I want to manage the list of available elective courses so that I can ensure the offerings are up-to-date and accurate.

2. *Monitoring Enrollment Trends:*
   - As a college administrator, I want to monitor enrollment trends and popular courses so that I can make data-driven decisions about future course offerings.

3. *Handling Prerequisites and Conflicts:*
   - As a college administrator, I want to ensure that students are informed about prerequisites and potential scheduling conflicts so that they can make suitable choices.

4. *Facilitating Registration Integration*:
   - As a college administrator, I want to integrate the application with the college's registration system so that the enrollment process is seamless for students.

5. Collecting Student Feedback:
   - As a college administrator, I want to collect and analyze student feedback on c

As a Faculty Member

1. *Creating and Updating Course Information:*
   - As a faculty member, I want to create and update the information for the courses I teach so that students have accurate and comprehensive details.

2. *Monitoring Enrollment Numbers:*
   - As a faculty member, I want to monitor the enrollment numbers for my courses so that I can anticipate class size and prepare accordingly.

3. *Receiving Student Feedback:*
   - As a faculty member, I want to read reviews and feedback from students about my courses so that I can understand their experiences and make improvements.

4. *Communicating with Students:*
- As a faculty member, I want to communicate with students who have enrolled in my courses so that I can provide updates and important information prior to the semester starting.

5. *Reviewing Enrollment Prerequisites:*
- As a faculty member, I want to review the prerequisites for my courses to ensure that enrolled students meet the necessary requirements.

By addressing these user stories, the Elective Course Selection Application will cater to the needs of students, administrators, and faculty members, providing a comprehensive and efficient solution for elective course management.

ACCEPTANCE CRITERIA:

Acceptance Criteria for Elective Course Selection Application

As a Student

1. Browsing and Searching Courses
- Browse Courses
- User can view a list of all available elective courses.
- The list includes course titles, course codes, and instructors.
- Search Courses
- User can search for courses by entering keywords, course codes, or instructor names.
- Search results are displayed dynamically as the user types.
- The search function is responsive and displays relevant courses within 2 seconds.

2. Reading Course Descriptions
- User can click on a course to view a detailed description.
- The description includes course objectives, content summary, credit hours, and prerequisites.
- Descriptions are clear, concise, and provide all necessary information within 150 words.

3. Favoriting and Comparing Courses
- Favoriting Courses
- User can mark courses as favorites with a single click.
- Favorited courses are saved in a personalized list accessible from the user's profile.
- Comparing Courses
- User can select up to three courses to compare.

- The comparison view displays key details side-by-side, including course title, code, instructor, description, and schedule.

4. Checking Schedule Compatibility
   - User can view their current schedule.
   - When selecting courses, the application checks for and highlights any scheduling conflicts.
   - The schedule view updates dynamically to show selected courses alongside existing commitments.

5. Enrolling in Courses
   - User can enroll in selected courses directly through the application.
   - Enrollment is confirmed within 5 seconds, and the course is added to the user's schedule.
   - User receives a confirmation email upon successful enrollment.

As a College Administrator

1. Managing Course Listings
   - Administrator can add, update, or remove courses from the listing.
   - Changes to the course listings are reflected in real-time.
   - The interface allows bulk uploads of course information via CSV files.

2. Monitoring Enrollment Trends
   - Administrator has access to a dashboard showing current enrollment numbers for all courses.
   - The dashboard includes visualizations (e.g., graphs, charts) of enrollment trends. - Data can be filtered by department, semester, and other relevant criteria.

3. Handling Prerequisites and Conflicts
   - The system checks prerequisites when students attempt to enroll.
   - If prerequisites are not met, the student is informed with a clear message.
   - Scheduling conflicts are highlighted, and students are prompted to choose alternate courses.

4. Facilitating Registration Integration
   - The application is integrated with the college's existing registration system.
   - Data synchronization between the application and the registration system occurs seamlessly.
   - The integration ensures real-time updates to student enrollment records.

5. Collecting Student Feedback
   - Students can provide feedback on courses after completion.
   - Feedback includes ratings and comments.
   - Administrators can view and analyze feedback through a dedicated dashboard.

As a Faculty Member

1. Creating and Updating Course Information
   - Faculty can create and update course details through an intuitive interface.
   - Changes are saved and updated in real-time.
   - The interface allows uploading of syllabus documents and other course materials.

2. Monitoring Enrollment Numbers
   - Faculty can view real-time enrollment numbers for their courses.
   - Alerts are generated when enrollment reaches capacity or falls below a certain threshold.

3. Receiving Student Feedback
   - Faculty can access feedback left by students for their courses.
   - Feedback is presented in an organized manner, including average ratings and individual comments.
   - Faculty can respond to feedback within the platform.

4. Communicating with Students
   - Faculty can send messages to students enrolled in their courses.
   - Notifications are sent to students via email and within the application.
   - Faculty can schedule messages and announcements to be sent at specific times.

5. Reviewing Enrollment Prerequisites
   - Faculty can review and update prerequisite requirements for their courses.
   - The system ensures that only eligible students can enroll.
   - Faculty receive notifications of any prerequisite updates or issues.

POKER PLANNING:

User Stories:

1. User Story 1:
As a user, I want to log in using my email and password, so that I can access my personal dashboard.
2. User Story 2:
As a user, I want to reset my password via email, so that I can regain access if I forget it.
3. User Story 3:
As a user, I want to view and edit my profile information, so that I can keep my information up to date.
4. User Story 4:
As an admin, I want to view a list of all registered users, so that I can manage user accounts.
5. User Story 5:
As an admin, I want to generate usage reports, so that I can analyze user engagement and system performance.

Poker Planning:

1. Prepare the User Stories: The stories above are well-defined with clear objectives and roles.
2. Explain the User Stories:
   - User Story 1: Basic authentication functionality.
   - User Story 2: Password reset functionality, involving sending emails.
   - User Story 3: Profile management, involving form handling and database updates.
   - User Story 4: Admin feature to view users, requiring an admin interface and user listing.
   - User Story 5: Report generation, possibly involving data aggregation and export features. 3. Discuss the User Stories: The team discusses any technical challenges, dependencies, or uncertainties.
4. Estimate:
   - User Story 1: Team members might estimate 3, 5, 3, 8, 5 (indicating moderate complexity due to security considerations).
   - User Story 2: Team members might estimate 5, 8, 5, 8, 5 (due to email integration and security concerns).
   - User Story 3: Team members might estimate 8, 8, 13, 8, 8 (due to complexity in editing and validation).
   - User Story 4: Team members might estimate 3, 5, 3, 5, 3 (relatively straightforward list display).
   - User Story 5: Team members might estimate 13, 21, 13, 13, 21 (due to the complexity of report generation and data handling).
5. Reveal: Team members reveal their estimates.

6. Discuss Differences:

- User Story 1: Team discusses why someone estimated 8 while others estimated lower. They agree that 5 is reasonable after considering all points.
- User Story 2: Similar discussion about the 8 estimates, agreeing on 5 after considering email sending libraries.
- User Story 3: Agreement on 8 after discussing the data validation and possible edge cases.
- User Story 4: Agreement on 3 as it is straightforward.
- User Story 5: Longer discussion due to high variability. They settle on 13 after considering data export features.

7. Re-estimate: The team finalizes their estimates as:

- User Story 1: 5
- User Story 2: 5
- User Story 3: 8 - User Story 4: 3
- User Story 5: 13 Final Estimates:

- User Story 1: 5 points
- User Story 2: 5 points
- User Story 3: 8 points - User Story 4: 3 points
- User Story 5: 13 points

These estimates can now be used for sprint planning and workload management.

NON FUNCTIONAL REQUIREMENTS:

PERFORMANCE:

Response Time: The system should respond to user actions within 2 seconds under normal load conditions.

Scalability: The system should be able to handle up to 10,000 concurrent users without degradation in performance.

Availability: The system should be available 99.9% of the time, ensuring minimal downtime.

Data Processing: Course registrations and updates should be processed within 1 second of submission.

SECURITY:

Authentication: All users must authenticate using secure college credentials.

Authorization: Role-based access control (RBAC) should be implemented to ensure only authorized users can access speci c features (students, faculty, administrators).
Data Encryption: Sensitive data, including login credentials and personal information, should be encrypted both in transit and at rest.
Audit Logs: The system should maintain audit logs of all user activities, especially related to course registration submissions, approvals, and rejections.
Session Management: Secure session management should be enforced, including timeout policies to log out inactive users after 15 minutes of inactivity.

USABILITY:

User Interface: The UI should be intuitive and user-friendly, allowing users to navigate easily through different sections.
Accessibility: The system should comply with accessibility standards (e.g., WCAG 2.1) to ensure it is usable by people with disabilities.
Responsiveness: The system should be responsive and work seamlessly on various devices, including desktops, tablets, and smartphones.
Error Handling: Clear and concise error messages should be provided to guide users in case of incorrect inputs or system errors.

RELIABILITY:

Fault Tolerance: The system should handle failures gracefully, ensuring data integrity and consistent state.
Backup and Recovery: Regular backups should be taken, and a recovery plan should be in place to restore data in case of failures.

MAINTAINABILITY:

Code Quality: The codebase should follow industry best practices for readability, modularity, and documentation.

Con gurability: System con gurations (e.g., course details, registration policies) should be easily manageable through an admin interface without requiring code changes.
Logging and Monitoring: The system should include comprehensive logging and monitoring to detect and troubleshoot issues promptly.

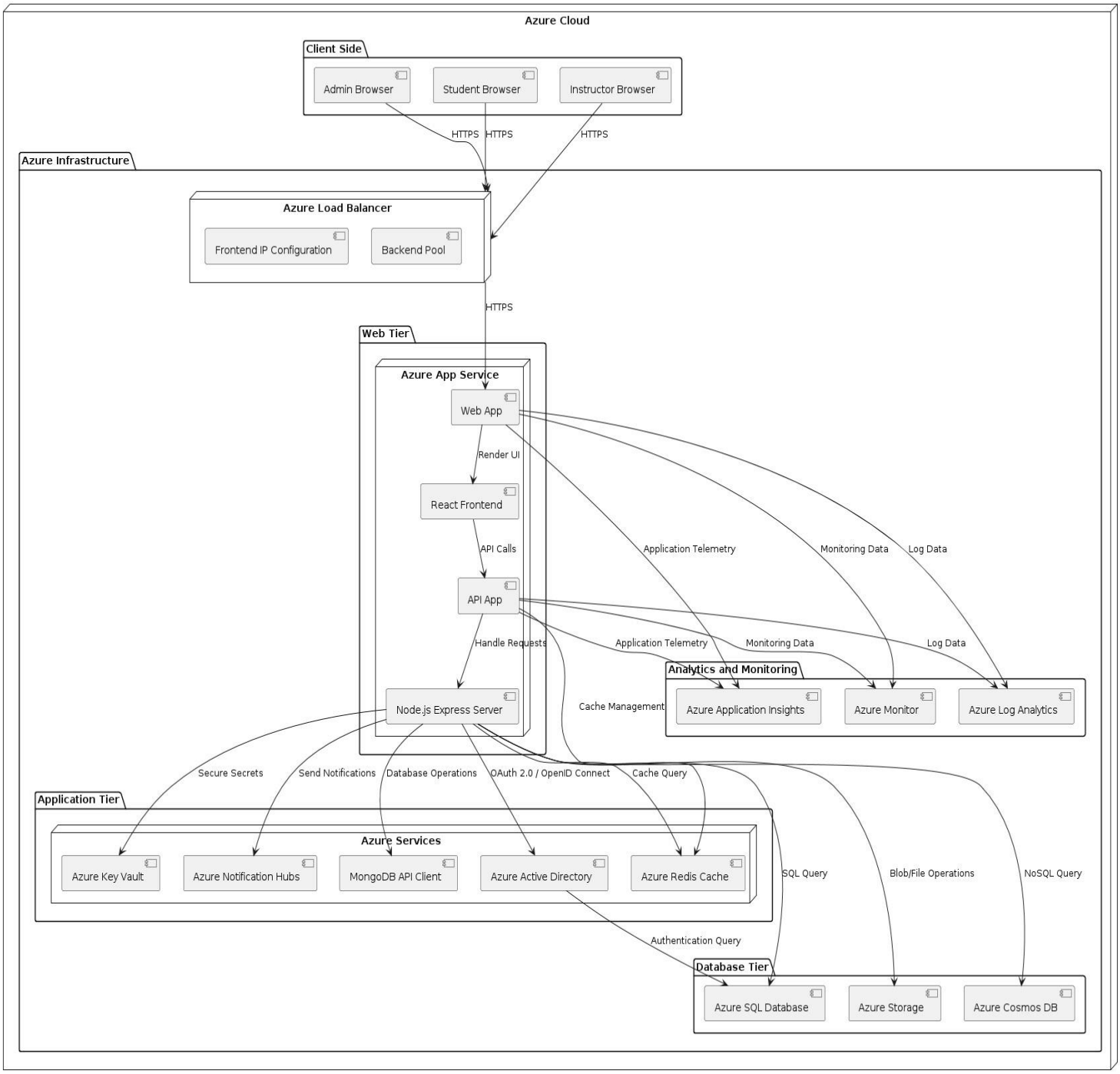
INTEROPERABILITY:
Integration: The system should be able to integrate with existing college systems, such as the student information system (SIS) and human resources (HR) system.
Data Export: Provide options to export data in standard formats (CSV, Excel) for reporting and analysis.

LEGALANDCOMPLIANCE:
Data Protection: The system must comply with relevant data protection regulations (e.g., GDPR, FERPA) to ensure the privacy and security of student data.
Record Retention: Course registration records should be retained according to college policies and legal requirements.

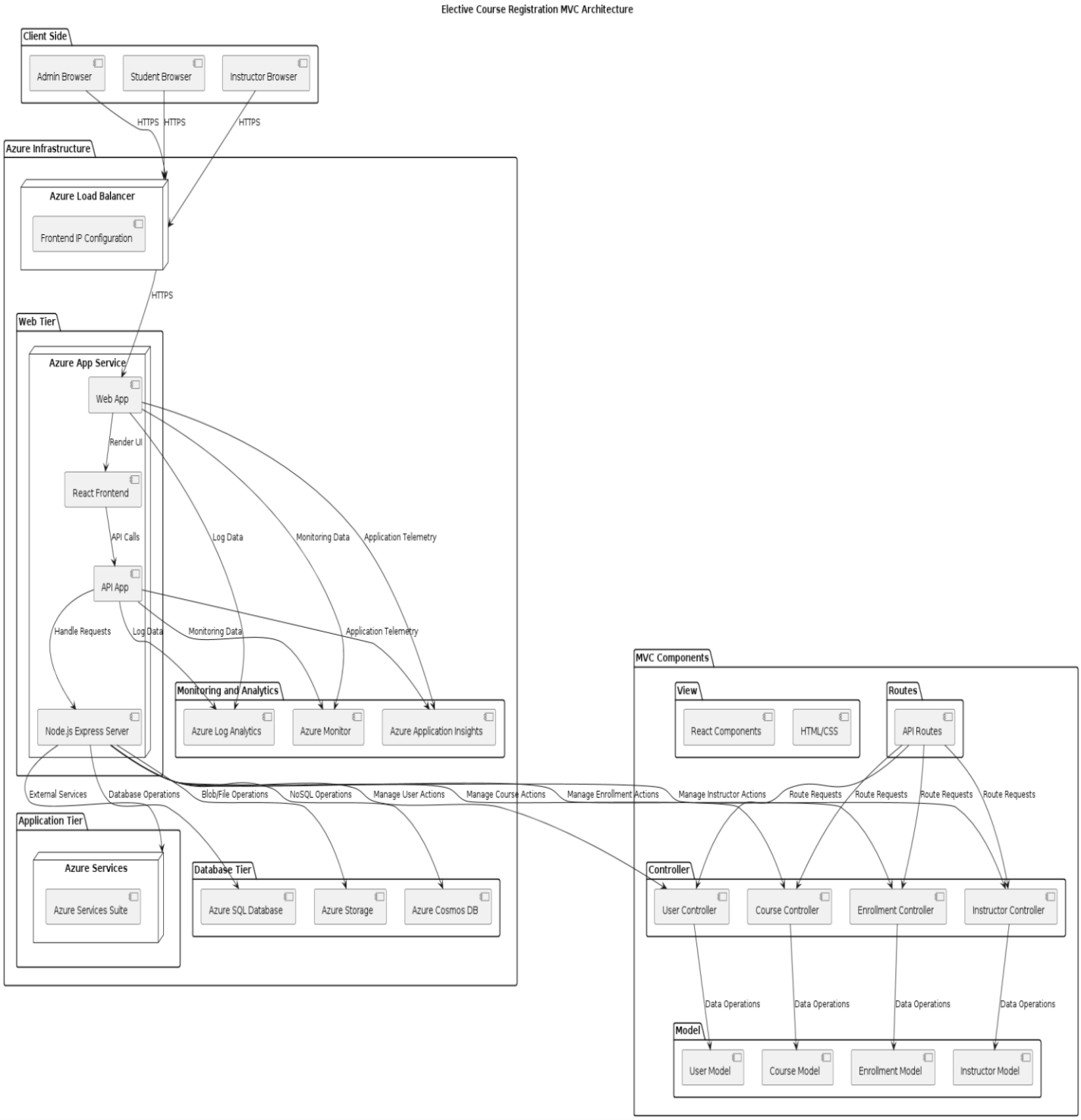
EFFICIENCY:
Resource Utilization: The system should optimize resource usage, including CPU, memory, and network bandwidth, to ensure ef cient operation.
Power Consumption: For any hardware components, power consumption should be minimized, especially for systems hosted in data centers.

ARCHITECTURAL DIAGRAM:

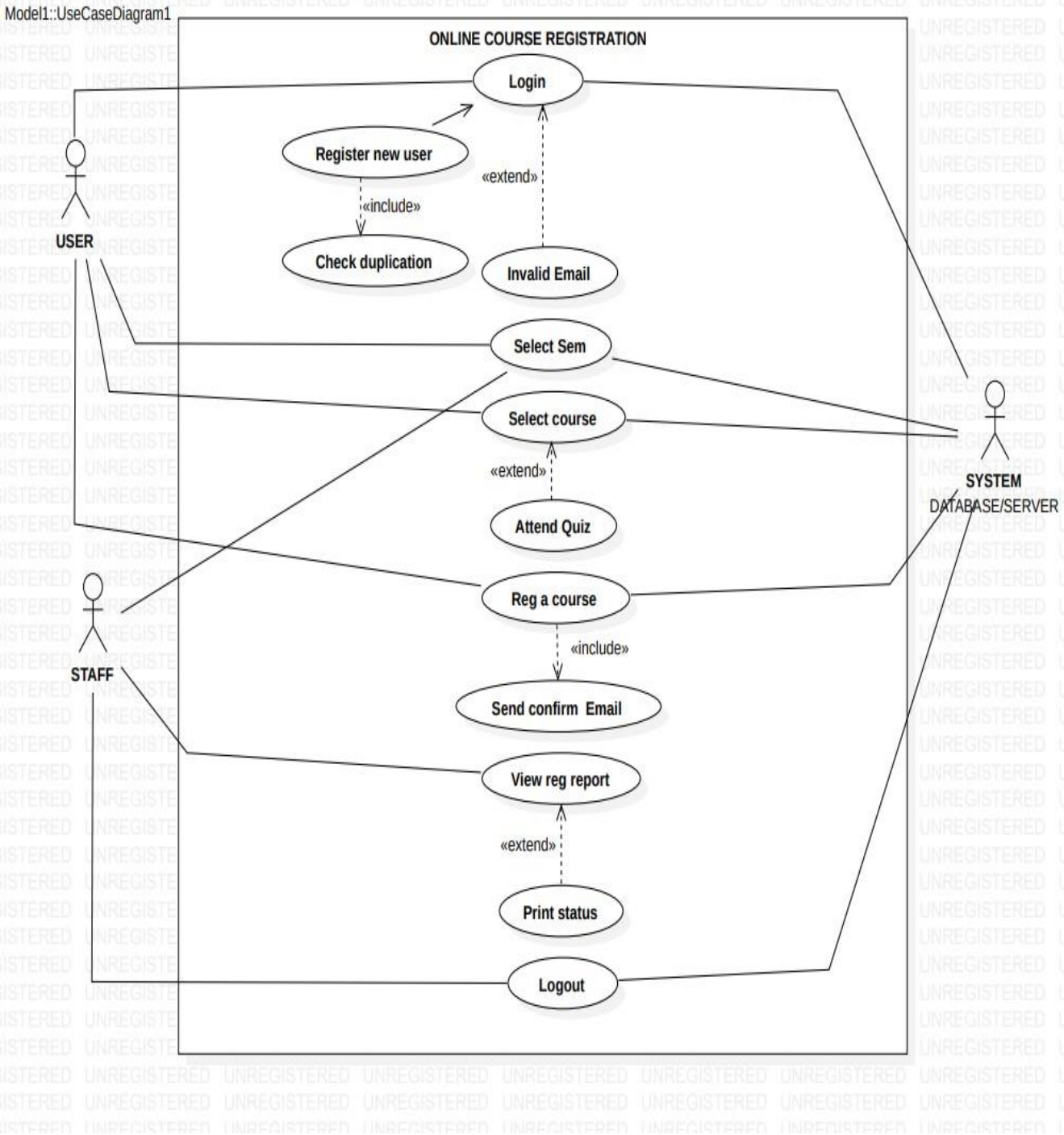
Elective Course Registration System Architecture



MVC ARCHITECTURAL DIAGRAM:



USE CASE DIAGRAM:



DESIGN PRINCIPLES:

Design Principles for Elective Course Selection Application

1. User-Centric Design

- Empathy with Users:
- Conduct user research to understand the needs, behaviors, and pain points of students, faculty, and administrators.
- Ensure the application's interface and interactions align with users' expectations and preferences.
- Provide features like search, filtering, and course comparisons to facilitate easy decision-making for students.

- Accessibility:
- Design the application to be usable by people with varying abilities.
- Follow WCAG (Web Content Accessibility Guidelines) standards, ensuring features like screen reader compatibility, keyboard navigation, and sufficient color contrast.

2. Simplicity and Clarity

- Minimalistic Design:
- Keep the interface clean and straightforward, focusing on essential features to prevent user overload.
- Use clear, concise language and avoid technical jargon to make the application understandable for all users.

- Clear Navigation:
- Provide intuitive and straightforward navigation paths so users can easily find and access different sections (e.g., course browsing, enrollment).
- Use a consistent layout and labeling system to help users familiarize themselves with the application quickly.

3. Consistency

- Uniform Interface Elements:
- Maintain consistency in the use of colors, fonts, buttons, and other UI components throughout the application.
- Ensure similar actions produce similar results to help users predict and understand the application's behavior.

- Design Patterns:
- Utilize familiar design patterns to create a seamless user experience.
- Ensure consistency across different devices and screen sizes by following responsive design principles.

4. Feedback and Responsiveness

- Immediate Feedback:
- Provide real-time feedback for user actions (e.g., successful enrollment, error messages) to confirm that the application is responsive to their inputs.
- Use visual indicators like loading spinners, success notifications, and error messages to keep users informed about the status of their actions.

- Responsive Design:
- Design the application to work flawlessly on various devices, including desktops, tablets, and smartphones.
- Optimize performance and usability across different screen sizes and orientations.

5. Performance and Efficiency

- Optimized Performance:
- Minimize load times by optimizing images, scripts, and other resources.
- Implement efficient data fetching and caching strategies to enhance performance.

- Efficient Workflows:
- Design workflows that reduce the number of steps required to complete tasks, such as enrolling in courses or marking favorites.
- Enable users to achieve their goals with minimal effort, ensuring a smooth and pleasant user experience.

6. Scalability and Flexibility

- Modular Design:
- Develop the application using a modular architecture to facilitate easy updates and the addition of new features.
- Design components that can be reused across different parts of the application, enhancing maintainability and scalability.

- Adaptability:
- Ensure the application can handle increasing amounts of data and users without compromising performance.

- Make the application adaptable to future changes and technological advancements, allowing for seamless upgrades and enhancements.

7. Security and Privacy

- Data Protection:
- Implement robust security measures to protect user data from unauthorized access and breaches.
- Use best practices for encryption, authentication, and authorization to ensure data integrity and confidentiality.

- Privacy Compliance:
- Ensure the application complies with relevant data protection regulations (e.g., GDPR, CCPA).
- Provide clear privacy policies and obtain user consent for data collection and usage, respecting their privacy rights.

8.Collaboration and Communication

- Transparent Communication:
- Facilitate clear and open communication among team members throughout the development process.
- Use collaborative tools and regular meetings to keep everyone aligned and informed.

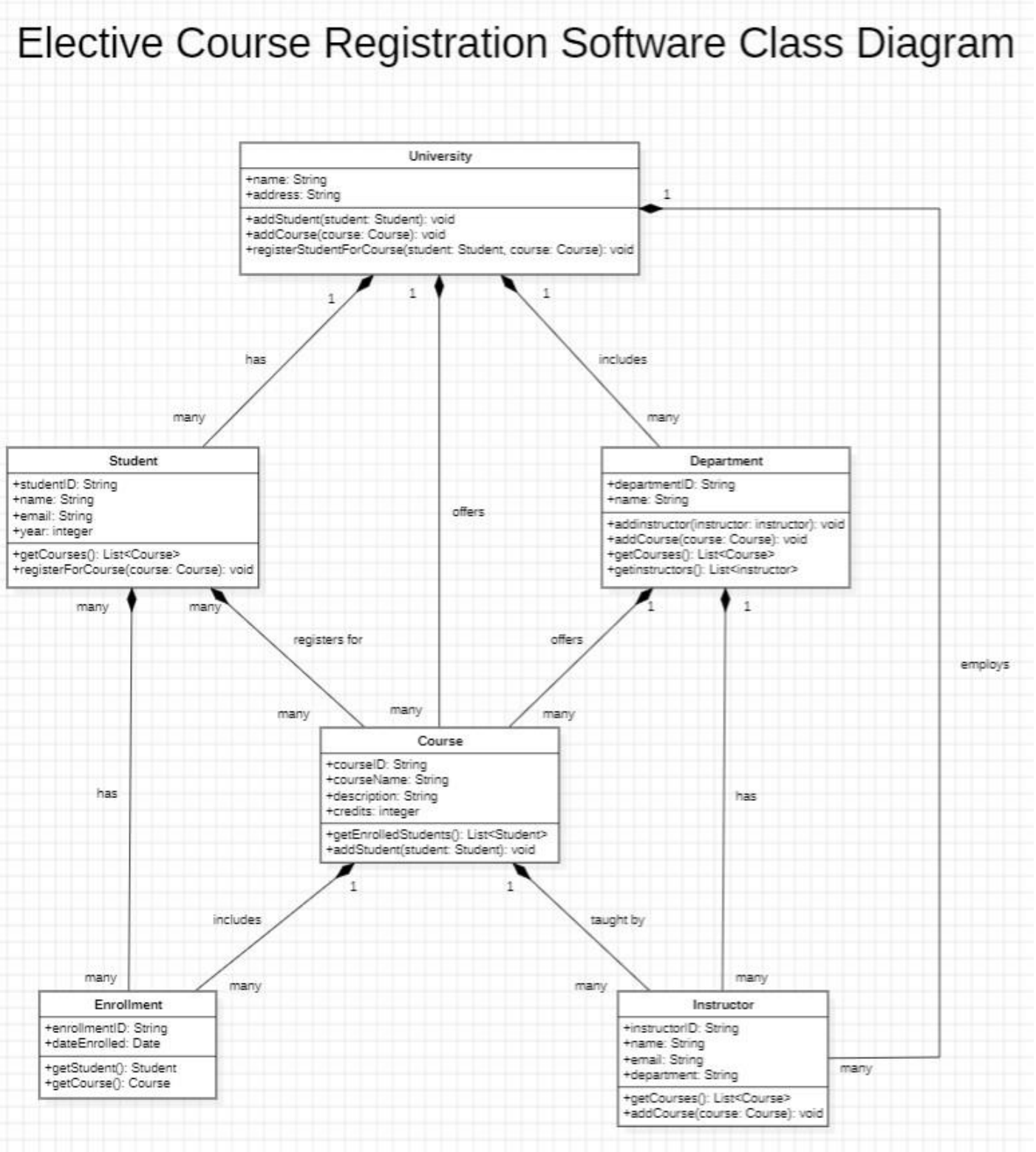
- User Feedback Loop:
- Establish a continuous feedback loop with users to gather insights and improve the application iteratively.
- Encourage users to provide feedback through surveys, ratings, and reviews, and act on their suggestions to enhance the application.

9. Testing and Quality Assurance

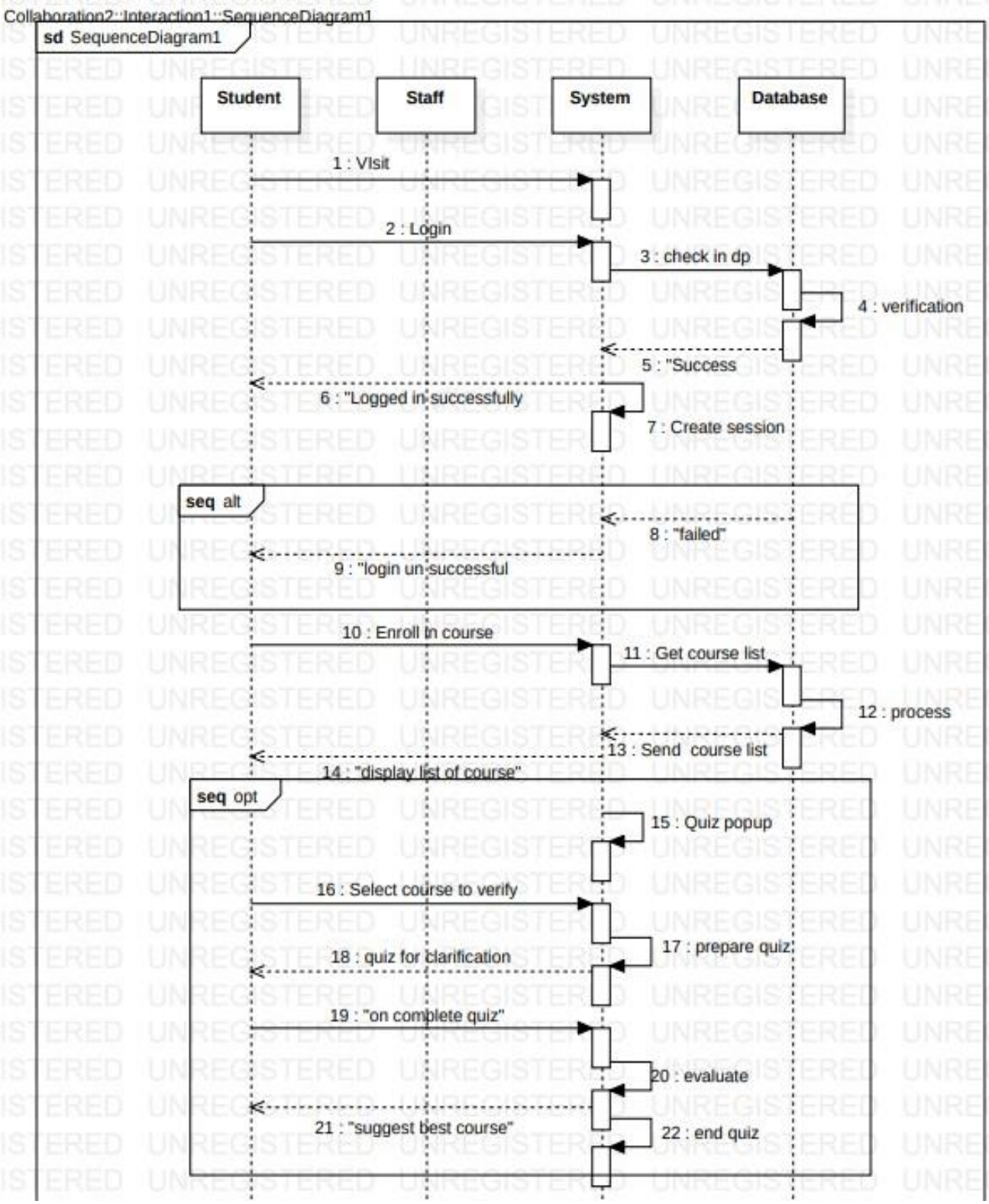
- Continuous Testing:
- Implement automated testing (unit, integration, end-to-end) to ensure high quality and reliability.
- Conduct regular user testing to identify and resolve usability issues, ensuring the application meets user needs effectively.

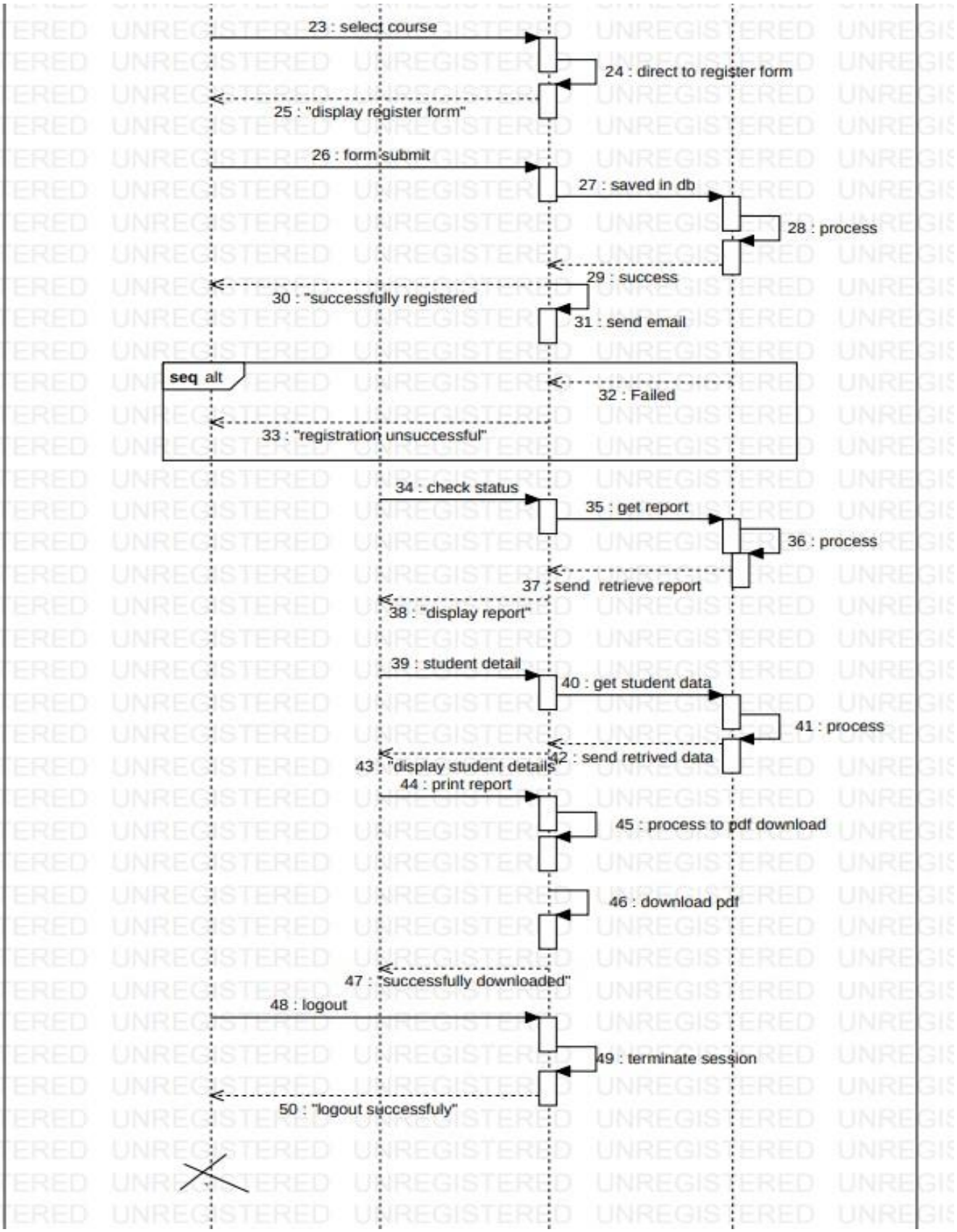
- Quality Metrics:
- Define and track key performance indicators (KPIs) to measure the application's success and quality.

CLASS DIAGRAM:

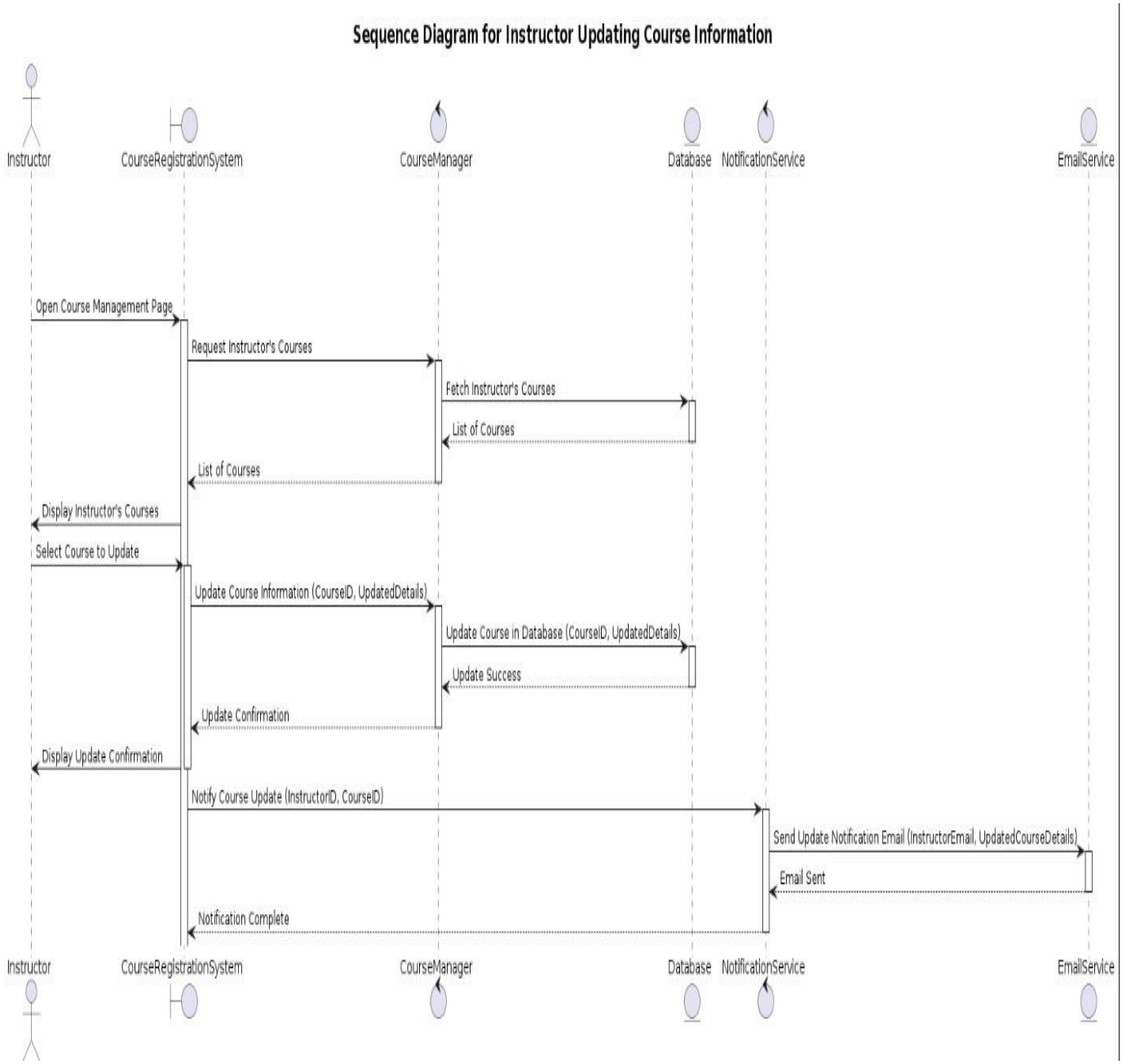


SEQUENCE DIAGRAM:

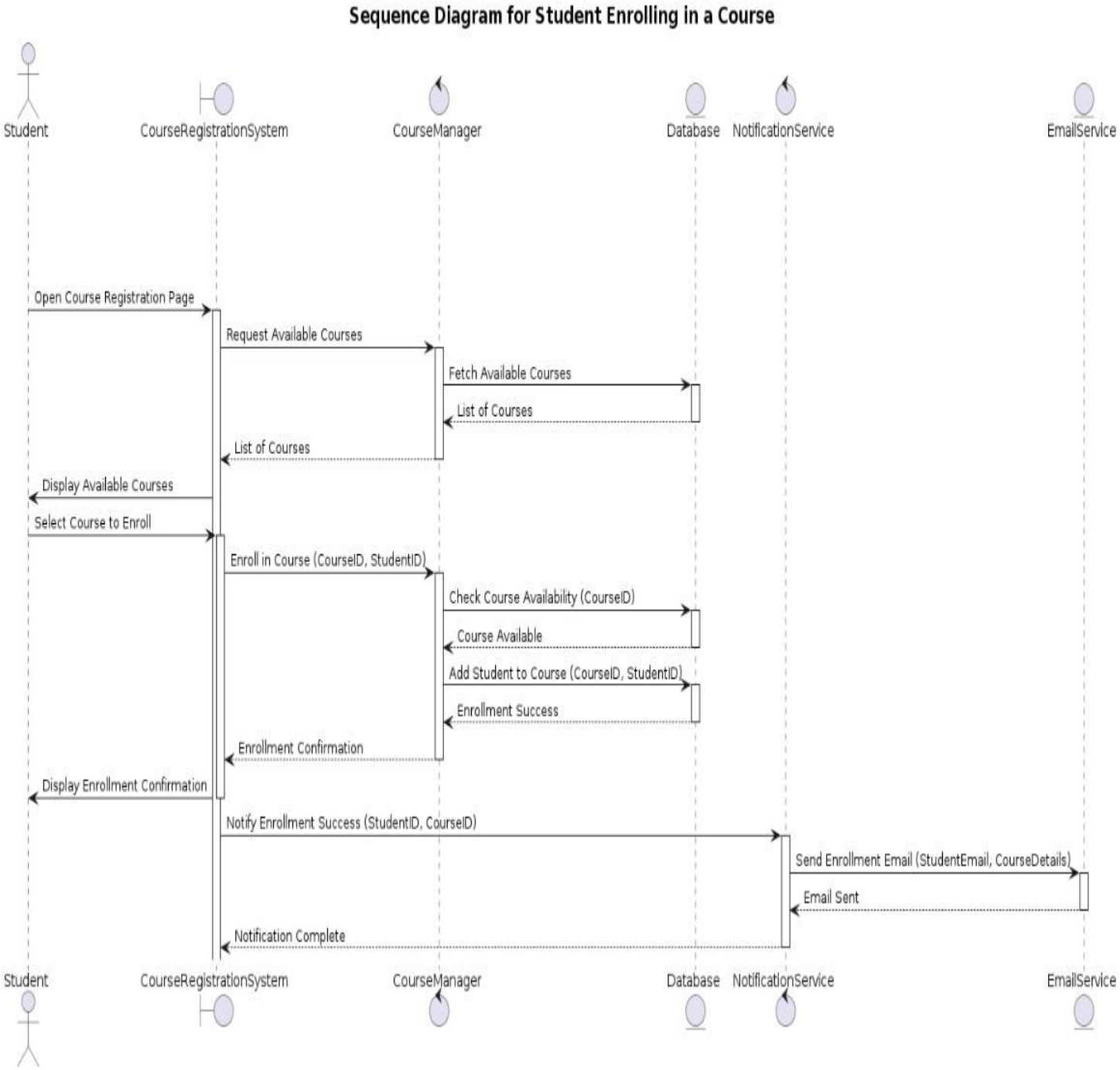



23 : select course
24 : direct to register form
25 : "display register form"
26 : form submit
27 : saved in db
28 : process
29 : success
30 : "successfully registered
31 : send email
seq alt
32 : Failed
33 : "registration unsuccessful"
34 : check status
35 : get report
36 : process
37 : send retrieve report
38 : "display report"
39 : student detail
40 : get student data
41 : process
42 : send retrived data
43 : "display student details
44 : print report
45 : process to pdf download
46 : download pdf
47 : "successfully downloaded"
48 : logout
49 : terminate session
50 : "logout successfuly"

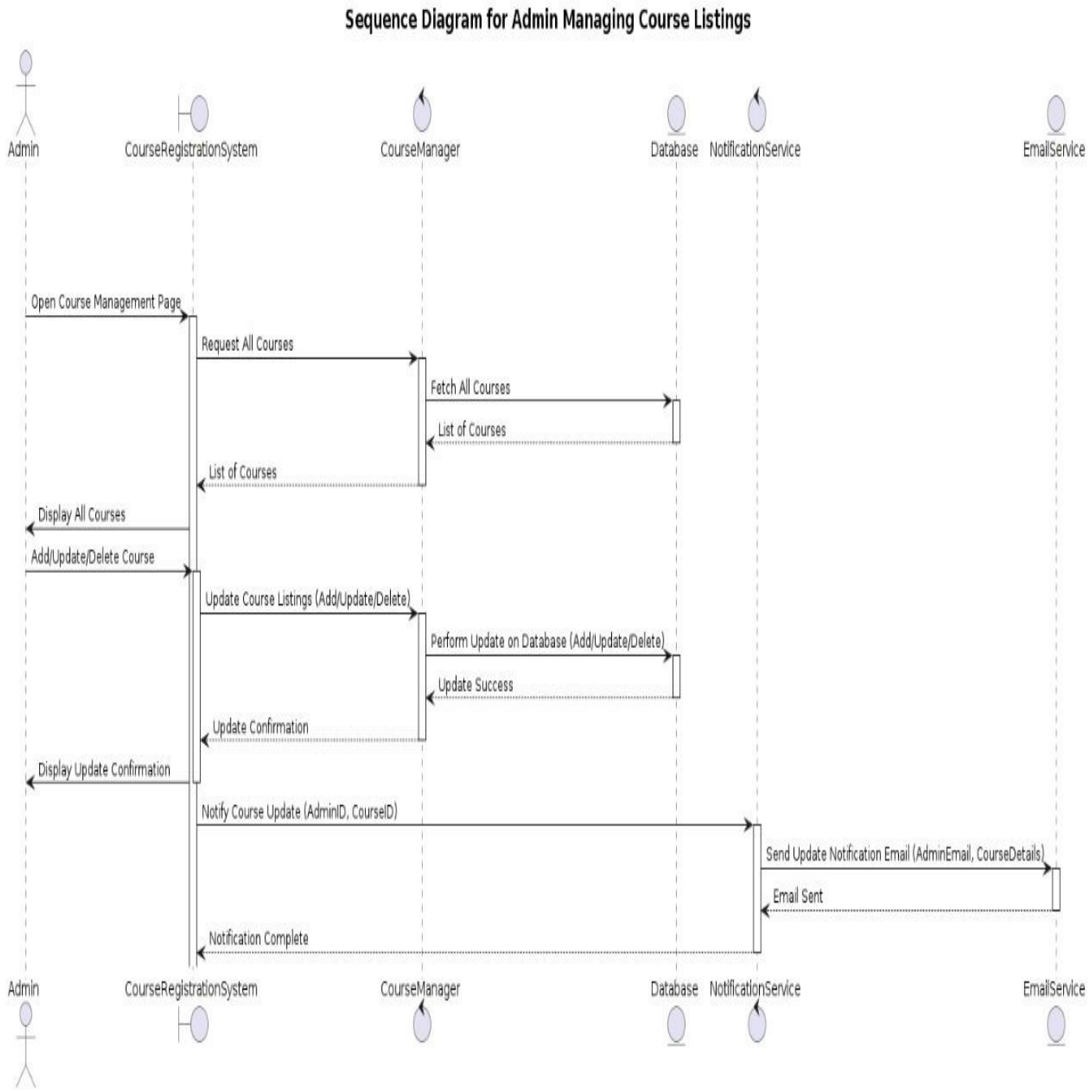
INSTRUCTOR UPDATING COURSE INFORMATION:



STUDENT ENROLLING IN A COURSE:



ADMIN MANAGING COURSE LISTING:



TEST STRATEGY:

Test Strategies for Elective Course Selection Application

1. Test Strategy Overview

- Objective: Ensure the application is functional, reliable, and provides a seamless user experience.
- Scope: Testing covers all user-facing features, backend services, and integrations with external systems.
- Types of Testing: Unit Testing, Integration Testing, System Testing, Acceptance Testing, Performance Testing, Security Testing, and Usability Testing.

2. Types of Testing

1. Unit Testing
- Objective: Verify that individual components and functions work as expected.
- Tools: Jest (JavaScript), Mocha (JavaScript), Chai (JavaScript) - Scope: Test all functions, methods, and components in isolation.

2. Integration Testing
- Objective: Ensure that different modules and services interact correctly.
- Tools: Postman (API testing), Supertest (JavaScript)
- Scope: Test interactions between the frontend and backend, database operations, and third-party services.

3. System Testing
- Objective: Validate the complete and integrated application meets the requirements.
- Tools: Selenium (WebDriver), Cypress (JavaScript)
- Scope: Perform end-to-end testing of user stories and scenarios.

4. Acceptance Testing
- Objective: Confirm that the application meets the acceptance criteria and is ready for deployment.
- Tools: Cucumber (BDD)
- Scope: Involve stakeholders to validate the application's functionality against user stories.

5. Performance Testing
- Objective: Ensure the application performs well under expected and peak loads.
- Tools: JMeter, LoadRunner
- Scope: Test response times, throughput, and scalability.

6. Security Testing

- Objective: Identify and mitigate security vulnerabilities.
- Tools: OWASP ZAP, Burp Suite
- Scope: Perform vulnerability scanning, penetration testing, and security audits.

7. Usability Testing
- Objective: Ensure the application is user-friendly and intuitive.
- Tools: UsabilityHub, UserTesting
- Scope: Conduct usability sessions with actual users, gather feedback, and iterate on design.

TEST CASES:

Test Cases for Elective Course Selection Application

1. User Authentication and Profile Management

- Test Case 1.1: User Registration
- Description: Verify that a new user can register with valid credentials.
- Steps:
1. Navigate to the registration page.
2. Enter valid details (name, email, password).
3. Submit the form.
- Expected Result: User is registered and redirected to the profile setup page.

- Test Case 1.2: User Login
- Description: Verify that a registered user can log in with correct credentials.
- Steps:
1. Navigate to the login page.
2. Enter valid email and password.
3. Submit the form.
- Expected Result: User is logged in and redirected to the dashboard.

- Test Case 1.3: Role-Based Access Control
- Description: Verify that different roles (student, faculty, admin) see appropriate content. - Steps:
1. Log in as a student.
2. Log in as a faculty member.
3. Log in as an administrator.
- Expected Result: Each role sees content relevant to their role.

2. Course Browsing and Search

- Test Case 2.1: Course List Display
- Description: Verify that all available courses are displayed correctly.
- Steps:
1. Navigate to the course browsing page.
- Expected Result: List of all courses is displayed with correct information.

- Test Case 2.2: Course Search Functionality
- Description: Verify that users can search for courses using keywords. - Steps:

1. Enter a keyword in the search bar.
2. Click the search button.
- Expected Result: Courses matching the keyword are displayed.

- Test Case 2.3: Filtering Courses
- Description: Verify that users can filter courses by department, credit hours, and prerequisites.
- Steps:
1. Apply filters.
- Expected Result: Courses matching the filters are displayed.

3. Course Descriptions

- Test Case 3.1: View Course Description
- Description: Verify that users can view short and detailed descriptions of courses. -
Steps:
1. Click on a course title.
- Expected Result: Short and detailed descriptions are displayed correctly.

4. Favorites and Comparison

- Test Case 4.1: Mark Course as Favorite
- Description: Verify that users can mark courses as favorites.
- Steps:
1. Click the favorite icon on a course.
- Expected Result: Course is added to the favorites list.

- Test Case 4.2: Compare Courses
- Description: Verify that users can compare up to three courses.
- Steps:
1. Select three courses.
2. Click the compare button.
- Expected Result: Comparison table is displayed with selected courses.

5. Schedule Integration

- Test Case 5.1: Display Course Schedule
- Description: Verify that the selected courses are displayed in the user's schedule. - Steps:
1. Enroll in a course.
2. Navigate to the schedule page.
- Expected Result: Enrolled courses are displayed in the calendar view.

- Test Case 5.2: Conflict Detection
- Description: Verify that time conflicts are detected and alerted.
- Steps:
1. Enroll in two courses with overlapping times.
- Expected Result: Alert is displayed indicating the time conflict.

6. Course Enrollment

- Test Case 6.1: Enroll in a Course
- Description: Verify that users can enroll in courses successfully.
- Steps:
1. Select a course.
2. Click the enroll button.
- Expected Result: User is enrolled, and the course status is updated.

- Test Case 6.2: Waitlist Status
- Description: Verify that users can see their waitlist status if a course is full.
- Steps:
1. Attempt to enroll in a full course.
- Expected Result: User is added to the waitlist, and the status is displayed.

7. Notifications

- Test Case 7.1: Receive Enrollment Notifications
- Description: Verify that users receive notifications for enrollment updates.
- Steps:
1. Enroll in a course.
- Expected Result: Notification is received confirming enrollment.

- Test Case 7.2: Notification Bell Icon
- Description: Verify that the notification bell icon updates with new notifications.
- Steps:
1. Trigger a notification (e.g., enrollment confirmation).
- Expected Result: Bell icon shows a badge indicating a new notification.

8. User Reviews and Ratings

- Test Case 8.1: Submit a Review
- Description: Verify that users can submit reviews for courses.
- Steps:

1. Navigate to a course detail page.
2. Fill out the review form with a rating and comments.
3. Submit the review.
- Expected Result: Review is submitted and displayed on the course detail page.

- Test Case 8.2: View Course Reviews
- Description: Verify that users can view reviews and ratings for courses.
- Steps:
1. Navigate to a course detail page.
- Expected Result: Reviews and average ratings are displayed correctly.

.

DEPLOYMENT DIAGRAM:

Overview

- The deployment diagram illustrates the physical layout of the system, showing how different components are deployed within the Azure Cloud environment.

ClientSide

- Browsers: Accessed by Students, Faculty, and Admins through HTTPS.

WebServer

- Azure App Service (Web App): Hosts the web application.
- Azure Load Balancer: Distributes incoming web traf c to ensure reliability and performance.
- React Frontend: Renders the user interface for client-side interactions.

Machines

- Web Server VMs: Handle HTTP requests from the clients.
- App Server VMs: Process business logic and API requests.
- Database Server VM: Manages database operations.

ApplicationServer

- Azure App Service (API App): Manages API calls and handles requests using Node.js Express Server.
- Azure Services: Facilitates various backend services including SQL queries, Blob/File operations, and NoSQL queries.

DatabaseServer

- Azure SQL Database: Primary database for storing structured data.
- Azure Storage: Handles le and blob storage requirements.
- Azure Cosmos DB: Manages NoSQL data for high performance and scalability.

AnalyticsandMonitoring

- Azure Application Insights: Collects application telemetry data for monitoring performance and usage.
- Azure Monitor: Tracks system performance and health.
- Azure Log Analytics: Aggregates and analyzes log data for insights and troubleshooting.

Summary

- These diagrams collectively describe the business architecture and deployment setup of the Elective Course Registration System, showcasing the seamless integration of various stakeholders, activities, and technologies to ensure ef cient course registration and management. The system leverages Azure cloud services to provide robust performance, scalability, and monitoring capabilities.

AZURE DEPLOYMENT DIAGRAM: