CAIRO UNIVERSITY

FACULTY OF SCIENCE
DEPARTMENT OF MATHEMATICS

Uplearn
Learning Management System

PRESENTERS

Mohamed Atef Shata
Mahmoud Atef
Omar Kenawi
Muhammed Waild
Abdallah Mohamed
Hazem Mohamed
Ahmed Hany

Supervised By Professor

Dr. Hossam Hassan

UPLEARN

Mohamed Atef Shata (2027115)
Mahmoud Atef Mahmoud (2027453)
Omar Kenawi Elsayed (2027471)
Muhammed Waild Mahrous (1927194)
Abdallah Mohamed Sayed (2027069)
Hazem Mohamed Fawzy (2027220)
Ahmed Hany Ahmed (1928275)

Department of Mathematics
Faculty of Science
Cairo University

Contents

1 Introduction

1.1 Purpose/Motivation

The purpose of Up Learn is to create a comprehensive and user-friendly Learning Management System (LMS) designed to enhance the educational experience for both students and educators. Our motivation stems from the need to provide an accessible, efficient, and engaging platform that facilitates the management, delivery, and tracking of educational courses and materials. Up Learn aims to bridge the gap between traditional educational methods and modern technological advancements, ensuring that learners have access to high-quality education. Improving educational outcomes is a central goal of Up Learn. The system supports adaptive learning technologies that tailor educational content and assessments to individual learner needs and preferences. This personalization enhances engagement and helps learners achieve better outcomes by focusing on their unique strengths and areas for improvement. Additionally, comprehensive reporting and analytics provide educators with valuable insights into student performance, enabling them to identify trends and make data-driven decisions to improve teaching strategies and learning experiences.Facilitating efficient course management is another key motivation behind Up Learn. The system simplifies administrative tasks such as course creation, user enrollment, grade management, and communication, allowing educators to spend more time focusing on teaching and interacting with students. Designed to accommodate educational institutions of all sizes, from small training centers to large universities, Up Learn can scale according to the needs of the institution, ensuring it remains effective as the user base grows also Supporting continuous improvement is a key aspect of Up Learn. The system includes features for collecting and analyzing feedback from both learners and educators, allowing for ongoing improvements to ensure it evolves in response to user needs and technological advancements. Additionally, Up Learn offers resources and training for educators to enhance their teaching skills and stay updated with the latest educational practices and technologies.

1.2 Scope

This documentation provides an in-depth overview of the Up Learn LMS, detailing its features, functionality, and technical specifications. It is intended for use by developers, administrators, educators, and users who are involved in the deployment, customization, and use of the Up Learn system. The scope includes:

- System architecture and design
- Key features and functionalities
- User roles and permissions
- Technical specifications and requirements
- Installation and setup guide
- Usage instructions
- Maintenance and troubleshooting

1.3 Overview

Up Learn is designed to cater to various educational environments, from schools and universities to corporate training programs. The system offers a range of features aimed at enhancing the educational experience for all users. This documentation will provide a comprehensive guide to the following key features and functionalities:

- Course Creation and Management: Up Learn allows educators to create and manage courses with ease, providing tools for structuring content, scheduling classes, and organizing materials. Detailed instructions will be provided on how to utilize these tools to develop engaging and effective courses.
- User Enrollment and Management: The system supports efficient user enrollment processes, including manual enrollment, self-registration, and bulk enrollment options. We will discuss how to manage user roles and permissions to ensure a secure and organized learning environment.
- Technical Specifications and Requirements: Detailed information on the technical requirements for installing and running Up Learn, including hardware, software, and network prerequisites, will be provided to ensure a smooth setup and operation.
- Installation and Setup Guide: Step-by-step instructions for installing and configuring Up Learn will be covered, ensuring that administrators can get the system up and running efficiently.
- Usage Instructions: Comprehensive guides on how to use the various features of Up Learn will be included, helping users to navigate and utilize the system effectively.
- Maintenance and Troubleshooting: Best practices for maintaining the system and resolving common issues will be discussed, ensuring that Up Learn remains reliable and efficient over time.

1.4 Software Used

The development and collaboration process for Up Learn involved the use of several software tools to ensure effective communication, project management, and system design. The key tools used by our team include:

- Trello: Used for managing the product backlog and tracking project tasks. It provided a visual and organized way to prioritize work, set deadlines, and monitor progress.
- Draw.io: Utilized for designing the database schema and system architecture. It allowed us to create detailed and accurate diagrams that facilitated clear communication of the system design.
- Slack: Primary communication platform for team conferences, meetings, and real-time discussions. It enabled efficient collaboration and ensured that all team members were aligned and informed throughout the development process.

1.5 Definitions and Acronyms

This section provides definitions of all terms, acronyms, and abbreviations that might exist to properly interpret the documentation.

Term	**Definition**
LMS	Learning Management System
ERD	Entity-Relationship Diagram
UI	User Interface
API	Application Programming Interface
DBMS	Database Management System

2 System Overview

Uplearn is an LMS system for Course Registration and Faculty Communication System designed to streamline the processes of course enrollment for students and enhance communication among faculty members within an academic institution. The system offers the following core

2.1 Functionalities

- **User Authentication and Authorization**

  Detailed Description: Implement a robust system to handle user login, registration, and role-based access control. Ensure secure authentication methods, such as hashed passwords and multi-factor authentication (MFA). Authorization should define what different roles (e.g., students, teachers, administrators) can and cannot access.

  Explanation: This functionality ensures that only authorized users can access the system and perform actions according to their roles. For example, a student can enroll in courses and view grades, while a teacher can create and manage courses, and an administrator can manage users and the system settings.

- **Course Management**

  Detailed Description: Allow instructors to create, update, and delete courses. Each course should include a title, description, syllabus, schedule, and other relevant information. Instructors should also be able to upload course materials, such as documents, videos, and links.

  Explanation: This feature enables teachers to manage the content and structure of their courses, ensuring that students have access to all necessary materials and information. It helps keep the learning process organized and structured.

- **Enrollment Management**

  Detailed Description: Facilitate the process for students to enroll in courses. This includes viewing available courses, enrolling in selected courses, and handling prerequisites or enrollment limits. Administrators should have the ability to manage and override enrollments when necessary.

  Explanation: This system ensures that students can easily join courses they are interested in while maintaining control over the enrollment process. It helps manage class sizes and prerequisites efficiently.

- **Content Delivery**

  Detailed Description: Provide a platform for delivering course content, including lectures, assignments, quizzes, and exams. Support various content types, such as text, multimedia (videos, audio), and interactive elements (quizzes, discussions).

Explanation: This functionality is crucial for delivering educational content to students in an engaging and accessible manner. It supports different learning styles and keeps students engaged with diverse types of content.

- **Communication Tools**

  Detailed Description: Integrate communication tools such as discussion forums, chat, and email to facilitate interaction between students and instructors. These tools should support both one-on-one and group communications.

  Explanation: Effective communication is essential for a successful learning experience. These tools help students and teachers interact, collaborate, and clarify doubts, enhancing the overall educational experience.

- **Assessment and Evaluation**

  Detailed Description: Develop features for creating, distributing, and grading assignments, quizzes, and exams. Support automated grading for objective questions and provide tools for instructors to grade subjective answers. Include analytics to track student performance.

  Explanation: This functionality allows for the systematic assessment of student understanding and progress. Automated grading saves time, while detailed analytics help instructors identify areas where students may need additional support.

- **Gradebook**

  Detailed Description: Implement a gradebook that records and displays student grades for assignments, quizzes, exams, and overall course performance. Allow students to view their grades and instructors to manage and update them.

  Explanation: The gradebook is essential for tracking academic performance. It provides transparency for students regarding their progress and enables instructors to manage grades efficiently.

- **Reporting and Analytics**

  Detailed Description: Provide reporting tools that generate detailed reports on student performance, course effectiveness, and system usage. Include data visualization tools to help administrators and instructors make informed decisions based on analytics.

  Explanation: These tools help stakeholders understand trends and patterns in learning and system usage. They are essential for continuous improvement of the learning process and system efficiency.

- **User Profile Management**

  Detailed Description: Allow users to manage their profiles, including personal information, contact details, and preferences. Ensure that users can update their profiles and control their privacy settings.

  Explanation: User profiles personalize the learning experience and allow for better communication and interaction. Privacy settings ensure users feel secure sharing their information.

- **Calendar and Scheduling**

  Detailed Description: Integrate a calendar system to schedule classes, assignments, exams, and events. Allow users to sync the LMS calendar with their personal calendars and receive notifications for upcoming deadlines and events.

  Explanation: This functionality helps students and instructors keep track of important dates and manage their time effectively. Notifications ensure that users do not miss critical deadlines.

- **Mobile Accessibility**

  Detailed Description: Ensure the LMS is accessible on mobile devices through a responsive web design or a dedicated mobile app. Include offline capabilities to allow users to access content without an internet connection.

  Explanation: Mobile accessibility ensures that students and instructors can access the LMS anytime, anywhere, enhancing flexibility and convenience in learning and teaching.

- **Integration with External Tools**

  Detailed Description: Provide integration with external tools and platforms, such as video conferencing software, plagiarism checkers, and third-party educational resources. Ensure smooth interoperability and data exchange.

  Explanation: Integration with external tools expands the functionality of the LMS and provides a seamless experience for users. It enhances the learning and teaching process by incorporating additional resources and tools.

- **Security and Data Privacy**

  Detailed Description: Implement robust security measures to protect user data and ensure privacy. This includes encryption, regular security audits, compliance with data protection regulations (e.g., GDPR), and secure data storage practices.

  Explanation: Security and privacy are paramount in an LMS to protect sensitive information and maintain user trust. Compliance with regulations ensures that the system adheres to legal standards and best practices.

- **Customization and Branding**

  Detailed Description: Allow institutions to customize the LMS interface and branding to match their unique identity. This includes themes, logos, and color schemes, as well as custom domain names and user interfaces.

  Explanation: Customization enhances the user experience by providing a familiar and branded environment. It helps institutions create a cohesive and professional look for their LMS.

- **Technical Support and Documentation**

Detailed Description: Provide comprehensive technical support and documentation for users, including FAQs, tutorials, user guides, and a helpdesk system. Ensure support is available through multiple channels, such as email, chat, and phone.

Explanation: Effective technical support and documentation help users troubleshoot issues and make the most of the LMS features. It enhances user satisfaction and reduces the learning curve for new users.

2.2 Context

The LMS is developed for use in universities and colleges to address common administrative challenges and improve operational efficiency. The context in which this system operates includes:

- **Academic Institutions**: The primary users are students and faculty members within universities and colleges.
- **Administrative Processes**: The system supports the academic administration by automating course registration and facilitating faculty communication.
- **User Roles**:Key user roles include students, faculty members, and administrative staff, professors and also students. Each role has specific access and functionalities tailored to their needs.

2.3 Design

The design of the LMS focuses on usability, scalability, and security to ensure it meets the needs of a diverse user base and adapts to growing institutional demands. The system's design components include:

- **System Architecture**: a System architecture is a structured to conceptualize, design, and manage the components of a system and their interactions.
  It encompasses the overall design, structure, and behavior of a system, ensuring that all components work together effectively to meet the system's requirements and goals.
- **Data Flow**: The data flow and relations between entities used for the best performance results, also Ensuring Data Integrity , Quality , Validation,Consistency,Scalability and Flexibility
- **Security Measures**: Security is paramount in a Course Registration and Faculty Communication System (LMS) due to the sensitive nature of the data it handles. so it importent to ensure the Security of the System Authentication , Authorization: Role-based access control (RBAC) restricts functionalities based on user roles (e.g., students, professor, admin).
- **Performance and Scalability** Performance and Scalability is key points because there is a lot of LMS like systems where it fail to handle the increase of users of over loading for this exact matter we take this into consideration thought the system Design

3 Database Design

3.1 ERD Overview

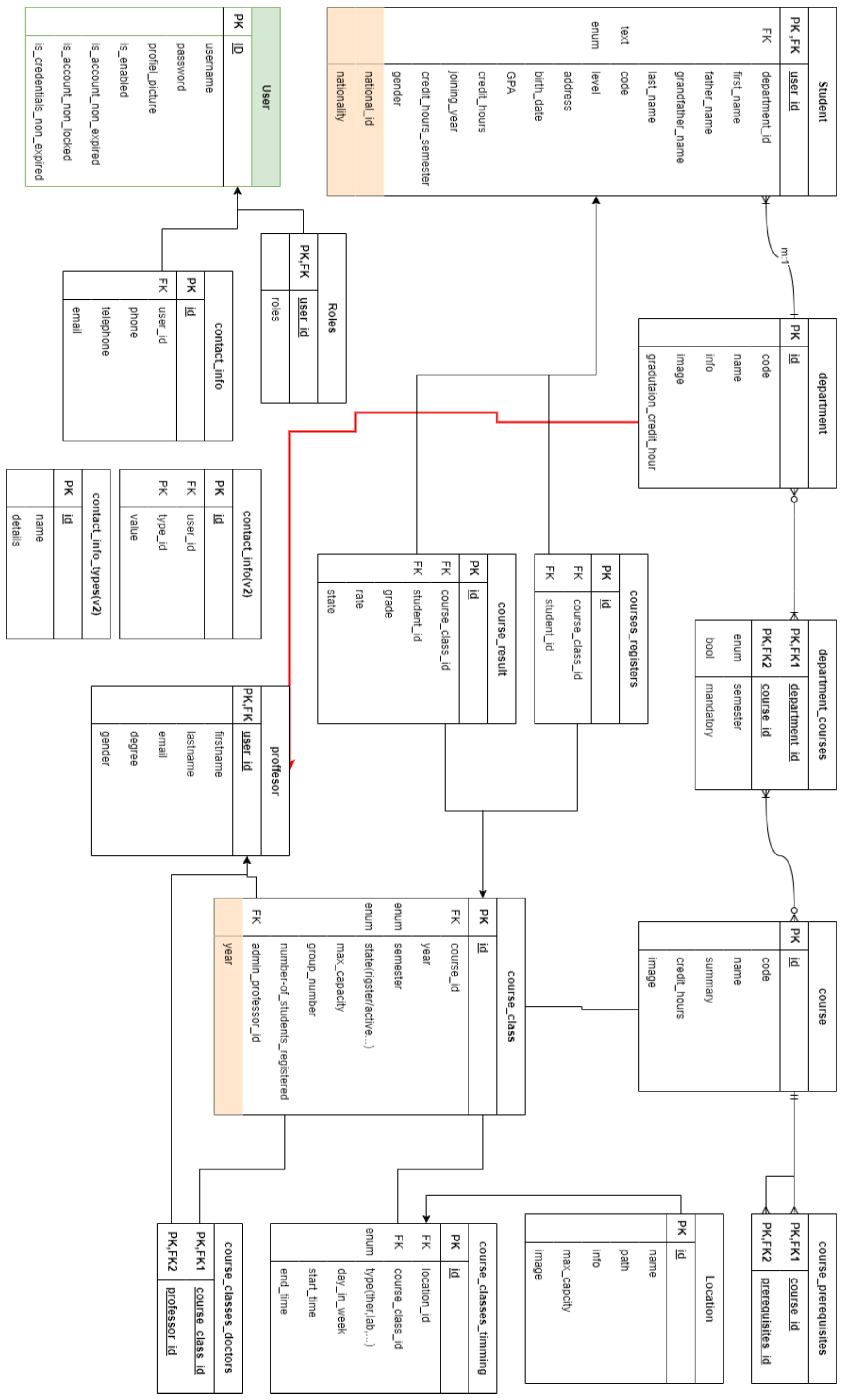


Figure 1: ERD Diagram

3.2 Entities and Attributes

Attribute	Description
user_id (PK)	Unique identifier for each user
username	User's login name
password	User's password
profile_picture	URL or path to the user's profile picture
is_enabled	Indicates whether the user's account is enabled
is_account_non_expired	Indicates whether the user's account is expired
is_account_non_locked	Indicates whether the user's account is locked
is_credentials_non_expired	Indicates whether the user's credentials are expired

Table 1: User

Attribute	Description
user_id (PK, FK)	Foreign key referencing `user.user_id`
role	Role assigned to the user (e.g., admin, student, professor)

Table 2: Role

Attribute	Description
user_id (PK, FK)	Foreign key referencing `user.user_id`
type_id (PK, FK)	Foreign key referencing `contact_info_types.type_id`
value	Contact information value (e.g., phone number, email)

Table 3: Contact info

Attribute	Description
type_id (PK)	Unique identifier for each contact info type
name	Name of the contact info type (e.g., phone, email)
details	Additional details about the contact info type

Table 4: Contact info type

Attribute	**Description**
student_id (PK, FK)	Foreign key referencing `user.user_id`
department_id (FK)	Foreign key referencing `department.department_id`
first_name	Student's first name
last_name	Student's last name
father_name	Student's father's name
grandfather_name	Student's grandfather's name
national_id	Student's national ID
nationality	Student's nationality
gender	Student's gender
birth_date	Student's birth date
address	Student's address
GPA	Student's GPA
credit_hours	Total credit hours completed by the student
joining_date	Date the student joined
semester	Current semester of the student

Table 5: Student

Attribute	**Description**
department_id (PK)	Unique identifier for each department
name	Name of the department
info	Additional information about the department
graduation_credit_hour	Credit hours required for graduation from the department

Table 6: department

Attribute	**Description**
department_id (PK, FK)	Foreign key referencing `department.department_id`
course_id (PK, FK)	Foreign key referencing `course.course_id`
is_mandatory	Indicates whether the course is mandatory for the department

Table 7: Department_course

Attribute	**Description**
course_id (PK)	Unique identifier for each course
code	Course code
name	Course name
summary	Course summary
credit_hours	Credit hours for the course
image	URL or path to the course image

Table 8: Course

.

Attribute	Description
course_id (PK, FK)	Foreign key referencing `course.course_id`
prerequisite_id (PK, FK)	Foreign key referencing `course.course_id`

Table 9: Course_Prerequisites

Attribute	Description
course_class_id (PK)	Unique identifier for each course class
course_id (FK)	Foreign key referencing `course.course_id`
year	Year the course class is offered
semester	Semester the course class is offered
status	Status of the course class
type	Type of the course class
location_id (FK)	Foreign key referencing `location.location_id`
admin_professor_id (FK)	Foreign key referencing `professor.professor_id`
max_capacity	Maximum capacity of the course class
group_number	Group number of the course class
students_registered	Number of students registered in the course class

Table 10: Course_Class

Attribute	Description
professor_id (PK, FK)	Foreign key referencing `user.user_id`
first_name	Professor's first name
last_name	Professor's last name
degree	Professor's degree
email	Professor's email
gender	Professor's gender

Table 11: Professor

Attribute	Description
course_class_id (PK, FK)	Foreign key referencing `course_class.course_class_id`
professor_id (PK, FK)	Foreign key referencing `professor.professor_id`

Table 12: Course_Classes_Doctors

Attribute	Description
course_result_id (PK)	Unique identifier for each course result
student_id (FK)	Foreign key referencing `student.student_id`
course_class_id (FK)	Foreign key referencing `course_class.course_class_id`
grade	Grade received by the student
rate	Rate of the course
state	State of the course result

Table 13: Course_Result

.

Attribute	**Description**
course_register_id (PK)	Unique identifier for each course registration
student_id (FK)	Foreign key referencing `student.student_id`
course_class_id (FK)	Foreign key referencing `course_class.course_class_id`
semester	Semester of the course registration

Table 14: Courses_Registers

Attribute	**Description**
location_id (PK)	Unique identifier for each location
name	Name of the location
info	Additional information about the location
max_capacity	Maximum capacity of the location
image	URL or path to the location image

Table 15: Location

Attribute	**Description**
course_class_timing_id (PK)	Unique identifier for each course class timing
course_class_id (FK)	Foreign key referencing `course_class.course_class_id`
location_id (FK)	Foreign key referencing `location.location_id`
day_in_week	Day of the week
start_time	Start time of the class
end_time	End time of the class

Table 16: Course_Classes_Timing

.

3.3 Relationships

- A **User** can have multiple **Roles**.
- A **User** can have multiple **Contact_Info** entries, and each contact info is associated with a **Contact_Info_Type**.
- A **Student** is a **User** with additional attributes and belongs to a **Department**.
- A **Department** can offer multiple **Courses**.
- A **Course** can have multiple **Course_Prerequisites**.
- A **Course_Class** is associated with a **Course** and has a location and an administering **Professor**.
- A **Professor** can teach multiple **Course_Classes**, and each **Course_Class** can have multiple **Professors**.
- A **Student** can register for multiple **Course_Classes** and receive multiple **Course_Results**.
- **Course_Classes** have scheduled **Timings** at specific **Locations**.

4 System Architecture

Importance of System Architecture :

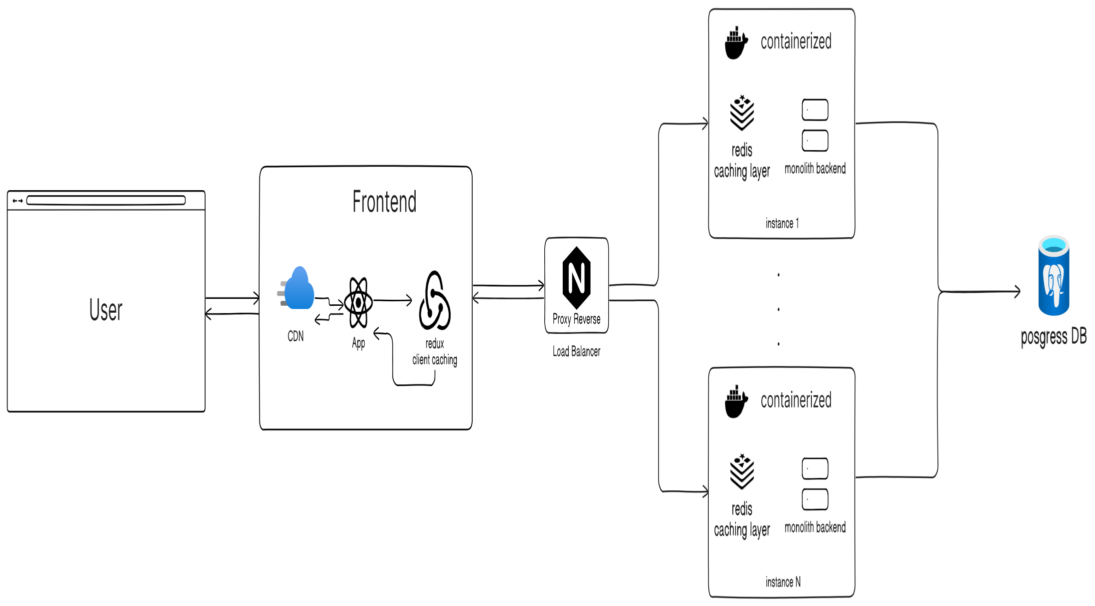
- **Foundation for Development**:
  - **Guidance**: Provides a blueprint for developers, ensuring consistency and alignment with the system's requirements and objectives.
  - **Organization**: Helps in organizing and structuring the development process, making it easier to manage and coordinate efforts.
- **Scalability and Flexibility**:
  - **Scalability**: A well-designed architecture supports scaling the system to handle increased loads and user demands.
  - **Flexibility**: Enables the system to adapt to changes, such as new features, technologies, or business requirements.
- **Performance and Reliability**:
  - **Optimization**: Facilitates the optimization of system performance by clearly defining how components interact and process data.
  - **Reliability**: Ensures the system is robust and reliable, with defined mechanisms for handling failures and ensuring uptime.
- **Security**:
  - **Protection**: Helps in designing security measures at various levels, such as data encryption, access control, and secure communication.
  - **Compliance**: Assists in meeting regulatory and compliance requirements by structuring the system to adhere to standards.
- **Maintainability and Manageability**:
  - **Easier Maintenance**: A clear architecture makes it easier to identify and fix issues, perform updates, and manage system components.
  - **Documentation**: Serves as documentation for understanding system behavior and dependencies, aiding in troubleshooting and knowledge transfer.
- **Interoperability**:
  - **Integration**: Facilitates the integration with other systems and services, ensuring smooth data exchange and interaction.
  - **Standards**: Promotes the use of standards and best practices, enhancing compatibility and reducing integration complexities.

- **Cost Efficiency**:
  - **Resource Management**: Helps in efficient resource allocation and management, reducing operational costs.
  - **Future Planning**: Assists in planning for future expansions and upgrades, minimizing the cost of changes and rework.

High-Level Overview:

- **Client (Front end)**: End-user devices that interact with the application via web browsers .
- **APIs**: RESTful APIs for communication between frontend, backend, and external services.
- **Nginx Server**: Serves as a reverse proxy and load balancer
- **Security**: OAuth 2.0 for authentication, TLS for data encryption in transit, and role-based access control.
- **Application Server (backend)**: Hosts the monolithic application for business logic.
- **Database Server**: Stores the application data.

Figure 2: Architectural Design



4.1 Architectural Design

We used Monolithic Architecture Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high level overview of how responsibilities of the system were partitioned and then assigned to subsystems.
While monolithic architecture presents certain challenges, its simplicity and performance benefits make it suitable for many applications. Using Nginx as a reverse proxy,
load balancer, and caching layer can significantly enhance the
capabilities and performance of a monolithic application.
Proper configuration and deployment of Nginx ensure that the application can handle high traffic, provide secure communication,
and offer quick response times, thereby improving the overall user experience.

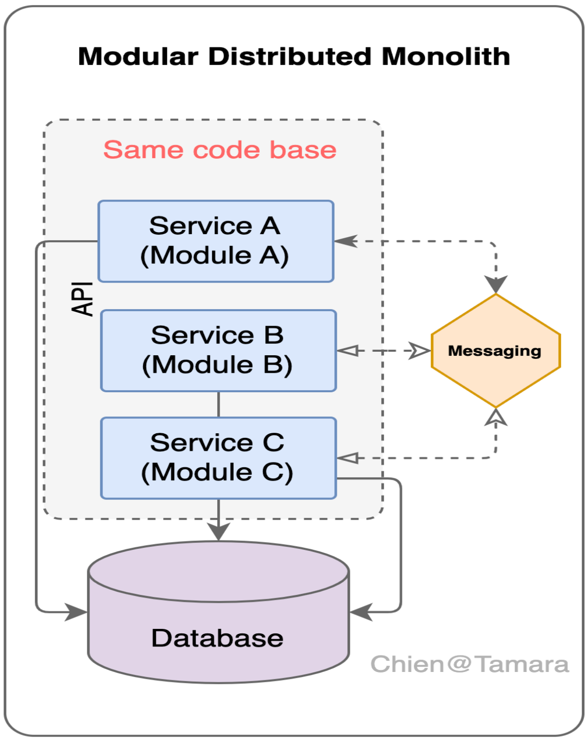
4.1.1 Advantages of Monolithic Architecture

- **Simplicity**: Easier to develop, test, and deploy in the early stages.
- **Performance**: Direct calls within the same process can be faster than inter-process communication in a Monolithic architecture.
- **Deployment**: Only one executable or deployment unit needs to be managed.
- **Development Tools**: Comprehensive tooling support for monolithic applications, including robust Integrated Development Environments (IDEs) and debugging tools.

4.1.2 Challenges with Monolithic Architecture

- **Scalability**: Difficult to scale individual components independently.
- **Maintainability**: As the codebase grows, making changes becomes more complex and risk-prone.
- **Flexibility**: Hard to adopt new technologies gradually since the entire system is tightly coupled.
- **Reliability**: A single bug or issue can bring down the entire application.

Figure 3: Architectural Design 2



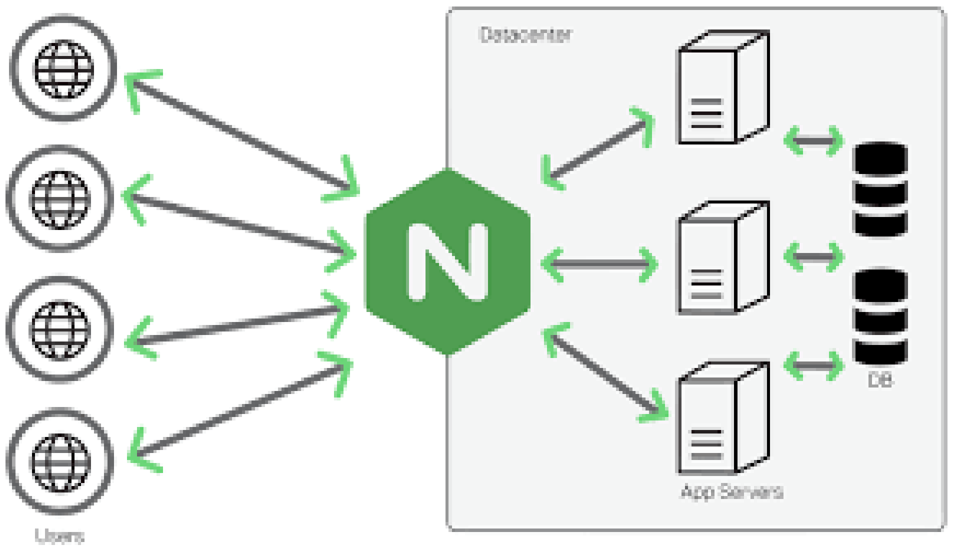
4.2 Decomposition Description

let's dig deep into the every point in architecture to understand it more carefully

4.2.1 Key Roles of Nginx in a Monolithic Architecture

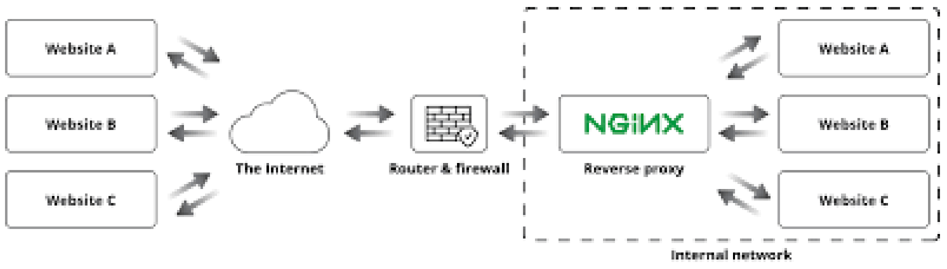
- **Load Balancing**:
  - Distribute incoming traffic across multiple instances of the monolithic application.
  - Improve resource utilization, reduce response times, and ensure no single server is overwhelmed.
  - Nginx supports various load balancing algorithms, such as round-robin, least connections, and IP hash.

Figure 4: Loadbalancing



- **Reverse Proxy**:
  - Acts as an intermediary for requests from clients seeking resources from the monolithic application.
  - Provides caching, compression, and SSL termination, improving performance and security.
  - Simplifies client-server communication by abstracting the backend servers.
- **Caching**:
  - Store copies of frequently accessed resources, reducing load on the application servers and improving response times.
  - Nginx's caching capabilities can significantly enhance the performance of read-heavy applications.
- **Security**:
  - Protects the application from various types of attacks by filtering and managing incoming traffic.
  - Implements security measures such as rate limiting, IP whitelisting/blacklisting, and web application firewall (WAF) integrations.
- **Static Content Serving**:
  - Efficiently serves static content (e.g., images, CSS, JavaScript) directly from the Nginx server, reducing the load on the application server.

Figure 5: Loadbalancing



4.2.2 Application Server (backend)

for backend we used Spring Boot for a popular framework for building microservices and stand-alone, production-grade Spring-based applications. It is designed to simplify the development process by providing default configurations and a wide array of features that enable developers to get started quickly.

Simplified Setup and Development

- **Auto-Configuration**: Spring Boot automatically configures your application based on the dependencies you include, reducing the need for manual setup.
- **Starter POMs**: Provides starter templates (starter POMs) for Maven and Gradle to simplify dependency management.
- **Embedded Servers**: Comes with embedded servers like Tomcat, Jetty, and Undertow, allowing you to run your application without needing an external application server.

Production-Ready Features

- **Metrics and Health Checks**: Built-in support for application metrics, health checks, and other monitoring tools.
- **Externalized Configuration**: Supports externalized configuration, making it easy to manage different environments (e.g., development, testing, production).
- **Spring Boot Actuator**: Provides production-ready features to help you monitor and manage your application, including endpoints for checking application health, metrics, and environment properties.

Microservices Support

- **Spring Cloud Integration**: Seamlessly integrates with Spring Cloud for building robust, scalable, and flexible microservices architectures.
- **API Gateway and Service Discovery**: Supports service discovery (e.g., Netflix Eureka) and API Gateway (e.g., Spring Cloud Gateway) for managing microservices communication.

Enhanced Developer Productivity

- **Rapid Application Development**: Provides a range of features and tools that accelerate the development process, including Spring Boot DevTools for live reload and H2 in-memory database for quick testing.
- **Comprehensive Documentation**: Offers extensive documentation and a large community, making it easier to find solutions and best practices.
- **Spring Initializr**: An online tool that helps you quickly generate a new Spring Boot project with the dependencies you need.

Security and Testing

- **Spring Security Integration**: Easy integration with Spring Security for authentication and authorization.

- **Testing Support**: Provides various testing utilities and annotations (e.g., @SpringBootTest) to simplify unit and integration testing.

Flexibility and Scalability

- **Modular Architecture**: Allows you to pick and choose the components you need, making it flexible and modular.
- **Scalability**: Suitable for building scalable applications, from small-scale to enterprise-level projects.

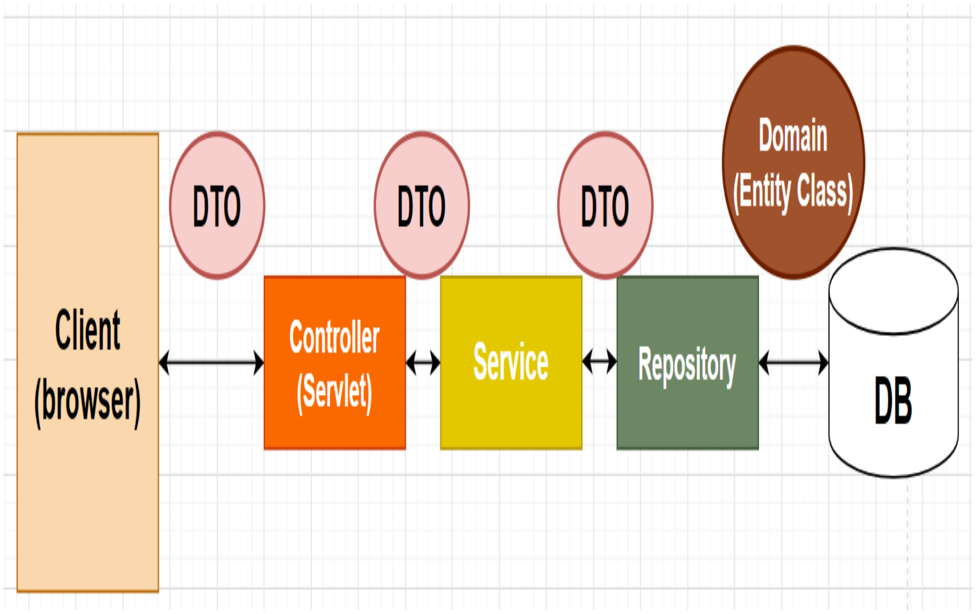
Model-View-Controller (MVC) Architecture in Spring Boot

- **Model** Represents the data and business logic of the application. In Spring Boot, the model typically consists of POJOs (Plain Old Java Objects) or entities that represent data stored in a database or used within the application.
- **View** Represents the presentation layer of the application, usually involving HTML/CSS templates, frontend frameworks, or even RESTful APIs that return data directly to clients.
- **Controller** Acts as an intermediary between the model and the view, handling user input and responding to it by invoking appropriate actions on the model and selecting the view to display. In Spring Boot, controllers are typically annotated with `@Controller` or `@RestController` annotations.
- **Services** In Spring Boot, services (annotated with `@Service`) are used to implement business logic. They encapsulate the application's business logic and orchestrate transactions between controllers and repositories. Services contain methods that manipulate data retrieved from repositories and prepare data to be displayed by the controllers or views.
- **Repositories** Repositories (annotated with `@Repository`) provide an abstraction layer on top of data access mechanisms (like JPA, JDBC, etc.). They interact directly with the database or any other external data source. Repositories manage the operations related to data storage, retrieval, and persistence.

Supporting the Architecture

- **Separation of Concerns**: Divides responsibilities among models, views, controllers, services, and repositories, making the codebase easier to understand, maintain, and test.
- **Dependency Injection (DI)**: Facilitates loose coupling between components. Controllers can inject services, and services can inject repositories, allowing for flexible and modular development.
- **Scalability and Maintainability**: Enhances scalability and maintainability of applications. It enables developers to extend and modify the application's behavior without impacting other parts of the system.

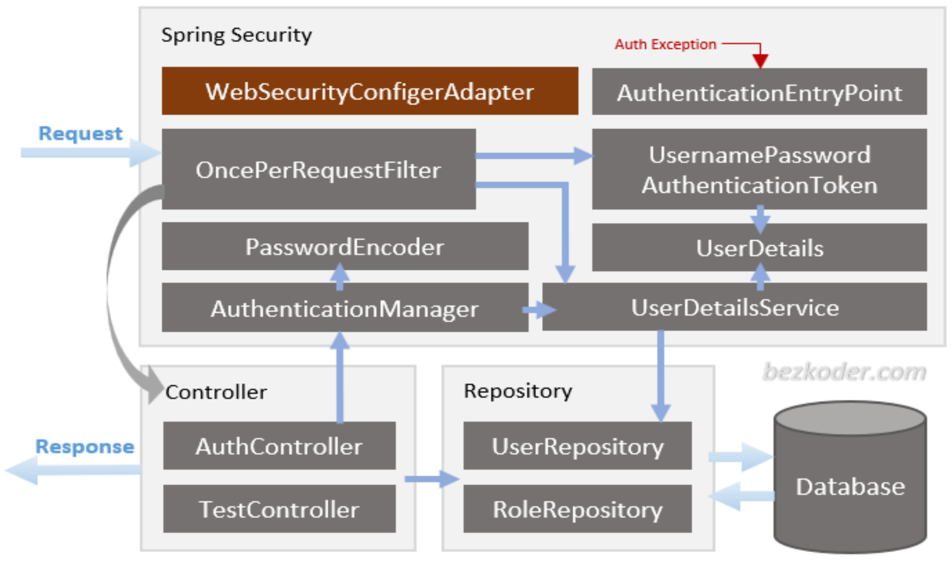
Figure 6: login architecture Design



4.2.3 Security

for Security we used Spring boot Security but implemented a design that fits our needs First let's describe Login Architecture **Login Architecture**

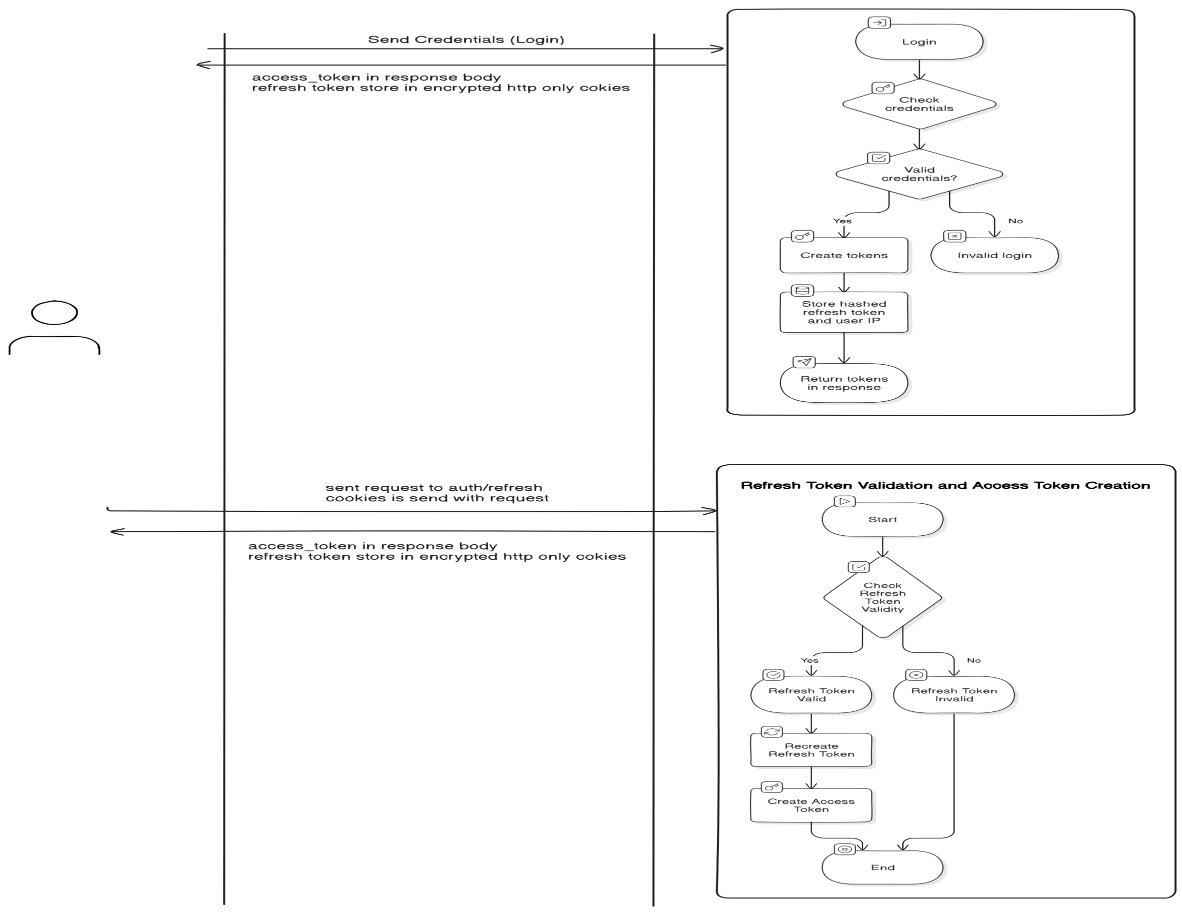
Figure 7: login architecture Design

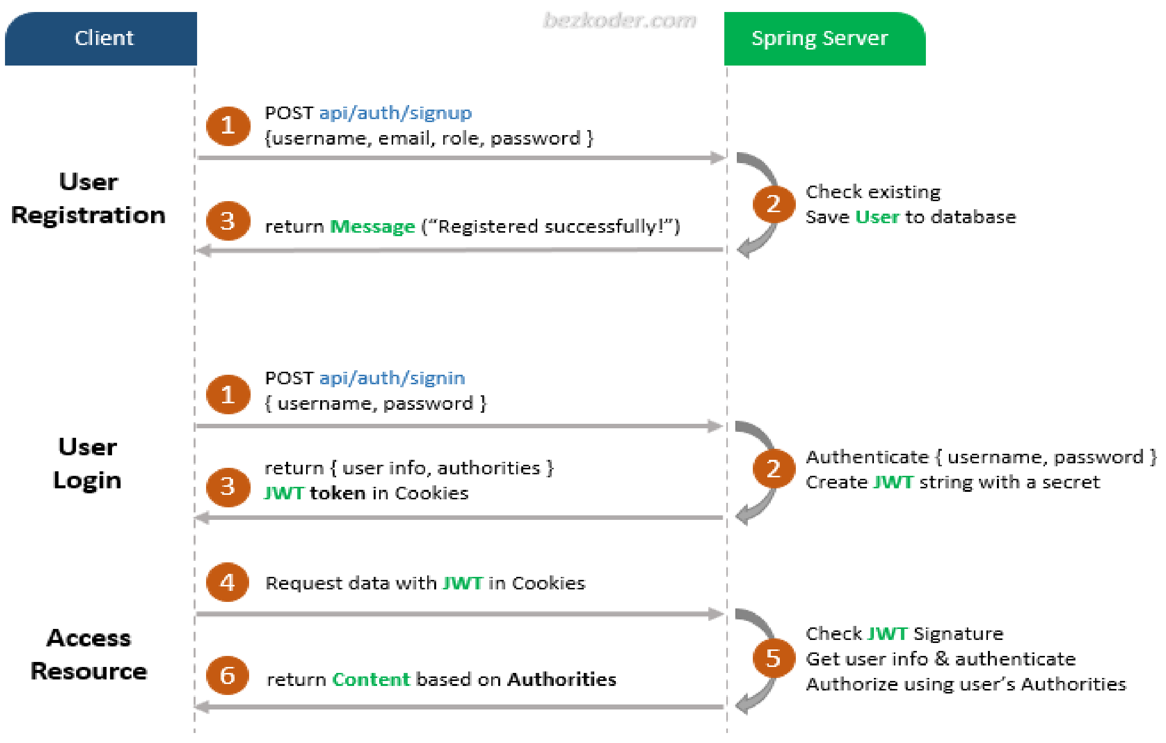


1. **Send Credentials**: The client sends login credentials to the server.
2. **Check Credentials**: The server verifies the provided credentials.
3. **Valid Credentials**
   - **Yes**: If the credentials are valid:
     (a) **Create Tokens**: The server generates an access token and a refresh token.
     (b) **Store Refresh Token**: The refresh token is hashed and stored along with the user's IP address.
     (c) **Return Tokens**: The access token is returned in the response body, and the refresh token is stored in encrypted HTTP-only cookies.
   - **No**: If the credentials are invalid, an invalid login response is returned.

1. **Send Refresh Request**: The client sends a request to refresh the tokens, including the refresh token stored in HTTP-only cookies.
2. **Check Refresh Token Validity**
   - **Valid**: If the refresh token is valid:
     (a) **Recreate Refresh Token**: A new refresh token is generated and stored.
     (b) **Create Access Token**: A new access token is generated.
     (c) **Return Tokens**: The new tokens are returned to the client, with the refresh token stored in encrypted HTTP-only cookies.
   - **Invalid**: If the refresh token is invalid, an appropriate error response is returned.

Authorization process .


Send Credentials (Login)
access_token in response body
refresh token store in encrypted http only cokies
Login
Check credentials
Valid credentials?
Yes
No
Create tokens
Invalid login
Store hashed refresh token and user IP
Return tokens in response
sent request to auth/refresh
cookies is send with request
access_token in response body
refresh token store in encrypted http only cokies
Refresh Token Validation and Access Token Creation
Start
Check Refresh Token Validity
Yes
No
Refresh Token Valid
Refresh Token Invalid
Recreate Refresh Token
Create Access Token
End


Client
bezkoder.com
Spring Server
User Registration
1 POST api/auth/signup
{username, email, role, password }
2 Check existing
Save User to database
3 return Message ("Registered successfully!")
User Login
1 POST api/auth/signin
{ username, password }
2 Authenticate { username, password }
Create JWT string with a secret
3 return { user info, authorities }
JWT token in Cookies
Access Resource
4 Request data with JWT in Cookies
5 Check JWT Signature
Get user info & authenticate
Authorize using user's Authorities
6 return Content based on Authorities

4.2.4 Database

Features of using PostgresSql

- **ACID Compliance**: Ensures reliable transactions through atomicity, consistency, isolation, and durability.
- **Full Text Search**: Built-in support for full text search capabilities.
- **PostGIS**: Extends PostgreSQL to handle geographic objects, making it an excellent choice for spatial databases.
- **JSON Support**: Efficient storage and querying of JSON data with JSONB.

Performance and Scalability

- **Indexes**: Supports various types of indexes (e.g., B-tree, Hash, GiST, GIN, BRIN) to enhance query performance.
- **Partitioning**: Table partitioning for handling large tables more efficiently.
- **Concurrency**: Excellent concurrency handling with Multi-Version Concurrency Control (MVCC).

Extensibility

- **Custom Functions**: Write custom functions in languages such as PL/pgSQL, PL/Tcl, PL/Perl, and PL/Python.
- **Extensions**: Extend PostgreSQL functionalities with extensions like PostGIS, pgRouting, and pgstatstatements.

Security

- **Authentication**: Supports multiple authentication methods, including LDAP, GSSAPI, SSPI, and certificate-based authentication.
- **Access Controls**: Granular access controls with role-based permissions.
- **Encryption**: Data encryption at rest and in transit.

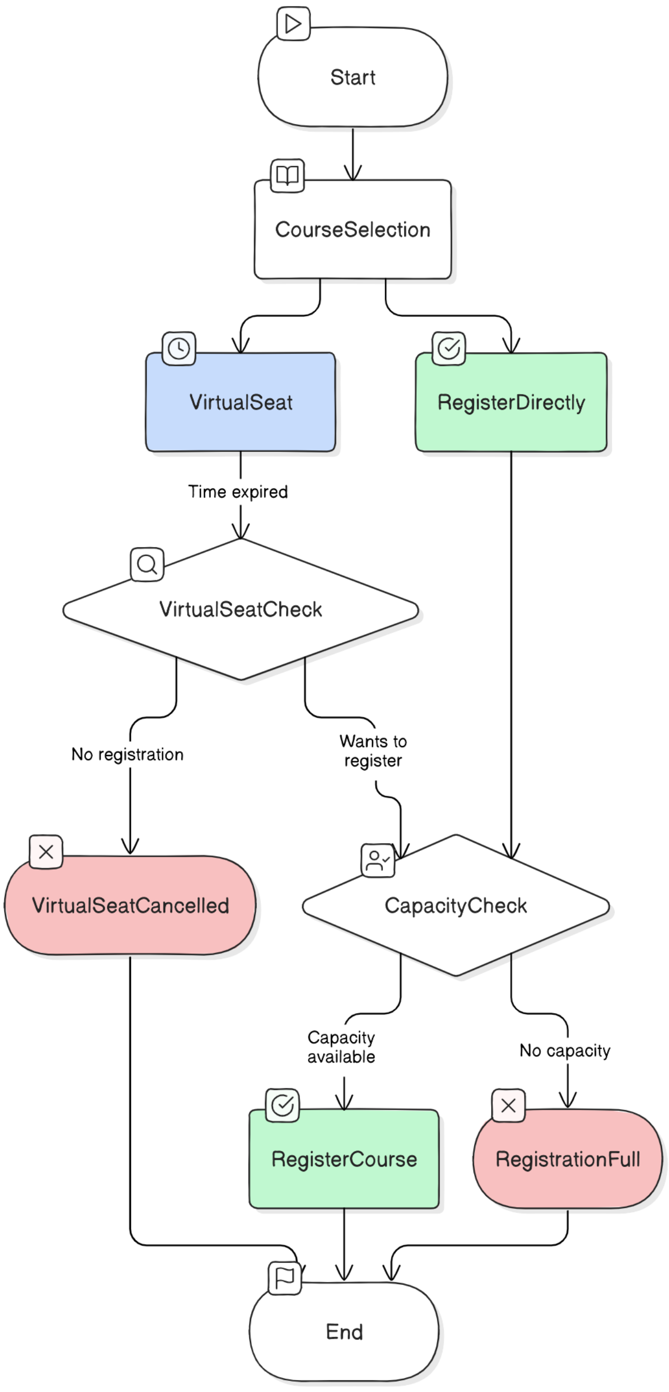
Community and Support

- **Active Community**: A vibrant, active community offering extensive documentation, tutorials, and forums.
- **Commercial Support**: Available from various vendors for those who need professional assistance.

5 Overcoming Challenges Throughout the Project

5.1 Course Registration

creating a live course registration system that caters to the needs of all students simultaneously demands careful planning, robust technology infrastructure, effective communication, and continuous improvement based on user feedback and system performance analysis.



5.2 Handling User Overload

5.2.1 Nginx

Nginx can be utilized not only as a web server but also as a powerful tool for handling high traffic loads and improving application security. It serves as a reverse proxy, which enables efficient load balancing across multiple servers, ensuring that your applications remain available and responsive even during periods of high user activity. Additionally, Nginx can function as a firewall, offering protection against malicious attacks and unauthorized access attempts.

5.2.2 Limiting User Sign-ins from Multiple IPs

To enhance security measures, limiting the number of simultaneous user sign-ins from multiple IP addresses can prevent unauthorized access and mitigate risks associated with credential stuffing attacks. Implementing this restriction involves tracking and validating user sessions based on IP addresses, thereby ensuring that each user is authenticated from a limited number of distinct IPs within a specified timeframe.

5.2.3 Caching Strategies

Implementing effective caching strategies can significantly improve application performance and handle user overload more efficiently. By caching frequently accessed content or data at various levels (server-side, client-side, CDN), you can reduce the load on your backend servers and enhance the overall responsiveness of your application. Nginx supports robust caching mechanisms that can be configured to store and serve static content or dynamic content with appropriate caching headers, optimizing resource utilization and user experience.

5.2.4 Rate Limiting

Implementing rate limiting mechanisms helps control the number of requests a user or IP address can make to your application within a defined period. This prevents abuse or overloading of your server resources by limiting the rate at which requests are processed. Nginx provides flexible configurations for rate limiting based on various criteria such as IP addresses, HTTP methods, or request URIs, allowing you to protect your application from excessive traffic spikes and ensure fair resource allocation among users.

5.3 Real-time Features: Chatting and Announcements

5.3.1 Chatting (Instant Messaging)

- **Real-time Messaging**: Allows users to send and receive messages instantly, creating a conversational experience without delays.
- **Presence Indicators**: Shows users' online or offline status, indicating their availability for communication.
- **Typing Indicators**: Shows when the other party is typing, providing real-time feedback during conversations.
- **Read Receipts**: Indicates when messages have been seen by the recipient, adding transparency to communication.

5.3.2 Announcements

- **Real-time Updates**: Allows administrators or designated users to broadcast messages or announcements to a large audience immediately.
- **Push Notifications**: Sends notifications to users' devices instantly, ensuring important announcements are received promptly.
- **Customization**: Allows for targeted announcements to specific groups or individuals based on roles or preferences.
- **Archiving**: Stores announcements for reference, ensuring users can review past messages as needed.

Implementation Considerations:

- **Scalability**: Ensure the system can handle concurrent users and messages efficiently.
- **Security**: Implement encryption and secure protocols to protect user data and messages.
- **User Experience**: Design intuitive interfaces that make it easy for users to chat and receive announcements.
- **Integration**: Consider integrating with other systems (like email or CRM) for seamless communication across platforms.

These features are essential for applications ranging from social media platforms and messaging apps to enterprise collaboration tools and educational platforms, enhancing real-time communication and information dissemination. Server-Sent Events (SSE) is a technology that enables servers to push real-time updates to clients over HTTP connections. It's particularly useful for implementing features like chatting and announcements in web applications.

Conceptual Overview

SSE operates on a unidirectional basis, where the server continuously sends updates to clients without requiring them to poll for new data. This makes it efficient for scenarios where real-time updates are needed, such as live messaging and announcements.

Key Components

- **Server Endpoint**: Establish a dedicated endpoint on the server to handle SSE connections. This endpoint is responsible for sending updates to clients.
- **Client Connection**: Clients (e.g., web browsers) initiate a connection to the SSE endpoint using standard HTTP methods (e.g., GET request).
- **Event Streaming**: Once connected, the server uses the SSE protocol to stream events to clients. Each event is typically formatted as a text/event-stream and sent with appropriate headers.
- **Real-Time Messaging**: Implement messaging features by sending chat messages as SSE events. Clients receive these messages instantly, enabling real-time communication.
- **Announcements**: Administrators or designated users can push announcements through the SSE connection. These announcements are broadcasted to all connected clients, ensuring timely delivery of important information.
- **Scalability**: Consider scalability factors such as server capacity and concurrent connections management to handle large numbers of clients effectively.
- **Integration**: SSE can be integrated into various web frameworks and technologies, allowing flexibility in implementation across different platforms.

Advantages

- **Efficiency**: SSE reduces unnecessary data transfer by only sending updates when new information is available.
- **Real-Time Updates**: Provides instant updates to clients, enhancing user experience for interactive features.
- **Simplicity**: Compared to other real-time technologies like WebSockets, SSE requires less overhead for both server and client implementations.

Considerations

- **Browser Compatibility**: Ensure clients support SSE; modern browsers generally provide robust support.
- **Error Handling**: Implement mechanisms to handle connection interruptions and server errors gracefully to maintain reliable communication.
- **Security**: Secure SSE connections using HTTPS to protect data privacy and prevent unauthorized access to transmitted information.

Implementing SSE for real-time features like chatting and announcements can significantly enhance the interactivity and responsiveness of web applications, making it a valuable tool for modern web development.

5.4 Continuous integration / Continuous development (CI/CD)

Continuous Integration (CI) and Continuous Deployment (CD) are pivotal practices in modern software development, particularly in large teams. This is why we used applied this using GitHub and Azure Cloud.

Importance in Big Teams

In large development teams, CI/CD offers several significant benefits:

1. **Automated Testing**: CI/CD pipelines automate the process of running tests whenever new code is committed. This ensures that bugs are caught early, preventing issues from escalating and affecting other team members or downstream processes.
2. **Consistency and Reliability**: By automating builds and deployments, CI/CD pipelines enforce consistency in how code is built, tested, and deployed across different environments. This reduces human error and ensures that everyone is working with the latest version of the software.
3. **Faster Feedback Loop**: Developers receive immediate feedback on the impact of their code changes. This allows them to address issues quickly and iterate faster, leading to shorter development cycles.
4. **Collaboration**: CI/CD encourages collaboration by providing visibility into the status of builds and deployments. Team members can easily track progress, review changes, and coordinate efforts seamlessly.

Using GitHub and Azure Cloud

GitHub and Azure Cloud offer robust tools to implement CI/CD pipelines:

- **GitHub**: Acts as a central repository where developers store and collaborate on code. It integrates with CI/CD tools like GitHub Actions or Jenkins, allowing teams to automate workflows directly from their code repositories.
- **Azure Cloud**: Provides a comprehensive suite of cloud services that support CI/CD, such as Azure Pipelines, Azure DevOps, and Azure Container Registry. These services enable teams to build, test, and deploy applications across various platforms and environments seamlessly.

In conclusion, CI/CD is indispensable in large teams as it enhances productivity, quality, and collaboration while reducing risks and time-to-market. By leveraging tools like GitHub and Azure Cloud, teams can establish efficient development workflows that scale with the complexity and demands of modern software projects.

This integration of CI/CD practices with GitHub and Azure Cloud forms a cornerstone in achieving agility, reliability, and innovation in software development at scale.

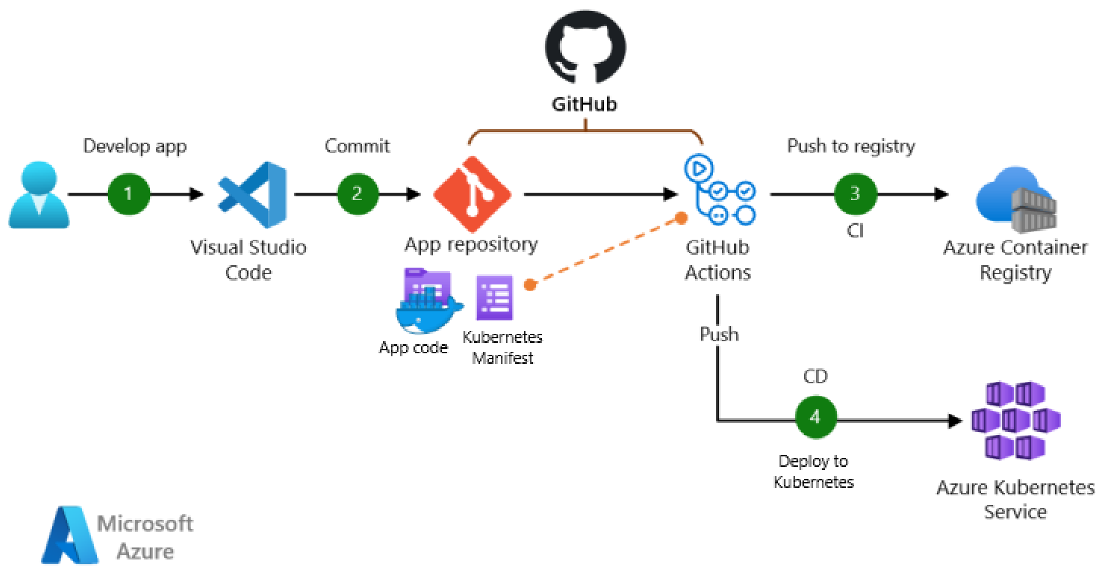


Figure 8: CI/CD

6 User Interface Design

6.1 Overview of User Interface

User Interface Design (UID) focuses on creating intuitive, efficient,
and user-friendly interfaces
that enhance the user experience. For a Course Registration and Faculty Communication System,
HID ensures that both students and faculty can easily navigate and use the system to accomplish their tasks effectively.
Here are the key principles and elements of Human Interface Design for this system: Understand User Needs:

- Conduct user research to understand the needs, preferences, and behaviors of students and faculty members
- Use personas and user journey maps to identify pain points and opportunities for improvement.

- be easy and clear for user to navigate and use the system
- Performance and Scalability is key points because there is a lot of LMSs like systems where it fail to handle the increase of users of over loading for this exact matter we take this into consideration thought the system Design

6.2 Designing UI Perspectives for Diverse User Needs

Consistent Navigation: Ensure navigation menus and icons maintain consistency across all user perspectives to enhance usability.

Role-Based Access: Implement role-based access controls to ensure users only see relevant information and tools based on their roles.

Customization and Personalization: Allow users to personalize their dashboards and interface settings to align with individual preferences and workflows.

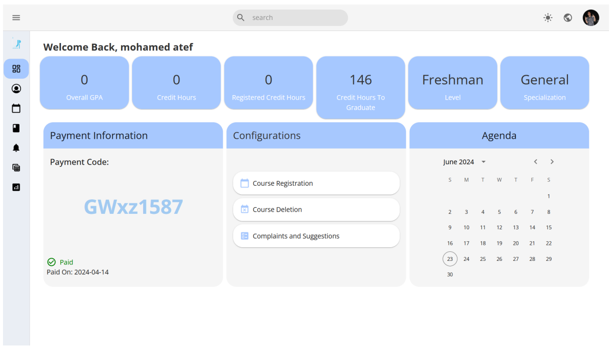
Responsive Design: Ensure the UI is responsive and accessible across various devices, including desktops, tablets, and smartphones.

By incorporating these tailored perspectives into the UI design, the system effectively meets the diverse needs of students, faculty, and administrative staff, thereby enhancing overall user satisfaction and operational efficiency.

6.2.1 Student Perspective

Dashboard:

- **Overview:** Provides a snapshot of upcoming classes, deadlines, and notifications.
- **Quick Actions:** Includes intuitive buttons for tasks like course registration, dropping courses, and viewing grades.


Welcome Back, mohamed atef

0 Overall GPA
0 Credit Hours
0 Registered Credit Hours
146 Credit Hours To Graduate
Freshman Level
General Specialization

Payment Information
Payment Code:
GWxz1587
Paid
Paid On: 2024-04-14

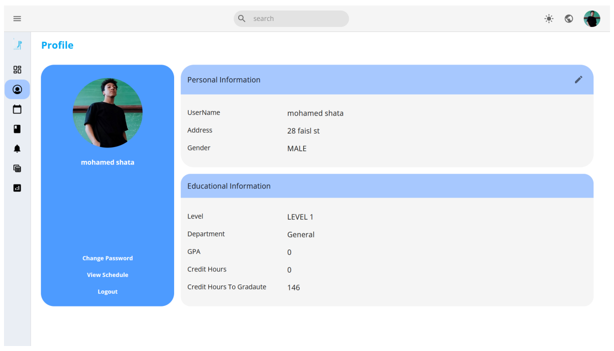
Figure 9: Student Dashboard

Course Registration:

- **Search and Filter:** Enables easy course discovery by keywords, departments, and schedules.
- **Course Details:** Offers comprehensive information on course content, instructors, prerequisites, and availability.
- **Enrollment Process:** Guides students step-by-step through enrollment, offering real-time feedback and conflict resolution.

Profile Management:

- **Personal Information:** Allows updates to personal details and contact information.
- **Academic Records:** Displays grades, enrolled courses, and transcripts.



Personal Information

UserName	mohamed shata
Address	28 faisl st
Gender	MALE

Educational Information

Level	LEVEL 1
Department	General
GPA	0
Credit Hours	0
Credit Hours To Gradaute	146

Figure 10: profile

6.3 Faculty Perspective

Dashboard:

- **Overview:** Displays class schedules, upcoming meetings, and recent communications.
- **Announcements:** Facilitates posting and viewing course-related announcements and faculty activities.

Course Management:

- **Course Details:** Provides interfaces for adding/editing course information, materials, and assignments.
- **Student Roster:** Manages enrolled students, tracks attendance, and inputs grades.

Communication:

- **Messaging System:** Supports direct and group messaging with students and peers.
- **Document Sharing:** Facilitates uploading and sharing of course materials, syllabi, and research papers.

6.4 Administrative Staff Perspective

Dashboard:

- **Overview:** Highlights institutional announcements, system alerts, and key performance indicators.
- **Administrative Tools:** Provides quick access to manage course offerings, schedules, and faculty assignments.

Student Management:

- **Enrollment Reports:** Generates reports on student enrollment, waitlists, and drop rates.
- **User Management:** Manages student and faculty accounts, permissions, and roles.

System Configuration:

- **Settings:** Configures system settings, academic calendar, and notification preferences.

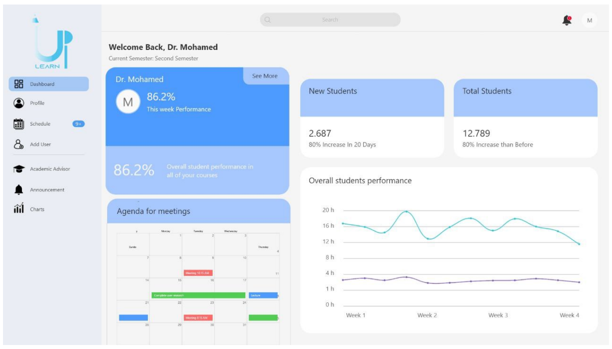


Figure 11: Admin Dashboard

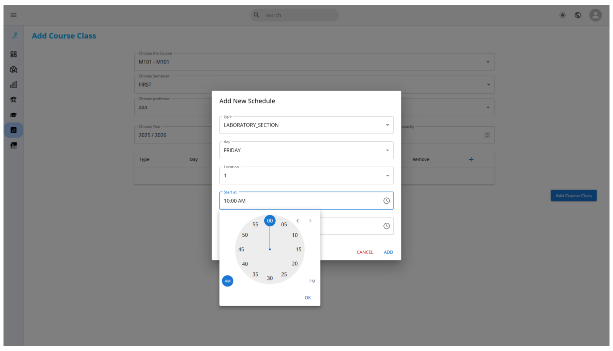


Figure 12: Admin classes timing add

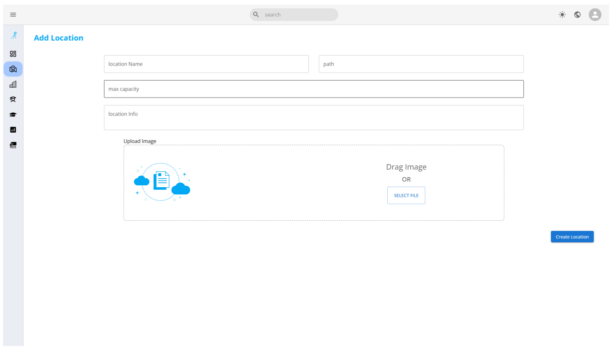


Figure 13: add location

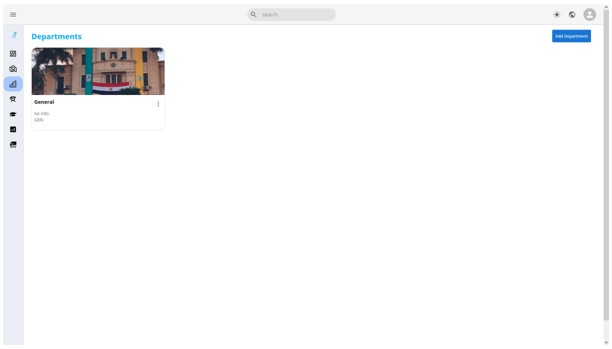


Figure 14: departments

6.5 Screen Objects and Actions

6.5.1 Notification for Announcement

Uplearn incorporates a robust notification system designed to facilitate effective communication between professors and students. One of the key features of Uplearn is the ability to send announcements through notifications. This feature ensures that important information is promptly delivered to the intended recipients, enhancing the overall communication efficiency within the platform.

6.5.2 Targeted Notifications

Professors can send notifications specifically to individual students or groups of students registered in a particular course on Uplearn. This targeted approach allows professors to tailor their messages to the relevant audience, ensuring that the information reaches those who need it most. For instance, if a professor needs to send a reminder about an assignment deadline, they can choose to notify only the students enrolled in that course.

6.5.3 Course-Based Group Notifications

In addition to individual notifications, Uplearn allows professors to send announcements to all students registered in a particular course. This feature is particularly useful for disseminating information that is relevant to an entire class, such as changes in the course schedule, upcoming exams, or new learning resources. By using course-based group notifications, professors can ensure that all students in a course receive the same information simultaneously.

6.5.4 System-Wide Notifications

The notification system in Uplearn also supports sending announcements to all students across different courses. This feature is essential for communicating general information that affects the entire student body, such as campus-wide events, administrative updates, or emergency alerts. System-wide notifications ensure that critical information reaches every student, regardless of their course enrollment.

6.5.5 Notification Delivery Methods

Notifications can be delivered through various methods on Uplearn to ensure that students receive the information promptly. These methods include:

6.5.6 Notification Delivery Methods

Notifications can be delivered through various methods on Uplearn to ensure that students receive the information promptly. These methods include:

- **Email Notifications:** Announcements can be sent directly to the students' email addresses, allowing them to receive updates even when they are not logged into Uplearn.
- **In-System Notifications:** Notifications can be displayed within the Uplearn interface, providing a convenient way for students to access announcements while they are using the platform.

- **Push Notifications:** For students using the Uplearn mobile application, push notifications can be sent to their devices, ensuring instant delivery of important messages.

6.6 Chat System

Uplearn incorporates a real-time chat feature designed to facilitate seamless communication between students and professors. This chat system ensures fast response times and is accessible via a user-friendly interface on both desktop screens and mobile devices, enhancing accessibility and usability across different platforms.

6.6.1 Real-Time Communication

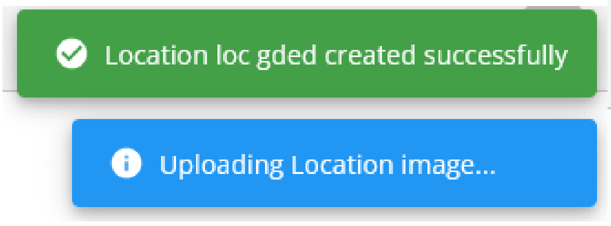
The chat feature in Uplearn enables students and professors to communicate in real-time. This capability allows for immediate interaction, fostering quicker resolution of queries and enhancing collaborative learning experiences within the platform.

6.6.2 User-Friendly Interface

The chat interface on Uplearn is designed to be intuitive and easy to access. Whether accessed through desktop screens or mobile devices, students and professors can navigate the chat effortlessly, ensuring that communication remains smooth and efficient regardless of the device used.

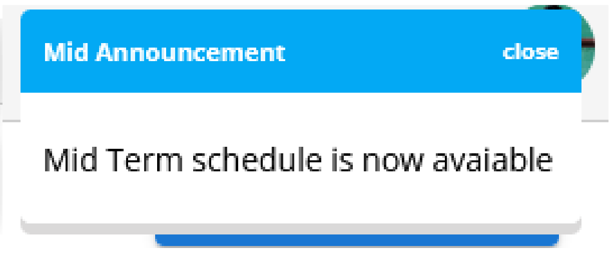
6.6.3 Responsive Design

To accommodate various screen sizes and devices, the chat interface in Uplearn employs responsive design principles. This responsiveness ensures that the chat interface adapts seamlessly to different screen resolutions, providing an optimal viewing and interaction experience on both desktops and mobile devices.


Location loc gded created successfully

Uploading Location image...

Figure 15: notification


Mid Announcement close

Mid Term schedule is now avaiable

Figure 16: notification for announcement