# System Specification Document

## **Project Title**: AI Learning


**Ahmed Ibrahim Zeb** (Enrollment No: 01-131212-006)

**Ahsan Khan** (Enrollment No: 01-131212-007)

**Supervisor**: ENGR. SUBAS BILAL

# Table of Contents

## System Requirements Specification (SRS)

## System Design Specification (SDS)

# Software Requirements Specification

# 1. Introduction

## 1.1 Purpose

The purpose of this SRS is to document the software requirements for the AI Learning Application. This document defines the functional and non-functional requirements, detailing the scope of the system to be developed. This document will present a detailed description of the functionality of application including the purpose and features of the system.

## 1.2 Document Conventions

This document follows IEEE standards for software requirements specification. The format for headings is as follows:

- Major headings are in bold 16pt font, and concurrent headings in bold 14pt. font with font type ‘Times New Roman’.
- The format for text is as follows: font type ‘Times New Roman’ and font size 12pt.

## 1.3 Product Scope

The AI Learning platform is designed to address gaps in current educational systems by providing personalized learning paths, adaptive quizzes, and location-based recommendations. Using TinyLlama for AI-powered text generation, the platform dynamically adapts to individual user needs. The app offers a personalized learning experience that includes:

- AI-generated weekly learning roadmaps.
- Aggregation of online courses.
- Quizzes with adaptive feedback.
- Recommendations for nearby educational institutions.

## 1.4 References


1. Zhang, P., Zeng, G., Wang, T., and Lu, W. (2024). Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.
2. Li, Dehui. (2017). An intelligent and effective e-learning system that provides tailored lessons to students.
3. Singrodia, V., Mitra, A., and Paul, S. (2019). A review on web scraping and its applications. *IEEE*.

4. Hu, S., and Dai, T. (2013). Online map application development using Google Maps API, SQL database, and ASP.NET.
5. Hendler, J. (2014). Data integration for heterogeneous datasets. *Big Data, 2(4)*, 205–215.

---

# 2. Overall Description

## 2.1 Product Perspective

The **AI Learning** is a next-generation app-based application designed to personalize and enhance the learning experience for individuals through advanced AI and machine learning technologies. Its modular design ensures scalability, flexibility, and the ability to cater to diverse user needs. The system integrates multiple technologies, including web technologies for seamless interaction, AI models for intelligent content generation, and backend services for robust data handling.

**Key Elements of the Product Perspective**:

- **Integration with Existing Educational Ecosystem**:
  - The application serves as a complementary tool to existing online education platforms by providing personalized learning roadmaps, generating lectures dynamically, and recommending relevant courses.
  - It also identifies nearby educational institutes using mapping APIs, ensuring a hybrid approach between digital and physical learning opportunities.
- **Scalable AI Integration**:
  - **AI Models**: The system employs state-of-the-art AI models like **TinyLlama** or **OpenAI GPT APIs** for text generation tasks, enabling tailored lecture creation, quiz generation, and adaptive learning roadmaps.
  - **Dynamic Adaptation**: The AI dynamically adjusts to user progress, providing real-time updates to learning roadmaps and personalized feedback on quizzes.
- **Seamless Frontend-Backend Communication**:
  - The frontend, built with modern web technologies like Flutter, ensures a responsive and interactive user experience.
  - The backend, implemented using Flask or Django, facilitates secure API communication with AI services, databases, and third-party APIs.
- **API Integrations**:
  - **Google Maps API**: For identifying educational institutions nearby based on user preferences.
  - **External Learning Platforms**: API integration with platforms like Coursera and edX to recommend courses matching user interests.
- **Database and Data Management**:

    - Uses **PostgreSQL** for managing user profiles, tracking progress, and storing data for roadmaps, quizzes, and recommendations.
    - Ensures efficient handling of large datasets generated by AI interactions and external API calls.
- **User-Centric Design**:
    - Focused on providing an intuitive interface that simplifies access to personalized learning tools.
    - Incorporates user feedback loops for continuous improvement in recommendations and AI-generated content.
- **Modular and Extensible Architecture**:
    - Each feature is designed as a self-contained module, allowing for easy addition or modification of components, such as integrating new APIs or upgrading AI models.
- **Target Audience**:
    - **Primary Users**: Students, professionals, and self-learners looking to enhance their skills in a personalized manner.
    - **Secondary Users**: Educational administrators and instructors who can use the tool to monitor student progress and provide additional support.

## 2.2 Product Functions

The core features include:

1. Dynamic weekly learning roadmaps.
2. AI-generated quizzes with adaptive difficulty.
3. Aggregation of online courses and content summaries.
4. Location-based recommendations for offline learning.

## 2.3 User Classes and Characteristics

- **Students**: Primary users who require structured, personalized learning.
- **Educators**: Users monitoring performance and recommending resources.
- **Lifelong Learners**: Individuals seeking to upgrade their skills.

## 2.4 Operating Environment

### 2.4.1 Operating System and Versions

- **Mobile:** Android 8.0+ and iOS 12.0+

### 2.4.2 Software Components

- **Frontend**: Flutter

- **Backend**: Flask/Django with PostgreSQL.
- **AI Models**: TinyLlama or OpenAI GPT APIs
- **AI Model Hosting**: Google Colab

## 2.5 Design and Implementation Constraints

- Model fine-tuning requires GPU resources.
- Web scraping might face restrictions from some platforms.

## 2.6 User Documentation

The app will include:

1. Interactive tutorials and onboarding guides.
2. FAQs are within the help menu.
3. Printable guides and reports.

## 2.7 Assumptions and Dependencies

### 2.7.1 Assumptions

- **User Accessibility and Technology Proficiency**:
    - Users have access to devices (e.g., smartphones, tablets, or desktops) with internet connectivity.
    - Users possess basic technical skills to navigate the application and interact with its features.
- **Reliable Third-Party Services**:
    - APIs (e.g., Google Maps API, GPT APIs) and other external services will remain accessible and functional.
    - Educational platforms like Coursera, edX, or others will provide consistent data feeds for course recommendations.
- **Sufficient Training Data**:
    - The AI models are assumed to have been fine-tuned with high-quality, domain-specific datasets for effective performance in tasks like lecture generation and quiz creation.
- **Stable Operating Environment**:
    - Hosting environments (e.g., cloud servers or local servers for AI models) will provide the necessary computational resources.

    - Software libraries and dependencies used in development are compatible and stable.

- **User Engagement**:
    - Users will input accurate preferences (e.g., topics of interest, time availability) for generating personalized learning roadmaps.
    - Users will actively engage with quizzes and lectures to allow the system to adapt effectively.
- **Educational Ecosystem Stability**:
    - The availability of online courses, resources, and institutional data remains consistent for integrations and recommendations.

### 2.7.2 Dependencies

- **Technology and Infrastructure**:

  **Frontend Technologies**: The system depends on modern web technologies like React or Flutter for building user interfaces.

  **Backend Frameworks**: Frameworks like Flask or Django are critical for managing API calls and data flow.

  **Database**: PostgreSQL is used for storing user data, course information, and progress metrics.

- **Third-Party APIs and Integrations**:

  **Google Maps API**: For identifying nearby institutions and mapping user preferences to physical locations.

  **GPT/TinyLlama APIs**: For generating lectures, quizzes, and learning roadmaps.

  **External Educational Platforms**: For retrieving data on available courses and resources.

- **AI Hosting and Computational Resources**:

  The AI model hosting (e.g., Google Colab or local GPU environments) is essential for running text generation and other machine learning tasks.

  Computational performance depends on stable hardware and cloud services.

- **Security and Authentication Services**:

  OAuth 2.0 or similar authentication protocols are critical for secure user login and data access.

Encryption services (e.g., AES-256) are required for protecting sensitive user information.

- **Regulatory Compliance**:

  Adherence to data privacy laws like GDPR or CCPA for managing user data and ensuring transparency.

  Dependence on educational platforms to comply with intellectual property guidelines.

- **Development and Maintenance Team**:

  Availability of skilled developers, AI engineers, and support staff for system updates, bug fixes, and feature enhancements.

- **User Feedback Mechanism**:

  The success of personalized recommendations and adaptive quizzes depends on collecting and processing feedback from users to improve the AI models and system logic.

---

# 3. External Interface Requirements

## 3.1 User Interfaces

- **Dashboard:** Displays roadmap, lectures, learning progress and quiz results.
- **Quiz Interface:** User-friendly quiz environment with real-time evaluation.
- **Course Viewer:** Course browsing and institutional recommendations.

## 3.2 Hardware Interfaces

**Minimum Device Requirements**:

- Mobile: 4GB RAM, 1.5GHz processor.

## 3.3 Software Interfaces

- **Development Tool:** Visual Studio
- **Front-End:** Flutter
- **Back-End:** Flask/Django with PostgreSQL.
- **REST API:** Communication between backend and frontend.
- **PostgreSQL:** Storing user profiles and progress data.

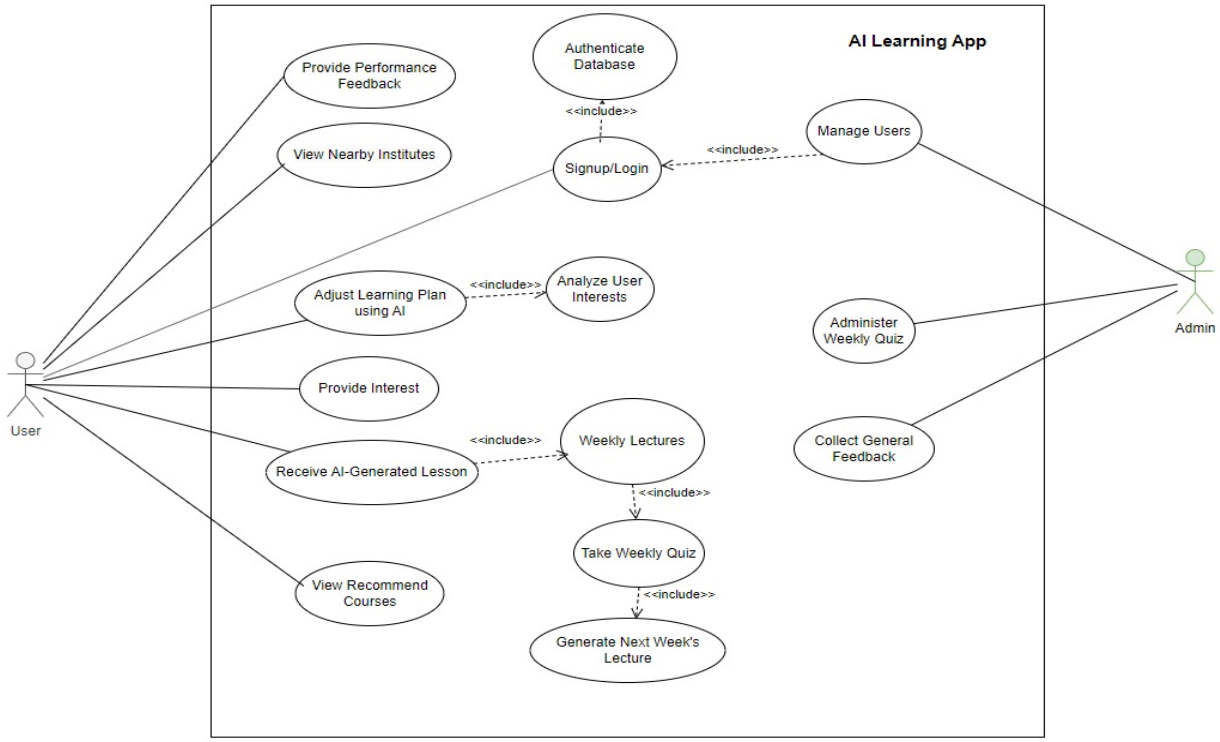
## 3.4 Communication Interfaces

- Secure HTTPS for data transfer.
- Notifications for reminders and roadmap updates.

---

# 4. System Features

## 4.1 Use Case




## 4.2 User Management

### 4.2.1 User Login

| **Use Case ID** | **UC1.1** |
|---|---|
| **Use Case Name** | Login |
| **Actor(s)** | User (learner, administrator) |
| **Pre-Conditions** | User opens the application and navigates to the login screen. |

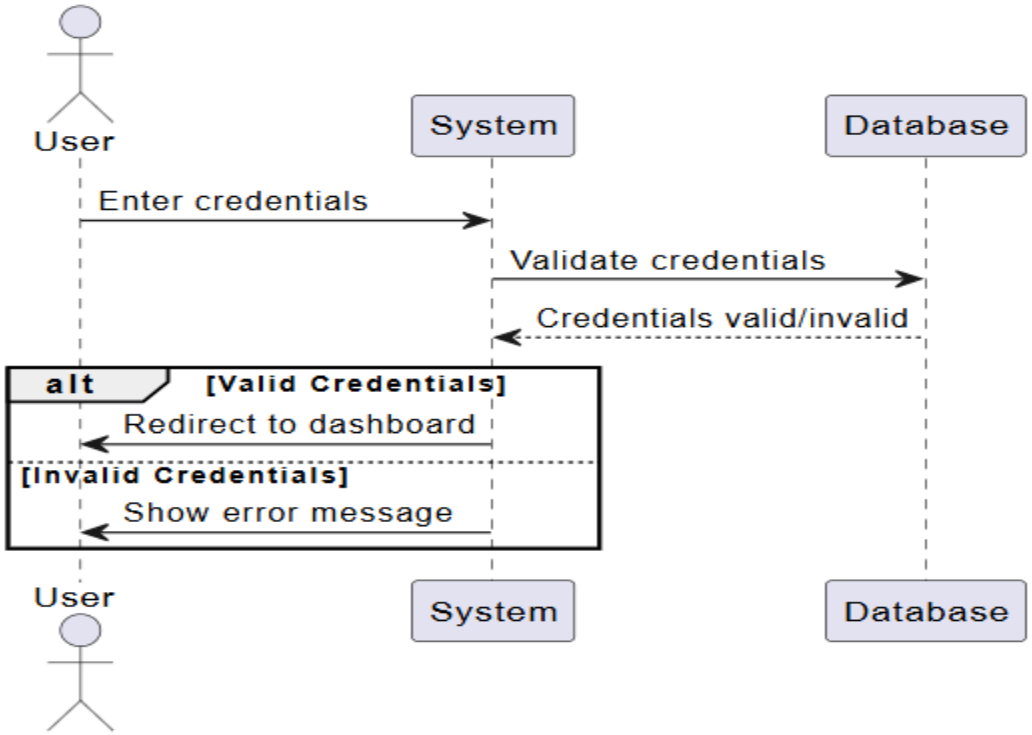
| | |
|---|---|
| **Priority** | High |
| **Basic Flow** | User enters login credentials and clicks the login button. |

**Detailed Flow**

| **Actor Actions** | **System Response** |
|---|---|
| 1. User enters valid credentials (username and password) on the login screen. | System validates the credentials against the database and grants access if valid. |
| 2. User presses the login button. | System redirects the user to the dashboard on successful login. |

**Alternative Course of Action (if any):**

| **Actor Action** | **System Response** |
|---|---|
| 1. User enters invalid credentials. | System displays an error message: "Invalid credentials. Please try again." |
| 2. User leaves required fields empty and presses login. | System prompts: "All fields are required. Please fill them out." |




### 4.2.2 User Signup

| | |
|---|---|
| **Use Case ID** | UC1.2 |
| **Use Case Name** | Signup |
| **Actor(s)** | New User |
| **Pre-Conditions** | User accesses the registration page. |
| **Priority** | High |
| **Basic Flow** | User provides required information and clicks the "Signup" button. |

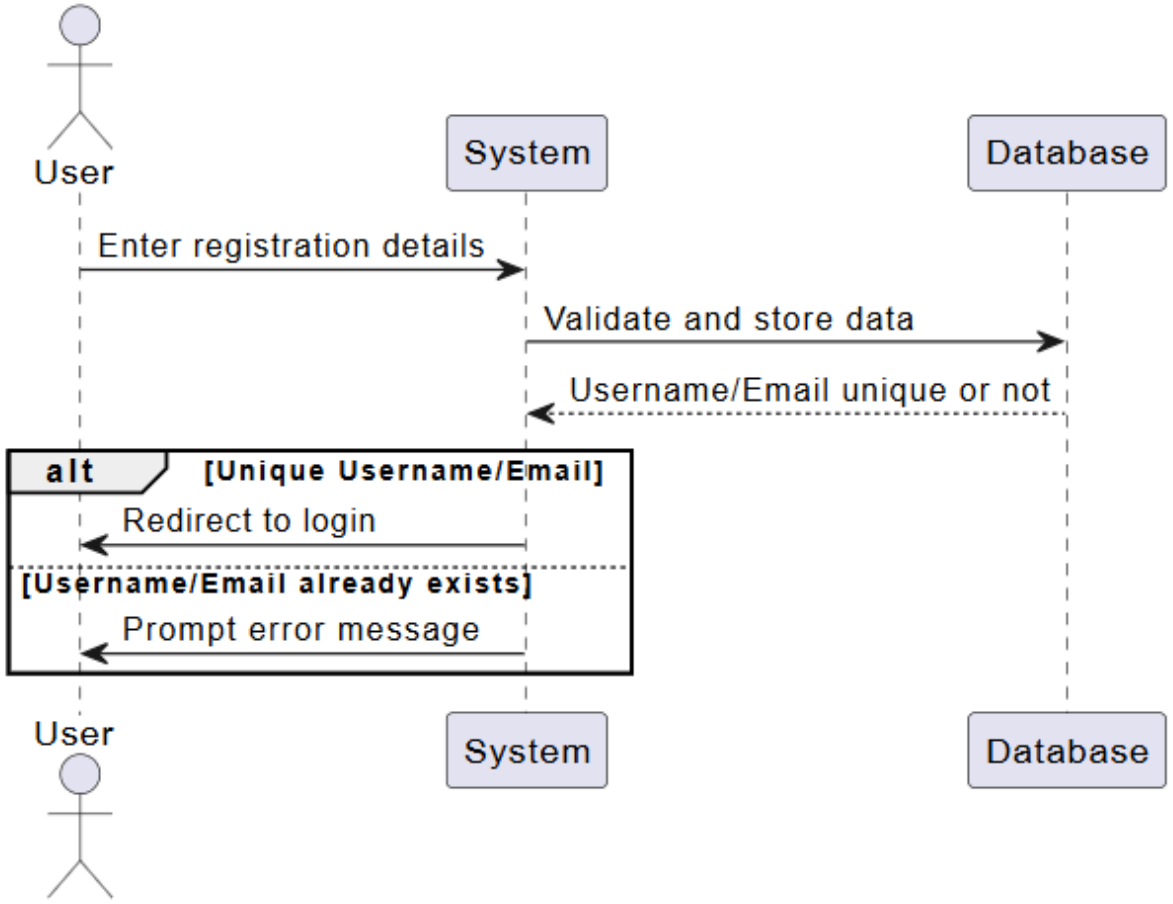
**Detailed Flow**

| **Actor Actions** | **System Response** |
|---|---|
| 1. User enters required details (username, email, password, etc.) and submits. | System validates input and creates a new user record in the database. |

| | |
|---|---|
| 2. User presses the "Signup" button. | System confirms successful signup and redirects the user to the login screen. |

**Alternative Course of Action (if any):**

| Actor Action | System Response |
|---|---|
| 1. User provides incomplete information. | System prompts: "All fields are required. Please complete the form." |
| 2. Email or username already exists. | System displays: "Email/Username already in use. Please choose a different one." |




## 4.3 Course Recommendations

### 4.3.1 Use Case: Course Recommendations

| | |
|---|---|
| **Use Case ID** | UC2.1 |

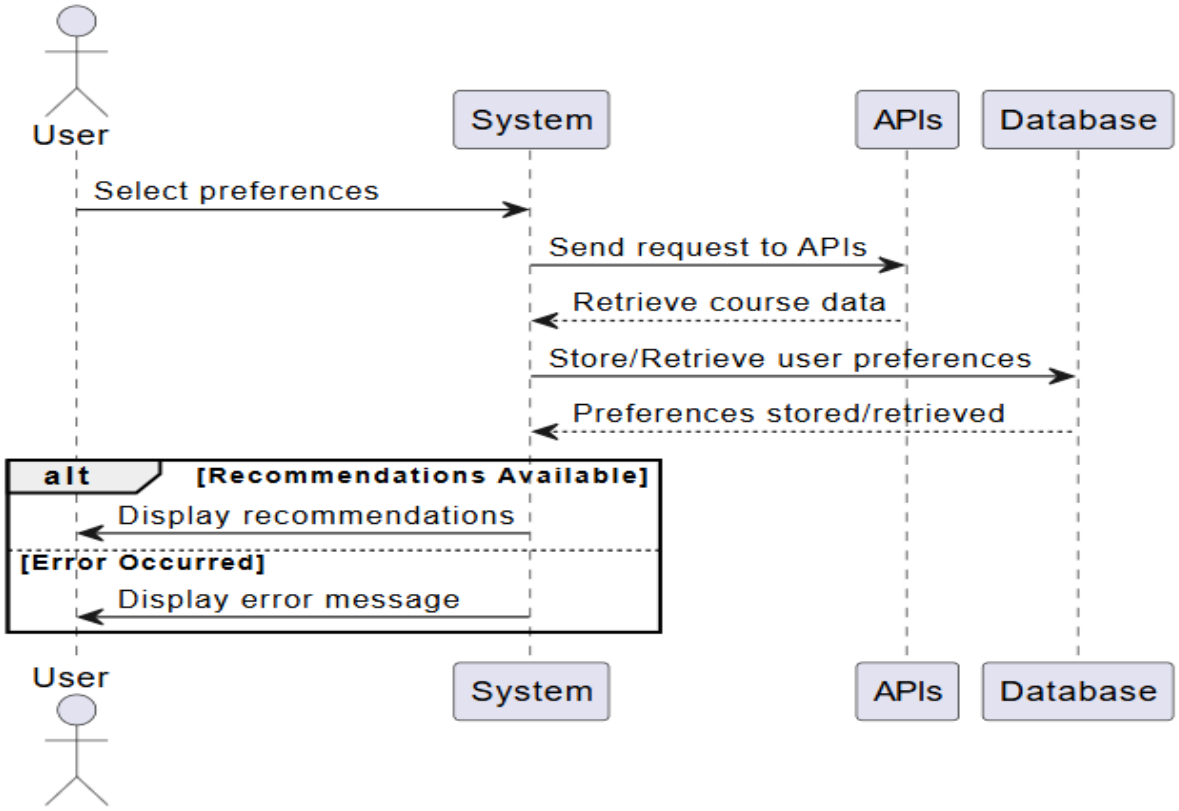
| | |
|---|---|
| **Use Case Name** | Course Recommendation |
| **Actor(s)** | User (learner, administrator) |
| **Pre-Conditions** | User is logged into the system. |
| **Priority** | High |
| **Basic Flow** | User selects preferences and clicks the "Recommend Courses" button. |

**Detailed Flow**

| **Actor Actions** | **System Response** |
|---|---|
| 1. User selects preferences (topics, duration, difficulty) on the dashboard. | System fetches matching courses from external APIs (Coursera, edX, etc.). |
| 2. User clicks "Recommend Courses." | System displays a list of recommended courses with details like title and link. |

**Alternative Course of Action (if any):**

| **Actor Action** | **System Response** |
|---|---|
| 1. User does not select any preferences and presses "Recommend Courses." | System prompts: "Please select at least one preference." |
| 2. No courses match the user’s preferences. | System displays: "No matching courses found. Adjust your filters and try again." |




## 4.4 Institutes Nearby

### 4.4.1 Use Case: Institutes Nearby

| | |
|---|---|
| **Use Case ID** | UC3.1 |
| **Use Case Name** | Institutes Nearby |
| **Actor(s)** | User (learner) |
| **Pre-Conditions** | User is logged in and has location services enabled. |
| **Priority** | Medium |
| **Basic Flow** | User clicks "Find Nearby Institutes." |

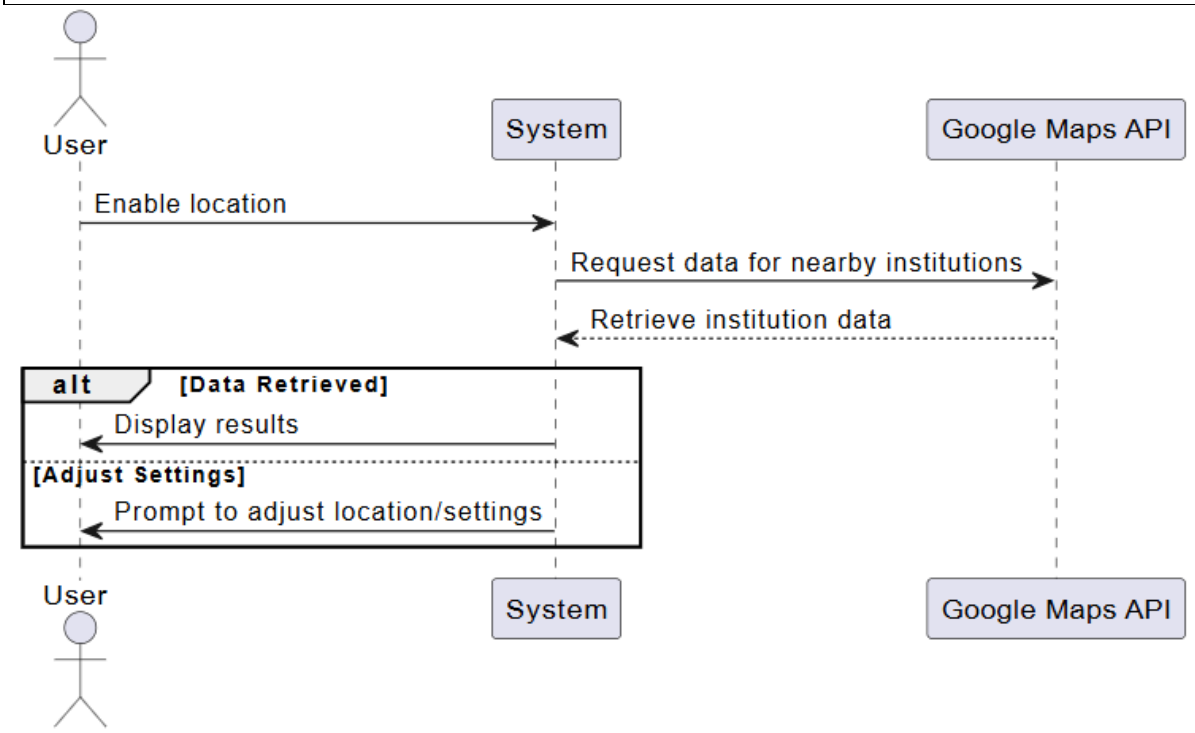
**Detailed Flow**

| **Actor Actions** | **System Response** |
|---|---|

| | |
|---|---|
| 1. User clicks "Find Nearby Institutes" on the dashboard. | System uses Google Maps API to fetch nearby institutions based on user location. |
| 2. User views the list of nearby institutions. | System displays a list with institution name, address, and distance. |

**Alternative Course of Action (if any):**

| **Actor Action** | **System Response** |
|---|---|
| 1. Location services are disabled. | System prompts the user to enable location services. |
| 2. No institutes found nearby. | System displays: "No institutes found within your location radius." |




## 4.5 AI Features

### 4.5.1 Use Case: Lecture Generator

| | |
|---|---|
| **Use Case ID** | UC4.1 |
| **Use Case Name** | Lecture Generator |

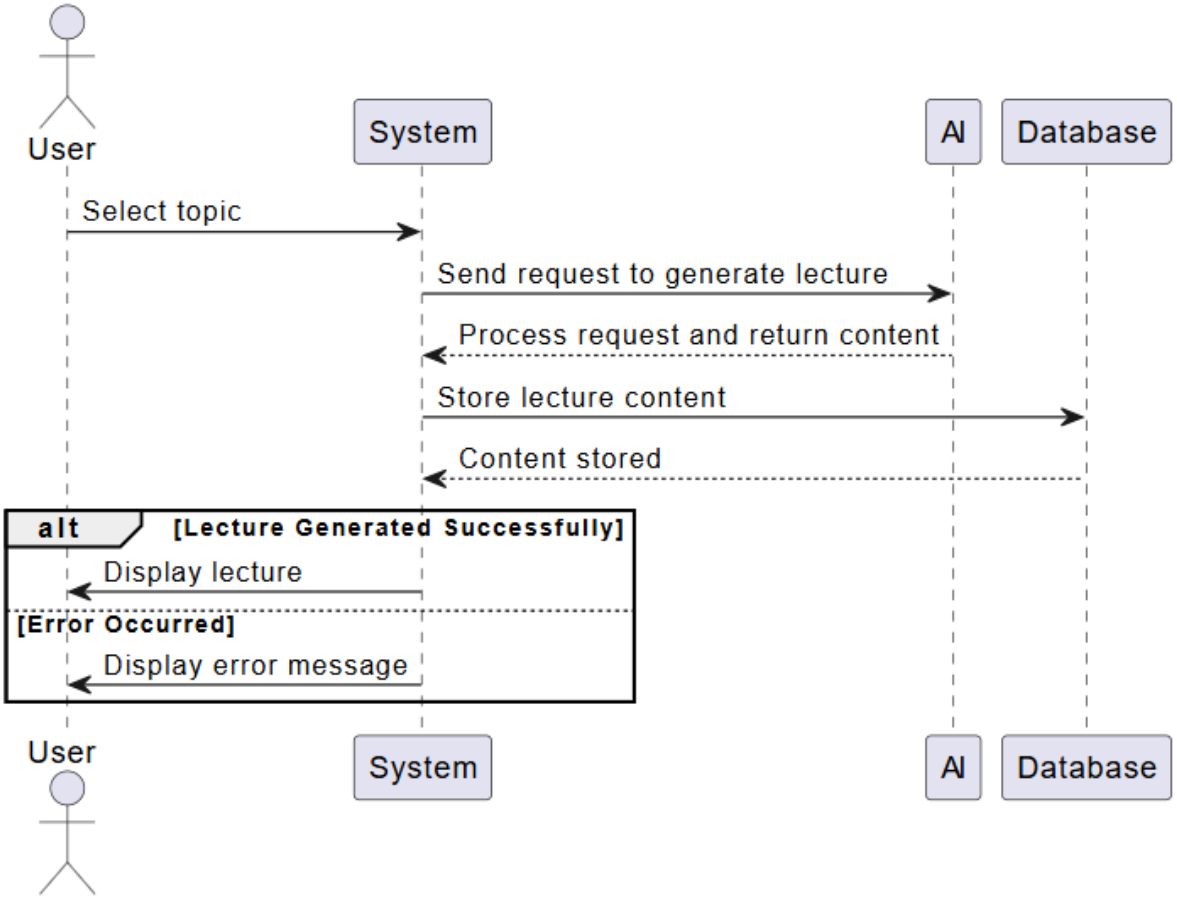
| | |
|---|---|
| **Actor(s)** | User (learner) |
| **Pre-Conditions** | User is logged in and selects a topic. |
| **Priority** | High |
| **Basic Flow** | User selects a topic and clicks "Generate Lecture." |

**Detailed Flow**

| **Actor Actions** | **System Response** |
|---|---|
| 1. User selects a topic from the roadmap or dashboard. | System sends a request to the AI model for generating lecture content. |
| 2. User clicks "Generate Lecture." | System displays the generated lecture text on the lecture screen. |

**Alternative Course of Action (if any):**

| **Actor Action** | **System Response** |
|---|---|
| 1. Selected topic is not supported by the AI model. | System displays: "Unable to generate lecture. Please select another topic." |
| 2. AI model fails to generate content due to connectivity issues. | System displays: "Unable to process the request. Please try again later." |


User
System
AI
Database
Select topic
Send request to generate lecture
Process request and return content
Store lecture content
Content stored
alt
[Lecture Generated Successfully]
Display lecture
[Error Occurred]
Display error message
User
System
AI
Database

### 4.5.2 Use Case: Quiz Creation

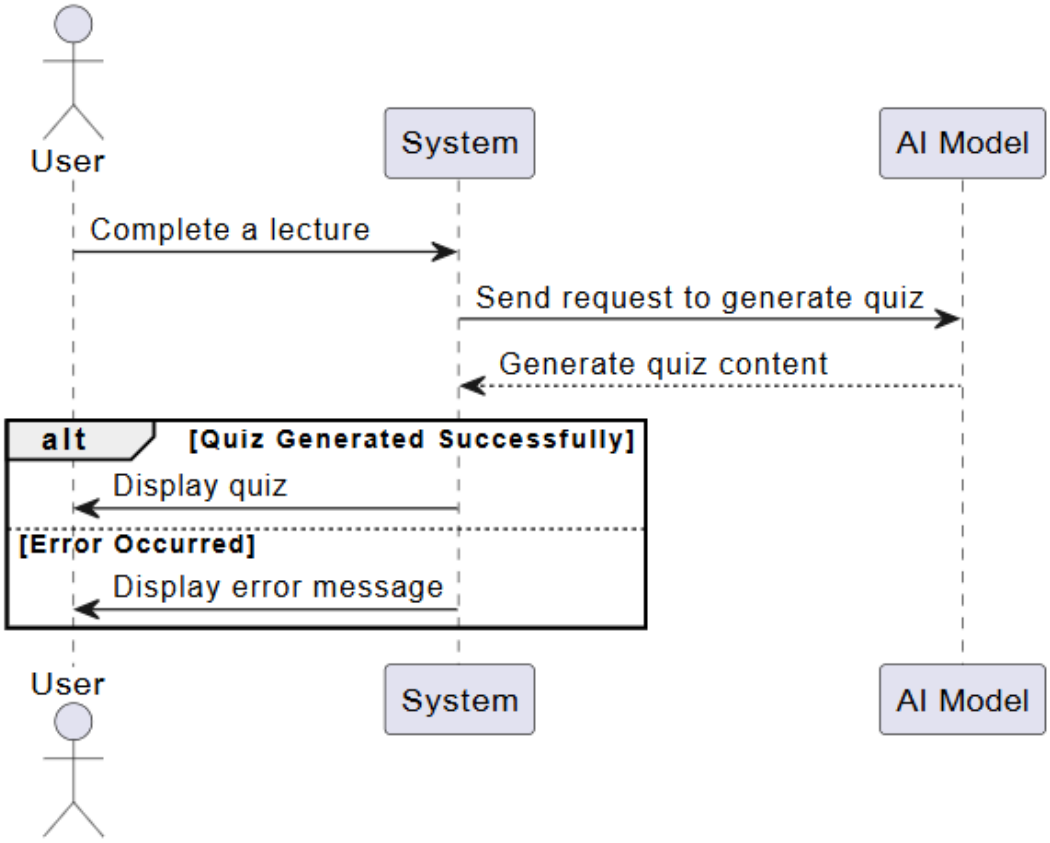
| | |
|---|---|
| **Use Case ID** | UC4.2 |
| **Use Case Name** | Quiz Creation |
| **Actor(s)** | User (learner) |
| **Pre-Conditions** | User completes a lecture. |
| **Priority** | High |
| **Basic Flow** | User clicks "Generate Quiz" after completing a lecture. |

**Detailed Flow**

| **Actor Actions** | **System Response** |
|---|---|
| 1. User clicks "Generate Quiz" after finishing a lecture. | System retrieves key points from the lecture and creates relevant questions. |
| 2. User starts the quiz. | System evaluates responses in real-time and provides feedback. |

**Alternative Course of Action (if any):**

| **Actor Actions** | **System Response** |
|---|---|
| 1. Quiz generation fails due to AI model issues. | System displays: "Quiz generation failed. Please try again later or select a different topic." |
| 2. User skips the quiz generation process. | System stores the lecture as completed and provides an option to generate the quiz later. |




# 5. Agile Planning

## 5.1 Product Backlog

The product backlog includes all the key features and functionalities required for the project. Each feature is prioritized based on its importance and complexity.

1. **Frontend Development**:
    - Design and implement user interfaces for roadmaps, lectures, and quizzes.
    - Ensure responsive design for cross-platform usability.
2. **Backend Development**:
    - Develop REST APIs for communication between the frontend and backend.
    - Integrate AI models for lecture generation, quizzes, and personalized recommendations.
3. **AI Features**:

- Fine-tune AI models like TinyLlama for specific tasks such as text generation.
- Implement AI-driven adaptive learning roadmaps.

4. **External Integrations**:
   - Integrate Google Maps API for finding nearby institutes.
   - Connect to external platforms (e.g., Coursera, edX) for course recommendations.
5. **Database Management**:
   - Create schemas for user data, course information, and progress tracking.
   - Optimize queries for scalability.
6. **Testing and Deployment**:
   - Perform unit, integration, and system testing.
   - Deploy on a cloud platform with continuous integration and delivery pipelines.

## 5.2 Sprint Backlog

Sprints are planned in two-week iterations to deliver incremental value.

1. **Sprint 1: UI/UX Design**:
   - Create wireframes and prototypes for the user dashboard, roadmap generator, and quiz interface.
   - Develop basic components using React or Flutter.
2. **Sprint 2: Backend Setup**:
   - Set up Flask/Django with PostgreSQL.
   - Implement basic API endpoints for handling user preferences and course data.
3. **Sprint 3: AI Model Integration**:
   - Connect AI models (TinyLlama or GPT APIs) to backend APIs.
   - Test lecture and quiz generation features.
4. **Sprint 4: External Integrations**:

- Integrate Google Maps API for nearby institutes.
- Connect APIs for online course recommendations.

5. **Sprint 5: Testing and Optimization**:
   - Conduct end-to-end testing.
   - Optimize API calls, database queries, and AI model performance.
6. **Sprint 6: Deployment**:
   - Deploy the application to a cloud environment.
   - Ensure proper scaling and monitoring tools are in place.

---

## 5.3 Product Backlog Table

| ID | Feature | Description | Priority |
|---|---|---|---|
| 1 | User Authentication | Secure login and registration using OAuth 2.0. | High |
| 2 | Courses Recommendation | Fetch and display relevant courses based on user preferences. | High |
| 3 | Institutes Nearby | Use Google Maps API to list nearby educational institutions and filter by type. | Medium |
| 4 | Roadmap Generator | Generate personalized weekly learning roadmaps based on user goals and progress. | High |
| 5 | Lecture Generator | Produce dynamic, topic-specific lectures with examples and summaries. | High |
| 6 | Quiz Creation and Assessment | Create quizzes dynamically from lecture content and assess user responses with feedback. | High |
| 7 | Progress Tracking | Track user engagement and provide actionable feedback with weekly reports. | Medium |
| 8 | Database Management | Create schemas for user data, course recommendations, and learning history; ensure optimized queries for efficient operations. | High |

| ID | Feature | Description | Priority |
|---|---|---|---|
| 9 | Integration with Educational APIs | Integrate APIs from Coursera, edX, and other platforms to fetch course data. | Medium |
| 10 | Testing and Deployment | Conduct comprehensive testing and deploy the system to a cloud platform for scalability and reliability. | High |

## 5.4 Sprint Backlog Table

| Sprint | Tasks | Feature IDs | Priority | Owner |
|---|---|---|---|---|
| Sprint 1 | Design and implement the user interface (UI) for dashboard, roadmaps, and quizzes. | 1, 2, 4 | High | Frontend Team |
| Sprint 2 | Set up Flask/Django backend with PostgreSQL database; implement basic user authentication API. | 1, 8 | High | Backend Team |
| Sprint 3 | Integrate Google Maps API for locating nearby institutes and fetching user location data. | 3 | Medium | Backend Team |
| Sprint 4 | Implement AI model integration for generating lectures and quizzes; fine-tune models for performance. | 4, 5, 6 | High | AI Team |
| Sprint 5 | Develop the progress tracking system and weekly feedback mechanism; refine database schemas. | 7, 8 | Medium | Backend Team |
| Sprint 6 | Test all features for functionality, scalability, and usability; deploy the system to the cloud with monitoring tools. | 9, 10 | High | DevOps Team |

# 6. Functional Requirements

## 6.1 Courses Recommendation:

- Retrieve and display relevant courses based on user preferences.
- Support filtering by course duration, difficulty level, and provider.

## 6.2 Institutes Nearby:

- Use Google Maps API to list nearby educational institutions.
- Filter by institution type (e.g., university, training center).

## 6.3 AI Features:

- Generate personalized learning roadmaps.
- Create lectures dynamically based on user-selected topics.
- Design adaptive quizzes to evaluate user understanding.

## 6.4 User Authentication:

- Enable secure login and registration.
- Manage user profiles and preferences.

## 6.5 Progress Tracking:

- Record and display user progress through roadmaps and quizzes.
- Provide weekly insights and improvement suggestions.

---

# 7. Nonfunctional Requirements

## 7.1 Performance Requirements

- Response time for model queries: $\leq 2$ seconds.
- Support for up to 10,000 simultaneous users.

## .2 Safety Requirements

- Safeguards against data loss during high usage.
- Secure handling of user information.

### 7.3 Security Requirements

- Use encrypted protocols for sensitive data.
- Employ user authentication via OAuth 2.0.

### 7.4 Software Quality Attributes

- **Usability**: Easy-to-navigate interface for users.
- **Reliability**: 99.9% uptime.
- **Scalability**: Modular architecture for future growth.

### 7.5 Business Rules

- Content must align with user preferences and course objectives.

---

## 8. Other Requirements

- Fine-tuning the TinyLlama model on educational datasets.
- Adherence to copyright and ethical scraping practices.

---

# System Design Specification

## 1. Introduction

### 1.1 Purpose

The purpose of this section is to detail the architectural and design aspects of the **AI Learning** platform. It explains the system's logical structure, interactions, and deployment to ensure proper alignment with the SRS.

### 1.2 Document Conventions

Design diagrams are presented using UML standards, and key system modules are defined with detailed functionality.

---

# 2. Software Architecture

## 2.1 Architectural Overview

The system follows a modular architecture to separate responsibilities, ensure scalability, and facilitate maintenance:

- **Frontend Layer**: Built with Flutter for a cross-platform mobile experience.
- **Backend Layer**: Implemented using Flask/Django, handling API requests, business logic, and AI model integration.
- **AI Model Integration**: TinyLlama model hosted locally or on a server with GPU support.
- **Database Layer**: PostgreSQL for persistent storage of user data, quizzes, and progress tracking.

## 2.2 Key Design Patterns

- **Model-View-Controller (MVC)**: Used to manage interactions between the user interface, data, and control logic.
- **RESTful API**: Facilitates communication between the frontend and backend layers.

---

# 3. Detailed Design

## 3.1 Logical View

- **Class Diagram**:
    - **User**: Represents app users, including personal data, preferences, and progress.
    - **Quiz**: Stores quiz questions, user attempts, and results.
    - **Roadmap**: Tracks the weekly learning roadmap for each user.

## 3.2 Dynamic View

- **Sequence Diagram**:
    - **User Interaction with AI Model**:
        1. User inputs preferences or requests a quiz.
        2. Request is sent to the backend.

3. Backend interacts with TinyLlama for content generation.
4. Generated content is returned to the frontend for display.

## 3.3 Deployment View

- **System Deployment**:
  - **Frontend**: Deployed on Google Play Store and Apple App Store for end-user access.
  - **Backend**: Hosted on a cloud platform (e.g., AWS, Azure) or local server.
  - **AI Model**: TinyLlama hosted locally or on a dedicated GPU-enabled server.

## 3.4 Development View

- **Work Breakdown Structure (WBS)**:
  1. **Frontend Development**:
     - UI/UX design and integration with backend APIs.
  2. **Backend Development**:
     - API creation, database integration, and AI model interfacing.
  3. **AI Model Integration**:
     - Fine-tuning TinyLlama and implementing Flask-based endpoints.

## 3.5 Data Model

- **Entity-Relationship (ER) Diagram**:
  - **Entities**: Users, Courses, Quizzes, Roadmaps.
  - **Relationships**:
    - Users can have multiple roadmaps and quizzes.
    - Quizzes are linked to roadmaps and progress tracking.

---

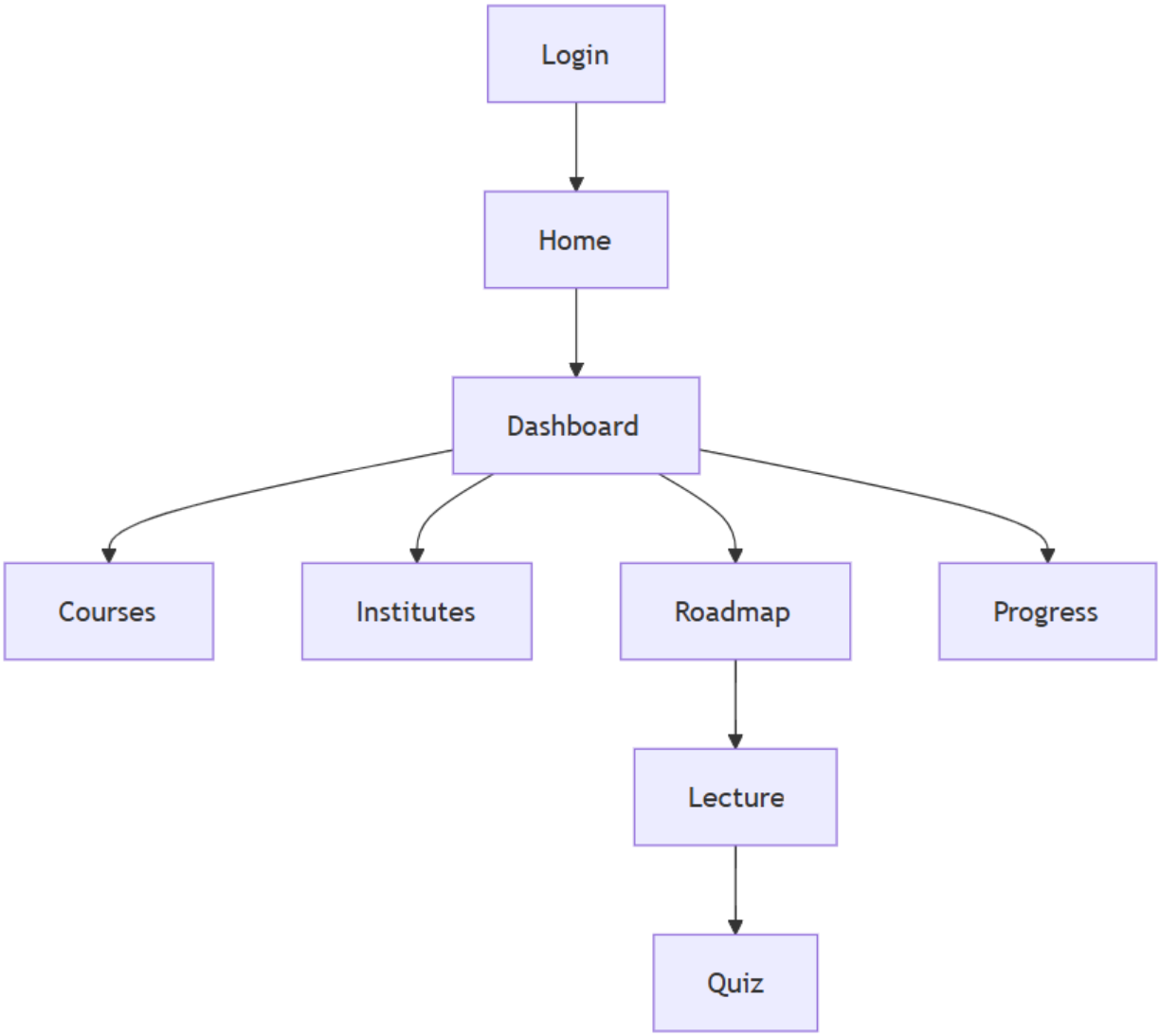
# 4. User Interface Design

## 4.1 Wireframes

- **Dashboard**:

    - Displays weekly progress, recommended courses, and quiz scores.
- **Quizzes**:
    - Interactive question-answer screens with instant feedback.
- **Learning Roadmaps**:
    - Visual representation of weekly tasks and goals.

## 4.2 Navigation Flow



# 5. Formats for Reports

## 5.1 Progress Reports

- **Content**:
    - Weekly roadmap completion percentage.
    - Quiz performance metrics (accuracy, speed).
- **Export Options**:
    - PDF and Excel formats for easy sharing.