Adama Science and Technology University

School of Electrical Engineering and Computing

Department of Computer Science and Engineering

Title: AI Powered E-Learning Platform

Group Members	ID_No
1. Fetene Abebe	ugr/22767/13
2. Firaol Andarsa	ugr/22782/13
3. Fitsum L/birhan	ugr/23363/13
4. Gemechu Deressa	ugr/22751/13
5. Mintesnot Yohannis	ugr/23297/13

Advisor: Genet Shanko

June-2025

DECLARATION

We are students of Adama Science and Technology University in the School of Electrical Engineering and Computing in the Department of Computer Science and Engineering. The information found in this project is our original work. And all sources of materials that will be used for the project work will be fully acknowledged.

NO	Name	ID	Signature
1.	Fetene Abebe	ugr/22767/13	
2.	Firaol Andarsa	ugr/22782/13	
3.	Fitsum L/birhan	ugr/23363/13	
4.	Gemechu Deressa	ugr/22751/13	
5.	Mintesnot Yohannis	ugr/23297/13	

Date of Submission: June 2025,

This project has been submitted for examination with our approval as a university advisor.

Advisor Name: **Genet Shanko** **Signature**

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONOMY

1. UI ---------- User Interface
2. UX ---------- User Experience
3. UAT ---------- User Acceptance Testing
4. AI ---------- Artificial Intelligence
5. HTML—---------- HyperText Markup Language
6. CSS —---------- Cascading Style Sheet
7. JS —---------- Java Script

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who contributed to the development of this project proposal.

We also appreciate the contributions of our team members, for their dedication, creativity, and collaborative spirit. Your hard work and innovative ideas have been instrumental in shaping this proposal and bringing it to fruition.

Finally, we would like to thank all the educators, learners, and industry experts who shared their insights and feedback. Your input has been crucial in identifying the needs and challenges that our project aims to address.

Thank you all for your support and encouragement as we embark on this exciting journey to enhance learning through technology.

ABSTRACT

The "Skill Path" project delivers an AI-Powered E-Learning Platform using Django, Bootstrap, and OpenAI APIs to offer personalized and interactive education. It tackles the lack of individual learning paths and instructor overload by providing AI-driven course recommendations, AI-assisted assessment feedback, progress tracking, and intelligent chatbot support with tool-use capabilities, thereby enhancing learner engagement and outcomes.

Chapter One

1.1 Introduction

The rapid evolution of digital technologies continues to reshape the educational landscape, prompting a shift towards more flexible, accessible, and effective learning modalities. Traditional educational models often struggle to cater to diverse learner needs, pacing, and engagement levels. This project, titled "Skill Path: An AI-Powered E-Learning Platform," aims to address these inherent limitations by developing an intelligent online learning environment. By strategically integrating advanced Artificial Intelligence (AI) capabilities, specifically leveraging Large Language Models (LLMs) via OpenAI APIs and other machine learning techniques, Skill Path endeavors to create an inclusive and adaptive platform. This platform is designed to empower learners by providing personalized learning paths, immediate support, and engaging interactions, while also offering educators tools for efficient assessment and valuable insights into student progress. The motivation behind Skill Path is to harness the transformative potential of AI to foster a more dynamic, responsive, and ultimately more impactful educational experience in an increasingly complex world.

1.2 Background of the Project

Conventional e-learning systems, while offering accessibility, frequently adopt a one-size-fits-all curriculum. This approach often fails to accommodate the unique learning styles, paces, and prior knowledge of individual students, leading to disengagement, suboptimal learning outcomes, and a gap in personalized educational support. Students may experience difficulties with rigid learning schedules and a lack of individualized pathways. Concurrently, educators face significant challenges in managing assessments at scale, providing timely and

nuanced feedback, especially for qualitative responses, and offering continuous support outside of direct interaction. Existing e-learning solutions often lack the real-time intelligent support and the deep adaptability required to effectively address these issues. The emergence and accessibility of powerful AI APIs, particularly from OpenAI, present a unique opportunity to build platforms that can offer more personalized guidance, automate aspects of assessment, and provide on-demand assistance, thereby addressing missed opportunities for truly individualized education.

1.3 Statement of the Problem

In the current educational landscape, learners frequently encounter challenges stemming from standardized learning approaches, a lack of personalized support, and limited meaningful engagement with course materials. Traditional e-learning platforms often fail to adapt to diverse learning needs, resulting in decreased motivation and a potential compromise in learning efficacy. A significant pain point is the difficulty in scaling personalized instructor attention and feedback, particularly for open-ended questions or creative tasks. Furthermore, students often require immediate assistance with their queries or conceptual clarifications, which traditional systems cannot always provide outside scheduled instructor availability. This project identifies the core problem as the gap in providing truly adaptive, engaging, and supportive online learning experiences at scale.

1.4 Objective of the Project

The overarching goal of this project is the design and implementation of "Skill Path," a sophisticated AI-Powered E-Learning Platform. This platform aims to significantly enhance the educational journey for students by delivering

personalized, adaptive, and highly interactive learning solutions. Central to this endeavor is the strategic integration of Artificial Intelligence, primarily through OpenAI APIs, to create a system that is not only user-friendly but also scalable and effective in addressing the core problems identified.

1.4.1. General Objectives

To develop a scalable, user-friendly AI-Powered E-Learning Platform, utilizing Django, Bootstrap 5 (with a custom theme), and OpenAI APIs, that enhances the educational experience through personalized, adaptive, and interactive features.

4.2. Specific Objectives

To achieve the general objective, the **"Skill Path"** project will focus on the following specific, measurable aims:

- **Implement Secure User Authentication**
  - Use email as the primary login credential
  - Enforce mandatory email verification to ensure account security and authenticity
- **Develop a Comprehensive Course Management System**
  - Enable instructors (with admin privileges) to create, organize, and manage courses
  - Support various content formats, including text, video, and interactive quizzes
- **Develop an AI-Powered Course Recommendation Engine**

- Utilize OpenAI language models to generate personalized course suggestions
- Analyze available courses and student history to recommend tailored learning paths with justifications
- Plan future integration of OpenAI Embeddings for deeper personalization beyond basic content similarity

❖ **Build an Automated Assessment System**

- Enable automatic grading for Multiple Choice Questions (MCQs)
- Leverage OpenAI's GPT models to provide AI-generated feedback and preliminary scoring for Short Answer Questions

❖ **Deploy a Real-Time AI Chatbot for Learner Support**

- Power the chatbot with OpenAI GPT models (e.g., GPT-4o-mini)
- Integrate tool use (function calling) to access and interact with course data, user progress, and other platform-specific resources in real time

❖ **Create Intuitive Dashboards for Users**

- Provide students and administrators with real-time insights into learning progress, course engagement, and statistics
- Display gamification elements such as points, badges, and achievements

❖ **Establish an Interactive Discussion Forum**

- Encourage peer-to-peer interaction, support, and collaborative learning through structured discussions

❖ **Implement a Comprehensive Gamification System**

- Incorporate points, achievement badges, progressive levels, and leaderboards

- Increase learner motivation and engagement through structured gamified rewards

❖ **Introduce a Course Review and Rating System**

- Allow students to leave feedback and rate courses
- Help future learners make informed decisions based on peer insights

❖ **Set Up a Digital Certification System**

- Issue verified digital certificates to students upon successful course completion

1.5 Scope and Limitation

This section outlines the defined boundaries, core functionalities, and inherent constraints of the "Skill Path" AI-Powered E-Learning Platform project.

1.5.1. Scope of the Project

The primary scope of the *Skill Path* project is to develop a fully functional, web-based e-learning platform featuring:

- **Technology Stack:**
  - **Backend:** Django (Python) for core application logic and user roles management
  - **Frontend:** Custom-themed Bootstrap 5 for a responsive, mobile-friendly UI
  - **Database:** SQLite as the production relational database

- **AI Integration:** OpenAI APIs (GPT-4o-mini) for intelligent features including chatbot, feedback generation, and recommendations

Core Functionalities

- **User Management**
  - Secure email-based registration and login
  - Mandatory email verification
  - Password management (reset/change)
  - User profile with bio and avatar
  - Three primary user roles:
    - **Student**
    - **Instructor** (`is_staff` via Django Admin and Group Permissions)
    - **Administrator** (Superuser access)
- **Course Delivery**
  - Instructors can create structured courses with:
    - Modules and lessons
    - Diverse content: text, embedded video, downloadable files, and quizzes
  - Students can browse and (conceptually) enroll in courses
  - Sequential navigation with lesson locking/unlocking based on completion
- **AI-Powered Chatbot**
  - Accessible platform-wide via a floating widget
  - Powered by OpenAI GPT models (e.g., GPT-4o-mini)

    - Provides real-time assistance and query resolution
    - Uses **Tool Use (Function Calling)** to access platform data (e.g., course info, user progress)
    - Session-based conversation history managed via Django sessions
- **Intelligent Assessments**
    - **MCQ grading:** Automatically scored programmatically
    - **SAQ feedback and scoring:** AI-generated feedback and a 0–10 scale score using GPT models, with an optional instructor review flag

- **AI-Driven Course Recommendations**
    - For authenticated users (dashboard):
        - Personalized suggestions based on completed courses, interests, and available options using OpenAI
        - Fallback to newest or trending courses if AI output is unavailable
    - For public users (landing page): Featured or latest courses
- **Progress Tracking**
    - Students can mark lessons as complete
    - Tracks:
        - Per-lesson completion
        - Module-level progress percentage
        - Overall course progress percentage
- **Student Dashboard**
    - Displays:
        - Active courses and progress

        - Recently completed lessons
        - Earned gamification rewards (points, badges, levels)
        - Quiz history
        - AI-based course recommendations
- **Instructor Dashboard**
    - For users with `is_staff` permissions
    - Displays:
        - Course statistics (e.g., enrolled students, average progress)
        - Access to course management (via Django Admin)
        - Insights into individual student progress

- **Admin Dashboard**
    - Platform-wide analytics for superusers and admins
    - Includes:
        - Total users, courses, and active learners
        - Forum activity and platform engagement metrics
- **Discussion Forum**
    - Structured discussion areas with:
        - Categories
        - Threads
        - Posts
    - Promotes peer learning and community interaction
- **Gamification System**
    - Points awarded for actions (e.g., lesson completion, forum posts)
    - Achievement badges
    - Leveling system based on cumulative points
    - Public leaderboard for competitive engagement

- **Course Reviews & Ratings**
    - Authenticated students can submit:
        - Star ratings
        - Written feedback
    - Average course ratings are displayed publicly
- **Digital Certification**
    - Automatic generation of verifiable digital certificates
    - Available for download/viewing upon course completion
- **UI Theming**
    - Custom Bootstrap 5 theme
    - Responsive design with light/dark mode toggle for enhanced accessibility and user experience

1.5.2. Limitations

While Skill Path is designed to be a robust and feature-rich e-learning platform, the following limitations are acknowledged due to constraints in time, resources, and complexity:

- Absence of Real-Time Collaboration Tools
    - Features such as live co-editing, interactive whiteboards, or integrated video conferencing are outside the current project scope
- No Plagiarism Detection
    - The system does not include automated plagiarism detection for Short Answer Questions (SAQs) or forum content
- Lack of Native Mobile Applications

    - The platform is designed as a responsive web application
    - Native mobile apps for iOS or Android are not included in this development phase
- Single-Language Interface
    - The user interface is primarily in English
    - Full internationalization (i18n) and localization (l10n) are not implemented, though blocktrans is used in some Django templates
- No Offline Access Support
    - The platform requires an active internet connection
    - Offline content access or sync capabilities are not available

1.5.3 Target Groups

the target audience for the AI-Powered E-Learning Platform would be:

1. **University Students**:
    - They often require flexible learning options and personalized resources, making them a prime audience for an adaptive learning platform.
2. **Adult Learners**:
    - This segment is increasingly motivated to acquire new skills for career advancement or personal growth, making them ideal users of an adaptive e-learning solution.

1.6 Deliverables

The primary deliverables for the "Skill Path: AI-Powered E-Learning Platform" project include:

- **A Fully Functional Web Application:** The deployed "Skill Path" platform accessible via a web browser, incorporating all features outlined in the project scope, built using Django, Bootstrap 5, and integrated with OpenAI APIs.
- **Source Code Repository:** A complete Git repository containing all backend (Python/Django) and frontend (HTML, CSS, JavaScript) code for the platform.
- **Project Documentation (This Document):** A comprehensive report detailing the project's inception, objectives, design, implementation, testing, and conclusions, including relevant diagrams and system descriptions.

1.7. Feasibility Study

1.7.1. Technical Feasibility

The *Skill Path* project is considered technically feasible, supported by a mature tech stack and accessible development tools:

- **Backend Framework**
    - **Django (Python):** A robust, secure, and scalable web framework with a well-established ecosystem and comprehensive documentation

    - Ideal for rapid development of complex, database-driven applications
- **Frontend Development**
    - **Bootstrap 5:** Enables fast development of a responsive, modern, and mobile-friendly user interface with customizable theming options
- **AI Integration**
    - **OpenAI APIs:** Pre-trained large language models are accessed via well-documented APIs, eliminating the need for local training and infrastructure
    - **Python Libraries:**
        - openai: For GPT-based features like chat, feedback generation, and recommendations
        - scikit-learn: Used for initial implementation of the TF-IDF course recommendation engine

- **Development Tools**
    - **Git:** For version control and collaborative development
    - **VS Code (or similar IDE):** Provides an efficient coding environment with Django and Python support
- **Deployment Strategy**
    - **Platform-as-a-Service (PaaS):** Services like **Render** simplify deployment and infrastructure management, allowing for continuous integration and delivery

1.7.2. Operational Feasibility

The *Skill Path* platform is designed with user-friendliness and operational efficiency in mind:

- **Usability and Functionality**
    - The custom-themed interface is designed to be intuitive and accessible across all user roles: students, instructors, and administrators
    - Core functionalities—including course navigation, AI chatbot interaction, quiz taking, forum participation, and dashboard access—are implemented with simplicity and ease of use as priorities
- **User Acceptance and Satisfaction**
    - Personalized features such as AI-driven course recommendations and AI-assisted feedback aim to enhance user engagement
    - Gamification elements (points, badges, levels) and community forums encourage continued interaction and motivation
    - The course review system provides a structured channel for user feedback and satisfaction measurement
- **Maintenance and Support**
    - Django's modular architecture supports long-term maintainability and simplifies the process of future enhancements
    - Initial support will be provided through comprehensive documentation and the availability of the project team
    - As the platform matures, a dedicated support mechanism may be introduced

    - Routine maintenance tasks include regular dependency updates and security patching
- **Security and Privacy**
    - Django’s built-in security features include protections against XSS, CSRF, and secure password hashing
    - Mandatory email verification enhances account-level security
    - Sensitive API keys are securely managed via environment variables
    - Data handling practices will be guided by privacy considerations and best practices for safeguarding user information

1.7.3. Economic Feasibility

Development Costs

- Primarily consist of the development team's time and effort
- Utilization of open-source frameworks and libraries—Django, Bootstrap, and Python—minimizes software licensing expenses

Infrastructure Costs

- Hosting:
    - Initial deployment on Render's free or starter tiers helps maintain low hosting costs
    - Scaling to paid tiers may be required as the user base grows
- AI API Usage:

    - Costs are usage-dependent, driven by OpenAI API calls (e.g., chatbot, SAQ grading, future embeddings)
    - Cost control strategies include:
        - Careful prompt engineering
        - Caching frequent requests
        - Choosing cost-effective models over higher-priced alternatives
    - Initial development can take advantage of free credits or low-usage tiers

Resource Allocation

- The project is dependent on the skills and contributions of the development team
- Time is the primary resource constraint

1.8 Significance of the Project

The "Skill Path" project holds significance in demonstrating a practical and modern approach to e-learning by effectively integrating Artificial Intelligence. It aims to democratize access to personalized learning by providing tools that adapt to individual student needs, a feature often lacking in traditional or simpler online platforms. By automating aspects like MCQ grading and providing AI-assisted feedback for SAQs, it offers a scalable solution to reduce instructor workload and provide more immediate feedback to learners. The AI chatbot enhances user support and engagement. Furthermore, the project showcases how current AI APIs can be integrated into a robust web framework like Django to create intelligent,

interactive, and data-driven applications. This contributes to a more inclusive and effective educational environment, preparing learners more adeptly for future challenges.

1.9 Beneficiaries of the project

The key beneficiaries of the "Skill Path" AI-Powered E-Learning Platform include:

- **Students:** Gain access to personalized learning paths, tailored course recommendations, instant AI-assisted feedback, 24/7 chatbot support, engaging gamified experiences, and tools to track their progress effectively. This leads to a more motivating, efficient, and customized learning journey.
- I**nstructors (Content Creators/Admins):** Benefit from tools for organizing course content (via Django Admin), automated MCQ grading, and AI assistance in providing initial feedback for SAQs, potentially reducing their grading workload. They also gain insights into student progress through administrative dashboards.
- **Educational Institutions (Potential Adopters):** Could utilize such a platform to enhance their online learning offerings, improve student engagement and outcomes, and provide more scalable personalized education.
- **The Development Team:** Gains practical experience in full-stack web development with Django, frontend design with Bootstrap, AI API integration (OpenAI), database management, and project deployment.

1.10 Methodology

This project adopted an Agile development methodology, emphasizing iterative development, continuous feedback, and flexible adaptation to evolving requirements. The development process was structured into distinct phases, each building upon the previous one:

1. Planning & Design (Initial Phase): Defining project objectives, scope, initial feature set, technology stack (Django, Bootstrap, SQLite, OpenAI APIs), and system architecture (MTV). Creating initial wireframes and data models.

2. Core Functionality Development (Phase 1 of Implementation): Implementing user authentication (custom user model, email verification), basic course structure (Course, Module, Lesson models), and initial UI setup.

3. AI Integration & Initial Enhancements (Phase 2): Integrating the OpenAI API, developing the AI chatbot (with streaming and session-based history, then tool use), setting up AI-assisted SAQ feedback, and implementing basic student progress tracking.

4. Community & Engagement Features (Phase 3): Developing the discussion forum, and the full gamification system (points, badges, levels, leaderboard), and course review/rating system.

5. Advanced Features & Polish (Phase 4): Implementing course completion certificates, refining recommendation systems (OpenAI based), and enhancing dashboards (student, instructor, admin).

6. Refinement, Testing, & Deployment (Phase 5 - Current/Ongoing): Focusing on UI/UX polish, comprehensive testing (unit, integration, manual),

performance optimization, security review, documentation, and deployment to Render.

1.11 Development tools

The primary tools and technologies utilized in the development of the "Skill Path" platform include:

- **Backend Framework:** Django (Python) for robust, scalable server-side logic, ORM, admin interface, and template rendering.
- **Frontend Technologies:**
    - HTML5, CSS3.
    - Bootstrap 5 for responsive design and pre-built UI components, customized with a unique theme using CSS variables.
    - JavaScript (Vanilla JS) for client-side interactivity, including the AI chatbot interface, theme toggling, and dynamic content updates.
- **Database:**
    - SQLite (for production deployment on Render).
- **AI Integration:**
    - OpenAI Python library for interacting with OpenAI APIs.
    - OpenAI GPT models (e.g., GPT-4o-mini) for the AI chatbot (including tool use) and for generating feedback/preliminary scores for Short Answer Questions.

    - (Mention OpenAI Embeddings if you plan to fully integrate them for recommendations, e.g., text-embedding-ada-002).

- **Version Control:** Git and GitHub for source code management and collaboration.
- **Development Environment:**
    - Visual Studio Code (VS Code) as the primary Integrated Development Environment (IDE).
    - Python virtual environments (venv) for managing project dependencies.
- **Deployment & Hosting:**
    - Render as the Platform as a Service (PaaS) for deploying the web application and SQLite database.
    - Gunicorn as the WSGI HTTP server for running the Django application in production.
    - Whitenoise for serving static files in production.

11. Required resources with cost

The development of "Skill Path" primarily utilized readily available open-source software and cloud services with free or low-cost initial tiers.

- Human Resources:
    - The project was undertaken by a team of five students.
    - Guidance from academic advisor: Genet Shanko.
- Technical Resources (Software & Services):
    - Software (No direct cost): Python, Django, SQLite, Git, VS Code, Bootstrap 5.
    - OpenAI API: Utilized initial free credits or a pay-as-you-go plan with low usage during development.
    - Hosting (Render): Leveraged Render's free tiers for the web service.
- Financial Resources:
    - Internet Access: Standard internet access costs for development team members.
    - Time: The most significant resource was the time invested by the development team.

12. Task and Schedule

Total estimation of days:

Milestone	Tasks	Reporting	Hrs	Date
1 - Planning Phase				
1.1	Proposal preparation	none		oct 16, 2024
1.2	Proposal Submission	none		Nov 4, 2024
1.3	Proposal defence	Report proposal to advisor		Nov 8, 2024
1.4	Chapter 2 Submission			Nov 15, 2024
1.5	Chapter 3 Submission			Nov 29, 2024
1.6	Chapter 4 Submission			Dec 06, 2024
1.7	Chapter 5 Submission			Dec 12, 2024
2 - Development Phase				
3 - Testing Phase				
4 - Deployment Phase				

Table 12.1 Task breakdown and Milestone

13. Team Composition

The project team will comprise:

- Project Manager: Firaol Andarsa
- System Developer(s): Fetene Abebe, Firaol Andarsa, Mintesnot Yohannis, Fitsum L/birhan, Gemechu Deressa.
- System Tester: Fitsum L/birhan, Gemechu Deressa.

Chapter 2: Description of Existing System and Literature Review

2.1 Major Function of Existing System

The existing e-learning systems primarily serve to deliver educational content through online platforms. Their main functions include:

- **Content Delivery:** Providing access to a wide range of courses and learning materials, including videos, readings, and quizzes.
- **User Management:** Enabling user registration, profile management, and course enrollment.
- **Assessment and Feedback:** Offering assessment tools such as quizzes and tests, along with feedback mechanisms for learners.
- **Progress Tracking:** Allowing users to monitor their learning progress through dashboards and performance metrics.

These systems typically focus on a standardized curriculum, offering a set learning path for all users without significant personalization.

2.2 Users of Current System

The current e-learning systems cater to various stakeholders, including:

- **Students:** Seeking flexible learning opportunities that fit their schedules and learning preferences. They expect engaging, interactive content and personalized learning paths.

- **Instructors:** Looking for tools to manage courses, assess student performance, and provide feedback efficiently. They require systems that reduce administrative burdens and enhance student interaction.
- **Administrators:** Responsible for overseeing the platform's operation, ensuring it meets educational standards, and managing user data. They need robust reporting tools and analytics to monitor system effectiveness.

2.3 Drawbacks of Current System

Despite their functionalities, existing e-learning platforms exhibit several significant drawbacks:

- **Lack of Personalization:** Most platforms do not tailor learning experiences to individual student needs, leading to disengagement and ineffective learning.
- **Rigid Learning Paths:** Students are often constrained to a fixed curriculum, which does not accommodate different learning styles or paces.
- **Manual Workload for Instructors:** Grading and providing feedback on assessments typically require considerable manual effort, which can be time-consuming and prone to inaccuracies.
- **Limited Interaction:** Many systems lack features that promote meaningful interaction between students and instructors, hindering collaborative learning experiences.

- **Access Issues:** While online learning offers flexibility, not all students have equal access to technology or reliable internet, exacerbating educational inequalities.

These limitations highlight the need for a more adaptive and efficient e-learning solution.

2.4 Literature Review

The literature review provides an overview of existing research relevant to AI in e-learning, emphasizing the need for personalized learning environments. Key findings from the literature include:

- **Personalized Learning:** Studies indicate that personalized learning enhances student engagement and improves academic performance. Tailoring educational experiences to individual needs can significantly increase motivation and retention (Smith et al., 2020).
- **AI in Education:** Research shows that integrating AI can facilitate adaptive learning by analyzing student data to provide real-time recommendations and feedback (Johnson & Lee, 2021). Machine learning algorithms can identify patterns in student behavior and performance, allowing for timely interventions.
- **Challenges of Traditional Systems:** Various studies highlight the inefficiencies of conventional e-learning systems, particularly regarding their inability to accommodate diverse learning styles (Brown & Davis, 2019). These systems often lack the flexibility necessary to support self-directed learning.

Chapter 3: Proposed System

3.1. Overview

The "Skill Path" platform is an AI-Powered E-Learning System designed to overcome the limitations inherent in traditional online education. It aims to provide a highly personalized, flexible, engaging, and interactive learning experience for students, while simultaneously offering tools to reduce instructor workload through intelligent automation. Key features of the "Skill Path" platform, built upon a robust Django backend and a responsive Bootstrap 5 (custom-themed) frontend, include AI-driven personalized course recommendations, automated MCQ assessment with AI-assisted SAQ feedback, real-time intelligent chatbot support using OpenAI's GPT models with tool-use capabilities, comprehensive student progress tracking, a dynamic gamification system, an interactive discussion forum, and course review/rating functionalities. The system supports distinct roles for students, instructors (managing content via the Django Admin and an instructor dashboard), and administrators, ensuring a tailored experience for each user group.

To achieve its objectives, "Skill Path" leverages advanced technologies, including Python with the Django framework for backend development, SQLite for data persistence in production, and direct API integrations with OpenAI for its core AI functionalities. The architecture is designed for scalability and user-friendliness, featuring a responsive web interface and a modular backend for efficient data management and future enhancements.

3.2. Functional requirement

The functional requirements define the specific actions, behaviors, and operations that the "Skill Path" platform must perform to meet user expectations and achieve its objectives. The proposed system must deliver the following:

1. **User Role Management & Authentication:**
    - Allow multi-role access for Students, Instructors (staff users), and Administrators (superusers).
    - Enable secure user registration using email as the primary identifier, with mandatory email verification.
    - Provide robust login/logout functionality and secure password management (change and reset via email).
    - Allow users to create and manage their profiles (bio, avatar).
    - Track user activities related to course progress, forum participation, and gamification.
2. **Personalized Learning & Recommendations:**
    - (Authenticated Users) Recommended courses based on their interaction history (completed lessons) and (optionally) stated preferences, utilizing OpenAI LLMs to analyze available courses and suggest relevant next steps.
    - (Anonymous Users/Fallback) Display newest or generally popular courses.

- (Future Enhancement) Implement adaptive learning paths that adjust based on ongoing performance.

3. **Course Management (Instructor/Admin Focus):**
    - Allow instructors (via Django Admin) to create, update, organize (into modules and lessons), and delete courses.
    - Support various content formats within lessons, including rich text, embedded videos (e.g., YouTube), and downloadable files.
    - Enable creation and management of quizzes associated with lessons.
4. **Automated & AI-Assisted Assessment:**
    - Facilitate the creation of quizzes comprising Multiple Choice Questions (MCQs) and Short Answer Questions (SAQs).
    - Provide automated, instant grading for MCQs.
    - Utilize OpenAI APIs to generate qualitative feedback and a preliminary rubric-based score (0-10) for SAQs, flagging answers that may require further instructor review.
    - Allow students to view their quiz results, including detailed feedback and scores.
5. **Progress Tracking & Dashboards:**
    - Enable students to mark lessons as complete.
    - Display individual lesson completion status, module progress percentages, and overall course completion percentages in real-time.

- Provide a personalized Student Dashboard showcasing enrolled/active courses, progress metrics, recent activity, earned gamification rewards, and certificates.
- Offer an Instructor Dashboard for instructors to monitor their courses, view aggregated student progress for their courses, and access management links.
- Include an Administrator Dashboard for platform-wide analytics and user/content management overviews.

6. **Interactive Features:**
    - AI Chatbot Assistance: Feature an integrated, streaming AI chatbot (powered by OpenAI GPT models with Tool Use) accessible throughout the platform, providing instant answers to common questions, explanations, platform navigation help, and fetching specific data (like course lists or user progress) via backend tool execution.
    - Discussion Forums: Provide a structured discussion forum with categories, threads, and posts to enable students and instructors to engage in course-related discussions, ask questions, and share knowledge.
7. **Feedback Mechanism:**
    - Allow authenticated students who have engaged with a course to submit star ratings (1-5) and textual reviews for courses.

    - Display average course ratings and individual reviews on course detail pages.

8. **Gamification Elements:**
    - Implement a points system where users earn points for activities like completing lessons and participating in forums.
    - Award badges for specific achievements (e.g., completing first lesson, reaching point milestones, course completion).
    - Incorporate a leveling system based on accumulated points.
    - Display a public leaderboard ranking users by points.
9. **Certification:**
    - Automatically issue a digital certificate upon a student's successful completion of all lessons in a course.
    - Allow students to view their earned certificates via their dashboard.

3.3. Non-functional requirement

The non-functional requirements define the quality attributes and operational characteristics of the "Skill Path" platform:

- **Performance:** The system must handle concurrent user requests efficiently, ensuring quick page load times and responsive interactions. AI API calls should be optimized (e.g., with caching for recommendations) to minimize perceived latency.

- **Scalability:** The platform's architecture (Django, SQLite, Render hosting) should be capable of accommodating a growing number of users, courses, and data without significant performance degradation.
- **Reliability & Availability:** The platform should maintain high availability, aiming for minimal downtime. Database backups and robust error handling will be implemented.
- **Security:**
    - Implement robust security measures to protect user data, including secure password hashing, CSRF protection, XSS prevention (leveraging Django's built-in features).
    - Ensure secure handling of API keys (OpenAI, email services) via environment variables.
    - Utilize HTTPS for all data transmission in production.
    - Implement email verification to validate user accounts.
- **Usability & Accessibility:**
    - The user interface must be intuitive, easy to navigate, and accessible to users with varying technical skills.
    - The design will adhere to responsive web design principles for compatibility across desktops, tablets, and mobile devices.
    - Basic web accessibility considerations (semantic HTML, ARIA attributes where appropriate, keyboard navigability) will be incorporated.

- **Maintainability:** The system architecture, based on Django's modular app structure, should support straightforward updates, bug fixes, and future feature enhancements. Code will adhere to defined coding standards for readability.
- **Responsiveness (UI):** The frontend, built with Bootstrap 5 and a custom theme, will be designed with a mobile-first approach, ensuring a consistent and usable experience across various screen sizes and devices.

3.4. System model

The system model for "Skill Path" illustrates its structure, functionality, user interactions, and data flow, primarily following the Model-Template-View (MTV) architectural pattern inherent to Django.

3.4.1. Scenario

This section outlines critical user scenarios demonstrating real-world interactions with the "Skill Path" AI-Powered E-Learning Platform. The following examples reflect the final implementation, including features such as email verification, AI-powered chatbot with tool use, AI-assisted SAQ feedback, and gamification reward mechanisms.

Scenario 1: Sign-Up Process

User: New Student

Description:

1. Accessing the Platform: The Student navigates to the "Skill Path" homepage.
2. Initiating Sign-Up: Clicks the “Sign Up” button.

3. Filling Out the Registration Form: A registration form is presented requiring:
    - Email Address (used as the username)
    - Password and Confirm Password
    - Optional: First Name, Last Name
      (The user role defaults to "Student.")
4. Submitting the Form: Clicks “Create Account”.
5. Account Creation (Inactive): System creates a new user with is_active=False.
6. Email Verification Sent: A verification email is dispatched with a secure tokenized link.
7. User Verifies Email: The student accesses their inbox and clicks the link.
8. Account Activation: The token is validated, is_active is set to True, email marked verified, and the user is auto-logged in.
9. Redirection: A success message is displayed, and the user is redirected to their personalized dashboard.

Scenario 2: Sign-In Process

User: Registered Student.

Description:

1. Accessing the Platform: The student returns to the homepage of the AI-Powered E-Learning Platform.

2. Initiating Sign-In: He clicks on the "Sign In" button located at the top right corner of the homepage.

3. Entering Credentials: A sign-in form appears, and the student enters credentials:

- Email Address

- Password

4. Submitting the Form: He clicks the "Sign In" button.

5. Successful Authentication: The system verifies his credentials. Upon successful authentication, the student is redirected to his personalized dashboard, where he can access recommended courses and learning materials.

6. Handling Errors: If the student enters incorrect credentials, an error message appears, prompting him to re-enter his email or password. He has the option to click "Forgot Password?" to initiate a password reset process.

Scenario 3: Personalized Course Recommendation

- User: An authenticated Student
- Description:
    - Login & Initial Interaction: Alex logs into the "Skill Path" platform. Initially, as a new user or one with limited activity, the "Recommended For You" section on their dashboard might display generally popular or recently added courses.

- Course Engagement: Alex enrolls in and completes several lessons in courses like "Introduction to Python" and "Data Analysis Fundamentals." The system records these lesson completions (UserLessonProgress).
- Accessing Recommendations: Later, Students visit their Student Dashboard or the platform's homepage.
- AI Recommendation Generation: analyzes the textual content (titles, descriptions, categories) of the courses Alex has interacted with.
- Displaying Recommendations: The system presents a list of "Recommended For You" courses that are most similar in content to those Student has shown interest in (e.g., "Advanced Python Techniques," "Introduction to Machine Learning").
- Exploration: Students review these tailored recommendations and may choose to explore or enroll in a suggested course, further refining their learning path.

Scenario 4: Real-Time Progress Tracking (Reflecting Student Dashboard)

- User: Student
- Description:

1. Accessing Dashboard: Student logs into "Skill Path" and navigates to his/ her personalized Student Dashboard.
2. Overall Progress Overview: Samira immediately sees summary statistics: the total number of courses she's active in, the total number

of lessons she has completed across all courses, her current points, level, and earned badges.

3. Individual Course Progress: She scrolls to the "My Courses Progress" section. For each course she has started, like "Digital Marketing Fundamentals," she sees:
    - The course title.
    - A progress bar visually representing her completion percentage for that specific course.
    - Text indicating "X out of Y lessons completed" for the "Digital Marketing Fundamentals" course.
4. Identifying Areas for Improvement/Next Steps: If Samira's progress in "Digital Marketing Fundamentals" is, for example, 60%, she can see which modules or lessons are pending. The "Continue Learning" button on the course detail page (accessed from the dashboard) would ideally link to her next uncompleted lesson.
5. Reviewing Recent Activity: Samira also sees a list of her "Recently Completed Lessons" and "Recent Quiz Attempts" on the dashboard, helping her recall what she last worked on and review her quiz performance.
6. Motivation: Seeing her progress visually and her achievements (points, badges, level) motivates Student to continue her learning journey.

Scenario 5: Automated Assessments and AI-Assisted Feedback

- User: Instructor (interacting via Django Admin for setup) and Student (taking the quiz)
- Description:

1. Instructor Assigns Quiz: An Instructor uses the Django Admin interface to create a quiz for a specific lesson (e.g., "Python Basics - Module 2 Quiz"). They add Multiple Choice Questions (MCQs) with correct answers defined, and Short Answer Questions (SAQs).
2. Student Takes Quiz: A Student, reaches the quiz within the "Python Basics" course and starts the attempt.
3. Student Submits Quiz: answers all MCQs and provides textual answers for the SAQs, then submits the quiz.
4. System Processes MCQs: The "Skill Path" platform automatically grades all MCQ answers by comparing them against the predefined correct choices.
5. System Processes SAQs (AI Assistance):
   a. For each SAQ Student's answers, the system sends her answer along with the question text (and a pre-defined rubric/prompt) to an OpenAI GPT model via API.
   b. The OpenAI model analyzes SAQ answer and generates:
   i. Constructive textual feedback (e.g., highlighting strengths, areas for improvement, or concepts to revisit).
   ii. A preliminary score (e.g., 7/10) based on the rubric provided in the

prompt.

c. The system stores this AI-generated feedback and preliminary score for Student's SAQ answers. It also flags answers that might need further instructor review (e.g., if the AI score is very low or the AI expressed uncertainty).

6. System Calculates Overall Score: The system calculates an overall preliminary score for Student's quiz attempt, combining the auto-graded MCQ results with the AI-assisted SAQ scores (using a defined strategy, e.g., averaging percentages or a points system).

7. Student Views Results: Student is immediately redirected to her quiz results page. She sees:

    - Her overall preliminary quiz score.
    - A breakdown of her MCQ answers (correct/incorrect, correct options).
    - Her submitted SAQ answers.
    - The AI-generated textual feedback for each SAQ.
    - The AI-generated preliminary score for each SAQ, with a note that it may be subject to instructor review.

8. Instructor Review (Conceptual/Future): The Instructor can later access an admin view (or a dedicated instructor dashboard) to see student quiz attempts, particularly SAQs flagged for review. They can see the

AI's feedback and score, and have the option to override the AI score if necessary, leading to a final adjusted grade for the student.

Scenario 6: Chatbot Assistance

User: Student

Description:

1. **Accessing the Chatbot**: The student opens the AI chatbot widget during learning or navigation.
2. **Typing a Question**: The student submits a query (e.g., "What are the best Python courses?" or "Explain recursion.").
3. **Sending to AI**: The chatbot sends the message, conversation history (via Django sessions), and tool definitions (e.g., `get_courses_by_category`) to the GPT model via API.
4. **AI Processing and Tool Invocation (if applicable)**:
    - If tool use is needed (e.g., course query), the AI makes a tool call.
    - The backend executes the tool and returns the result.
    - The AI integrates the result into a final natural-language response.
5. **Receiving the Response:** The chatbot receives and displays the AI's message using rich formatting (e.g., Markdown for code or lists).

3.4.2. Use case model

The use case model identifies the primary interactions within the system.

Actors:

- Student
- Instructor
- Administrator
- AI

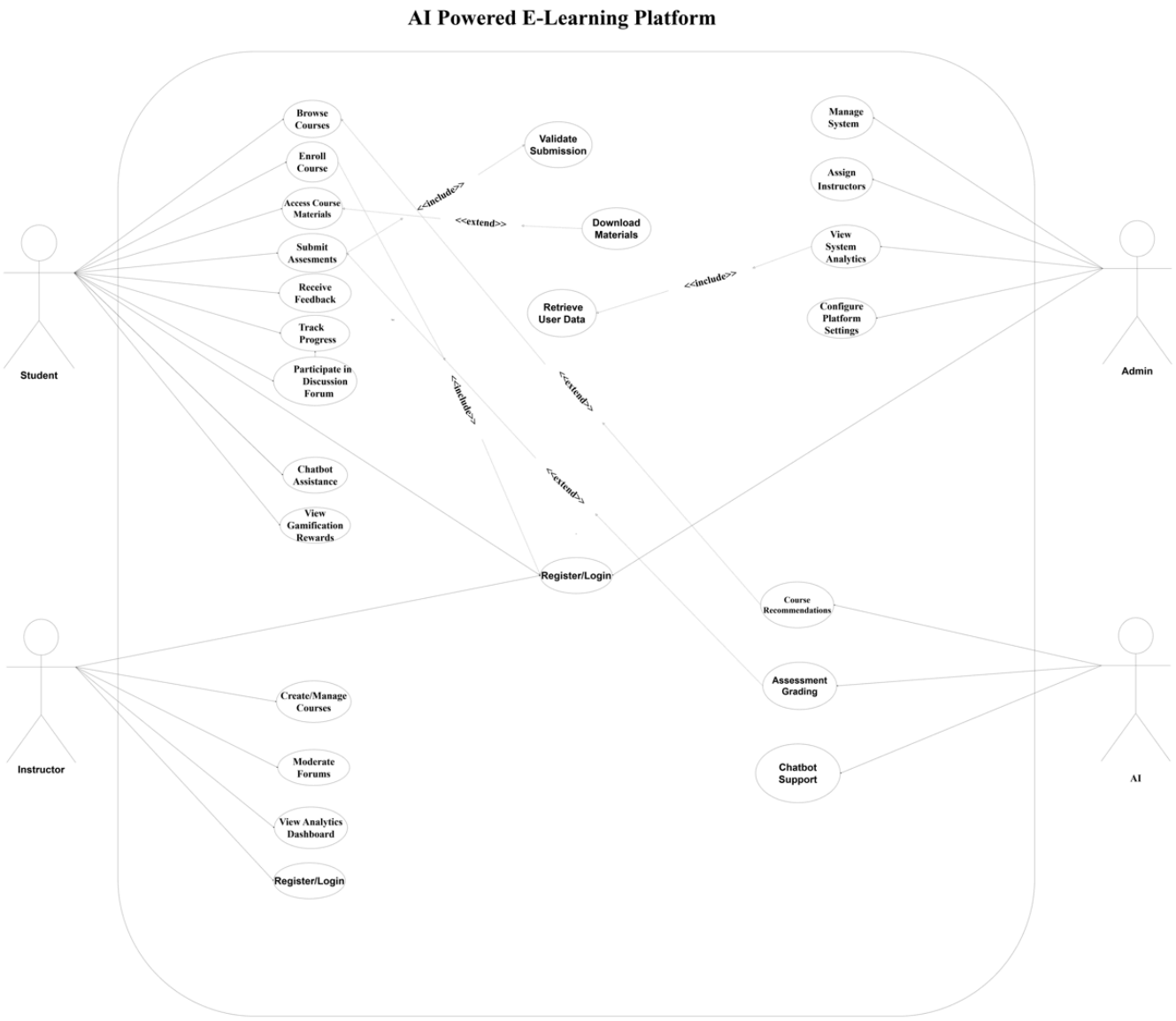

Figure 3.1 Use Case Diagram

3.5. Object Model

3.5.1. Data Dictionary

Field/Property	Data Type	Description	Constraints/Notes
id	AutoField	Primary Key	
password	CharField	Hashed password	
last_login	DateTimeField	Timestamp of last login	Nullable
is_superuser	BooleanField	Grants full admin privileges	Default: False
email	EmailField	Login credential and primary ID	Unique, Max length 254
first_name	CharField	Optional first name	Max length 150, Blank allowed
last_name	CharField	Optional last name	Max length 150, Blank allowed
is_staff	BooleanField	Access to Django Admin instructors)	Default: False
is_active	BooleanField	Active status (controlled via email verification)	Default: False
date_joined	DateTimeField	Date of registration	Default: timezone.now
groups	ManyToManyFiel	Group membership (e.g., Instructors)	Linked to auth.Group
user_permissions	ManyToManyFiel	Individual user permissions	Linked to auth.Permission
profile (reverse)	OneToOneRel	Associated Profile object	
is_instructor_type	Property	Logic to determine instructor role	Returns Boolean
get_full_name()	Method	Returns full name	Returns String

Table 3.1 Data Dictionary

3.5.2. Class diagram

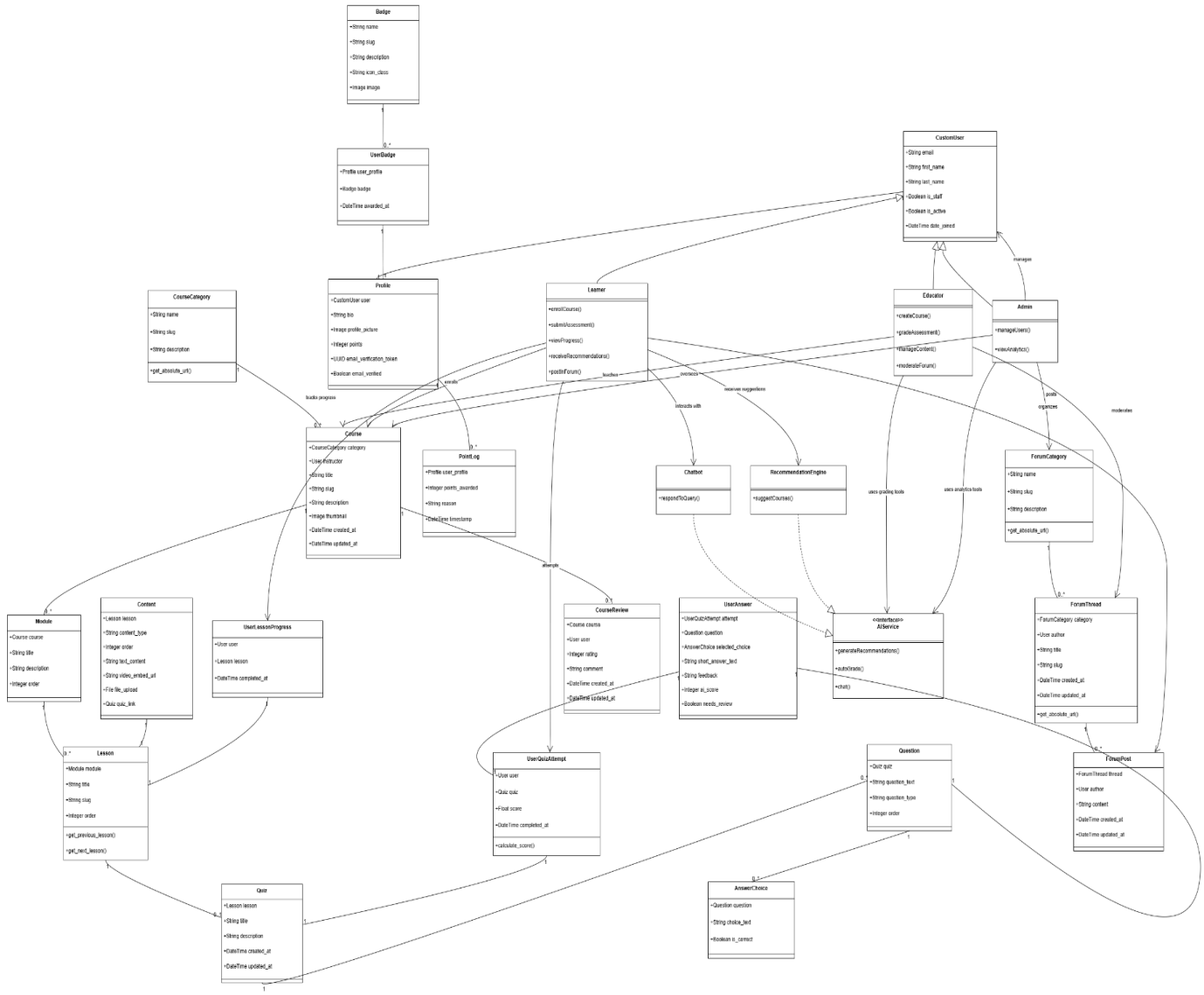

Figure 3.2 Class Diagram

3.6. Dynamic model

3.6.1. Sequence diagram

Sequence Diagram shows the sequence or order of the messages or communications between the objects or components of the system or solution, such as the sender, receiver, message, or time of a system operation or function.

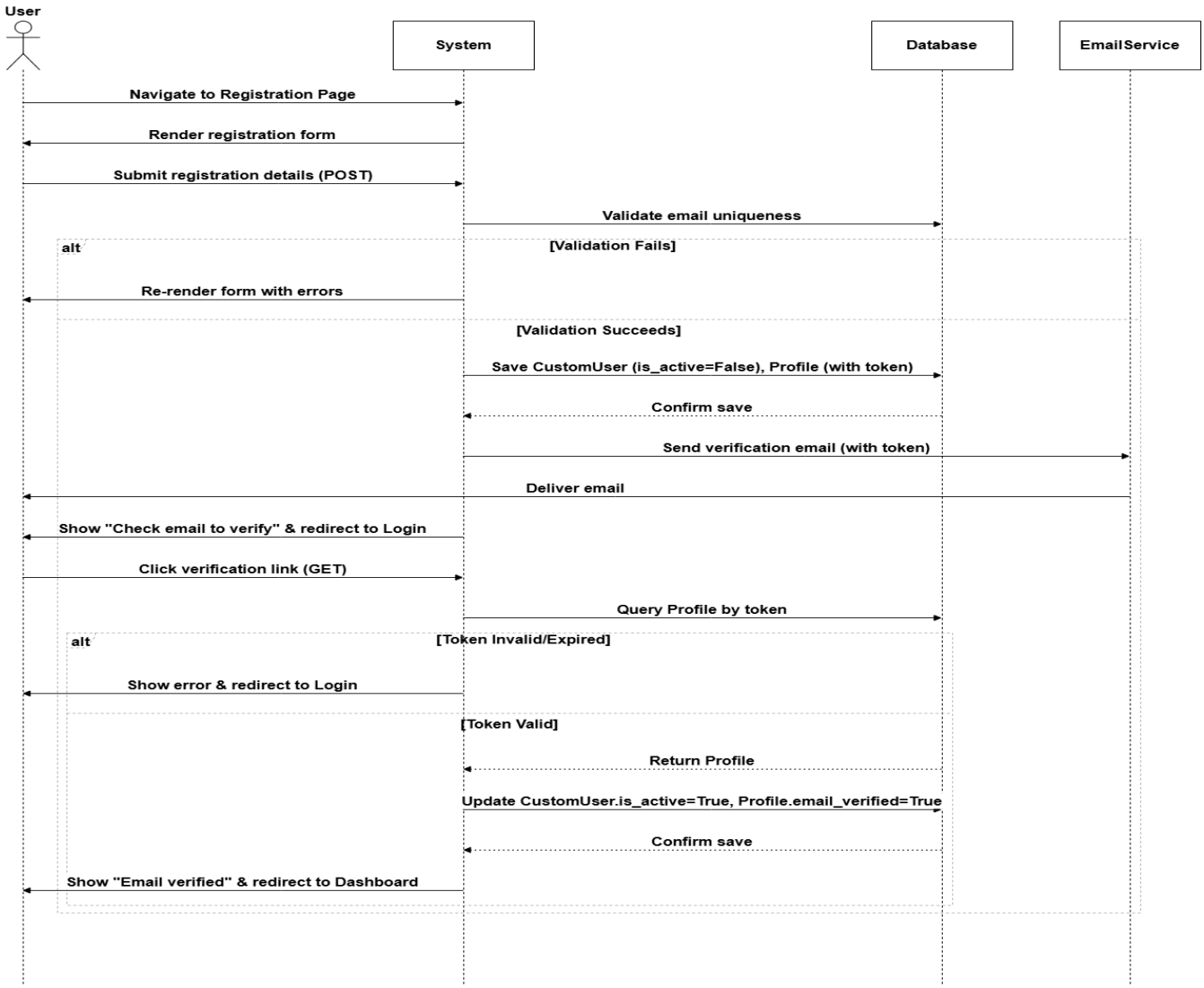

Figure 3.3 Sequence Diagram For Registration

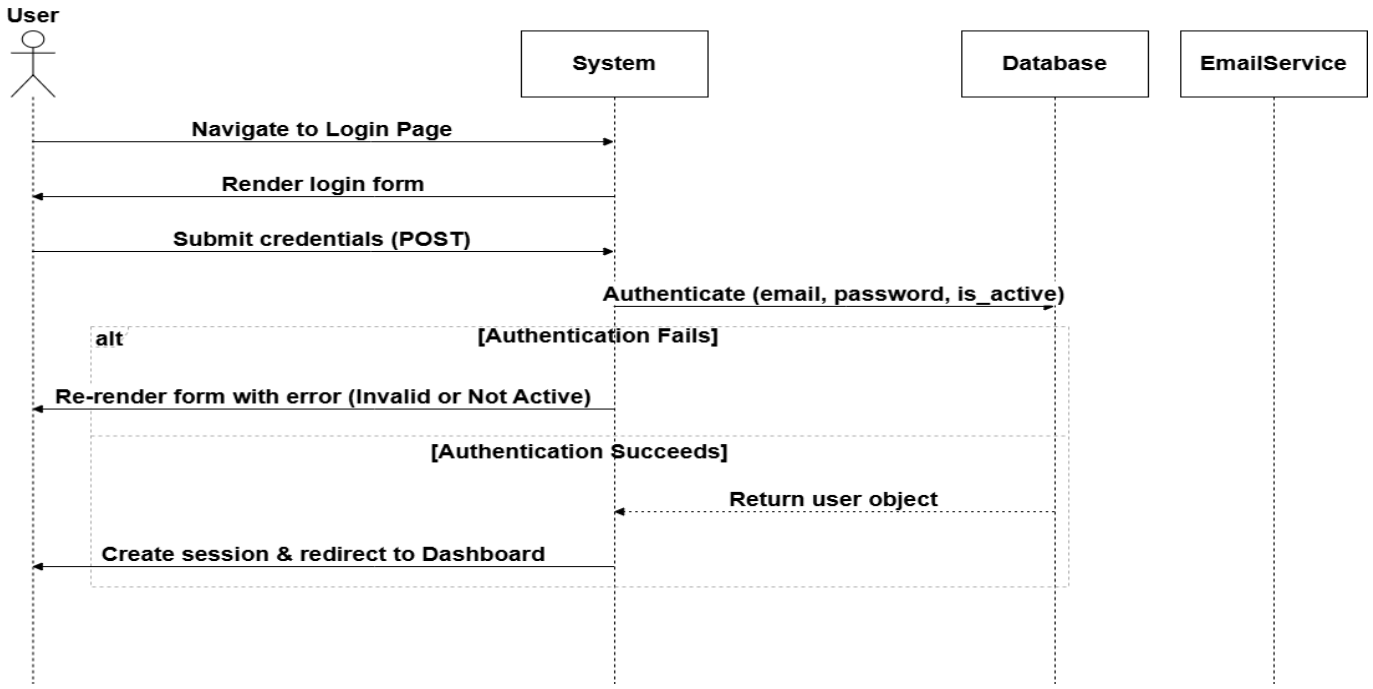

Figure 3.4 Sequence Diagram For Login

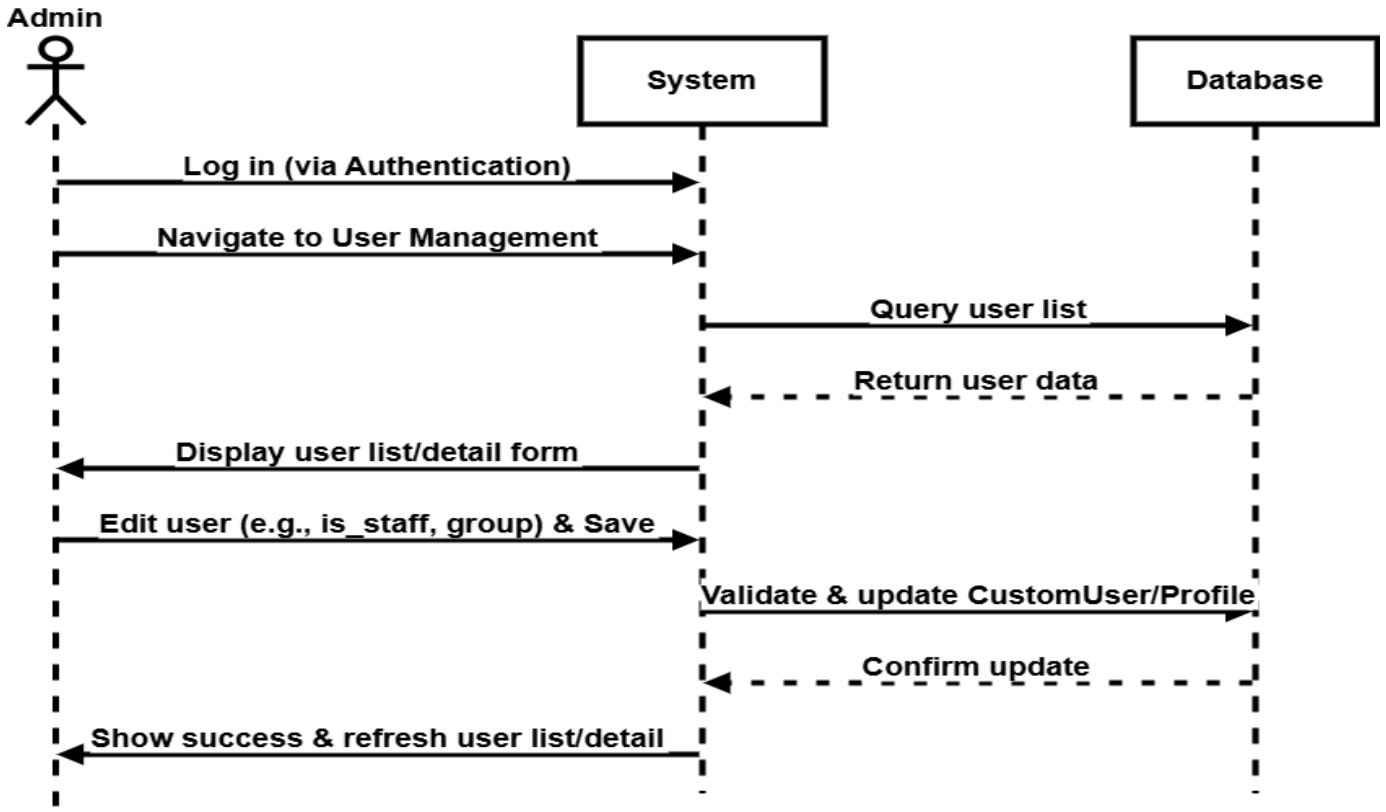

Figure 3.5 User Role Management Sequence Diagram

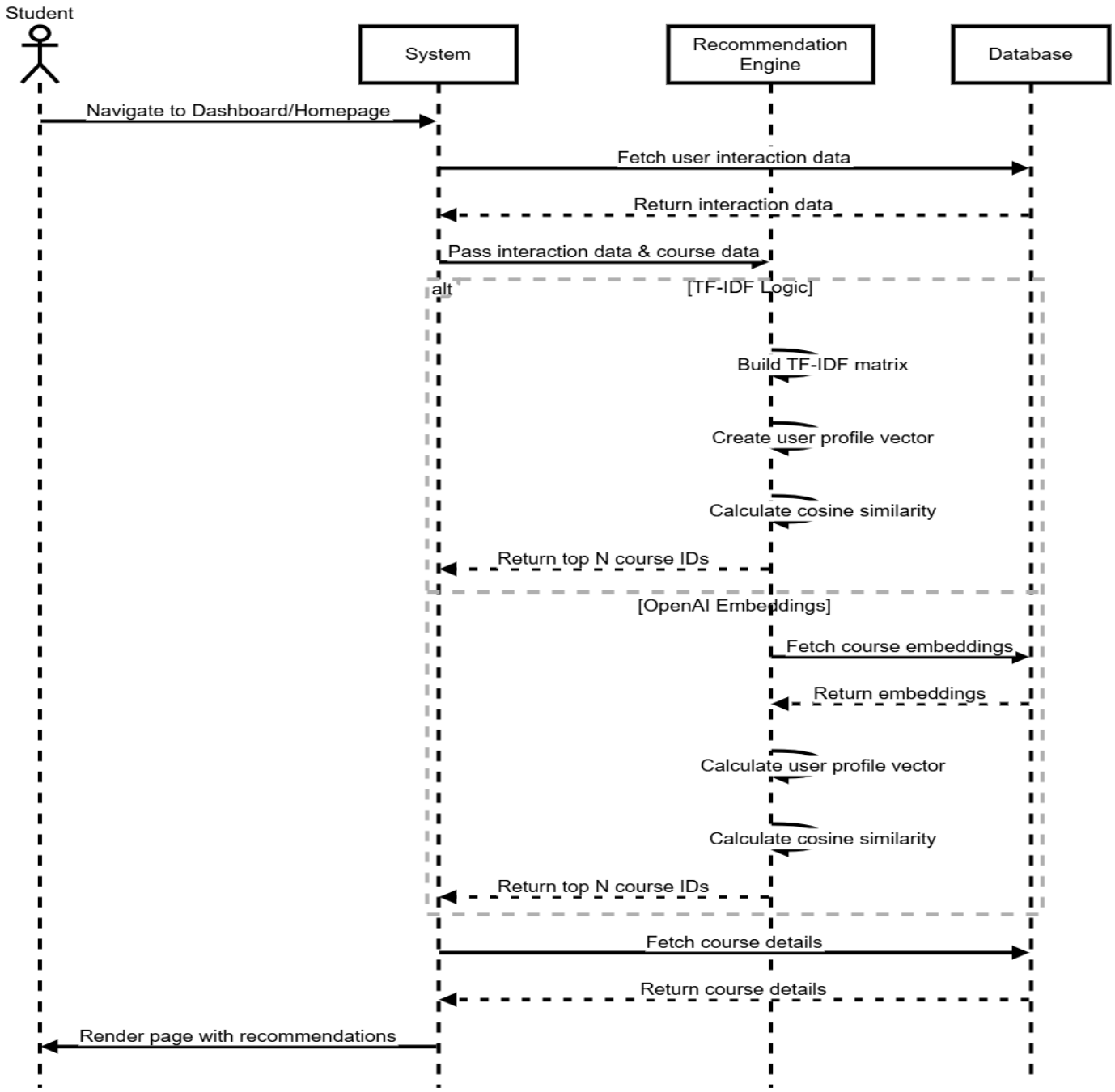

Figure 3.6 Course Recommendation Sequence Diagram

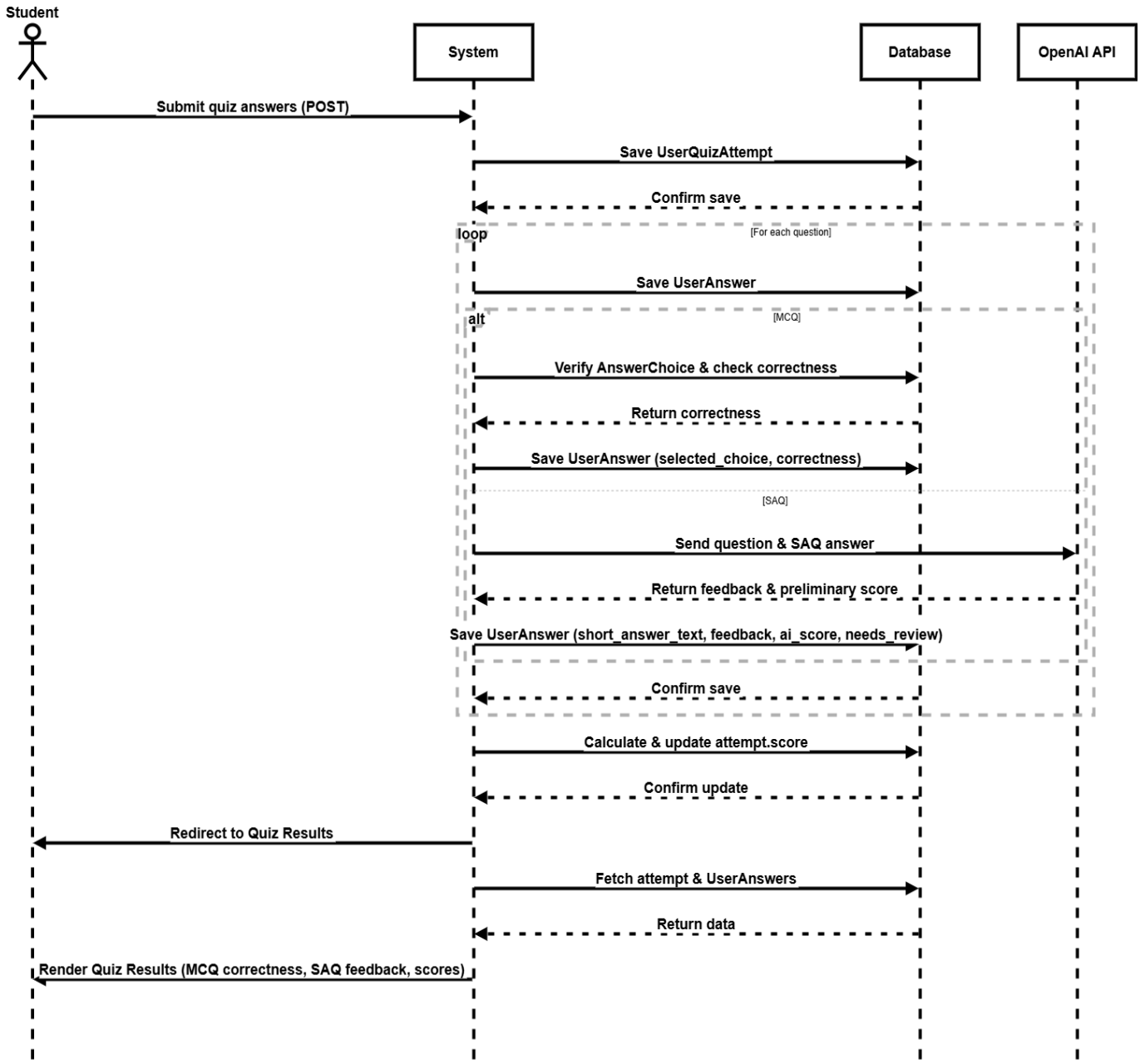

Figure 3.7 Automated Assessment Sequence Diagram

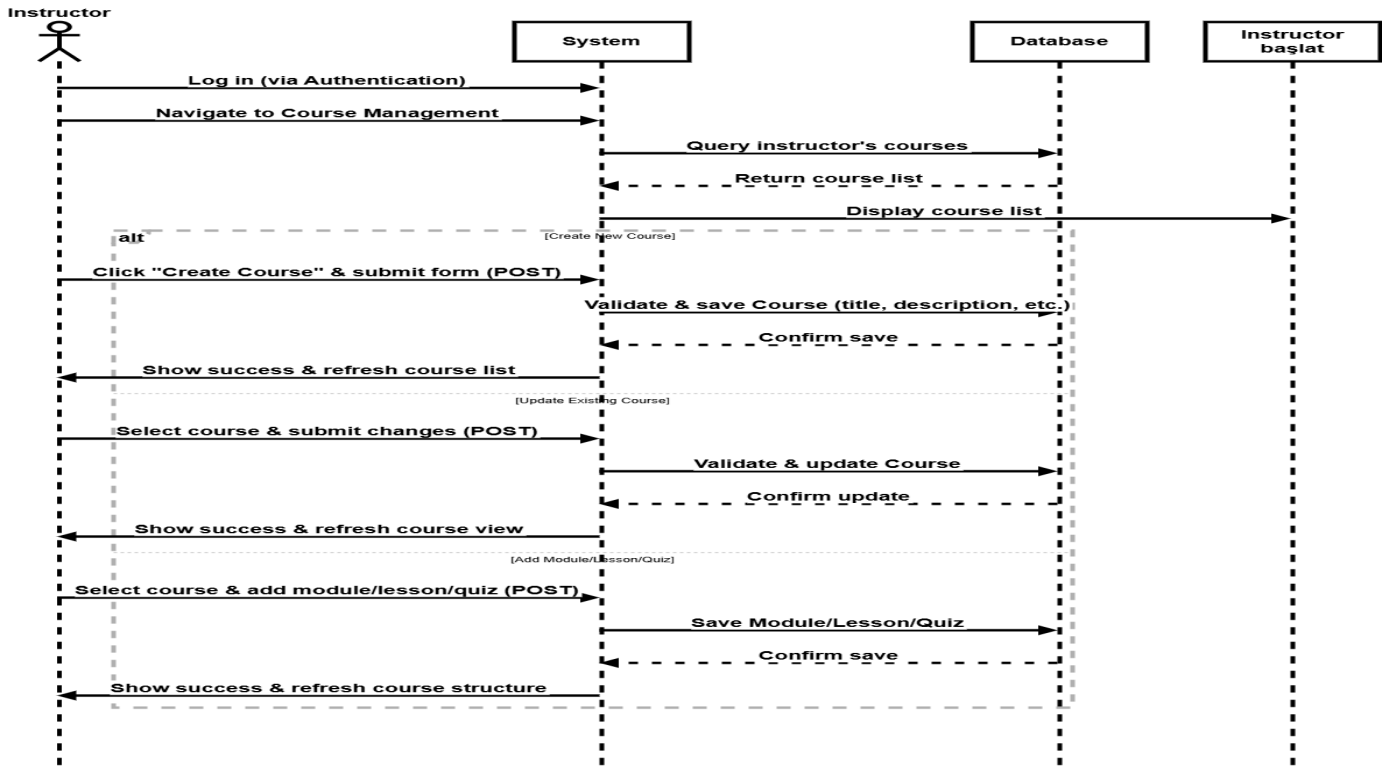

Figure 3.8 Course management Sequence Diagram

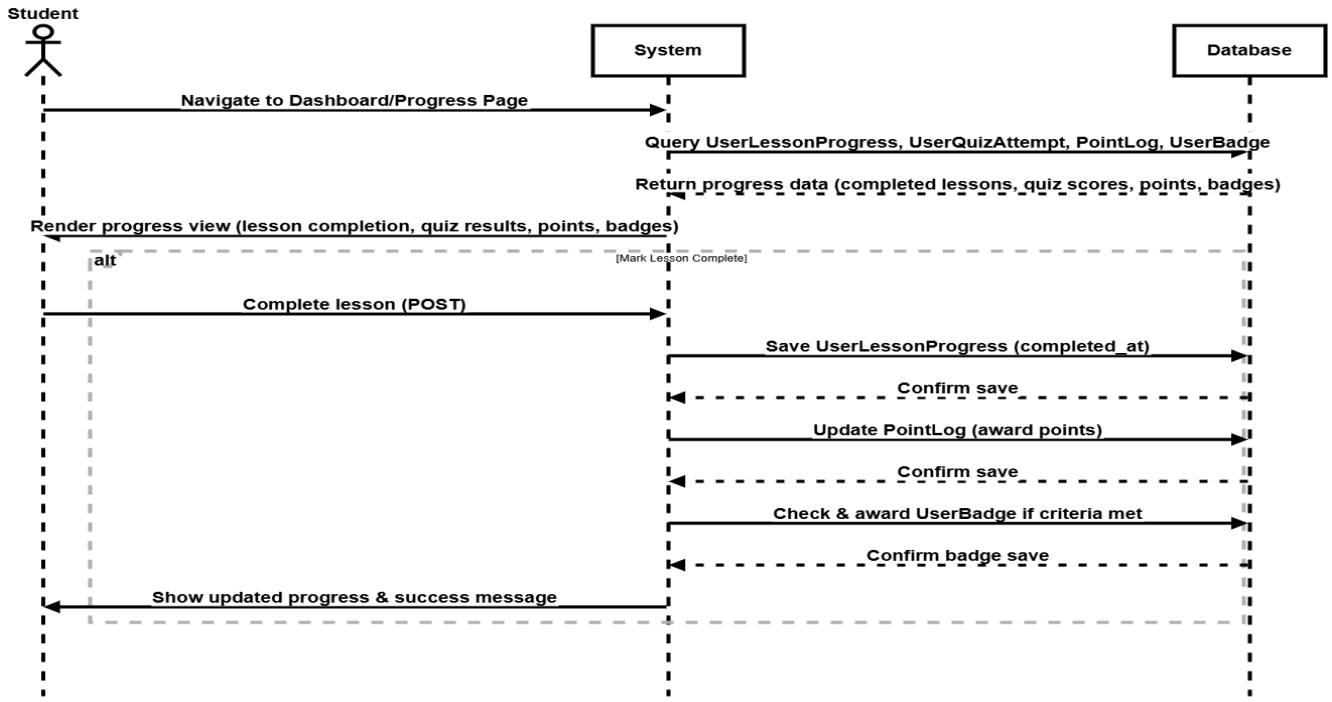

Figure 3.9 Progress Tracking Sequence Diagram

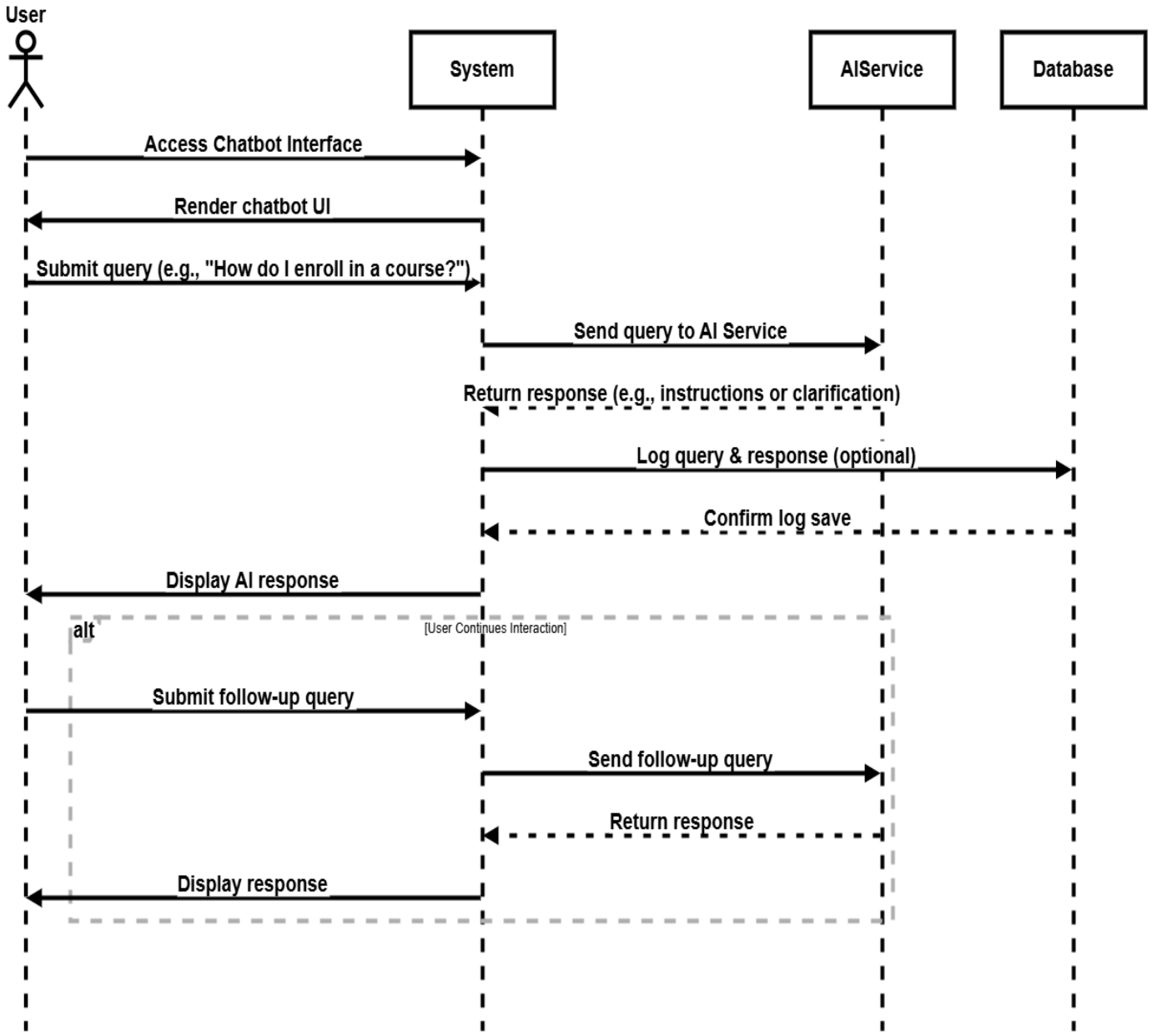

Figure 3.10 Chatbot Support Sequence Diagram

Figure 3.11 Discussion Forum Sequence Diagram

3.6.2. Activity diagram

Activity Diagram shows the activity or flow of the actions or operations of the system or solution, such as the start, end, decision, fork, join, or synchronization of a system process or task.

- The process begins with the user logging into the system.
- The system determines the user's role (student, instructor, or admin) and directs them to the appropriate actions.
- The student View recommendations based on their preferences, Enroll in courses and track their progress, Take quizzes and receive results, followed by providing feedback.
- The instructor creates and upload course content,automates assessments related to their courses.
- Admin Manage users and view user reports.
- End: The process concludes after all actions are completed.

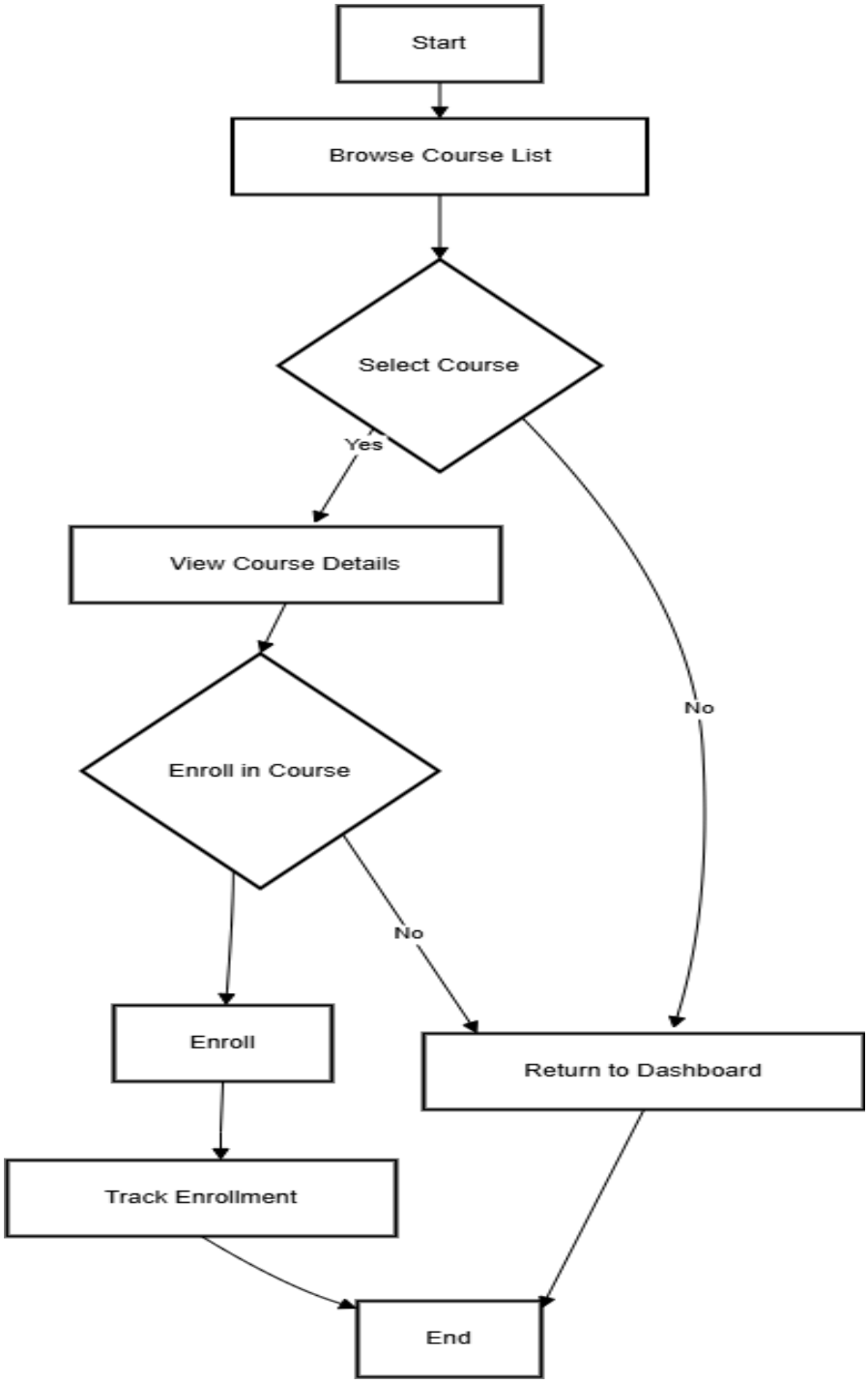

Figure 3.12 Activity Diagram for browse course and enrollment

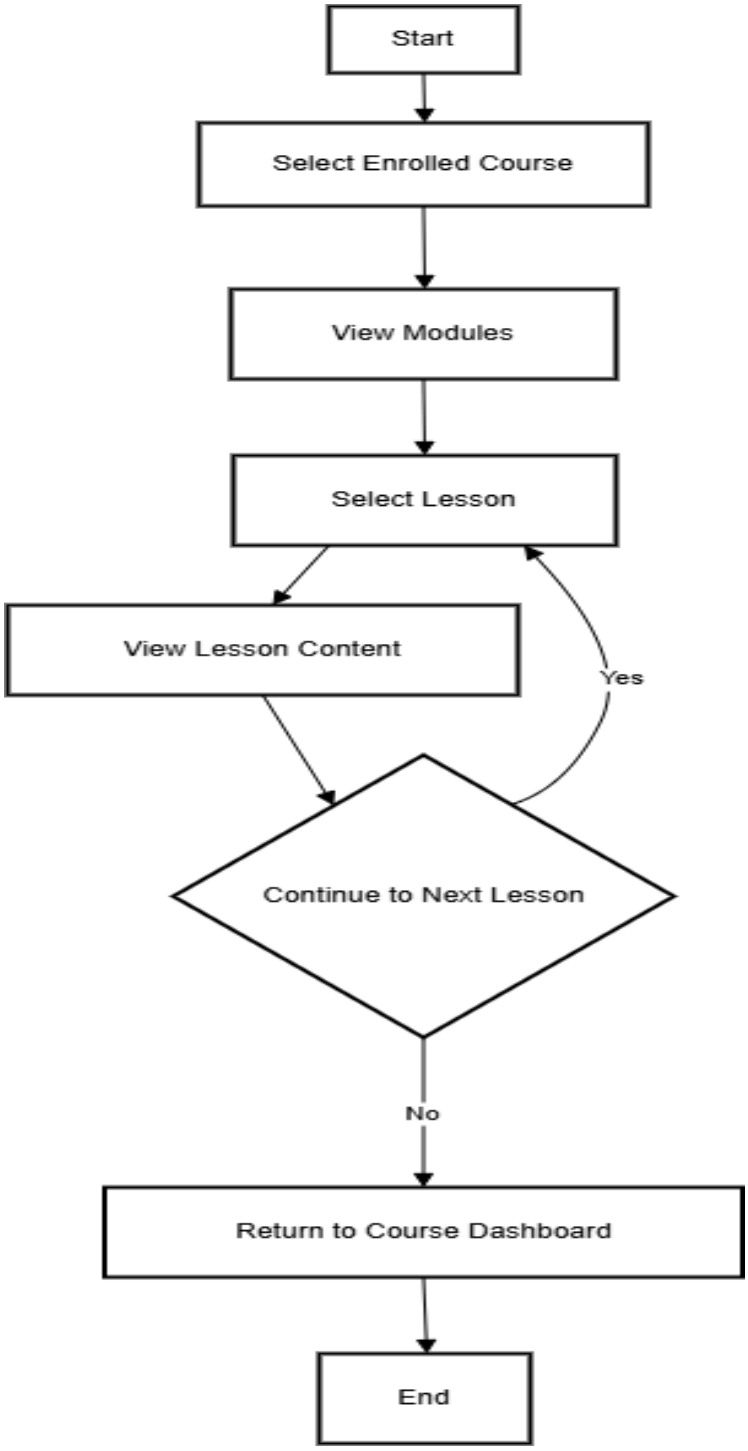

Figure 3.13 Activity Diagram for View course content

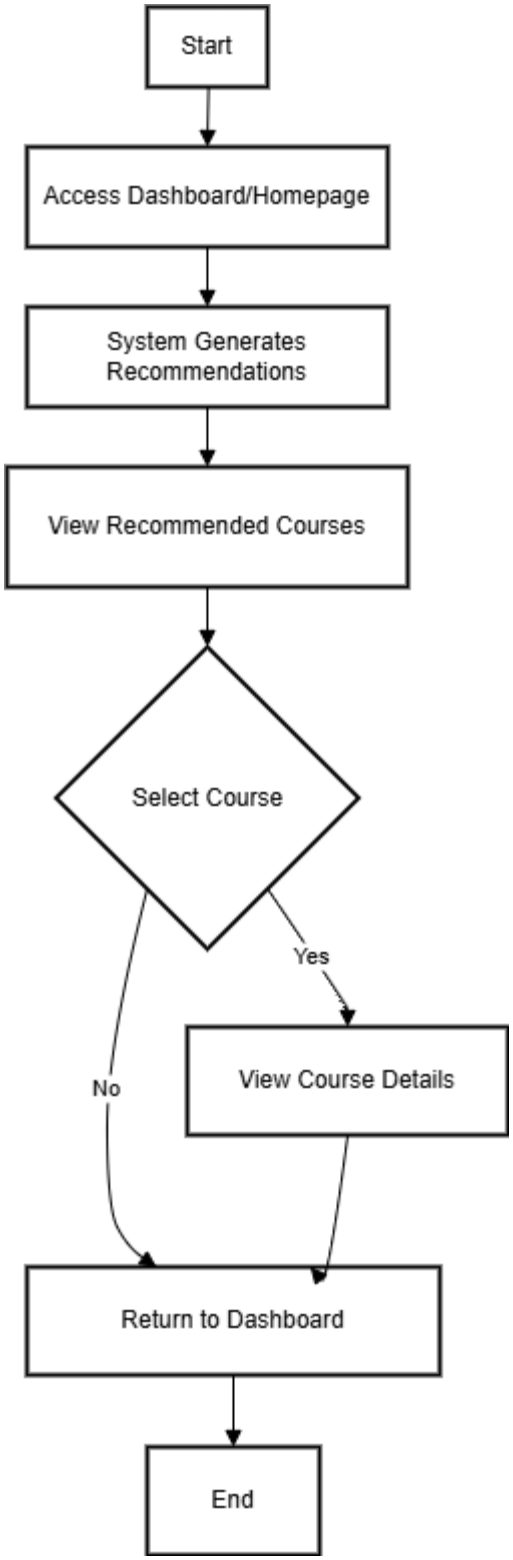

Figure 3.14 Activity Diagram for view recommendation

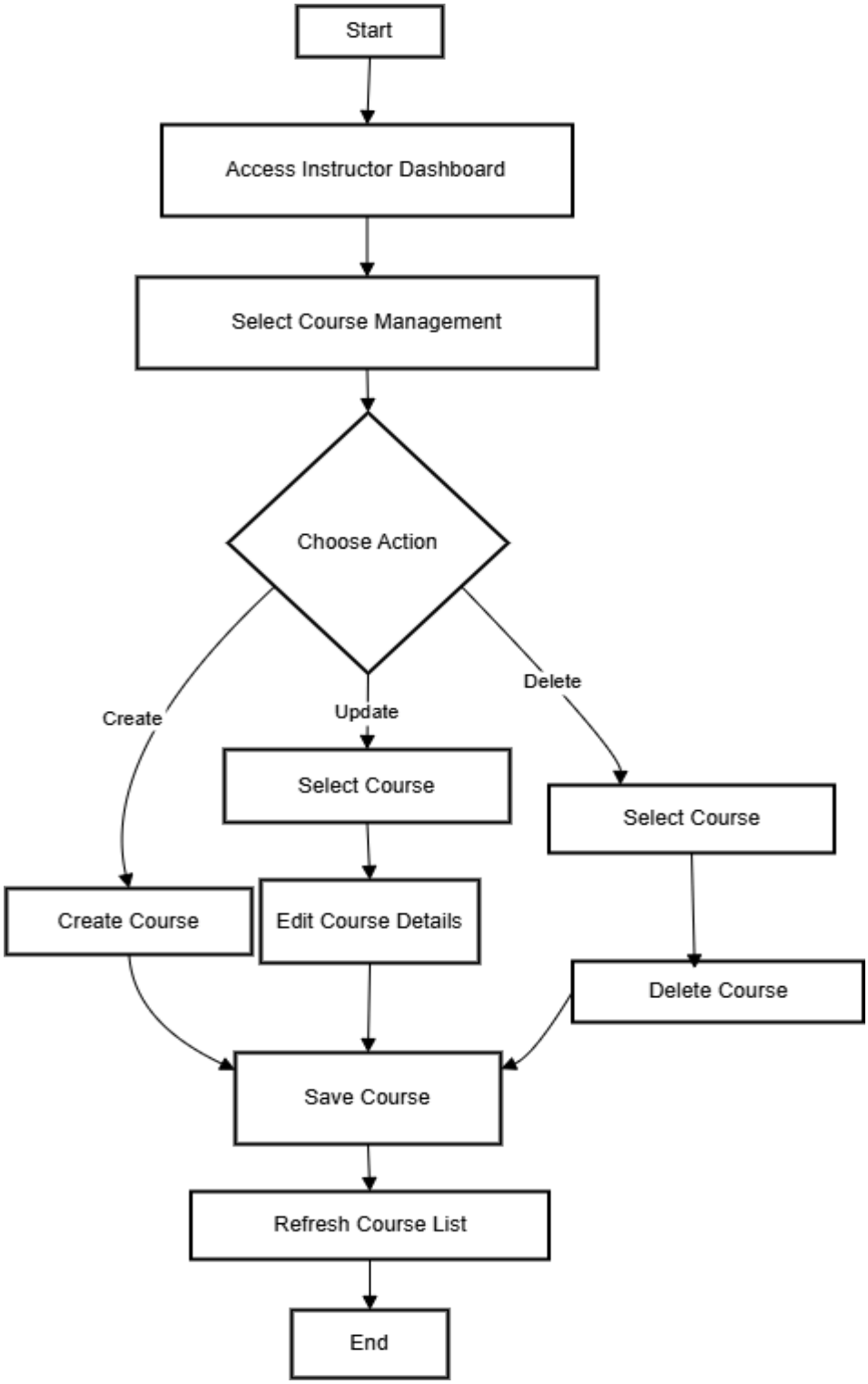

Figure 3.15 Activity Diagram for course management

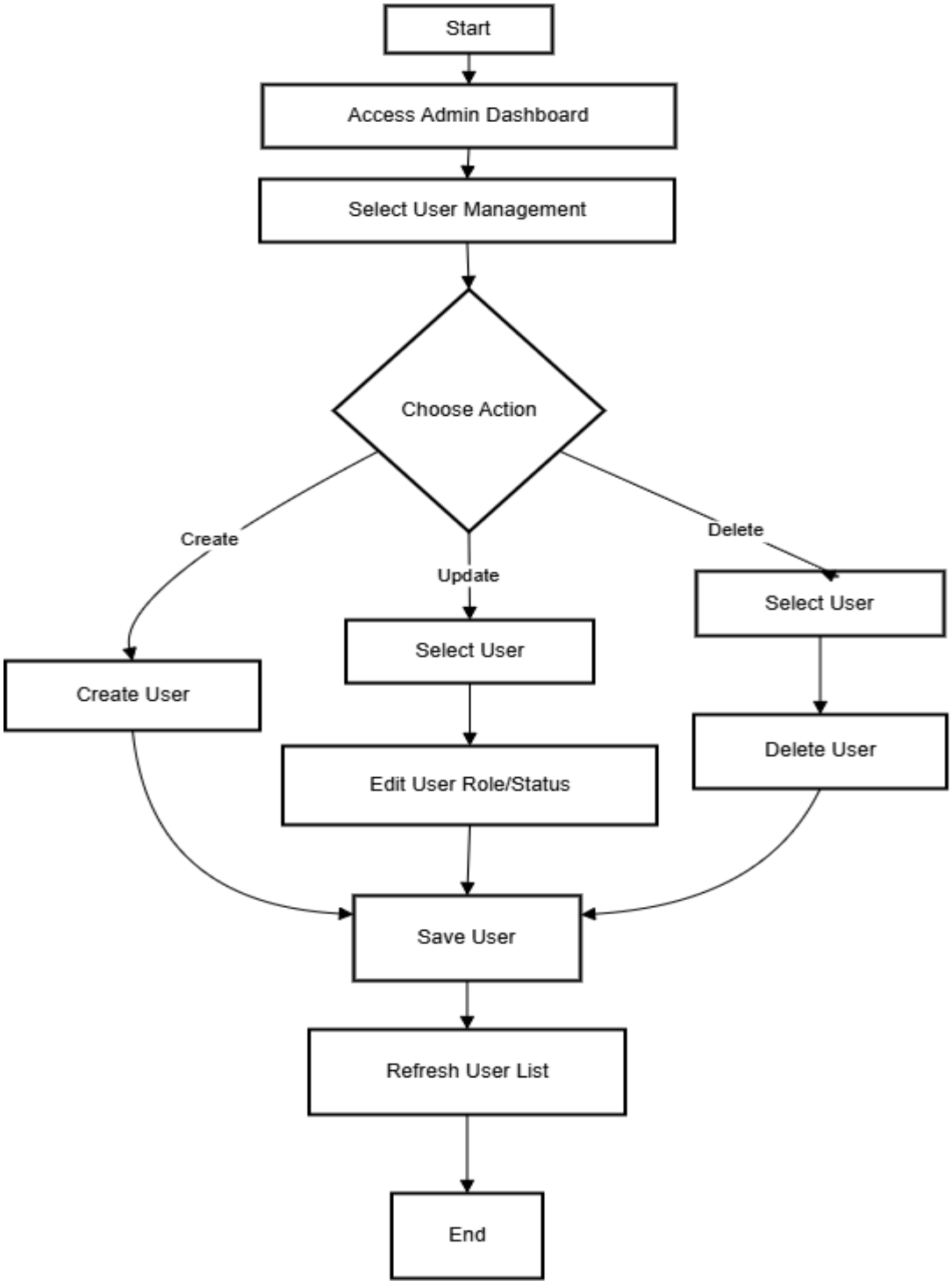

Figure 3.16 Activity Diagram for Manage users

3.6.3. State chart diagram

State chart shows the state or condition of the objects or components of the proposed system or solution, such as the initial, final, active, passive, or transitional state of a system component or feature.

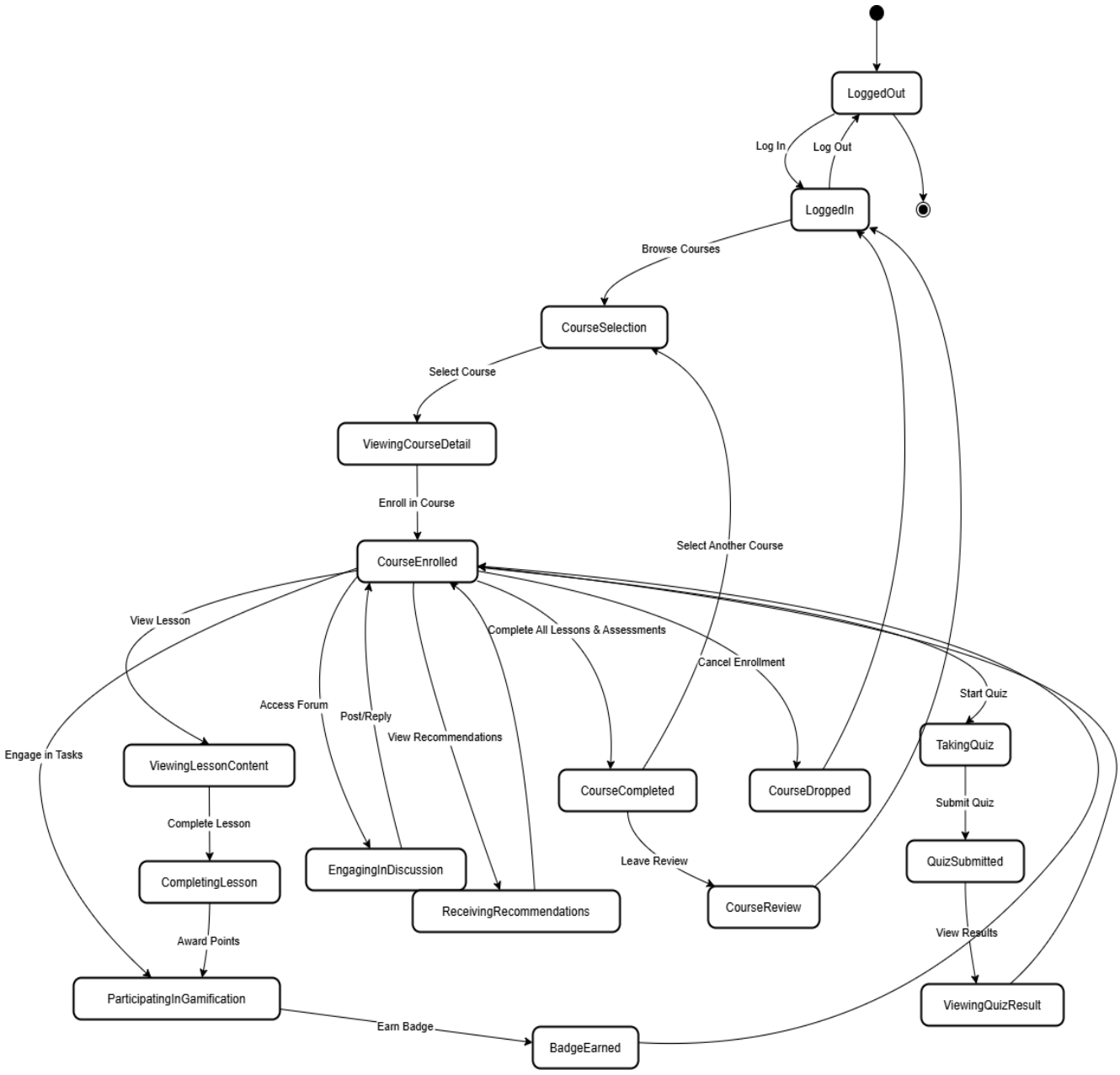

Figure 3.17 State Diagram for student course interaction

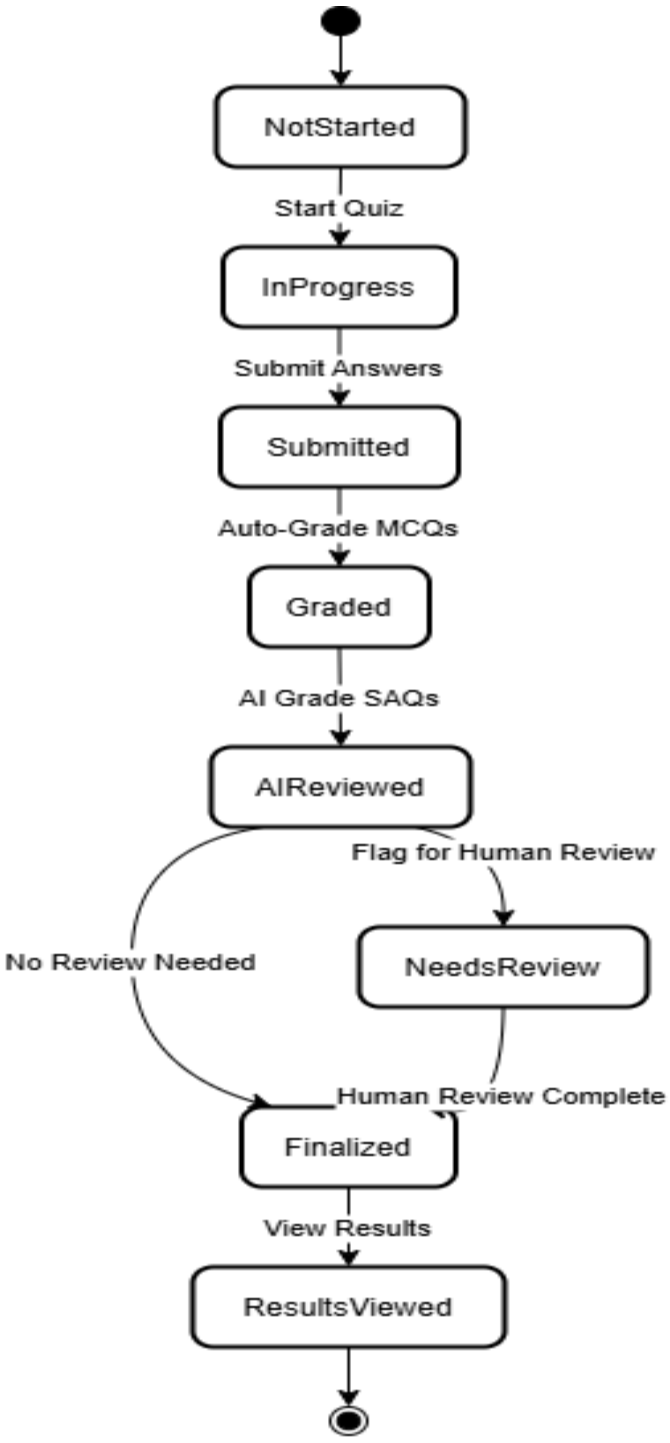

Figure 3.18 State diagram for quiz attempt life cycle

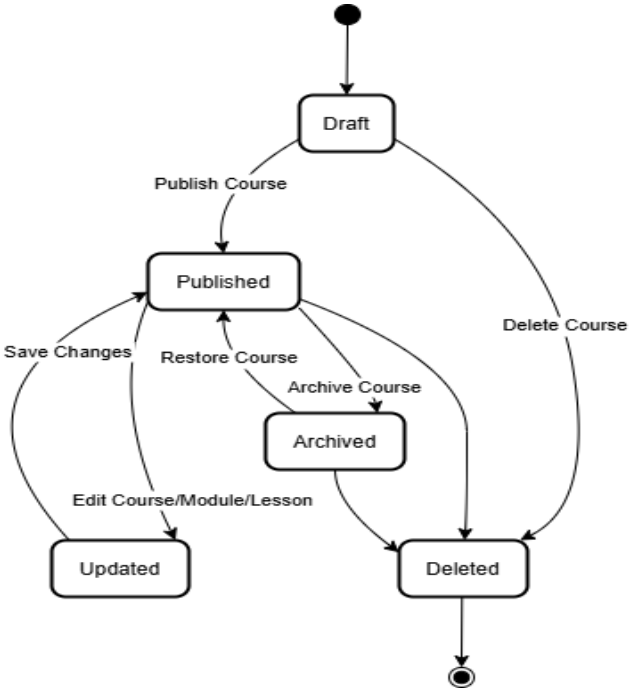

Figure 3.19 State diagram for course management

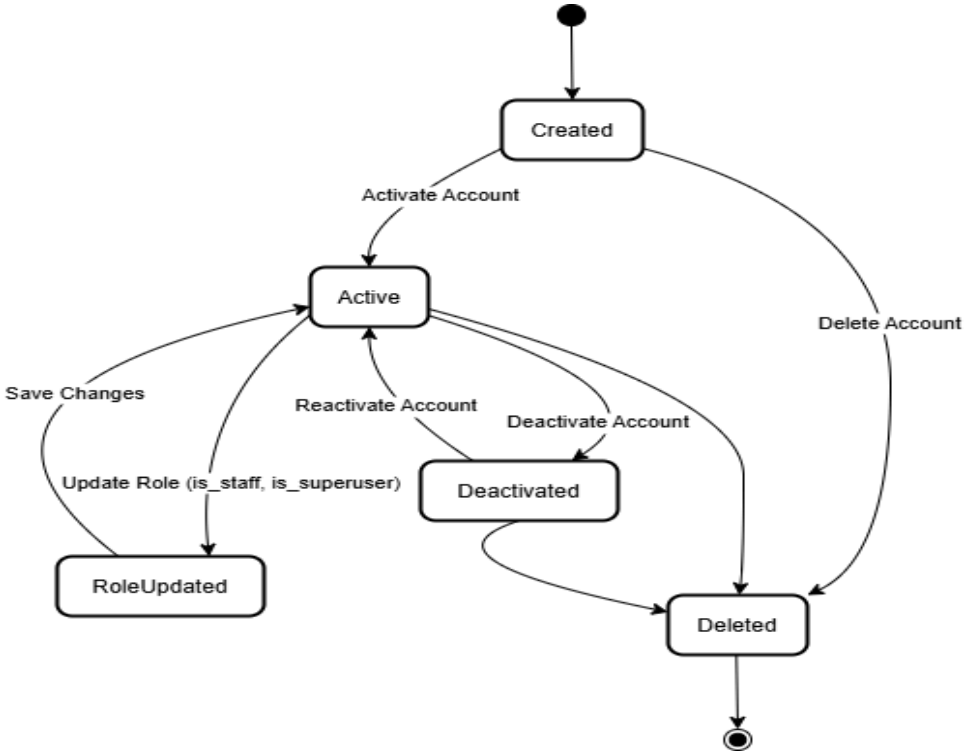

Figure 3.20 State diagram for user account management

Chapter 4: System design

4.1. Overview

The system design for "Skill Path," the AI-Powered E-Learning Platform, translates the defined requirements and objectives into a concrete architectural blueprint. This chapter details the key components, their interactions, interfaces, and the data management strategies employed. The design prioritizes modularity for maintainability and scalability, user-centric interfaces for ease of use, and efficient integration of AI services to deliver an intelligent and responsive learning environment. The architecture is structured to support seamless user interaction, robust backend processing, and dynamic AI-driven automation.

4.2. Purpose of the System Design

The primary purpose of this system design is to provide a comprehensive technical specification that bridges the gap between the project's requirements (as defined in Chapter 3) and its actual implementation (detailed in Chapter 5). It ensures that:

- Functional Requirements such as personalized learning paths, AI-assisted automated grading, interactive chatbot support, and real-time progress tracking are adequately supported by the chosen architecture and components.
- Non-Functional Requirements like performance, scalability, reliability, security, and usability are systematically addressed through specific design choices.

- The System Model (Use Cases, Scenarios) and Object Model (Class Diagram, Data Dictionary) are coherently supported by the proposed architectural framework.
- The Implementation and Testing phases are streamlined through a modular and well-defined component structure, promoting reusability and simplifying integration.
- The design facilitates efficient Data Management for user profiles, course content, progress data, and AI interaction logs.

4.3. Design Goals

The design of "Skill Path" is guided by several key principles to ensure a robust, maintainable, and effective platform:

- **Modularity:** The system is decomposed into distinct, independent modules (Django apps: core, users, courses, forum, and utility modules like ai_utils, tool_functions), each responsible for a specific set of functionalities. This approach enhances manageability, allows for parallel development (if applicable), and simplifies updates or replacements of individual components.
- **Cohesion:** Each module is designed to have high cohesion, meaning its internal elements are strongly related and focused on a single, well-defined purpose (e.g., the users app handles all user authentication and profile logic).
- **Low Coupling:** Dependencies between modules are minimized to ensure that changes in one module have limited impact on others. This is achieved through well-defined interfaces (e.g., Django views acting as controllers,

utility functions) and leveraging Django's signal system where appropriate for decoupled communication.

- **Scalability:** The architecture is designed to support an increasing number of users and courses. This includes using efficient querying (ORM optimizations like select_related, prefetch_related, annotations), and planning for stateless application servers (Gunicorn) that can be scaled horizontally. Caching strategies are employed for frequently accessed data and computationally intensive AI recommendations.

- **Reusability:** Components such as Django forms, template tags, utility functions, and themed UI elements (Bootstrap 5 custom theme) are designed for reusability across the platform to speed up development and ensure consistency.

- **Security:** Strong authentication (custom user model with email verification), authorization(Django permissions, @login_required, @staff_member_required), CSRF protection, XSS prevention, and secure handling of API keys (via environment variables) are integral to the design. HTTPS is enforced in production.

- **User-Friendliness (Usability):** The design prioritizes an intuitive and responsive user interface, catering to users with varying technical skills. Clear navigation, consistent design patterns, and helpful feedback mechanisms are key.

- **Maintainability:** Well-structured code, adherence to Django best practices, modular design, and clear documentation (including this document) aim to make the system easier to understand, debug, and enhance over time.

- **AI Integration Efficiency:** Calls to external AI APIs (OpenAI) are designed to be efficient, with prompt engineering to elicit desired responses, and caching mechanisms for features like recommendations to manage costs and latency. Streaming is used for the chatbot to enhance perceived responsiveness.

4.4. Proposed System Architecture

4.4.1 Overview

"Skill Path" is built on the **Model-Template-View (MTV)** architectural pattern—Django's conventional adaptation of the widely adopted Model-View-Controller (MVC) paradigm. This architecture emphasizes **separation of concerns**, dividing the system into three core components:

- **Model:** Responsible for data management, business logic, and application rules. Directly interacts with the database.
- **Template:** Manages the presentation layer—how data is displayed to the user—utilizing HTML, CSS, and JavaScript.
- **View:** Functions as the request handler, receiving user input, engaging the Model for data operations, and determining which Template should be rendered in response.

This structure supports efficient development and maintenance by enabling isolated modifications to each layer—data logic, business rules, and UI rendering. Additionally, the architecture integrates **AI components** through API calls, typically managed within the **View layer** or designated **utility modules**.

4.4.2 Architectural Components

4.4.2.1 Model(Data Layer)

The **Model** layer encapsulates the application's data structure and core logic. It interfaces with a **SQLite database**, leveraging Django's **Object-Relational Mapper (ORM)** for all interactions. The main models include:

- **`users.CustomUser`** – Handles user authentication details, using email as the primary identifier. Includes fields like `password`, `is_active`, and `is_staff`.
- **`users.Profile`** – Extends user information with fields such as bio, avatar, point tracking, verification status, and email tokens.
- **`courses.CourseCategory`** – Categorizes courses for browsing and filtering.
- **`courses.Course`** – Stores course metadata: title, description, instructor reference, category, and thumbnail.
- **`courses.Module`** – Breaks each course into modular, organized units.
- **`courses.Lesson`** – Represents the instructional content within modules.
- **`courses.Content`** – Manages various types of lesson materials (text blocks, video embeds, file uploads, and quiz links).
- **`courses.Quiz`**, **`courses.Question`**, **`courses.AnswerChoice`** – Define assessments for each course, including both MCQs and SAQs.
- **`courses.UserQuizAttempt`**, **`courses.UserAnswer`** – Track student attempts, submitted answers, and AI-generated SAQ feedback and scoring.
- **`courses.UserLessonProgress`** – Records individual student progress on lessons.

- **`courses.CourseReview`** – Allows students to submit course ratings and written reviews.
- **`courses.Certificate`** – Tracks certificates issued upon successful course completion.
- **`forum.ForumCategory`**, **`forum.ForumThread`**, **`forum.ForumPost`** – Provide the structure and content of the discussion forum.
- **`users.Badge`**, **`users.UserBadge`**, **`users.PointLog`** – Power the gamification system, tracking points and awarding badges.
- **`core.Notification`** – Stores and manages in-app notifications for system events.

Each model not only defines data attributes but may also include **custom methods** implementing business logic—such as `Profile.award_points()` or `Lesson.get_next_lesson()`.

4.4.2.2 Template(Presentation Layer)

The **Template** layer governs the user interface. "Skill Path" uses Django's built-in **templating engine**, enhanced by modern front-end technologies for a dynamic and responsive UI.

- **Bootstrap 5** – Serves as the primary CSS framework, providing responsive design, layout grids, and styled components.
- **Custom Theming** – Applied using CSS variables in `custom.css` to give the platform a distinct visual identity, including support for light/dark mode toggling.
- **JavaScript (Vanilla + Libraries)** – Enables client-side interactions such as:

    - AI chatbot widget: streaming and updating responses
    - Theme toggling functionality
    - AJAX-based dynamic notifications
    - Enhanced forms (e.g., image preview before upload)
    - Third-party libraries (e.g., AOS for scroll animations, Marked.js for Markdown rendering)
- **Django Template Language (DTL)** – Utilized for server-side rendering of dynamic content using template inheritance, reusable components , and control structures.

4.4.2.3 View

The **View** layer serves as the controller in Django's MTV pattern. It processes HTTP requests, manages user interaction, coordinates with Models, and selects the appropriate Templates for response generation.

- **Function-Based Views (FBVs) and Class-Based Views (CBVs)** – A combination may be used, with Django's built-in CBVs handling authentication (login, logout, password reset), while FBVs are employed for custom application views.
- **Business Logic** – Views contain logic for operations such as tracking lesson completion, determining certificate eligibility, submitting reviews, awarding badges, and integrating with AI APIs.
- **Request Handling** – Manages both GET and POST methods, including form validation, session control (for chatbot history), and user authentication.
- **Context Preparation** – Data is queried from Models and passed to Templates via context dictionaries.

- **Redirection and Response Management** – Handles:
    - HTML rendering (via `render`)
    - Redirects (via `redirect`)
    - AJAX responses (via `JsonResponse`)
    - Streaming outputs (via `StreamingHttpResponse` for chatbot)
- **Decorators** – Key decorators used include:
    - `@login_required` – Ensures user is authenticated
    - `@staff_member_required` – Restricts access to staff-only views
    - `@require_POST` – Enforces POST-only access for certain endpoints

4.5. Subsystem Decomposition

The "Skill Path" platform is logically decomposed into several key subsystems, each responsible for a distinct set of functionalities:

1. User Management & Authentication Subsystem: Handles user registration (with email verification), login/logout, password management, profile creation and updates, and role-based access control (Student, Instructor via is_staff, Administrator via is_superuser).
    - Interface: Web forms, Django Admin.
    - Dependencies: users app models, Django auth system.
2. Course & Content Management Subsystem: Allows instructors to create, organize, and manage courses, modules, lessons, and various content types

(text, video, files, quizzes). Facilitates student access and navigation through course material, including lesson locking.

- Interface: Django Admin (for instructors), course/lesson detail web pages (for students).
- Dependencies: courses app models.

3. AI-Powered Recommendation Subsystem: Provides personalized course recommendations to authenticated users on their dashboard.
   - Algorithm: Primarily OpenAI LLM-based (prompting with user history and available courses), with a fallback to newest courses. (TF-IDF was an earlier consideration but has been deprioritized for the primary authenticated user flow).
   - Interface: Student Dashboard.
   - Dependencies: courses models, UserLessonProgress, OpenAI API via core.ai_utils.
4. Assessment & AI Feedback Subsystem: Manages the creation, delivery, and grading of quizzes (MCQs and SAQs). MCQs are auto-graded. SAQs receive AI-generated feedback and preliminary scores via OpenAI API.
   - Interface: Quiz taking pages, quiz result pages, Django Admin (for quiz creation).
   - Dependencies: courses app quiz models, core.ai_utils (for get_saq_feedback).

5. Progress Tracking Subsystem: Monitors and records student progress, including lesson completions, course percentages, and quiz attempts.
    - Interface: Student Dashboard, Instructor Dashboard (course-level progress), Course Detail page.
    - Dependencies: courses app progress models (UserLessonProgress, UserQuizAttempt).
6. AI Chatbot Subsystem: Offers real-time, interactive support to users via a streaming chatbot widget.
    - Algorithm: OpenAI GPT models with Tool Use capabilities.
    - Interface: Chatbot widget UI.
    - Dependencies: Django sessions (for history), core.views.chatbot_api_stream_view, core.ai_utils, core.tool_functions, OpenAI API.
7. Forum & Community Subsystem: Enables peer-to-peer interaction through categorized discussion threads and posts.
    - Interface: Forum web pages.
    - Dependencies: forum app models.
8. Gamification Subsystem: Manages points, badges, levels, and leaderboards to enhance user motivation.
    - Interface: Student Dashboard, Leaderboard page.

- Dependencies: users app gamification models (Profile.points, Badge, UserBadge, PointLog), logic for awarding points/badges.

9. Review & Rating Subsystem: Allows users to rate and review courses.
    - Interface: Course Detail page.
    - Dependencies: courses.CourseReview model.
10. Notification Subsystem: Provides in-app notifications to users for important events.
    - Interface: Navbar notification dropdown.
    - Dependencies: core.Notification model, utility functions for creation.
11. Certification Subsystem: Issues and manages digital certificates for course completion.
    - Interface: Student Dashboard, Certificate view page.
    - Dependencies: courses.Certificate model.

4.6. Subsystem Description

Subsystem Description

Subsystem	Function	Apps	Models	Views / Interfaces
User Management	Registration, login, profiles, roles	users	CustomUser, Profile	Login/Register, Profile, Admin
Course Mgmt	Course/module/lesson CRUD, lesson locking	courses	Course, Module, Lesson, Content, CourseCategory	Course/Lesson Detail, Admin
Recommendations	Personalized suggestions for users	users, courses, core	Course, UserLessonProgress	Student Dashboard (student_dashboard_view)
Assessment & Feedback	Quizzes, MCQ auto-grading, SAQ AI feedback	courses, core	Quiz, Question, AnswerChoice, UserQuizAttempt, UserAnswer	Quiz Detail/Submit/Result
Progress Tracking	Tracks course/lesson completion, quiz scores	courses, users	UserLessonProgress, UserQuizAttempt	Dashboards, Course Detail
AI Chatbot	Support chat, queries, platform tool use	core	(Session History)	Chatbot (chatbot_api_stream_view)
Forum & Community	Threaded discussions, posts	forum	ForumCategory, ForumThread, ForumPost	Forum Pages
Gamification	Points, badges, leaderboard	users	Profile (points), Badge, UserBadge, PointLog	Student Dashboard, Leaderboard
Reviews & Ratings	Course reviews and ratings	courses	CourseReview	Course Detail, Review Form
Notifications	In-app alerts and updates	core	Notification	Navbar Dropdown (get_unread_notifications_api)
Certification	Displaying course certificates	courses	Certificate	Student Dashboard, Certificate View

Table 4.1 Subsystem Description Table

4.7. Persistent Data Management

"Skill Path" utilizes a SQLite relational database. Django's Object-Relational Mapper (ORM) provides a high-level abstraction for all database operations, ensuring data integrity and simplifying queries. Key data entities and their corresponding tables (derived from Django models) include:

- Users & Profiles (Stores user credentials (securely hashed passwords), email, status (active, staff, superuser), personal information (bio, avatar), gamification points, and email verification details.
- Courses & Structure (Stores the educational content, including course details, their categorization, modular structure, individual lessons, and various content types within lessons (text, video links, files, quiz links).
- Assessments (Defines quizzes, their questions (MCQ, SAQ), and the answer choices for MCQs.
- Student Interactions & Progress (Tracks student attempts at quizzes, their specific answers (including AI scores/feedback for SAQs), and their completion status for each lesson.
- Community & Feedback (Manages discussion forum content (categories, threads, posts) and user-submitted reviews and ratings for courses.
- Gamification & Rewards (Stores definitions of available badges, tracks badges earned by users, logs point transactions, and records issued course completion certificates.
- Platform Operations (Manages in-app notifications, user sessions, and administrative logs.

Data relationships (one-to-one, one-to-many, many-to-many) are defined within the Django models using fields like ForeignKey, OneToOneField, ensuring relational integrity.

4.8. Component Diagram

Component Diagram shows the components or modules of the system, such as their ports, interfaces, or connections.

4.8.1 Explanation of the Components

- **User Interface**: The front-end interface through which users interact with the platform.
- **Django Backend**: The server-side application that processes requests and manages business logic.
- **User Management**: Manages user authentication, registration, and profiles.
- **Course Management**: Handles course creation, updates, and enrollment.
- **Assessment Management**: Manages the creation, administration, and grading of assessments.
- **Feedback and Evaluation**: Collects and processes user feedback on courses and instructors.
- **Progress Tracking**: Monitors and records user progress through courses.
- **Recommendation Engine**: Analyzes data to provide personalized course recommendations.
- **Database**: The persistent storage layer for all application data.

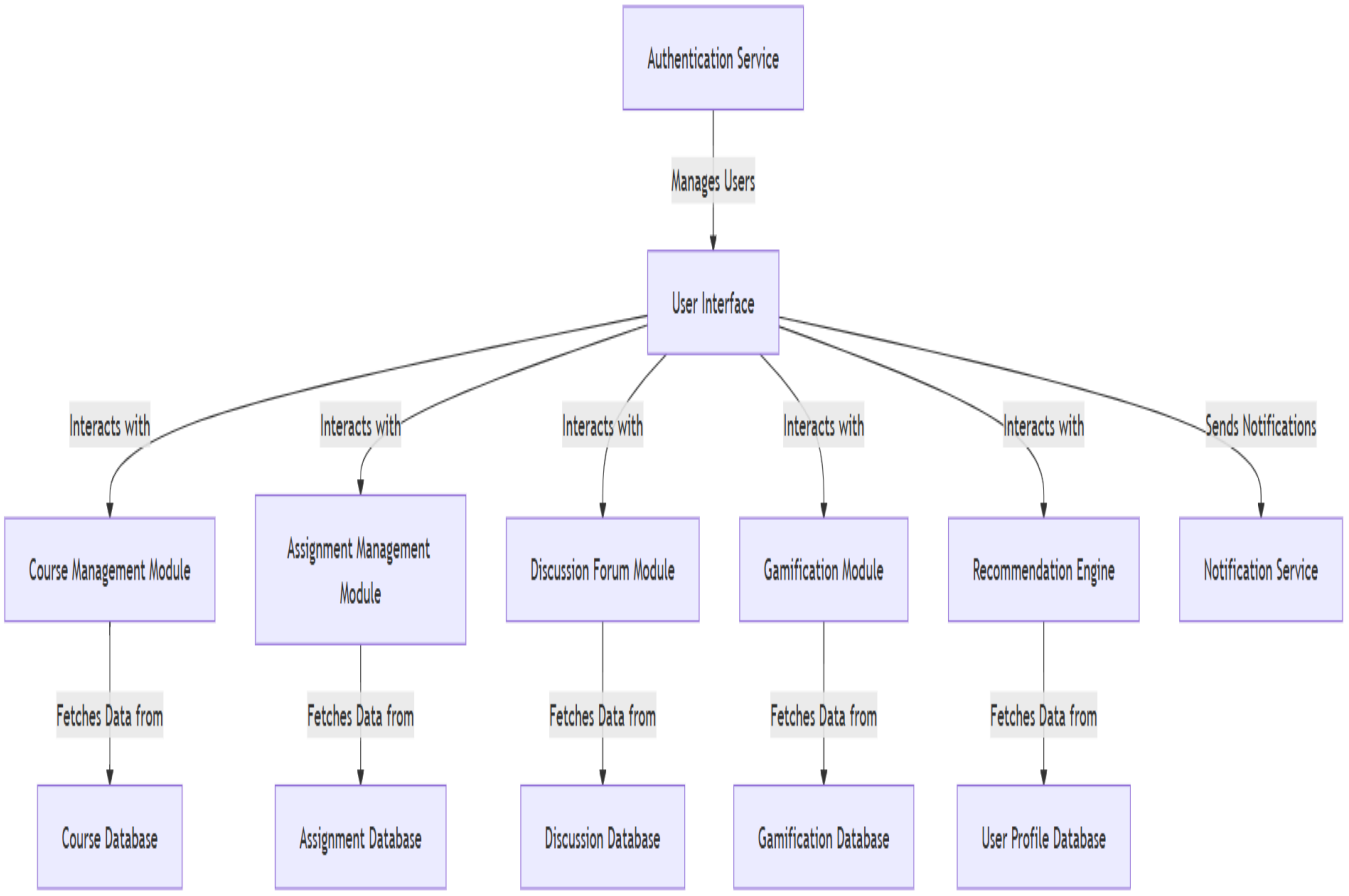

Figure 4.1 Component Diagram

4.9. Database Diagram

The database diagram visually represents the key entities (tables) and their relationships within the "Skill Path" SQLite database. It details the schema derived from the Django models, including primary keys, foreign keys establishing relationships (e.g., a Course has many Modules, a UserAnswer belongs to a UserQuizAttempt and a Question), and key attributes for each entity.

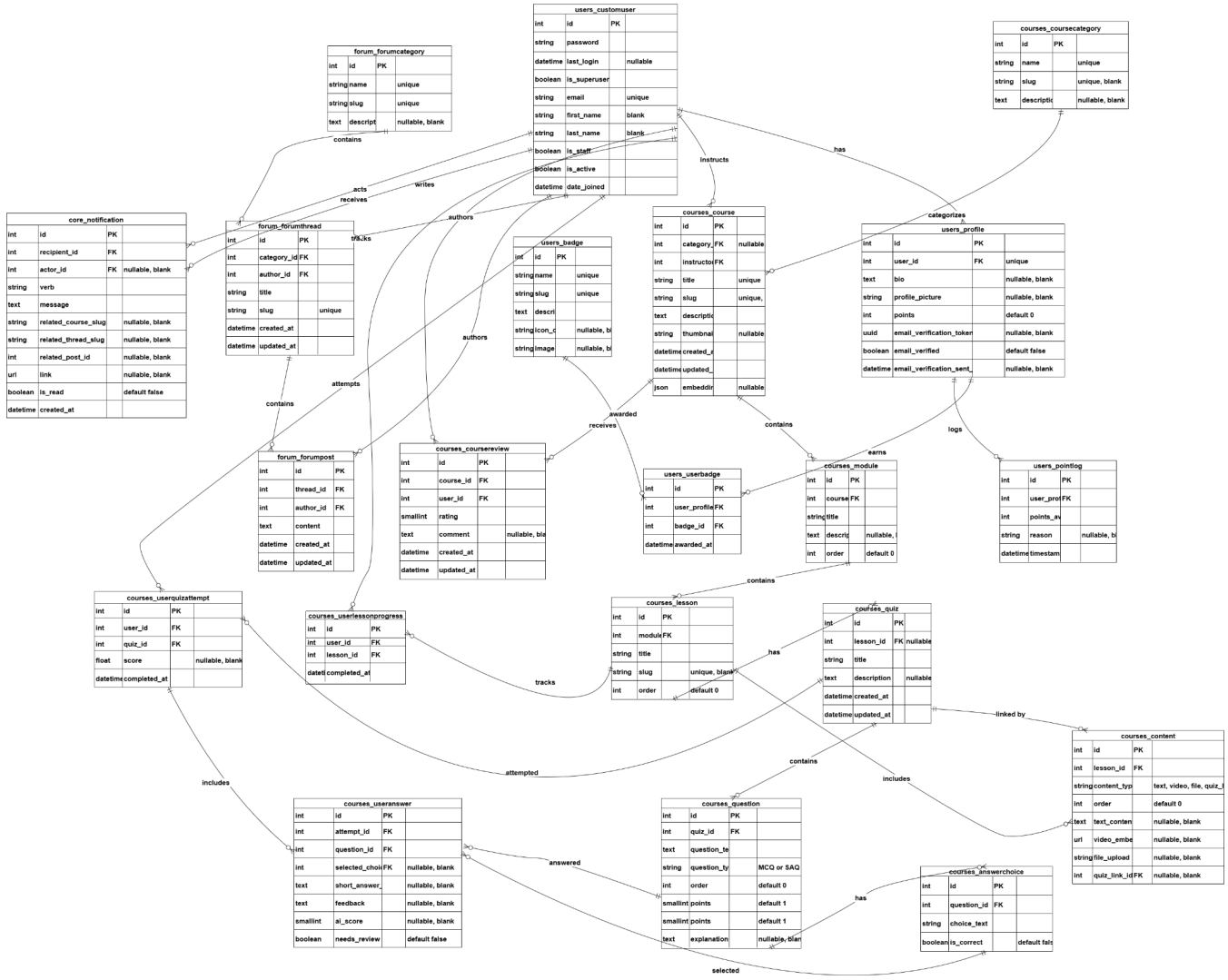

Figure 4.2 Database Diagram

4.10. Access Control

Access control in "Skill Path" is critical for ensuring data security and appropriate user permissions. It leverages Django's built-in authentication and authorization features, tailored for the platform's specific roles:

1. User Roles and Permissions:
    - Students (Authenticated

        - Can register (with email verification), log in, update their profile.
        - Can browse courses, enroll (conceptually by starting lessons), access course materials, submit quizzes, write course reviews.
        - Can view their progress dashboard, earned gamification items, certificates.
        - Can participate in forums (create threads/posts).
        - Can interact with the AI chatbot.
    - Instructors (
        - Can perform all Student actions.
        - Can create, update, and delete courses and their content (modules, lessons, quizzes) primarily via the Django Admin interface.
        - Can view the Instructor Dashboard to monitor their courses and student progress within those courses.
        - (Future) Can manually review/grade SAQs.
    - Administrators
        - Have full control over the platform via the Django Admin interface, including user management (all users), all course and content management, forum moderation, badge creation, etc.
        - Can view the platform-wide Administrator Dashboard.

2. Django Authentication System:
    - Custom User Model (Uses email as the USERNAME_FIELD for login.
    - User Registration: Secure form captures email and password. Passwords are hashed. Email verification is mandatory for account activation (is_active=False until verified).
    - Login/Logout: Standard Django session-based authentication.
    - Password Management: Built-in Django views for secure password reset (via email link) and password change.
3. Authorization Mechanisms:
    - Model Permissions: Django automatically creates add, change, delete, view permissions for each model. These are assigned to the "Instructors" group or directly to staff/superusers via the Django Admin to control CRUD operations on courses, quizzes, etc.
    - View-Level Access Control (Decorators):
        - @login_required: Restricts access to views for authenticated users only (e.g., student dashboard, lesson details, submitting reviews/quizzes, forum posting).
        - @staff_member_required: Restricts access to views intended for staff/instructors and administrators (e.g., admin dashboard, instructor dashboard).
        - Custom permission checks within views (e.g., ensuring an instructor can only edit their own courses in custom views.

Chapter 5: Implementation

5.1 Overview

The implementation phase of the "Skill Path" AI-Powered E-Learning Platform focused on transforming the detailed system design (Chapter 4) into a tangible, functional software product. This crucial phase involved adhering to established coding standards, meticulously utilizing appropriate development tools, and constructing an iterative prototype to validate design assumptions and refine functionality based on progressive development. The implementation was logically divided into three primary components: the client-side (user interface and experience), the server-side (backend logic, database integration, and API development), and the integration of Artificial Intelligence capabilities, primarily through OpenAI APIs. Each component was developed iteratively, often in parallel where feasible, to ensure a cohesive and functional end-to-end system.

5.2 Coding Standard

To maintain high-quality, readable, and maintainable code throughout the project, the development team adhered to the following coding standards and best practices:

- Naming Conventions:
    - Python (Server-Side - Django): Followed PEP 8 guidelines. Variables and function names used snake_case (e.g., student_progress_data, calculate_average_rating). Class names used PascalCase (e.g., CourseCategory, UserLessonProgress). Constants were in UPPER_SNAKE_CASE (e.g., MIN_SAQ_LENGTH).

    - JavaScript (Client-Side): Employed camelCase for variables and functions (e.g., chatbotWidget, sendMessageToAI). Class names (if any custom JS classes were used) would typically follow PascalCase.
    - HTML/CSS: HTML tags and attributes were lowercase. CSS classes used kebab-case (e.g., course-detail-section, chat-header) or followed Bootstrap 5 conventions where applicable.
- Code Structure & Modularity:
    - Django Apps: The project was divided into logical Django apps (core, users, courses, forum) to promote modularity and separation of concerns. Each app contains its own models, views, templates, forms, and URLs.
    - Utility Modules: Helper functions and business logic not specific to a single view were organized into utility files (e.g., core/ai_utils.py, core/tool_functions.py, users/utils.py).
    - Templates: Followed Django's template inheritance ({% extends "base.html" %}) and inclusion ({% include "..." %}) to maintain consistency and reduce redundancy. App-specific templates were namespaced within templates/<app_name>/.
    - Static Files: Organized into static/<app_name>/ directories for app-specific static files and a project-level static/ directory for global assets (like custom.css, images).

- Documentation (Code Comments):
    - Inline comments were added to explain complex logic, non-obvious code sections, and important assumptions.
    - Docstrings were used for functions, classes, and methods to describe their purpose, arguments, and return values, particularly for utility functions and model methods.
- Framework-Specific Standards:
    - Django: Adhered to Django's Model-Template-View (MTV) architecture. Utilized Django ORM best practices (e.g., select_related, prefetch_related, annotate for query optimization). Followed Django's security guidelines.
    - Bootstrap 5: Leveraged Bootstrap's grid system, utility classes, and pre-styled components for responsive design, customized with a unique theme via CSS variables.
- Readability: Code was formatted for readability with consistent indentation (4 spaces for Python), appropriate line breaks, and meaningful variable/function names.
- Version Control (Git): Maintained a clean Git history with descriptive commit messages. Feature development was intended to be done on separate branches before merging into a main development or production branch (though for a solo/small academic team, development might happen directly on a main branch).

5.3 Development Tools

The following tools and software were integral to the development process of "Skill Path":

- **Integrated Development Environment (IDE):**
    - Visual Studio Code (VS Code): Utilized for both client-side (HTML, CSS, JavaScript) and server-side (Python/Django) development due to its flexibility, extensive plugin ecosystem (e.g., Python, Django, Pylint, Prettier), integrated terminal, and Git support.
- **Version Control System:**
    - Git: For tracking changes, managing branches, and collaborating on code.
    - GitHub: As the remote repository for code hosting, backup, and potentially CI/CD integration in future iterations.
- **Backend Framework:**
    - Django (Python): Chosen for its "batteries-included" nature, robust ORM, built-in admin panel (which significantly sped up content management setup), security features, scalability, and extensive community support. Python's readability and vast library ecosystem also made it ideal for AI integrations.
- **Frontend Libraries & Technologies:**
    - HTML5, CSS3: Standard web technologies for structure and presentation.

    - Bootstrap 5: Selected for its comprehensive set of responsive UI components and grid system, enabling rapid development of a mobile-first interface. It was customized using CSS variables to achieve the platform's unique visual theme.
    - JavaScript (Vanilla JS): Implemented for client-side interactivity, particularly for the AI chatbot widget (DOM manipulation, AJAX for streaming), theme toggling, image previews, and enhancing user experience with dynamic elements (e.g., AOS library).
- **AI Integration & Machine Learning Libraries:**
    - OpenAI Python Library: The official client library for interacting with OpenAI APIs (GPT models for chatbot and SAQ feedback).
- **API Testing Tool:**
    - **Postman**: Potentially used for testing the chatbot's streaming API endpoint (chatbot_api_stream_view) independently during development.
- **Deployment Platform & Tools:**
    - Render: Chosen as the PaaS for deploying the web application and SQLite database, simplifying infrastructure management.
    - Gunicorn: The WSGI HTTP server used to run the Django application in the production environment on Render.
    - Whitenoise: Used for serving static files efficiently directly from the Django application in production.

    - python-dotenv: For managing environment variables during local development.
    - dj_database_url: For parsing database connection URLs from environment variables.

5.4 Prototype

The development of "Skill Path" followed an iterative prototyping approach. Initial prototypes focused on core functionalities, which were then progressively enhanced with more features and UI refinements based on the evolving design and simulated user feedback.

- **Initial Prototype (Phase 1 Focus):**
    - Basic user registration (username/password initially, later refined to email-centric with verification), login, and logout functionality using Django's built-in auth system.
    - Core Django models for Course, Module, and Lesson were established, with basic content display using unstyled or minimally styled Bootstrap templates.
    - Navigation was rudimentary, focusing on accessing the course list and individual course/lesson pages.
- **Iteration 2 (Adding AI & Core Interactions - Phase 2 & 3 Focus):**
    - The AI chatbot widget was introduced, initially with synchronous responses, then iterated to streaming with history.

    - Quiz functionality (MCQ grading, SAQ submission) was built. AI-assisted feedback for SAQs was integrated.
    - Student and Admin dashboards were created with initial statistics.
    - The custom Bootstrap theme with light/dark modes started to take shape, influencing the look of new components.

- **Iteration 3 (Engagement Features & UI Polish - Phase 4 & 5 Focus):**
    - The discussion forum, gamification system (points, badges, levels, leaderboard), course reviews, and certificates were implemented.
    - Significant UI/UX polish was applied across all pages, refining the custom theme, ensuring responsiveness, and improving navigation (e.g., lesson locking, "Continue Learning" buttons).
    - The OpenAI-based recommendation system (using LLM prompting) was developed as the primary recommendation engine for authenticated users.
    - The instructor dashboard was enhanced.
- **Final Prototype (Current State)**: The platform now reflects a comprehensive set of features with a consistent, custom-themed user interface, as detailed throughout this document. Screenshots in subsequent sections (or an appendix) will illustrate the final state of key pages like the landing page, student dashboard, course detail page, and chatbot interface.

5.5 Implementation Detail

5.5.1 Client-Side

The client-side of "Skill Path" focuses on delivering a responsive, intuitive, and visually appealing interface that interacts seamlessly with the Django backend.

- **Structure (HTML5):**

  Pages are built with semantic HTML5 to ensure accessibility and proper content hierarchy. The Django Template Language (DTL) is used to dynamically inject backend data into templates. Template inheritance (via base.html) and {% include %} blocks are extensively used for layout consistency across pages.

- **Styling (CSS3 & Bootstrap 5):**

  - Bootstrap 5 provides the foundational styling framework, offering a responsive grid system and a suite of components (e.g., navbar, modals, cards, forms, accordions).

  - A custom theme is layered on top of Bootstrap using CSS custom properties defined in static/css/custom.css. This allows flexible theming (light/dark modes) and consistent branding through variables like --primary-color, --accent-color, and classes such as glass-card and btn-accent.

  - Theme switching is handled via JavaScript by toggling the data-theme attribute on the <html> tag, dynamically applying the respective CSS variable overrides.

- **Interactivity (JavaScript - Vanilla JS):**
    - AI Chatbot Widget: Handles widget toggling, user input capture, and asynchronous communication with the chatbot_api_stream_view. Responses are streamed as newline-delimited JSON, parsed, and rendered with Markdown support using Marked.js. It also manages scroll locking and suggestion button handling.
    - Theme Toggle: Manages light/dark mode switching and persistence using localStorage.
    - Image Previews: Provides real-time previews for profile and course thumbnail uploads.
    - AOS (Animate On Scroll): Adds subtle animations to elements upon entering the viewport, enhancing visual polish.
    - Bootstrap JS Components: Enables interactivity for dropdowns, modals, and accordions through Bootstrap's built-in JavaScript.
- **User Interface Design:**
  The design prioritizes usability with responsive layouts, intuitive navigation, and modern aesthetics. Elements of glass morphism and neumorphism are incorporated via the custom theme for a distinctive UI.

5.5.2 Server-Side

The backend of "Skill Path" is built using Django's MTV architecture, managing all business logic, database operations, user sessions, and AI integrations.

- **Django Framework (MTV Architecture):**

    - **Models (`models.py`):**

      Defined in each app to represent core data (e.g., `CustomUser`, `Course`, `Lesson`, `Quiz`, `ForumPost`, `Notification`). Models leverage Django ORM and encapsulate related logic (e.g., `Profile.award_points()`, `Lesson.get_next_lesson()`).

    - **Views (`views.py`):**

      Handle incoming HTTP requests, form processing, CRUD operations, and API orchestration. Views may be function-based or use Django's class-based views (e.g., for authentication). Access control is enforced using decorators like `@login_required` and `@staff_member_required`.

    - **Templates (`templates/`):**

      HTML files enhanced with DTL for dynamic content rendering.

    - **Forms (`forms.py`):**

      Django forms (`Form`, `ModelForm`) are used for input handling and validation, including user registration, profile editing, reviews, and quiz submissions. `django-crispy-forms` with `crispy-bootstrap5` was optionally used, but manual HTML rendering is preferred for custom styling.

- **URLs (`urls.py`):**

  Configures URL routing at both the project and app levels. Namespaces help organize app-specific routes.

- **Database (SQLite):**

  All database interactions are abstracted via Django ORM. Schema changes are managed via Django migrations.

- **Session Management:**

  Django sessions store user authentication state and chatbot conversation history.

- **APIs:**

  Although not built with Django REST Framework, a custom endpoint (`chatbot_api_stream_view`) supports the AI chatbot. It returns newline-delimited JSON using `StreamingHttpResponse` for real-time interaction.

- **Security:**

  Security best practices are followed using Django's built-in protections: CSRF, XSS (via template auto-escaping), password hashing, and email verification. API keys and secrets are stored securely using `python-dotenv` locally and environment settings in Render for production.

- **Static & Media Files:**

  Static files are served using Whitenoise in production. Media files (user uploads) are stored locally in development, with plans for S3-based cloud storage in production.

Chapter 6: System Testing

6.1. Introduction

System testing is an indispensable phase in the software development life cycle of the "Skill Path" AI-Powered E-Learning Platform. Its primary objective is to verify that all integrated components of the software function correctly together and meet the specified functional, non-functional, and AI-integration requirements. This chapter details the testing strategy employed, the types of tests conducted, and the processes for identifying and managing defects. The overarching goals of system testing for "Skill Path" were to identify and rectify bugs, validate system performance under typical conditions, ensure data integrity, confirm the accuracy and relevance of AI-driven features, and ultimately, to guarantee a high-quality, reliable, and user-friendly application ready for deployment and use.

6.2. Scope of Testing

The scope of system testing for "Skill Path" encompassed all major modules and functionalities developed, focusing on end-to-end user scenarios and critical system interactions. Key areas covered during testing include:

- **Core Platform Functionalities:**
    - User registration (including email verification flow), login, logout, and password management (reset, change).
    - User profile creation and updates (bio, avatar).
    - Course catalog display, course detail page navigation, and category filtering.

    - Lesson access, content display (text, video, files), and lesson completion marking.

- **AI-Integrated Features:**
    - AI Chatbot: Responsiveness, accuracy of general answers, effectiveness of tool use (e.g., fetching courses, user progress), conversation history management, and streaming of responses.
    - Automated Assessment: Correctness of MCQ auto-grading. Quality, relevance, and parsing of AI-generated feedback and preliminary scores for SAQs.
    - Course Recommendations: Relevance and accuracy of AI-driven recommendations on the student dashboard and the functionality of fallback mechanisms.
- **Interactive and Community Features:**
    - Discussion Forum: Category browsing, thread creation, posting replies, and ordering of threads/posts.
    - Gamification: Correct awarding of points for defined actions, badge issuance based on criteria, level progression, and leaderboard accuracy.
    - Course Review and Rating System: Submission of reviews/ratings, calculation and display of average ratings, and prevention of duplicate reviews.

    - Certification: Automatic issuance of certificates upon course completion and their display on the student dashboard and certificate view page.

- **User Dashboards:**
    - Student Dashboard: Accuracy of displayed progress, gamification data, certificates, and recommendations.
    - Instructor Dashboard: Accuracy of course statistics, student progress for their courses, and functionality of links to management actions.
    - Admin Dashboard: Accuracy of platform-wide statistics and links.
- **Non-Functional Aspects (Basic Checks):**
    - Responsiveness of the UI across different device sizes (desktop, tablet, mobile).
    - Basic usability and navigation clarity.
    - Functionality of the light/dark theme toggle.
    - Error handling and display of user-friendly messages.
    - Basic security checks (e.g., access control for different user roles, protection against common inputs like simple XSS in forms).

6.3. Features to be Tested and Not to be Tested

6.3.1. Features to be Tested

A comprehensive set of features and functionalities were subjected to testing. This includes, but is not limited to, the following:

• **User Authentication & Profile Management**

- **Purpose:** Ensure users can securely create, access, and manage their accounts.
- **Expected Behavior:** Users should be able to register with email verification, log in using valid credentials (email/password), log out, change their password, reset it via email, and update their profile (bio, avatar). Proper redirection should occur after each action.
- **Success Criteria:** All authentication and profile actions complete successfully without errors. User data is correctly saved and reflected in the UI. Verification and reset emails are sent and contain functional links.

• **Course Discovery & Navigation**

- **Purpose:** Ensure users can effectively browse and access course content.
- **Expected Behavior:** The course list displays accurate information (title, instructor, thumbnail, average rating). Users can filter by category and search for relevant results. Course detail pages show all modules, lessons (with correct lock/unlock status), reviews, and support enrollment/access.
- **Success Criteria:** Courses display correctly. Filtering and search return expected results. Navigation across course pages is seamless.

- **Lesson Interaction & Completion**

    - **Purpose:** Ensure lesson content can be consumed and completion is tracked.
    - **Expected Behavior:** Lesson components (text, videos, files) render correctly. The "Mark as Complete" button updates progress and awards points. Lock/unlock logic enforces sequential access. Navigation between lessons is functional and respects lock status.
    - **Success Criteria:** Content loads properly. Progress tracking works accurately. Locking and unlocking operate as intended.

- **Quiz Taking & Assessment**

    - **Purpose:** Validate quiz features, including AI-assisted SAQ grading.
    - **Expected Behavior:** Students can take quizzes. MCQs are auto-graded. SAQs receive AI-generated feedback and a preliminary score. The results page shows total scores, user responses, correct MCQ answers, and SAQ feedback.
    - **Success Criteria:** Quizzes are submitted successfully. MCQ grading is correct. SAQ feedback is relevant. Results are clearly displayed.

- **AI Chatbot Functionality**

    - **Purpose:** Ensure the chatbot assists users effectively and uses tools properly.
    - **Expected Behavior:** The chatbot should handle general queries and, where applicable, invoke tools (e.g., `get_courses_by_category`,

`get_user_overall_progress`) and incorporate tool output into a streamed response. Session-based conversation history is maintained.

- **Success Criteria:** Chatbot responses are accurate and timely. Tool calls are contextually appropriate. Streaming is uninterrupted.

- **Discussion Forum Interaction**

  - **Purpose:** Validate discussion and engagement features.
  - **Expected Behavior:** Users can browse categories, view threads and posts, create threads, and reply. New replies should update thread activity.
  - **Success Criteria:** All forum interactions succeed. Content is correctly displayed. Points are awarded for participation.

- **Gamification System**

  - **Purpose:** Verify motivational features like points, badges, and leaderboards.
  - **Expected Behavior:** Users receive points for specific actions. Badges are awarded upon meeting criteria. Levels update with accumulated points. Leaderboard reflects accurate rankings.
  - **Success Criteria:** Points, badges, levels, and leaderboard display correct values.

- **Course Reviews and Ratings**

  - **Purpose:** Confirm the review system works as intended.

- **Expected Behavior:** Eligible users can submit course reviews and ratings. Average ratings update accordingly. Duplicate reviews by the same user are disallowed.
- **Success Criteria:** Reviews are saved properly. Ratings are accurately calculated.

• **Certificate Issuance**

- **Purpose:** Ensure automatic generation and display of completion certificates.
- **Expected Behavior:** A certificate is generated upon completing all course lessons and is viewable on the student dashboard and certificate page.
- **Success Criteria:** Certificates generated correctly with the proper user and course details.

• **Dashboards (Student, Instructor, Admin)**

- **Purpose:** Ensure dashboards show relevant and up-to-date information.
- **Expected Behavior:** Dashboards reflect correct statistics and data. Action links are functional and visible based on user roles.
- **Success Criteria:** Information is accurate. Navigation and access are appropriate per role.

• **Theme and Responsiveness**

- **Purpose:** Verify consistent theming and cross-device compatibility.

- **Expected Behavior:** The UI adheres to the custom theme (including light/dark modes). All pages are responsive across desktop, tablet, and mobile.
- **Success Criteria:** Layouts display correctly on all devices. Theme toggle functions properly. No visual or functional issues across screen sizes.

6.3.2. Features Not to be Tested (or Out of Scope for This Testing Phase)

The following features were excluded from formal testing during this phase due to typical academic project constraints:

• **Scalability Under Extreme Load (Stress/Load Testing)**
Formal load testing with thousands of simultaneous users was not conducted, though scalability considerations were incorporated into the design.

• **Advanced Security Penetration Testing**
While Django's security features and standard practices were followed, deep penetration testing by security professionals was not performed.

• **Long-term Data Integrity Checks**
While data persistence was verified during normal use, year-long or long-term data integrity scenarios were not simulated.

• **AI Model Hallucination/Bias (In-depth)**
While prompt engineering minimized risk and outputs were informally evaluated, a formal audit of AI model bias or hallucination risks was out of scope.

• **Offline Functionality**

As the platform is web-based, offline use was not tested or supported during this development phase.

6.4. Pass/Fail Criteria

For each test case derived from the "Features to be Tested," the following general pass/fail criteria were applied:

- **Pass:**
    - The feature functions exactly as per its specified requirements and expected behavior.
    - All inputs are handled correctly (valid, invalid, boundary conditions where applicable).
    - The user interface elements related to the feature are displayed correctly and are responsive.
    - Data is correctly created, read, updated, or deleted in the database as expected.
    - No critical or major errors (e.g., application crashes, data corruption, security vulnerabilities) are encountered.
    - AI-generated content (feedback, recommendations, chatbot responses) is relevant, coherent, and does not contain harmful or nonsensical output for typical inputs.
    - Navigation to and from the feature is seamless.

- **Fail:**
    - The feature does not perform its intended function.
    - The system crashes or throws unhandled exceptions.
    - Incorrect data is displayed or saved.
    - Security vulnerabilities are exposed.
    - The UI is broken, unresponsive, or significantly deviates from the design.
    - AI outputs are consistently irrelevant, nonsensical, or harmful for typical inputs.
    - Minor errors that significantly impact usability or core functionality.

6.5. Key Test Cases

Key test cases were designed to cover critical user flows and functionalities. Examples include:

- **TC_AUTH_001:** Successful New User Registration and Email Verification
    - Objective: Verify a new user can register, receive a verification email, and activate their account.
    - Preconditions: User does not exist. Email system (console or SMTP) is functional.
    - Steps:
        1. Navigate to the registration page.

2. Enter valid, unique email, password, and confirm password.
3. Submit the form.
4. Check for success messages and instructions to verify email.
5. Check console/email inbox for verification email.
6. Click the verification link from the email.
7. Check for a success message indicating account activation.
8. Attempt to log in with the new credentials.

- Expected Results: User account created as inactive initially. Verification email sent with a valid link. Clicking the link activates the account. Users can successfully log in. Profile created.
- Actual Results: Pass
- Status: Pass/Fail

- **TC_COURSE_005**: Student Completes a Lesson and Progress Updates
  - Objective: Verify lesson completion updates progress and awards points/badges.
  - Preconditions: Student logged in, enrolled in a course with uncompleted lessons.
  - Steps:
    1. Navigate to an uncompleted lesson page.
    2. View content.

3. Click "Mark as Complete."
4. Check for success messages and point award messages.
5. Navigate to the Student Dashboard.
6. Navigate to Course Detail page

- Expected Results: Lesson marked as complete. Points awarded and reflected on the dashboard. Relevant badges awarded (e.g., "First Steps"). Course/module progress percentages update accurately on dashboard and course detail.
- Actual Results:Fail
- Status: Pass/Fail

- **TC_AI_CHAT_002:** AI Chatbot Tool Use - Get Course Recommendations
  - Objective: Verify the AI chatbot can use the get_courses_by_category tool.
  - Preconditions: User logged in. The OpenAI API key is valid. get_courses_by_category tool is defined.
  - Steps:
    1. Open chatbot widget.
    2. Type query: "Show me courses on Python."
    3. Observe chatbot response.
  - Expected Results: Chatbot understands the intent. Backend logs show a tool call to get_courses_by_category with "Python" as an

argument. Chatbot provides a natural language response listing relevant Python courses (if any exist) based on the tool's output, including titles and reasons.

    - Actual Results: Pass
    - Status: Pass/Fail

6.6. Test Execution

Test execution was performed iteratively throughout the development lifecycle and more formally at the end of major development phases.

- **Development Testing:** Developers performed informal unit and integration tests as features were built.
- **Manual System Testing:** The primary method for system testing involved manual execution of defined test cases (as exemplified in 6.5) covering all user flows and functionalities. This was conducted in a local development environment simulating production settings where possible (e.g., DEBUG=False for some tests, using SQLite but with production-like data structures).
- **Environment:**
    - Local Development Environment: Windows, Python 3, Django 5, SQLite, VS Code, Modern Web Browsers (Chrome, Firefox).
    - Staging/Production Environment (Render): Linux-based containers, Python 3,Gunicorn, Whitenoise. Testing was repeated on Render after deployment.

- **Tools & Methods:**
    - Browser Developer Tools: Used extensively for inspecting HTML, CSS, JavaScript execution, network requests (AJAX, API calls), and responsive design emulation.
    - Django Debug Toolbar (Local Dev): Used to monitor database queries, template rendering times, and other performance metrics during local testing.
    - Django Admin Interface: Used to set up test data (users, courses, etc.) and verify data integrity after test case execution.
    - Manual Input & Observation: Test cases were executed by manually interacting with the web interface and observing if the system behaved as expected.
    - OpenAI API Monitoring (If available): Checking OpenAI platform usage dashboard for API call success/failure during AI feature testing.
- Process: Each test case was executed, and the actual results were compared against the expected results. Pass/Fail status was recorded. Defects were logged if discrepancies were found.

6.7. Defect Management

A simple defect management process was followed:

1. Identification: Defects (bugs, errors, deviations from requirements) were identified during unit, integration, system testing.
2. Logging: Defects were logged with details such as:

    - Defect ID (simple numbering).
    - Description of the issue.
    - Steps to reproduce.
    - Expected result vs. Actual result.
    - Severity/Priority (e.g., Critical, Major, Minor).
    - Assigned to (developer).
3. Prioritization & Assignment: Defects were prioritized based on their impact on core functionality.
4. Resolution: Developers fixed the assigned defects.
5. Verification (Retesting): The original test case that revealed the defect was re-executed to confirm the fix. Related areas were also smoke-tested.
6. Closure: Once a defect was verified as fixed, it was marked as closed.

This systematic approach helped ensure that issues were addressed promptly and effectively.

6.8. Resources (for Testing)

Testing resources for "Skill Path" included:

- **Personnel (The Project Team):**
    - All team members participated in manual system testing and exploratory testing of features they developed or were familiar with.

    - The team members may have taken lead roles in coordinating testing efforts and defect tracking.

- **Tools:**
    - Web Browsers (Chrome, Firefox, Edge) and their built-in developer tools.
    - Django Development Server.
    - Django Admin Interface (for data setup/verification).
    - Postman for API endpoint testing .
- **Infrastructure:**
    - Local development machines for initial testing.
    - Render PaaS (free/starter tiers) for staging/production environment testing.

6.9. Schedule (for Testing)

System testing was not a single, isolated phase but an ongoing activity integrated with the Agile development sprints.

- **Iterative Testing:** Basic testing of features was conducted at the end of each development iteration/sprint corresponding to the implementation phases (Core, AI Integration, Community, Refinement).
- **Dedicated System Testing Cycles:** More comprehensive system testing cycles were planned:
    - After major feature sets were completed.

    - Before the initial deployment to Render (pre-production testing).
    - Post-deployment smoke testing on the Render environment.
- **User Acceptance Testing (UAT - Informal):** While formal UAT with external users was limited by the academic project scope, team members acted as proxy users to test usability and gather feedback throughout.

6.10. Estimated Risk and Contingency Plan (for Testing)

Potential risks during the testing phase and their mitigation strategies included:

- **Risk:** Discovery of critical bugs late in the development cycle impacting deadlines.
    - Likelihood: Medium.
    - Impact: High.
    - Contingency/Mitigation: Iterative testing throughout development to catch bugs early. Prioritize fixing critical/major bugs immediately. Allocate buffer time in the schedule for bug fixing.
- **Risk:** Inconsistent AI API responses (OpenAI) affecting testability of AI features.
    - Likelihood: Medium (especially with LLMs).
    - Impact: Medium.
    - Contingency/Mitigation: Robust error handling in API call utilities. Focus testing on typical inputs. Use mock responses for isolated unit

tests of surrounding logic. Accept a degree of variability in AI textual output.

- **Risk:** Limited availability of diverse test data (e.g., for recommendations, a wide range of user interactions).
    - Likelihood: High for an academic project.
    - Impact: Medium (might not fully test the robustness of AI personalization).
    - Contingency/Mitigation: Manually create diverse test user profiles and interaction data. Focus testing on the logic with available data. Clearly document this limitation.
- **Risk:** Time constraints limiting the depth of testing for all edge cases.
    - Likelihood: High.
    - Impact: Medium.
    - Contingency/Mitigation: Prioritize test cases based on critical functionalities and high-risk areas. Utilize exploratory testing to uncover unexpected issues.
- **Risk:** Discrepancies between local development environment and Render production environment.
    - Likelihood: Medium.
    - Impact: Medium.

- Contingency/Mitigation: Thorough post-deployment testing on Render. Ensure environment variables are correctly configured in Render to match production needs (e.g., DEBUG=False, production database URL).

6.11. Conclusion

The system testing phase for the "Skill Path" AI-Powered E-Learning Platform was integral to ensuring the delivery of a functional, reliable, and user-friendly application. Through a combination of iterative manual testing, focused testing of AI integrations, and basic non-functional checks, critical functionalities were validated against the specified requirements. Defects identified during the process were logged and addressed to improve software quality. While exhaustive testing across all possible scenarios and advanced performance/security metrics was constrained by the project's scope, the testing conducted provides confidence in the platform's core stability and its ability to deliver the intended AI-enhanced learning experience. Further testing, particularly User Acceptance Testing with a broader audience and more rigorous performance and security assessments, would be key next steps for a production-grade, commercially deployed system.

Chapter 7: Conclusion and Recommendation

7.1. Conclusion

The "Skill Path: AI-Powered E-Learning Platform" project successfully achieved its primary objective of developing a modern, interactive, and intelligent online learning environment. By integrating the robust Django framework for backend development, a custom-themed Bootstrap 5 for a responsive user interface, and leveraging the advanced capabilities of OpenAI APIs, the platform demonstrates a significant step towards addressing the limitations of traditional e-learning systems. Throughout its development, "Skill Path" has materialized key functionalities envisioned at the outset, including personalized course recommendations (initially via TF-IDF with a transition to OpenAI LLM-based suggestions), AI-assisted feedback and preliminary scoring for Short Answer Questions, a dynamic AI chatbot with tool-use capabilities, comprehensive student and administrative dashboards, an engaging gamification system, and community-building features like discussion forums and course reviews.

The project successfully navigated the complexities of integrating external AI services, managing user data, and creating a cohesive user experience across diverse features. The iterative, Agile-based development methodology allowed for flexibility and continuous refinement, ensuring that core features were implemented and user feedback (simulated or actual from team testing) could be incorporated. Key accomplishments include the implementation of a custom user model with email-based authentication and verification, a structured course delivery system with lesson locking, a functional streaming AI chatbot, and a multi-faceted gamification system that demonstrably enhances learner motivation.

The deployment to Render.com showcases the platform's viability in a production-like environment.

Challenges encountered, such as initial API quota limitations for embeddings (leading to the adoption of TF-IDF as an interim solution for some recommendation aspects and then focusing on LLM-based recommendations), the intricacies of prompt engineering for consistent AI behavior, and ensuring a seamless user experience across both light and dark themes, provided valuable learning experiences. These challenges were largely overcome through iterative problem-solving, careful API management, and focused UI/UX refinement.

The "Skill Path" platform, in its current state, serves as a strong proof-of-concept and a functional prototype that effectively showcases how AI can be harnessed to create more personalized, supportive, and engaging educational experiences. It contributes to the field by providing a practical example of integrating contemporary AI tools into a full-stack web application designed for learning.

7.2. Recommendation (for Future Work)

Based on the insights gained during the development of "Skill Path" and an understanding of the evolving e-learning landscape, the following recommendations are proposed for future work, improvements, and potential expansions of the project:

1. **Enhanced AI-Powered Course Recommendations:**
    - OpenAI Embeddings: Fully implement and A/B test course recommendations using OpenAI Embeddings (text-embedding-ada-002 or newer models) for deeper semantic

understanding of course content and user preferences, potentially outperforming the current LLM-prompting or TF-IDF methods in certain scenarios.

- Collaborative Filtering: Once sufficient user interaction data (especially course ratings and detailed completion patterns) is collected, integrate collaborative filtering techniques (e.g., using libraries like Surprise or scikit-learn) to recommend courses based on "users who liked X also liked Y" patterns.
- Hybrid Recommendation Engine: Combine content-based (Embeddings/TF-IDF) and collaborative filtering approaches for a more robust and accurate recommendation system.

2. **Advanced AI in Assessments:**
    - Full SAQ Auto-Grading with Confidence Scores: Further refine prompts and potentially fine-tune smaller models (if feasible and cost-effective) to achieve higher accuracy in SAQ scoring, with the AI providing a confidence level for its grade to better guide instructor review.
    - AI-Generated Quiz Questions: Explore using LLMs to generate diverse quiz questions (MCQs, SAQs) based on lesson content, assisting instructors in assessment creation.
    - Plagiarism Detection: Integrate tools or develop mechanisms for basic plagiarism detection in SAQs or forum posts.
3. **Sophisticated Instructor Tools & Analytics:**

- Custom Frontend for Course Management: Develop dedicated instructor interfaces (beyond Django Admin) for creating, editing, and organizing course content, modules, lessons, and quizzes with a rich text editor and more intuitive UI.
- Detailed Student Analytics: Provide instructors with more granular analytics for their students within a course, such as time spent on lessons, specific quiz answer analysis, and identification of struggling students based on AI-inferred patterns.
- SAQ Review Interface: A dedicated interface for instructors to efficiently review SAQs flagged by the AI (or all SAQs), view AI's suggested feedback/score, and input their final grade.

4. **Enhanced Personalization & Adaptivity:**
    - Adaptive Learning Paths: Develop logic where the sequence or content of lessons can dynamically adapt based on a student's performance in preceding lessons or quizzes.
    - Personalized Learning Styles: Allow users to specify learning preferences (e.g., visual, auditory, kinesthetic) and explore how AI can tailor content presentation or suggest supplementary materials accordingly.
5. **Improved Chatbot Capabilities:**
    - Deeper Course Content Knowledge (RAG): Implement Retrieval Augmented Generation (RAG) by creating a vector database of course

content, allowing the chatbot to answer highly specific questions about lesson materials accurately.

- Proactive Interventions: Enable the chatbot to proactively offer help or suggestions if it detects a user might be struggling (e.g., spending too long on a lesson, repeatedly failing a quiz).

6. **Community and Collaboration Enhancements:**
    - Peer Review System: Allow students to review each other's assignments or SAQs (with guidance).
    - Study Groups: Facilitate the formation of study groups within courses.
    - Direct Messaging/Notifications for Forum Mentions: Implement @mentions and direct notifications for forum interactions.
7. **Technical & Operational Improvements:**
    - Comprehensive Testing: Implement a more extensive suite of automated unit and integration tests. Conduct formal User Acceptance Testing (UAT) with a larger, diverse group of target users.
    - Performance Optimization: Conduct thorough performance profiling and optimize database queries, AI API call strategies, and caching mechanisms for scalability.
    - Advanced Security Audit: Perform a professional security audit.
    - Full Internationalization (i18n) & Localization (l10n): Add support for multiple languages.

- Mobile Applications: Develop native mobile applications (iOS/Android) for enhanced accessibility and user experience on mobile devices.
- CI/CD Pipeline: Fully automate testing and deployment using a CI/CD pipeline (e.g., GitHub Actions with Render).

These recommendations aim to build upon the solid foundation of "Skill Path," further leveraging AI and enhancing user experience to create an even more powerful and impactful e-learning platform. The successful implementation of the current version demonstrates the potential, and these future steps provide a clear roadmap for continued innovation and development.