**Project Proposal**

**Project Name:**

"Grade A" is an e-learning platform that delivers engaging, interactive, and accessible educational content.

**1. Project Idea**

The e-learning management system is a web-based platform designed to facilitate online education. It allows instructors to create and manage courses, students to enroll and access materials, and administrators to oversee the system's operations. The platform aims to provide a seamless and engaging learning experience while addressing common challenges in online education, such as accessibility, scalability, and interactivity.

**2. Problem Statement**

In the era of digital transformation, education systems face several challenges:

1. Limited accessibility to quality education for remote learners.
2. Difficulty in tracking students’ progress and engagement in a virtual setting.
3. Inefficient tools for instructors to create, manage, and assess courses.
4. Lack of a centralized system that integrates features such as content delivery, assignments, and performance tracking.

The proposed e-learning management system addresses these challenges by providing a user-friendly platform that fosters collaboration between instructors, students, and administrators.

**3. User Personas**

1. **Student Persona:**
   * **Description**: college students who seek to enroll in online courses to enhance their learning experience.
   * **Goals**:
     + Enroll in courses based on approve of admission
     + Access materials easily and track their progress.(grades)
     + Receive timely feedback on assignments and tests.
   * **Pain Points**:
     + Limited interactivity with course options, .
     + Lack of clear progress tracking.
2. **Instructor Persona:**
   * **Description**: Educators or professionals who want to create and manage online courses.
   * **Goals**:
     + Upload and organize course materials efficiently.
     + Engage with students through assignments and discussion forums.
     + Track student performance and provide feedback.
   * **Pain Points**:
     + Managing large classes virtually.
     + Difficulty assessing students’ engagement and progress.
3. **Admin Persona:**
   * **Description**: System administrators responsible for managing users, courses, and platform configurations.
   * **Goals**:
     + Ensure the platform runs smoothly for all users.
     + Monitor user activity and troubleshoot issues.
     + Maintain data security and integrity.
   * **Pain Points**:
     + Handling large volumes of data.
     + Ensuring platform reliability under high user traffic.

**4. High-Level Functional Requirements**

1. **User Authentication and Role Management**
   * **Users can register and log in as one of three roles: students, instructors, or administrators.**
   * **Secure login system with session management to prevent unauthorized access.**
   * **Role-based access control:**
     + **Students can access course content and track progress.**
     + **Instructors can manage their courses and monitor student performance.**
     + **Admins can oversee platform operations and manage user roles.**
2. **Course Management**
   * **Instructors:**
     + **Can create, edit, and delete courses, including uploading associated materials.**
   * **Students:**
     + **Can browse and enroll in courses based on their academic level or eligibility.**
     + **Can view course materials after enrollment.**
   * **Admins:**
     + **Can manage courses, including approval or deactivation of submitted courses.**
3. **Content Delivery**
   * **Instructors can upload various types of course materials, such as:**
     + **PDFs, Word documents, and videos.**
     + **Quizzes and interactive assignments.**
   * **Students can:**
     + **Download course materials.**
     + **Submit assignments through the platform.**
4. **Progress Tracking**
   * **Students:**
     + **Can view detailed progress reports, including:**
       - **Completed course modules.**
       - **Grades on assignments, quizzes, and exams.**
   * **Instructors:**
     + **Can monitor student performance with visual analytics, such as:**
       - **Grade distribution curves.**
       - **Trends indicating improvement or decline.**
5. **Communication Tools (Optional)**
   * **Basic message boards for discussions between students and instructors.**
   * **Notification system to alert users about:**
     + **Course updates, deadlines, or grades.**

**5. Using Supabase for the Prototype**

**Reason for Using Supabase:**

**For the prototype, we will use Supabase as a remote database and backend. Supabase provides a PostgreSQL-based backend-as-a-service that offers a range of features, including authentication, real-time data synchronization, and API generation.**

**Advantages of Supabase Over Local Databases:**

1. **No Infrastructure Management:**
   * **Supabase eliminates the need for manual server setup and maintenance, allowing the team to focus on the application logic.**
2. **Scalability:**
   * **Supabase automatically scales as the application grows, ensuring consistent performance for concurrent users.**
3. **Real-Time Capabilities:**
   * **Supabase supports real-time data updates, making it easier to reflect changes such as grades or course progress instantly across the platform.**
4. **Cost-Efficiency:**
   * **Using Supabase is cost-effective during the early stages of development, as it avoids expenses associated with hosting and managing a local database.**
5. **Secure Authentication:**
   * **Built-in authentication features simplify secure user management, which is crucial for the prototype.**

**Transition to Django Backend in Later Stages:**

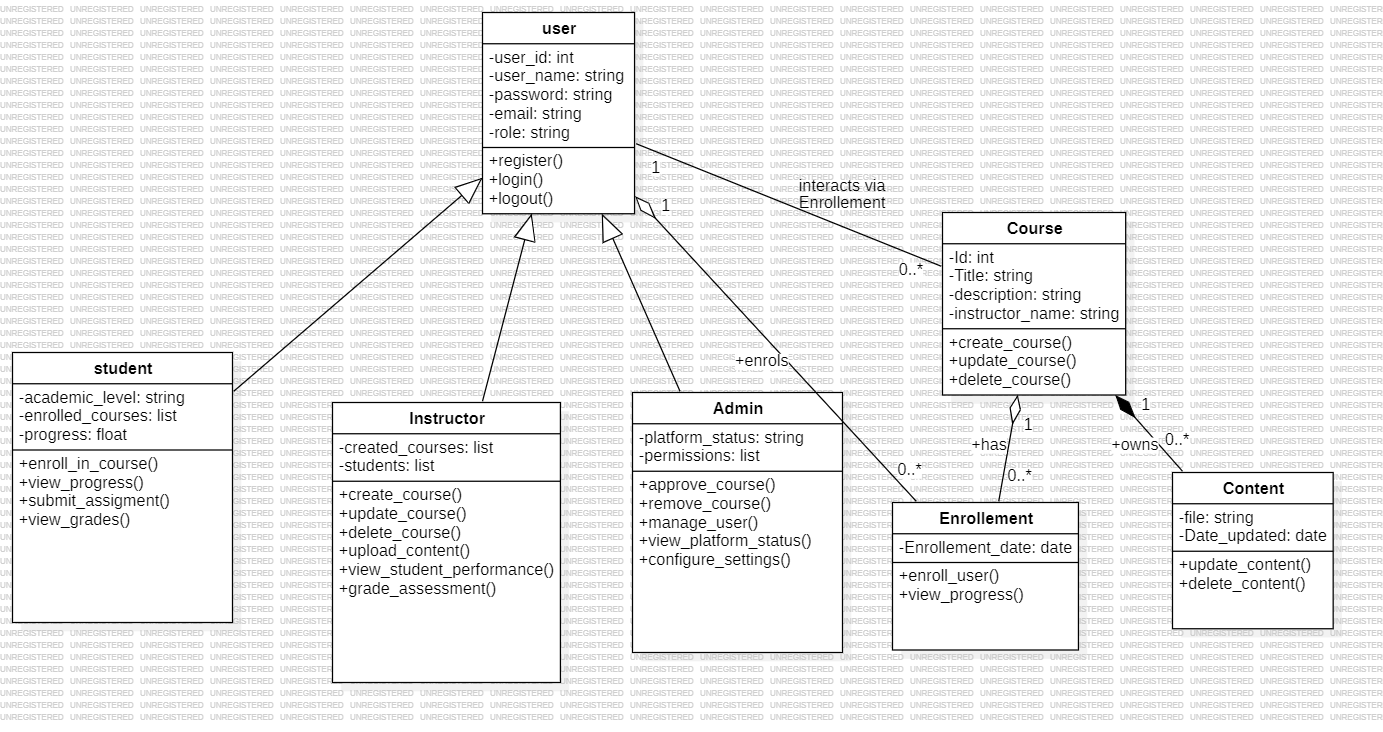
**In the final stages of the project, we will transition to Django for the backend to ensure:**

1. **Custom Business Logic: Django allows greater control over complex workflows and backend processes.**
2. **Advanced Security Features: Django's security mechanisms, such as CSRF protection and ORM-based SQL injection prevention, enhance platform reliability.**
3. **Maintainability: A Django-based backend ensures that the project adheres to best coding practices and supports long-term development goals.**

* **5. High-Level Non-Functional Requirements**

1. **Performance**
   * **The platform must support up to 100 concurrent users without noticeable degradation.**
   * **Response time for most user actions should not exceed 2 seconds under normal load.**
2. **Security**
   * **Utilize Django's built-in authentication framework for secure user management.**
   * **Encrypt sensitive data, such as passwords, using industry-standard algorithms.**
   * **Enforce HTTPS for all communications to ensure data security in transit.**
3. **Usability**
   * **Responsive design to ensure usability across various devices, including desktops, tablets, and smartphones.**
   * **An intuitive and accessible user interface tailored for:**
     + **Students for ease of navigation and course tracking.**
     + **Instructors for efficient course management.**
     + **Admins for seamless platform oversight.**
4. **Maintainability**
   * **Adopt Django's recommended coding standards and practices for long-term maintainability.**
   * **Modular architecture to facilitate:**
     + **Rapid implementation of new features.**
     + **Quick resolution of bugs or security vulnerabilities.**

**Class diagram:**

****

**Use case diagram**

**A diagram of a diagram

Description automatically generated**

**1. Architecture Diagram**

The architecture diagram provides a high-level view of how the system's components interact. The e-learning platform will follow a **three-tier architecture**:

* **Frontend**: User interface, accessible via web browsers (HTML, CSS, JavaScript).
* **Backend**: Django framework, handles business logic and communicates with the database.
* **Database**: SQL database (e.g., PostgreSQL) for managing data such as users, courses, enrollments, and content.

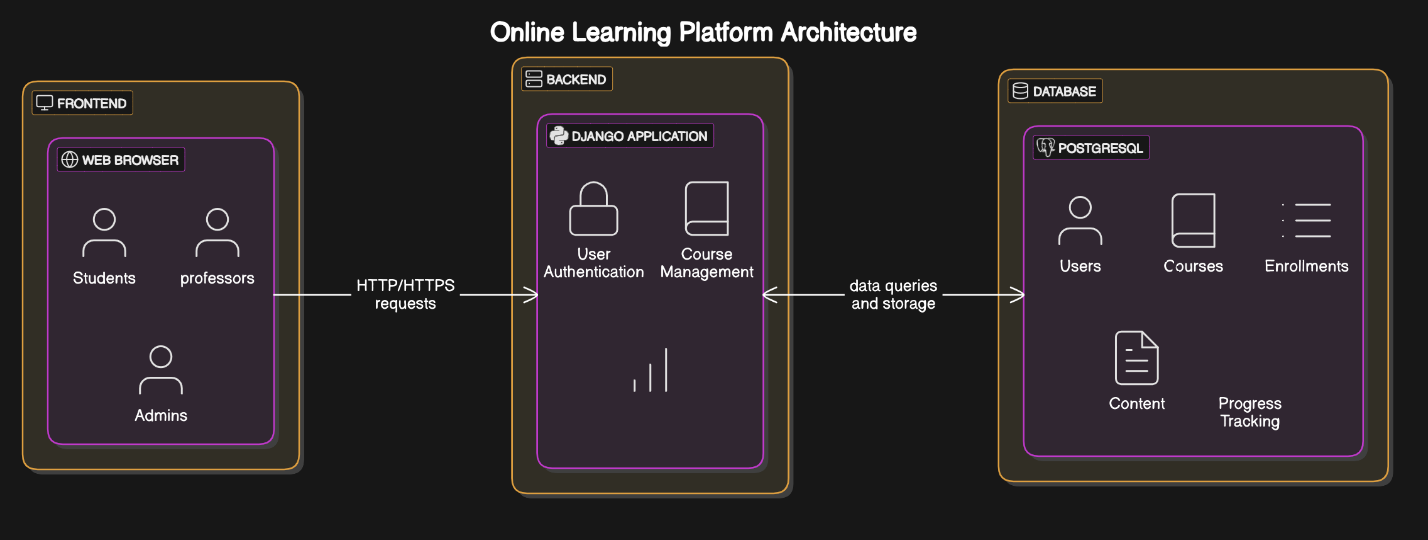
**High-Level Components:**

* **Web Browser (Client)**: Users (students, instructors, admins) interact with the application via a responsive web interface.
* **Django Backend (Application Layer)**:
  + User Authentication
  + Course Management
  + Progress Tracking
* **Database Layer**:
  + Stores user data, course details, and progress reports.

**Architecture Diagram**

The architecture diagram would include the flow of requests from the frontend to the backend and database. Here's how you can represent it:

* User requests (e.g., login, view course) are sent via HTTP/HTTPS.
* Django processes these requests, interacts with the SQL database, and sends the response back.

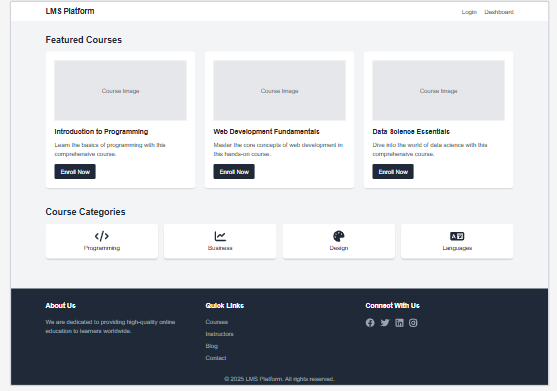


**3. Wireframes**

Wireframes are simple, visual blueprints of how the user interface will look. They help in understanding user navigation and interactions.

**Wireframe Components:**

1. **Home Page**:
   * Navbar with login/logout, dashboard link.
   * Course categories or featured courses.
2. **Login Page**:
   * Input fields for email and password.
   * Login button and "Forgot password?" link.
3. **Instructor Dashboard**:
   * Create a course.
   * View and manage existing courses.
   * Upload content.
4. **Student Dashboard**:
   * View enrolled courses.
   * Access course materials.
   * Track progress.
5. **Admin Dashboard**:
   * User management (approve instructors, manage students).
   * View platform reports.



A computer screen shot of a computer

Description automatically generated