**Assignment 1: Django Web Application**

**Objective:** Develop a basic Django web application for a simple task management system.

**Requirements:**

1. Create a Django project named "TaskManager" with an app named "Tasks."
2. Implement a model for tasks with fields like title, description, due date, and status.
3. Create CRUD (Create, Read, Update, Delete) views for managing tasks.
4. Include a responsive user interface using Django templates and Bootstrap for styling.
5. Implement a search functionality to filter tasks by title or status.
6. Use Django forms for task creation and updating.

**Submission Instructions:**

* Provide a GitHub link to the repository containing the Django project.
* Include a README file with instructions on setting up and running the application.

**Assignment 2: FastAPI API Development**

**Objective:** Build a RESTful API using FastAPI for a simple note-taking application.

**Requirements:**

1. Create a FastAPI project named "NoteAPI" with CRUD operations for notes.
2. Implement a Pydantic model for notes with fields like title, content, and creation timestamp.
3. Use an in-memory database (e.g., a Python list) to store and retrieve notes.
4. Include API routes for creating, reading, updating, and deleting notes.
5. Implement validation to ensure that required fields are present for note creation and updating.
6. Add documentation for the API using FastAPI's automatic OpenAPI and Swagger support.

**Submission Instructions:**

* Provide a GitHub link to the repository containing the FastAPI project.
* Include a README file with instructions on setting up and running the API.
* Include sample API requests and responses in the README.

**Assignment 3: Django and FastAPI Integration**

**Objective:** Integrate Django and FastAPI to create a unified application with both web and API components.

**Requirements:**

1. Extend the TaskManager Django project from Assignment 1 to include a FastAPI app for task-related API endpoints.
2. Use FastAPI to create API routes for listing tasks, retrieving a single task, and updating task status.
3. Ensure that the Django web application and FastAPI components share the same task model.
4. Implement authentication for the FastAPI API using OAuth2 (optional: use Django's authentication for web views).
5. Include a feature in the Django web interface to display task details using data fetched from the FastAPI API.

**Submission Instructions:**

* Provide a GitHub link to the repository containing the integrated Django and FastAPI project.
* Include a README file with instructions on setting up and running the integrated application.
* Demonstrate the integration by showing how changes in the web interface affect the API and vice versa.

**Optional Assignment 4: Full-Stack Task Management Application**

**Objective:** Extend the integrated Django and FastAPI project from Assignment 3 by adding a frontend using JavaScript and Node.js.

**Requirements:**

1. Integrate a frontend application using a JavaScript framework/library (e.g., React, Vue, or Angular) with the Django and FastAPI backend.
2. Create views for listing tasks, displaying task details, and updating task status in the frontend application.
3. Use AJAX or Fetch API to interact with the FastAPI API endpoints from the frontend.
4. Implement a user authentication system for the frontend application, ensuring that only authenticated users can view and manage tasks.
5. Include a responsive and user-friendly design for the frontend using a CSS framework or custom styling.
6. Ensure that changes made in the frontend application are reflected in the Django web interface and FastAPI API, maintaining synchronization.

**Submission Instructions:**

* Provide a GitHub link to the repository containing the full-stack project.
* Include a README file with instructions on setting up and running the entire application (frontend, Django, and FastAPI components).
* Include screenshots or a demo video showcasing the integrated application, with a focus on the frontend functionality.