Project Name: ToDOList

# Overview:

The task management web application aims to streamline project management processes within a small project management company. The current reliance on disparate tools and spreadsheets has led to inefficiencies and confusion. To address these challenges, the project focuses on creating a centralized system for managing tasks and projects.

## Key Features:

### Task Management:

* Users can create, update, and delete tasks on a centralized to-do list.
* Each task can have associated details such as due date, priority level, status, and assigned team members.
* CRUD operations are implemented for managing tasks efficiently.

### Technology Stack:

* Backend Framework: Django or FastAPI
* Database: SQLite (default database for Django)
* Frontend: HTML/CSS, JavaScript (for client-side interactions)

### Project Objectives:

* Streamlined Task Management: Provide a user-friendly interface for managing tasks efficiently, reducing reliance on disparate tools.
* Centralized System: Create a centralized system for storing and managing task-related information, enhancing organization and accessibility.

### Expected Benefits:

* Efficiency: Centralized task management reduces manual efforts and improves task tracking and prioritization.
* Clarity: Clear task details and real-time communication enhance clarity and understanding among team members.
* Productivity: Streamlined processes and improved collaboration lead to increased productivity and project efficiency.

### Project Goals:

* Develop a robust task management system with CRUD operations for tasks.
* Integrate a real-time chatroom for seamless communication and collaboration.
* Ensure user authentication, permission handling, and data security.
* Provide comprehensive documentation for easy setup and usage.

### Future Enhancements:

* Integration with project management methodologies (e.g., Agile, Scrum) for enhanced project planning and execution.
* Advanced reporting and analytics features for tracking project progress and performance.
* Integration with external tools and services for extended functionality and flexibility.
* By delivering a comprehensive task management web application with real-time communication capabilities, the project aims to address the challenges faced by small project management companies and improve overall efficiency and collaboration in project management processes.

# Technologies Used:

* Django
* Python
* Django REST Framework
* SQLite (default database)

# Project Structure:

## Project Directory Structure:

* Copy code

project\_root/

├── chatroom/

│ ├── migrations/

│ ├── templates/

│ ├── admin.py

│ ├── apps.py

│ ├── forms.py

│ ├── models.py

│ ├── serializers.py

│ ├── urls.py

│ └── views.py

├── myapp/

│ ├── migrations/

│ ├── templates/

│ ├── admin.py

│ ├── apps.py

│ ├── forms.py

│ ├── models.py

│ ├── serializers.py

│ ├── urls.py

│ └── views.py

├── todo/

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

├── db.sqlite3

└── manage.py

# Components:

### Models:

### Task Model:

#### Fields:

task (CharField)

member (ForeignKey to User)

due\_date (DateField)

### Task Model:

#### Fields:

user (Forignkey to member)

member (TextFeild)

timestamp (DateField)

#### Serializers:

1. adduserserializer:
   * Serializes User model for creating new users.
   * Creates hashed passwords and generates authentication tokens for users.
2. addtaskserializer:
   * Serializes Task model for creating new tasks.

#### Views:

##### adduser:

* + POST: Creates a new user.

##### login:

* + POST: Authenticates user and generates authentication token.
  + GET: Retrieves tasks for authenticated user.

##### reset\_password:

* + POST: Resets user password.

##### addtask:

* + POST: Creates a new task.

##### gettask:

* + GET: Retrieves details of a specific task.

##### tasklist:

* + GET: Retrieves a list of all tasks.

##### deletetask:

* + POST: Deletes a specific task.

##### create\_task:

* + POST: Creates a new task (alternative view).

##### logout:

* + POST: Logs out the user.

##### Message( Chatroom ):

* + GET: Retrieves the message.
  + POST: Create new message.

# URLs:

1. /adduser/: Endpoint for adding a new user.
2. /login/: Endpoint for user login.
3. /reset/: Endpoint for resetting user password.
4. /createtask/: Endpoint for creating a new task.
5. /logout/: Endpoint for user logout.
6. /message/: Endpoint for user to retrive or create message

#### Usage

* + Register a new user by visiting /adduser/.
  + Login with your credentials at /login/.
  + Create tasks using the /createtask/ endpoint.
  + View and manage tasks via the provided endpoints.
  + Logout using the /logout/ endpoint.
  + /message/ endpoint is used to create or retrieve all the message.

# Installation

#### Clone the repository:

* + git clone <repository-url>

#### Navigate to the project directory:

* + cd project-directory

#### Install dependencies:

* + pip install -r requirements.txt

#### Run migrations:

* + python manage.py migrate

#### Start the development server:

* + python manage.py runserver

Access the project at http://localhost:8000/