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 , DRF ( Django Rest Framework )
* Database: SQLite (default database for Django)

### 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)

### Chatmessage 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:

TEAM – MEMBERS:

1. admin/: This URL pattern is used for accessing the Django admin interface. It is provided by default by Django for managing site administration.
2. Teamlead/: This URL pattern includes URLs from the myapp application. It likely provides endpoints related to task management, such as creating, updating, or deleting tasks.
3. login/: This URL pattern is mapped to the login view in the views.py file. It is typically used for user authentication, allowing users to log in to the application.
4. reset/: This URL pattern is mapped to the reset\_password view in the views.py file. It might be used for resetting user passwords or initiating password recovery processes.
5. logout/: This URL pattern is mapped to the logout view in the views.py file. It is used to log out users from the application.
6. message/: This URL pattern includes URLs from the chatroom application. It is likely related to real-time messaging functionality, allowing uers to communicate with each other in a chatroom.

TEAM - LEAD:

1. Add User:
   * URL: /adduser/
   * View Function: views.adduser
   * Description: This URL pattern is used to add a new user to the system.
2. Add Task:
   * URL: /addtask/
   * View Function: views.addtask
   * Description: This URL pattern is used to add a new task to the system.
3. Task List:
   * URL: /tasklist/
   * View Function: views.tasklist
   * Description: This URL pattern is used to display the list of tasks.
4. Delete Task:
   * URL: /delete task/<int:id>/
   * View Function: views.deletetask
   * Description: This URL pattern is used to delete a specific task identified by its ID.

I gave the permissions only to the team leader to add the user, add task, delete task, task list etc

# Installation

#### Clone the repository:

* + git clone <repository-url>

#### Navigate to the project directory:

* + cd project-directory

#### Install dependencies:

* + pip install -r requirements.txt

#### Run migration:

* + python manage.py migrate

#### Start the development serveR:

* + python manage.py runserver

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

Super user for admin login username: teamleader, password: pwd

Access the project using api tool (postman)