Project Description: ToDo Web Application

1. Project Overview

The **ToDo Web Application** is a productivity tool designed to help users create, organize, and manage daily tasks efficiently. The application will provide a clean, intuitive, and responsive user interface where users can add tasks, categorize them, set deadlines, track progress, and mark them as complete. The backend will handle data persistence securely, ensuring that tasks are always saved and retrievable across sessions.

The aim is to create a **minimal yet powerful productivity application** with features like task prioritization, search, filters, reminders, and user authentication for personalized task management.

---

2. Objectives

- Provide users with a simple way to **add, edit, delete, and manage tasks**.
- Allow users to **set due dates, priorities, and categories** for each task.
- Support **marking tasks as completed/in-progress/pending**.
- Offer **search and filtering options** to quickly find tasks.
- Provide a **responsive design** that works across desktop, tablet, and mobile devices.
- Enable **user authentication** so each user can manage their own ToDo list securely.
- Store all task data in a **database** for persistence.

---

3. Key Features

A. Core Features

1. **Task Management**
    - Add a new task with title, description, due date, priority, and category.
    - Edit existing tasks.
    - Delete tasks.
    - Mark tasks as completed/pending/in-progress.
2. **Task Organization**
    - Categorize tasks (e.g., Work, Personal, Shopping, etc.).
    - Add tags/labels for better grouping.
    - Assign priority levels (High, Medium, Low).
    - Sort tasks by due date, category, or priority.
3. **Search & Filter**
    - Search tasks by title or description.
    - Filter by categories (e.g., only show Work tasks).
    - Filter by task status (Completed, Pending, Overdue).
4. **User Authentication**
    - Secure user registration and login (using email + password).
    - Password hashing for security.
    - Each user should only access their own tasks.
5. **UI/UX**
    - Minimal and modern design with clean typography.
    - Responsive layout (works across devices).
    - Intuitive interactions (drag & drop tasks between categories).

- Dark mode toggle.

B. Advanced Features (Optional but recommended for “stunning” app)

1. **Reminders & Notifications**
    - Notify users about upcoming or overdue tasks.
    - Option for daily/weekly reminder summaries.
2. **Recurring Tasks**
    - Allow users to set tasks that repeat (daily, weekly, monthly).
3. **Task Progress Tracking**
    - Show percentage of tasks completed for a day/week.
    - Dashboard with progress bar/charts.
4. **Collaboration (Future Scope)**
    - Share task lists with other users (team collaboration).

---

4. Technology Stack

Frontend (User Interface)

- **Framework**: React.js (for a modern, responsive, and interactive UI).
- **Styling**: Tailwind CSS / Material UI for sleek and consistent design.
- **State Management**: Redux / Context API.
- **Routing**: React Router.

Backend (Server & APIs)

- **Framework**: Python Flask / FastAPI (lightweight and scalable).
- **Authentication**: JWT-based authentication.
- **Database ORM**: SQLAlchemy (for structured DB handling).

Database (Persistence)

- **SQLite** (for local dev) → **PostgreSQL/MySQL** (for production).
- Schema:
    - **Users Table**: `user_id`, `name`, `email`, `password_hash`.
    - **Tasks Table**: `task_id`, `user_id`, `title`, `description`, `due_date`, `priority`, `status`, `category`, `created_at`, `updated_at`.

Hosting/Deployment

- **Frontend**: Vercel / Netlify.
- **Backend**: Render / AWS / Heroku.
- **Database**: Cloud PostgreSQL (e.g., Supabase, RDS).

---

5. System Flow

1. **User Authentication Flow**
    - New users sign up.
    - Returning users log in.
    - Token-based authentication to protect APIs.
2. **Task Management Flow**

- User creates task → API stores in DB.
- User edits/deletes task → API updates DB.
- Fetch tasks dynamically on login.

3. **Search/Filter Flow**
    - User applies filters → Backend fetches filtered data.
    - Search queries by keyword return matching tasks.
4. **Progress Tracking Flow**
    - Count completed tasks vs. total tasks.
    - Display progress bar or chart in UI.

---

6. Project Milestones

1. **Phase 1 (MVP)**
    - User Registration & Login.
    - Add, Edit, Delete, View Tasks.
    - Mark tasks as complete.
    - Basic UI.
2. **Phase 2**
    - Task categorization & prioritization.
    - Search & filter functionality.
    - Responsive UI with dark mode.
3. **Phase 3 (Enhancements)**

    - Notifications & reminders.
    - Recurring tasks.
    - Dashboard with progress tracking.
4. **Phase 4 (Future)**
    - Team collaboration.
    - File attachments in tasks.

---

7. Deliverables

- **Frontend React App** with task management UI.
- **Backend Flask/FastAPI APIs** with authentication & CRUD operations.
- **Database Schema** for users and tasks.
- **Deployment Guide** with setup instructions.
- **Documentation** (README + API docs).

---

8. Success Criteria

- Users can **securely register and manage tasks**.
- The app runs seamlessly across devices.
- Tasks are **stored persistently** and not lost after logout.
- UI is **clean, modern, and user-friendly**.
- Advanced features (search, filters, reminders) work as intended.