Approach Document

1. Backend Technology Choice

I chose **Python with Flask** for the backend implementation.

Flask is a lightweight and beginner-friendly web framework that allows quick development of REST APIs with minimal configuration, and I already have used Flask in my college Project so I am familier with it

The backend uses an in-memory list to store tasks, as required. This keeps the implementation straightforward and aligned with the assignment instructions.

2. API Structure and Design

The API was designed following basic REST principles.

I have created a global list and task id, the task id variable will help us to give unique id's to the tasks for make each task unique. The fetch is used to connect to frontend to the backend.

Endpoints Implemented

1. **POST /tasks**
    - Adds a new task.
    - Accepts title and description in JSON format.
    - Automatically assigns a unique ID.
    - Default status is set to completed: false.
2. **GET /tasks**
    - Fetches all tasks.
    - Supports filtering by status (pending/completed).
    - Returns data in JSON format.
3. **PUT /tasks/:id/complete**
    - Marks a specific task as completed.
    - Updates the task status to true.

Design Considerations

- Each task contains:
    - id
    - title
    - description
    - completed
- IDs are auto-incremented to ensure uniqueness.
- Error handling is included for invalid task IDs.
- CORS is enabled to allow communication between frontend and backend running on different ports.

The API structure is simple, readable, and modular, making it easy to extend in the future.

3. Forntend State Management

I have used react for the frontend. It is easy use and has its state management hooks.

State Management Approach

The application uses:

- **useState** to manage:
  - Task list
  - Input fields
  - Confirmation dialog visibility
- **useEffect** to:
  - Fetch tasks when the component loads.

State is updated after every API call to ensure the UI reflects the latest backend data.

---

4. Future Improvements

If given more time, the following enhancements could be implemented:

- Integration with a database (MongoDB or PostgreSQL)
- Divide the code in components for better understanding and reusability.
- Add edit and delete functionality
- Add user authentication
- Improve UI responsiveness for mobile devices
- Add better validation and error handling
- Implement loading states and notifications