Todoist Clone Project Documentation

Damianos Zoumpos

August 9, 2025

Contents

1 Introduction

This project is a personal implementation of a **Todoist-like task management application**. It was designed to practice and demonstrate full-stack development skills (**with more focus on the backend,APIs, security and communication between front-end and back-end**) using Python (Flask), JavaScript (React), Tailwind CSS, SQLAlchemy, and other modern web development tools. The system supports task creation, editing, deletion (completion of the task), authentication, and API-based communication between the frontend and backend.

2 Tools Used

- **Python (Flask)** – Backend framework to handle API routes, business logic, and database interactions.
- **JavaScript (React)** – Frontend library for building dynamic user interfaces.
- **Tailwind CSS** – Utility-first CSS framework for styling.
- **SQLAlchemy** – Python ORM for database modeling and queries.
- **HTML5** – Structure and layout for front-end templates.
- **Flask-Login** – Session management and authentication.
- **JWT (JSON Web Tokens)** – Token-based authentication for secure API calls and communication between the frontend and backend.
- **Postman** – API testing tool.
- **Git & GitHub** – Version control and project hosting.

3 Frontend

The front-end was developed using **React** to provide a responsive and interactive user experience.

- **React Components** – Modular components for tasks, forms, and navigation.
- **Axios** – For making API requests to the backend.
- **Tailwind CSS** – For styling and maintaining a consistent UI design.
- **Dynamic Rendering** – Data fetched from APIs is displayed in real-time without reloading the page.

4 Backend

The backend was implemented with **Flask** and follows a RESTful architecture.

- **Blueprints** – Separation of concerns between authentication, main app logic, and API endpoints (breakdown to **API** for users and **API** for tasks).
- **Flask-SQLAlchemy** – Object-relational mapping for database management.
- **Flask-Migrate** – Database migrations and schema management.
- **Flask-CORS** – Cross-Origin Resource Sharing to allow communication with the React front-end and block any other source that tries to communicates with the backend

5 APIs

The system exposes a set of RESTful APIs for task and user management.

- **Authentication Endpoints** – Login, registration, and JWT token validation.
- **Task Management Endpoints** – Create, read, update, and delete tasks.
- **Profile Management Endpoints** – Update user profile details.
- All API responses are in **JSON** format.

6 Database

The application uses **SQLAlchemy ORM** to interact with a relational database (SQLite/MySQL/Post
depending on deployment).

- **User Table** – Stores user credentials, profile information, and authentication data.
- **Task Table** – Stores task details, deadlines, and user associations.
- **Relationships** – One-to-many relationship between users and tasks.

7 Authentication and Security

The application implements multiple layers of security:

- **Flask-Login** – Handles user session management.
- **JWT Authentication** – Protects API routes and ensures only authenticated requests are processed.
- **Password Hashing** – User passwords are securely stored using hashing algorithms (using the **bcrypt** library).
- **CORS Configuration** – Restricts access to trusted front-end origins.

8 User Flow and Access

When a user first visits the application, they are presented with the **main marketing site**. This landing area contains:

- Information about the application and its features.
- Pricing details for the different usage plans.
- Access to the **Login** and **Registration** pages.

To access the main task management application, a user must first **create an account**.

- During registration, both the email address and the username must be unique.
- Attempting to register with an email or username that is already taken will result in an error message.

Once registered, the user can log in using their credentials and will be redirected to the main application interface, where they can create, edit, and manage their tasks.

9 Future Improvements

Several enhancements are planned for future versions:

- **OAuth2 Integration** – Allow users to sign up and log in with Google, Facebook, or Apple.
- **Offline Mode** – Enable task creation and updates without an internet connection.
- **Real-time Notifications** – WebSocket-based notifications for task reminders.
- **Advanced Filtering and Search** – Improve task management with better filters.
- **Mobile App Version** – Build a mobile version using React Native or Flutter.

10 Conclusion

This project demonstrates the integration of modern frontend and backend technologies to build a functional and scalable task management system. It showcases skills in REST API development, authentication, database management, and responsive UI design.

With further improvements like OAuth2 integration and mobile support, the system can evolve into a production-ready task management platform.