Project Documentation

Project Title: To-Do List Manager Webapp

Developer: Hamza Badshah

Technology Stack: Python, Gradio

Hosting Platform: Hugging Face Spaces

TABLE OF CONTENTS

1. Introduction

The **To-Do List Manager Webapp** is a productivity-focused application built with Python and Gradio. It allows users to manage their daily tasks through a simple and responsive web interface, accessible both locally and on Hugging Face Spaces. Designed for ease of use and rapid deployment, the app is ideal for users seeking minimal yet functional task management tools.

2. Objectives

- Build a fully functioning web-based To-Do List manager using Python.
- Provide a user-friendly interface for task tracking.
- Deploy the application on Hugging Face with zero front-end code.
- Learn the use of Gradio SDK for rapid UI generation.

3. Problem Statement

People often struggle with organizing their daily tasks in a clear, interactive, and accessible way. Many existing apps are either too complex or require sign-ups. This project aims to offer a no-login, instant-use task manager that runs directly from the browser.

4. Scope of the Project

- User-friendly task creation and deletion
- Web-based interface using Python
- Hosted online for universal access
- No persistent database (tasks reset on refresh)

5. Tools & Technologies Used

Tool/Tech	Purpose
Python 3.10+	Core language
Gradio 5.38.1	Front-end interface toolkit
Hugging Face	Hosting and deployment of the webapp
GitHub	Version control & project repository

6. System Requirements

Minimum System Requirements (for local run):

- Python 3.10 or higher
- pip package manager
- Internet connection (for Hugging Face deployment)

Required Python Packages:

gradio==5.38.1

7. System Design & Architecture

The application is designed with a **modular functional structure**. The UI and logic are tightly integrated using Gradio components. Key elements:

- Textbox for input
- Buttons for actions (Add, Delete, Edit)
- Checkboxes for marking completed
- Progress Bar to show overall progress

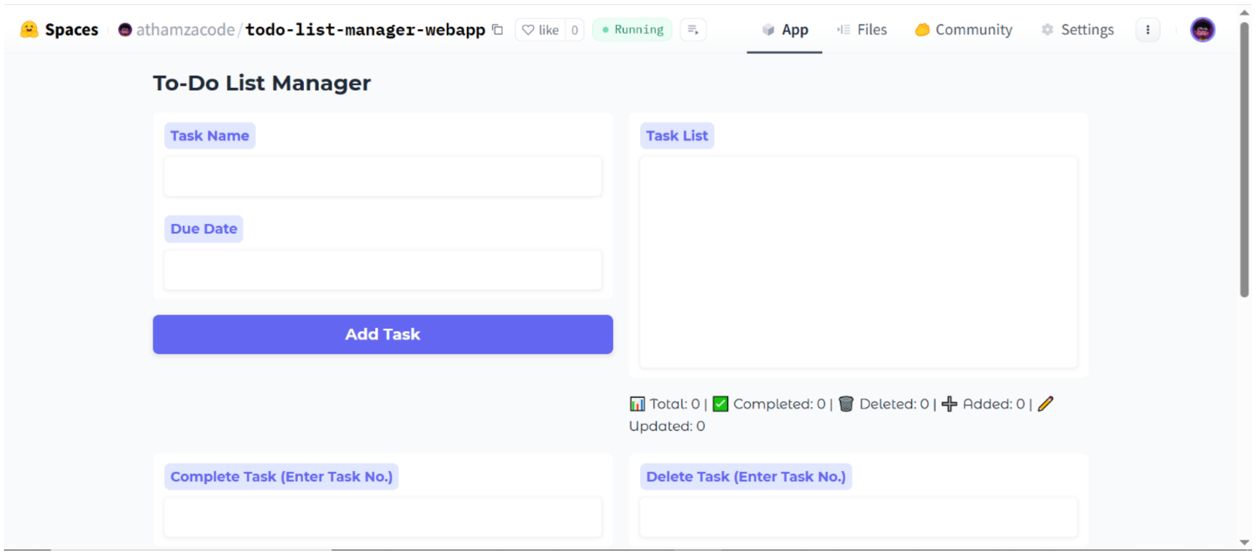
8. Implementation Details

- **Frontend** is built using Gradio components.
- **Backend logic** is written in Python functions, managing task lists.
- Tasks are stored in a temporary list (session_state) and updated dynamically.
- User actions (like checking a task or editing it) trigger corresponding function calls.

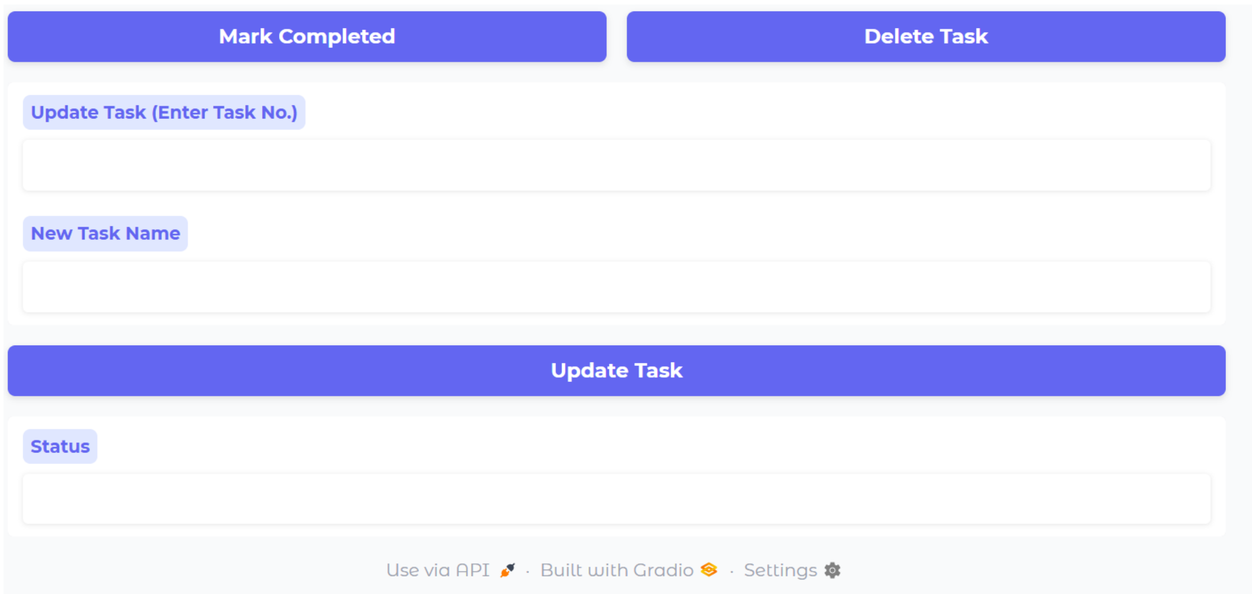
9. Features

- Add a new task
- Delete specific tasks
- Edit task names
- Mark tasks as complete/incomplete
- Show progress bar of task completion
- Clear all tasks

Application Interface



📊 Total: 0 | ✅ Completed: 0 | 🗑 Deleted: 0 | ➕ Added: 0 | ✏️ Updated: 0


Mark Completed
Delete Task
Update Task (Enter Task No.)
New Task Name
Update Task
Status
Use via API · Built with Gradio · Settings

11. Testing & Evaluation

- Manual functional testing of each feature
- Tested on both local machine and Hugging Face
- No automated tests (due to simplicity)
- Cross-browser compatibility tested

12. Installation & Deployment

A. Run Locally

```
pip install gradio==5.38.1

python app.py
```

B. Deploy on Hugging Face

1. Upload your project files (app.py, requirements.txt)
2. Set up Space metadata:

```
sdk: gradio
sdk_version: 5.38.1
app_file: app.py
pinned: false
license: mit
```

3. Click **Create Space**

13. Project Structure

```
├── app.py                # Main app logic
├── requirements.txt      # Python packages
├── README.md             # GitHub info
├── Project_Snippets      # Screenshots
        ├── todo_list_manager_webapp (1).png
        └── todo_list_manager_webapp (2).png
```

14. Limitations & Future Work

Limitations

- No user authentication
- No persistent storage (tasks are session-based)
- No task deadlines/reminders

Future Improvements

- Add login & user profiles
- Integrate database for task storage
- Implement dark mode & themes
- Add notification/reminder features

15. Conclusion

This project is a successful demonstration of how Python alone can be used to build interactive web applications with Gradio. It emphasizes simplicity, clarity, and rapid development while still maintaining functionality. It is well-suited for beginners and developers exploring quick app prototyping in Python.

16. License

This project is licensed under the MIT License.