To-Do App

A simple and intuitive task management application that allows users to create, update, and track their tasks efficiently.

Features

- User Authentication: Sign up and log in with a username and password.
- Create Tasks: Add new tasks with a title, description, due date, priority, and category.
- View Tasks: See all tasks at a glance, categorized by status (All, Pending, Completed).
- Edit Tasks: Modify task details and due dates.
- Delete Tasks: Remove tasks once they're completed or no longer relevant.
- Task Filtering: Filter tasks by category, status, priority, or due date.
- Mark as Complete: Mark tasks as done
- Responsive Design: The app is designed for ease of use on both mobile and desktop devices.

Tech Stack

Backend: Python, MongoDB (for user sessions and task storage)

Frontend: Kivy (for the app's user interface)

Security: bcrypt for password hashing

Installation

Prerequisites

Make sure you have the following installed on your system:

- Python 3.x
- MongoDB Atlas (or local MongoDB setup)
- Kivy (for GUI)

Steps

1- Clone the repository

```
git clone https://github.com/elsie200/My-To-Do-List-Manager.git
cd todo-app
```

2- Install required Python packages

3- Set up your MongoDB database

- Create a MongoDB Atlas cluster or set up a local MongoDB server. I used MongoDB Atlas cluster and I actually wrote an article about it that you can find on my blog (https://techmemoirsbyelsie.hashnode.dev/)
- Update the mongo.py file with your MongoDB connection string.

4- Run the app

```
python main.py
```

Usage/Examples

1- **Registration & Login:** Sign up for a new account or log in with an existing account.

2- **Creating Tasks:** Once logged in, navigate to the '*Add Task*' page to add new tasks.

3- **Managing Tasks:** View your tasks on the dashboard. Click on tasks to view details, edit, or delete them.

4- **Filtering:** Use the filter option to customize your task list view.

Project Structure

```
My-To-Do-List-Manager/
│
├── Commands/
│   ├── __pycache__/
│   ├── actions_pages.py        # Adding and Editing classes
│   ├── commands.py             # Contains core app commands (adding, deleting, 
modifying tasks, etc) but are destined to a command line use
│   ├── gcommands.py            # Global commands or utility functions used across the 
graphical app
│   ├── globals.py              # Globals: connected user and categories
│   ├── graphic.py              # Graphics classes (Welcome, Login & Registration 
Screen) and ToDoApp class
│   ├── manage_tasks.py         # Classes related to task displaying (Filter)
│   ├── mongo.py                # MongoDB connection and queries
│   ├── main.py                 # Entry point for running the app (initialization and 
screens setup)
│   └── read_commands.py        # Reads commands from user input or files
│
├── Users/
│
├── env/                        # Virtual environment files (optional for isolation)
├── venv/                       # Virtual environment related folder
│
├── playground-1.mongodb.js     # MongoDB playground script
├── project_UML.png             # UML diagram for the project (project structure/flow)
├── task_details.png            # Wireframe for task details screen background
└── todo_workflow.png           # Workflow diagram for task creation and management
```

Contributing

Contributions are welcome! Please follow these steps to contribute:

1- Fork the repository.

2- Create a new branch with a meaningful name.

3- Make your changes.

4- Submit a pull request with a description of your changes.

Lessons Learned

While building this project, I learnt:

- How to set up a MongoDB instance and use it in a Python project
- How to interact with a MongoDB atlas cluster using Python code
- How to setup a Kivy app and its special features

I encountered challenges like:

- Getting **the logic written before the Kivy setup** fit the new algorithm (with classes and Kivy elements). **To overcome** this, I created new functions to implement my features, based on the initial ones (still included in the project), and adding or removing some arguments to call them in my Kivy classes
- Understanding **how to position things correctly**. **To overcome** this challenge, I just tried different values to see the results which help me better understand the functionnalities.
- **Creating the TabbedPannels and displaying the right elements on each Tab**. This took me two days to fix. **To overcome** this, I used the documentation and re-assessed my logic understanding where to create the TabbedPannels and where to effectively display there contents depending on my code flow.