Project Overview – 00SR:
Develop Native Applications Without a Database

This project, part of the *"Secure Desktop, Mobile, and Web Applications"* program at Vanier College, focuses on developing a native desktop application without the use of a database, using Python and Tkinter.
Students are expected to demonstrate:

- **GUI design and implementation using Tkinter**
- **Local file-based data storage (e.g., JSON)**
- **Interaction between the user and the graphical interface**
- **Communication with system-level features or peripherals**
- **Use of multithreading for background operations**
- **Application testing and quality assurance**

Native Application Development Without a Database

Prepared by:
Kourosh Moghadessi

Date: July 27, 2025

Contents

Objective

The objective of this project is to design and implement a native desktop application, named To-Do Task Manager, that allows users to manage their personal tasks efficiently. The application is developed using Python and Tkinter, without using any database. Tasks will be saved locally in a structured JSON file. The goal is to demonstrate the ability to build a functional GUI-based program, ensure proper interaction between the user and the interface, handle file-based data management, and implement multithreading for background operations such as auto-saving.

Key Features

The "To-Do Task Manager" application includes the following key features:

- Add new tasks with a custom title
- Display a dynamic list of tasks in a graphical user interface
- Mark tasks as completed
- Delete tasks from the list
- Save all tasks locally in a structured JSON file
- Automatically load saved tasks when the application starts
- Use multithreading (threads) to auto-save data in the background
- Clean and user-friendly interface built with Tkinter
- Input validation to prevent empty or duplicate tasks
- Display user notifications or alerts (e.g., save success, input error)

Technologies Used

This project will be developed using the following technologies and tools:

- **Programming Language:** Python
- **GUI Library:** Tkinter – for building the graphical user interface
- **Data Storage:** JSON file – for storing tasks locally (no database)
- **Multithreading:** Python threading module – for background auto-save
- **File I/O:** Standard Python libraries (json, os)
- **Operating System:** Cross-platform (Windows, Linux, macOS)
- **Development Environment:** Visual Studio Code
- **Version Control:** Git (optional)

Class Design Overview

Class Name	Description
`Task`	Represents a single task with title, status, and unique ID.
`TaskManager`	Manages the list of tasks: add, delete, and update operations.
`TaskStorage`	Handles saving and loading tasks to/from a JSON file.
`MainApp`	Entry point of the application. Launches and manages the app lifecycle.
`MainWindow`	The main GUI window of the application.
`TaskInputFrame`	GUI component that allows the user to add new tasks.
`TaskListFrame`	Displays the full list of tasks in the user interface.
`TaskItemFrame`	Represents a single task in the list, with buttons to update or delete.
`SaveThread`	A background thread that auto-saves the task list periodically.
`Validator`	Validates input fields to prevent empty or duplicate entries.
`Logger`	Logs application activity and errors for debugging or audit.
`NotificationManager`	Manages user messages, alerts, and notifications.
`ThemeManager`	Handles UI themes, colors, and font settings.
`SettingsManager`	Stores and loads user preferences and app settings.
`AppController`	Coordinates the interaction between the UI and the backend logic.

Tasks to be Carried Out

The following tasks outline the development process for the "To-Do Task Manager" application. These tasks are aligned with the project requirements and demonstrate the expected steps for completing the application from design to testing and documentation.

1. Analyze the project scope and define the application objective
2. Identify required features and functionalities
3. Design the application architecture and class structure
4. Set up the development environment
5. Implement the Task class and data model
6. Build the TaskManager for managing tasks in memory
7. Create the GUI layout using Tkinter
8. Integrate input forms for adding new tasks
9. Display tasks dynamically using frames and scrollable lists
10. Implement functionality to mark tasks as completed or delete them
11. Add file-based data storage using the TaskStorage class and JSON
12. Load tasks automatically when the application starts
13. Add multithreading (SaveThread) for background auto-saving
14. Validate user inputs with the Validator class
15. Add notifications and user feedback (e.g., save confirmation)
16. Apply a visual theme using the ThemeManager
17. Create a Logger to track errors and user activity
18. Implement settings persistence (SettingsManager)
19. Perform functional and usability testing
20. Document the development and testing process
21. Record a demo video of the application
22. Prepare for the final interview and code explanation

Conclusion

The “To-Do Task Manager” project represents a simple yet practical native desktop application that allows users to manage their daily tasks efficiently without relying on a database. Developed using Python, Tkinter, and JSON files, the application offers a clean and responsive user experience.

With its object-oriented design, use of multithreading for background processes, and focus on testing and quality assurance, this project fulfills the learning objectives of the 00SR evaluation. It also demonstrates the ability to develop a standalone, stable, and scalable software solution.