Cailey Keown, Andrii Kuleshchak, Harold Ligon, Krystal Michaelis, Zachary Fike, Sean Kennedy

SDEV265

Task App

# Software Introduction:

The purpose of this project is to develop a lightweight digital task and weekly planner application that allows users to create, organize, and track tasks efficiently. The system is intended to support personal productivity by providing basic task management features with optional habit tracking, while remaining simple, stable, and easy to use.

The application must achieve reliable task creation, completion tracking, and data persistence across sessions. It is designed to run on multiple operating systems, specifically Windows and macOS, and to meet the scope and timeline of an academic software development project. This document defines *what* the system must accomplish to meet user and project requirements, without specifying implementation details.

# System Requirements:

The system must meet the following user requirements:

* Users must be able to create tasks with a title and optional notes
* Users must be able to edit existing tasks
* Users must be able to delete tasks
* Users must be able to mark tasks as complete
* Completed tasks must be removed from the active task list
* Users must be able to view tasks organized by week
* Users must be able to view task completion status
* Users must be able to manage basic habit entries that remain visible after completion
* The system must save user data when the program closes
* The system must load previously saved data when the program starts
* The user interface must be clear, readable, and easy to navigate
* The application must behave consistently across Windows and macOS

These requirements define the core functionality agreed upon by the team. Optional features such as habit tracking and motivational elements will be implemented only if time and risk constraints allow.

# System Architecture:

The system will be built using a modular software architecture to allow clear separation of responsibilities and easier collaboration among team members. The application will consist of distinct components responsible for user interaction, task and habit management, and data storage.

The user interface component will handle all user input and display of planner information. The application logic component will manage task creation, completion status, and weekly organization. A separate data-handling component will manage saving and loading planner data using file-based storage. This modular structure supports maintainability, testing, and future expansion.

# System Evolution:

The system is expected to evolve incrementally throughout the development process. Initial versions will focus on core planner functionality, including task creation, completion, and reliable data storage. As development progresses, refinements may be made to improve usability, interface clarity, and performance.

Future enhancements beyond the course scope could include reminders, notifications, cloud synchronization, or integration with calendar systems. While these features are not required for the current project, the system is designed in a way that allows potential expansion without disrupting existing functionality.

# Systems Development Lifecycle (SDLC)

The project will follow the standard stages of the Systems Development Life Cycle:

* **Planning:** Define project goals, scope, roles, risks, and schedule
* **Requirements Analysis:** Identify user and system requirements for the planner
* **Design:** Create system structure, modular layout, and interface concepts
* **Implementation:** Develop application logic, user interface, and data handling
* **Testing:** Test functionality, file saving/loading, and cross-platform behavior
* **Deployment:** Prepare the application for demonstration and submission
* **Maintenance:** Fix defects, refine features, and improve stability as needed

Each stage ensures the system meets requirements while managing risk and scope.

# Appendices:

**Hardware Requirements**

* Personal computer capable of running Windows or macOS

**Software Requirements**

* Python development environment
* Integrated Development Environment (IDE)
* GitHub for source control and documentation
* Discord for team communication

**Proposed Programming Language**

* Python