IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Development Of A Gui-Based To-Do List Application Using Python And Sqlite

Mrs. R. Divya Vani Mr. Susheel Mr. Kanturi Vikas

Ms. Jannapureddy Kalyani Ms. Guduru Shruthilaya Kiran Mr. Bitla Uday Kiran

ABSTRACT

This document outlines the creation of a Python-based graphical to-do list application aimed at simplifying daily task organization and enhancing productivity. The application uses Python's Tkinter library to provide an intuitive user interface and relies on SQLite for its database, offering reliable and scalable task storage. Users can perform core task operations – creating, editing, deleting, and setting priorities for tasks – along with advanced features such as due-date notifications and priority tagging (high/medium/low).

The interface supports dynamic task management, enabling users to mark tasks as complete and to filter or categorize tasks based on deadlines or custom labels. Under the hood, Python code with embedded SQL commands handles all Create, Read, Update, and Delete (CRUD) operations, ensuring that any change in the GUI is immediately reflected in the database. The database schema is designed to store essential task attributes such as a unique ID, description, due date, priority level, and completion status, promoting efficient data integrity and retrieval.

By leveraging Python's simplicity and SQL's robustness, this project demonstrates a practical integration of GUI development, event-driven programming, and database management. Its modular architecture is built for scalability, laying the groundwork for future enhancements like cloud data synchronization, calendar API integration, or support for multiple users. Emphasizing clean interface design and maintainable code, the application offers a solid example of a desktop tool that balances user-friendly functionality with technical robustness to meet real-world time-management needs.

INTRODUCTION

In today's fast-paced world, efficient task management is crucial for maintaining productivity and reducing stress. Traditional to-do lists, while simple, often lack the persistence and flexibility needed for modern workflows. To address this gap, this document describes the development of a desktop to-do list application built with Python and SQLite. It uses Python's Tkinter library to create an intuitive graphical interface and SQLite to manage data storage, offering a scalable, user-friendly solution for organizing daily tasks.

The application is designed to let users seamlessly create, modify, and delete tasks. Key capabilities include assigning due dates to tasks, marking tasks as completed, and designating priority levels (high, medium, or low). By integrating SQLite, the application ensures that users' task lists are saved persistently between sessions. Meanwhile, the Tkinter-based interface provides an intuitive and responsive experience, making the tool accessible to users of all backgrounds.

This project emphasizes the synergy between graphical interface design and database systems. Embedded SQL queries in the Python code perform the CRUD (Create, Read, Update, Delete) operations, dynamically synchronizing user actions with the underlying database. The database schema is optimized to store each task's attributes — such as a unique ID, description, due date, priority, and completion status — enabling efficient data retrieval and integrity.

Beyond its practical functionality, this project also serves as an educational demonstration of core programming concepts like event-driven design, modular code architecture, and database integration. The documentation outlines the system's scalability and proposes possible enhancements, such as cloud synchronization, support for multiple users, or integration with external APIs. By focusing on clean design and maintainable code, the application provides an example of how a desktop program can combine user-centric features with technical depth to meet real-world productivity needs.

LITERATURE SURVEY

- **GUI Development in Python**: The application's interface is built with Tkinter, the standard GUI toolkit for Python. Resources such as Python's official documentation and GUI programming guides provide foundational knowledge for this aspect.
- **Database Management**: The project employs SQLite for lightweight, local data storage, accessed through Python's built-in sqlite3 module for executing SQL commands.
- **Task Management Principles**: Core task management features are incorporated into the design, including due dates, priority levels, subtasks, and file attachments, aligning with research on productivity and task organization.
- **Enhanced User Interaction**: Usability is improved with the tkcalendar library for date selection via a calendar widget, and tkinter.filedialog for attaching external files to tasks.
- **Related Projects**: This work builds on ideas from open-source Python to-do list projects, focusing on simplicity and essential features that benefit personal users.

SYSTEM ARCHITECTURE

The application follows a multi-layer architecture that separates the user, interface, logic, and data storage components:

- **User**: The end user interacts with the application through the interface.
- **Presentation Layer (User Interface)**: The front-end of the application, implemented using Python's Tkinter library. This layer displays information to the user and captures user input, operating as a desktop GUI.
- **Business Logic Layer (Application Logic)**: The core processing component, written in Python. It receives input from the interface, performs operations such as adding, updating, or deleting tasks, and communicates with the database.
- **Data Layer (Database)**: A SQLite database that provides persistent storage of tasks. SQLite is a lightweight, file-based relational database well-suited for smaller or standalone applications, storing all task data on the local file system.

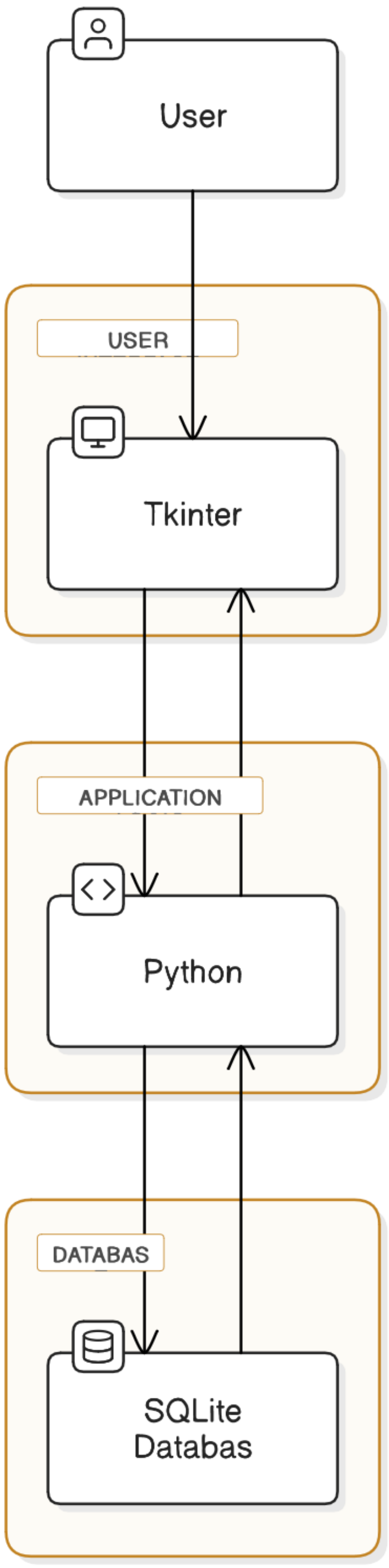

Fig: System architecture of TO-DO LIST GUI program

MODULES

- **Tkinter**: Provides the GUI framework with widgets like frames, labels, buttons, and entry fields for creating the interface.
- **tkinter.messagebox** and **tkinter.filedialog**: The messagebox module displays alert, info, or error dialogs to the user; the filedialog module allows users to select files from the system for attaching to tasks.
- **sqlite3**: A Python module for interacting with the SQLite database. It is used to execute SQL commands that create, read, update, and delete task records in the database (including marking tasks as completed).
- **datetime**: Manages date and time functions in Python, used here to record current timestamps for tasks and handle due-date calculations.
- **tkcalendar**: An additional library that provides a calendar widget, enabling users to pick due dates from a graphical calendar control.

CONCLUSION

This application serves as an intuitive tool for organizing and tracking tasks. It supports setting priority levels and due dates for tasks, creating subtasks, and attaching files or other resources to tasks. The application also keeps track of which tasks have been completed and offers ways to review them, helping users maintain an organized workflow. Together, these features provide a streamlined approach to handling daily activities.

Looking ahead, the application has significant potential for future enhancements, including:

- **Recurring Tasks**: Automating tasks to repeat on a chosen schedule (daily, weekly, etc.).
- **Task Dependencies**: Allowing tasks to be linked so that one task depends on another’s completion.

- **Collaboration Features**: Enabling shared task lists and real-time synchronization for teams or multiple users.
- **Integration with External Services**: Connecting the to-do app to calendars, email, or other project management tools for enhanced productivity.

Implementing these enhancements would transform the application into an even more versatile and powerful tool for managing tasks both individually and collaboratively.

FUTURE SCOPE

- **Task Management Improvements**:
  - Add recurring tasks that automatically schedule themselves (daily, weekly, or custom intervals).
  - Implement task dependencies to manage the order of task execution and visualize their relationships.
  - Allow users to create and manage subtasks under larger tasks.
- **Collaboration and Multi-User Features**:
  - Enable shared task lists to facilitate team collaboration.
  - Implement real-time updates so changes sync immediately across multiple users.
- **Integration with External Services**:
  - Connect the to-do list with third-party services (such as calendars, email clients, or project management tools).
  - Add cloud synchronization so users can access their tasks from multiple devices.
- **User Experience Enhancements**:
  - Improve filtering and sorting options for easier task organization.
  - Introduce a calendar view to visualize tasks by date.
  - Implement drag-and-drop functionality for rearranging tasks and allow customizable themes.
- **Accessibility and Cross-Platform Support**:
  - Develop a mobile application (Android/iOS) to complement the desktop version.
  - Create a web-based version of the task manager for access through web browsers.
- **Notifications and Alerts**:
  - Implement advanced notifications (email, SMS, or push notifications) to remind users of due tasks.
  - Allow users to customize their alert preferences and settings.
- **Enhanced Task Details and Attachments**:
  - Include a rich text editor for more detailed and formatted task descriptions.
  - Enable attaching various files and documents to tasks for comprehensive information tracking.