**Technical Design Document**

**Personal Scheduling Assistant**

**Project Overview**

The **Personal Scheduling Assistant** is a Python-based desktop application built with Tkinter for task management. It allows users to add, view, and delete tasks, track deadlines, view missed tasks, and visualize daily task loads with a stacked bar chart. The application also provides real-time notifications for tasks that are due.

**Features**

1. **Add Task**: Users can add a task with a title, type, deadline, and duration.
2. **View Upcoming Tasks**: Displays tasks that are due within the next week.
3. **View Missed Tasks**: Shows tasks with deadlines that have already passed.
4. **Delete Task**: Allows users to remove a task by title.
5. **Visualize Task Load**: Generates a stacked bar chart showing daily task loads, categorized by task type (personal or academic).
6. **Real-Time Notifications**: Notifies users of tasks that are due or overdue.
7. **Clear Fields**: Clears all input fields after an action.

**Design**

The project is organized into three main classes:

1. **Task**: Represents an individual task with attributes for title, type, deadline, and duration.
2. **Scheduler**: Manages tasks, including adding, deleting, viewing, and generating data for visualization.
3. **SchedulerApp**: Provides the GUI, connects the user interface to the scheduler functionalities, and manages notifications.

**Technical Team**

* Ezamamti Ronald Austine
* Wangobi Nicholas Kakulu
* Wangolo Bachawa

**Pseudocode**

**1. Task Class**

**Purpose:**

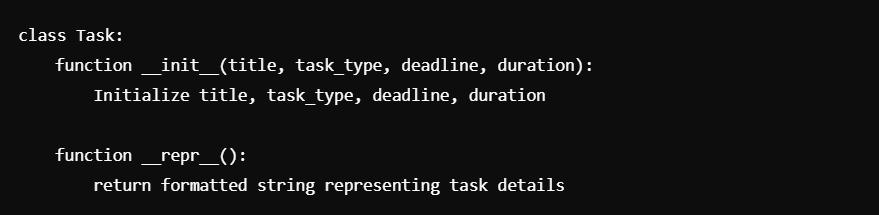
Represents each task with attributes for title, type, deadline, and duration.

**Attributes:**

* title: The title of the task (string).
* task\_type: Type of the task (e.g., 'personal' or 'academic').
* deadline: The deadline for the task (datetime).
* duration: Duration of the task in hours (float).

**Methods:**

* **\_\_repr\_\_**: Returns a string representation of the task for easy display.



**2. Scheduler Class**

**Purpose:**

Handles task management operations such as adding, deleting, retrieving upcoming and missed tasks, searching, and calculating daily task loads.

**Attributes:**

* tasks: List to hold all tasks.

**Methods:**

1. **add\_task(task)**

* Adds a new task to tasks list.
* Sorts tasks by deadline to maintain order.



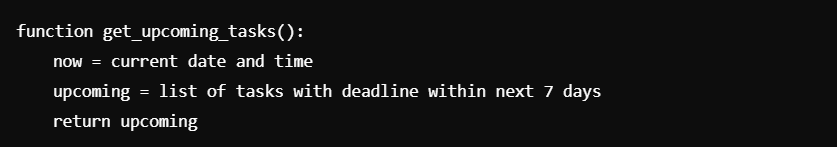
1. **delete\_task(title)**

* Searches for a task by title and removes it from tasks.
* Returns True if deleted, False if not found.



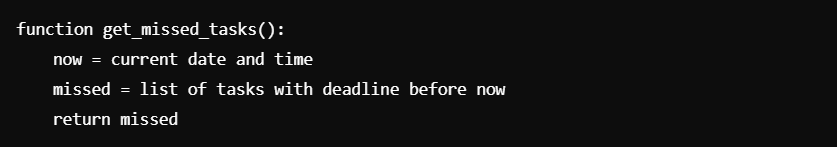
1. **get\_upcoming\_tasks()**

* Returns tasks due within the next 7 days.



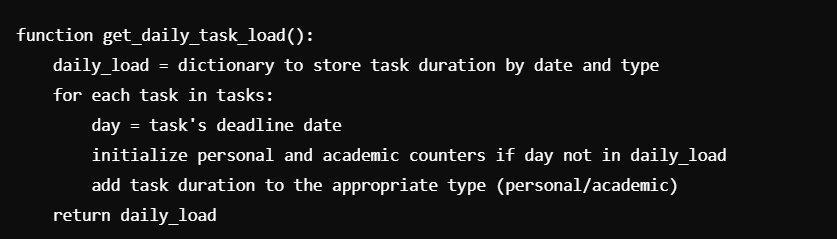
1. **get\_missed\_tasks()**

* Returns tasks that have passed their deadlines.



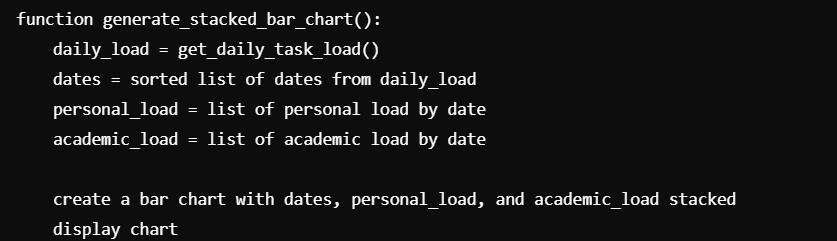
1. **get\_daily\_task\_load()**

* Calculates total duration of tasks for each day and categorizes by task type.



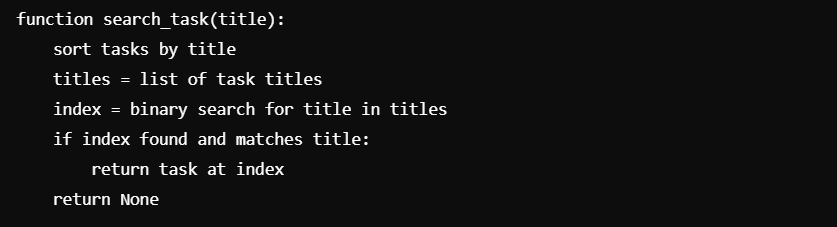
1. **generate\_stacked\_bar\_chart()**

* Generates a stacked bar chart based on daily\_task\_load, differentiating personal and academic tasks.



1. **search\_task(title)**

* Searches for a task by title using binary search.
* Returns the task if found, otherwise None.



**3. SchedulerApp Class (GUI)**

**Purpose:**

Handles user interface operations, connects GUI elements to Scheduler functions, manages input, and displays notifications.

**Methods:**

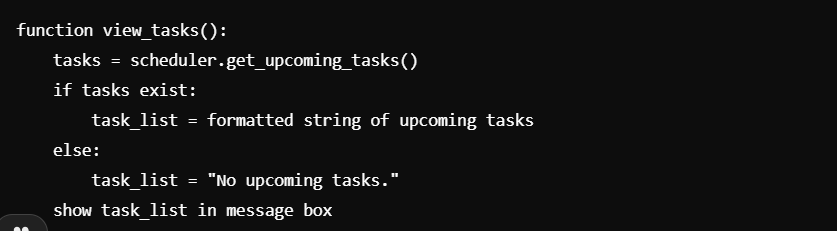
1. **add\_task()**

* Retrieves user input, creates a Task object, and adds it to the Scheduler.
* Clears input fields and shows a confirmation message.



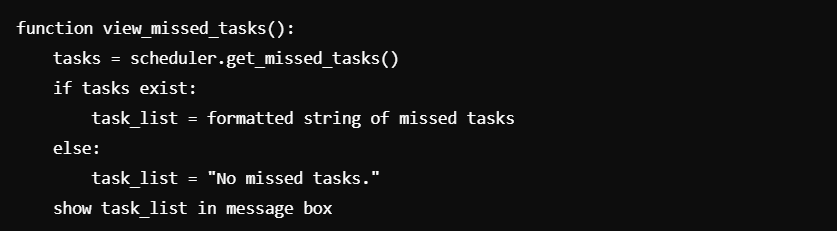
1. **view\_tasks()**

* Retrieves upcoming tasks and displays them in a message box.



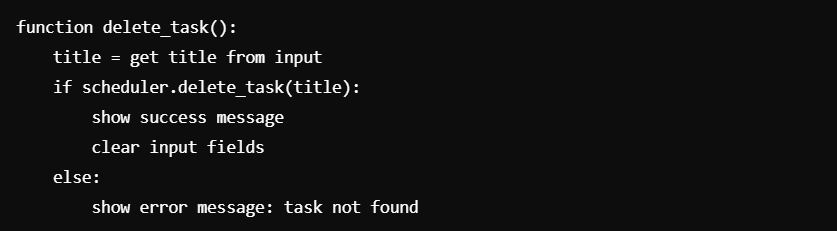
1. **view\_missed\_tasks()**

* Retrieves missed tasks and displays them in a message box.



1. **delete\_task()**

* Deletes a task by title from the Scheduler.
* Clears input fields and shows a confirmation message.



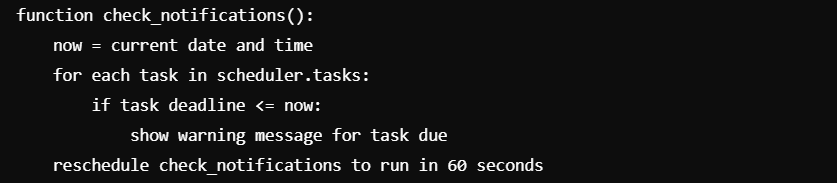
1. **show\_stacked\_bar\_chart()**

* Generates and displays the daily task load chart.



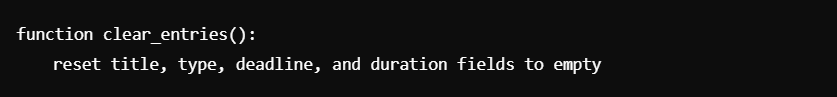
1. **check\_notifications()**

* Checks for tasks due or past due every minute and displays notifications.

 reschedule check\_notifications to run in 60 seconds

1. **clear\_entries()**

* Clears all input fields after an action.



## Challenges

* **Learning Tkinter**: We were new to Tkinter and required additional time to learn its components and event handling system for GUI development.
* **Real-time Notifications**: Implementing continuous background checks for task deadlines was a new experience and involved using Tkinter’s after() method effectively.
* **Binary Search Implementation**: Adapting binary search for a list of task objects and handling sorting dynamically based on task attributes was challenging initially.