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.

```
class Task:
    function __init__(title, task_type, deadline, duration):
        Initialize title, task_type, deadline, duration

    function __repr__():
        return formatted string representing task details
```

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.

```
function add_task(task):
    append task to tasks list
    sort tasks by deadline
```

2. **delete_task(title)**

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

```
function delete_task(title):
    task = search_task(title)
    if task found:
        remove task from tasks
        return True
    return False
```

3. **get_upcoming_tasks()**

- Returns tasks due within the next 7 days.

```
function get_upcoming_tasks():
    now = current date and time
    upcoming = list of tasks with deadline within next 7 days
    return upcoming
```

4. **get_missed_tasks()**

- Returns tasks that have passed their deadlines.

```
function get_missed_tasks():
    now = current date and time
    missed = list of tasks with deadline before now
    return missed
```

5. **get_daily_task_load()**

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

```
function get_daily_task_load():
    daily_load = dictionary to store task duration by date and type
    for each task in tasks:
        day = task's deadline date
        initialize personal and academic counters if day not in daily_load
        add task duration to the appropriate type (personal/academic)
    return daily_load
```

6. **generate_stacked_bar_chart()**

- Generates a stacked bar chart based on daily_task_load, differentiating personal and academic tasks.

```
function generate_stacked_bar_chart():
    daily_load = get_daily_task_load()
    dates = sorted list of dates from daily_load
    personal_load = list of personal load by date
    academic_load = list of academic load by date

    create a bar chart with dates, personal_load, and academic_load stacked
    display chart
```

7. **search_task(title)**

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

```
function search_task(title):
    sort tasks by title
    titles = list of task titles
    index = binary search for title in titles
    if index found and matches title:
        return task at index
    return 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.

```
function add_task():
    retrieve title, type, deadline, and duration from inputs
    try:
        task = create Task(title, type, deadline, duration)
        scheduler.add_task(task)
        show success message
        clear input fields
    except error:
        show error message
```

2. **view_tasks()**

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

```
function view_tasks():
    tasks = scheduler.get_upcoming_tasks()
    if tasks exist:
        task_list = formatted string of upcoming tasks
    else:
        task_list = "No upcoming tasks."
    show task_list in message box
```

3. **view_missed_tasks()**

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

```
function view_missed_tasks():
    tasks = scheduler.get_missed_tasks()
    if tasks exist:
        task_list = formatted string of missed tasks
    else:
        task_list = "No missed tasks."
    show task_list in message box
```

4. **delete_task()**

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

```
function delete_task():
    title = get title from input
    if scheduler.delete_task(title):
        show success message
        clear input fields
    else:
        show error message: task not found
```

5. **show_stacked_bar_chart()**

- Generates and displays the daily task load chart.

```
function show_stacked_bar_chart():
    scheduler.generate_stacked_bar_chart()
```

6. **check_notifications()**

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

```
function check_notifications():
    now = current date and time
    for each task in scheduler.tasks:
        if task deadline <= now:
            show warning message for task due
    reschedule check_notifications to run in 60 seconds
```

7. **clear_entries()**

   - Clears all input fields after an action.

```
function clear_entries():
    reset title, type, deadline, and duration fields to empty
```

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.