Sprint 3 Requirements

Export user task data into CSV or another file

- Users can export all task and schedule information into a .csv file for backup or sharing.
- Choosing Export Data should create a file that lists each task's name, duration, start / end time, and flexible / fixed status.
- The system names the file automatically, for example MySchedule_2025-11-15.csv.
- If no schedule exists, an error message appears prompting the user to create one first.
- A confirmation message is displayed after export is complete.
- File to be created: export_data.py with export_tasks() and export_schedule() functions.

Add categories for planned events

- Tasks can be grouped into categories such as Work, Study, Leisure, and Health to make schedules clearer.
- While adding or editing a task, users can select an existing category or create a new one.
- Each category is stored in the database and linked to its related tasks.
- When viewing or exporting a schedule, category labels appear beside each task name.
- File to be created: categories.py to handle category creation, editing, and deletion.

Allow user to input a vacation location and date

- The application provides a form for users to enter their travel destination and date.
- This information personalizes the schedule and supports later features like weather and suggested activities.
- Saved data should be retrievable for building future trip schedules.
- File to be created: location_input.py with save_location() and get_location() functions.

Integrate with public API (like Google Places) to search for points of interest in City

- The system connects with a public API to let users find nearby points of interest such as restaurants, gyms, or parks.
- Searches can be filtered by city, address, or keyword, returning names, addresses, and ratings.
- Any selected place can be added as a new task with duration and category fields.
- Invalid or empty API responses should be handled with clear error messages.
- File to be created: places_api.py with search_places() function.

Generate schedule by adding suggested points of interest

- Based on the chosen location and available time, the program can suggest activities to fill open time slots.
- Suggested items appear in a review list where the user can accept or remove them before saving.
- Approved items are added to the schedule and marked as "Suggested."

- Files to be created: auto_suggest.py with generate_suggestions() and insert_suggestions() functions.

Obtain prices of points of interest if applicable

- When viewing points of interest, price information should appear whenever available.
- If pricing data is missing, display “Price info unavailable.”
- Price values are stored in the database for use in budget comparison.
- File to be updated: places_api.py to include price retrieval.

Allow user to add their budget for day / trip and compare that with each event’s price

- Users can set a daily or trip budget to track their expected expenses.
- Each task or location that includes a price contributes to the total cost calculation.
- The program displays both total cost and remaining budget, issuing a warning if limits are exceeded.
- File to be created: budget_checker.py with compare_budget() function.

Retrieve weather data for the location of your event

- Weather forecasts appear alongside each scheduled event based on its date and location.
- Typical display: “Sunny, 72 °F” or “Rainy, High 68 °F.”
- Cached or placeholder text is shown if the weather API is unavailable.
- Weather information refreshes automatically whenever the schedule is updated.
- File to be created: weather_api.py with get_weather(location, date) function.

Factor in travel time between locations

- The scheduler automatically adds travel time between consecutive tasks at different locations.
- Travel duration is estimated using a mapping API and prevents task overlaps.
- If a location is missing, a default five-minute gap is applied.
- File to be created: travel_time.py with calculate_travel_time() function.

User interface for entering tasks

- Provide a simple and clear interface for adding, editing, and viewing tasks.
- The interface should include fields for name, duration, category, and location.
- Validation prevents empty names or invalid durations, and confirmation messages appear after saving.
- File to be created: task_ui.py for the task entry menu and form handling.