Homework 1

ActivityTracker App

Software Technical Document

Student: **Șerban Iustinian-Bogdan**

Subject: **Industrial Informatics**

Contents

Tables

Figures

1. Introduction

ActivityTracker is a user-friendly app that lets the user log and review their activities. It also provides clear visualizations of the activity stats and GPX data, making it easy to track the user progress over time. ActivityTracker is built for fitness enthusiasts of all levels, including runners, hikers, and bikers.

This document explains the key functionalities, design principles, and technical aspects of the app.

2. Key Features

ActivityTracker offers a range of practical features designed to help the user manage and visualize their activities:

1. **Activity Categorization**: Easily filters and displays activities by type.
2. **Detailed Data & Stats**: Displays comprehensive statistics based on activity type (see Table 1).
3. **Flexible Data Entry**: Upload activities via CSV files or enter them manually.
4. **Local Storage**: Keeps all activity data stored securely on the user device.

Activity Type / Available Stats	Workout	Run	Hike	Bike Ride
Duration (hh:mm:ss)	X	X	X	X
Calories (cal)	X	X	X	X
Average Heart Rate (bpm)	X	X	X	X
Number of Sets	X			
Elevation (m)		X	X	X
Distance (km)		X	X	X
Average Pace (min/km)		X	X	
Average Speed (km/h)		X	X	X
GPX		X	X	X

Table 1 - Available Stats by Activity Type

3. Architecture & Design

3.1. Technologies

ActivityTracker is built using C# with the .NET Framework, which provides a robust and reliable runtime environment. The application employs a Windows Forms interface as its primary UI layer, leveraging a collection of controls such as FlowLayoutPanel, TableLayoutPanel, MenuStrip, Button, TextBox, ComboBox, RadioBox, NumericUpDown and DateTimePicker to create a dynamic and responsive user experience. Some of these controls have been customized to align with the application's overall appearance and style.

Data persistence is managed via a CSV file that serves as a lightweight database, stored locally in the „AppData/Local/ActivityTracker” folder, enabling efficient read/write operations without the complexity of a traditional database system. Additionally, a file watcher is implemented to monitor the CSV file for any modifications, ensuring that external changes are detected and the application's data remains up-to-date.

3.2. Classes

3.2.1. Forms

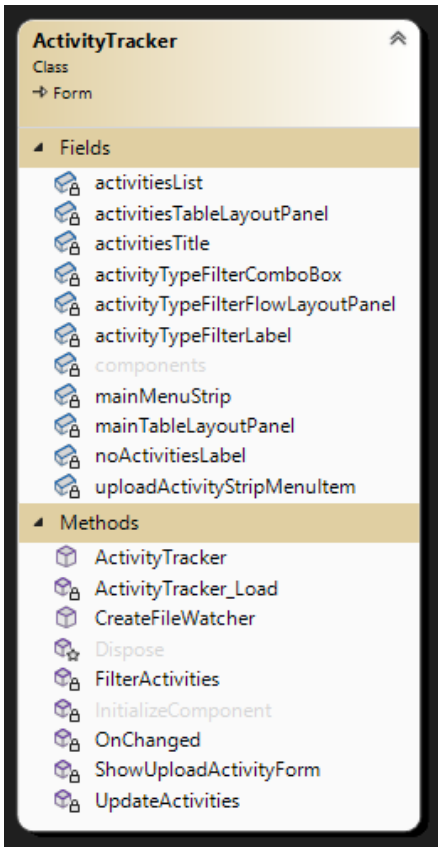


Figure 1 - Activity Tracker App

The first accessible form—the main interface—features a menu strip that includes an option to upload an activity. Below the menu, the form displays a list of user activities; if no activities have been recorded, an appropriate message is shown to inform the user (see

Figure 2). A filter is provided to allow users to display activities by type, making it easy to sort and view specific categories of activities.

Under the hood, a filewatcher continuously monitors the CSV file in the AppData/Local/ActivityTracker folder. When any modifications are detected, it automatically triggers an update of the activity list on the main form.

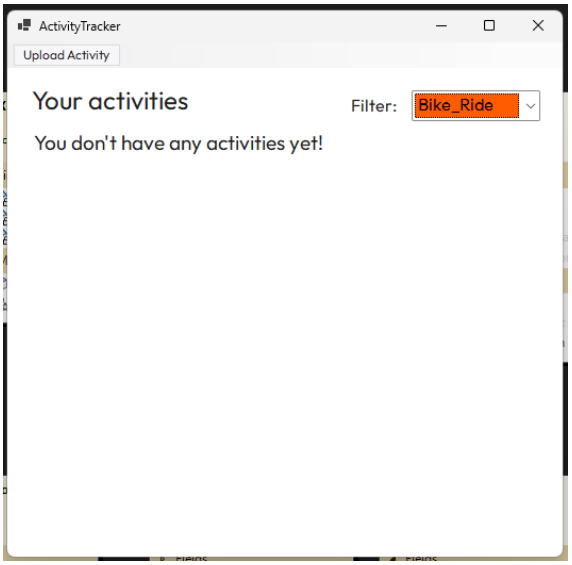


Figure 2 - Activity List

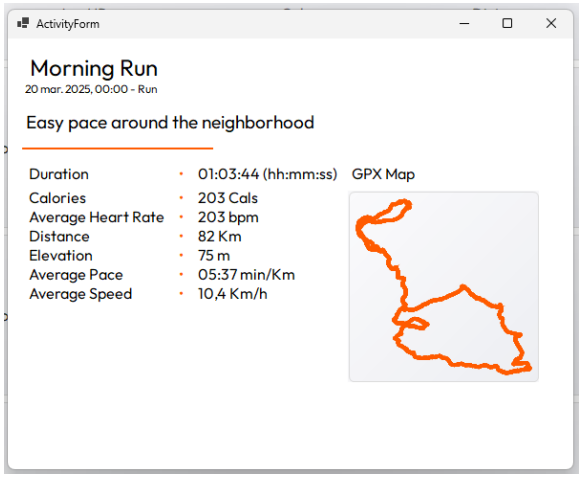


Figure 3 – ActivityForm UI

The second form, ActivityForm (see Figure 3), is displayed when a user clicks on an activity in the main list. This form presents detailed information about the selected activity. If GPX data is available, the ActivityForm incorporates a map to represent the recorded route, using a custom panel, GPXPanel (see section 3.2.5).

The final form, UploadActivityForm, features only a TabControl containing two distinct tab pages: "Manual Upload" and "File Upload" (see section 3.2.6). The Manual Upload tab allows users to input activity details directly, while the File Upload tab facilitates the import of activity data via CSV

files.

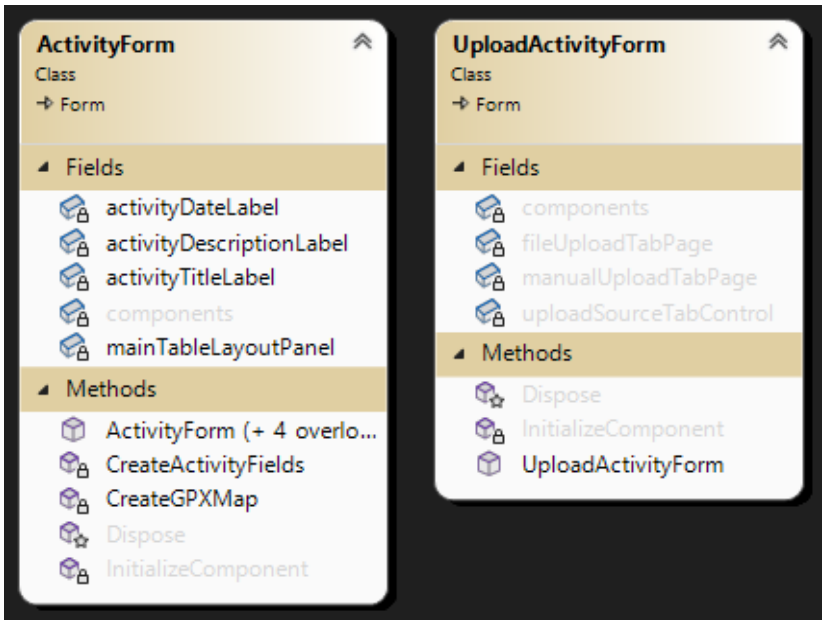
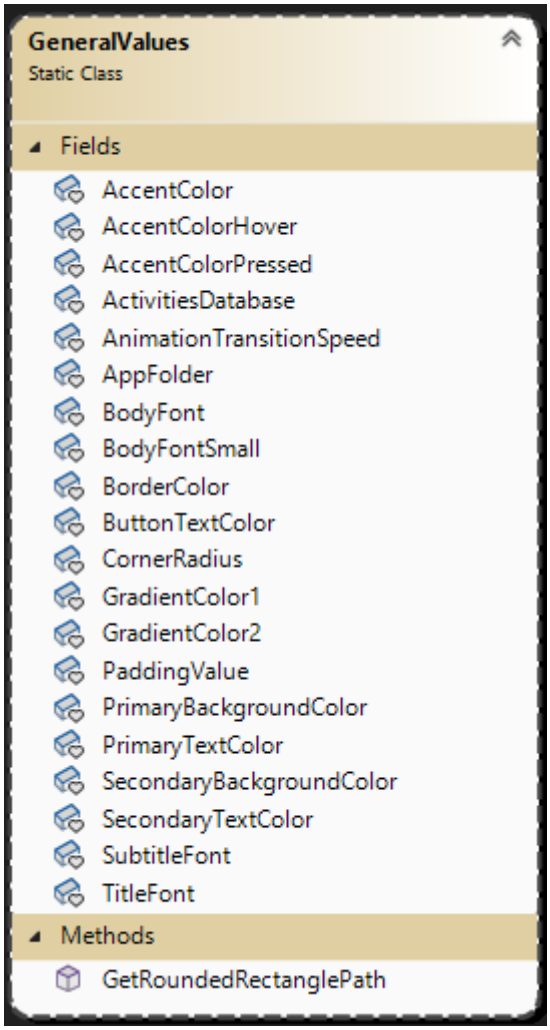


Figure 4 - ActivityForm and UploadActivityForm Classes

3.2.2. General Values



The GeneralValues class (Figure 5) functions as a centralized repository for global design parameters, ensuring consistency across the application's user interface. It defines key properties, including font settings, theme colors, and padding values, which standardize the visual style throughout the app. Additionally, the class includes a utility method that returns a rounded rectangle shape. This method is used when overriding the default OnPaint method of panels, allowing for a customized appearance.

Figure 5 - GeneralValues Class

3.2.3. Activity Class

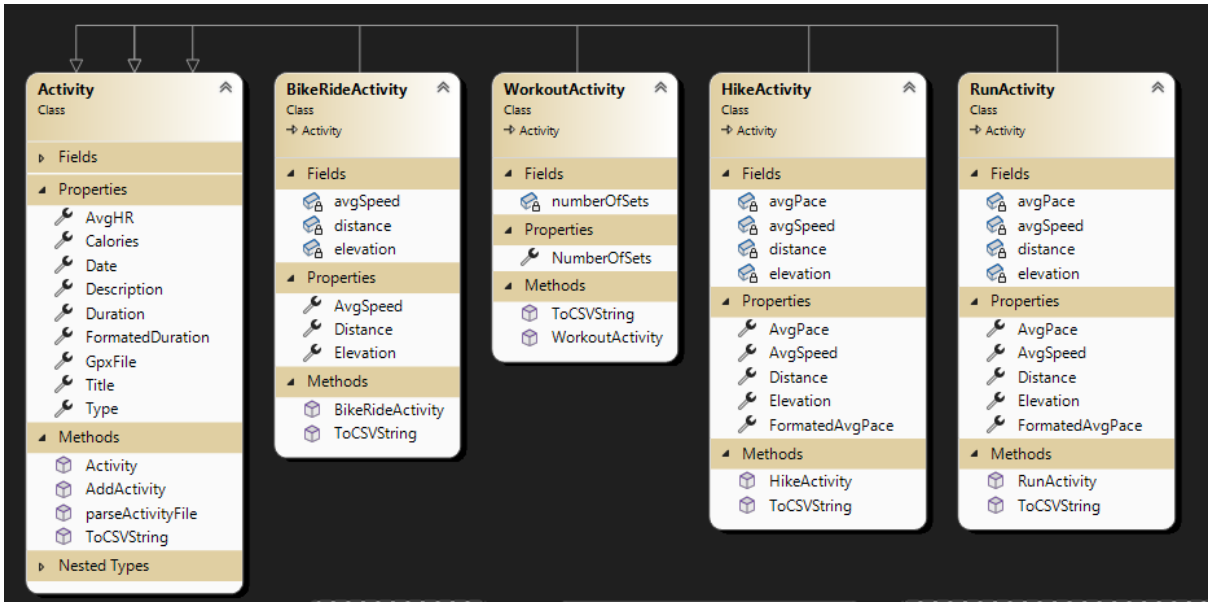


Figure 6 - Activity Class

In order to effectively manipulate the activity data, the Activity class was created. As seen in the diagram above, the base Activity class encapsulates shared properties and methods—such as average heart rate, date, and CSV serialization logic. Derived from this base class, specialized subclasses like BikeRideActivity, WorkoutActivity, HikeActivity, and RunActivity extend its functionality by incorporating fields and methods specific to each activity type. This design leverages polymorphism and promotes code reuse, making it straightforward to manage common features while allowing each subclass to implement different behaviors unique to its activity domain.

3.2.4. Custom Controls

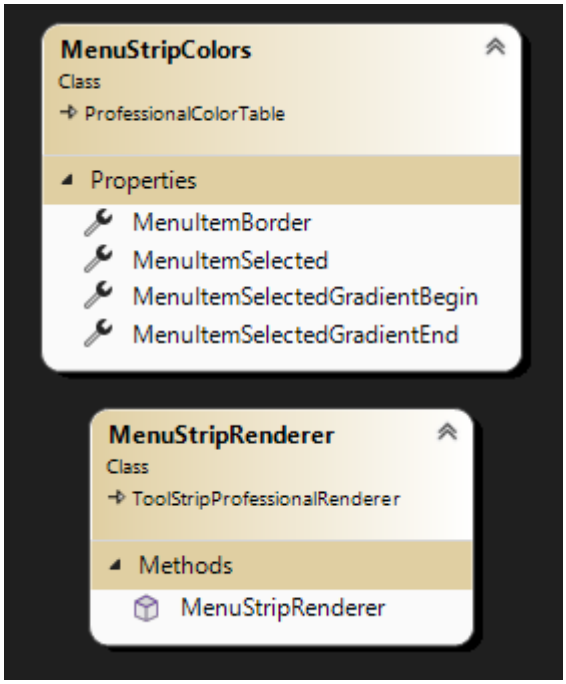


Figure 7 - MenuStripColors and Renderer Classes

For a better design of the application, custom controls were implemented (see Figure 8). All controls have been updated to use the color palette defined in the GeneralValues class, ensuring a consistent visual theme throughout the interface. Additionally, both the Button and NumericUpDown controls feature a rounded look, achieved by overriding the OnPaint method. The custom Button control also incorporates a unique transition effect that smoothly animates the change between its hover and clear states.

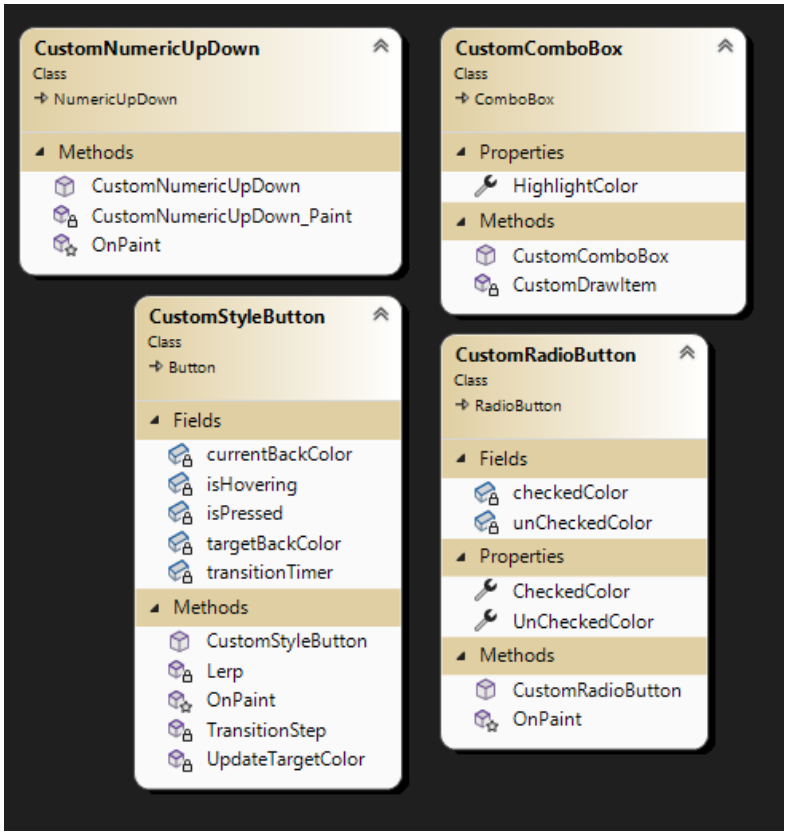


Figure 8 - Custom Controls

3.2.5. Panels

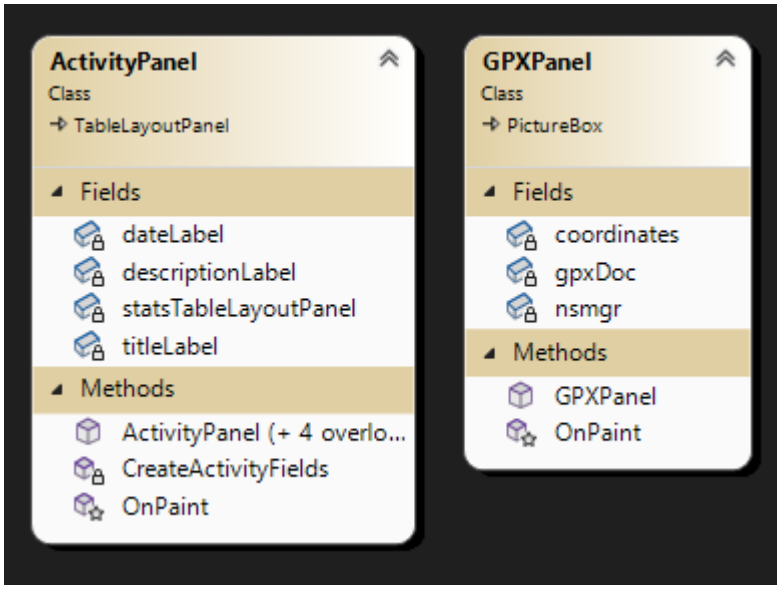


Figure 9 - ActivityPanel and GPXPanel Classes

Two custom panels were developed to further enhance the application's interface (Figure 9). The ActivityPanel displays preview statistics for an activity, including three general metrics and one specific to the activity type, offering users a concise overview of performance. Meanwhile, the GPXPanel is designed for the ActivityForm to clearly render GPX data. Both panels feature rounded corners achieved by overriding the OnPaint method.

3.2.6. Tab Pages

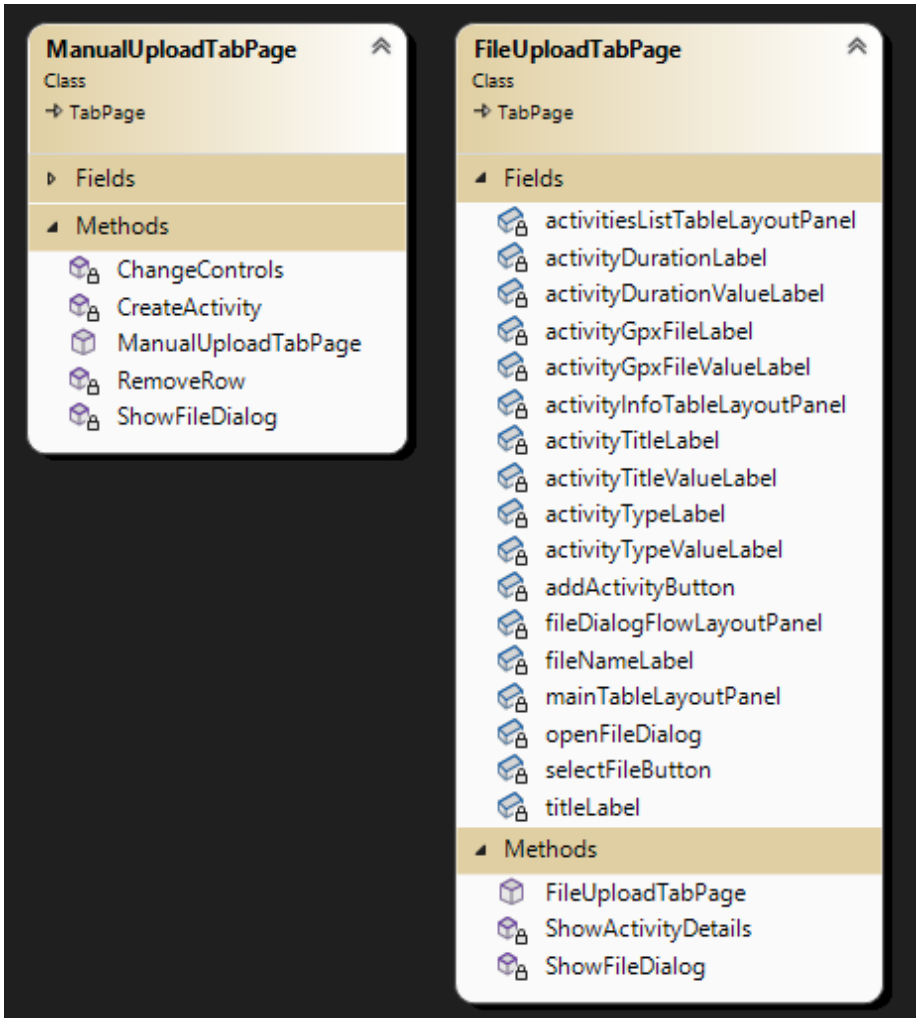


Figure 10 - ManualUploadTabPage and FileUploadTabPage Classes

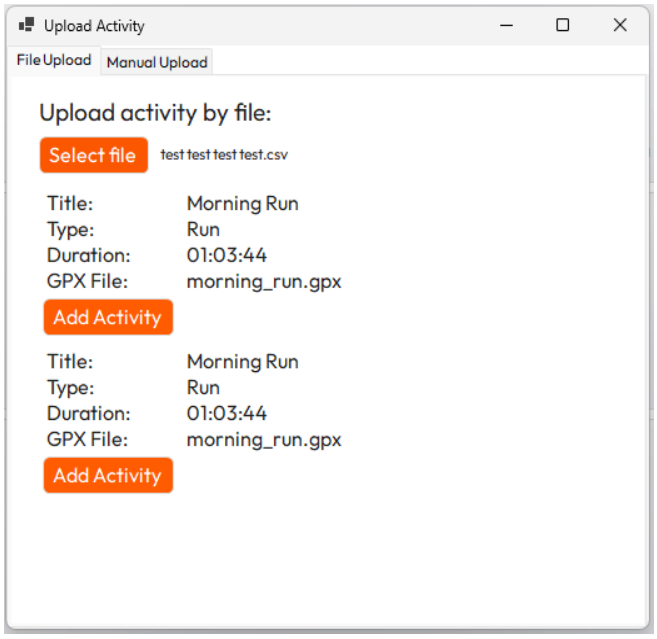


Figure 11 – FileUploadTabPage UI

In order to upload an activity, two distinct tab pages were created within the UploadActivityForm: FileUploadTabPage for uploading activity data from a CSV file and ManualUploadTabPage for manual entry.

FileUploadTabPage (see Figure 11) includes a button that, when clicked, opens a file dialog, allowing the user to select a CSV file. The CSV file is capable of containing multiple activities, and upon selection, the application parses the file to display all available activities. This design enables the user to choose and add any desired activity from the file.

ManualUploadTabPage (see Figure 12) includes multiple fields and controls designed to facilitate the entry of activity statistics. Based on the activity type selected by the user, the UI dynamically adjusts the available fields to match the specific stats required for that activity. This adaptive interface not only streamlines the data entry process but also minimizes clutter by displaying only the relevant input options for each type of activity.

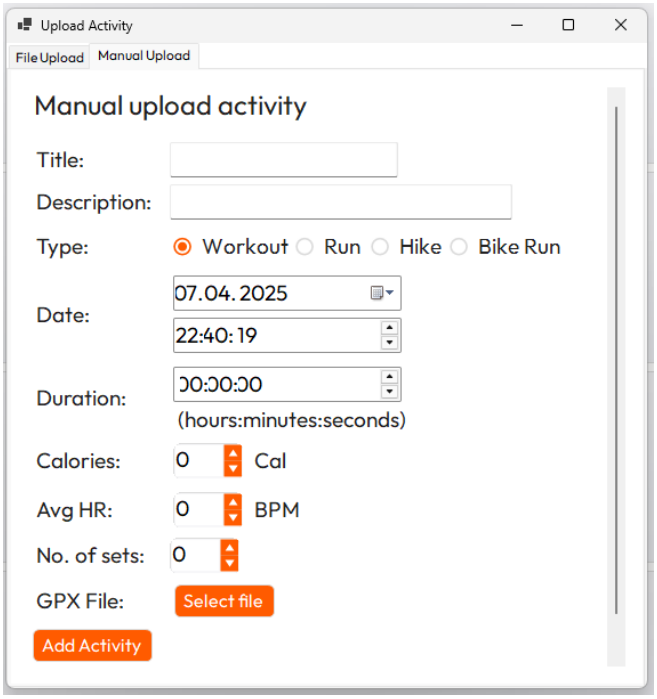


Figure 12 – ManualUploadTabPage UI

3.3. Data Format and Storage

The activity data is stored in a local CSV file located in the 'AppData/Local/ActivityTracker' folder. If this folder or the CSV file don't exist, the app automatically creates them. To prevent conflicts with commas that may appear in the description fields or in floating-point numbers, the file uses the pipe character (|) as the delimiter instead of a comma. The table below details the data format used in the CSV file, clearly outlining the structure and order of the fields.

Title	Description	Type	Date	Duration	Calories	AvgHR	GpxFile
Elevation	Distance	AvgPace	AvgSpeed	NumberOfSets			

Table 2 - Activity Data Format

4. Extras

In addition to this documentation, the archive includes the complete application source code as well as a CSV file populated with various activities to facilitate testing of the application's functionality.