Ryanair Price Scanner

Szabolcs Bencsik
PGDAI 2024.
ID: x23431962

The Problem Domain

Travelers need efficiency and convenience when planning trips.
Travel is always tied to a web of complex, interconnected questions—questions that need to be answered together, not in isolation.

These questions include things like:
"How do I get to the airport this time?"
"How will I leave the airport at my destination?"
"Can I connect this trip with other activities?"
"How long does the whole journey take?"
"Which flight should I choose?"

Each of these has multiple possible answers, each with its own impact. We can choose to focus on any one question and build the rest of the plan around it—based on our preferences or, often, dictated by circumstance. During planning, we aim to gather as much relevant information as possible and ultimately pick the best option: a solid, working chain of answers that maps out our trip from start to finish.

Even something as simple as getting flight prices and details—just one piece of the puzzle—can be a surprisingly is timewasting and nerve cracking.

Most current online tools are built around the booking process itself, earning commissions from each reservation. These platforms offer very limited access to broader price data, and trying to get a full picture becomes a repetitive, time-consuming chore.

They do their job—but there's a significant gap between the user's interests and the goals of booking platforms. Reservation sites are focused on purchase conversions. As a result, they fall short when it comes to helping users collect the information, they need for well-informed decisions. Using tools that weren't designed for research ends up being both inefficient and mentally exhausting.

Right now, users are essentially forced to rely on these sites, repeating the same searches over and over just to get the level of insight they need to feel confident in their decision. It's not uncommon for this process to take an hour or more each time.

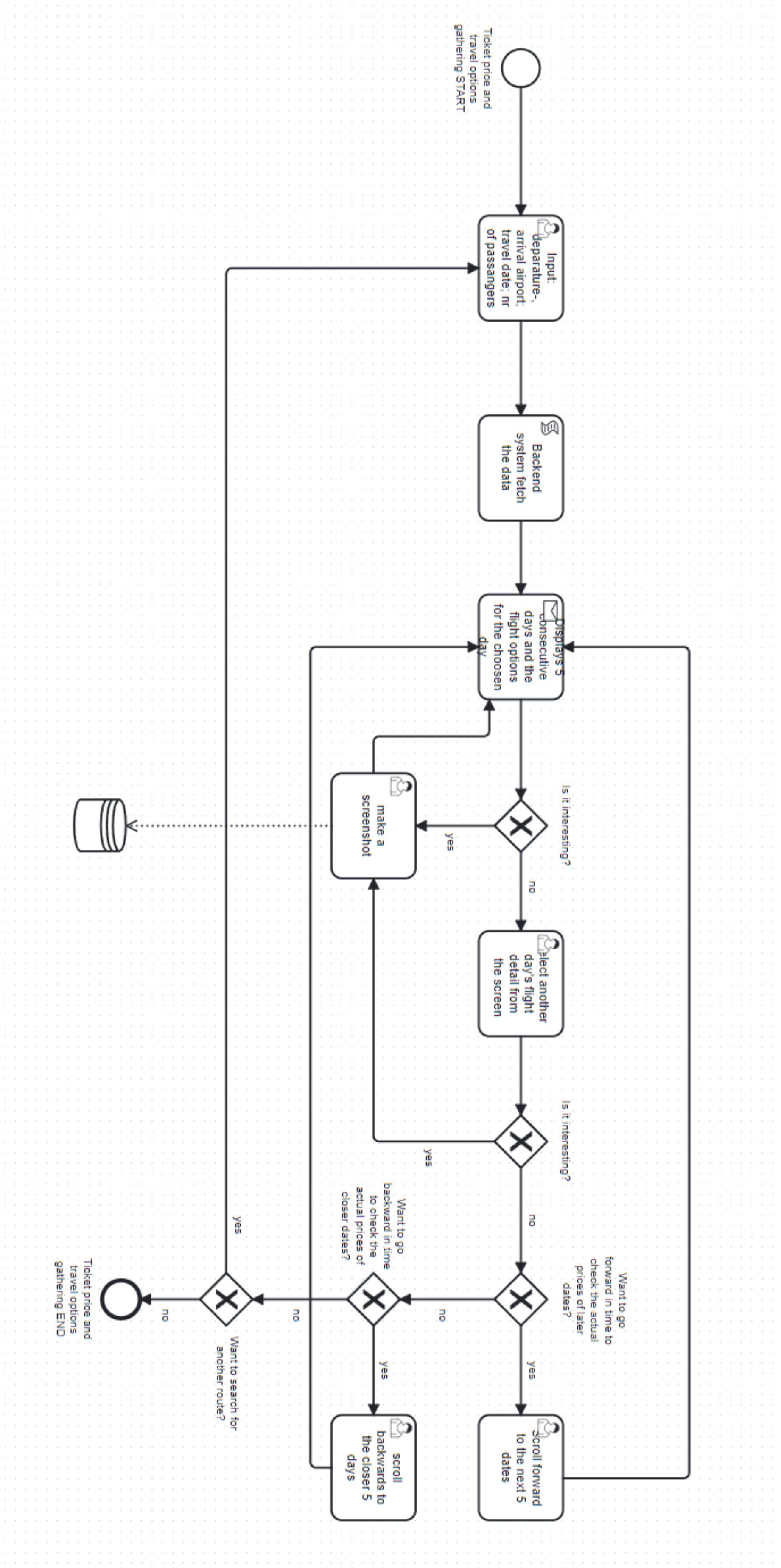
Context

To be specific, I'm focusing here on Ryanair flights, as I'm a frequent traveller and often fly with them. I can use multiple airports and quite flexible in dates. I find myself doing these searches frequently and repeatedly as my travel dates approach, weighing all the considerations I mentioned above. I'm going to review the existing tools, making clear the difference between the website and user goals, describing valid user scenarios and then design a proper tool what is up to the task. Using Business Process Model and Notation (BPMN) where necessary and python language for object orientated structural programming.

If it weren't for this project, I'd still be stuck using the old search methods and the same limited tools. But thanks to this exploration, I'm developing a working model—something I plan to make public for the benefit of fellow Ryanair passengers.

I. As-Is

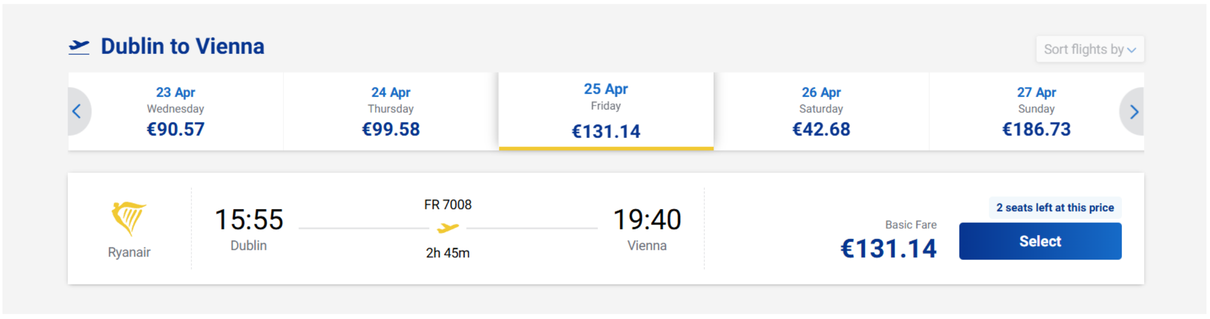
The BPMN diagram shows the whole journey of a user who tries to gather all flight price information for review.


Ticket price and travel options gathering START
Input: departure-, arrival airport, travel date, nr of passangers
Backend system fetch the data
Displays 5 consecutive days and the flight options for the choosen day
Is it interesting?
yes
no
make a screenshot
Select another day's flight detail from the screen
Is it interesting?
yes
no
Want to go forward in time to check the actual prices of later dates?
yes
no
Scroll forward to the next 5 dates
Want to go backward in time to check the actual prices of closer dates?
yes
no
scroll backwards to the closer 5 days
Want to search for another route?
yes
no
Ticket price and travel options gathering END

1. User visits Ryanair’s website.
2. Fills in departure airport, arrival airport, exact date, and number of passengers.
3. The website displays five prices (selected date plus two days before and after).
4. To check alternative dates, the user must manually click through multiple screens and save the interesting ones.
5. The user can only view fares for a single route at a time.

Analysis

Fundamentally, all booking sites share a common flaw: they don’t prioritize the task of information gathering. They were built for a different purpose—facilitating transactions—not for helping users explore options or understand the full picture. And yet, users are forced to rely on them because there are no better alternatives.
These platforms are optimized for spending money, not for making well-informed decisions. In fact, they often blur the big picture using a mix of techniques: overwhelming amounts of data, cluttered or inconsistent display formats, and even subtle psychological nudges designed to influence behaviour.
The example below is taken from the Ryanair website, but this issue extends to nearly all booking platforms. Even aggregators that display flights from multiple airlines are limited—they typically only return results for an exact date, offering little flexibility or visibility into the broader range of options.



Dublin to Vienna

Sort flights by

23 Apr Wednesday	24 Apr Thursday	25 Apr Friday	26 Apr Saturday	27 Apr Sunday
€90.57	€99.58	€131.14	€42.68	€186.73

Ryanair 15:55 Dublin — FR 7008 2h 45m — 19:40 Vienna — Basic Fare €131.14 — 2 seats left at this price — Select

Key Issues in Interface Design

Data Availability
Only five days of data are visible at any given time. This violates the principle of visibility and limits users' ability to understand the broader context or explore flexible travel options.

Data Layout
Prices are displayed in a flat, neutral style. There’s no visual hierarchy or design context that helps users compare prices immediately. We can’t “feel” the highs and lows—there’s no intuitive way to spot the peaks and dips.

Display Format
There are no trendlines or visual cues that reveal price movement over time. Each piece of information is siloed, leaving the user to do all the cognitive heavy lifting.

Manipulative Design Tactics
Messages like “Only 2 seats left at this price” act as psychological pressure points. They serve as implicit calls to action, nudging users to make quick decisions—often without taking the time to compare or plan.

These platforms are clearly designed to sell, not to support meaningful information gathering. The user's need for clarity, context, and exploration is sidelined in favour of urgency and conversion.

Manual Processes

1.User selects departure-, arrival airport names, travel date, the number of passengers and press a button.
2.User reviews the displayed 5 options and click to one what looks like a potential hit.
3.User saves potential hit to his computer as a screenshot.
4.User clicks to the forward button.
5.User repeating manual task 2, 3 and 4.
6.User clicks to backward button
7.User repeating manual task 2, 3 and 4.
8.User selecting another route and repeat manual tasks from 1 to 7.
9.User opens the saved screenshots.

Cost Impact

The overall cost of using current booking platforms isn't primarily monetary—but that doesn't mean it's insignificant. In fact, the opposite is true. The real pain lies in the time wasted and the frustration of using a tool that's fundamentally unfit for the task.
Monetary implications often appear indirectly, as side effects of other limitations. Since travel decisions are built on a chain of interconnected questions, poor visibility or access to information can distribute financial consequences unevenly across those decisions.

Inefficiency
The system doesn't support flexible travellers who need a broader price comparison. It doesn`t support comparison across dates and airports.

Missed Savings
Users may miss more suitable travel chain simply because the platform makes it too hard to recognise.

Time Consumption
Checking multiple date and airport combinations manually is time consuming. Each session can easily take an hour or more, time that could be monetized by applying the user's personal value of free time (which is for most people priceless) and maybe have to doublecheck again at the time of booking because the ticket prices are going up with the time.

Manual Labor and Inefficient Data Handling
Users must repeatedly input the same data, navigate multiple pages, select relevant pieces of information, save screenshots for comparison.

Error Rates

Rather non-existent at input entry. Giving a few inputs to the system is easy and any time we can correct it. Might be issue with the data saving where must pay attention the file names and saving location.
Also considering as an error a potentially fit flight option what was opted out due to the incapable data presentation.

Lost Productivity:

It is significant. Spending on wrong tool an hour for a simple task what could be done within seconds are nerve wrecking and wasted time completely.

II. To-Be

Instead of relying on tools that were never designed for thoughtful planning, let's build one that is. A tool that eliminates the weaknesses of existing platforms and puts the user's needs to focus!

Identifying User's Goals

User Scenarios:

Frequent Traveler with Flexible Dates:
- Often travels for business or leisure but prefers to book when fares are lowest.
- Needs a comprehensive price overview across a month or more to make cost-effective decisions.

Multi-Airport User:
- Willing to use different airports to find the best deals.
- Requires an easy way to compare prices across multiple departure and arrival points.

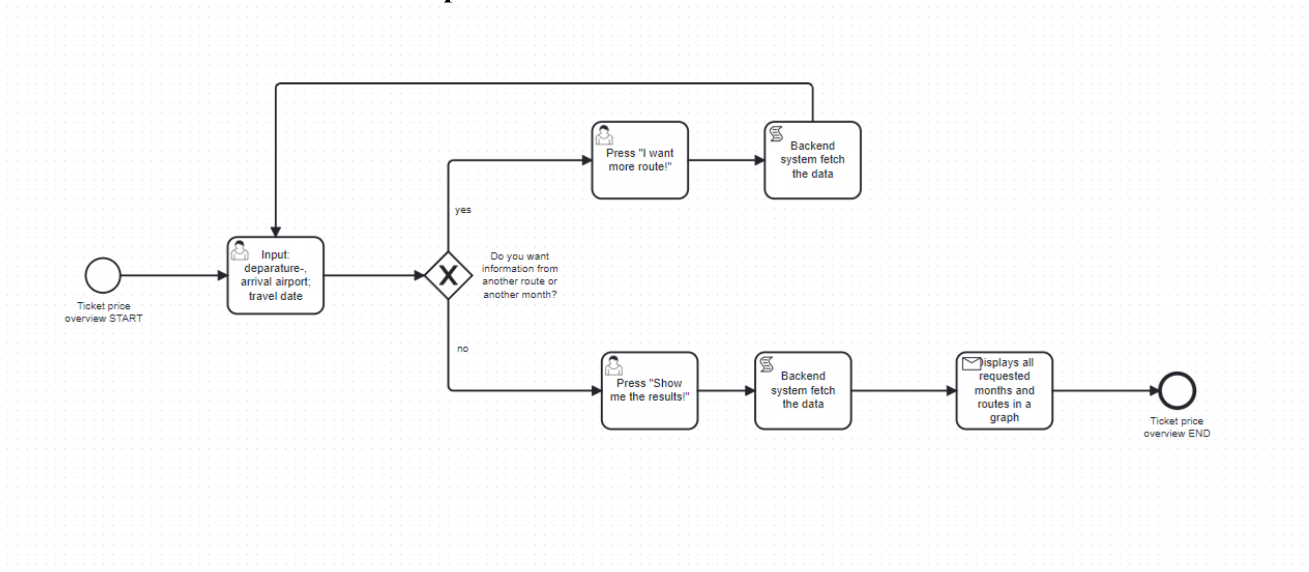
Flexible but Context-Driven Traveler:
- Factors beyond airfare impact travel choices, such as restaurant visits, airport shuttle availability, and shopping hours.
- Needs to compare fares alongside other logistical considerations.

Undecided Traveler:
- Knows they need to travel but is unsure when due to external life factors.
- Seeks a simple, real-time price overview to determine when to book before fares rise.

Combination User:
- A user who falls into multiple categories, needing flexible options with broad search criteria.

We need a solution that **visualizes real data**, with **minimal user interaction**, in a **fast, responsive, and repeatable** way. It should empower users to explore—not just buy.

Proposed Automation Solution


Ticket price overview START
Input: deparature-, arrival airport; travel date
Do you want information from another route or another month?
yes
no
Press "I want more route!"
Backend system fetch the data
Press "Show me the results!"
Backend system fetch the data
isplays all requested months and routes in a graph
Ticket price overview END

Proposed Process Flow

1. Inputs multiple queries with:
   - Departure and arrival airport codes. (3 letters)
   - Travel month nr, year box is filled out but can be changed.
2. Can ask info from multiple routes, dates before going to next step.
3. Clicks “Finish” to display the result of all queries at the same time.
4. The system retrieves all relevant fare data and displays it in a graphical format for at least a month’s period.

Cost Improvement and Tool Enhancement

The new tool eliminates the inefficiencies and frustrations of the old method, offering both a cost-effective and user-centric experience.

What the user gains:

Effortless comparison:
Users can easily compare fares across different dates and routes without multiple searches

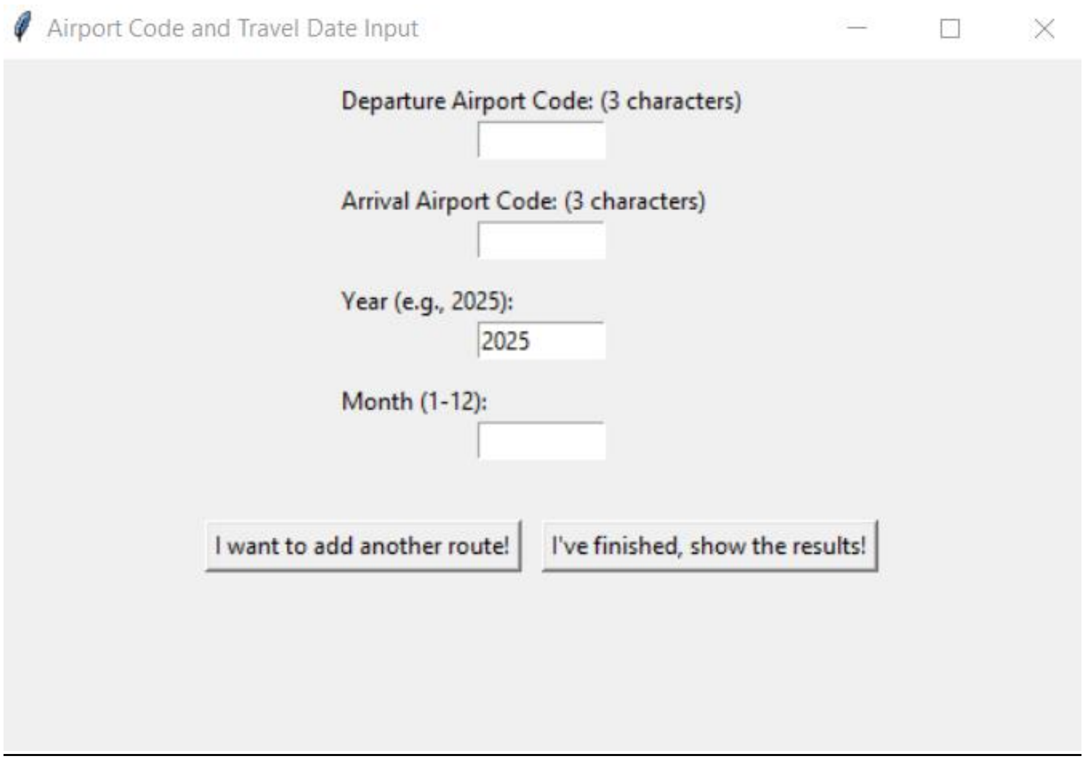
Instant, comprehensive data:
The full pricing picture is always available within seconds, at any time.

Clear visual context:
Data is presented in a way that highlights trends, peaks, and dips to make easy to identify the best options immediately.

Minimal input required:
The interface is unambiguous and requires only the bare minimum of user input to get started.

All costs eliminated:
All the time, effort, and frustration of the old data-hunting process is addressed.

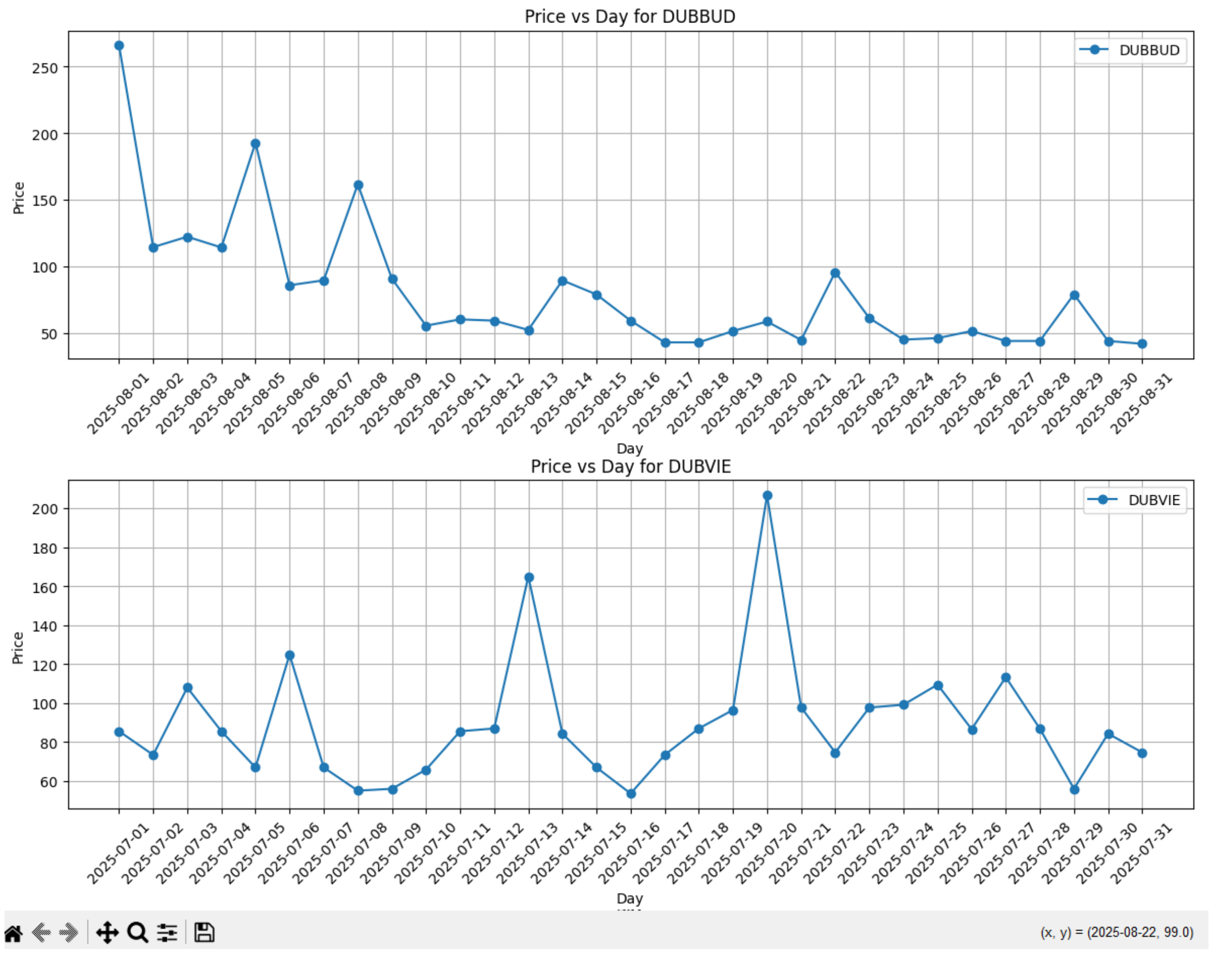
Solution Demonstration


Airport Code and Travel Date Input
Departure Airport Code: (3 characters)
Arrival Airport Code: (3 characters)
Year (e.g., 2025):
2025
Month (1-12):
I want to add another route!
I've finished, show the results!

User Interface Highlights:

-The **default year** is pre-filled as the current year but can easily be changed.
-Users can **navigate using the Tab key**, supporting multi-way navigation.
-Input expectations are clearly labelled next to each field.
-Case-insensitivity allows users to **type in capital or lowercase letters freely**.
-Instead of full date formats, users simply enter the **month number**, reducing the chance of formatting errors.
-In case of any incorrect input, the **error messages provide clear guidance**.

Below screenshot shows the actual price in Dublin-Budapest (August), and Dublin-Vienna (July) prices. When user hoovers on the graph at the left bottom corner the data displayed in numeric format as well.
The program concatenates the data for the same route and showing in the same graph. If we ask the 8th and 9th month of the same route, then we get a graph where both two months displayed.
All different route displayed below each other.

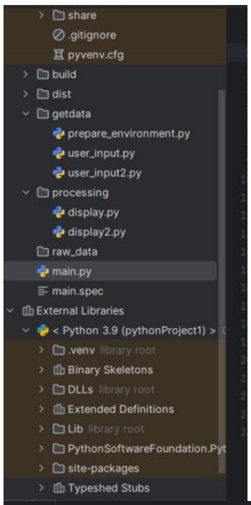


This tool shifts the experience from frustrating and time-wasting to seamless and empowering. It’s not just better, it’s fundamentally aligned with how people plan to travel.

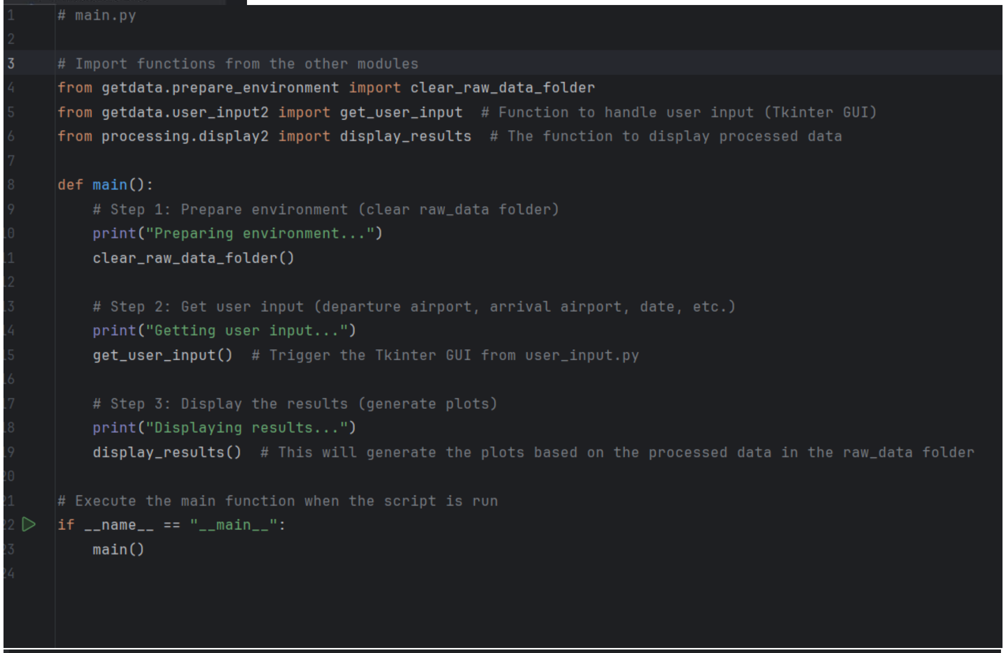
The solution was created with PyCharm in line with the OOP principles. Part of the python code shared in the appendix, for full solution visit:
https://github.com/bencsixabolcs/ryanair_pricescanner

Runnable windows version is available here:
https://drive.google.com/drive/folders/1rlG4P5TCEDNBzbRTMyNHe9O1j5YNXaPd?usp=sharing

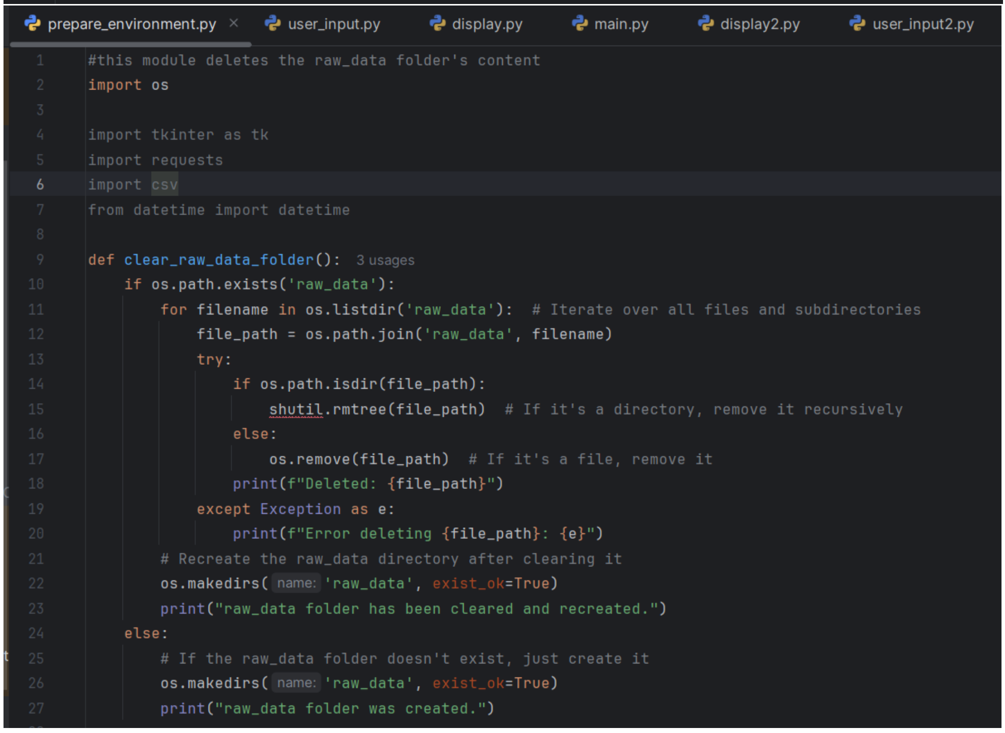
Walk through is part of the .pptx presentation.



```
> share
  .gitignore
  pyvenv.cfg
> build
> dist
v getdata
    prepare_environment.py
    user_input.py
    user_input2.py
v processing
    display.py
    display2.py
  raw_data
  main.py
  main.spec
v External Libraries
  v < Python 3.9 (pythonProject1) >
    > .venv library root
    > Binary Skeletons
    > DLLs library root
    > Extended Definitions
    > Lib library root
    > PythonSoftwareFoundation.Pyt
    > site-packages
    > Typeshed Stubs
```



```python
# main.py

# Import functions from the other modules
from getdata.prepare_environment import clear_raw_data_folder
from getdata.user_input2 import get_user_input  # Function to handle user input (Tkinter GUI)
from processing.display2 import display_results  # The function to display processed data

def main():
    # Step 1: Prepare environment (clear raw_data folder)
    print("Preparing environment...")
    clear_raw_data_folder()

    # Step 2: Get user input (departure airport, arrival airport, date, etc.)
    print("Getting user input...")
    get_user_input()  # Trigger the Tkinter GUI from user_input.py

    # Step 3: Display the results (generate plots)
    print("Displaying results...")
    display_results()  # This will generate the plots based on the processed data in the raw_data folder

# Execute the main function when the script is run
if __name__ == "__main__":
    main()
```



prepare_environment.py × | user_input.py | display.py | main.py | display2.py | user_input2.py

```python
#this module deletes the raw_data folder's content
import os

import tkinter as tk
import requests
import csv
from datetime import datetime

def clear_raw_data_folder():  3 usages
    if os.path.exists('raw_data'):
        for filename in os.listdir('raw_data'):  # Iterate over all files and subdirectories
            file_path = os.path.join('raw_data', filename)
            try:
                if os.path.isdir(file_path):
                    shutil.rmtree(file_path)  # If it's a directory, remove it recursively
                else:
                    os.remove(file_path)  # If it's a file, remove it
                print(f"Deleted: {file_path}")
            except Exception as e:
                print(f"Error deleting {file_path}: {e}")
        # Recreate the raw_data directory after clearing it
        os.makedirs( name: 'raw_data', exist_ok=True)
        print("raw_data folder has been cleared and recreated.")
    else:
        # If the raw_data folder doesn't exist, just create it
        os.makedirs( name: 'raw_data', exist_ok=True)
        print("raw_data folder was created.")
```