Chatbot project

(Travel Advisor Chatbot)

This report is to show and explain all the Chabot project info, missions, errors and how we overcame all of them.

Introduction about the project

What is a Chatbot?

A Chatbot is a computer program designed to simulate conversation with human users, especially over the internet. Chabot’s are typically powered by artificial intelligence algorithms that enable them to understand and respond to natural language inputs from users. They can be used for a variety of purposes, such as customer service, providing information, entertainment, or assisting with tasks like scheduling appointments or ordering products.

Historic outline

1950s-1960s: Alan Turing proposes the Turing Test in 1950. Joseph Weizenbaum develops ELIZA in 1966.

1970s-1980s: PARRY is developed in 1972. Racter is introduced in 1984.

1990s-2000s: Smarter Child is launched in 2000.

2010s: Siri is introduced in 2011. Google Assistant debuts in 2016. Amazon's Alexa is released in 2014.

Present: Chabot’s continue to advance with improvements in AI technologies, shaping human-computer interaction across various domains

Importance/Advantages of Chatbots:

* 24/7 Availability: Chatbots can provide round-the-clock assistance, ensuring that customers can access support or information anytime they need it.
* Cost-Effective: Chatbots can handle a large volume of inquiries simultaneously, reducing the need for human customer service representatives and saving costs for businesses.
* Efficiency: Chatbots can automate routine tasks, such as answering frequently asked questions or processing simple transactions, allowing human agents to focus on more complex issues.
* Scalability: Chatbots can scale easily to accommodate growing user demands without significant additional resources.
* Consistency: Chatbots provide consistent responses to inquiries, ensuring that users receive uniform information and service quality across interactions.

How do Chatbots work?

Chatbots work by processing natural language inputs from users and generating appropriate responses. The underlying technology typically involves:

* Natural Language Processing (NLP): NLP enables chatbots to understand and interpret human language inputs, including text and speech.
* Machine Learning: Many chatbots utilize machine learning algorithms to improve their understanding and response generation over time, based on interactions with users.
* Knowledge Base: Chatbots may be integrated with a knowledge base or database containing information relevant to their function, enabling them to retrieve and provide accurate information to users.
* Conversation Management: Chatbots manage conversational flow, context, and state to maintain coherent interactions with users.
* Integration: Chatbots can be integrated with various platforms and systems, such as messaging apps, websites, or business applications, to deliver their services.

Types of chatbots

* Rule-Based Chatbots: These chatbots operate based on pre-defined rules and patterns. They follow a decision tree or script to generate responses to user inputs.
* AI-Powered Chatbots: AI-powered chatbots leverage machine learning and natural language processing to understand and respond to user inputs more intelligently. They can handle more complex queries and adapt their responses based on context and user behavior.
* Transactional Chatbots: Transactional chatbots are designed to facilitate transactions, such as making purchases, booking appointments, or processing payments, through conversational interfaces.
* Informational Chatbots: Informational chatbots provide users with information on various topics, such as weather updates, news headlines, or product details, based on user queries.
* Virtual Assistants: Virtual assistants, like Siri, Google Assistant, and Alexa, are advanced chatbots that offer a wide range of functions beyond basic conversation, including device control, web search, and personal organization.

Motivation

The domain or topic selected

Travel advisor chatbot

Why did we choose this ?

* Specific Purpose: Travel planning and assistance is a common need for many people, whether it's for leisure or business purposes. By focusing on this domain, we can address a practical and widely relevant topic that can benefit a broad audience.
* User Engagement: Providing users with a travel advisor chatbot offers a valuable service that can enhance their travel experiences. It engages users in meaningful interactions, assisting them in itinerary planning, destination recommendations, accommodation bookings, and more.
* Market Demand: The travel industry is continuously evolving, and there's a growing demand for digital solutions that streamline travel planning and booking processes. By offering a travel advisor chatbot, we can tap into this market demand and potentially attract users seeking convenient and personalized travel assistance.
* Team Interest and Expertise: Our team members have a shared interest in travel and technology. Developing a travel advisor chatbot aligns well with our skills and expertise, allowing us to leverage our passion for travel to create a valuable product.

# Analysis and design

Components

1. TravelInfo
   * Purpose: Stores data about travel destinations including destination name, accommodation type, cost, and transport details.
   * Properties: destination (String), accommodationType (String), cost (Double), transport (String).
2. UserPreferences
   * Purpose: Manages and stores user-specific preferences to customize interactions and suggestions.
   * Methods:
     + setPreference(pref: String): Stores a preference.
     + clearPreference(): Clears stored preference.
     + getPreference: Retrieves current preference.
3. TravelAssistantChatbot
   * Purpose: Serves as the central management object handling all user interactions, data management, and flow control.
   * Properties:
     + userPreferences: Instance of UserPreferences for storing user session data.
     + travelData: List of TravelInfo objects representing available travel data.

Main Functionalities

1. Greeting and Introduction
   * Outputs a greeting and a brief explanation of the bot’s capabilities and how to interact with it.
2. Data Handling
   * Reads travel data from a CSV file, parses it, and stores it in a structured list.
   * Filters travel information based on user inputs like destination and budget.
3. User Interaction
   * Handles various user inputs to trigger specific functionalities (e.g., listing trips, booking).
   * Extracts keywords from user inputs to understand commands and preferences.
   * Dynamically responds based on detected keywords like "recommend", "book", or direct destination queries.
4. Booking and Recommendations
   * Manages the booking process by matching user's desired trip criteria against available data.
   * Recommends trips based on user preferences such as maximum price or specific destinations.

Methods Overview

* Utility Methods: Methods like extractKeywords, isGreeting, and extractMaxPriceFromInput parse and understand user inputs.
* Process Handlers: handleUserInput, initiateBooking, and recommendDestination handle major operational flows such as initiating a booking or recommending a destination based on user queries.
* Data Management: readCSVFile and saveDetails handle external data interactions both for reading initial data and storing transactional data like bookings.

Design Considerations

* Modularity: Each class and method handles specific aspects of the application, ensuring modularity and separation of concerns.
* Data Handling: Robust handling of data with try-catch blocks and pattern matching ensures that the application gracefully handles errors like format issues in the CSV file or incorrect user inputs.
* User Interaction: The application is designed to interact continuously with the user in a loop until the user decides to exit, making it user-driven and responsive.

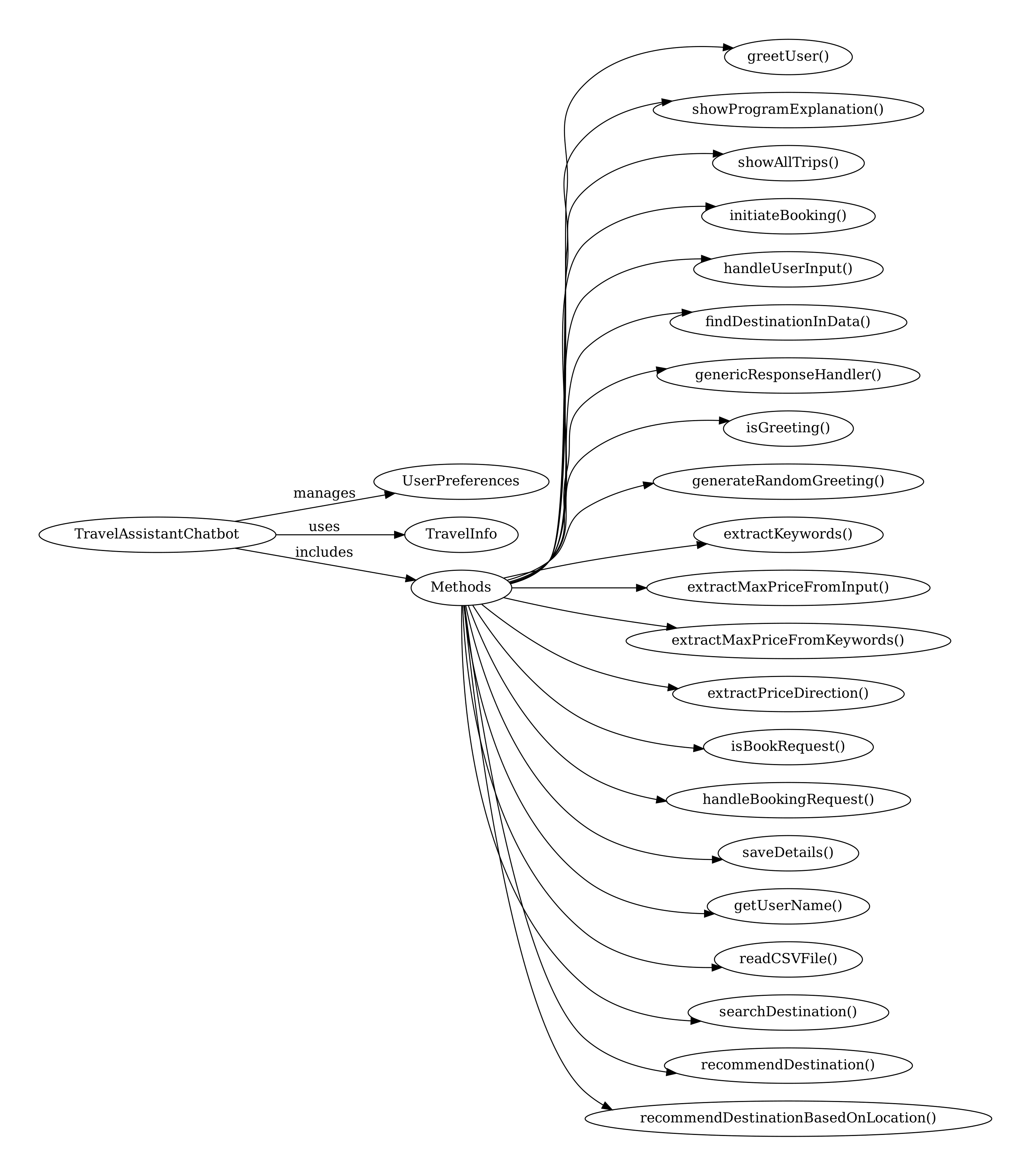
Improvement Suggestions

* Expand Error Handling: Further robustify the application by handling more edge cases and providing user-friendly error messages.
* Enhance User Experience: Introduce more detailed interactions and options, like filtering based on multiple criteria and providing detailed help commands.
* Scalability: For scalability, consider moving the data storage to a database system instead of a CSV file to handle larger datasets and more complex queries efficiently.

Conclusion

The design and functionality of the TravelAssistantChatbot are centered around a user-friendly experience for travel planning with an emphasis on simplicity and responsiveness. Future enhancements could focus on richer user interactions and more robust data management to scale up the application.

# Hierarchy Chart



# Dialogue or Conversation Flow

Description: The bot initiates the conversation by greeting the user and explaining its functionality. The user can then input commands or queries, which the bot processes through the handleUserInput function, invoking specific functions based on the input type (e.g., booking, recommendations).

# Description

# Theory Behind the Code

The chatbot leverages functional programming principles to manage data and state effectively. It uses case classes for structured data, encapsulation for user preferences, and pattern matching for input processing. Error handling is managed via Try, Success, and Failure constructs to handle potentially erroneous operations gracefully.

# Explanation of Chat Rules

The chatbot's rules are designed to interpret user inputs dynamically. It prioritizes recognizing greetings, extracting key phrases or numbers (like price limits), and determining user intents such as booking or getting recommendations. The chatbot maintains a flexible flow, allowing users to navigate back and forth between different functionalities seamlessly, guided by their inputs and the chatbot's responses.

# Workload distribution

### Hazem Refaat

**1.handleUserInput(input: String): String**

**2.** **showAllTrips()**

**3.initiateBooking(destination:Option[String]= None, cost: Option[Double] = None): String**

**4.isGreeting(input: String): Boolean**

**5.extractMultipleDestinations(keywords: List[String]): List[String]**

### Mohamed Hamdy

**1.handleBookingRequest(bookingType: String): String**

**2.searchDestination(userPreferences: UserPreferences): String**

**3.readCSVFile(filePath: String): Unit**

**4.generateRandomGreeting(name: Option[String]): String**

**5.isBookRequest(input: String): Boolean**

### Ali Amr

**1.recommendDestination(maxPrice: Double, destination: String):**

**String**

**2.extractKeywords(input: String): List[String]**

**3.extractPriceDirection(keywords: List[String]): Option[String]**

**4.destinationFromKeywords(keywords: List[String]): String**

### Yassine Hussein

### **1. main(args: Array[String]): Unit**

### **2. saveDetails(name: String, number: String, info:**

### **3. TravelInfo,bookingType: String): Unit**

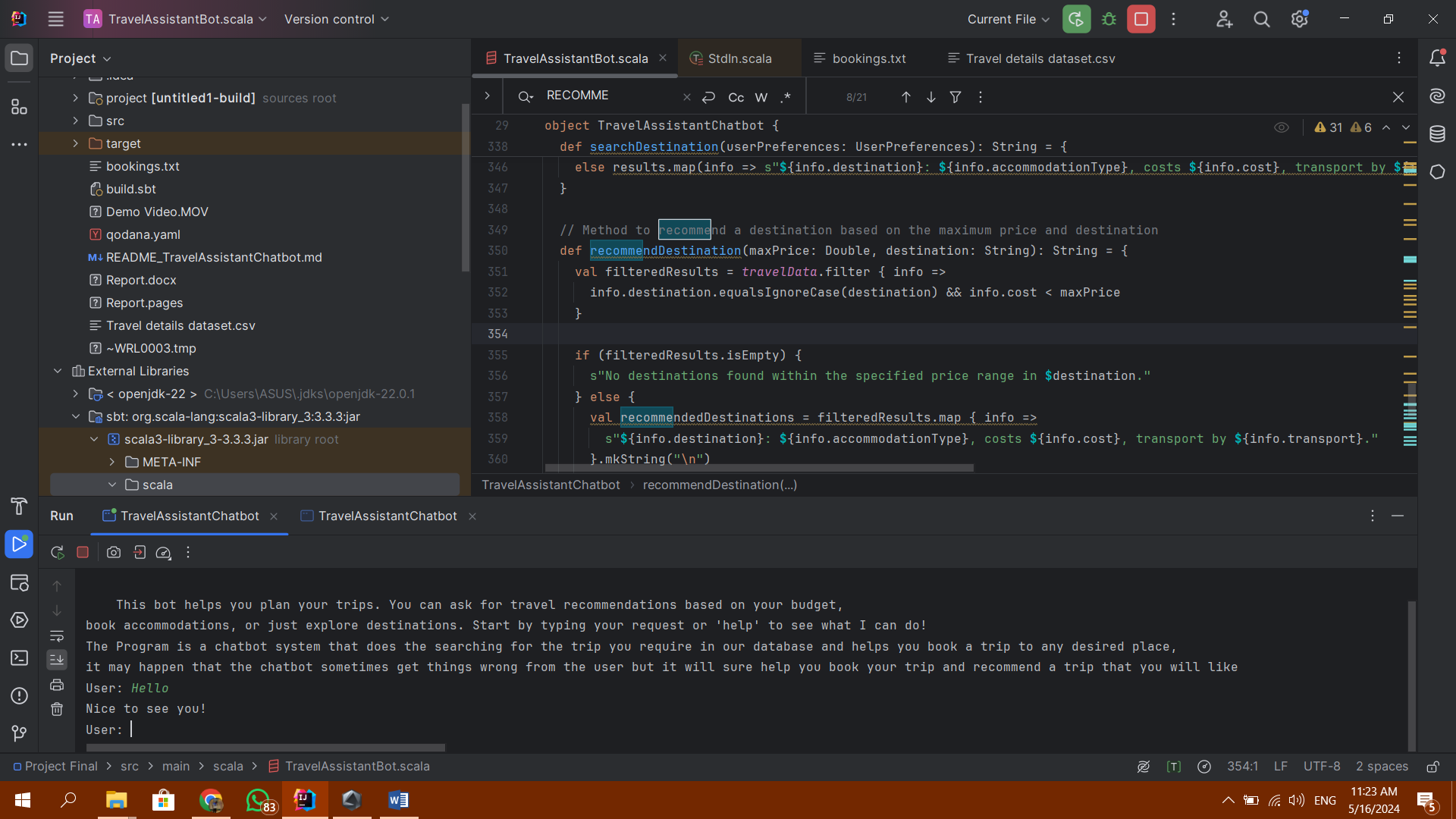
### **4 .recommendDestinationBasedOnLocation(input: String): String**

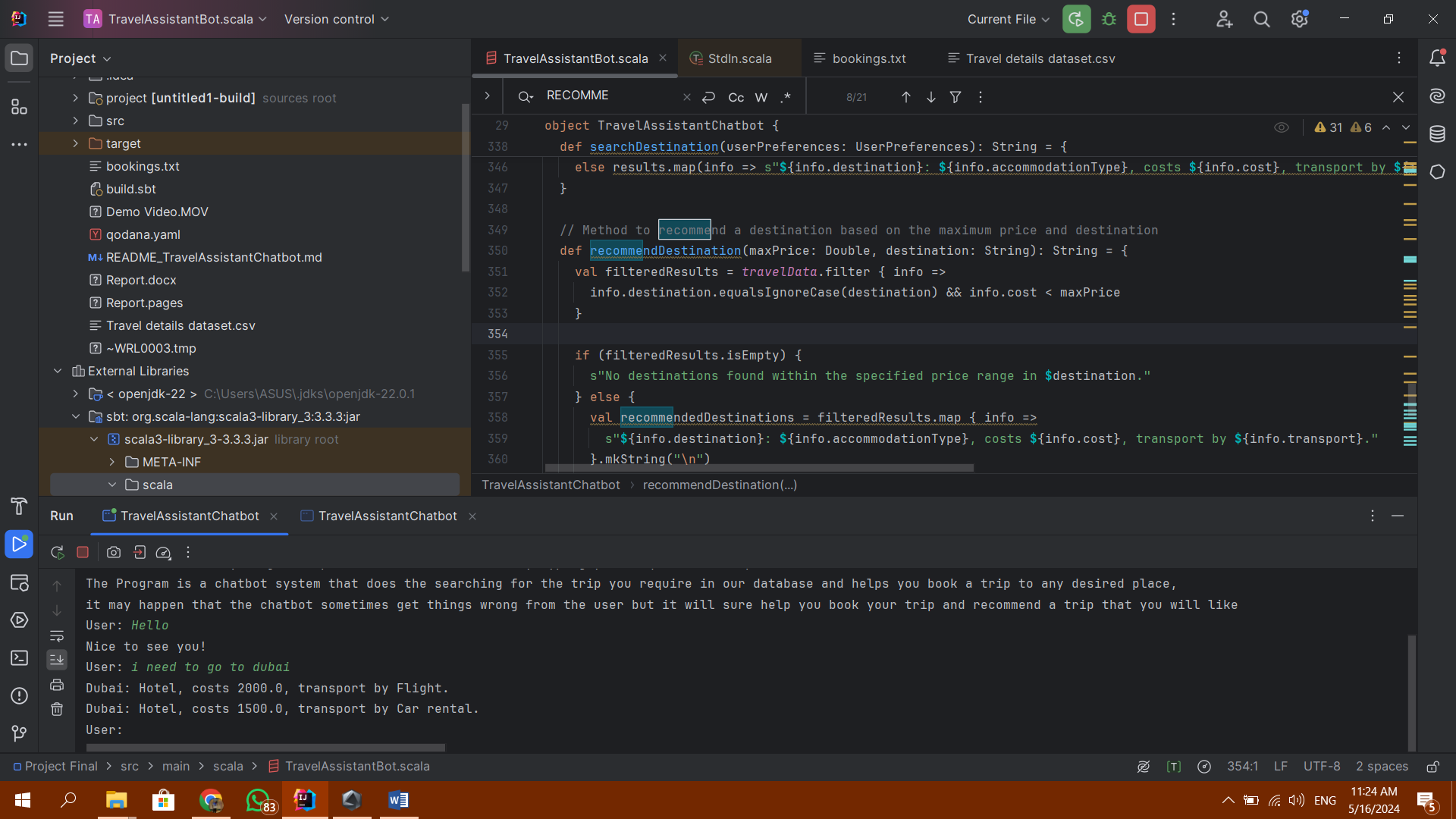
### **5.extractMaxPriceFromInput(keywords: List[String]): Option[Double]**

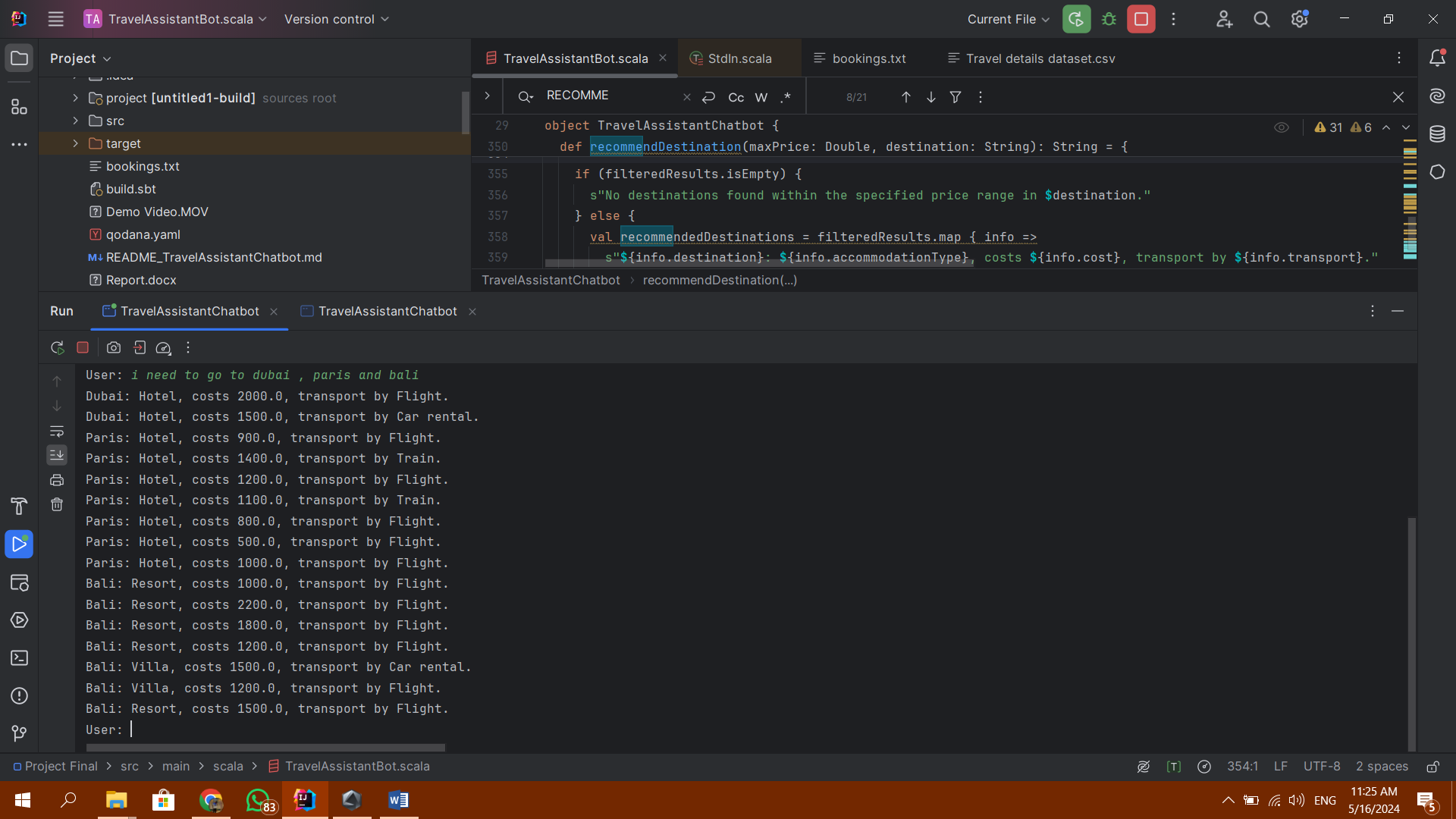
Important Note:

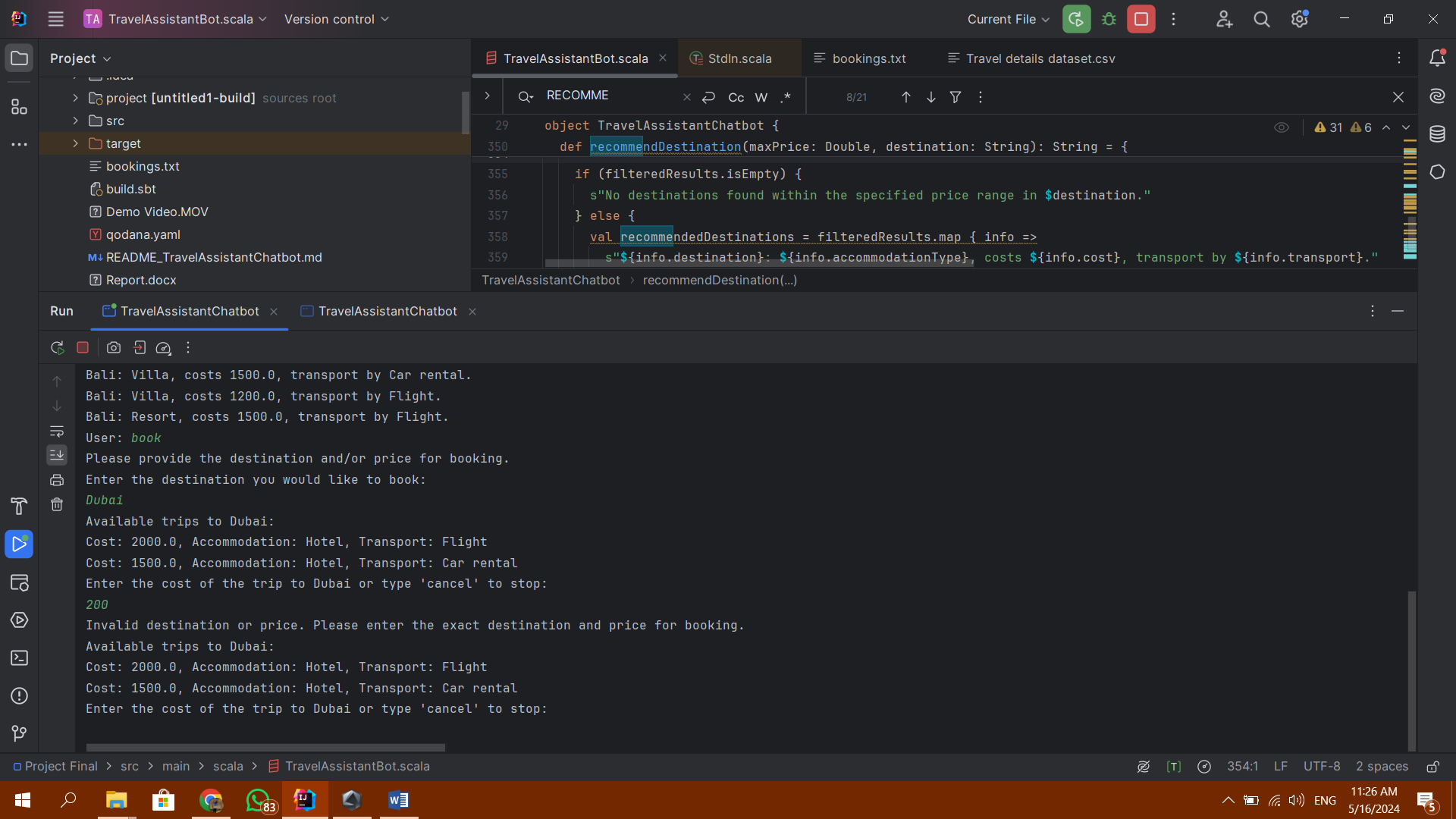
All modification made in group.

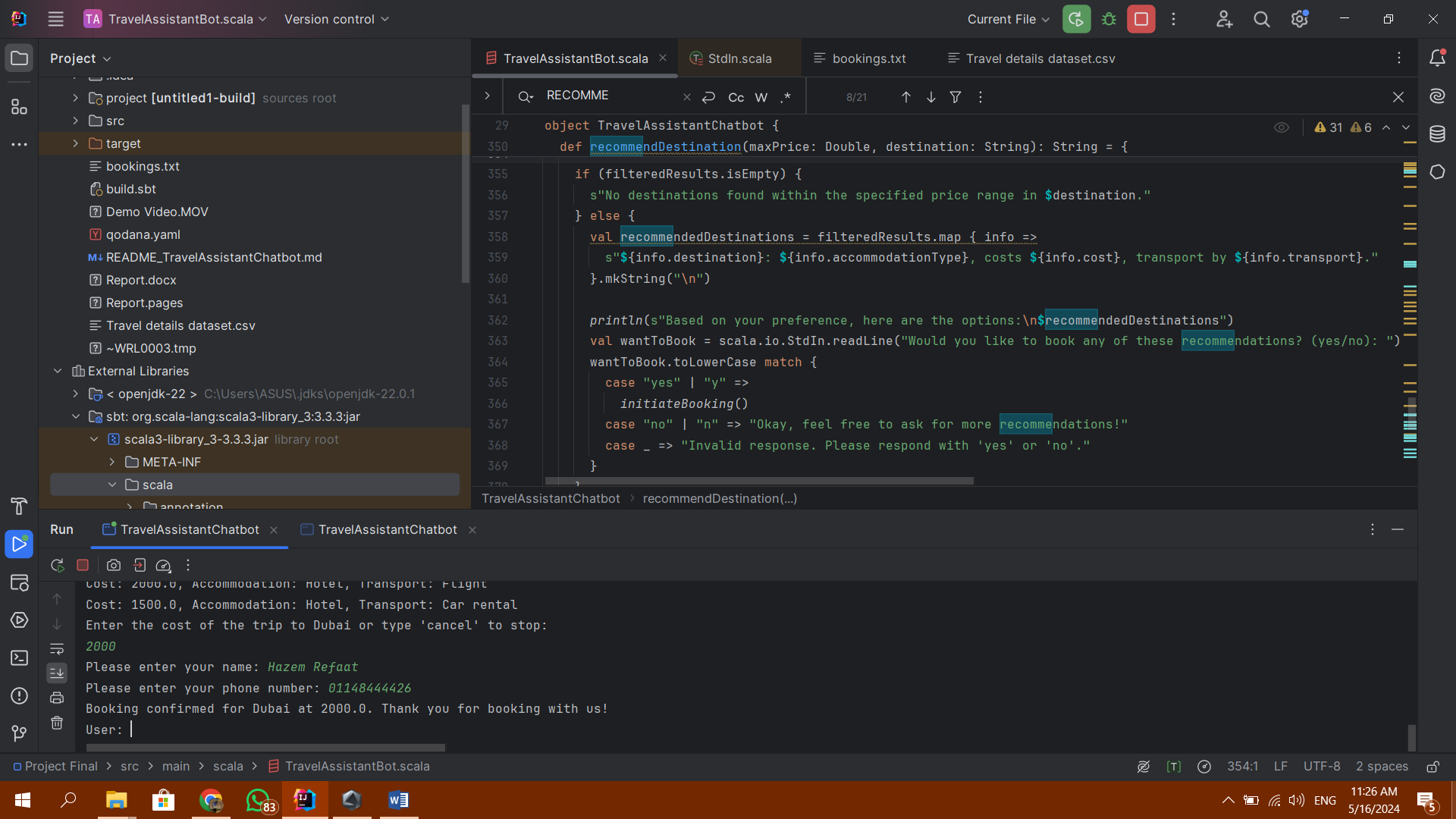
# Test Cases

Greeting The User:

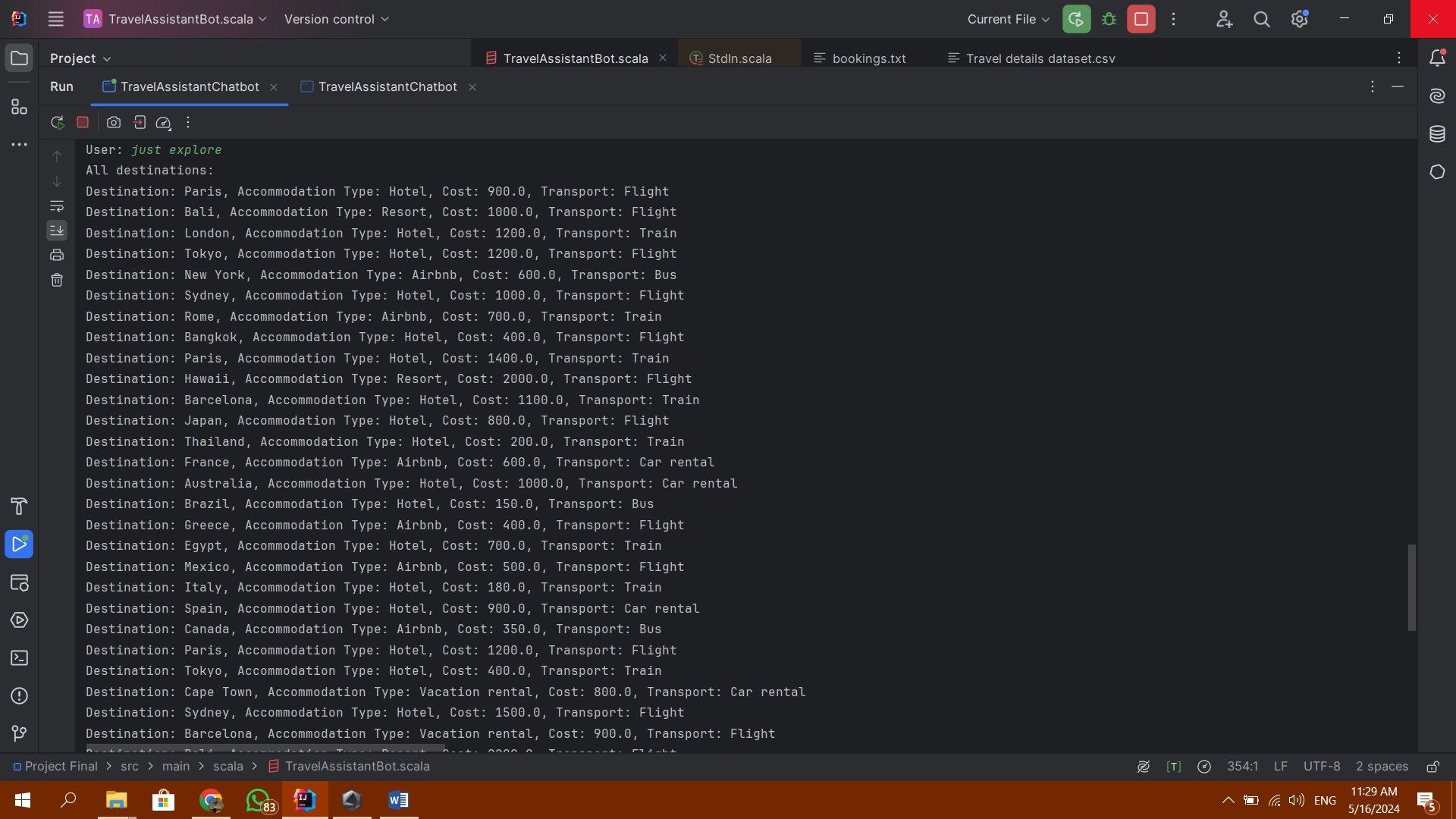
Searching for a certain location:

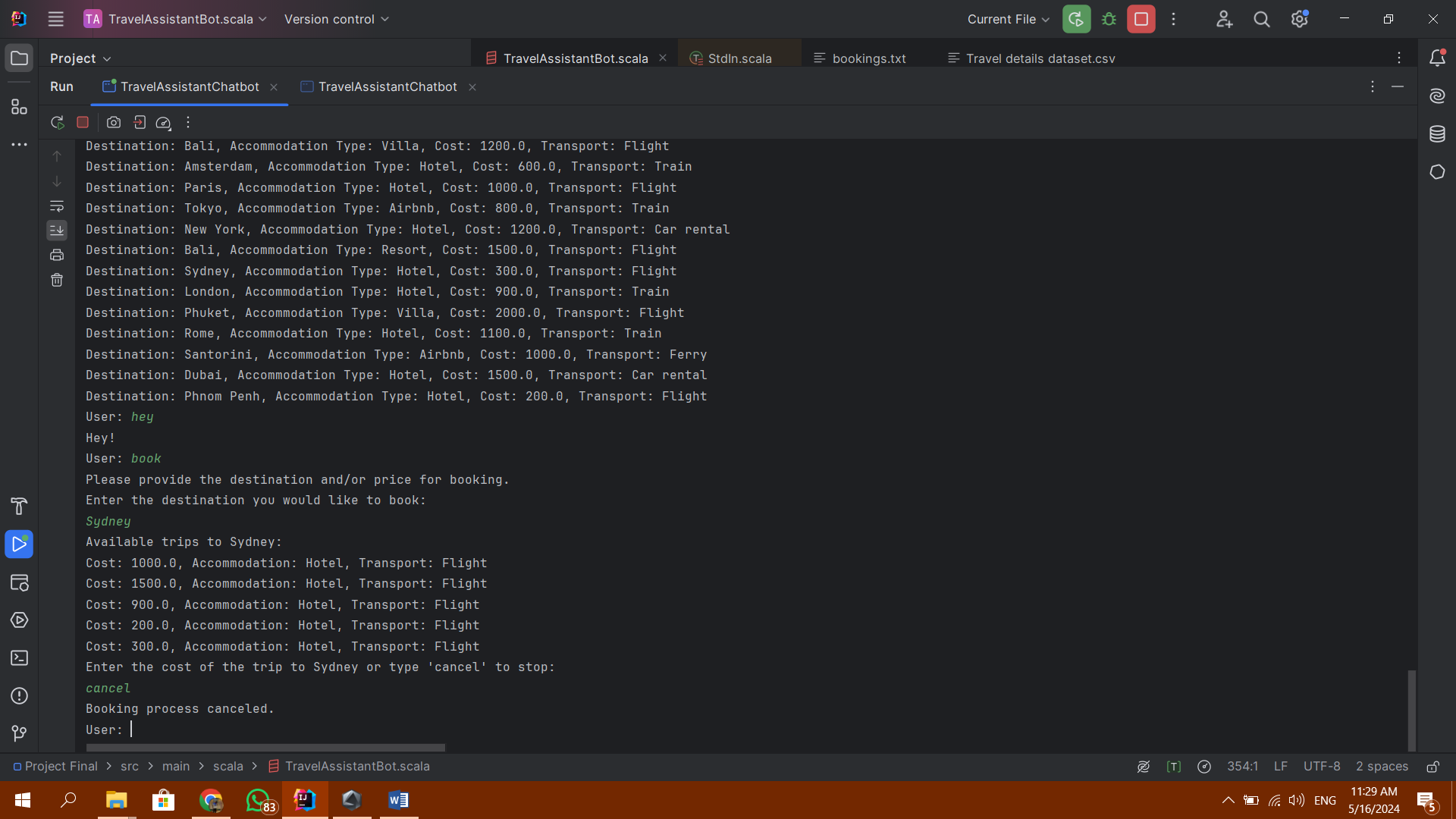
Searching for multiple locations:

Booking with Invalid cost: 

Correcting the Cost: 

Exploring Available Trips:



You can cancel the booking anytime:

# Challenges and problems

We faced multiple challenges at the beginning such that the code didn’t output the desired results and wasn’t fulfilling all the test cases and didn’t get the right money balance desired filter nor the city to travel to filter, also collecting all functions into one whole single code was a challenge to us but we did them all

# Conclusion and Discussion

After many trials we successfully achieved the code aims and missions and tested it on many test cases to make sure fulfilling nearly all popular cases that can be used

# suggestions of improvements/ future work.

We can improve the code by making it filter results faster than we do now and also make it closer to whole generated AI chatbot that can understand nearly most cases and sentences given by human language using some keywords in the input sentence

# List of references

* https://www.kaggle.com/
* https://www.geeksforgeeks.org/scala-programming-language/
* <https://chatgpt.com/>
* https://www.w3schools.com/sql/