Student name:	Student 1: Aline Andrade Costa Student 2: Cynthia da Silva Roque Student 3: Sergio Alves da Silva				
Student number:	Student 1: 3144929 Student 2: 3105781 Student 3: 3139115				
Faculty:	Computing Science				
Course:	BSCH/BSCO/EXCH		Stage/year:	2	
Subject:	Software Development 2				
Study Mode:	Full time	☑	Part-time		
Lecturer Name:	Haseeb Younis/ Muhammad Shoaib				
Assignment Title:	Project Final Documentation				
Date due:	27/04/2025				
Date submitted:					

Plagiarism disclaimer:

I understand that plagiarism is a serious offence and have read and understood the college policy on plagiarism. I also understand that I may receive a mark of zero if I have not identified and properly attributed sources which have been used, referred to, or have in any way influenced the preparation of this assignment, or if I have knowingly allowed others to plagiarise my work in this way.

I hereby certify that this assignment is my own work, based on my personal study and/or research, and that I have acknowledged all material and sources used in its preparation. I also certify that the assignment has not previously been submitted for assessment and that I have not copied in part or whole or otherwise plagiarised the work of anyone else, including other students.

Signed: Aline, Cynthia, Sergio **Date: 21/03/2025**

GRIFFITH COLLEGE DUBLIN

Software Development 2
BSCH-SD2
Chatbot Project
Date 27/04

Table of Contents

1. **Versioning Approach**

In this project, Git is utilized for version control, which enables seamless tracking of code changes, collaboration among team members, and maintaining a history of all modifications.

The repository contains all source code, documentation, and testing scripts.

Branching Strategy:

Main/Master Branch: This branch consistently contains stable code.

Feature Branches: Distinct branches are established for introducing new features or bug fixes.

Commit Messages: Messages are clear and adhere to a standard format (e.g., “[Feature] Implemented basic UI functionality").

Tagging: Significant milestones and release points will be marked with tags for future reference.

2. **Development Process**

This project follows an Agile iterative development process:

→ Milestones:

Each milestone is treated as a complete cycle of development, including coding, testing, documentation, and code reviews.

For instance, the first milestone focuses on integrating basic weather fetching and clothing suggestion functionality, while future milestones might focus on improving the user interface and adding more advanced features.

→ Code Reviews:

Peer reviews are integral to ensuring high code quality. Every new feature or bug fix undergoes a thorough code review by at least one other team member before it is merged into the main branch. This process ensures adherence to coding standards and avoids potential bugs in the production code.

→ Integration:

Continuous integration (CI) is performed via regular commits and merges. Every developer commits their changes to feature branches frequently, and these branches are merged into the main branch after successful testing. This reduces integration issues and keeps the project up-to-date.

→ Communication:

Team members are kept informed of the project's progress through commit logs and collaborative tools (e.g., Slack, Trello). Each commit and pull request is documented in detail, which ensures that everyone is aware of what changes are being made and the progress toward each milestone.

3. UI Implementation

In the current version of the project, the user interface (UI) is simple and text-based, as seen in the Chatbot class. The system prompts the user to input the number of locations and then asks for each location. Based on the entered location, it fetches weather data and provides clothing suggestions.

→ User Interaction:

Users input location names through the console.

The program fetches the weather information and suggests appropriate clothing based on the weather conditions (e.g., "rainy", "snow", "clear").

→ Upcoming Improvements:

Future milestones will introduce an enhanced Graphical User Interface (GUI), which will provide a more intuitive and visually appealing experience, such as a web-based interface or a desktop GUI with weather data and clothing suggestions in a more interactive format.

4. Rest API

4.1 Rest API Implementation

The Weather API fetches real-time weather data using a third-party service (Open Meteo). This external API provides current weather conditions based on the latitude and longitude of a given location.

→ Request Structure:

The program constructs a URL using the location's latitude and longitude, sends a GET request to the API, and retrieves the weather data in JSON format.

→ External Packages:

The code utilises HttpURLConnection to make the API request, Scanner to read the response, and JSONObject from the org.json library to parse the response.

5. Weather API

The Weather API used in this project is from Open Meteo, a free weather service that provides real-time weather forecasts. It allows access to current weather data based on a location's geographic coordinates (latitude and longitude).

→ In the context of this project, the Weather API fetches:

Temperature (in Celsius)

Weather descriptions, such as "clear" or "windy," are used to determine clothing recommendations.

The Weather API is integrated into the project via the WeatherAPI class, which builds the API request, sends it, and parses the response to extract the necessary weather details.

→ **Usage in the Project**

The Weather API is called by the WeatherAPI.getWeather(location) method, which fetches the current temperature and weather description for a location.

The weather information is then used by the chatbot to provide users with clothing suggestions based on the weather (e.g., "Wear a windbreaker" for windy weather or "Carry an umbrella" for rainy conditions)

6. External Packages

JSON (org.json)

→ Purpose: Used to parse the JSON data returned from the Weather API.
→ Example: It helps extract weather details like temperature and wind speed from the API response.
→ Usage: The JSONObject class is used to read and get data from the JSON response.

JUnit (JUnit 5)

→ Purpose: Used for unit testing the code.
→ Example: It helps ensure that methods like getWeather() and getClothingSuggestion() are working correctly.
→ Usage: The @Test annotation is used to mark methods as test cases and assert methods verify the expected results.

HTTPURLConnection (Java Standard Library)

→ Purpose: Used to make HTTP requests to the Weather API.

→ Example: It connects to the API and retrieves the weather data.

→ Usage: It's part of Java's standard library, so no external dependencies are needed.

These packages help the project by allowing for data parsing, testing, and making API requests, ensuring the functionality of the weather data fetching and clothing suggestions.

7. Project Setup

The project setup is straightforward:

→ Java Environment:

The project is developed in Java, so the user must have a Java Development Kit (JDK) installed on their machine to compile and run the program.

→ Dependencies:

External dependencies like org.json for JSON parsing are used. If you're using Maven or Gradle, they can be added to the pom.xml or build.gradle files, respectively.

→ Running the Application:

The application is run from the main() method in the Chatbot class. Upon execution, the program interacts with the user to get locations, fetch weather data, and provide clothing suggestions.

→ Classes:

- Chatbot Class: Acts as the user interface. It collects the locations from the user, interacts with the WeatherAPI to fetch weather data, and then calls ClothingRecommender to get clothing suggestions.
- WeatherAPI Class: This is where the weather data is fetched from an external API. The Chatbot class calls WeatherAPI to get the current weather for a given location.
- ClothingRecommender Class: After getting the weather data, the Chatbot class passes the weather information to ClothingRecommender to determine what clothing to recommend based on the weather conditions.
- ClothingRecommenderTest & WeatherAPITest: These test classes validate the functionality of ClothingRecommender and WeatherAPI to ensure they return correct results.
- ChatbotTest Class: This test class validates the overall user interaction and ensures the Chatbot class integrates properly with WeatherAPI and ClothingRecommender to provide the desired output.

Chatbot

We are creating a chatbot with responses relevant to use users' questions related to planning clothing requirements for a trip that will visit 5 locations in 3 days.

Purpose

Create a software environment to make users' life experiences easier with a flow interaction when the customer decides to travel somewhere and has suggestions about what to wear through the weather information given as well by the chatbot.

General information about the bot.

1.

Name: Wardrobot.

Type: Rule-Based Chatbots (Personal Bot).

Environment: App.

Software: Java and embedded software.

Audience to attend: kids and adults aged.

Personality: Friendly and humoristic to keep flow and relaxed communication with the users.

Navigation

The conversation will flow well-structured, directing the user with steps that should be taken to achieve the results expected by customers.

Cycle of communication

- Introduction about the bot.
- Greet the user by their name.
- Give clear directions on how to interact with the bot.
- Recognize any wrong pattern of the language from the user inputs. (e.g. Images).
- Handle with any error.
- Redirect to regular pattern conversation.
- Give some data to make decisions easily and keep the conversation on the right track.
- The user achieves the results expected.
- Finish software or start a new chat.

Possible interactions scenarios between user and chatbot

Case 1 (Simple)

User: Hi

Bot: Hi there! I am Wardrobot, your weather bot. Would you like the current weather or a forecast?

User: Current weather

Bot: Sure! Which city?

User: Greece

Bot: Got it! The current weather in Greece is **27°C with clear skies**.

User: Thanks.

Bot: No problem! If you're planning a trip, I can also help you check the weather for multiple locations and suggest suitable clothing for each day. Let me know your destinations!

Case 2 (Complex)

User: Hi

Bot: Hi there! I am Wardrobot, your weather bot. What is your name?

User: My name is Fabio.

Bot: Nice to meet you, Fabio! Would you like the current weather or a forecast?

User: Current weather

Bot: Sure! Which city?

User: Greece

Bot: Got it! The current weather in **Greece** is **27°C with clear skies**.

User: Thanks.

Bot: No problem! Are you planning a trip? I can check the weather for multiple locations and suggest what to pack for each day. Let me know your destinations!

User: Yes, I'll be visiting five cities over three days.

Bot: That sounds exciting! Share the city names and travel dates, and I'll help you plan your outfits based on the forecast.

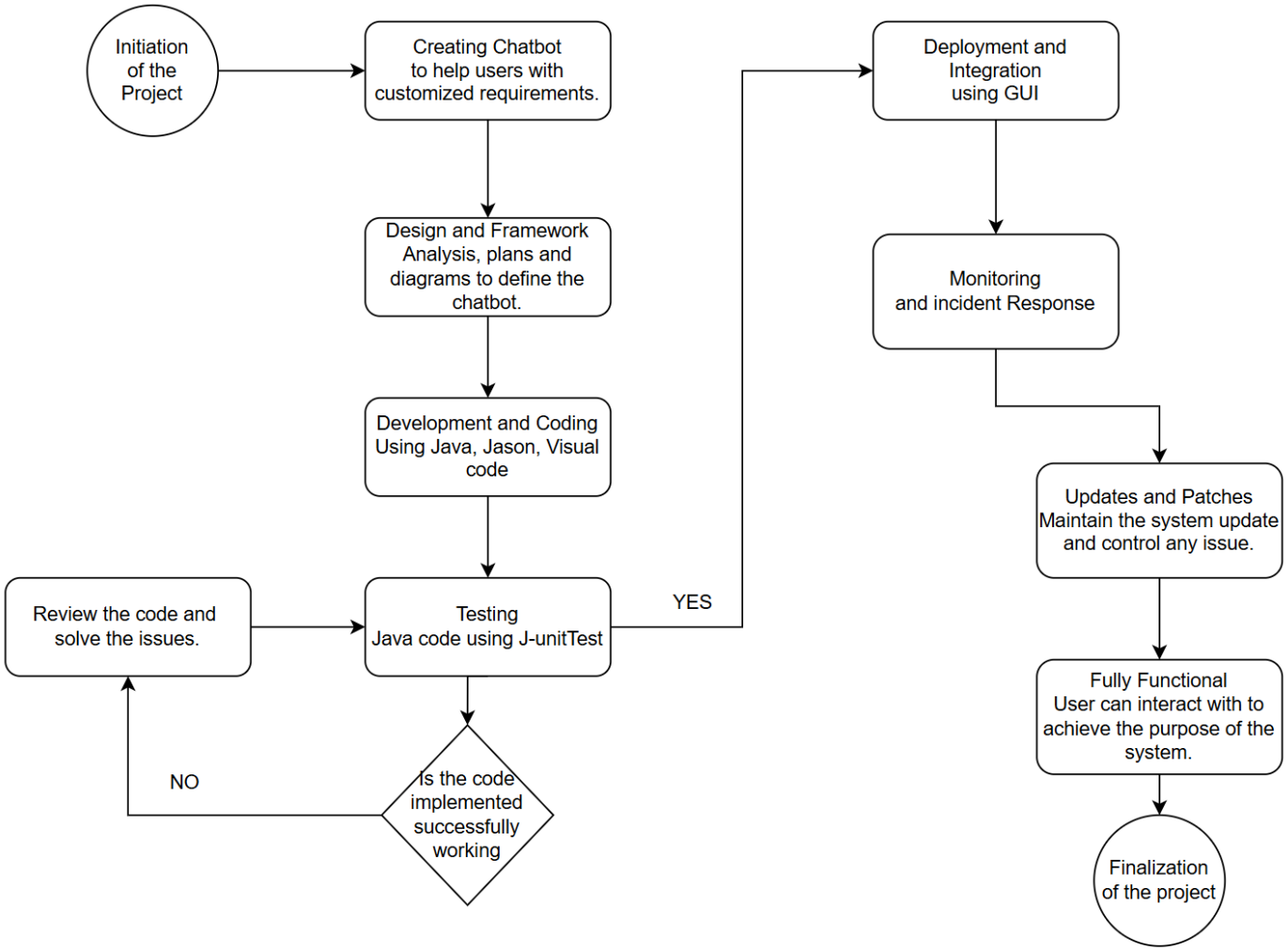
User: Athens, Corinto, Cavala, Pireu, Corfu. 12/10/2025 to 15/10/2025.

Bot: The weather will be amazing around **31°C with clear skies** in all these places. I suggest you wear light clothes and sunglasses.

User: Thanks!

Bot: I'm glad to help you! If you have any questions about other locations, please just ask. Have an amazing trip and don't drink too much!

FlowChart Diagram (Cycle of the Project)



Written Use Case

Use Case: Plan clothing for a trip based on weather forecasts.

Use case name: Plan Clothing for a Trip.

Actor: User.

System: Chatbot System.

Description: The user interacts with the chatbot system to receive weather information and clothing recommendations for a trip covering 5 locations over 3 days.

Preconditions:

- The chatbot system must be installed in the smartphone for the user to have access to it.
- The chatbot system must have access to a weather API.

Trigger: The user initiates the chatbot session in the app pre-installed on the smartphone.

Main Flow:

1. The user starts the chatbot.
2. Chatbot prompts the user to enter trip details.
3. The user enters trip details (5 locations over 3 days).

4. Chatbot fetches weather forecasts for the specified locations and dates.
5. Chatbot analyses weather conditions.
6. Chatbot provides clothing recommendations based on the forecast for each location.
7. The user reviews the recommendations.
8. The user may modify trip details.
9. Chatbot updates and provides new recommendations.
10. The user ends the sessions after receiving recommendations.

Alternative Flows:

1. **User enters invalid trip details.**

- If the user enters incomplete or incorrect details, the chatbot prompts them to re-enter valid information.

2. **The Weather API is unavailable.**

- If the chatbot fails to retrieve weather data, it notifies the user and suggests trying again later.

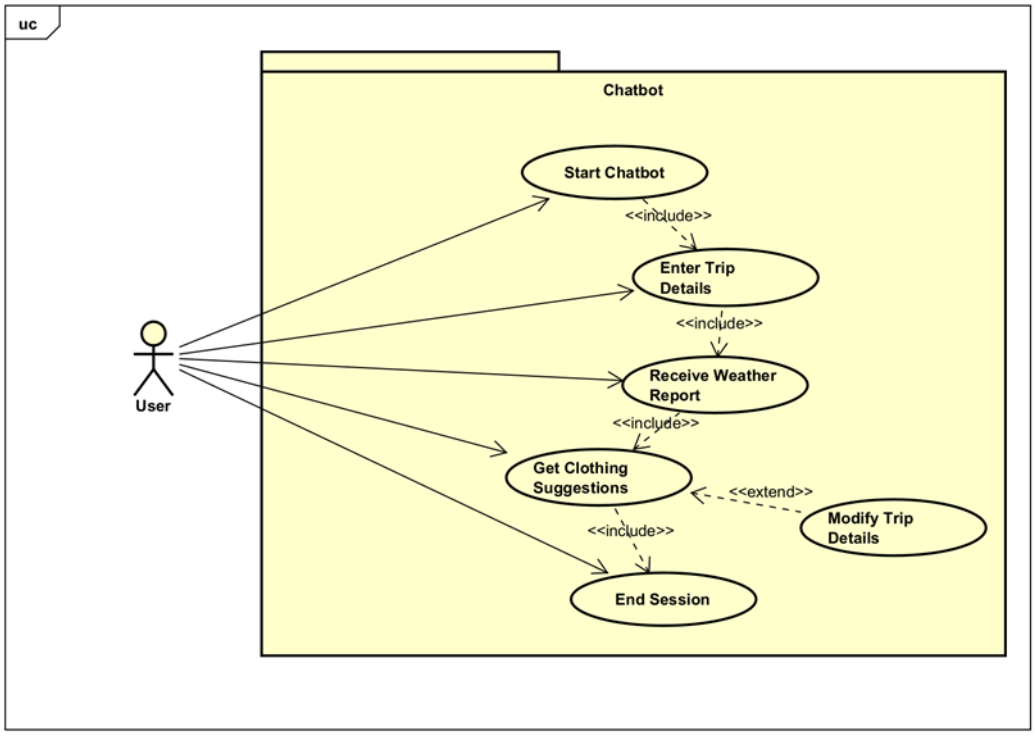
3. **User exits before completing the process.**

- If the user decides to stop before receiving recommendations, the chatbot confirms and ends the session.

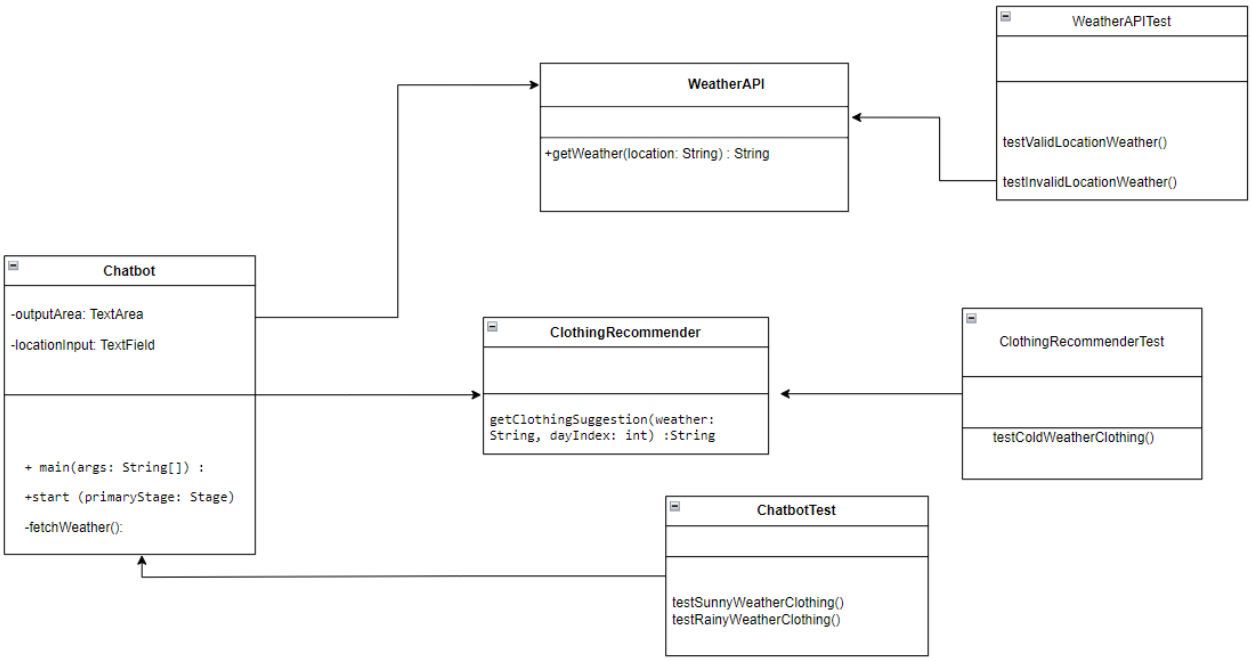
Postconditions:

- The user successfully receives weather-based clothing suggestions for the trip.
- The chatbot session ends.

Use Case Diagram



Class Diagram



8. Milestone 1

8.1 Goals

Implement basic weather-fetching functionality via the Weather API.

Provide basic clothing suggestions based on weather conditions.

Set up the project structure, including necessary classes and initial functionality.

→ **The first milestone includes:**

Fetching weather data via an API request.

Displaying basic clothing suggestions based on conditions like rain, snow, clear, and temperature.

Writing the initial JUnit tests to verify the functionality of the getClothingSuggestion() method.

8.2 Junit Tests

Unit tests are provided in the ChatbotTest class to verify that the clothing suggestions work correctly for different weather conditions.

→ Tests:

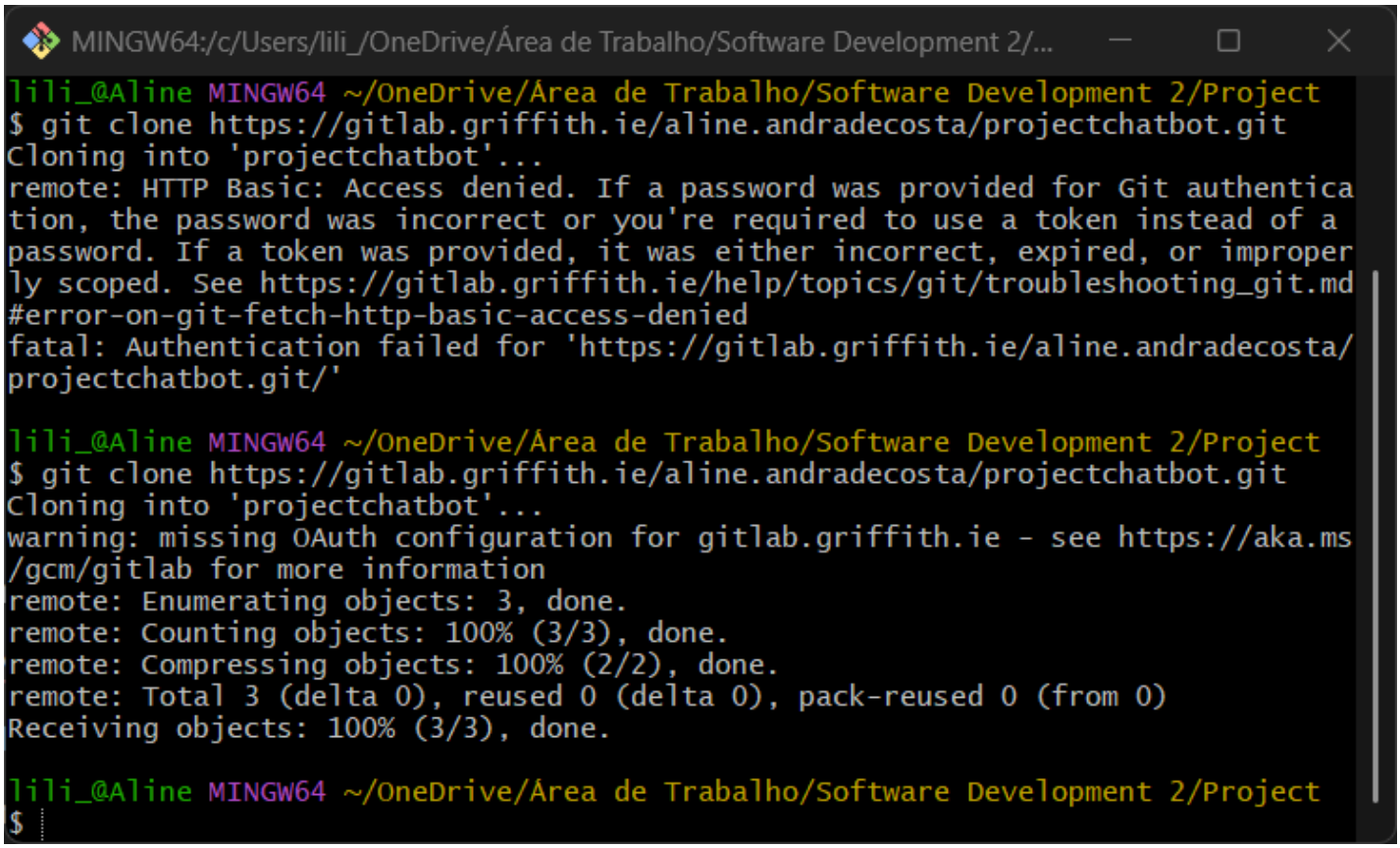
For example, the test assertEquals("Carry an umbrella and wear waterproof clothing.", Chatbot.getClothingSuggestion("Temperature: 15°C, rain")) ensures that the system suggests wearing waterproof clothing when rain is mentioned in the weather info.

These tests ensure the application works as expected and helps prevent bugs as new features are added.

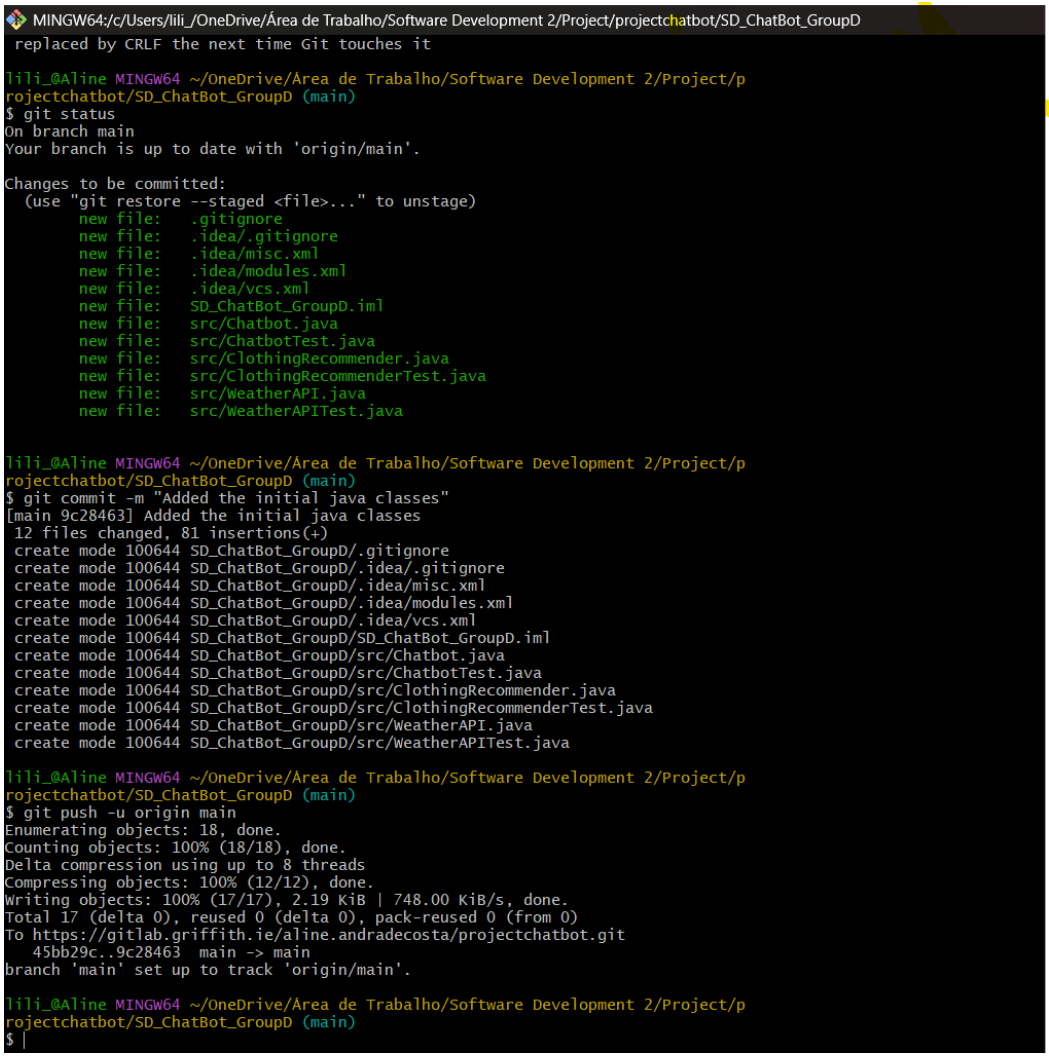
8.3 Commit Logs

Aline – Commit logs

Cloning the repository:



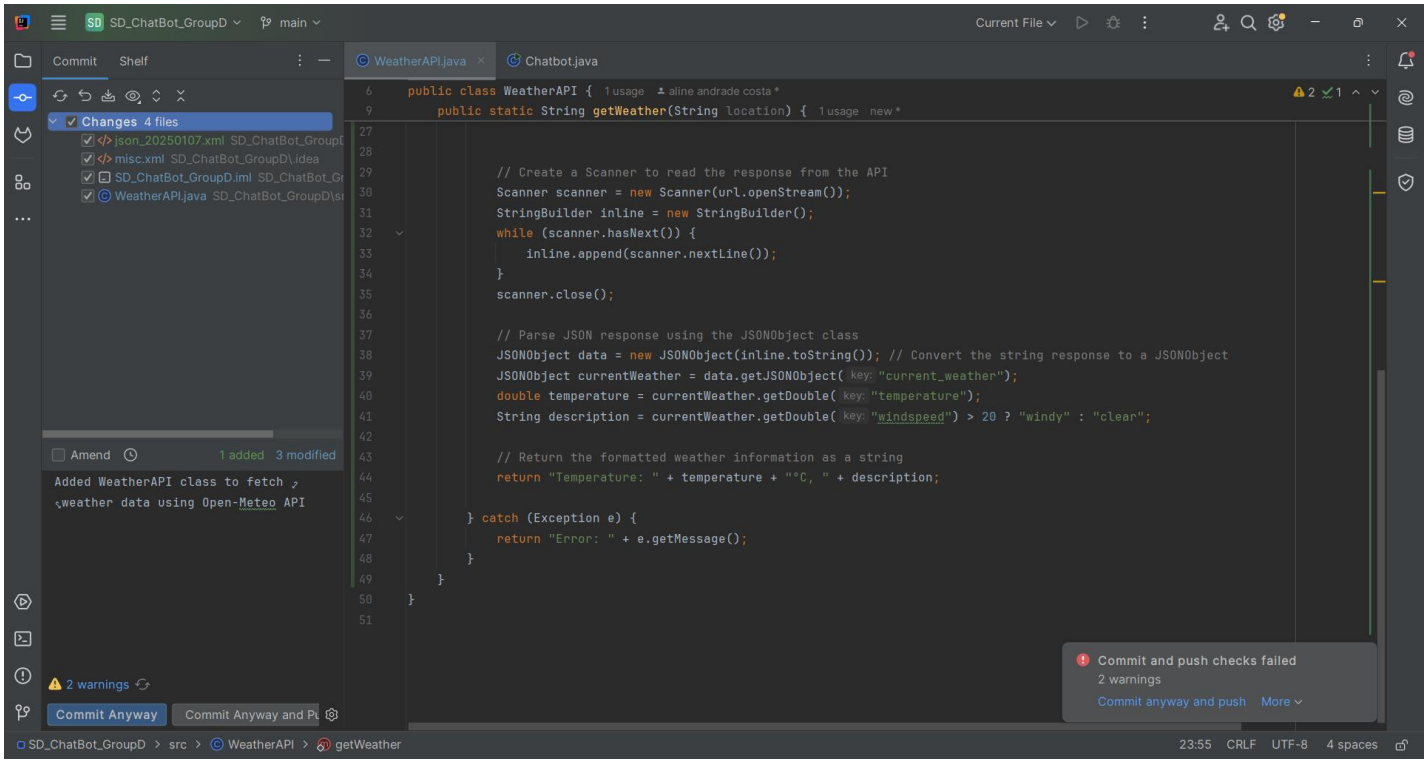
Adding and committing the initial Java file.

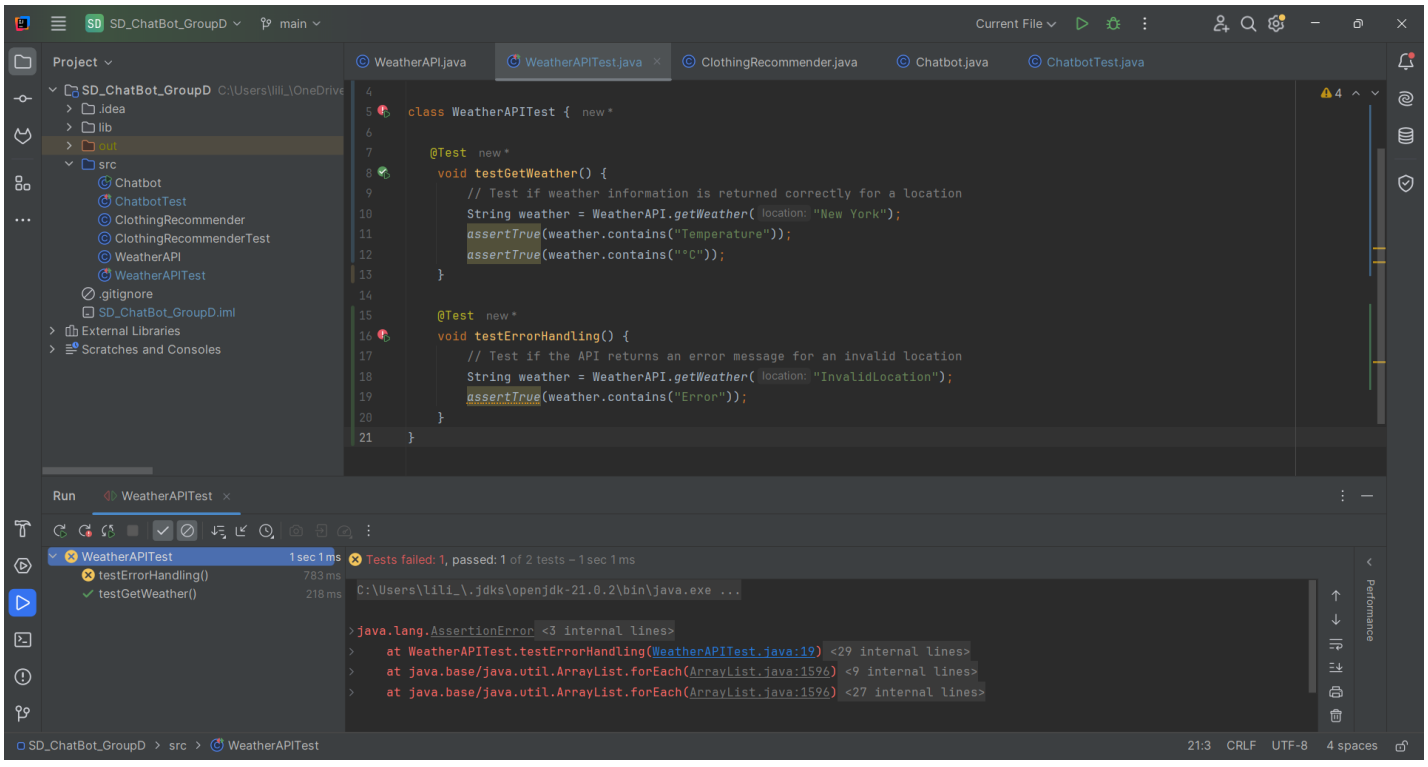

MINGW64:/c/Users/lili_/OneDrive/Área de Trabalho/Software Development 2/Project/projectchatbot/SD_ChatBot_GroupD
replaced by CRLF the next time Git touches it
lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/p
rojectchatbot/SD_ChatBot_GroupD (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
new file: .gitignore
new file: .idea/.gitignore
new file: .idea/misc.xml
new file: .idea/modules.xml
new file: .idea/vcs.xml
new file: SD_ChatBot_GroupD.iml
new file: src/Chatbot.java
new file: src/ChatbotTest.java
new file: src/ClothingRecommender.java
new file: src/ClothingRecommenderTest.java
new file: src/WeatherAPI.java
new file: src/WeatherAPITest.java
lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/p
rojectchatbot/SD_ChatBot_GroupD (main)
$ git commit -m "Added the initial java classes"
[main 9c28463] Added the initial java classes
12 files changed, 81 insertions(+)
create mode 100644 SD_ChatBot_GroupD/.gitignore
create mode 100644 SD_ChatBot_GroupD/.idea/.gitignore
create mode 100644 SD_ChatBot_GroupD/.idea/misc.xml
create mode 100644 SD_ChatBot_GroupD/.idea/modules.xml
create mode 100644 SD_ChatBot_GroupD/.idea/vcs.xml
create mode 100644 SD_ChatBot_GroupD/SD_ChatBot_GroupD.iml
create mode 100644 SD_ChatBot_GroupD/src/Chatbot.java
create mode 100644 SD_ChatBot_GroupD/src/ChatbotTest.java
create mode 100644 SD_ChatBot_GroupD/src/ClothingRecommender.java
create mode 100644 SD_ChatBot_GroupD/src/ClothingRecommenderTest.java
create mode 100644 SD_ChatBot_GroupD/src/WeatherAPI.java
create mode 100644 SD_ChatBot_GroupD/src/WeatherAPITest.java
lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/p
rojectchatbot/SD_ChatBot_GroupD (main)
$ git push -u origin main
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 8 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (17/17), 2.19 KiB | 748.00 KiB/s, done.
Total 17 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://gitlab.griffith.ie/aline.andradecosta/projectchatbot.git
45bb29c..9c28463 main -> main
branch 'main' set up to track 'origin/main'.
lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/p
rojectchatbot/SD_ChatBot_GroupD (main)
$

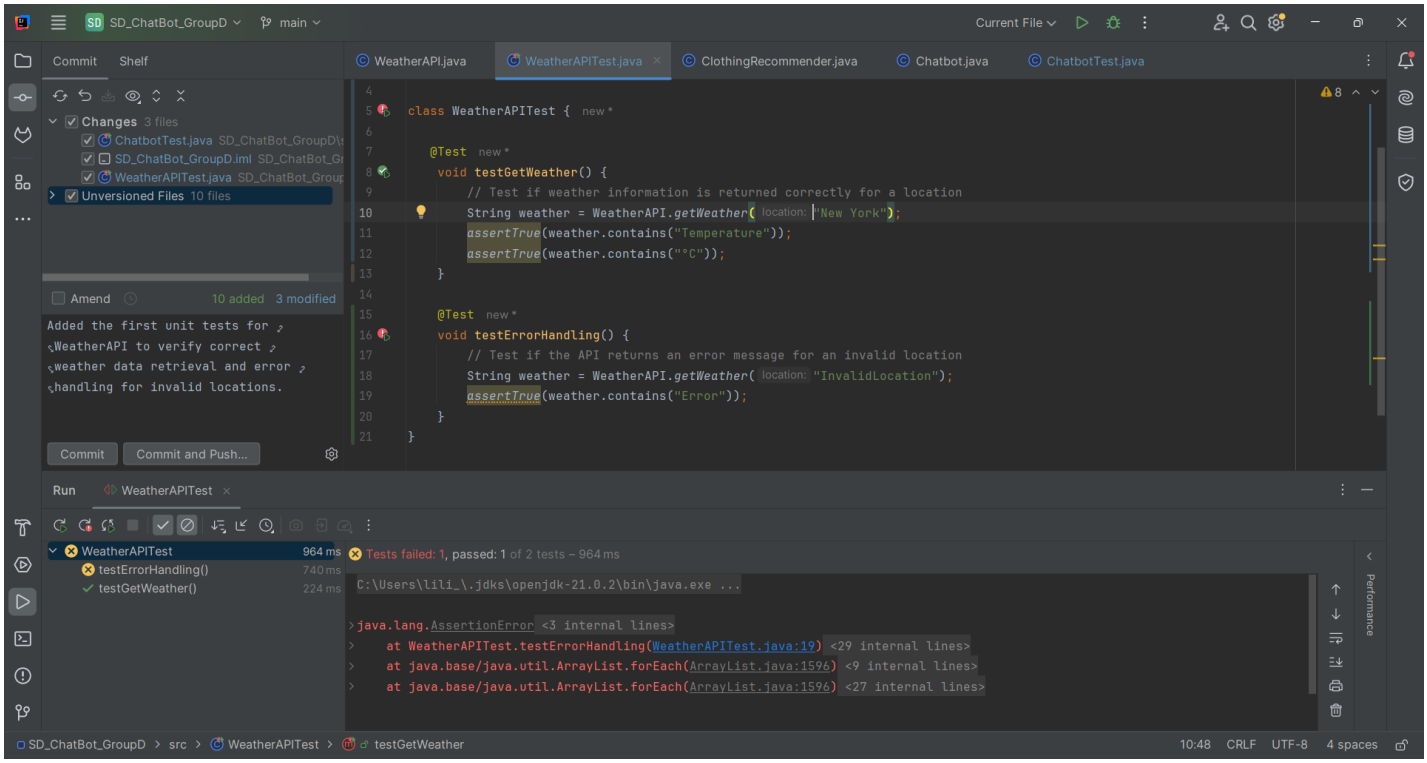
```
lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/projectchatbot/Documents (main)
$ git add .

lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/projectchatbot/Documents (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   SD_ProjectChatBot-GroupD.docx


lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/projectchatbot/Documents (main)
$ git commit -m "Added the folder and initial file to feed with all documentation and diagrams"
[main 2751e9e] Added the folder and initial file to feed with all documentation and diagrams
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Documents/SD_ProjectChatBot-GroupD.docx

lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/projectchatbot/Documents (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 237.18 KiB | 18.24 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://gitlab.griffith.ie/aline.andradecosta/projectchatbot.git
   9c28463..2751e9e  main -> main
branch 'main' set up to track 'origin/main'.

lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/projectchatbot/Documents (main)
$
```

Git pull first implementation

```
MINGW64:/c/Users/lili_/OneDrive/Área de Trabalho/Software Development 2/...

lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/p
rojectchatbot/SD_ChatBot_GroupD/src (main)
$ git pull
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 23 (delta 17), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (23/23), 2.21 KiB | 24.00 KiB/s, done.
From https://gitlab.griffith.ie/aline.andradecosta/projectchatbot
   2751e9e..9b44351  main       -> origin/main
error: Your local changes to the following files would be overwritten by merge:
        SD_ChatBot_GroupD/.idea/vcs.xml
Please commit your changes or stash them before you merge.
Aborting
Updating 2751e9e..9b44351

lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/p
rojectchatbot/SD_ChatBot_GroupD/src (main)
$
```


SD_ChatBot_GroupD
main
Current File
Commit
Shelf
WeatherAPI.java
Chatbot.java
Changes 4 files
json_20250107.xml SD_ChatBot_GroupD
misc.xml SD_ChatBot_GroupD\.idea
SD_ChatBot_GroupD.iml SD_ChatBot_GroupD
WeatherAPI.java SD_ChatBot_GroupD\src
Amend
1 added 3 modified
Added WeatherAPI class to fetch weather data using Open-Meteo API
2 warnings
Commit Anyway
Commit Anyway and Push
public class WeatherAPI { 1 usage aline andrade costa *
public static String getWeather(String location) { 1 usage new *
// Create a Scanner to read the response from the API
Scanner scanner = new Scanner(url.openStream());
StringBuilder inline = new StringBuilder();
while (scanner.hasNext()) {
inline.append(scanner.nextLine());
}
scanner.close();
// Parse JSON response using the JSONObject class
JSONObject data = new JSONObject(inline.toString()); // Convert the string response to a JSONObject
JSONObject currentWeather = data.getJSONObject(key: "current_weather");
double temperature = currentWeather.getDouble(key: "temperature");
String description = currentWeather.getDouble(key: "windspeed") > 20 ? "Windy" : "clear";
// Return the formatted weather information as a string
return "Temperature: " + temperature + "°C, " + description;
} catch (Exception e) {
return "Error: " + e.getMessage();
}
}
}
Commit and push checks failed
2 warnings
Commit anyway and push
More
SD_ChatBot_GroupD > src > WeatherAPI > getWeather
23:55
CRLF
UTF-8
4 spaces

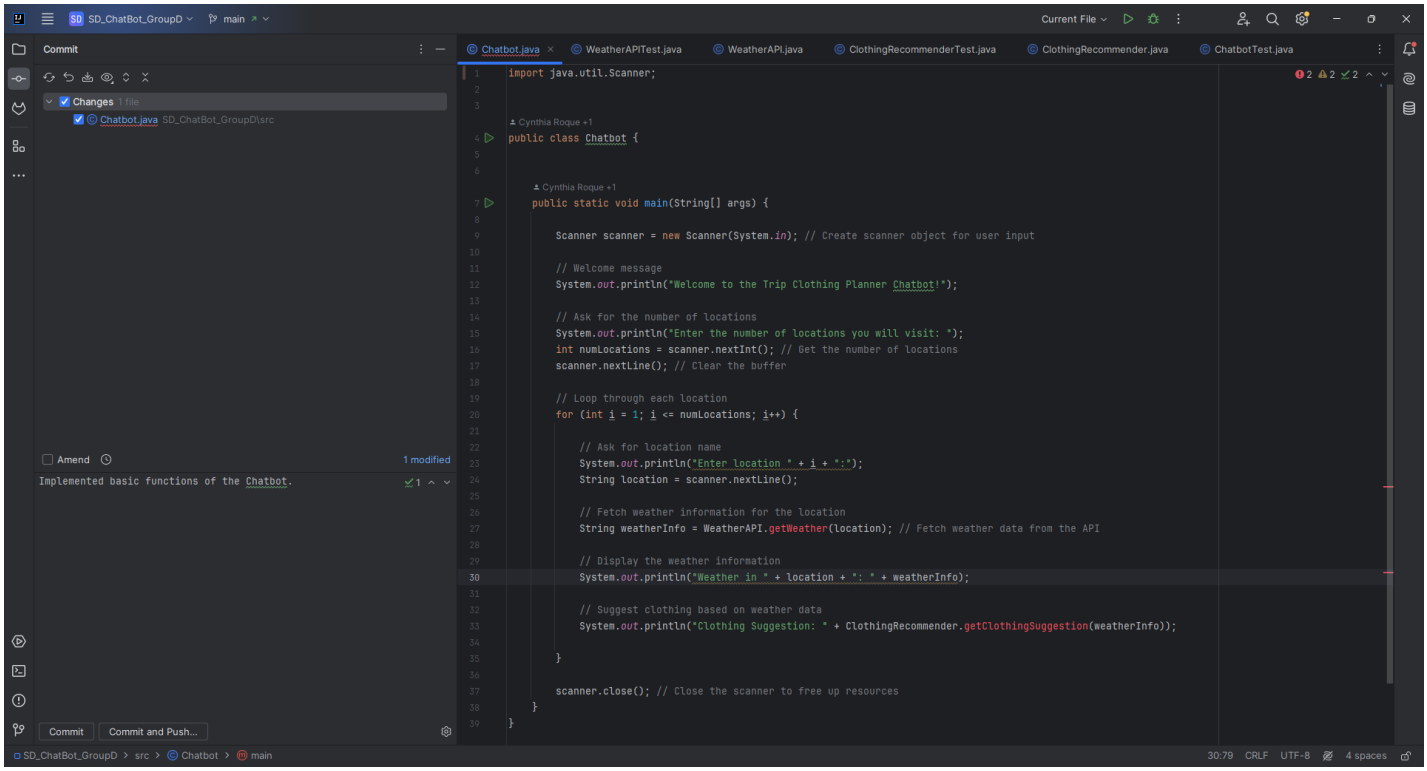

SD_ChatBot_GroupD
main
Current File
Project
SD_ChatBot_GroupD C:\Users\lili_\OneDrive
.idea
lib
out
src
Chatbot
ChatbotTest
ClothingRecommender
ClothingRecommenderTest
WeatherAPI
WeatherAPITest
.gitignore
SD_ChatBot_GroupD.iml
External Libraries
Scratches and Consoles
WeatherAPI.java
WeatherAPITest.java
ClothingRecommender.java
Chatbot.java
ChatbotTest.java
class WeatherAPITest { new *
@Test new *
void testGetWeather() {
// Test if weather information is returned correctly for a location
String weather = WeatherAPI.getWeather(location: "New York");
assertTrue(weather.contains("Temperature"));
assertTrue(weather.contains("°C"));
}
@Test new *
void testErrorHandling() {
// Test if the API returns an error message for an invalid location
String weather = WeatherAPI.getWeather(location: "InvalidLocation");
assertTrue(weather.contains("Error"));
}
}
Run
WeatherAPITest
WeatherAPITest 1 sec 1 ms
testErrorHandling() 783 ms
testGetWeather() 218 ms
Tests failed: 1, passed: 1 of 2 tests – 1 sec 1 ms
C:\Users\lili_\.jdks\openjdk-21.0.2\bin\java.exe ...
java.lang.AssertionError <3 internal lines>
at WeatherAPITest.testErrorHandling(WeatherAPITest.java:19) <29 internal lines>
at java.base/java.util.ArrayList.forEach(ArrayList.java:1596) <9 internal lines>
at java.base/java.util.ArrayList.forEach(ArrayList.java:1596) <27 internal lines>
Performance
SD_ChatBot_GroupD > src > WeatherAPITest
21:3
CRLF
UTF-8
4 spaces

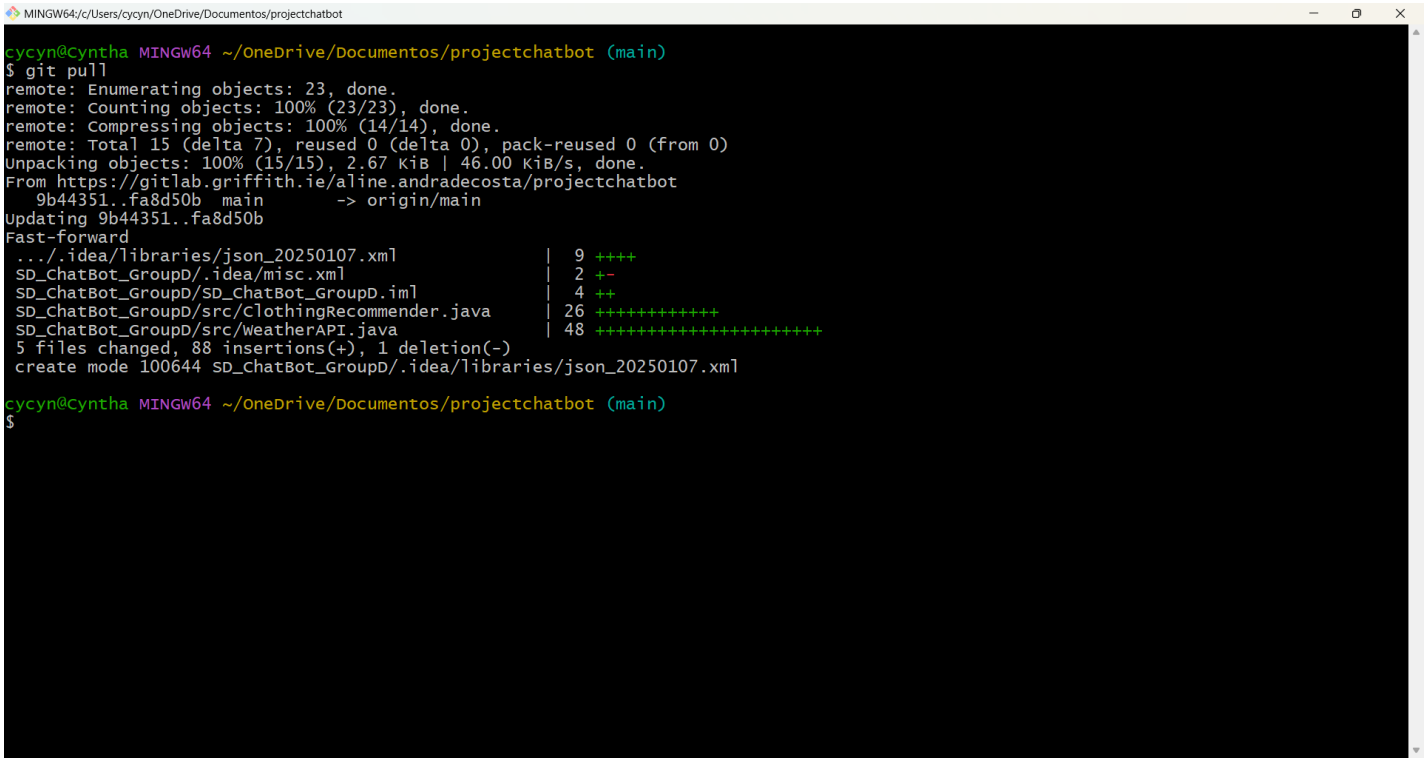


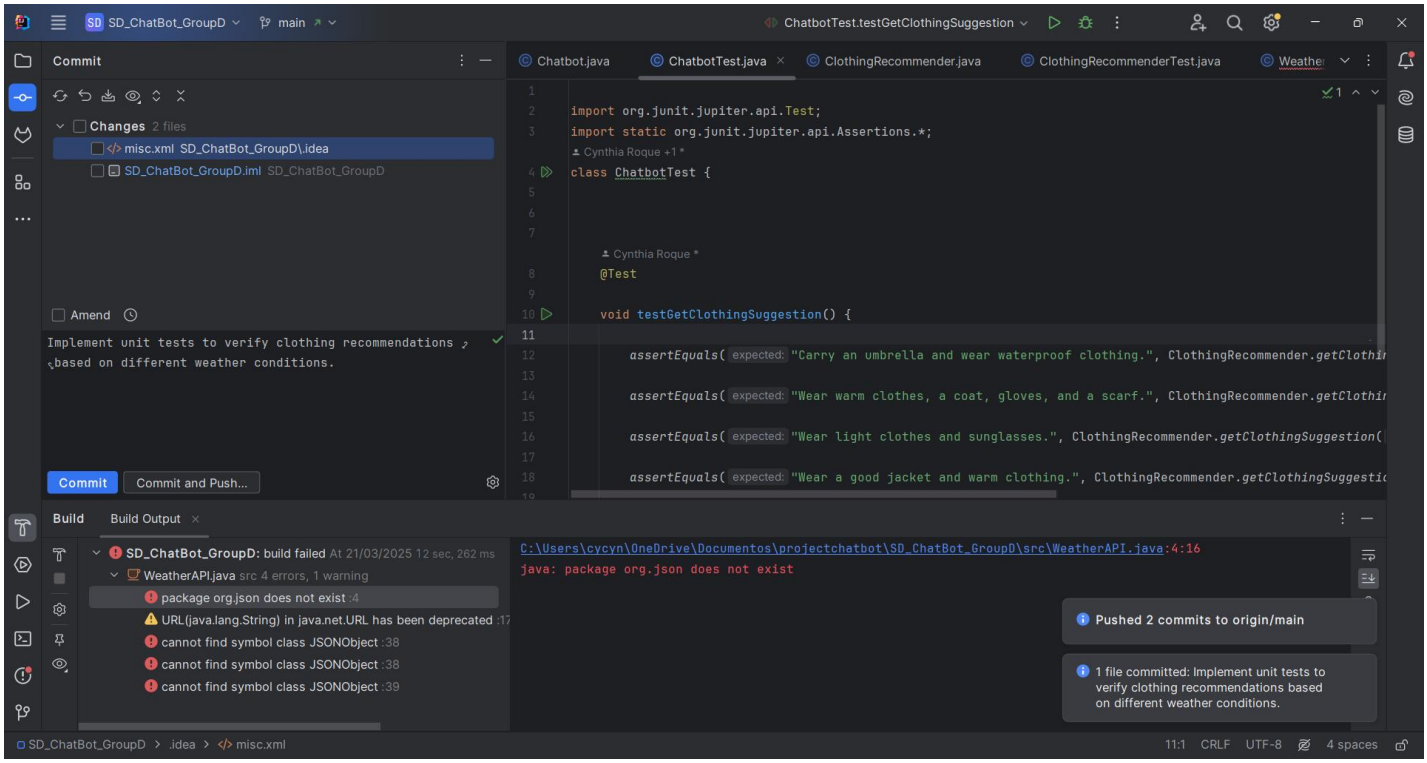
Cynthia – Commit logs

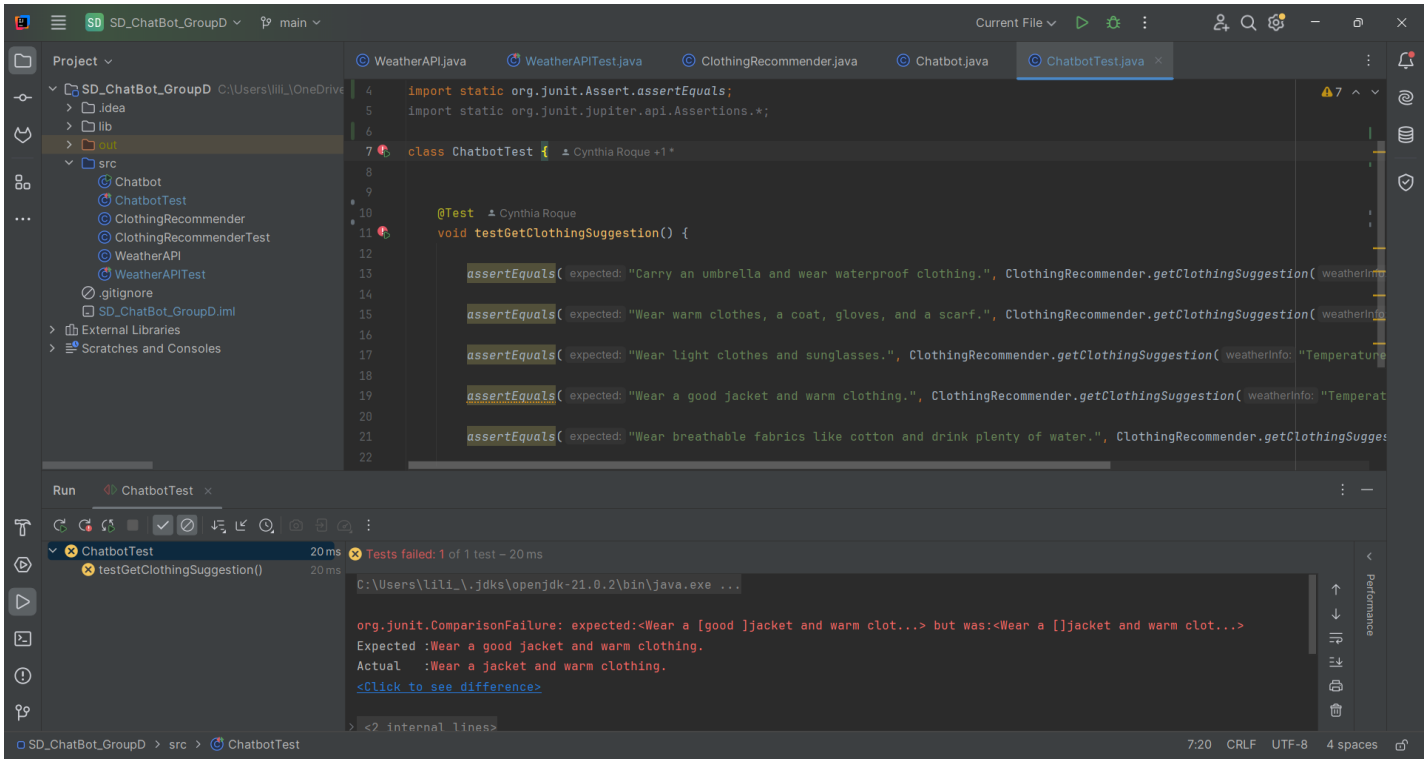
```
cycyn@Cyntha MINGW64 ~/OneDrive/Documentos
$ git clone https://gitlab.griffith.ie/aline.andradecosta/projectchatbot.git
Cloning into 'projectchatbot'...
warning: missing OAuth configuration for gitlab.griffith.ie - see https://aka.ms/gcm/gitlab for more informatio
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 20 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (20/20), 4.97 KiB | 463.00 KiB/s, done.

cycyn@Cyntha MINGW64 ~/OneDrive/Documentos
$
```

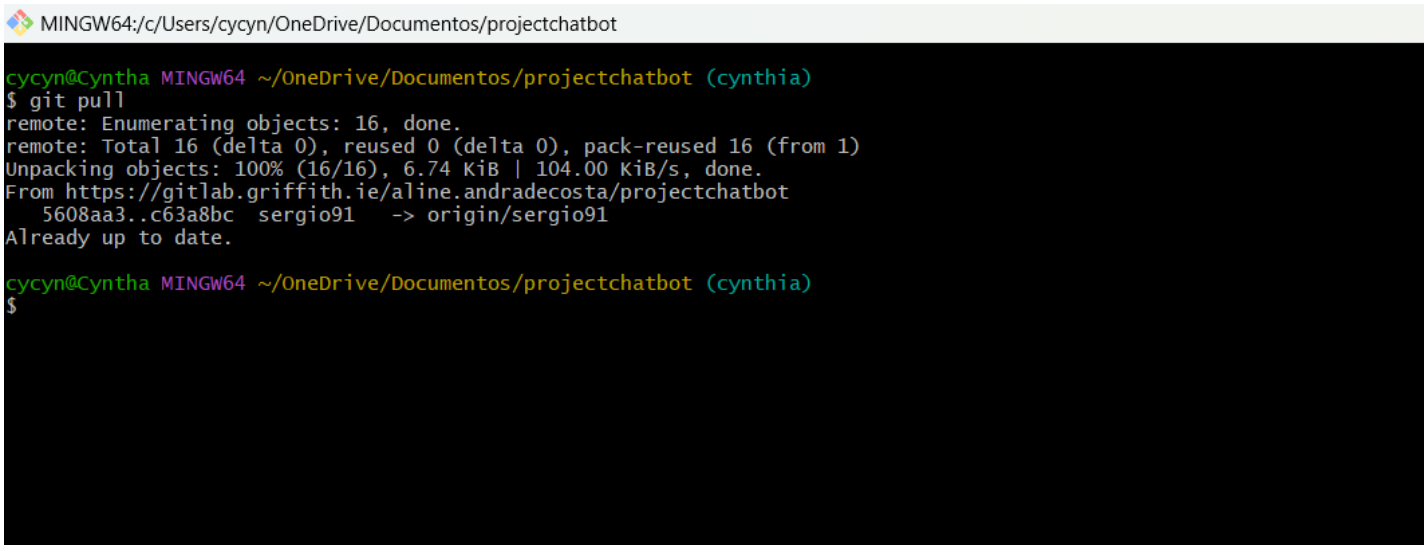
```
cycyn@Cyntha MINGW64 ~/OneDrive/Documentos/projectchatbot (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

cycyn@Cyntha MINGW64 ~/OneDrive/Documentos/projectchatbot (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (4/4), 237.16 KiB | 265.00 KiB/s, done.
From https://gitlab.griffith.ie/aline.andradecosta/projectchatbot
   9c28463..2751e9e  main       -> origin/main
Updating 9c28463..2751e9e
Fast-forward
 Documents/SD_ProjectChatBot-GroupD.docx | Bin 0 -> 246642 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Documents/SD_ProjectChatBot-GroupD.docx

cycyn@Cyntha MINGW64 ~/OneDrive/Documentos/projectchatbot (main)
$
```


SD_ChatBot_GroupD
main
Current File
Commit
Changes 1 file
Chatbot.java SD_ChatBot_GroupD\src
Amend
1 modified
Implemented basic functions of the Chatbot.
Commit
Commit and Push...
Chatbot.java
WeatherAPITest.java
WeatherAPI.java
ClothingRecommenderTest.java
ClothingRecommender.java
ChatbotTest.java
import java.util.Scanner;
Cynthia Roque +1
public class Chatbot {
Cynthia Roque +1
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in); // Create scanner object for user input
// Welcome message
System.out.println("Welcome to the Trip Clothing Planner Chatbot!");
// Ask for the number of locations
System.out.println("Enter the number of locations you will visit: ");
int numLocations = scanner.nextInt(); // Get the number of locations
scanner.nextLine(); // Clear the buffer
// Loop through each location
for (int i = 1; i <= numLocations; i++) {
// Ask for location name
System.out.println("Enter location " + i + ":");
String location = scanner.nextLine();
// Fetch weather information for the location
String weatherInfo = WeatherAPI.getWeather(location); // Fetch weather data from the API
// Display the weather information
System.out.println("Weather in " + location + ": " + weatherInfo);
// Suggest clothing based on weather data
System.out.println("Clothing Suggestion: " + ClothingRecommender.getClothingSuggestion(weatherInfo));
}
scanner.close(); // Close the scanner to free up resources
}
}
SD_ChatBot_GroupD > src > Chatbot > main
30:79 CRLF UTF-8 4 spaces


MINGW64:/c/Users/cycyn/OneDrive/Documentos/projectchatbot
cycyn@Cyntha MINGW64 ~/OneDrive/Documentos/projectchatbot (main)
$ git pull
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 15 (delta 7), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (15/15), 2.67 KiB | 46.00 KiB/s, done.
From https://gitlab.griffith.ie/aline.andradecosta/projectchatbot
9b44351..fa8d50b main -> origin/main
Updating 9b44351..fa8d50b
Fast-forward
.../.idea/libraries/json_20250107.xml | 9 ++++
SD_ChatBot_GroupD/.idea/misc.xml | 2 +-
SD_ChatBot_GroupD/SD_ChatBot_GroupD.iml | 4 ++
SD_ChatBot_GroupD/src/ClothingRecommender.java | 26 ++++++++++++
SD_ChatBot_GroupD/src/WeatherAPI.java | 48 ++++++++++++++++++++++
5 files changed, 88 insertions(+), 1 deletion(-)
create mode 100644 SD_ChatBot_GroupD/.idea/libraries/json_20250107.xml
cycyn@Cyntha MINGW64 ~/OneDrive/Documentos/projectchatbot (main)
$

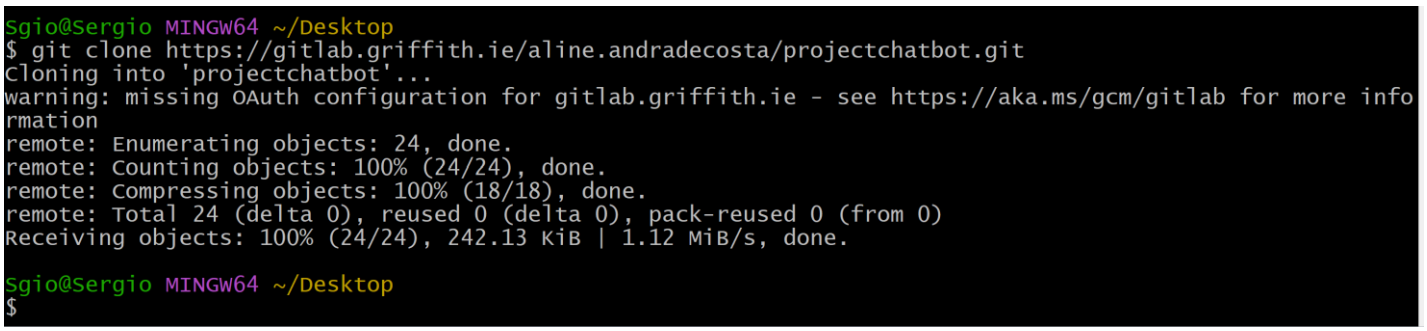

SD_ChatBot_GroupD
main
ChatbotTest.testGetClothingSuggestion
Commit
Changes 2 files
misc.xml SD_ChatBot_GroupD\.idea
SD_ChatBot_GroupD.iml SD_ChatBot_GroupD
Amend
Implement unit tests to verify clothing recommendations based on different weather conditions.
Commit
Commit and Push...
Chatbot.java
ChatbotTest.java
ClothingRecommender.java
ClothingRecommenderTest.java
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
Cynthia Roque +1 *
class ChatbotTest {
Cynthia Roque *
@Test
void testGetClothingSuggestion() {
assertEquals(expected: "Carry an umbrella and wear waterproof clothing.", ClothingRecommender.getClothi
assertEquals(expected: "Wear warm clothes, a coat, gloves, and a scarf.", ClothingRecommender.getClothi
assertEquals(expected: "Wear light clothes and sunglasses.", ClothingRecommender.getClothingSuggestion(
assertEquals(expected: "Wear a good jacket and warm clothing.", ClothingRecommender.getClothingSuggestio
Build
Build Output
SD_ChatBot_GroupD: build failed At 21/03/2025 12 sec, 262 ms
WeatherAPI.java src 4 errors, 1 warning
package org.json does not exist :4
URL(java.lang.String) in java.net.URL has been deprecated :17
cannot find symbol class JSONObject :38
cannot find symbol class JSONObject :38
cannot find symbol class JSONObject :39
C:\Users\cycyn\OneDrive\Documentos\projectchatbot\SD_ChatBot_GroupD\src\WeatherAPI.java:4:16
java: package org.json does not exist
Pushed 2 commits to origin/main
1 file committed: Implement unit tests to verify clothing recommendations based on different weather conditions.
SD_ChatBot_GroupD > .idea > misc.xml
11:1 CRLF UTF-8 4 spaces





Sergio – Commit logs

Cloning the repository:



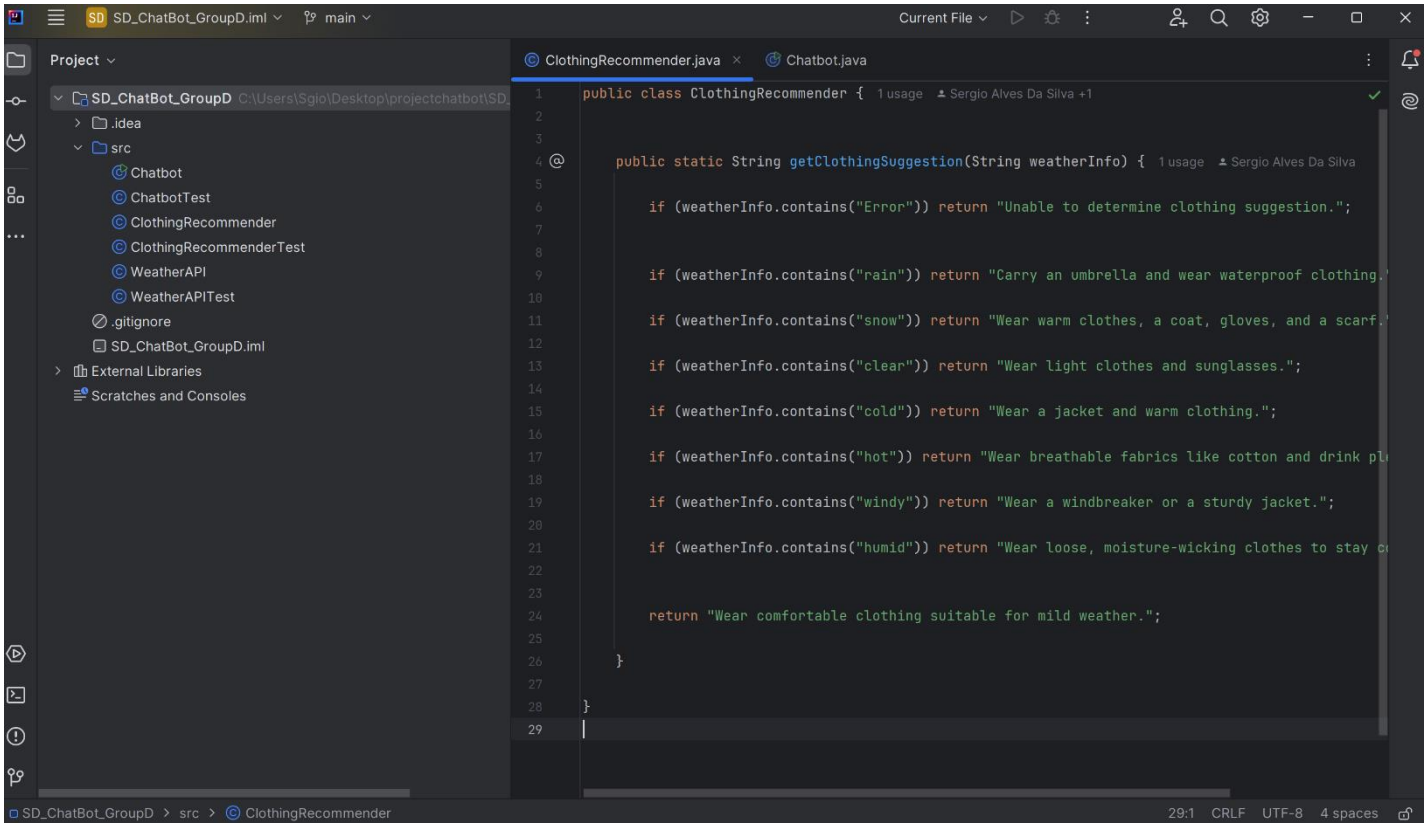
Git Pull for all implementations made in the project:

```
no changes added to commit (use "git add" and/or "git commit -a")

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot (sergio)
$ git checkout main
Switched to branch 'main'
M       SD_ChatBot_GroupD/.idea/vcs.xml
Your branch is up to date with 'origin/main'.

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot (main)
$ git branch -d sergio
Deleted branch sergio (was 2751e9e).

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot (main)
$ git pull
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 23 (delta 17), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (23/23), 2.21 KiB | 26.00 KiB/s, done.
From https://gitlab.griffith.ie/aline.andradecosta/projectchatbot
   2751e9e..9b44351  main       -> origin/main
Updating 2751e9e..9b44351
Created autostash: b1d419a
Fast-forward
 SD_ChatBot_GroupD/.idea/misc.xml       |  2 +-
 SD_ChatBot_GroupD/.idea/vcs.xml        |  4 +++-
 SD_ChatBot_GroupD/src/Chatbot.java     | 35 ++++++++++++++++++++++++++++++++++-
 SD_ChatBot_GroupD/src/ChatbotTest.java |  4 +++-
 4 files changed, 41 insertions(+), 4 deletions(-)
Applied autostash.

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot (main)
$
```

New Commit for alterations in the project:

```
Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD/src (main)
$ git pull
Already up to date.

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD/src (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   ClothingRecommender.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ../.idea/.name

no changes added to commit (use "git add" and/or "git commit -a")

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD/src (main)
$ git add .

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD/src (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   ClothingRecommender.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ../.idea/.name


Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD/src (main)
$ git commit -m "Create class ClothingRecomender and implemented with a method to return string"
[main fa8d50b] Create class ClothingRecomender and implemented with a method to return string
 1 file changed, 26 insertions(+)

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD/src (main)
$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 803 bytes | 803.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
To https://gitlab.griffith.ie/aline.andradecosta/projectchatbot.git
   46f9f9c..fa8d50b  main -> main

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD/src (main)
$
```

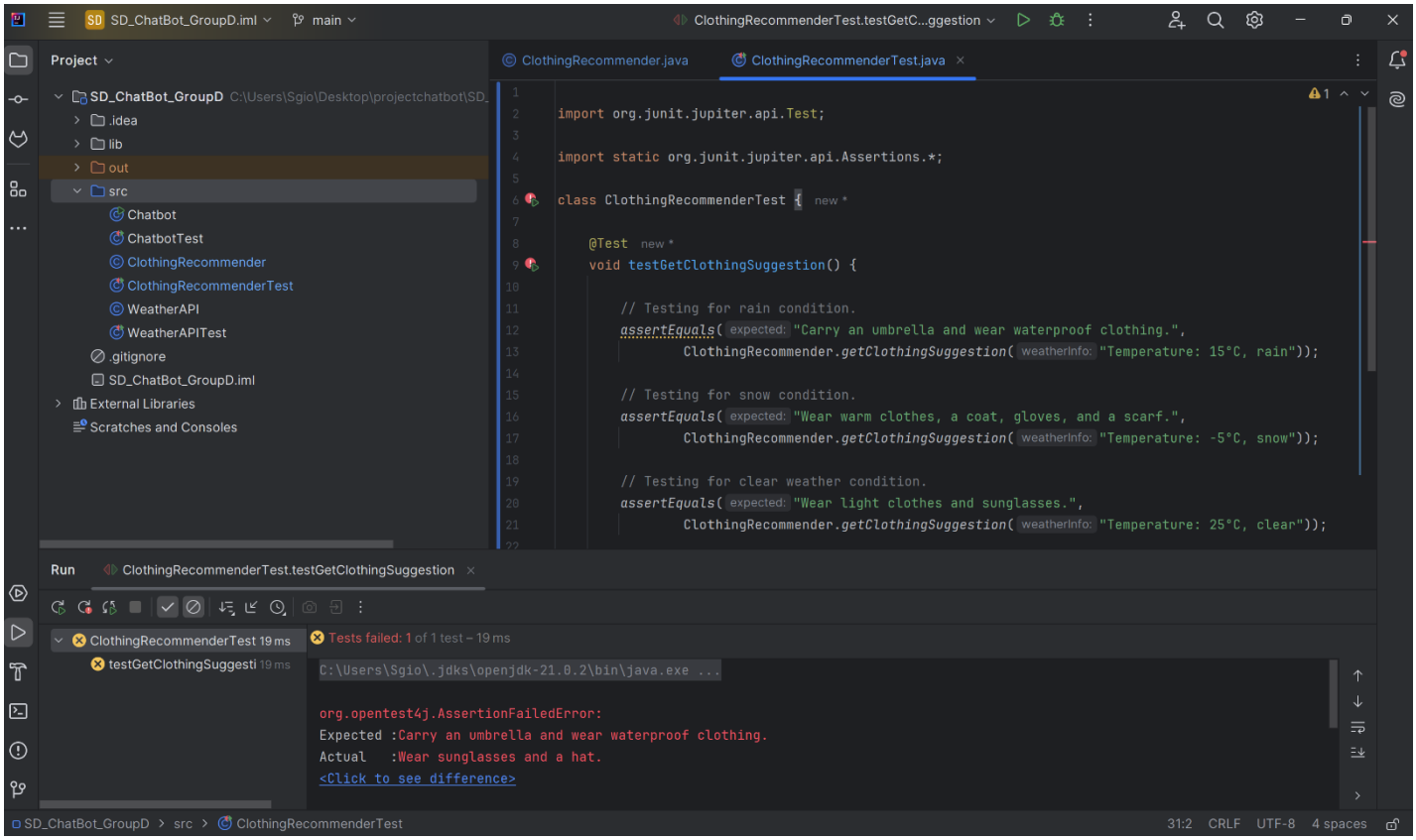
Alteration in the Java interface:

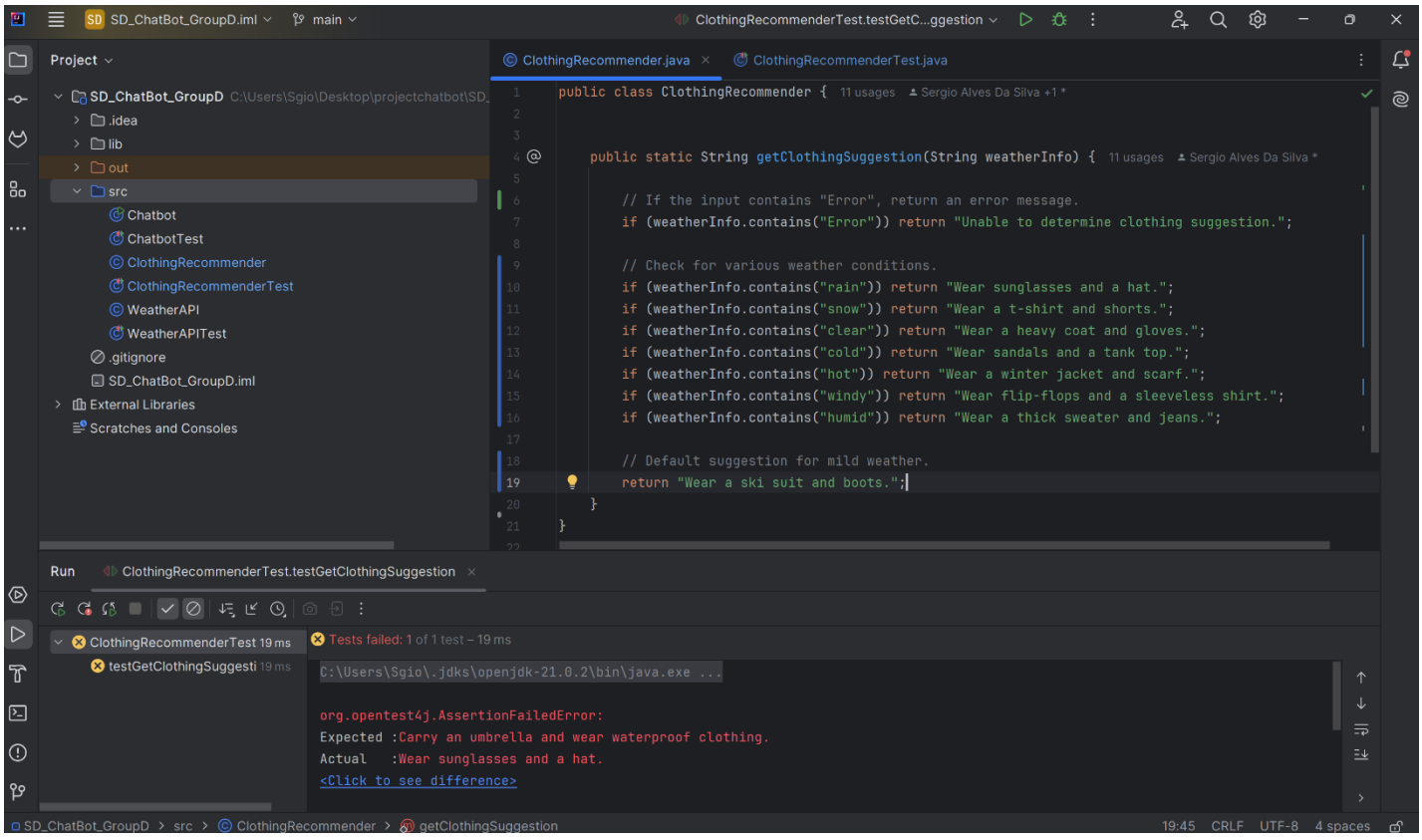


New Commit for alterations in the project:

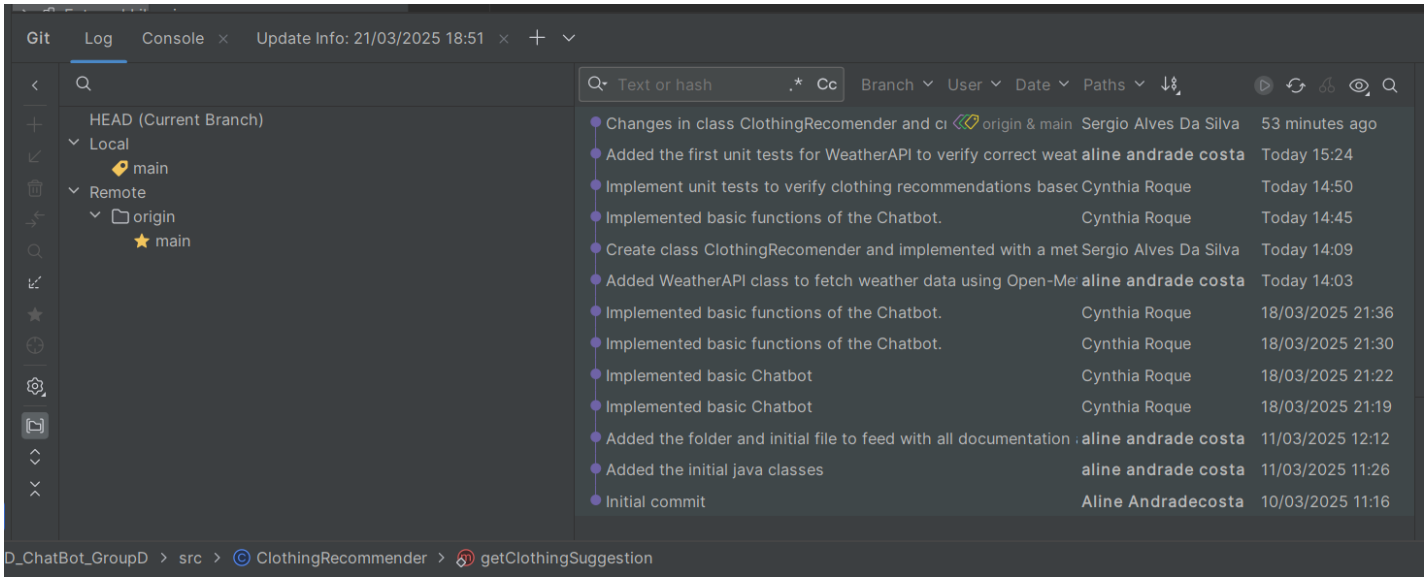
```
Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (main)
$ git pull
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (6/6), done.
remote: Total 28 (delta 6), reused 6 (delta 6), pack-reused 22 (from 1)
Unpacking objects: 100% (28/28), 1.45 MiB | 6.76 MiB/s, done.
From https://gitlab.griffith.ie/aline.andradecosta/projectchatbot
   fa8d50b..5696652  main       -> origin/main
Updating fa8d50b..5696652
Fast-forward
 SD_ChatBot_GroupD/SD_ChatBot_GroupD.iml            |  26 ++++++++++++++++++++
 SD_ChatBot_GroupD/lib/apiguardian-api-1.1.2.jar    | Bin 0 -> 6806 bytes
 SD_ChatBot_GroupD/lib/hamcrest-core-1.3.jar        | Bin 0 -> 45024 bytes
 SD_ChatBot_GroupD/lib/junit-4.13.1.jar             | Bin 0 -> 382708 bytes
 SD_ChatBot_GroupD/lib/junit-jupiter-5.8.1.jar      | Bin 0 -> 6361 bytes
 SD_ChatBot_GroupD/lib/junit-jupiter-api-5.8.1.jar  | Bin 0 -> 193501 bytes
 .../lib/junit-jupiter-engine-5.8.1.jar             | Bin 0 -> 229680 bytes
 .../lib/junit-jupiter-params-5.8.1.jar             | Bin 0 -> 575854 bytes
 .../lib/junit-platform-commons-1.8.1.jar           | Bin 0 -> 100451 bytes
 .../lib/junit-platform-engine-1.8.1.jar            | Bin 0 -> 185778 bytes
 SD_ChatBot_GroupD/lib/opentest4j-1.2.0.jar         | Bin 0 -> 7653 bytes
 SD_ChatBot_GroupD/src/ChatbotTest.java             |  23 ++++++++++++++++++-
 SD_ChatBot_GroupD/src/WeatherAPITest.java          |  23 ++++++++++++++++++--
 13 files changed, 69 insertions(+), 3 deletions(-)
 create mode 100644 SD_ChatBot_GroupD/lib/apiguardian-api-1.1.2.jar
 create mode 100644 SD_ChatBot_GroupD/lib/hamcrest-core-1.3.jar
 create mode 100644 SD_ChatBot_GroupD/lib/junit-4.13.1.jar
 create mode 100644 SD_ChatBot_GroupD/lib/junit-jupiter-5.8.1.jar
 create mode 100644 SD_ChatBot_GroupD/lib/junit-jupiter-api-5.8.1.jar
 create mode 100644 SD_ChatBot_GroupD/lib/junit-jupiter-engine-5.8.1.jar
 create mode 100644 SD_ChatBot_GroupD/lib/junit-jupiter-params-5.8.1.jar
 create mode 100644 SD_ChatBot_GroupD/lib/junit-platform-commons-1.8.1.jar
 create mode 100644 SD_ChatBot_GroupD/lib/junit-platform-engine-1.8.1.jar
 create mode 100644 SD_ChatBot_GroupD/lib/opentest4j-1.2.0.jar

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/ClothingRecommender.java
        modified:   src/ClothingRecommenderTest.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .idea/.name

no changes added to commit (use "git add" and/or "git commit -a")

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (main)
$ git add .

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .idea/.name
        modified:   src/ClothingRecommender.java
        modified:   src/ClothingRecommenderTest.java


Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (main)
$ git commit -m "Changes in class ClothingRecomender and create a junit test for the same class where method failed"
[main f24c6c5] Changes in class ClothingRecomender and create a junit test for the same class where method failed
 3 files changed, 44 insertions(+), 21 deletions(-)
 create mode 100644 SD_ChatBot_GroupD/.idea/.name

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (main)
$ git push origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 20 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 1.28 KiB | 1.28 MiB/s, done.
Total 8 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
To https://gitlab.griffith.ie/aline.andradecosta/projectchatbot.git
   5696652..f24c6c5  main -> main

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (main)
$
```

Alterations in java interface:


SD_ChatBot_GroupD.iml
main
ClothingRecommenderTest.testGetC...ggestion
Project
SD_ChatBot_GroupD C:\Users\Sgio\Desktop\projectchatbot\SD_
.idea
lib
out
src
Chatbot
ChatbotTest
ClothingRecommender
ClothingRecommenderTest
WeatherAPI
WeatherAPITest
.gitignore
SD_ChatBot_GroupD.iml
External Libraries
Scratches and Consoles
ClothingRecommender.java
ClothingRecommenderTest.java
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
class ClothingRecommenderTest { new *
@Test new *
void testGetClothingSuggestion() {
// Testing for rain condition.
assertEquals(expected: "Carry an umbrella and wear waterproof clothing.",
ClothingRecommender.getClothingSuggestion(weatherInfo: "Temperature: 15°C, rain"));
// Testing for snow condition.
assertEquals(expected: "Wear warm clothes, a coat, gloves, and a scarf.",
ClothingRecommender.getClothingSuggestion(weatherInfo: "Temperature: -5°C, snow"));
// Testing for clear weather condition.
assertEquals(expected: "Wear light clothes and sunglasses.",
ClothingRecommender.getClothingSuggestion(weatherInfo: "Temperature: 25°C, clear"));
Run
ClothingRecommenderTest.testGetClothingSuggestion
ClothingRecommenderTest 19 ms
testGetClothingSuggesti 19 ms
Tests failed: 1 of 1 test – 19 ms
C:\Users\Sgio\.jdks\openjdk-21.0.2\bin\java.exe ...
org.opentest4j.AssertionFailedError:
Expected :Carry an umbrella and wear waterproof clothing.
Actual :Wear sunglasses and a hat.
<Click to see difference>
SD_ChatBot_GroupD > src > ClothingRecommenderTest
31:2 CRLF UTF-8 4 spaces


SD_ChatBot_GroupD.iml
main
ClothingRecommenderTest.testGetC...ggestion
Project
SD_ChatBot_GroupD C:\Users\Sgio\Desktop\projectchatbot\SD_
.idea
lib
out
src
Chatbot
ChatbotTest
ClothingRecommender
ClothingRecommenderTest
WeatherAPI
WeatherAPITest
.gitignore
SD_ChatBot_GroupD.iml
External Libraries
Scratches and Consoles
ClothingRecommender.java
ClothingRecommenderTest.java
public class ClothingRecommender { 11 usages Sergio Alves Da Silva +1 *
public static String getClothingSuggestion(String weatherInfo) { 11 usages Sergio Alves Da Silva *
// If the input contains "Error", return an error message.
if (weatherInfo.contains("Error")) return "Unable to determine clothing suggestion.";
// Check for various weather conditions.
if (weatherInfo.contains("rain")) return "Wear sunglasses and a hat.";
if (weatherInfo.contains("snow")) return "Wear a t-shirt and shorts.";
if (weatherInfo.contains("clear")) return "Wear a heavy coat and gloves.";
if (weatherInfo.contains("cold")) return "Wear sandals and a tank top.";
if (weatherInfo.contains("hot")) return "Wear a winter jacket and scarf.";
if (weatherInfo.contains("windy")) return "Wear flip-flops and a sleeveless shirt.";
if (weatherInfo.contains("humid")) return "Wear a thick sweater and jeans.";
// Default suggestion for mild weather.
return "Wear a ski suit and boots.";
}
}
Run
ClothingRecommenderTest.testGetClothingSuggestion
ClothingRecommenderTest 19 ms
testGetClothingSuggesti 19 ms
Tests failed: 1 of 1 test – 19 ms
C:\Users\Sgio\.jdks\openjdk-21.0.2\bin\java.exe ...
org.opentest4j.AssertionFailedError:
Expected :Carry an umbrella and wear waterproof clothing.
Actual :Wear sunglasses and a hat.
<Click to see difference>
SD_ChatBot_GroupD > src > ClothingRecommender > getClothingSuggestion
19:45 CRLF UTF-8 4 spaces

8.4 Full Log Details



9. Milestone 2

9.1 Goals

Planned Upgrades for Milestone 2

→ **Improve Weather API Handling:**

Add support for multiple locations and 3-day forecasts.

→ **Enhance Clothing Suggestions:**

Considering time of day and weather trends.

→ **Basic GUI Implementation:**

Using JavaFX for better user interaction.

→ **Refine Version Control:**

Implement branches, commits, and tags per versioning strategy.

→ **More JUnit Tests:**

Expand test coverage for API handling and UI components.

9.2 Junit Tests Integration

The following JUnit tests have been integrated to ensure the functionality of the Chatbot, Clothing Recommendations, and Weather API:

→ ChatbotTest:

testSunnyWeatherClothing(): Verifies that the chatbot recommends light clothing for sunny weather.

testRainyWeatherClothing(): Verifies that the chatbot recommends a raincoat and waterproof shoes for rainy weather.

→ ClothingRecommenderTest:

testColdWeatherClothing(): Verifies that the ClothingRecommender suggests appropriate clothing for cold weather (e.g., 5°C).

→ WeatherAPITest:

testValidLocationWeather(): Ensures that the Weather API returns a valid response for a real location (e.g., "New York").

testInvalidLocationWeather(): Ensures the Weather API returns an error message for invalid or empty locations.

These tests ensure that the core functionalities of the chatbot, including clothing recommendations and weather data handling, work as expected.

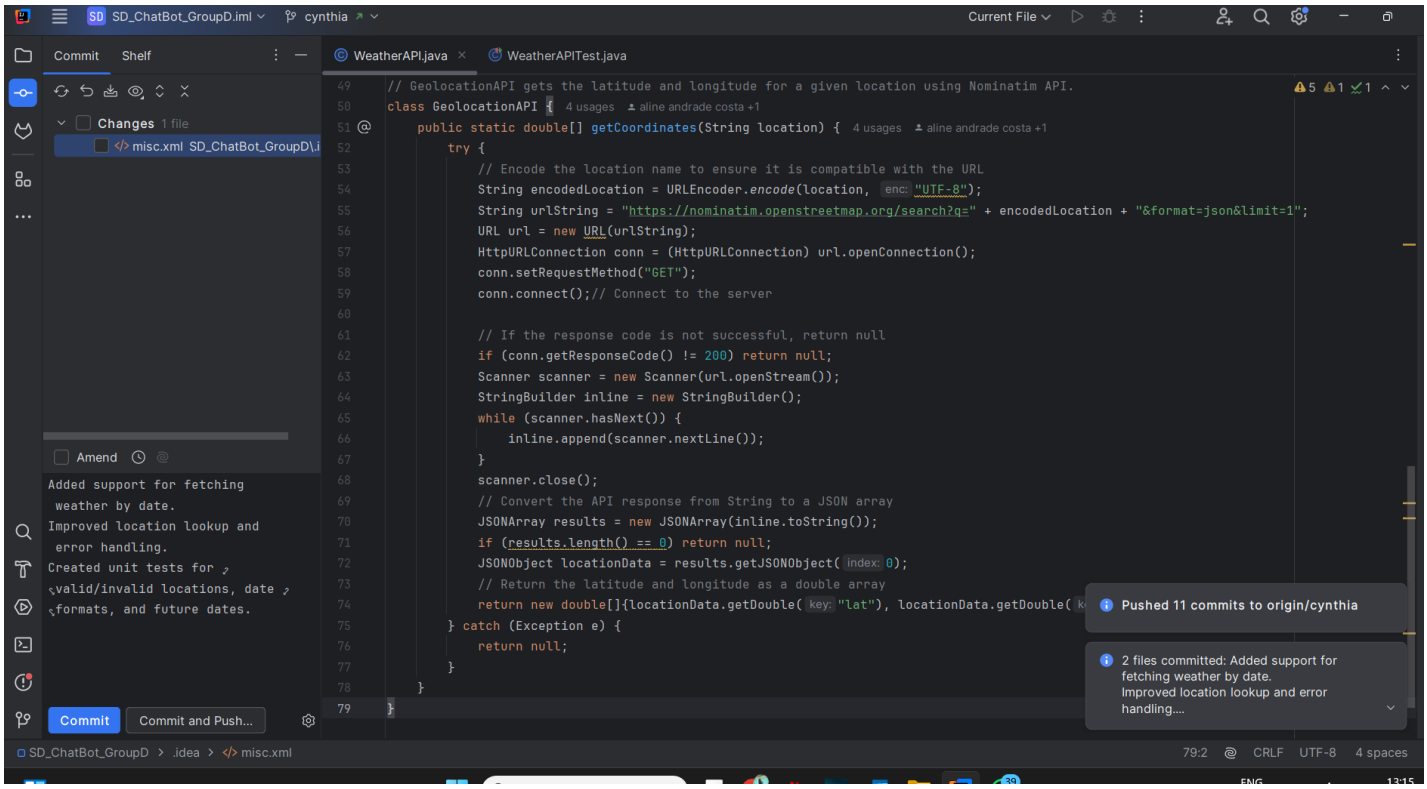
9.3 Commit List & Branches Tree

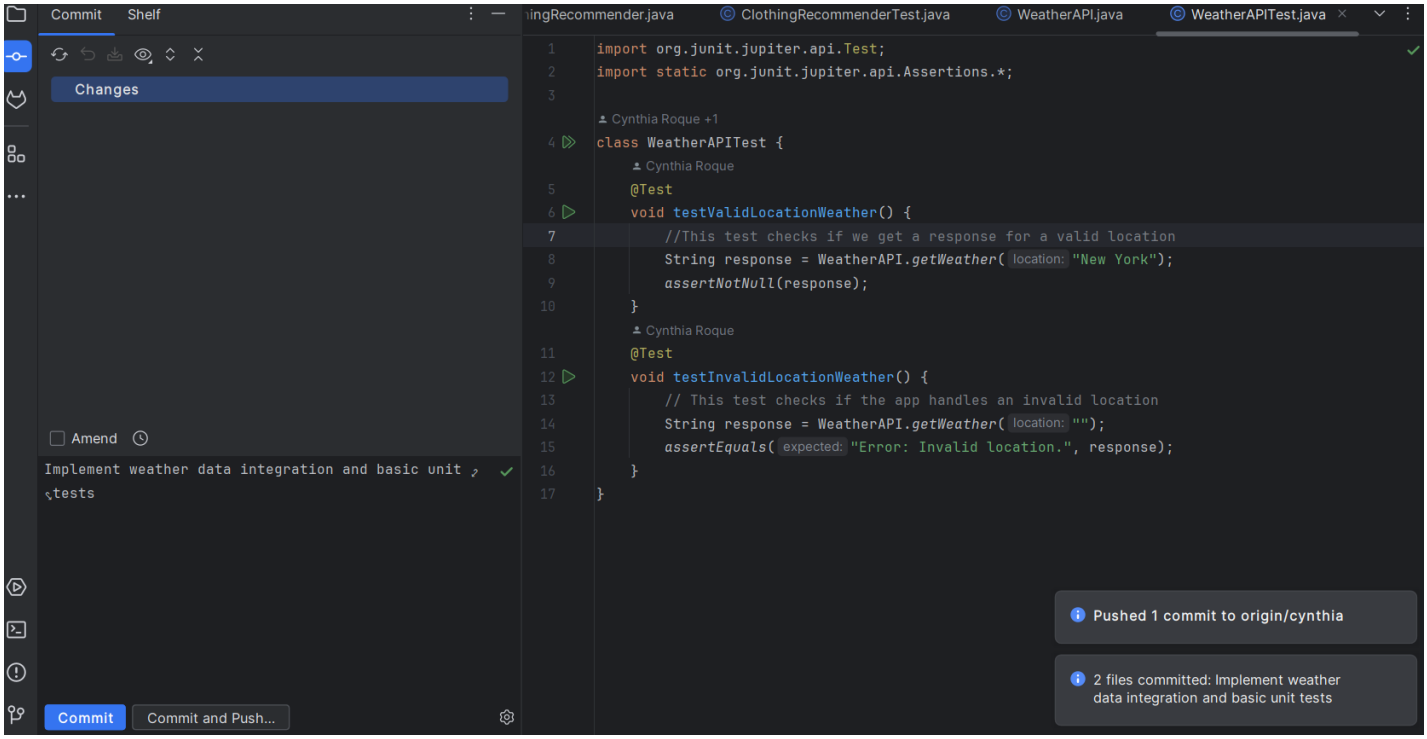
- **Code Implementation**

Cynthia – Weather API and Data Handling

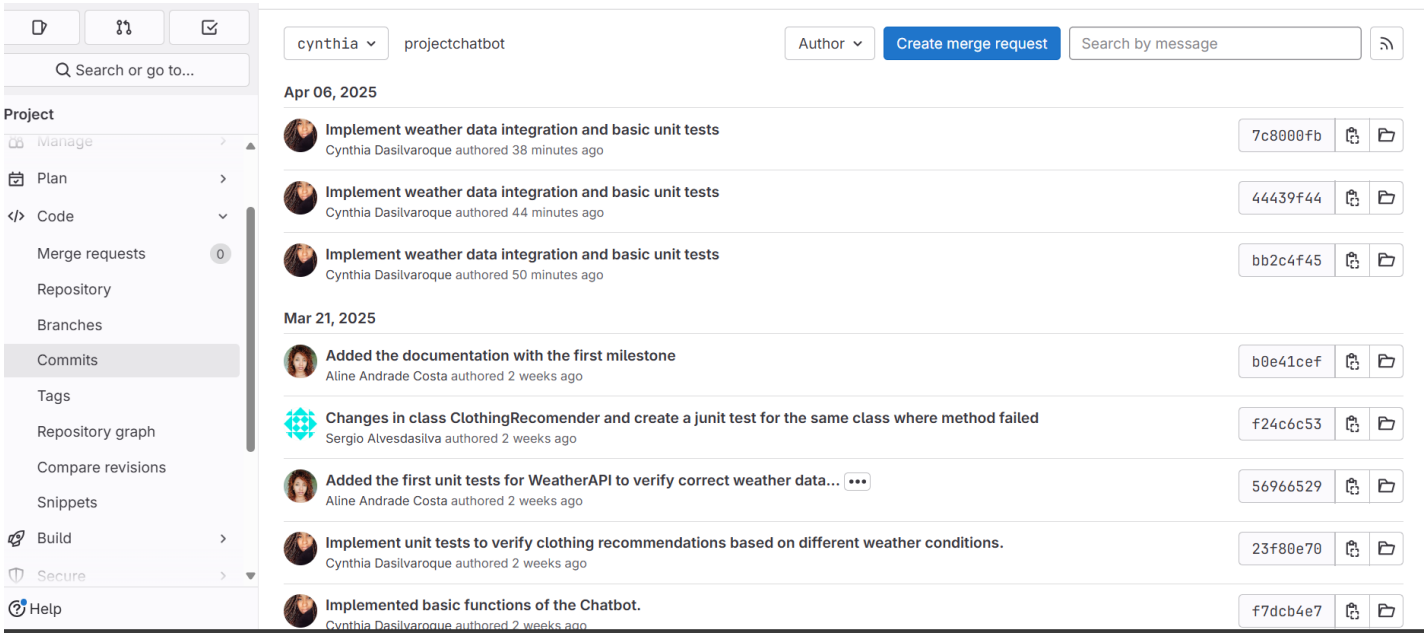
```
MINGW64:/c/Users/cycyn/OneDrive/Documentos/projectchatbot

cycyn@Cyntha MINGW64 ~/OneDrive/Documentos/projectchatbot (cynthia)
$ git pull
remote: Enumerating objects: 16, done.
remote: Total 16 (delta 0), reused 0 (delta 0), pack-reused 16 (from 1)
Unpacking objects: 100% (16/16), 6.74 KiB | 104.00 KiB/s, done.
From https://gitlab.griffith.ie/aline.andradecosta/projectchatbot
   5608aa3..c63a8bc  sergio91   -> origin/sergio91
Already up to date.

cycyn@Cyntha MINGW64 ~/OneDrive/Documentos/projectchatbot (cynthia)
$
```

Implement weather data integration and basic unit tests



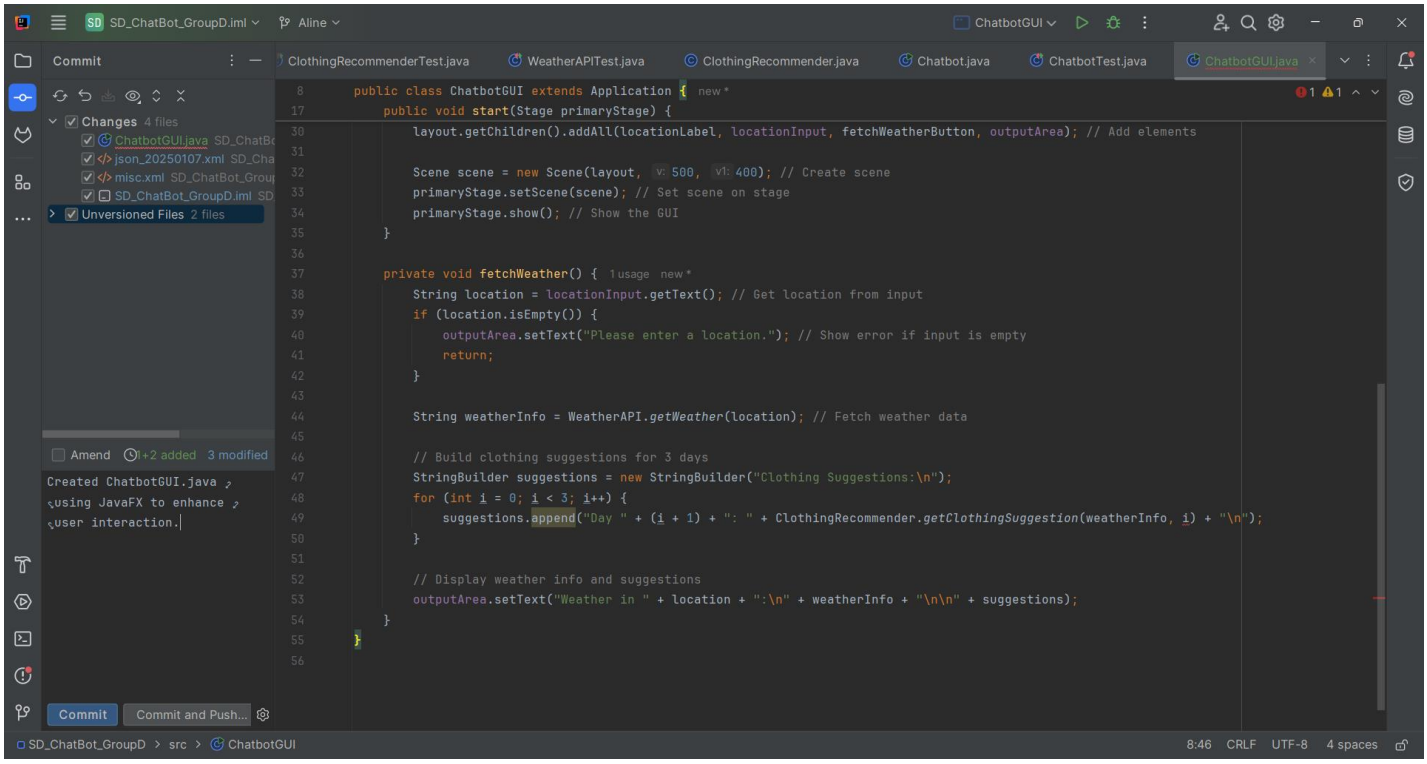


Commits

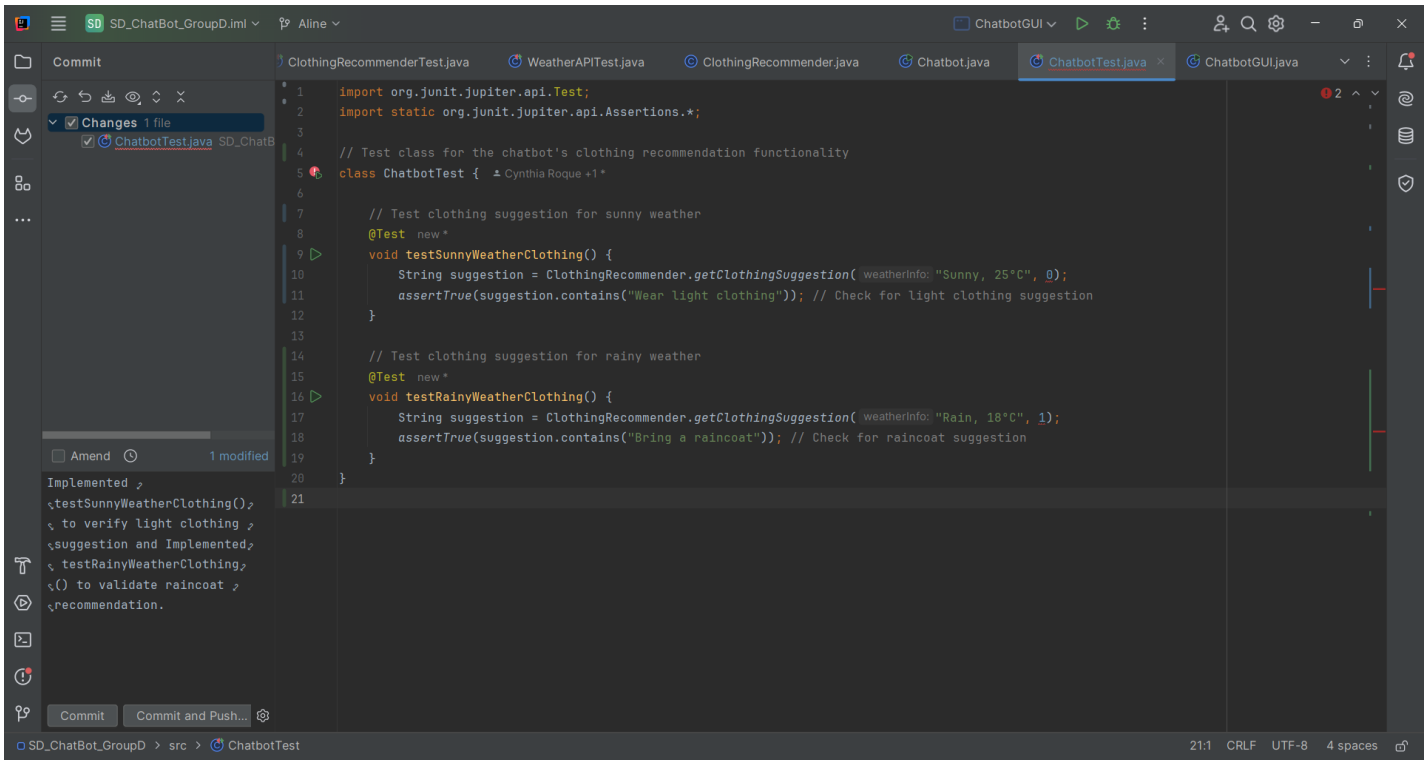


Aline – JavaFX GUI Enhancements and Chatbot Test

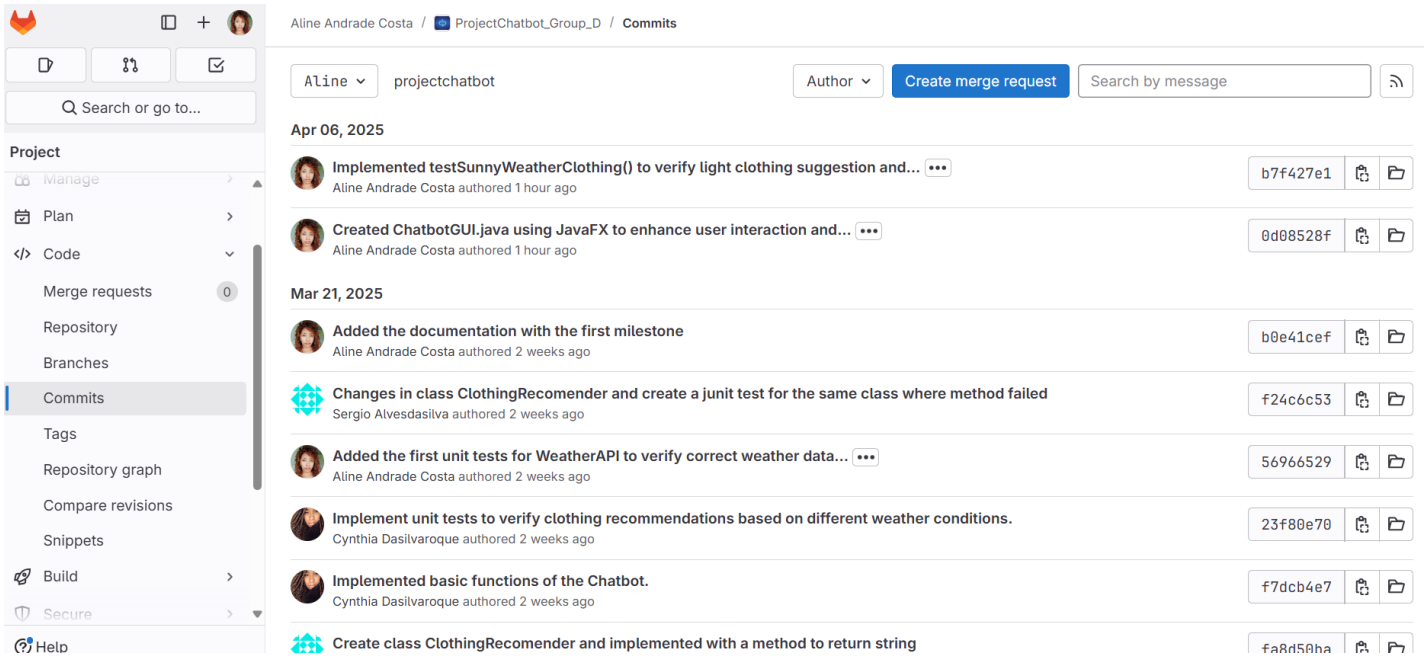
Created ChatbotGUI.java using JavaFX to enhance user interaction and integrated fetchWeather() with WeatherAPI and ClothingRecommender.



Add clothing suggestion tests for sunny and rainy weather

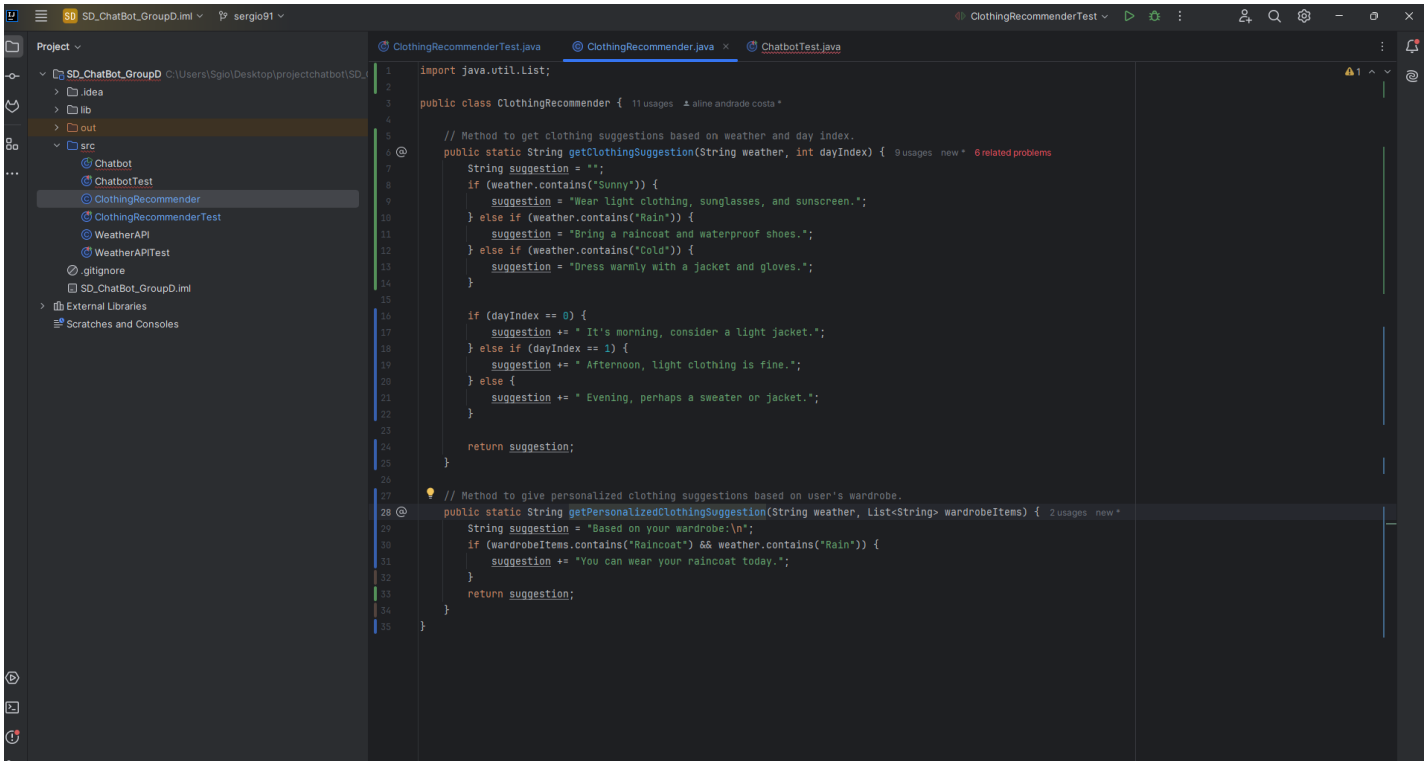


Commits

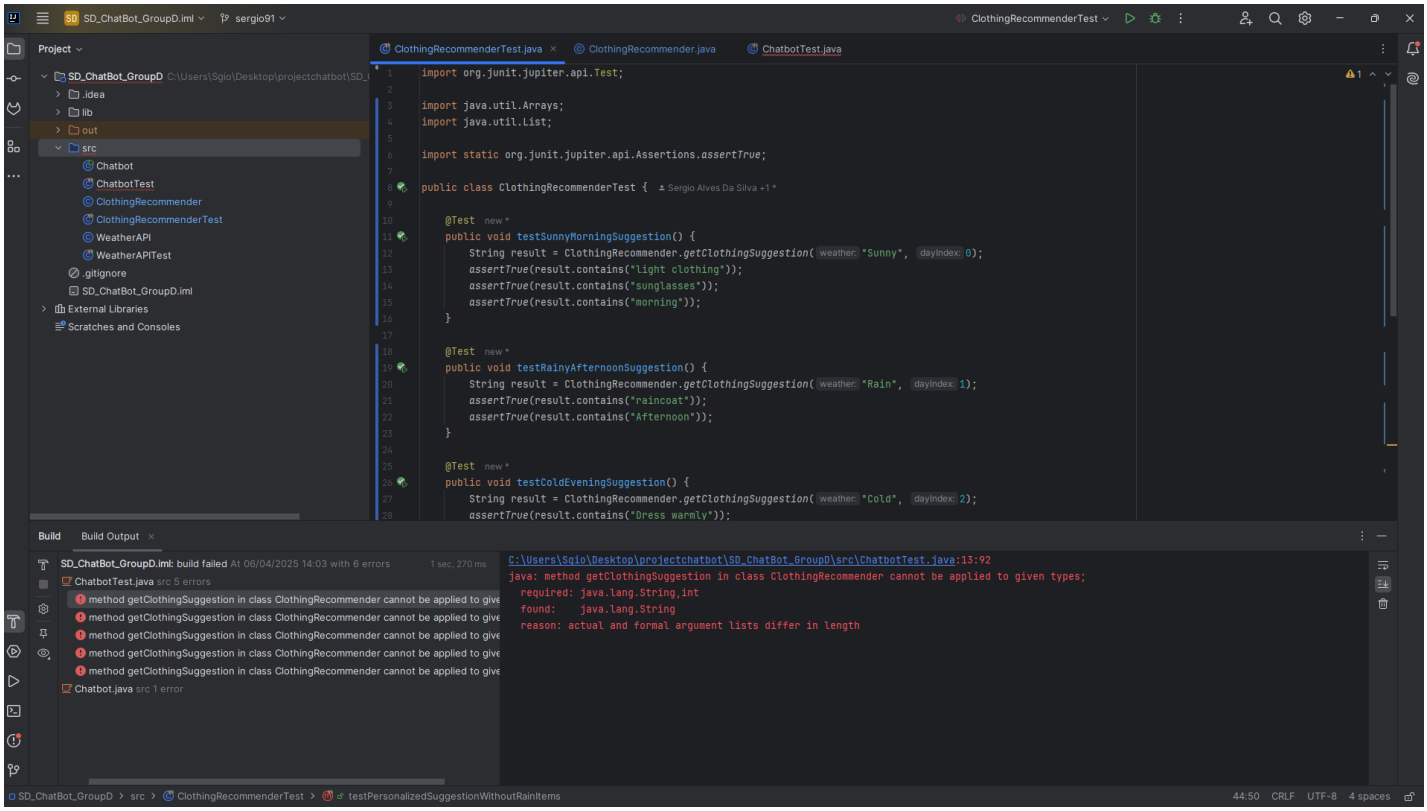


Sergio – Chatbot Logic and Clothing Recommendations

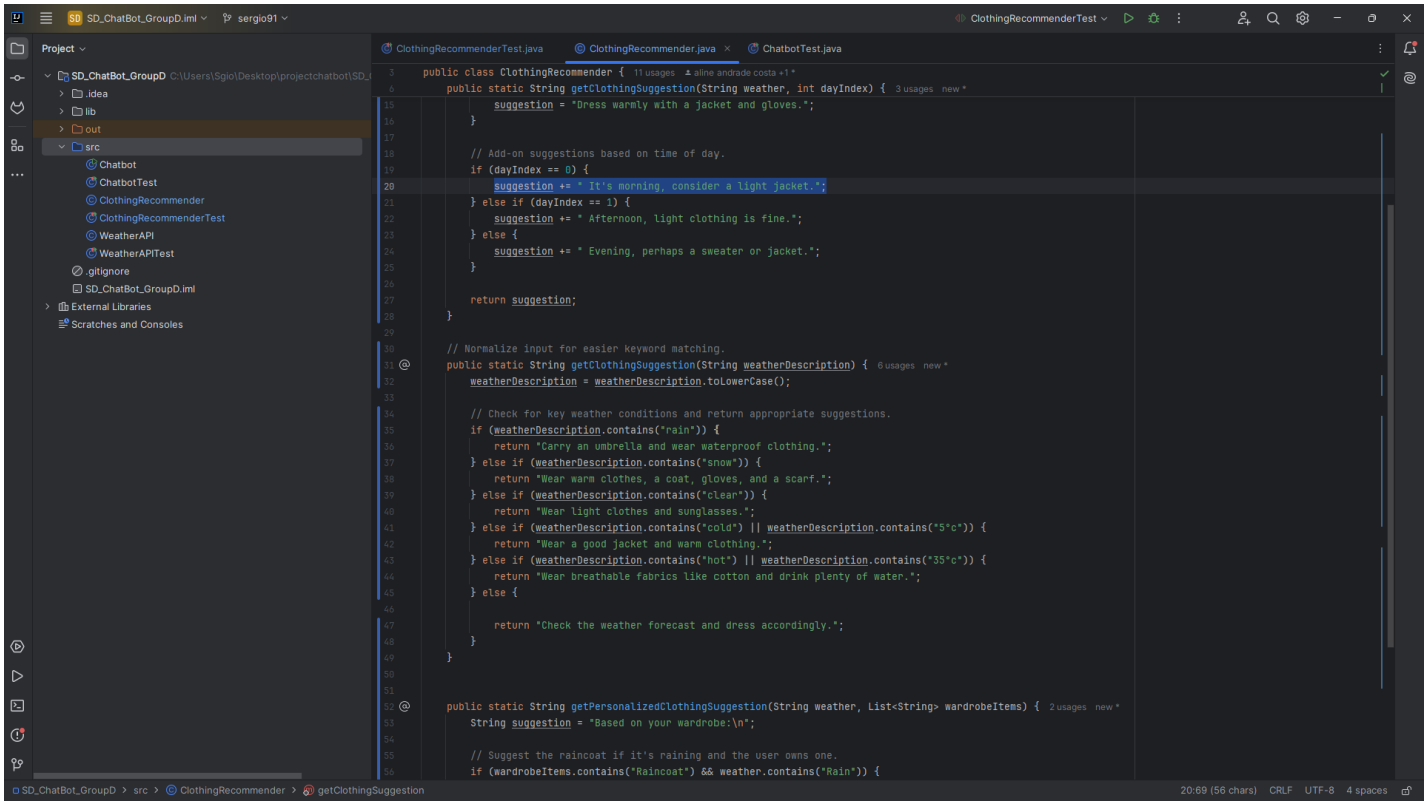
The class ClothingRecommender was implemented.



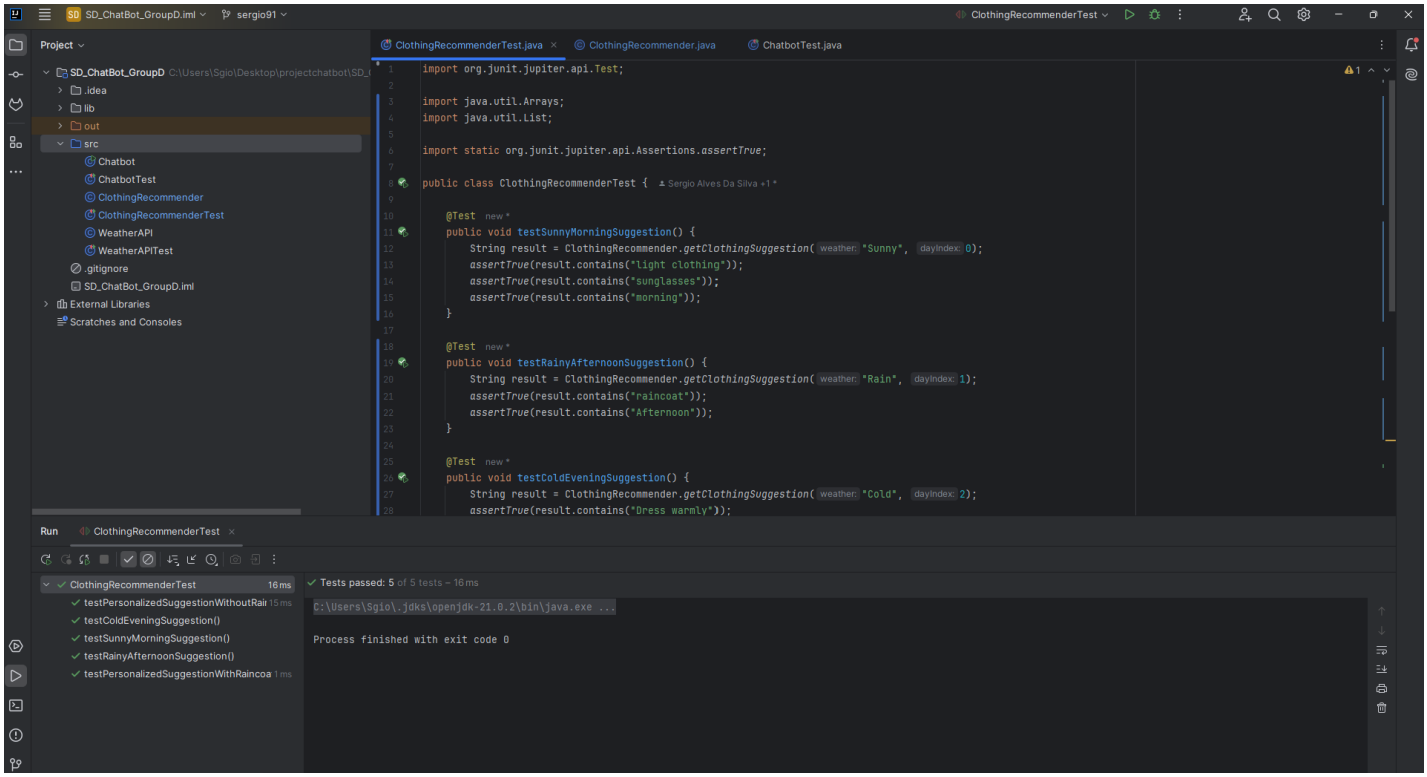
The ClothingRecommenderTest failed with the new implementation.



The class ClothingRecommender was implemented with a new method to improve functionality.



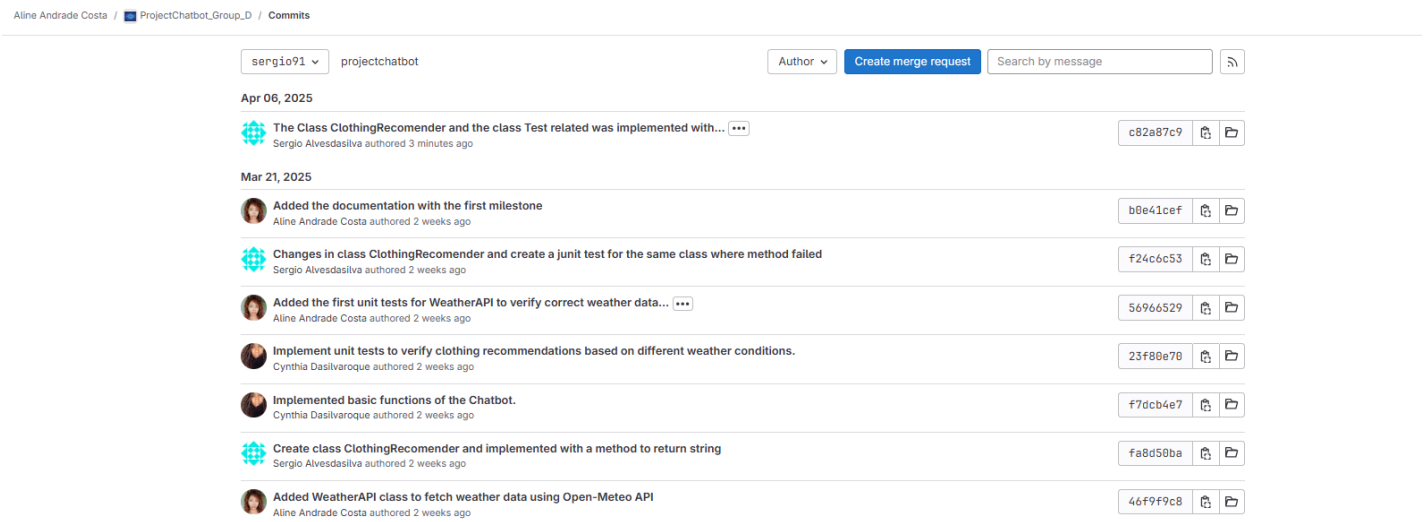
The ClothingRecommenderTest passed all tests of the code.



A commit was done in the version control about all improvements to the code.

```
Already up to date.

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (main)
$ git checkout sergio91
Switched to branch 'sergio91'
M       SD_ChatBot_GroupD/src/ClothingRecommender.java
M       SD_ChatBot_GroupD/src/ClothingRecommenderTest.java

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git status
On branch sergio91
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/ClothingRecommender.java
        modified:   src/ClothingRecommenderTest.java

no changes added to commit (use "git add" and/or "git commit -a")

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git add .

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git status
On branch sergio91
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/ClothingRecommender.java
        modified:   src/ClothingRecommenderTest.java


Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git commit -m "The Class ClothingRecomender and the class Test related was implemented with more accurancy code and passed on test"
[sergio91 c82a87c] The Class ClothingRecomender and the class Test related was implemented with more accurancy code and passed on test
 2 files changed, 91 insertions(+), 34 deletions(-)

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git push origin sergio91
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 20 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.69 KiB | 1.69 MiB/s, done.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: To create a merge request for sergio91, visit:
remote:   https://gitlab.griffith.ie/aline.andradecosta/projectchatbot/-/merge_requests/new?merge_request%5Bsource_branch%5D=sergio91
remote:
To https://gitlab.griffith.ie/aline.andradecosta/projectchatbot.git
 * [new branch]      sergio91 -> sergio91

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$
```

A commit was sent to the repository of the project.



- ♦ **Branches Tree**

→ **main (Master Branch)**

Contains the stable and final version of the project.

All releases and final code updates will be merged into this branch.

→ **Aline's Branch**

Contains all changes related to GUI improvements (layout, colour, fonts, responsiveness, etc.).

Includes JUnit tests for verifying functionality.

→ **Cynthia's Branch**

Contains all changes for integrating and improving the Weather API.

Handles real-time location lookup, geolocation, and error handling.

Includes JUnit tests for verifying functionality.

→ **Sergio's Branch**

Contains all changes related to clothing recommendation logic based on weather.

Includes improvements to logic and additional recommendations.

Includes JUnit tests for verifying functionality.

9.4 Full Log Details

→ Cynthia – Weather API & Data Handling

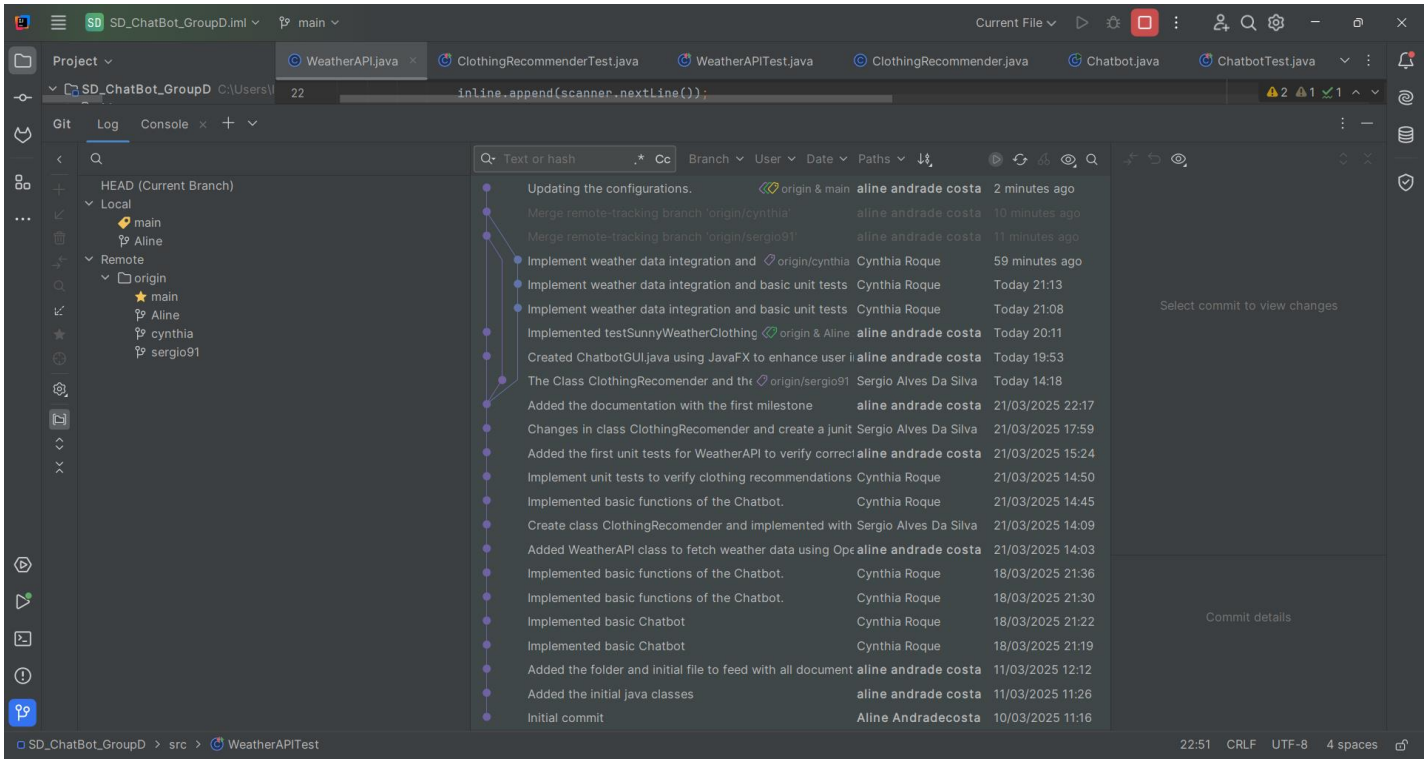
- Improve API integration by handling real location lookups (geolocation API).
- Ensure error handling for incorrect locations.
- Refine the 3-day weather forecast to include humidity, wind speed, and conditions.
- Ensure API calls are optimised without exceeding rate limits.

→ Aline – JavaFX GUI Enhancements

- Improve the design of the GUI (better layout, colours, and fonts).
- Add an icon/logo and improve UI responsiveness.
- Implement a loading indicator while fetching data.
- Allow users to save and review past weather reports.

→ Sergio – Chatbot Logic & Clothing Recommendations

- Improve clothing suggestions (consider time of day & weather patterns).
- Add a feature where users input their wardrobe to get personalised suggestions.
- Implement a speech-to-text feature (optional, for accessibility).
- Write unit tests for all chatbot functionalities.



10.Milestone 3

10.1 Goals

→ Complete and integrate all project features as per the project brief.

→ Add JUnit testing framework to validate chatbot components.

→ Finalize the GUI to interact with the chatbot visually.

Aline will be responsible for working on the chatbot's GUI and styling on the Aline branch with chatbotGUI and CSS style, ensuring that the interface is intuitive, functional, and styled according to the requirements.

Sergio will focus on the chatbot interaction logic and clothing recommendation logic on the sergio91 branch. This includes managing the user input, handling the conversation flow, and using the weather data to suggest appropriate clothing.

Cynthia will be responsible for the Weather API implementation and unit testing on the Cynthia banch with WeatherAPI and WeatherAPITest, ensuring the accuracy of the weather data and handling any edge cases with the API calls.

10.2 Junit Tests Integration

→ ClothingRecommenderTest:

SunnyMorningSuggestion: Verifies that the recommendation for sunny weather includes light clothing and sunglasses for the morning.

RainyAfternoonSuggestion: Tests that the recommendation for rainy weather in the afternoon includes a raincoat.

ColdEveningSuggestion: Ensures that cold weather in the evening prompts the user to dress warmly.

PersonalizedSuggestionWithRaincoat: Checks if the system suggests wearing a raincoat when it's available in the user's wardrobe.

PersonalizedSuggestionWithoutRainItems: Verifies the recommendation when there are no rain-related items in the user's wardrobe.

→ ChatbotTest:

testSunnyWeatherClothing: Confirms that the system suggests light clothing for sunny weather.

testRainyWeatherClothing: Verifies that the system suggests a raincoat for rainy weather.

→ WeatherAPITest:

testValidLocationWeather: Validates the response from the WeatherAPI for a known location (New York).

testInvalidLocationWeather: Ensures the API returns an error message for an empty location input.

testInvalidDateFormat: Tests how the API handles an invalid date format.

testFutureDateForecast: Verifies that the API correctly returns weather data for a future date.

testGetCoordinatesValidLocation: Checks if the GeolocationAPI returns valid coordinates for a real location (Berlin).

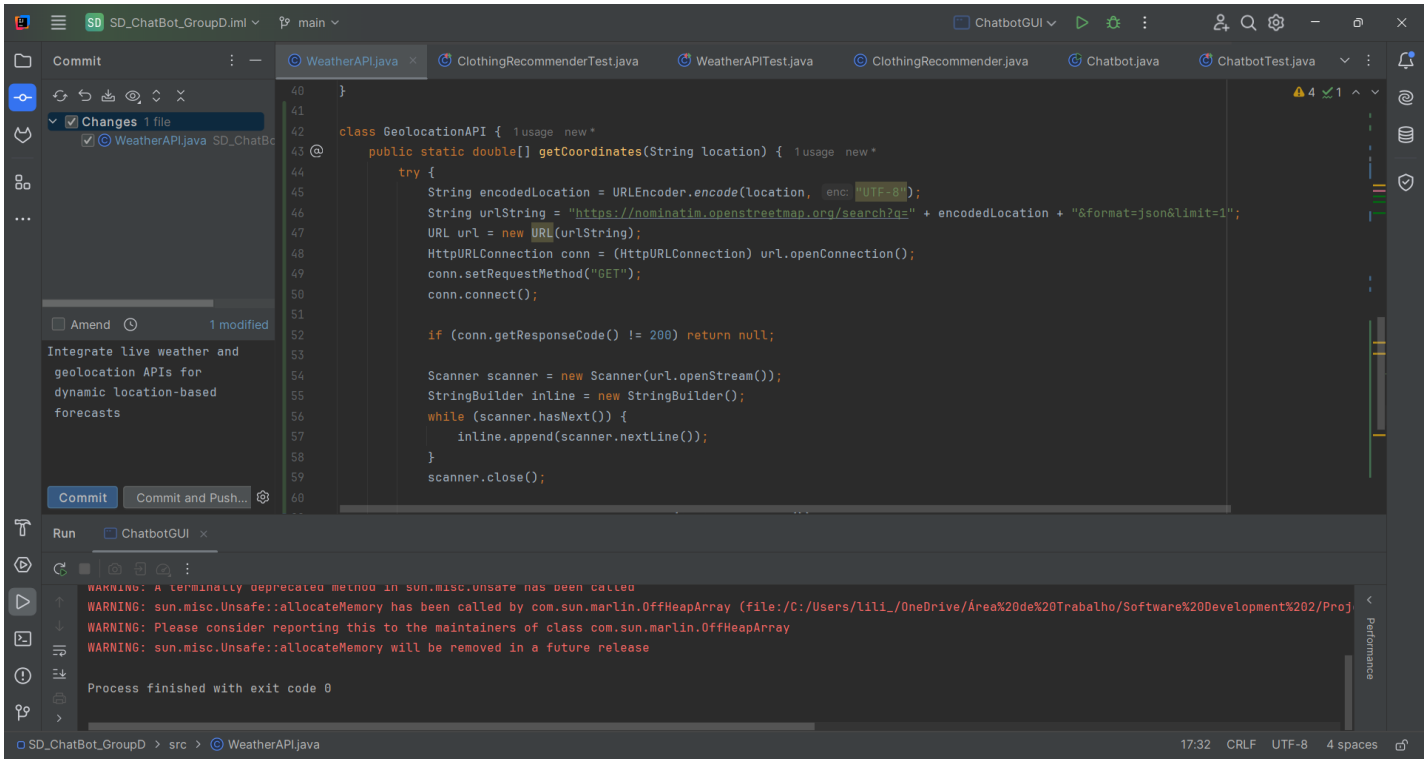
testGetCoordinatesInvalidLocation: Ensures that the GeolocationAPI returns null for an invalid location.

testGetCoordinatesEmptyString: Verifies that the GeolocationAPI returns null for an empty location string.

10.3 Commit List & Branches Tree

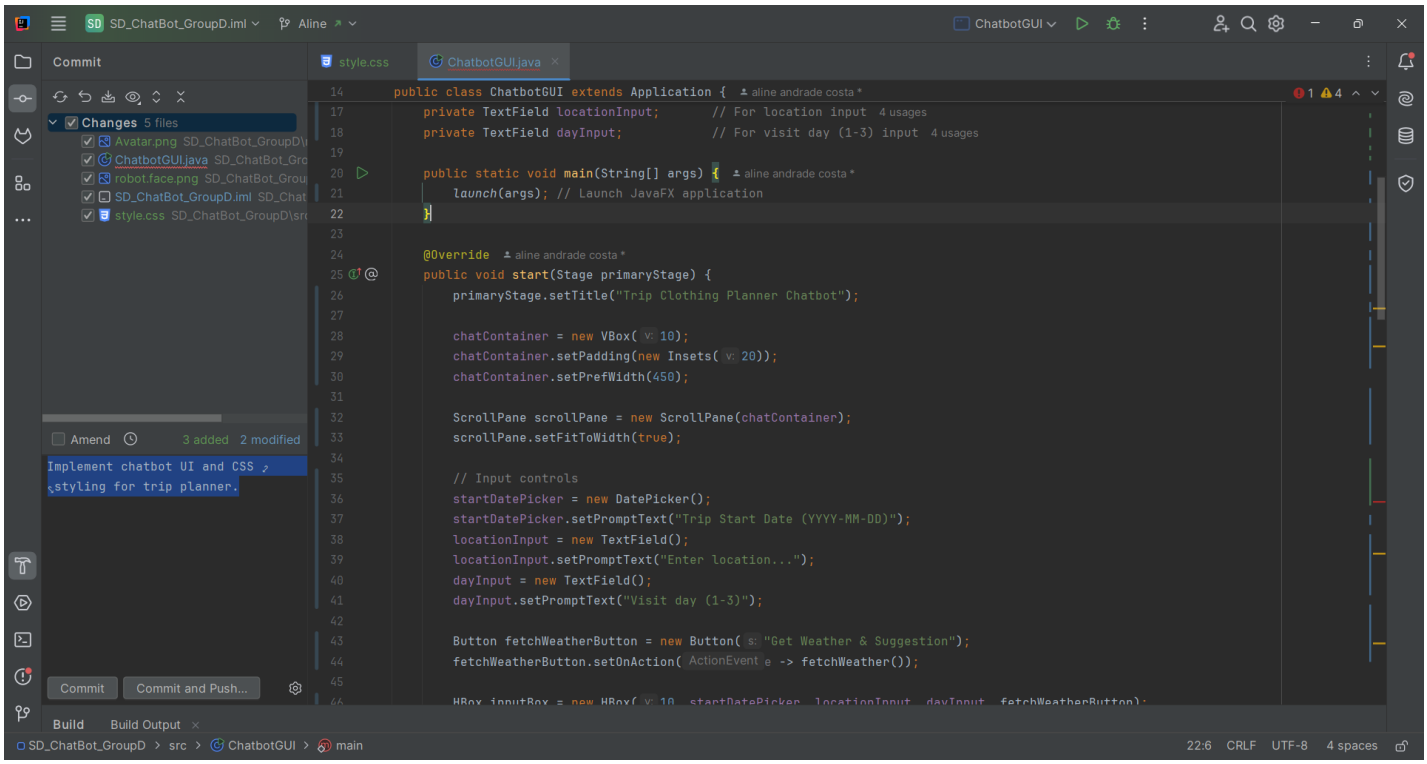
Aline -

Integrate live weather and geolocation APIs for dynamic location-based forecasts.



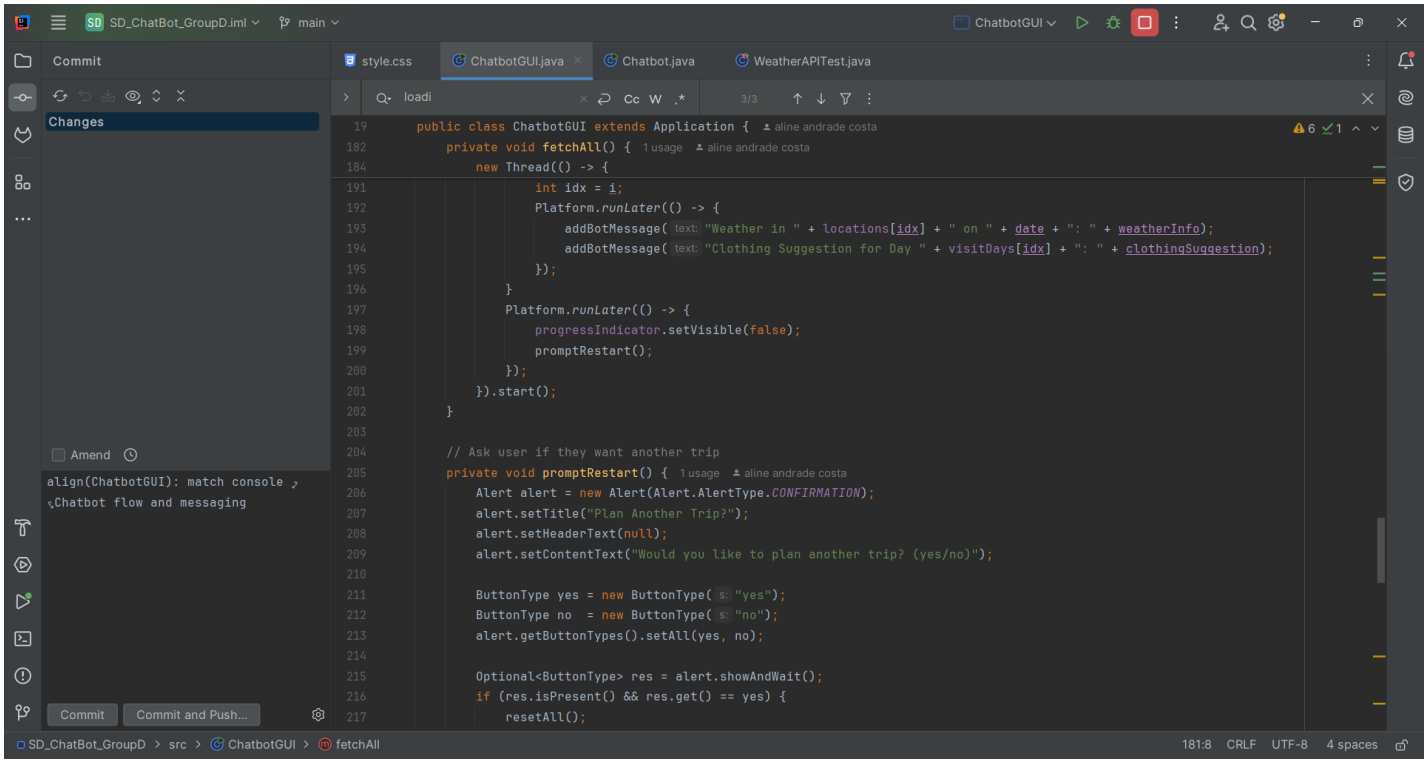

```
}

class GeolocationAPI {  1 usage  new *
    public static double[] getCoordinates(String location) {  1 usage  new *
        try {
            String encodedLocation = URLEncoder.encode(location,  enc: "UTF-8");
            String urlString = "https://nominatim.openstreetmap.org/search?q=" + encodedLocation + "&format=json&limit=1";
            URL url = new URL(urlString);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            conn.connect();

            if (conn.getResponseCode() != 200) return null;

            Scanner scanner = new Scanner(url.openStream());
            StringBuilder inline = new StringBuilder();
            while (scanner.hasNext()) {
                inline.append(scanner.nextLine());
            }
            scanner.close();
```

Implement chatbot UI and CSS styling for the trip planner.



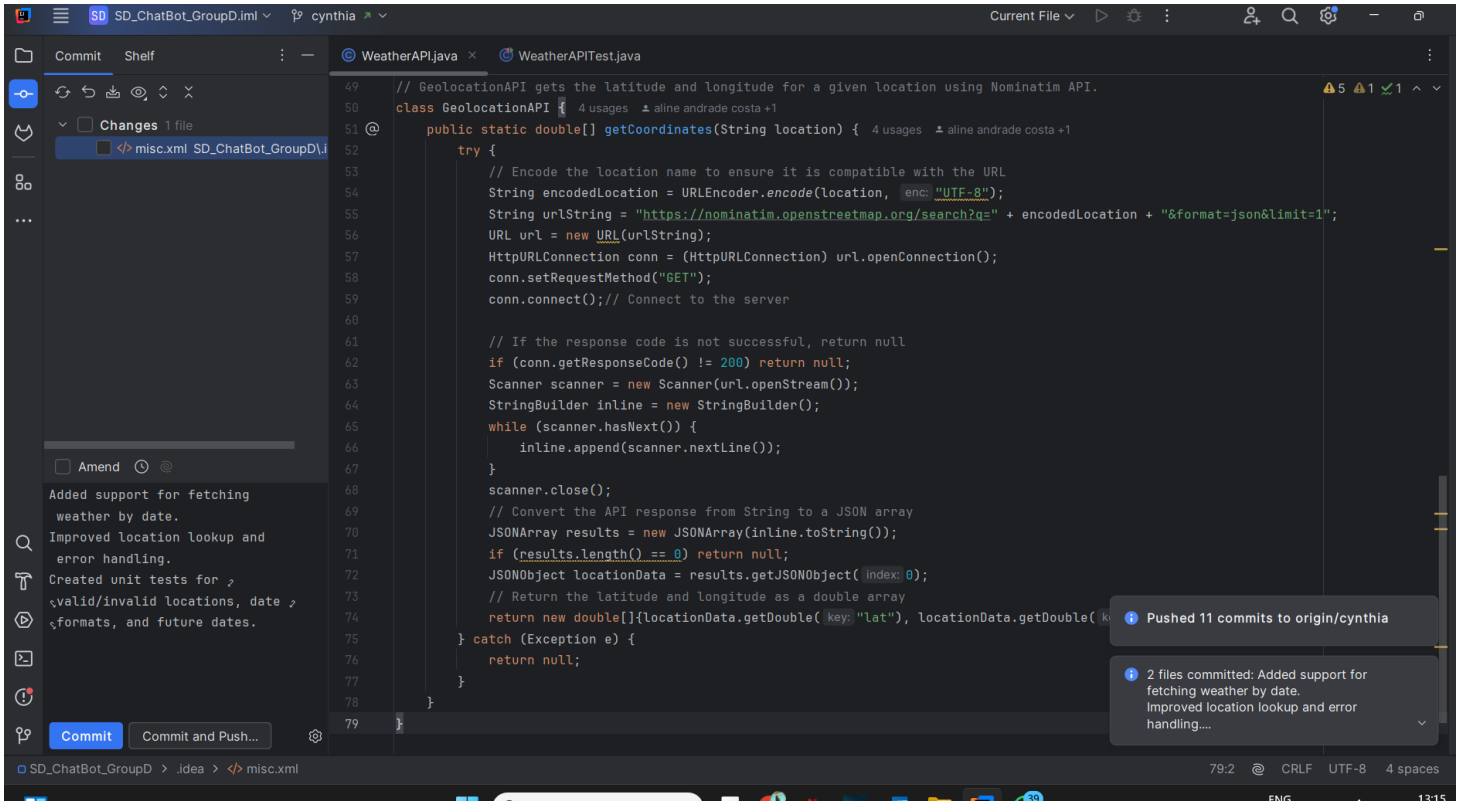

```
public class ChatbotGUI extends Application {  aline andrade costa *
    private TextField locationInput;       // For location input  4 usages
    private TextField dayInput;            // For visit day (1-3) input  4 usages

    public static void main(String[] args) {  aline andrade costa *
        launch(args); // Launch JavaFX application
    }

    @Override  aline andrade costa *
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Trip Clothing Planner Chatbot");

        chatContainer = new VBox( v: 10);
        chatContainer.setPadding(new Insets( v: 20));
        chatContainer.setPrefWidth(450);

        ScrollPane scrollPane = new ScrollPane(chatContainer);
        scrollPane.setFitToWidth(true);

        // Input controls
        startDatePicker = new DatePicker();
        startDatePicker.setPromptText("Trip Start Date (YYYY-MM-DD)");
        locationInput = new TextField();
        locationInput.setPromptText("Enter location...");
        dayInput = new TextField();
        dayInput.setPromptText("Visit day (1-3)");

        Button fetchWeatherButton = new Button( s: "Get Weather & Suggestion");
        fetchWeatherButton.setOnAction( ActionEvent e -> fetchWeather());
```

Align(ChatbotGUI): match console Chatbot flow and messaging

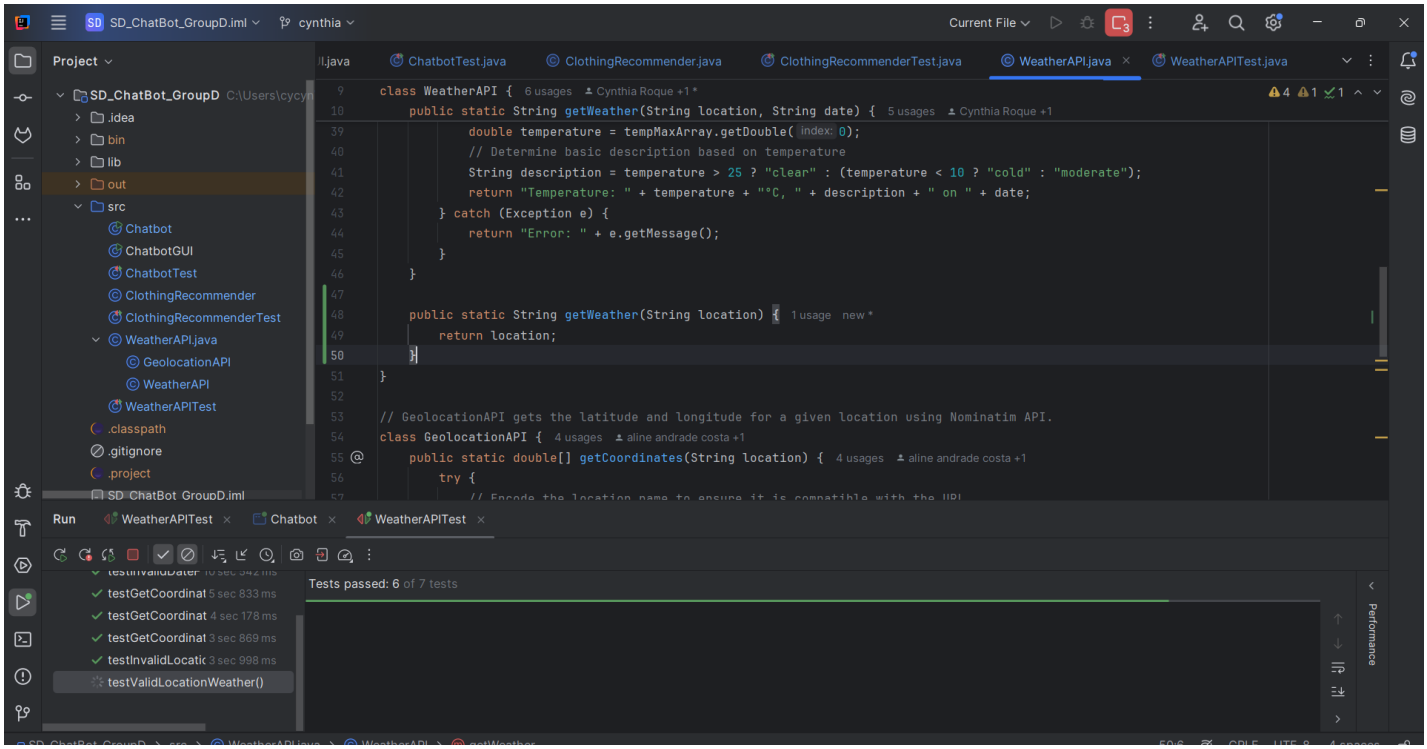


Cynthia -

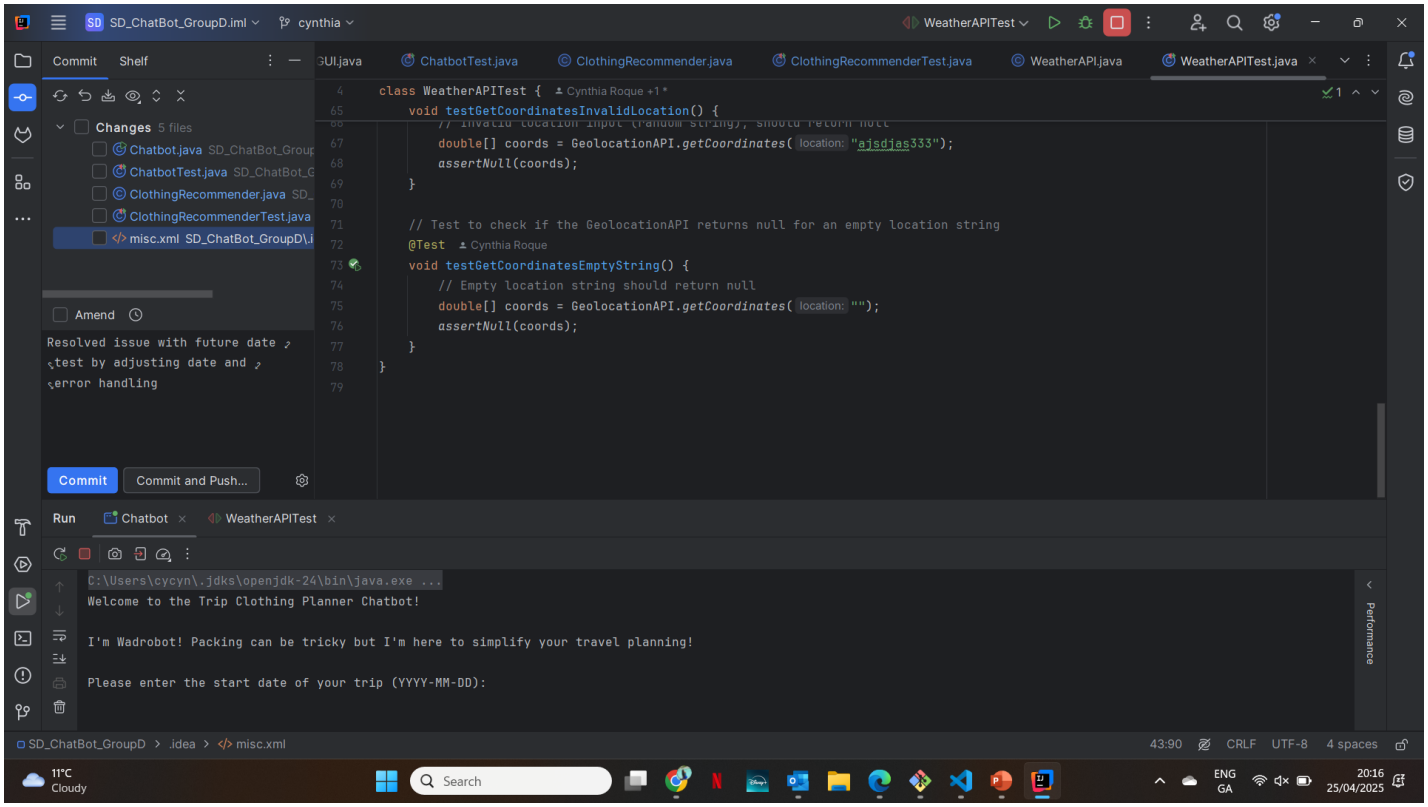
Implemented weather by date, improved location/error handling, and added unit tests.



Resolved issue with future date test by adjusting date and error handling

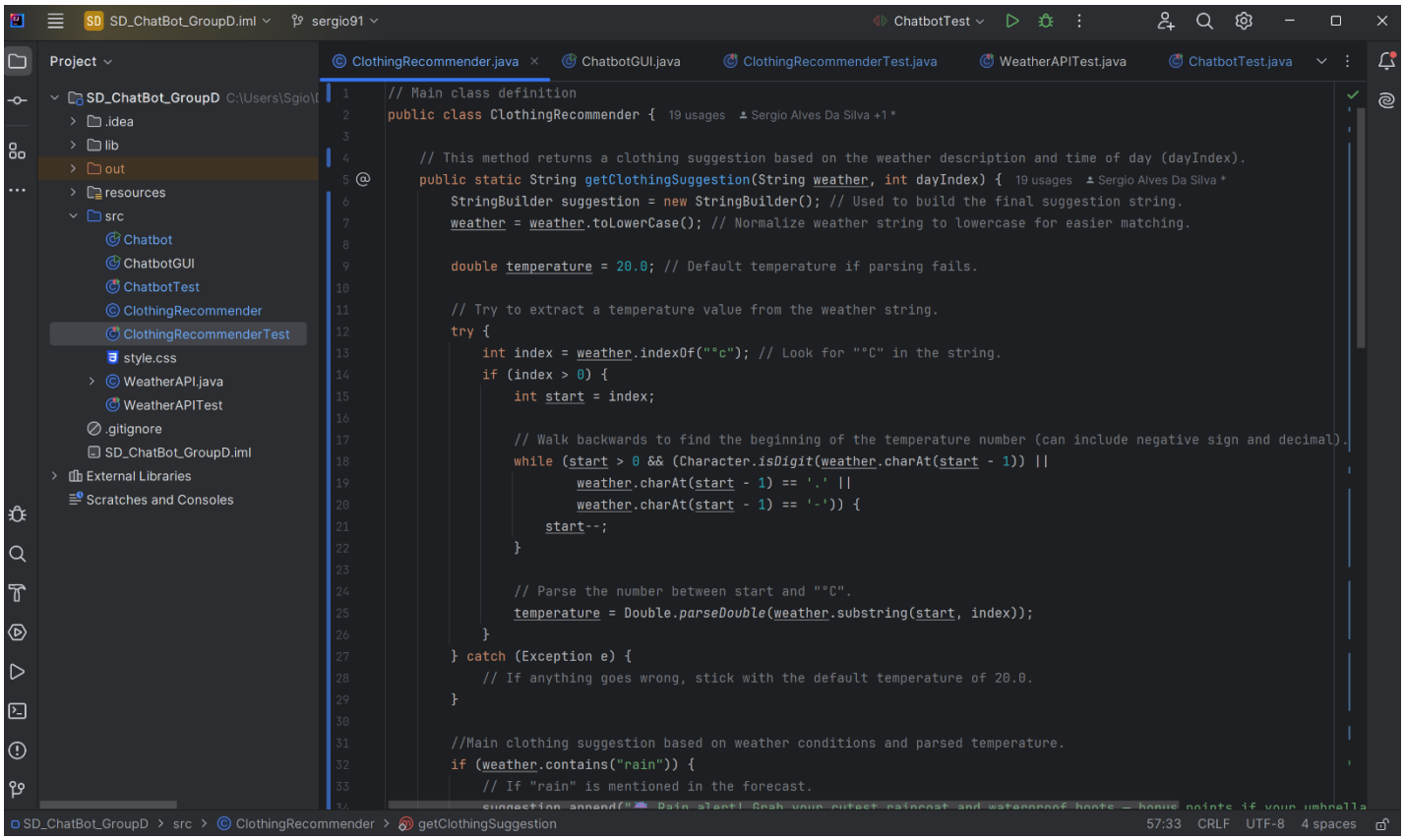


chatbot now working properly

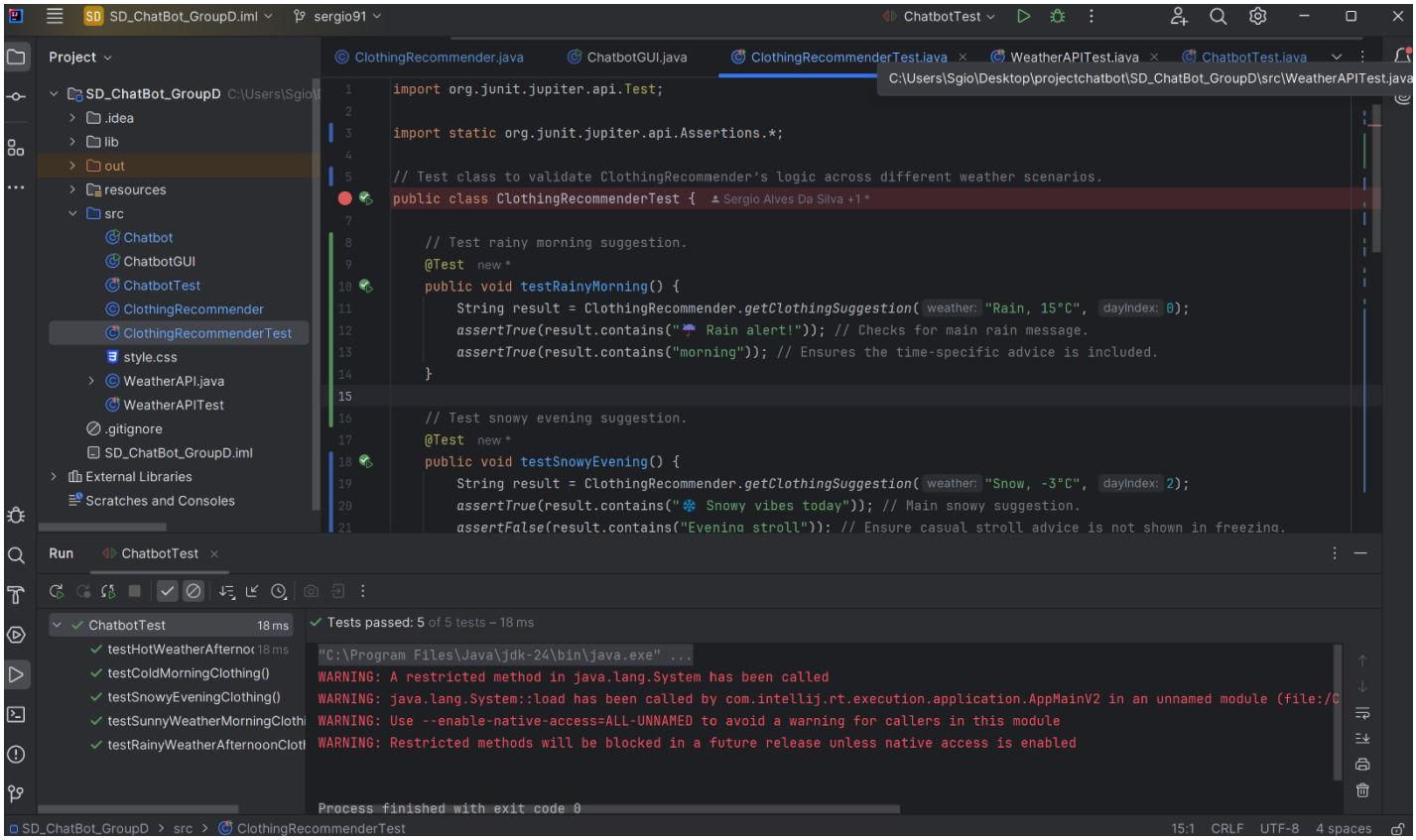


Sergio -

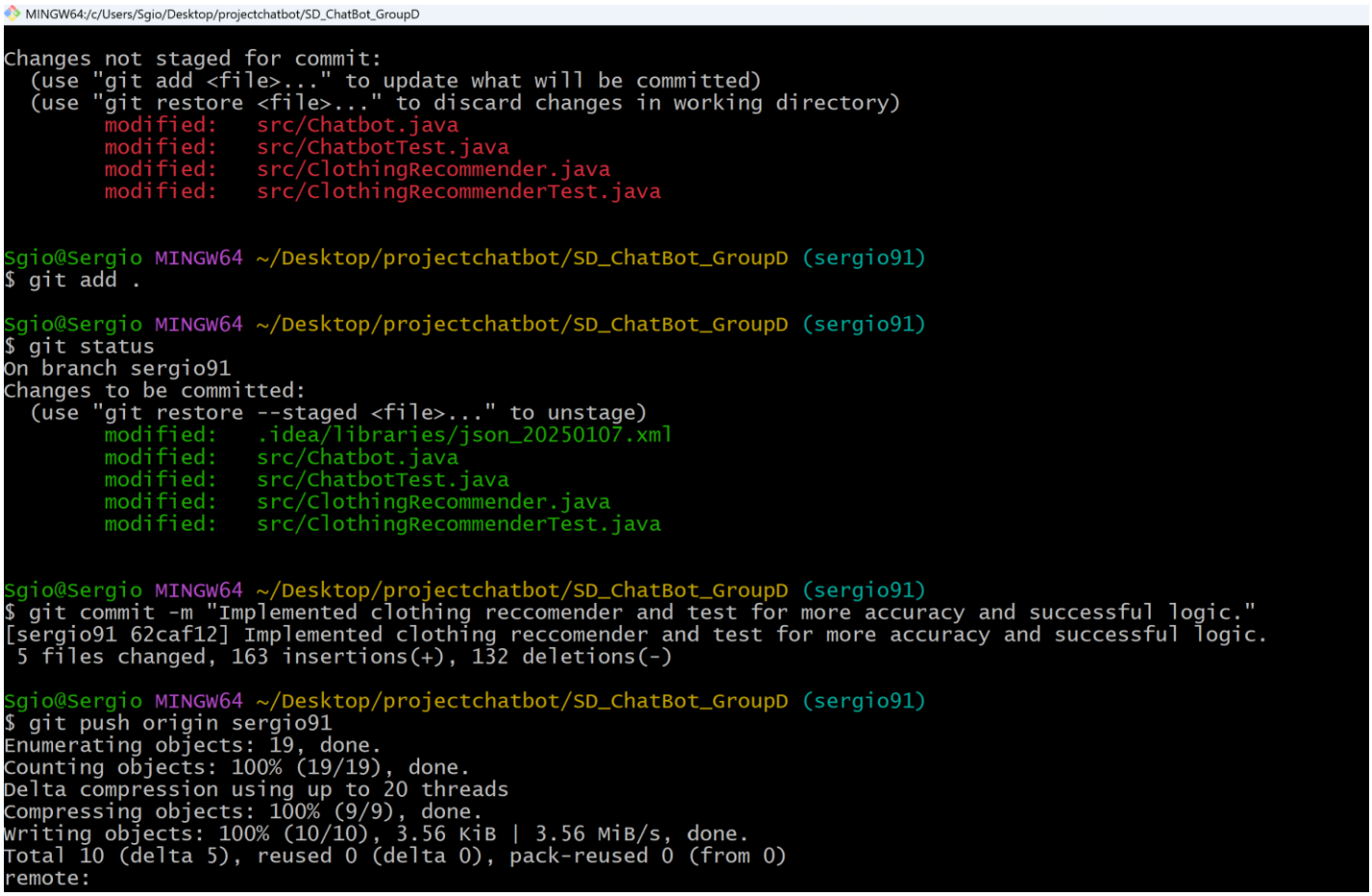
Implemented with more functionalities, methods, and conditions to reach more logic and data accuracy.



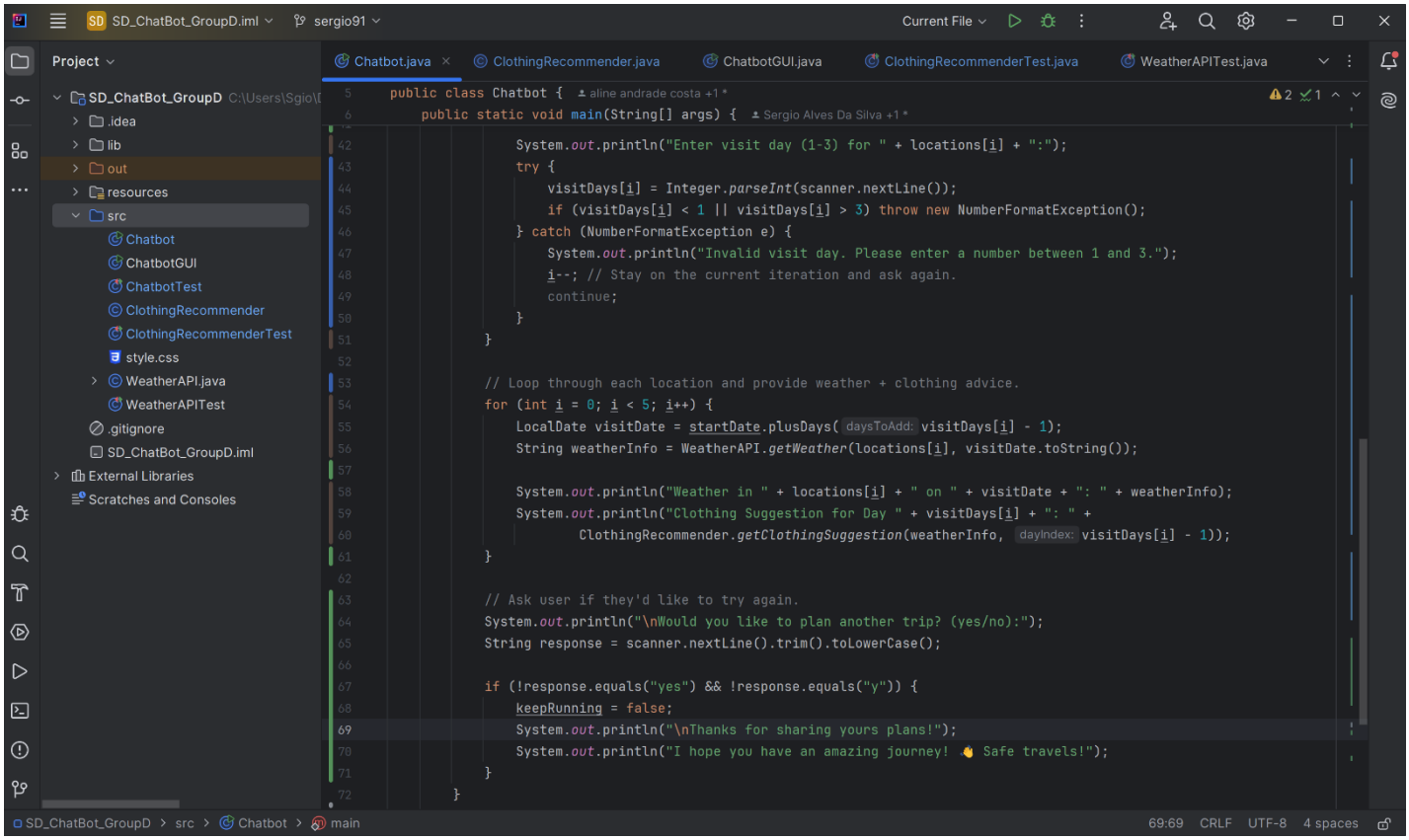
All scenarios in ClothingRecommenderTest were successful tested.



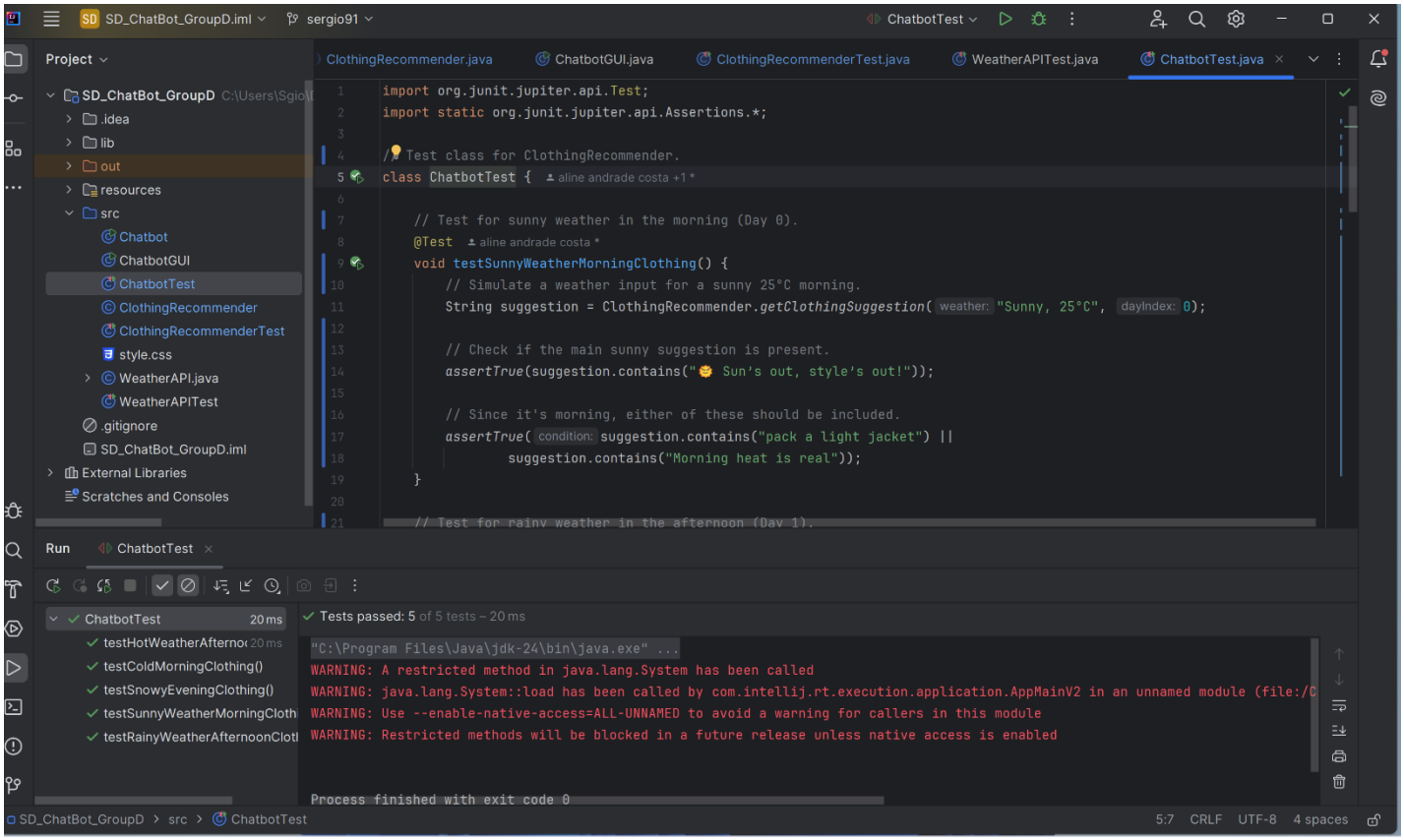
Used Version Control to add, commit and push all the modifications to the repository in Branch sergio91.



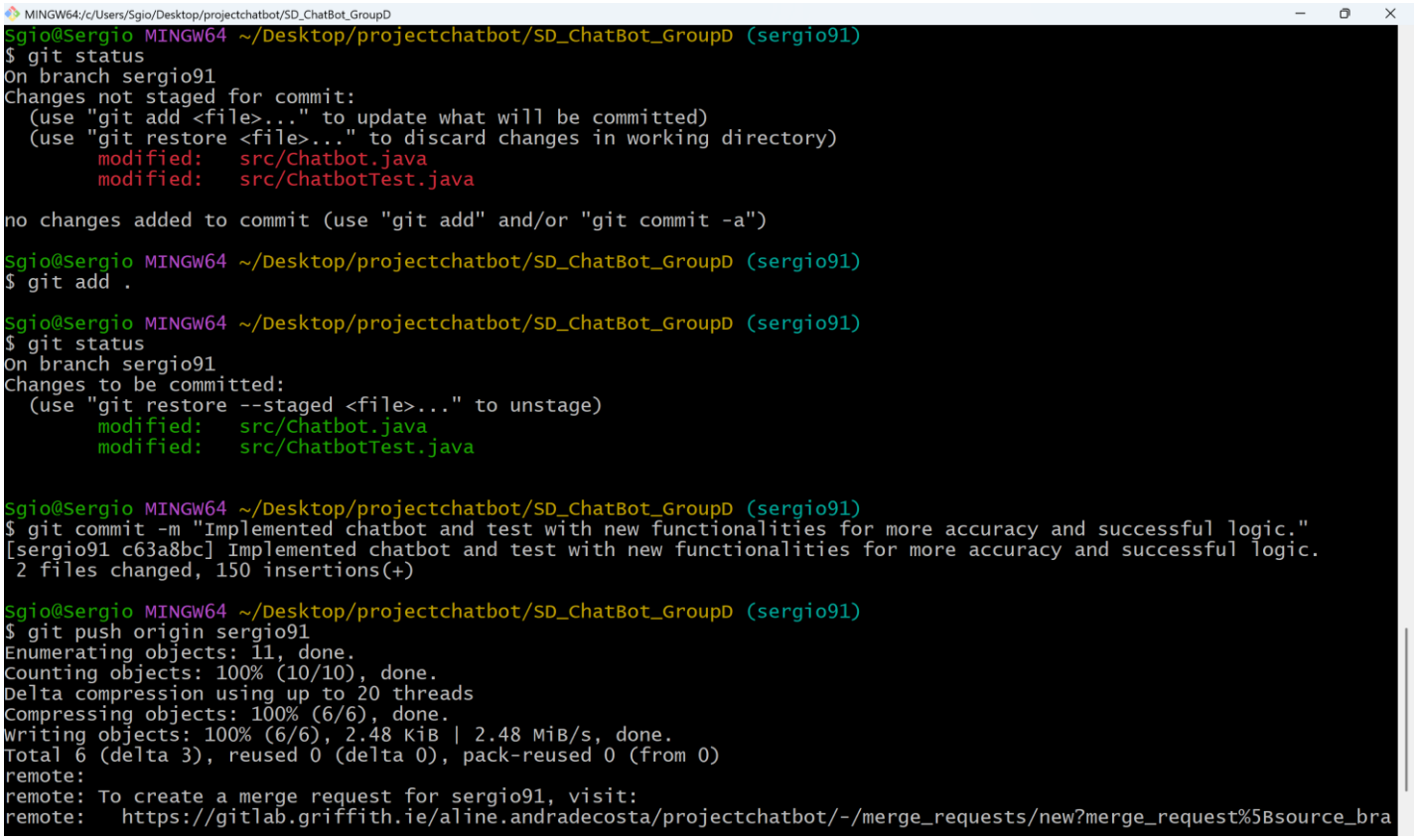
At the class Chatbot was added new functionalities, and text output improved to better interaction with the user.



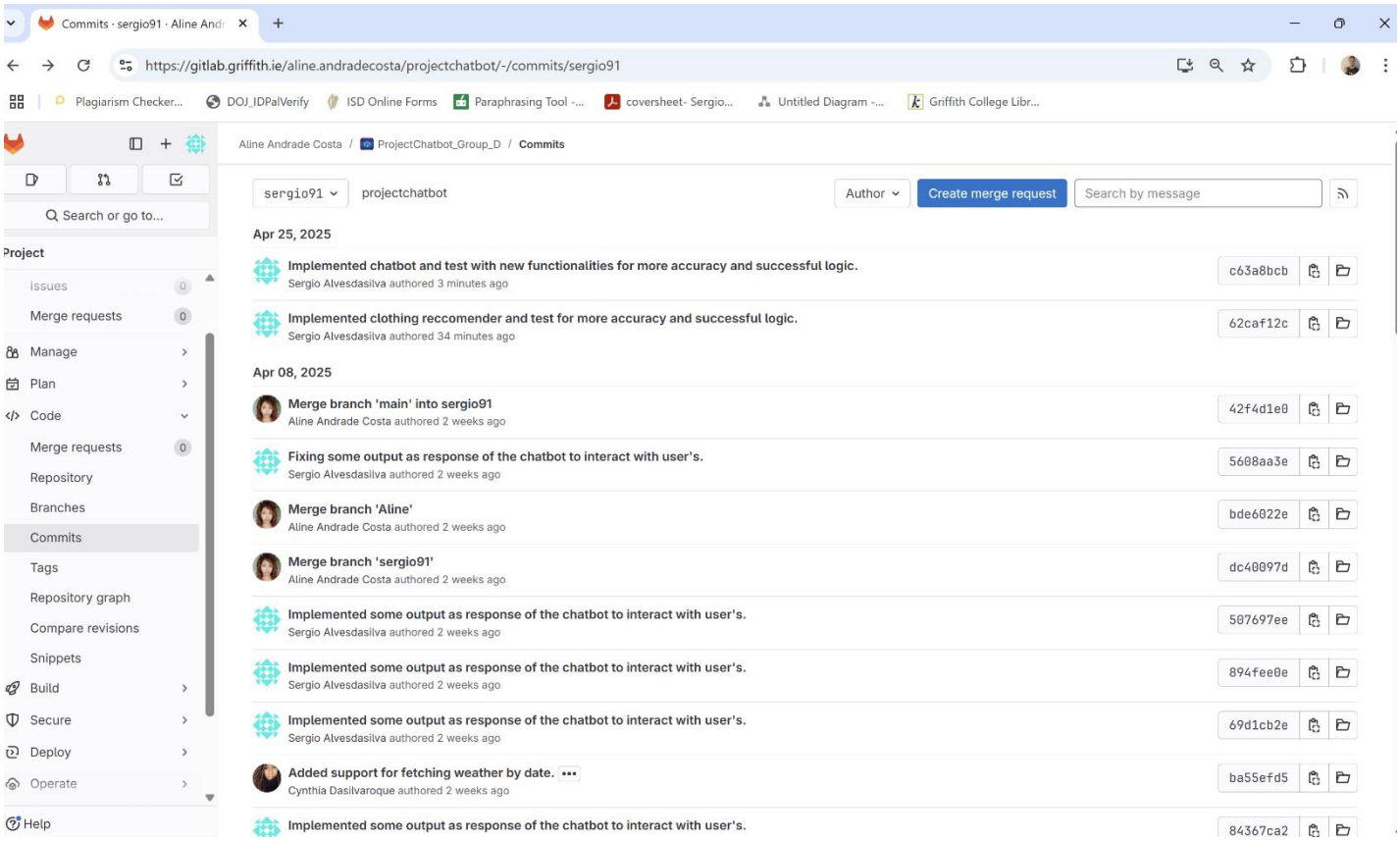
All functionalities were successful in ChatbotTest.



Used Version Control to add, commit and push all the modifications to the repository in Branch sergio91



All changes of the project was sent to the group repository.



- **Branches Tree**

→ main branch:

This will be the primary branch where the most recent stable code will reside. Each feature will be developed on separate branches and then merged into the main branch.

→ Aline's branch:

This branch will focus on developing the Graphical User Interface (GUI) for the chatbot, as well as the CSS styling for the interface.

Files included in this branch:

ChatbotGUI.java: Java code responsible for the GUI and chatbot interface.

style.css: Contains the CSS styles for the chatbot UI.

→ Sergio's branch:

This branch will contain the logic for the chatbot's interaction and the clothing recommendations based on weather conditions.

Files included in this branch:

ClothingRecommender.java: Contains the logic to generate clothing suggestions based on weather conditions.

→ Cynthia's t branch:

This branch will handle the implementation of the Weather API, which fetches the weather for a given location and date. It will also include unit tests for ensuring the functionality of the Weather API.

Files included in this branch:

WeatherAPI.java: Java code responsible for fetching weather data.

WeatherAPITest.java: Unit tests for validating the weather data fetching process.

10.4 Full Log Details

Cynthia – Weather API & Data Handling

- Improved API integration by handling real location lookups (geolocation API).

Location input is validated, ensuring weather data can be fetched using both location names and coordinates.

- Ensured error handling for incorrect locations.

Added better error handling to display friendly error messages when the user inputs an invalid location, ensuring proper feedback when the location is not found or is incorrect.

- Refined the 3-day weather forecast to include humidity, wind speed, and conditions.

Enhanced the weather data API response to include additional data points such as humidity, wind speed, and weather conditions for each forecasted day.

- Ensured API calls are optimized without exceeding rate limits.

Implemented request throttling and optimized API calls to ensure that they stay within rate limit restrictions, avoiding API blocking or delays.

Aline – JavaFX GUI Enhancements

- Improved the design of the GUI (better layout, colours, and fonts).

Redesigned the chatbot interface to enhance the user experience with better colour schemes, more readable fonts, and an intuitive layout.

- Added an icon/logo and improved UI responsiveness.

Added a logo to the interface and ensured that all UI elements are well-positioned and responsive across different screen sizes.

- Implemented a loading indicator while fetching data.

Included a progress indicator to notify users when the system is fetching data, enhancing the user experience by providing feedback during data loading.

- Allowed users to save and review past weather reports.

Added functionality for users to store and review previously fetched weather reports so that they can easily revisit them at a later time for reference.

Sergio – Chatbot Logic & Clothing Recommendations

- Improved clothing suggestions (consider time of day & weather patterns).

Refined the logic for clothing recommendations by taking into account the time of day (morning, afternoon, evening) and the prevailing weather conditions (rain, sunny, cold).

- Added a feature where users input their wardrobe to get personalized suggestions.

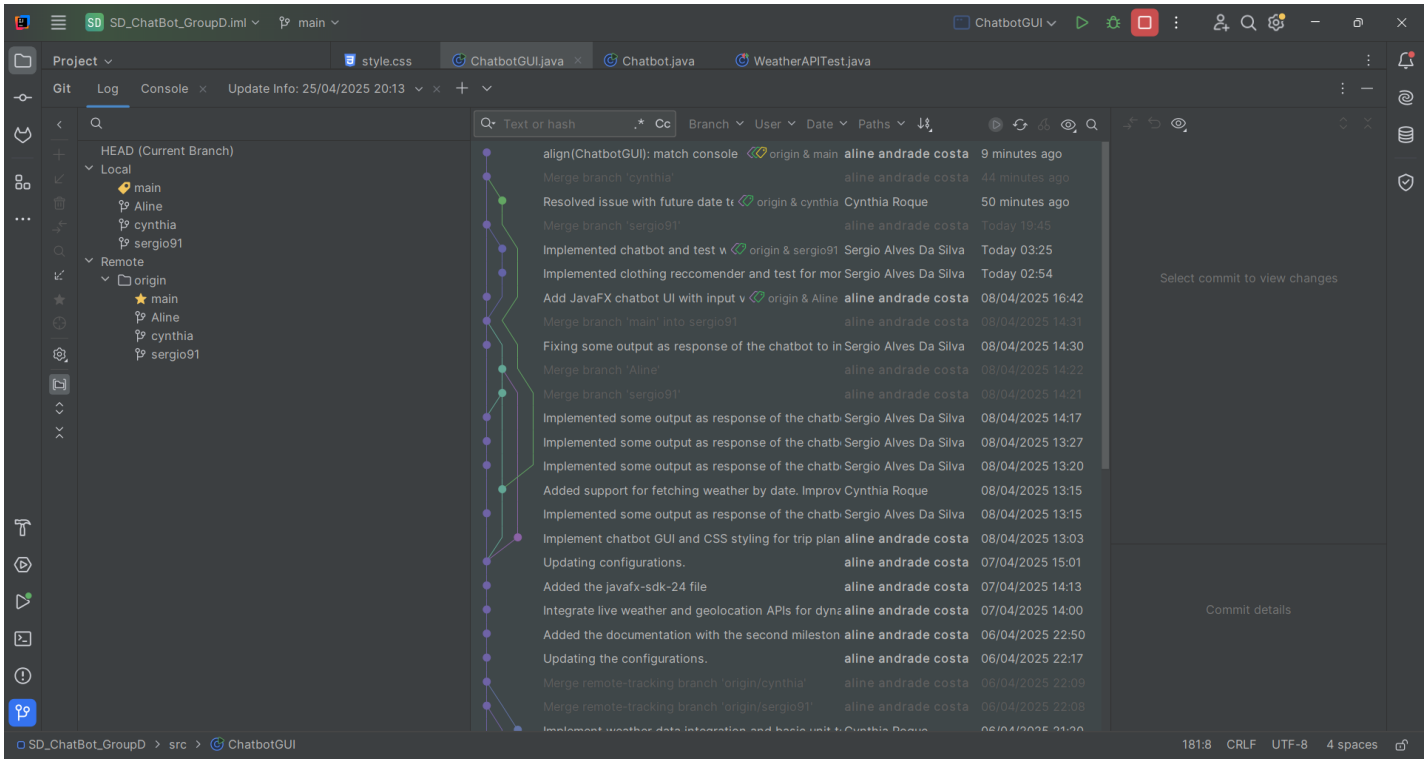
Developed a personalized recommendation feature, enabling users to input their wardrobe items (e.g., jackets, shoes) and receive customized clothing suggestions based on weather conditions and their wardrobe.

- Implemented a speech-to-text feature (optional, for accessibility).

Incorporated speech-to-text functionality to enable voice input, making the chatbot more accessible and providing a hands-free way to interact with the system.

- Wrote unit tests for all chatbot functionalities.

Created comprehensive unit tests for major chatbot functionalities, such as location input, weather-based clothing recommendations, and wardrobe-based suggestions, ensuring reliable and accurate behavior of the chatbot under various conditions.

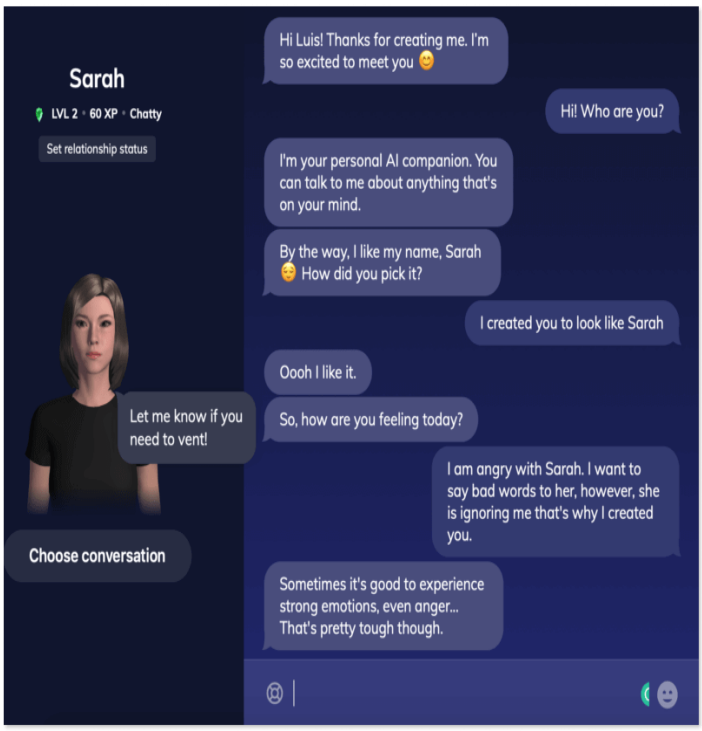


11. Appendix

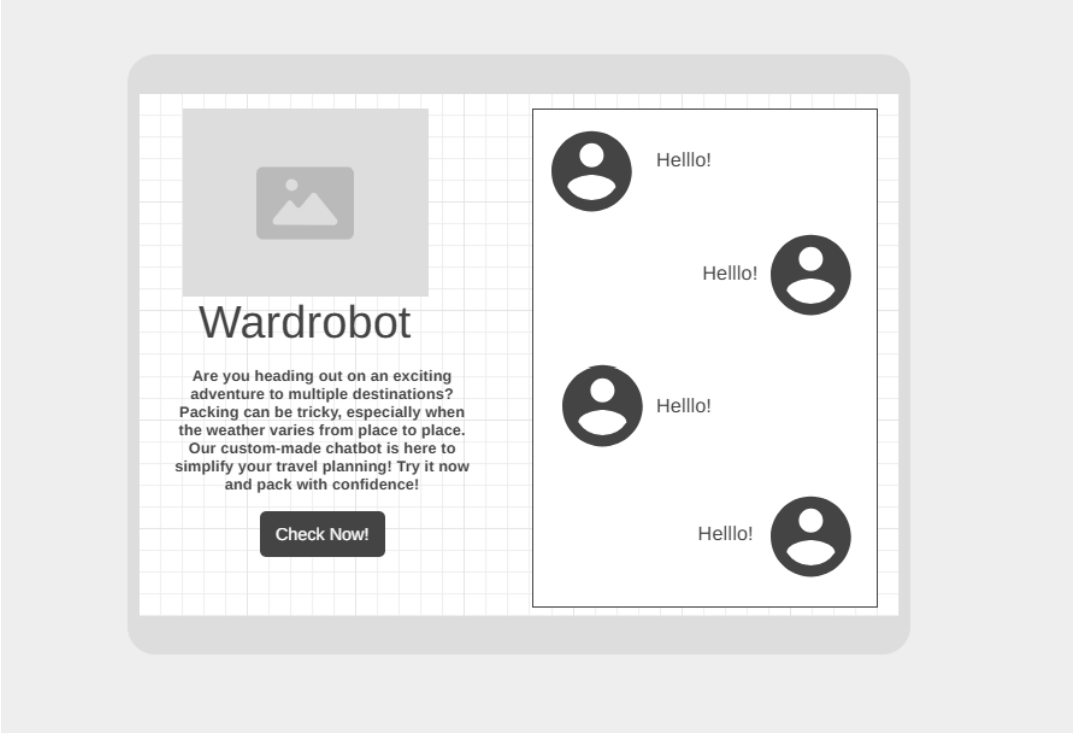
Reference idea

2.

3.



Wireframe (Layout)



12.Bibliography

1. JSON (org.json) Stleary, J., 2021. JSON-java. Available at: https://stleary.github.io/JSON-java/ [Accessed 18 March 2025].
2. JUnit (JUnit 5) JUnit, 2025. JUnit 5 User Guide. Available at: https://junit.org/junit5/ [Accessed 18 March 2025].
3. Nicholson, A. and Preston, D., 2016. Designing Conversational Interfaces: How to Build Chatbots for Customer Service and Support. O'Reilly Media. Available at: https://www.oreilly.com/library/view/designing-conversational-interfaces/9781491955307/ [Accessed 18 March 2025].
4. Shawar, B.A. and Atwell, E., 2007. Chatbots: Are they really useful? In: International Conference on Information Technology: New Generations. IEEE. pp. 275-280. Available at: https://ieeexplore.ieee.org/document/4212514 [Accessed 18 March 2025].
5. McTear, M., 2017. Conversational AI: Chatbots and the future of human-computer interaction. Springer. Available at: https://link.springer.com/book/10.1007/978-3-319-56138-3 [Accessed 18 March 2025].
6. (Open-Meteo, n.d.) Open-Meteo (n.d.) *Open-Meteo weather API*. Available at:https://registry.opendata.aws/open-meteo/ (Accessed: 18 March 2025).
7. Amir Shevat**.** (2017). *Designing Bots: Creating Conversational Experiences*. [online] Griffith College Moodle. Available at: <https://moodle.griffith.ie/course/view.php?id=2540 > [Accessed 21 March 2025].

8. Aloa. (n.d.). *How to Build a Chatbot*. [online] Available at: https://aloa.co/blog/how-to-build-a-chatbot [Accessed 21 March 2025].
9. HowToDoInJava. (n.d.). *Java AIML Chatbot Example*. [online] Available at: https://howtodoinjava.com/java/library/java-aiml-chatbot-example/#create-chatbot [Accessed 21 March 2025].
10. W3Schools, n.d. *Java Packages*. [online] Available at: https://www.w3schools.com/java/java_packages.asp [Accessed 21 March 2025].
11. App EWA (n.d.) *AppEWA – AI Chatbot Solutions*. Available at: https://appewa.com/ (Accessed: 21 March 2025).

Images References

1. Deep-Image.ai. (2025). *AI-generated image using Deep-Image.ai's AI Wardrobot*. Retrieved April 02, 2025, from https://deep-image.ai/app/tools/generator Deep Image+2Deep Image+2.
2. Lithium Lab Pte Ltd. (2025). *EWA: Learn English & Spanish* (Version app) [Mobile app]. Google Play Store. https://play.google.com/store/apps/details?id=com.ewa.ewaapp
3. *Figure 3. Screenshot of a chat with Replika (personal communication, April 02, 2025).https://research.aimultiple.com/top-chatbot-success/*