

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	<input checked="" type="checkbox"/>		Part-time		
Lecturer Name:	Haseeb Younis/ Muhammad Shoaib					
Assignment Title:	Project Final Documentation					
Date due:	27/04/2025					
Date submitted:						
<p>Plagiarism disclaimer:</p> <p><i>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></p> <p><i>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.</i></p> <p>Signed: <u>Aline, Cynthia, Sergio</u> Date: 21/03/2025</p>						

Software Development 2
BSCH-SD2
Chatbot Project
Date

Table of Contents

1. Versioning Approach.....	4
2. Development Process.....	4
3. UI Implementation.....	5
4. Rest API.....	5
4.1 Rest API Implementation	5
5. Weather API	6
6. External Packages	6
7. Project Setup	7
8. Milestone 1	11
8.1 Goals.....	11
8.2 Junit Tests.....	12
8.3 Commit Logs	12
8.4 Full Log Details.....	25
9. Milestone 2	25
9.1 Goals.....	25
9.2 Junit Tests Integration	25
9.3 Commit List & Branches Tree	25
9.4 Full Log Details.....	25
10. Milestone 3.....	25
10.1 Goals.....	25
10.2 Junit Tests Integration	25
10.3 Commit List & Branches Tree	25
10.4 Full Log Details.....	25
11. Appendix.....	25
12. Bibliography.....	26

1. Versioning Approach

In this project, Git is utilised 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.



Name: Wadrobot.

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.
- Recognise 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!

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.

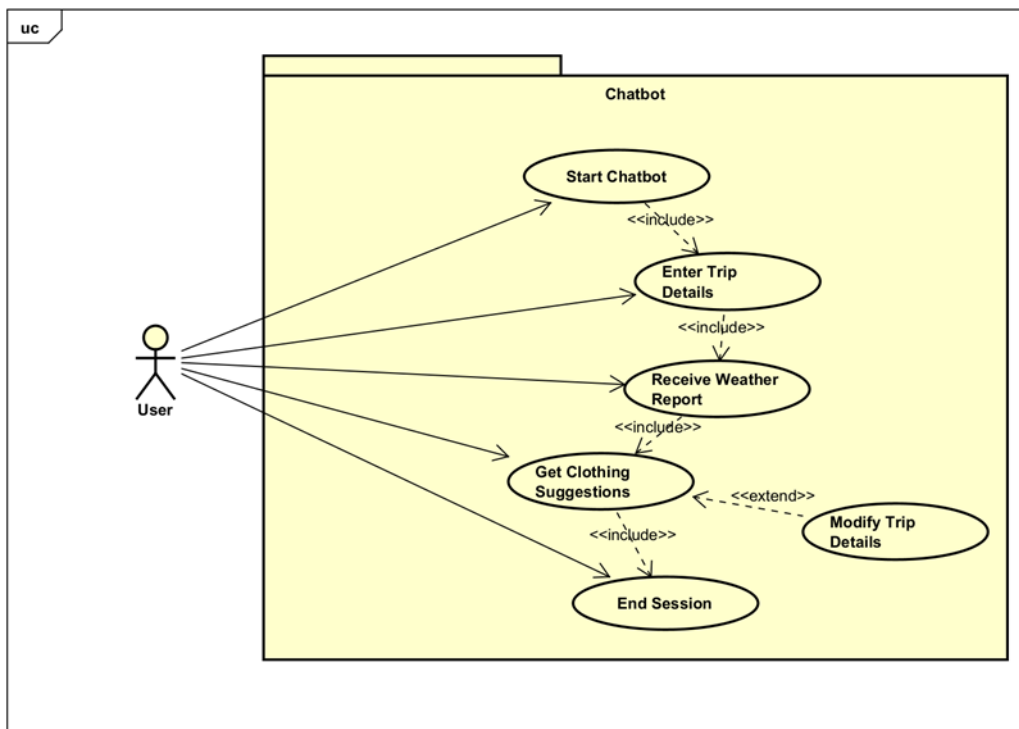
3. User exits before completing the process.

- 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



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:

```
MINGW64:/c/Users/lili_/OneDrive/Área de Trabalho/Software Development 2/...
lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project
$ git clone https://gitlab.griffith.ie/aline.andradecosta/projectchatbot.git
Cloning into 'projectchatbot'...
remote: HTTP Basic: Access denied. If a password was provided for Git authentication, the password was incorrect or you're required to use a token instead of a password. If a token was provided, it was either incorrect, expired, or improperly scoped. See https://gitlab.griffith.ie/help/topics/git/troubleshooting_git.md#error-on-git-fetch-http-basic-access-denied
fatal: Authentication failed for 'https://gitlab.griffith.ie/aline.andradecosta/projectchatbot.git/'

lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project
$ 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 information
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

lili_@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project
$
```

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'.

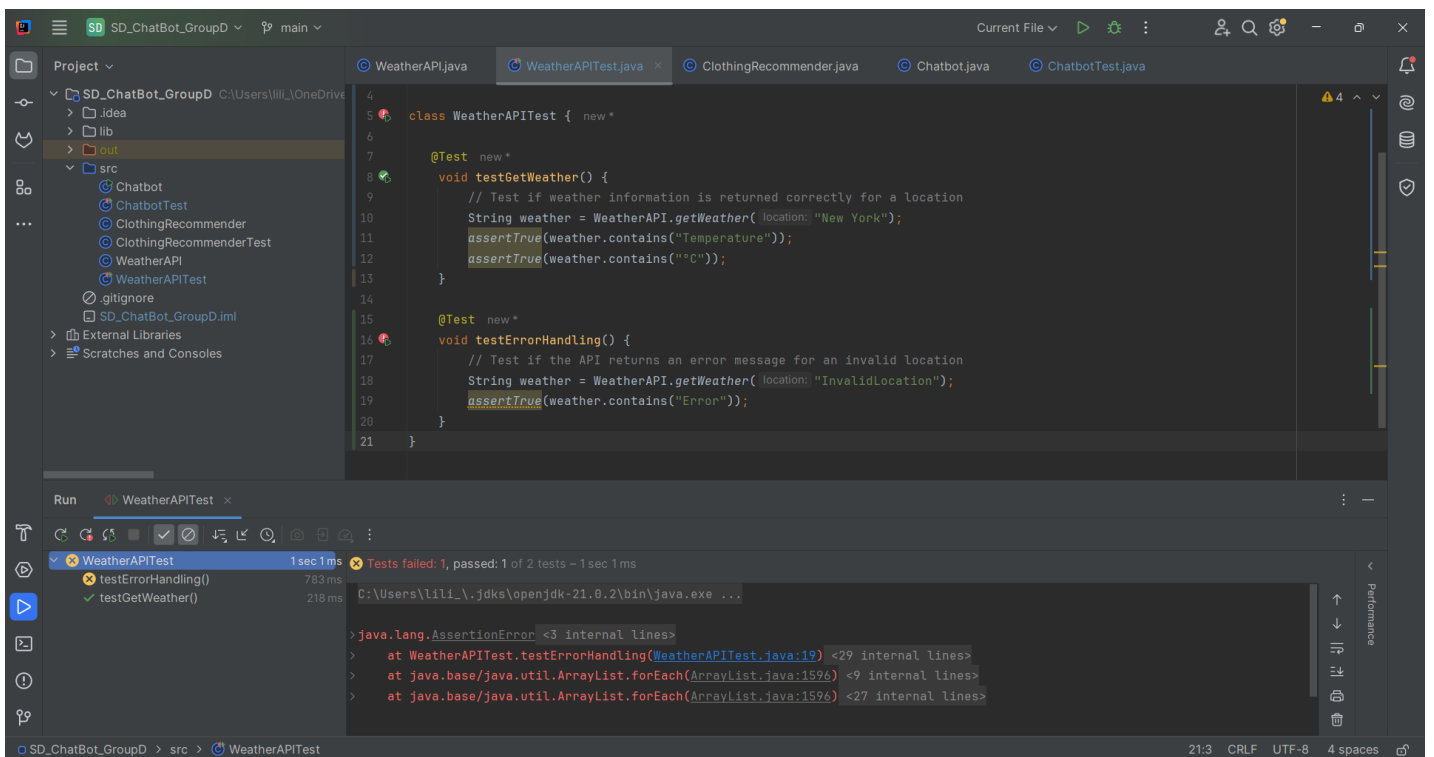
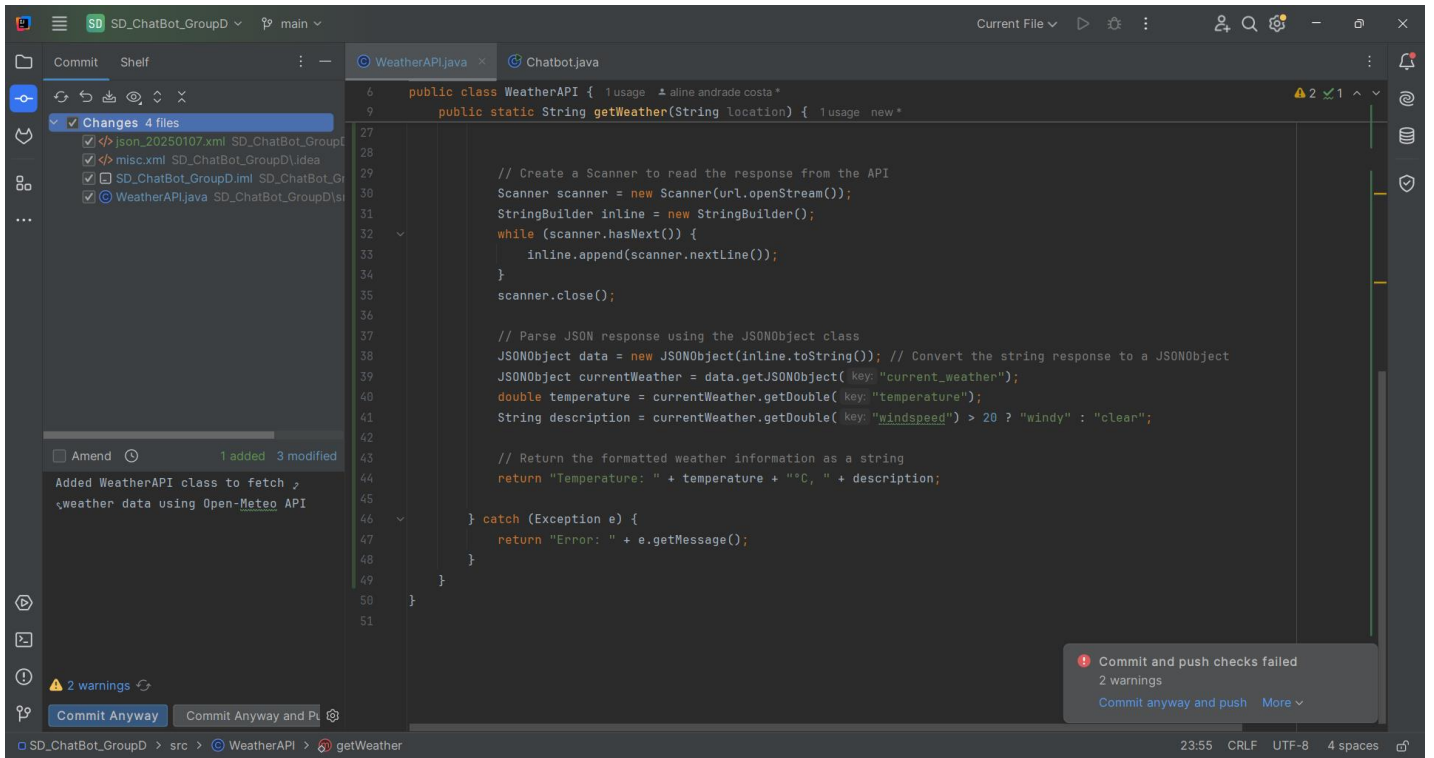
lili@Aline MINGW64 ~/OneDrive/Área de Trabalho/Software Development 2/Project/projectchatbot/Documents (main)
$

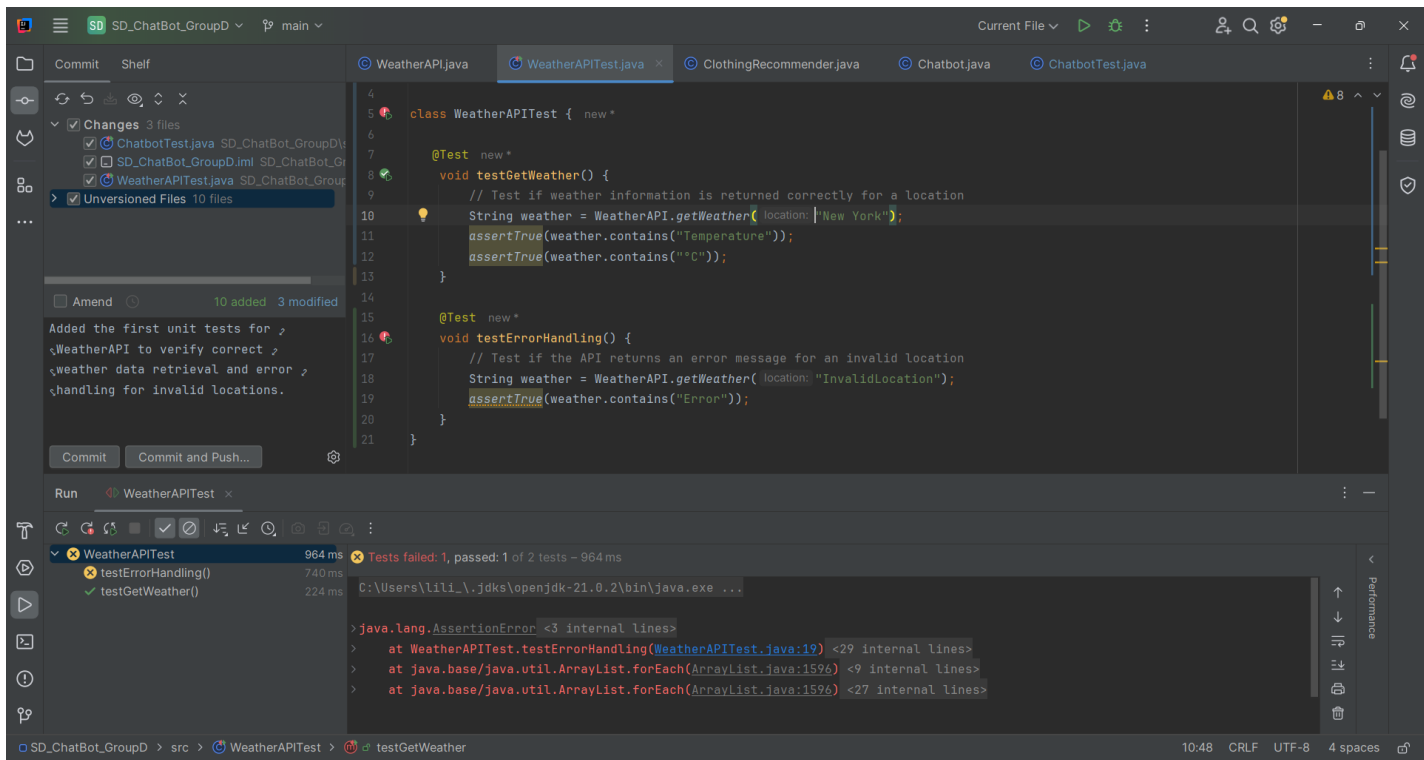
```

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
projectchatbot/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
projectchatbot/SD_ChatBot_GroupD/src (main)
$
```





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

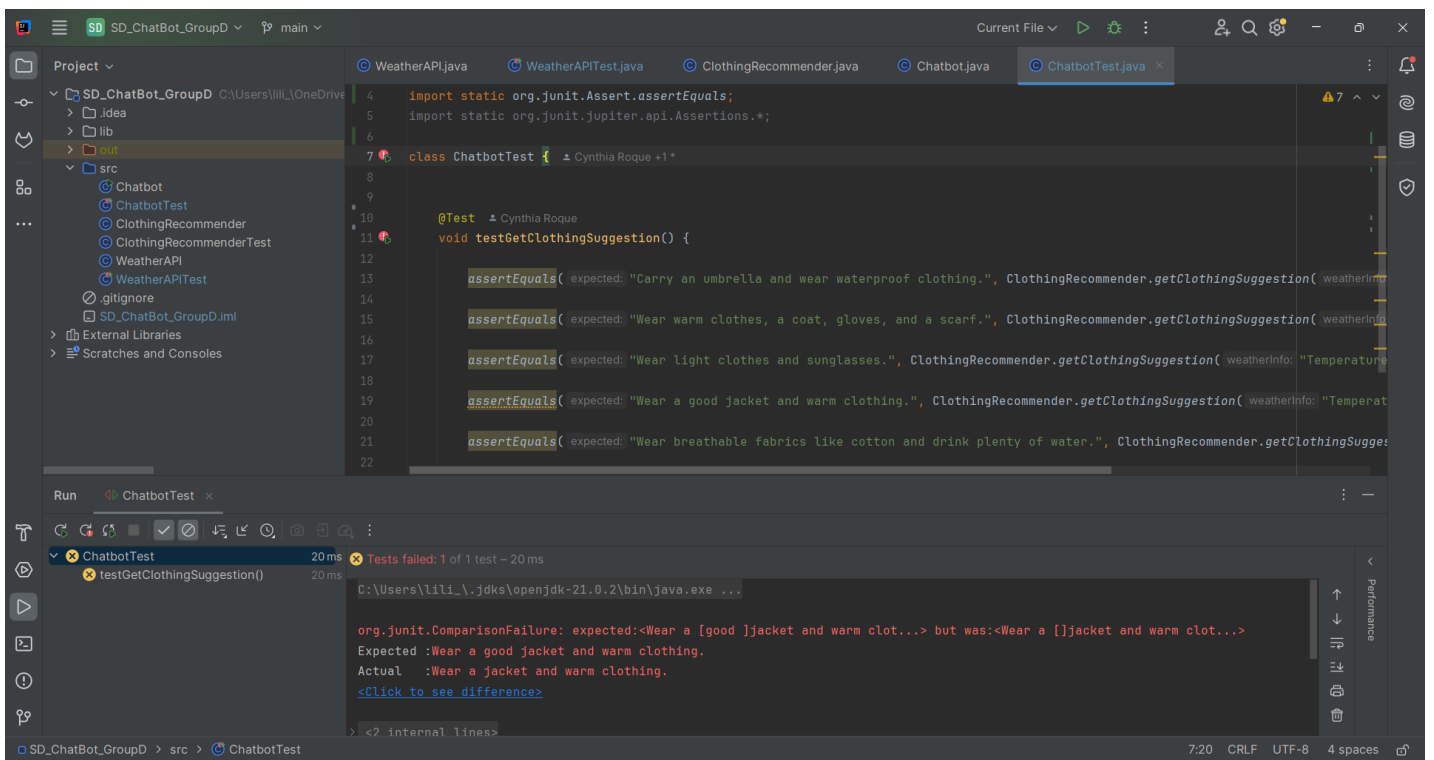
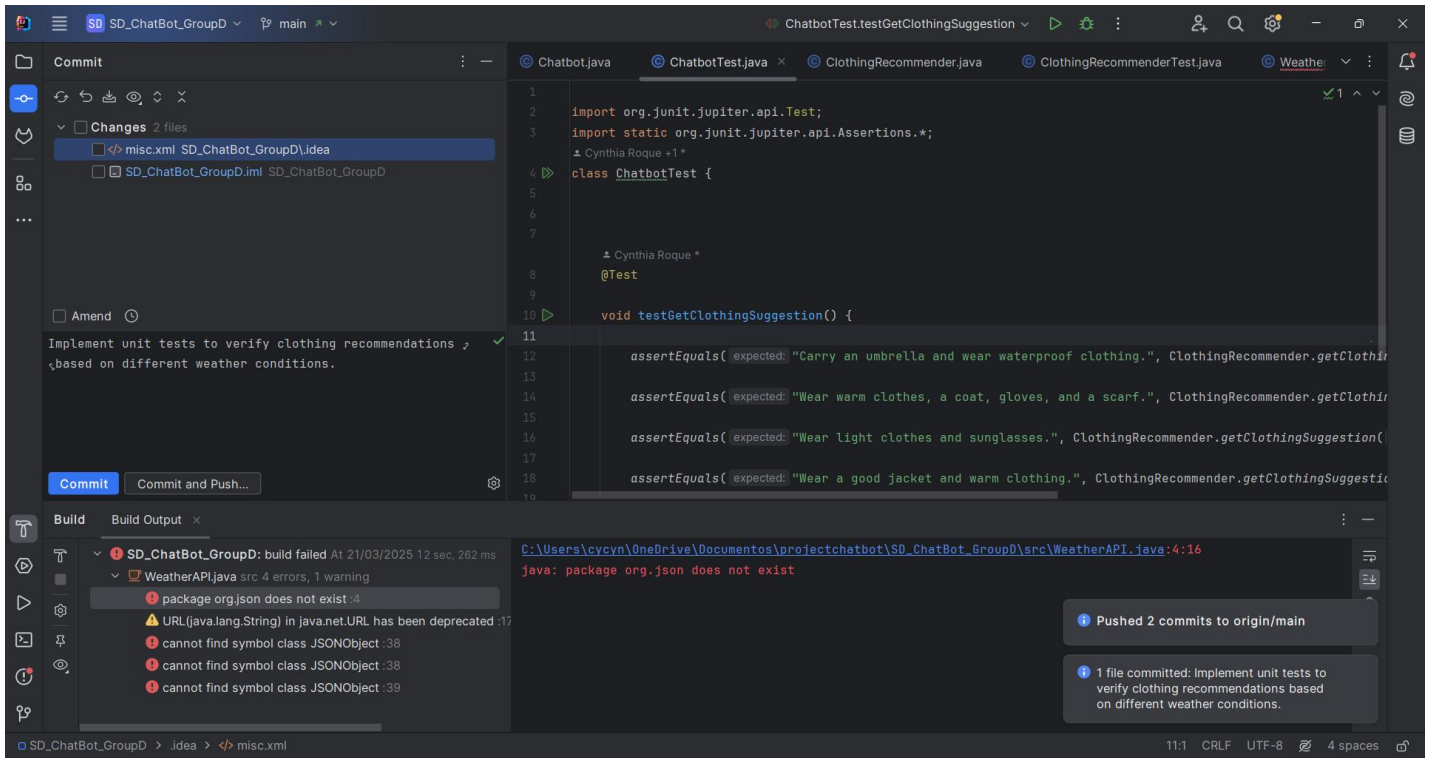
cycyn@Cyntha MINGW64 ~/OneDrive/Documentos/projectchatbot (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (4/4), 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)
$ |
```

```
1 import java.util.Scanner;
2
3
4 public class Chatbot {
5
6
7     // Cynthia Roque <1
8     public static void main(String[] args) {
9
10         Scanner scanner = new Scanner(System.in); // Create scanner object for user input
11
12         // Welcome message
13         System.out.println("Welcome to the Trip Clothing Planner Chatbot!");
14
15         // Ask for the number of locations
16         System.out.println("Enter the number of locations you will visit: ");
17         int numLocations = scanner.nextInt(); // Get the number of locations
18         scanner.nextLine(); // Clear the buffer
19
20         // Loop through each location
21         for (int i = 1; i <= numLocations; i++) {
22
23             // Ask for location name
24             System.out.println("Enter location " + i + ":");
25             String location = scanner.nextLine();
26
27             // Fetch weather information for the location
28             String weatherInfo = WeatherAPI.getWeather(location); // Fetch weather data from the API
29
30             // Display the weather information
31             System.out.println("Weather in " + location + ": " + weatherInfo);
32
33             // Suggest clothing based on weather data
34             System.out.println("Clothing Suggestion: " + ClothingRecommender.getClothingSuggestion(weatherInfo));
35         }
36
37         scanner.close(); // Close the scanner to free up resources
38     }
39 }
```

```
MINGW64/c:/Users/cycyn/OneDrive/Documents/projectchatbot
cycyn@Cynthia MINGW64 ~/OneDrive/Documents/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@Cynthia MINGW64 ~/OneDrive/Documents/projectchatbot (main)
$
```



Sergio – Commit logs

Cloning the repository:

```

Sgio@Sergio MINGW64 ~/Desktop
$ 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 information
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 24 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (24/24), 242.13 KiB | 1.12 MiB/s, done.

Sgio@Sergio MINGW64 ~/Desktop
$

```

Git Pull for all implementations made in the project:

```

SD_ChatBot_GroupD/.idea/.name
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:

```

1 public class ClothingRecommender {
2
3
4     public static String getClothingSuggestion(String weatherInfo) {
5
6         if (weatherInfo.contains("Error")) return "Unable to determine clothing suggestion.";
7
8         if (weatherInfo.contains("rain")) return "Carry an umbrella and wear waterproof clothing.";
9
10        if (weatherInfo.contains("snow")) return "Wear warm clothes, a coat, gloves, and a scarf.";
11
12        if (weatherInfo.contains("clear")) return "Wear light clothes and sunglasses.";
13
14        if (weatherInfo.contains("cold")) return "Wear a jacket and warm clothing.";
15
16        if (weatherInfo.contains("hot")) return "Wear breathable fabrics like cotton and drink plenty of water.";
17
18        if (weatherInfo.contains("windy")) return "Wear a windbreaker or a sturdy jacket.";
19
20        if (weatherInfo.contains("humid")) return "Wear loose, moisture-wicking clothes to stay comfortable.";
21
22        return "Wear comfortable clothing suitable for mild weather.";
23
24    }
25
26 }
27
28
29

```

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 -> 43024 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

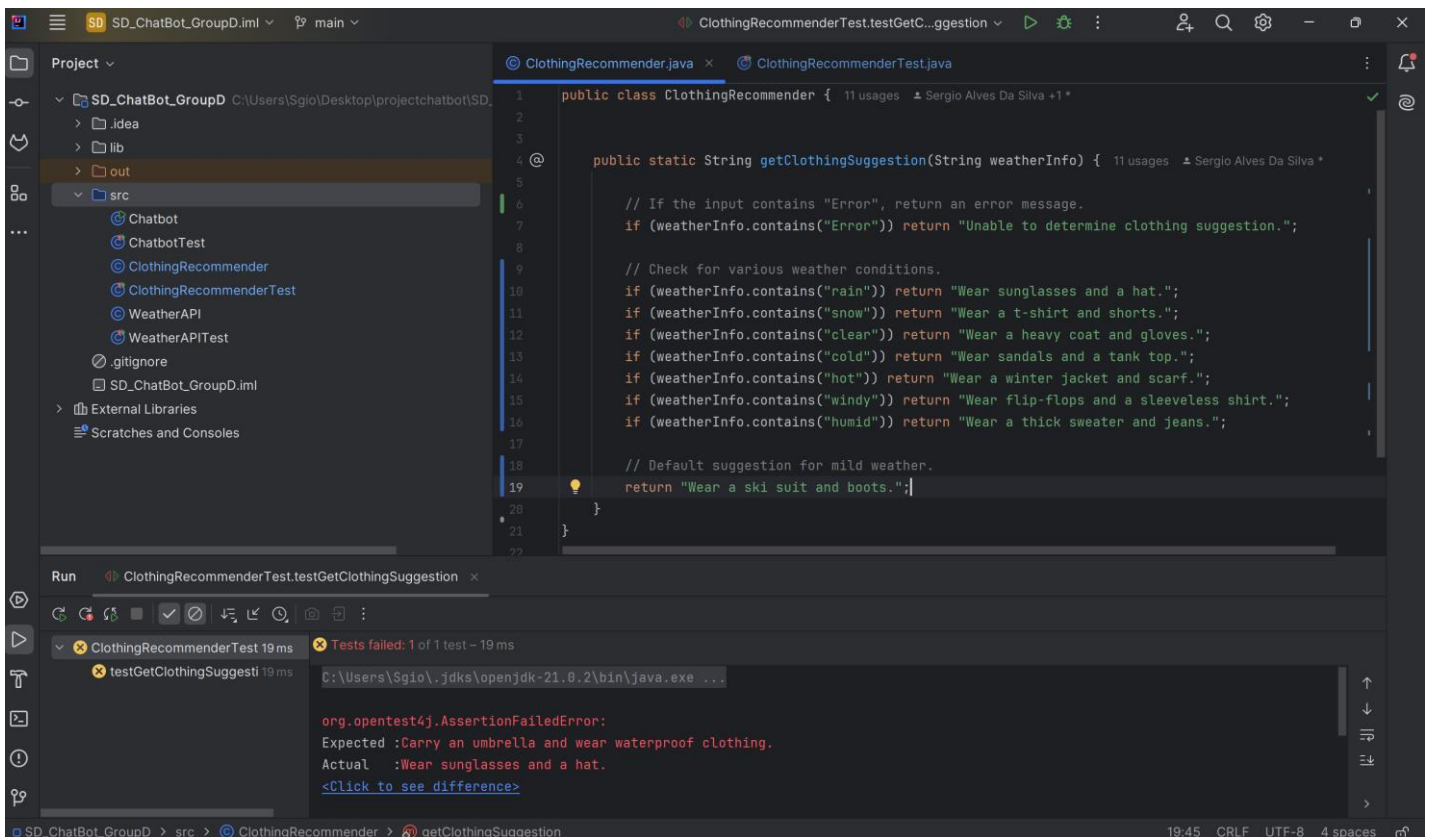
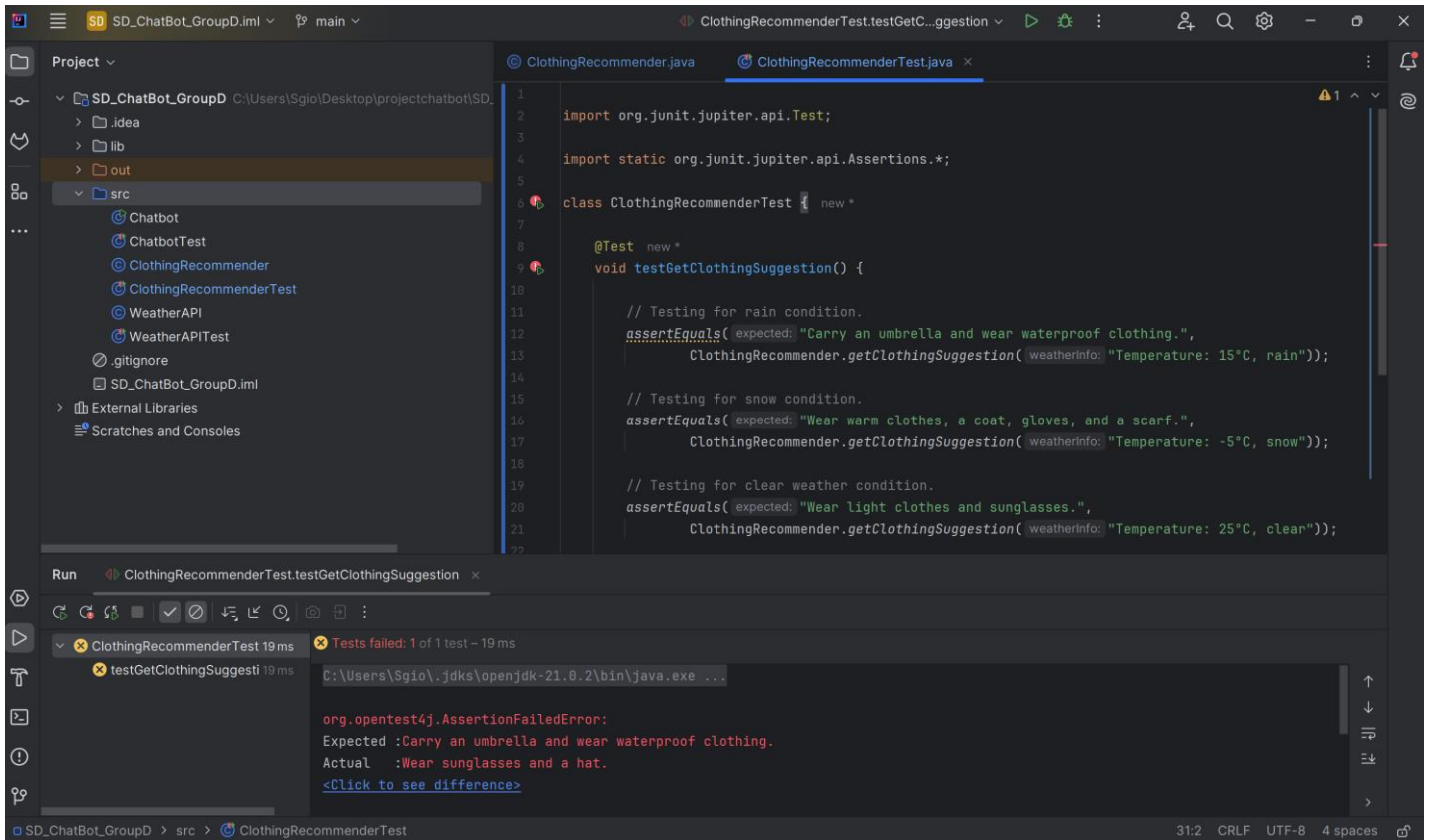
Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (main)
$ git commit -m "Changes in class ClothingRecomender and create a junit test for the same class where method failed"
[main f24c6c5] Changes in class ClothingRecomender and create a junit test for the same class where method failed
 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

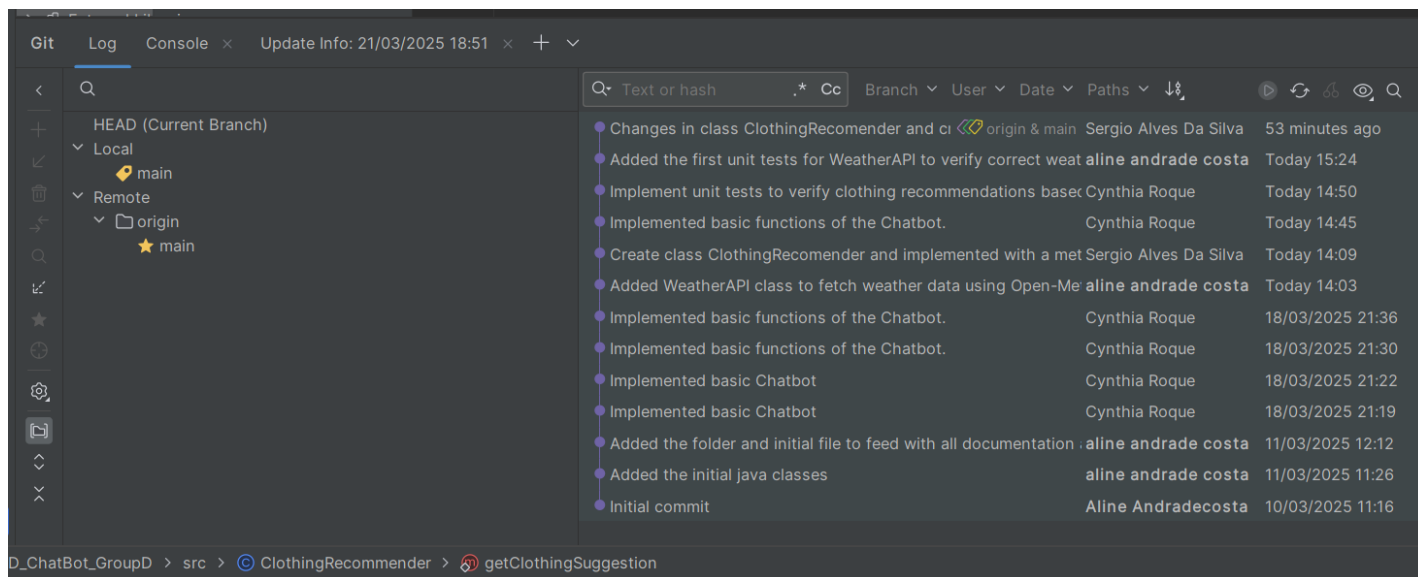
Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (main)
$ |

```

Alterations in java interface:



8.4 Full Log Details



9. Milestone 2

9.1 Goals

9.2 Junit Tests Integration

9.3 Commit List & Branches Tree

9.4 Full Log Details

10. Milestone 3

10.1 Goals

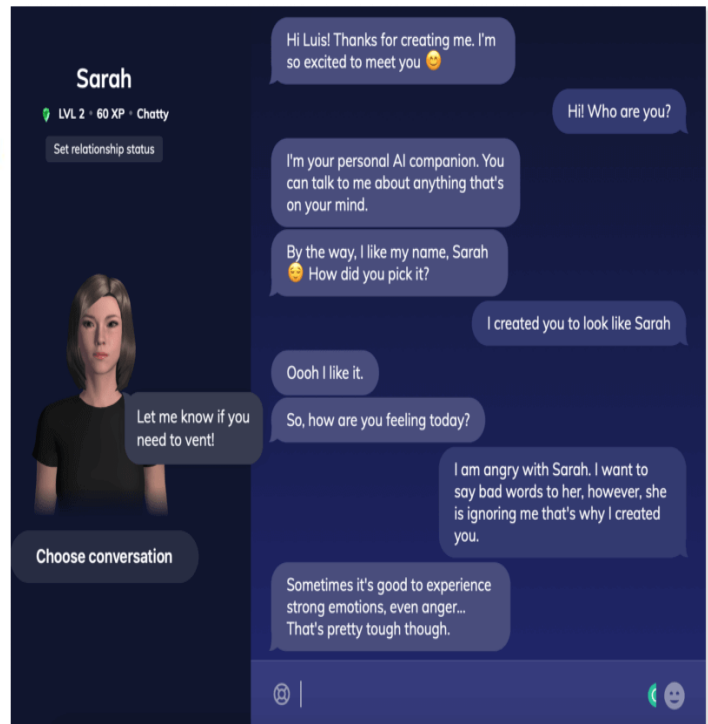
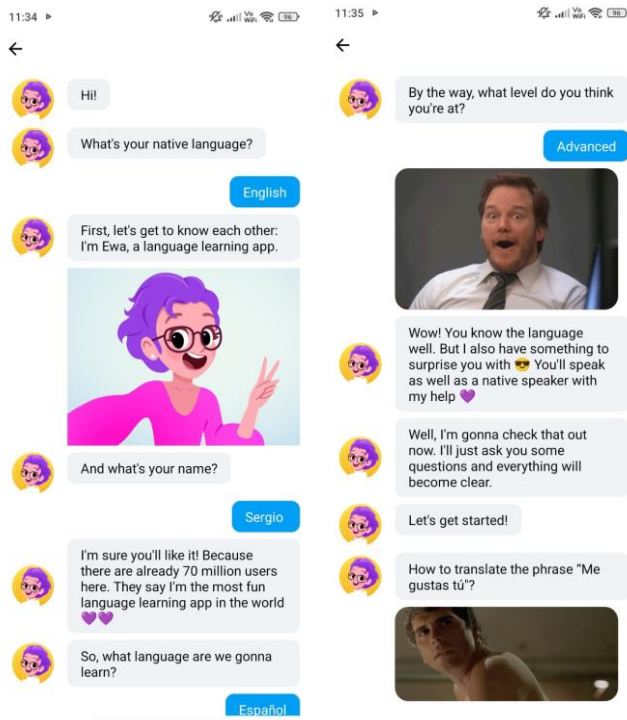
10.2 Junit Tests Integration

10.3 Commit List & Branches Tree

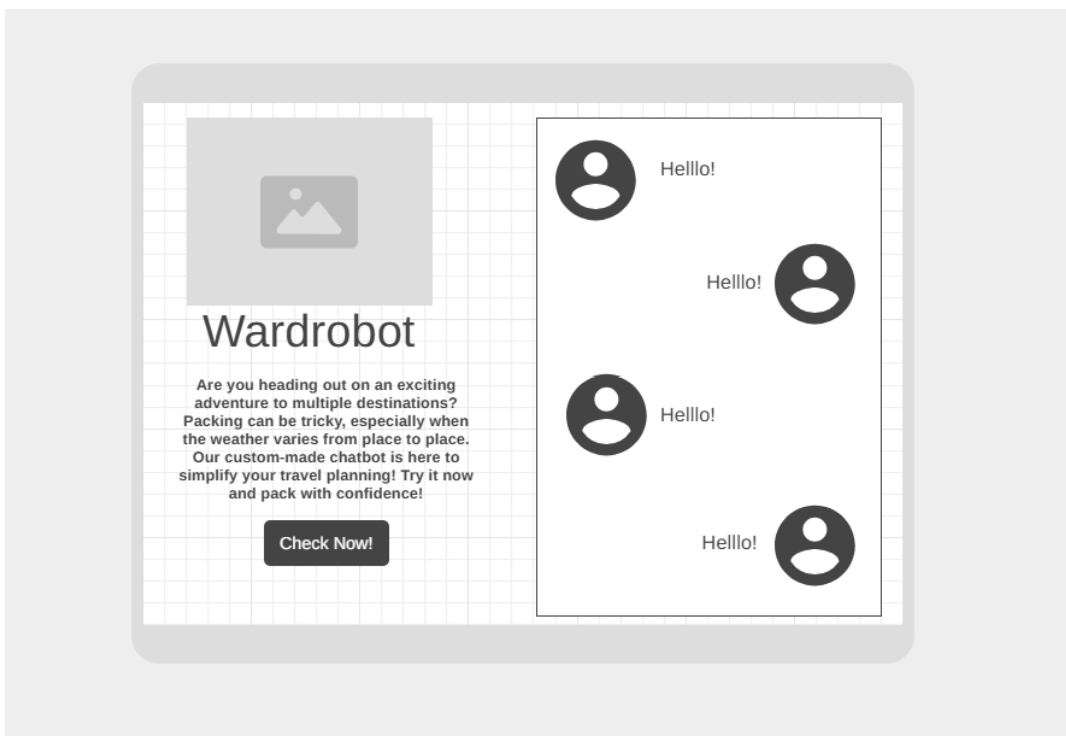
10.4 Full Log Details

11. Appendix

Reference idea



Wireframe (Layout)



12.Bibliography

JSON (org.json)

Stleary, J., 2021. JSON-java. Available at: <https://stleary.github.io/JSON-java/> [Accessed 18 March 2025].

JUnit (JUnit 5)

JUnit, 2025. JUnit 5 User Guide. Available at: <https://junit.org/junit5/> [Accessed 18 March 2025].

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].

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].

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].

(Open-Meteo, n.d.)

Open-Meteo (n.d.) *Open-Meteo weather API*. Available at: <https://registry.opendata.aws/open-meteo/> (Accessed: 18 March 2025).

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].

Aloa. (n.d.). *How to Build a Chatbot*. [online] Available at: <https://aloe.co/blog/how-to-build-a-chatbot> [Accessed 21 March 2025].

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].

W3Schools, n.d. *Java Packages*. [online] Available at: https://www.w3schools.com/java/java_packages.asp [Accessed 21 March 2025].