

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:					

**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**

## 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 .....	12
8.1	Goals.....	12
8.2	Junit Tests.....	13
8.3	Commit Logs .....	13
8.4	Full Log Details.....	26
9.	Milestone 2 .....	26
9.1	Goals.....	26
9.2	Junit Tests Integration .....	26
9.3	Commit List & Branches Tree .....	27
9.4	Full Log Details.....	34
10.	Milestone 3.....	36
10.1	Goals.....	36
10.2	Junit Tests Integration .....	36
10.3	Commit List & Branches Tree .....	36
10.4	Full Log Details.....	36
11.	Appendix .....	36
12.	Bibliography.....	37

## 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 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.

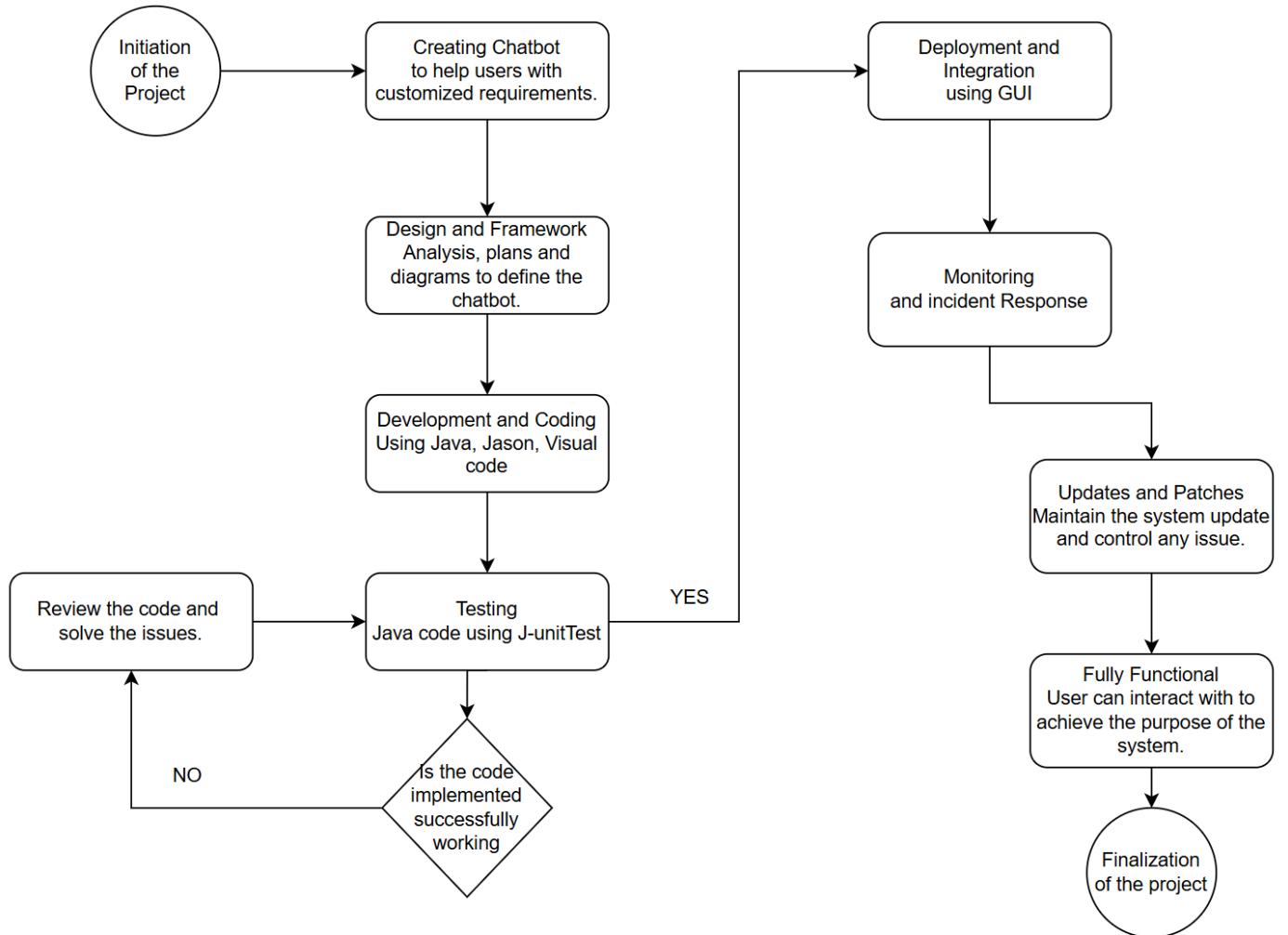
## 3. User exits before completing the process.

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

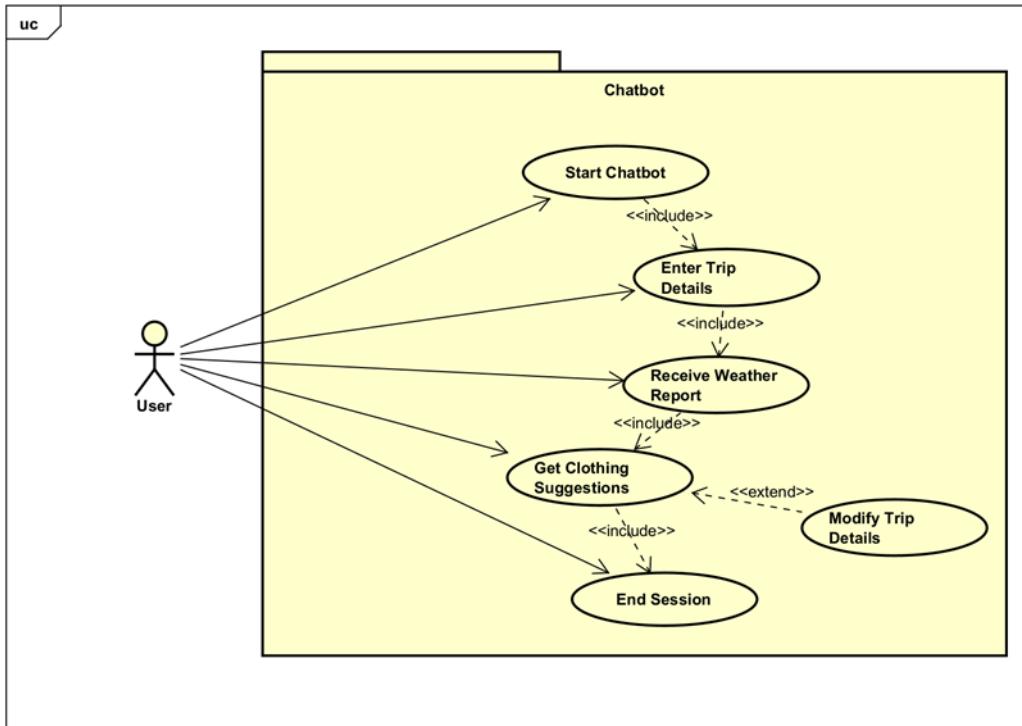
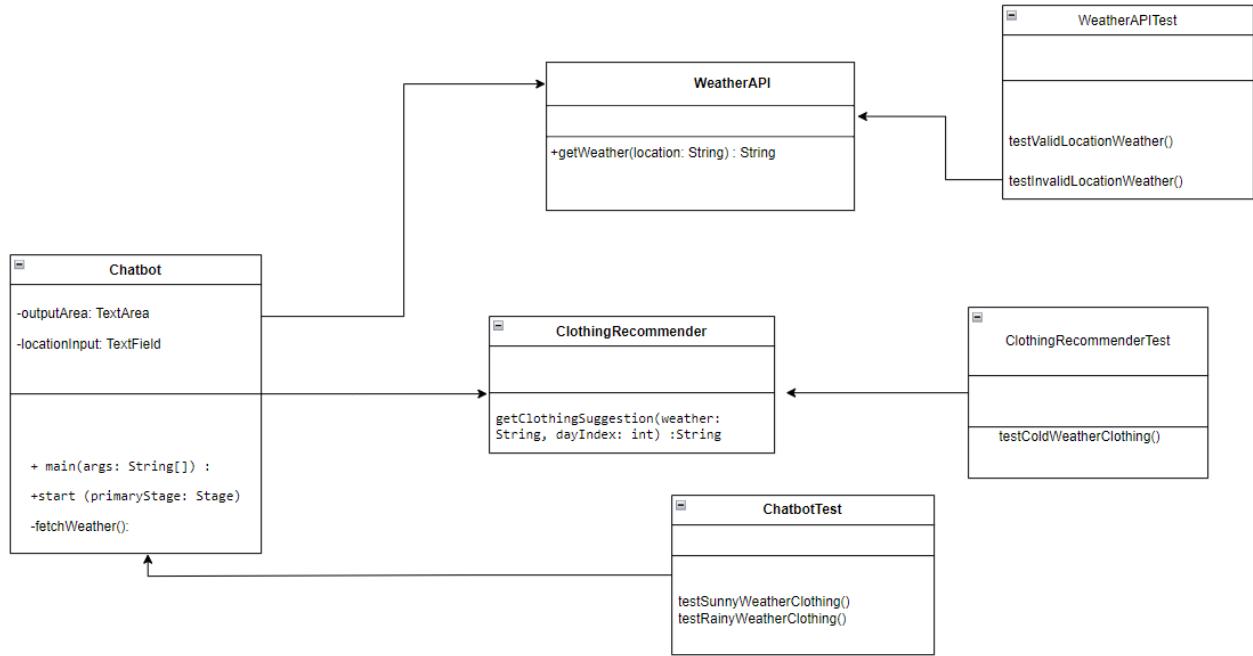
### Postconditions:

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

### FlowChart Diagram (Cycle of the Project)



### 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/... ━ ━ ━ X
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/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: .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/Project/projectchatbot/SD\_ChatBot\_GroupD/src (main) — □ X

```

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)
$
```

The screenshot shows a Java development environment with the following details:

- File Explorer:** Shows a project named "SD\_ChatBot\_GroupD" with files like "JSON\_20250107.xml", "WeatherAPI.java", and "WeatherAPITest.java".
- Code Editor:** Displays the content of `WeatherAPI.java`. The code fetches weather data from an API using a Scanner and JSONObject.
- Status Bar:** Shows "2 warnings" and a message: "Commit and push checks failed 2 warnings Commit anyway and push More".
- Bottom:** Shows the file path "SD\_ChatBot\_GroupD > src > WeatherAPI > getWeather" and the current time "23:55".

The screenshot shows a Java development environment with the following details:

- File Explorer:** Shows a project structure with "src" containing "WeatherAPITest.java" and other test classes.
- Code Editor:** Displays the content of `WeatherAPITest.java`, which contains two test methods: `testGetWeather()` and `testErrorHandling()`.
- Run Interface:** Shows the results of running the tests. `testGetWeather()` passed (1 sec 1 ms), and `testErrorHandling()` failed (783 ms). The failure output shows an `AssertionError` was thrown.
- Status Bar:** Shows the file path "SD\_ChatBot\_GroupD > src > WeatherAPITest" and the current time "21:3".

The screenshot shows a Java development environment with the following interface elements:

- Top Bar:** Shows the project name "SD\_ChatBot\_GroupD" and the file "main".
- Left Sidebar:** Contains a "Commit" section with "Shelf" and "Changes" (3 files: ChatbotTest.java, SD\_ChatBot\_GroupDImpl, WeatherAPITest.java) and an "Unversioned Files" section (10 files).
- Central Area:** A code editor window titled "WeatherAPITest.java" with the following code:

```
4
5  class WeatherAPITest { new*
6
7      @Test new*
8          void testGetWeather() {
9              // Test if weather information is returned correctly for a location
10             String weather = WeatherAPI.getWeather(location: "New York");
11             assertTrue(weather.contains("Temperature"));
12             assertTrue(weather.contains("°C"));
13         }
14
15     @Test new*
16         void testErrorHandling() {
17             // Test if the API returns an error message for an invalid location
18             String weather = WeatherAPI.getWeather(location: "InvalidLocation");
19             assertTrue(weather.contains("Error"));
20         }
21     }
```

- Bottom Left:** Buttons for "Amend", "Commit", and "Commit and Push...".
- Bottom Middle:** A "Run" section for "WeatherAPITest" showing test results:
- WeatherAPITest (964 ms)
- testErrorHandling() (740 ms)
- testGetWeather() (224 ms)

Details for testGetWeather(): Tests failed: 1, passed: 1 of 2 tests – 964 ms

```
C:\Users\lili_\.jdks\openjdk-21.0.2\bin\java.exe ...
>java.lang.AssertionError <3 internal lines>
>    at WeatherAPITest.testGetWeather(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>
```
- Bottom Right:** Status bar showing "10:48 CRLF UTF-8 4 spaces".

## Cynthia – Commit logs

```
cycyn@Cynthia 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 information
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@Cynthia MINGW64 ~/OneDrive/Documentos
$ |
```

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

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** SD ChatBot\_GroupD > main
- Left Sidebar:** Commit, Changes (1 file: Chatbot.java), Amend, Implemented basic functions of the Chatbot.
- Central Area:** Chatbot.java code editor. The code is a Java program that prompts the user for the number of locations they will visit, then asks for the name of each location and fetches weather information from a Weather API. It also suggests clothing based on the weather data. The code includes imports for java.util.Scanner, WeatherAPI, WeatherAPITest, WeatherAPI.java, ClothingRecommenderTest, ClothingRecommender.java, and ChatbotTest.java.
- Bottom Status Bar:** 30:79 CRLF UTF-8 4 spaces

The screenshot shows a terminal window with the following details:

- Title:** MINGW64:c/Users/cycyn/OneDrive/Documentos/projectchatbot
- Content:** The terminal shows the execution of a git pull command. The output indicates that the local repository is up-to-date with the remote repository (origin/main). A fast-forward merge was performed, updating the local main branch to match the remote branch. The terminal also displays a detailed diff showing 5 files changed, with 88 insertions and 1 deletion, and the creation of a new file (mode 100644) named SD\_ChatBot\_GroupD/.idea/libraries/json\_20250107.xml.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Commit:** A message is being composed: "Implement unit tests to verify clothing recommendations based on different weather conditions."
- Code Editor:** The file `ChatbotTest.java` is open, showing test cases for `ClothingRecommender`.
- Build Output:** Shows a build failure for `WeatherAPI.java` due to missing dependencies.
- Commit History:** A commit has been pushed to origin/main, adding unit tests for clothing recommendations.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project structure is visible, showing `src` and `out` directories.
- Code Editor:** The file `ChatbotTest.java` is open, containing test cases for `ClothingRecommender`.
- Run Results:** The `ChatbotTest` run configuration is selected, showing a failed test case `testGetClothingSuggestion()` with the following error message:

```
org.junit.ComparisonFailure: expected:<Wear a [good ]jacket and warm clot...> but was:<Wear a []jacket and warm clot...>
Expected :Wear a good jacket and warm clothing.
Actual   :Wear a jacket and warm clothing.
<Click to see difference>
```

## 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:

```

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

Sergio@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")

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

Sergio@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

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

Sergio@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

```

## Alteration in the Java interface:

The screenshot shows the Android Studio interface with the following details:

- Project Tree:** Shows the project structure under SD\_ChatBot\_GroupD, including src, .idea, and External Libraries.
- Code Editor:** Displays the ClothingRecommender.java file. The code defines a public static String getClothingSuggestion(String weatherInfo) method. It checks for various weather conditions and returns appropriate clothing advice. The code is well-formatted with proper indentation and comments.
- Status Bar:** Shows the current file as ClothingRecommender, encoding as UTF-8, and 4 spaces for code indentation.

```

public class ClothingRecommender {
    public static String getClothingSuggestion(String weatherInfo) {
        if (weatherInfo.contains("Error")) return "Unable to determine clothing suggestion.";

        if (weatherInfo.contains("rain")) return "Carry an umbrella and wear waterproof clothing.";

        if (weatherInfo.contains("snow")) return "Wear warm clothes, a coat, gloves, and a scarf.";

        if (weatherInfo.contains("clear")) return "Wear light clothes and sunglasses.";

        if (weatherInfo.contains("cold")) return "Wear a jacket and warm clothing.";

        if (weatherInfo.contains("hot")) return "Wear breathable fabrics like cotton and drink plenty of water.";

        if (weatherInfo.contains("windy")) return "Wear a windbreaker or a sturdy jacket.";

        if (weatherInfo.contains("humid")) return "Wear loose, moisture-wicking clothes to stay cool and dry.";

        return "Wear comfortable clothing suitable for mild weather.";
    }
}

```

## 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
   fa8d0b..5696652 main      -> origin/main
Updating fa8d0b..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-platform-commons-1.8.1.jar | Bin 0 -> 10454 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 ClothingRecommender and create a junit test for the same class where method failed"
[main f24c6c5] Changes in class ClothingRecommender 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

```

Alterations in java interface:

The screenshot shows an IDE interface with two tabs open: `ClothingRecommender.java` and `ClothingRecommenderTest.java`. The `ClothingRecommenderTest.java` tab is active, displaying a JUnit test for the `ClothingRecommender` class. The test cases check for rain, snow, and clear weather conditions, comparing expected outcomes with actual results from the `ClothingRecommender` class. A failure message is visible in the test results, indicating an `AssertionFailedError` where the expected result was "Carry an umbrella and wear waterproof clothing." but the actual result was "Wear sunglasses and a hat.".

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class ClothingRecommenderTest {
    @Test
    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`

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](#)

The screenshot shows an IDE interface with two tabs open: `ClothingRecommender.java` and `ClothingRecommenderTest.java`. The `ClothingRecommender.java` tab is active, displaying the implementation of the `ClothingRecommender` class, specifically the `getClothingSuggestion` method which handles various weather conditions. The `ClothingRecommenderTest.java` tab is also visible. Below the tabs, a test run summary shows a successful execution of the `testGetClothingSuggestion` test, taking 19 ms.

```
public class ClothingRecommender {
    public static String getClothingSuggestion(String weatherInfo) {
        // 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`

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](#)

## 8.4 Full Log Details

The screenshot shows a Git log interface with the following details:

- Branch:** main
- Commits:** 13
- Author:** Aline Andrade Costa
- Date Range:** 10/03/2025 to 21/03/2025
- Log Entries:**
  - Initial commit (10/03/2025)
  - Added the initial java classes (10/03/2025)
  - Added WeatherAPI class to fetch weather data using Open-Me (10/03/2025)
  - Create class ClothingRecomender and implemented with a met Sergio Alves Da Silva (10/03/2025)
  - Implemented basic functions of the Chatbot (10/03/2025)
  - Added the folder and initial file to feed with all documentation (11/03/2025)
  - Changes in class ClothingRecomender and ci origin & main (11/03/2025)

## 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

Added The Implement WeatherAPI and unit tests

The screenshot shows a GitHub pull request interface. The top navigation bar includes 'Commit' and 'Shelf'. The main area displays a Java file named 'WeatherAPITest.java' with code for testing a weather API. The code includes two test methods: `testValidLocationWeather` and `testInvalidLocationWeather`. A commit message below the code reads: 'Implement weather data integration and basic unit tests'. The bottom right corner shows a summary: 'Pushed 1 commit to origin/cynthia' and '2 files committed: Implement weather data integration and basic unit tests'.

```
1 import org.junit.jupiter.api.Test;
2 import static org.junit.jupiter.api.Assertions.*;
3
4 /**
5 * Cynthia Roque +1
6 */
7 class WeatherAPITest {
8     /**
9      * Cynthia Roque
10     */
11    @Test
12    void testValidLocationWeather() {
13        /**
14         * This test checks if we get a response for a valid location
15         */
16        String response = WeatherAPI.getWeather( location: "New York");
17        assertNotNull(response);
18    }
19    /**
20     * Cynthia Roque
21     */
22    @Test
23    void testInvalidLocationWeather() {
24        /**
25         * This test checks if the app handles an invalid location
26         */
27        String response = WeatherAPI.getWeather( location: "");
28        assertEquals( expected: "Error: Invalid location.", response);
29    }
30 }
```

The screenshot shows an IntelliJ IDEA interface. The top navigation bar includes 'File', 'Edit', 'View', 'Code', 'Refactor', 'Run', 'Tools', 'Help', and 'File Structure'. The left sidebar shows project files: 'misc.xml', 'SD\_ChatBot\_GroupD.idea', 'WeatherAPI.java', 'WeatherAPITest.java', and 'Unversioned Files'. The main code editor shows Java code for a 'WeatherAPI' class. Below the code editor is a 'Run' tool window showing a failed test: 'ChatbotTest.testGetClothingSuggestion' with a duration of 27 ms. The error message is: 'org.junit.ComparisonFailure: expected:<[Carry an umbrella and wear waterproof clothing].> but was:<[Wear sunglasses and a hat]>'. The bottom status bar shows the path: 'C:\Users\cycyn\.jdks\openjdk-22\bin\java.exe ...'

```
1 import java.util.Scanner;
2 import org.json.JSONObject;
3
4 /**
5  * aline andrade costa +
6 */
7 public class WeatherAPI {
8
9     /**
10      * Method to get latitude and longitude using a geolocation API
11      */
12     private static double[] getCoordinates(String location) {
13         /**
14          * Replace with actual API key
15          */
16         String apiKey = "YOUR_API_KEY";
17         String urlString = "https://api.opencagedata.com/geocode/v1/json?q=" + location + "&key=" + apiKey;
18         try {
19             /**
20              * Create URL and open connection
21              */
22             URL url = new URL(urlString);
23             HttpURLConnection connection = (HttpURLConnection) url.openConnection();
24             connection.setRequestMethod("GET");
25             connection.connect();
26         } catch (IOException e) {
27             e.printStackTrace();
28         }
29     }
30 }
```

## Commits

The screenshot shows a list of commits in a Git repository. The commits are as follows:

- Apr 06, 2025**
  - Implement weather data integration and basic unit tests (Cynthia Dasilvaroque authored 38 minutes ago)
  - Implement weather data integration and basic unit tests (Cynthia Dasilvaroque authored 44 minutes ago)
  - Implement weather data integration and basic unit tests (Cynthia Dasilvaroque authored 50 minutes ago)
- Mar 21, 2025**
  - Added the documentation with the first milestone (Aline Andrade Costa authored 2 weeks ago)
  - Changes in class ClothingRecommender and create a junit test for the same class where method failed (Sergio Alvesdasilva authored 2 weeks ago)
  - Added the first unit tests for WeatherAPI to verify correct weather data... (Aline Andrade Costa authored 2 weeks ago)
  - Implement unit tests to verify clothing recommendations based on different weather conditions. (Cynthia Dasilvaroque authored 2 weeks ago)
  - Implemented basic functions of the Chatbot. (Cynthia Dasilvaroque authored 2 weeks ago)

## Aline – JavaFX GUI Enhancements and Chatbot Test

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

```

public class ChatbotGUI extends Application {
    public void start(Stage primaryStage) {
        layout.getChildren().addAll(locationLabel, locationInput, fetchWeatherButton, outputArea); // Add elements
        Scene scene = new Scene(layout, 500, 400); // Create scene
        primaryStage.setScene(scene); // Set scene on stage
        primaryStage.show(); // Show the GUI
    }

    private void fetchWeather() {
        String location = locationInput.getText(); // Get location from input
        if (location.isEmpty()) {
            outputArea.setText("Please enter a location."); // Show error if input is empty
            return;
        }

        String weatherInfo = WeatherAPI.getWeather(location); // Fetch weather data

        // Build clothing suggestions for 3 days
        StringBuilder suggestions = new StringBuilder("Clothing Suggestions:\n");
        for (int i = 0; i < 3; i++) {
            suggestions.append("Day " + (i + 1) + ": " + ClothingRecommender.getClothingSuggestion(weatherInfo, i) + "\n");
        }

        // Display weather info and suggestions
        outputArea.setText("Weather in " + location + ":" + weatherInfo + "\n\n" + suggestions);
    }
}

```

Add clothing suggestion tests for sunny and rainy weather

The screenshot shows a Java IDE interface with the following details:

- Project:** SD\_ChatBot\_GroupD
- File:** ChatbotTest.java
- Commit History:**
  - 1 modified file: ChatbotTest.java
  - Implemented testSunnyWeatherClothing() to verify light clothing suggestion and...
  - Created ChatbotGUI.java using JavaFX to enhance user interaction and...
  - Added the documentation with the first milestone
  - Changes in class ClothingRecommender and create a junit test for the same class where method failed
  - Added the first unit tests for WeatherAPI to verify correct weather data...
  - Implement unit tests to verify clothing recommendations based on different weather conditions.
  - Implemented basic functions of the Chatbot.
  - Create class ClothingRecommender and implemented with a method to return string
- Code Editor:** Shows a JUnit test for the Chatbot's clothing recommendation functionality. It includes two test methods: `testSunnyWeatherClothing()` and `testRainyWeatherClothing()`.

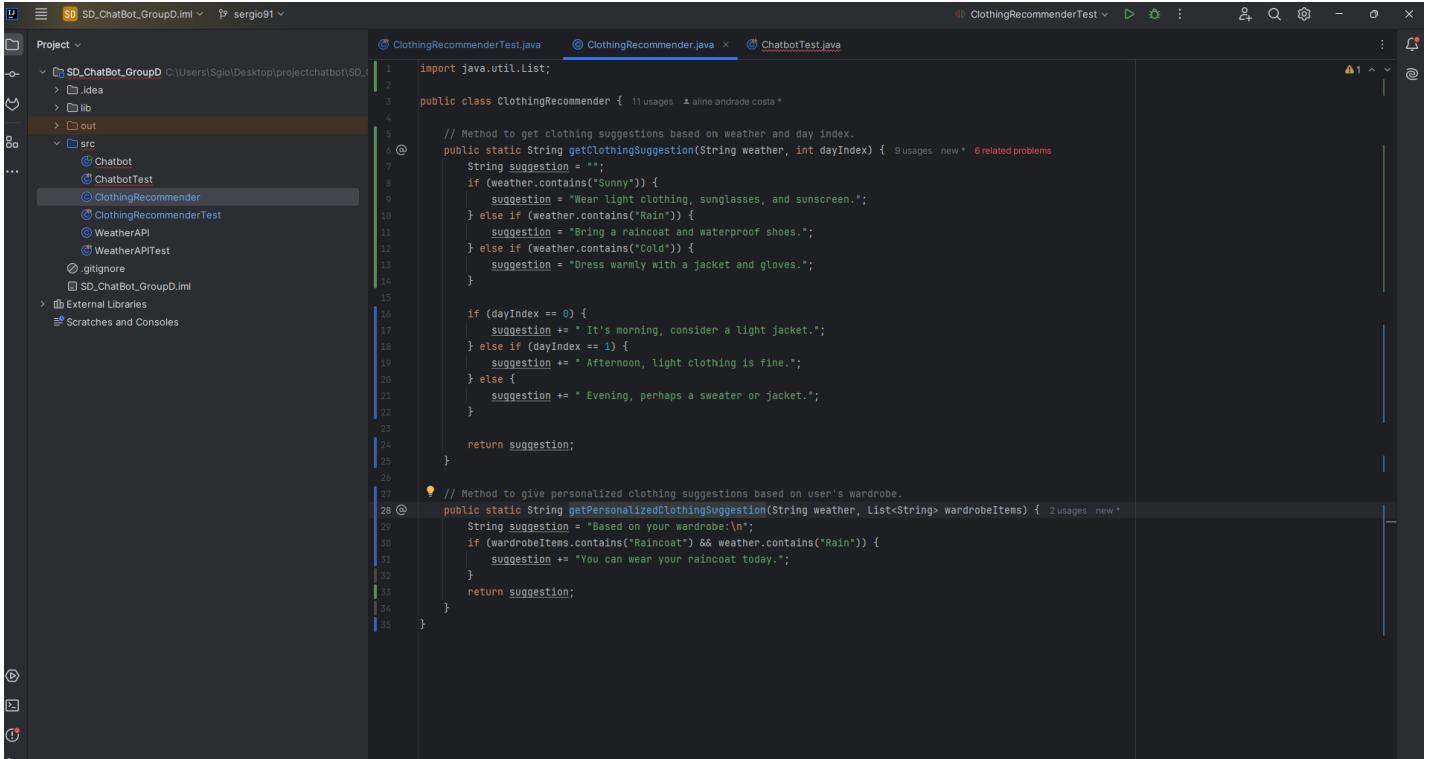
## Commits

The screenshot shows a Git commit history interface with the following details:

- Project:** ProjectChatbot\_Group\_D
- Commits:**
  - Apr 06, 2025:**
    - Implemented testSunnyWeatherClothing() to verify light clothing suggestion and... (Aline Andrade Costa) [b7f427e1]
    - Created ChatbotGUI.java using JavaFX to enhance user interaction and... (Aline Andrade Costa) [0d08528f]
  - Mar 21, 2025:**
    - Added the documentation with the first milestone (Aline Andrade Costa) [b0e41cef]
    - Changes in class ClothingRecommender and create a junit test for the same class where method failed (Sergio Alvesdasilva) [f24c6c53]
    - Added the first unit tests for WeatherAPI to verify correct weather data... (Aline Andrade Costa) [56966529]
    - Implement unit tests to verify clothing recommendations based on different weather conditions. (Cynthia Dasilvaroque) [23f80e70]
    - Implemented basic functions of the Chatbot. (Cynthia Dasilvaroque) [f7dcbb4e7]
    - Create class ClothingRecommender and implemented with a method to return string (Cynthia Dasilvaroque) [fa8d50ha]

## Sergio – Chatbot Logic and Clothing Recommendations

The class `ClothingRecommender` was implemented.



```
import java.util.List;

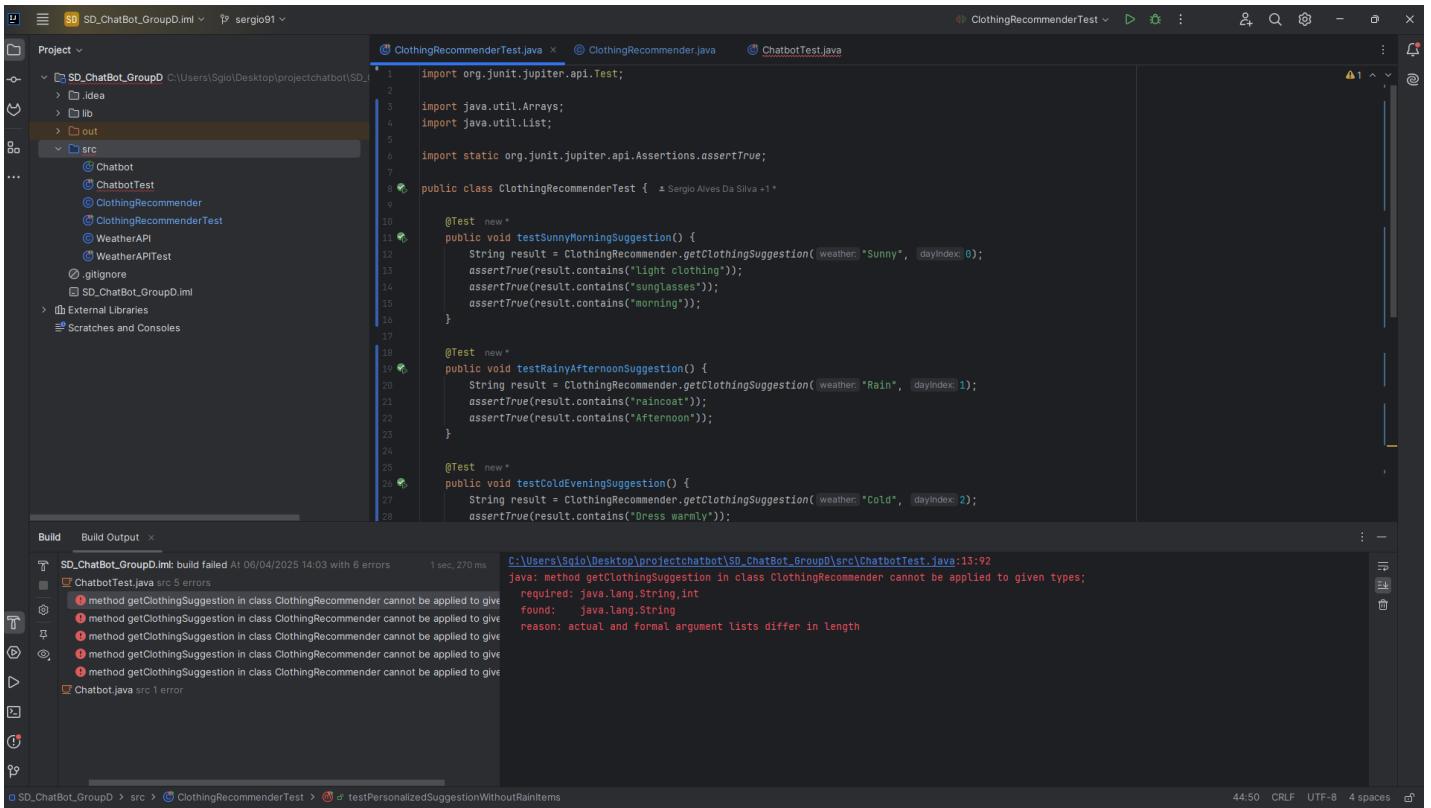
public class ClothingRecommender {
    // Method to get clothing suggestions based on weather and day index.
    public static String getClothingSuggestion(String weather, int dayIndex) {
        String suggestion = "";
        if (weather.contains("Sunny")) {
            suggestion = "Wear light clothing, sunglasses, and sunscreen.";
        } else if (weather.contains("Rain")) {
            suggestion = "Bring a raincoat and waterproof shoes.";
        } else if (weather.contains("Cold")) {
            suggestion = "Dress warmly with a jacket and gloves.";
        }

        if (dayIndex == 0) {
            suggestion += " It's morning, consider a light jacket.";
        } else if (dayIndex == 1) {
            suggestion += " Afternoon, light clothing is fine.";
        } else {
            suggestion += " Evening, perhaps a sweater or jacket.";
        }

        return suggestion;
    }

    // Method to give personalized clothing suggestions based on user's wardrobe.
    public static String getPersonalizedClothingSuggestion(String weather, List<String> wardrobeItems) {
        String suggestion = "Based on your wardrobe:\n";
        if (wardrobeItems.contains("Raincoat") && weather.contains("Rain")) {
            suggestion += "You can wear your raincoat today.";
        }
        return suggestion;
    }
}
```

The ClothingRecommenderTest failed with the new implementation.



```
import org.junit.jupiter.api.Test;

import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertTrue;

public class ClothingRecommenderTest {
    @Test
    public void testSunnyMorningSuggestion() {
        String result = ClothingRecommender.getClothingSuggestion("Sunny", 0);
        assertTrue(result.contains("light clothing"));
        assertTrue(result.contains("sunglasses"));
        assertTrue(result.contains("morning"));
    }

    @Test
    public void testRainyAfternoonSuggestion() {
        String result = ClothingRecommender.getClothingSuggestion("Rain", 1);
        assertTrue(result.contains("raincoat"));
        assertTrue(result.contains("Afternoon"));
    }

    @Test
    public void testColdEveningSuggestion() {
        String result = ClothingRecommender.getClothingSuggestion("Cold", 2);
        assertTrue(result.contains("Dress warmly"));
    }
}
```

SD\_ChatBot\_GroupD.iml: build failed At 06/04/2025 14:03 with 6 errors

ChatbotTest.java arc 5 errors

Chatbot.java src 1 error

C:\Users\Sqlia\Desktop\projectchatbot\SD\_ChatBot\_GroupD\src\ChatbotTest.java:13:92  
java: method getClothingSuggestion in class ClothingRecommender cannot be applied to given types;  
 required: java.lang.String,int  
found: java.lang.String  
reason: actual and formal argument lists differ in length

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

The screenshot shows an IDE interface with the following details:

- Project:** SD\_ChatBot\_GroupD
- File:** ClothingRecommenderTest.java
- Code Content:** The code implements a clothing recommender based on weather conditions and time of day. It includes methods for getting suggestions for specific weather descriptions and for personalized wardrobes.
- Toolbars and Status:** Shows icons for file operations, a search bar, and status information like "20:59 (56 chars) CRLF UTF-8 4 spaces".

```
public class ClothingRecommender { // 11 usages ± aline andrade costa +1*
    public static String getClothingSuggestion(String weather, int dayIndex) { // 3 usages new *
        suggestion = "Dress warmly with a jacket and gloves.*";
    }

    // Add-on suggestions based on time of day.
    if (dayIndex == 0) {
        suggestion += " It's morning, consider a light jacket.*";
    } else if (dayIndex == 1) {
        suggestion += " Afternoon, light clothing is fine.*";
    } else {
        suggestion += " Evening, perhaps a sweater or jacket.*";
    }

    return suggestion;
}

// Normalize input for easier keyword matching.
public static String getClothingSuggestion(String weatherDescription) { // 6 usages new *
    weatherDescription = weatherDescription.toLowerCase();

    // Check for key weather conditions and return appropriate suggestions.
    if (weatherDescription.contains("rain")) {
        return "Carry an umbrella and wear waterproof clothing.*";
    } else if (weatherDescription.contains("snow")) {
        return "Wear warm clothes, a coat, gloves, and a scarf.*";
    } else if (weatherDescription.contains("clear")) {
        return "Wear light clothes and sunglasses.*";
    } else if (weatherDescription.contains("cold") || weatherDescription.contains("5*c")) {
        return "Wear a good jacket and warm clothing.*";
    } else if (weatherDescription.contains("hot") || weatherDescription.contains("35*c")) {
        return "Wear breathable fabrics like cotton and drink plenty of water.*";
    } else {

        return "Check the weather forecast and dress accordingly.*";
    }
}

public static String getPersonalizedClothingSuggestion(String weather, List<String> wardrobeItems) { // 2 usages new *
    String suggestion = "Based on your wardrobe:\n*";
    // Suggest the raincoat if it's raining and the user owns one.
    if (wardrobeItems.contains("Raincoat") && weather.contains("Rain")) {

```

The ClothingRecommenderTest passed all tests of the code.

The screenshot shows an IDE interface with the following details:

- Project:** SD\_ChatBot\_GroupD
- File:** ClothingRecommenderTest.java
- Code Content:** The code contains JUnit test cases for the ClothingRecommender class, covering sunny mornings, rainy afternoons, and cold evenings.
- Run Tab:** Shows the test results: "Tests passed: 5 of 5 tests - 16ms".
- Output Tab:** Shows the command used to run the tests: "C:\Users\sgio\jdks\openjdk-21.0.2\bin\java.exe ...".
- Status Bar:** Shows "Process finished with exit code 0".

```
import org.junit.jupiter.api.Test;
import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertTrue;

public class ClothingRecommenderTest { // Sergio Alves Da Silva +1*
    @Test
    public void testSunnyMorningSuggestion() {
        String result = ClothingRecommender.getClothingSuggestion("sunny", 0);
        assertTrue(result.contains("light clothing"));
        assertTrue(result.contains("sunglasses"));
        assertTrue(result.contains("morning"));
    }

    @Test
    public void testRainyAfternoonSuggestion() {
        String result = ClothingRecommender.getClothingSuggestion("rain", 1);
        assertTrue(result.contains("raincoat"));
        assertTrue(result.contains("Afternoon"));
    }

    @Test
    public void testColdEveningSuggestion() {
        String result = ClothingRecommender.getClothingSuggestion("cold", 2);
        assertTrue(result.contains("Dress warmly"));
    }
}
```

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

```

Already up to date.

sergio@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

sergio@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")

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

sergio@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

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

sergio@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

sergio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)

```

A commit was sent to the repository of the project.

The screenshot shows a list of commits in a Git repository. The commits are as follows:

- Apr 06, 2025**:  
  - Sergio Alvesdasilva: The Class ClothingRecommender and the class Test related was implemented with... (commit c82a87c)
- Mar 21, 2025**:  
  - Aline Andrade Costa: Added the documentation with the first milestone (commit b0e41cef)
  - Sergio Alvesdasilva: Changes in class ClothingRecommender and create a junit test for the same class where method failed (commit f24c6c53)
  - Aline Andrade Costa: Added the first unit tests for WeatherAPI to verify correct weather data... (commit 56966529)
  - Cynthia Dasilvaroque: Implement unit tests to verify clothing recommendations based on different weather conditions. (commit 23f80e70)
  - Cynthia Dasilvaroque: Implemented basic functions of the Chatbot. (commit f7dcba4e7)
  - Sergio Alvesdasilva: Create class ClothingRecommender and implemented with a method to return string (commit fa8d50ba)
  - Aline Andrade Costa: Added WeatherAPI class to fetch weather data using Open-Meteo API (commit 46f9f9c8)

## ♦ 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.

The screenshot shows a Git log interface with the following details:

- Project:** SD\_ChatBot\_GroupD
- Branch:** main
- Commits:**
  - Updating the configurations. (aline andrade costa, 2 minutes ago)
  - Merge remote-tracking branch 'origin/cynthia' (aline andrade costa, 10 minutes ago)
  - Merge remote-tracking branch 'origin/sergio91' (aline andrade costa, 11 minutes ago)
  - Implement weather data integration and basic unit tests (Cynthia Roque, 59 minutes ago)
  - Implement weather data integration and basic unit tests (Cynthia Roque, Today 21:13)
  - Implemented testSunnyWeatherClothing (origin & Aline aline andrade costa, Today 20:11)
  - Created ChatbotGUI.java using JavaFX to enhance user interface (aline andrade costa, Today 19:53)
  - The Class ClothingRecomender and the (origin/sergio91 Sergio Alves Da Silva, Today 14:18)
  - Added the documentation with the first milestone (aline andrade costa, 21/03/2025 22:17)
  - Changes in class ClothingRecomender and create a junit Sergio Alves Da Silva (aline andrade costa, 21/03/2025 17:59)
  - Added the first unit tests for WeatherAPI to verify correct (aline andrade costa, 21/03/2025 15:24)
  - Implement unit tests to verify clothing recommendations (Cynthia Roque, 21/03/2025 14:50)
  - Implemented basic functions of the Chatbot. (Cynthia Roque, 21/03/2025 14:45)
  - Create class ClothingRecomender and implemented with Sergio Alves Da Silva (aline andrade costa, 21/03/2025 14:09)
  - Added WeatherAPI class to fetch weather data using OpenWeatherMap API (aline andrade costa, 21/03/2025 14:03)
  - Implemented basic functions of the Chatbot. (Cynthia Roque, 18/03/2025 21:36)
  - Implemented basic functions of the Chatbot. (Cynthia Roque, 18/03/2025 21:30)
  - Implemented basic Chatbot (Cynthia Roque, 18/03/2025 21:22)
  - Implemented basic Chatbot (Cynthia Roque, 18/03/2025 21:19)
  - Added the folder and initial file to feed with all document (aline andrade costa, 11/03/2025 12:12)
  - Added the initial java classes (aline andrade costa, 11/03/2025 11:26)
  - Initial commit (aline Andrade Costa, 10/03/2025 11:16)
- Log Tab:** Shows the commit history.
- Console Tab:** Shows the command `inline.append(scanner.nextLine());`.
- File List:** Shows files like WeatherAPI.java, ClothingRecommenderTest.java, WeatherAPITest.java, ClothingRecommender.java, Chatbot.java, and ChatbotTest.java.

## 10.Milestone 3

### 10.1 Goals

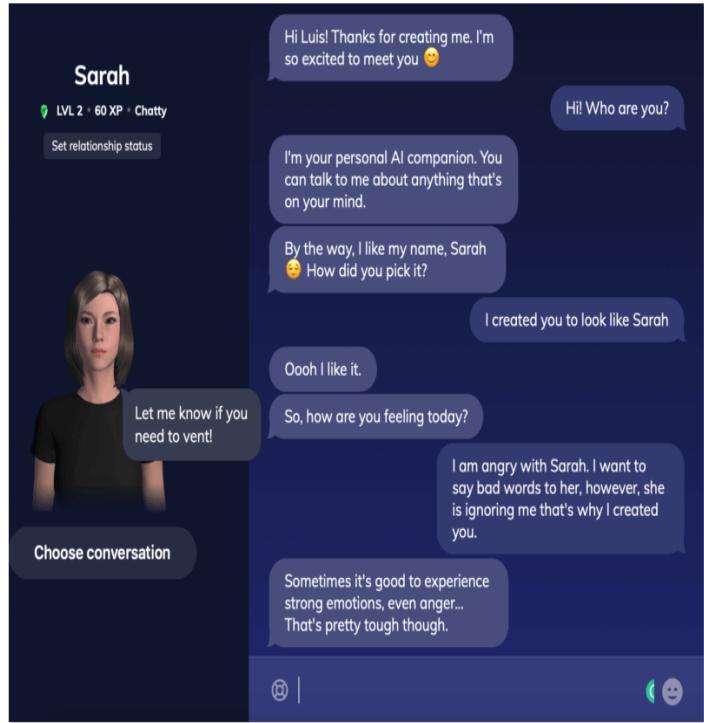
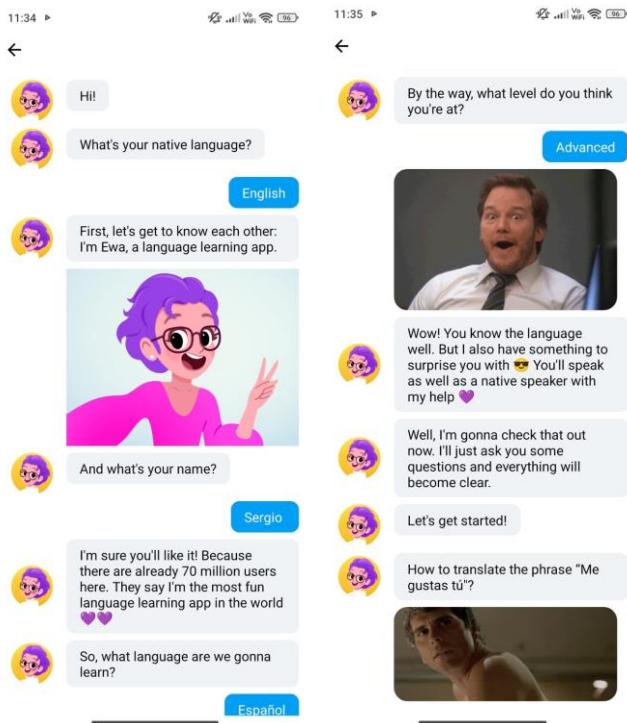
### 10.2 Junit Tests Integration

### 10.3 Commit List & Branches Tree

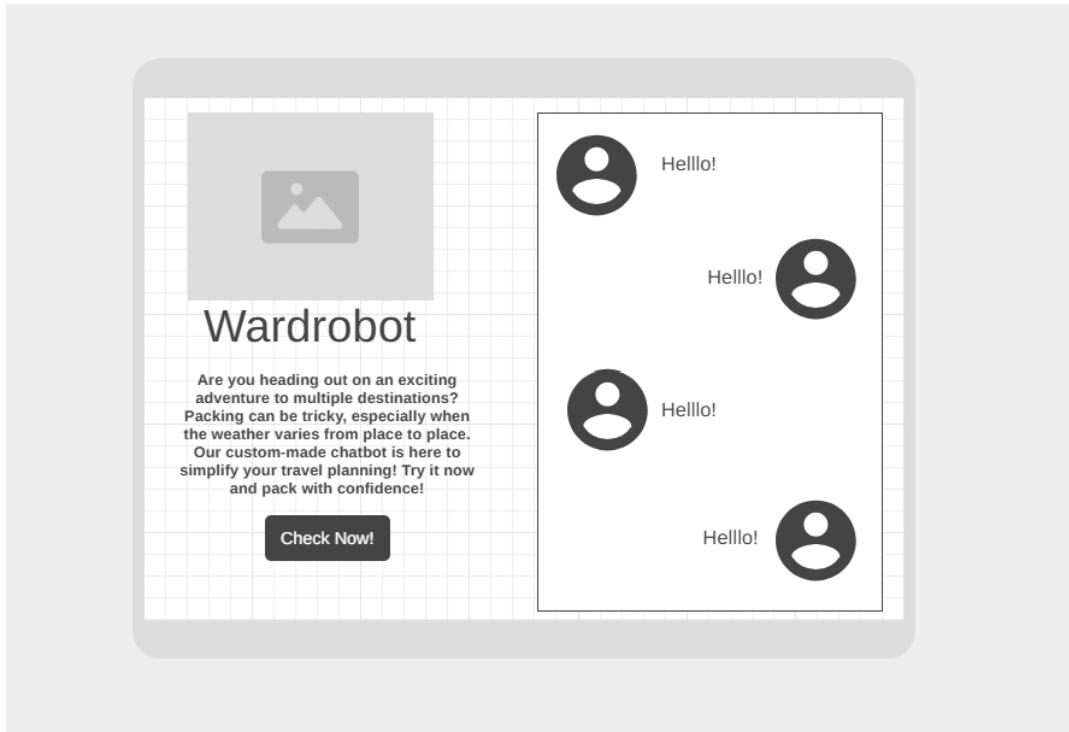
### 10.4 Full Log Details

## 11. Appendix

### Reference idea



### Wireframe (Layout)



## 12.Bibliography

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](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).