

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 27/04

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	13
8.1	Goals.....	13
8.2	Junit Tests.....	13
8.3	Commit Logs	14
8.4	Full Log Details.....	27
9.	Milestone 2	27
9.1	Goals.....	27
9.2	Junit Tests Integration	27
9.3	Commit List & Branches Tree	28
9.4	Full Log Details.....	36
10.	Milestone 3	37
10.1	Goals.....	37
10.2	Junit Tests Integration	37
10.3	Commit List & Branches Tree	38
10.4	Full Log Details.....	49
11.	Appendix.....	51
12.	Bibliography.....	52

1. Versioning Approach

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

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

Branching Strategy:

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

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

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

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

2. Development Process

This project follows an Agile iterative development process:

→ Milestones:

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

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

→ Code Reviews:

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

→ Integration:

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

→ Communication:

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

3. UI Implementation

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

→ User Interaction:

Users input location names through the console.

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

→ Upcoming Improvements:

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

4. Rest API

4.1 Rest API Implementation

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

→ Request Structure:

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

→ External Packages:

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

5. Weather API

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

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

Temperature (in Celsius)

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

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

- Usage in the Project

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

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

6. External Packages

JSON (org.json)

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

JUnit (JUnit 5)

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

HTTPURLConnection (Java Standard Library)

- Purpose: Used to make HTTP requests to the Weather API.
- Example: It connects to the API and retrieves the weather data.
- Usage: It's part of Java's standard library, so no external dependencies are needed.

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

7. Project Setup

The project setup is straightforward:

- Java Environment:

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

- Dependencies:

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

- Running the Application:

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

- Classes:

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

Chatbot

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

Purpose

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

General information about the bot.

1.



Name: Wardrobot.

Type: Rule-Based Chatbots (Personal Bot).

Environment: App.

Software: Java and embedded software.

Audience to attend: kids and adults aged.

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

Navigation

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

Cycle of communication

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

Possible interactions scenarios between user and chatbot

Case 1 (Simple)

User: Hi

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

User: Current weather

Bot: Sure! Which city?

User: Greece

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

User: Thanks.

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

Case 2 (Complex)

User: Hi

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

User: My name is Fabio.

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

User: Current weather

Bot: Sure! Which city?

User: Greece

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

User: Thanks.

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

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

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

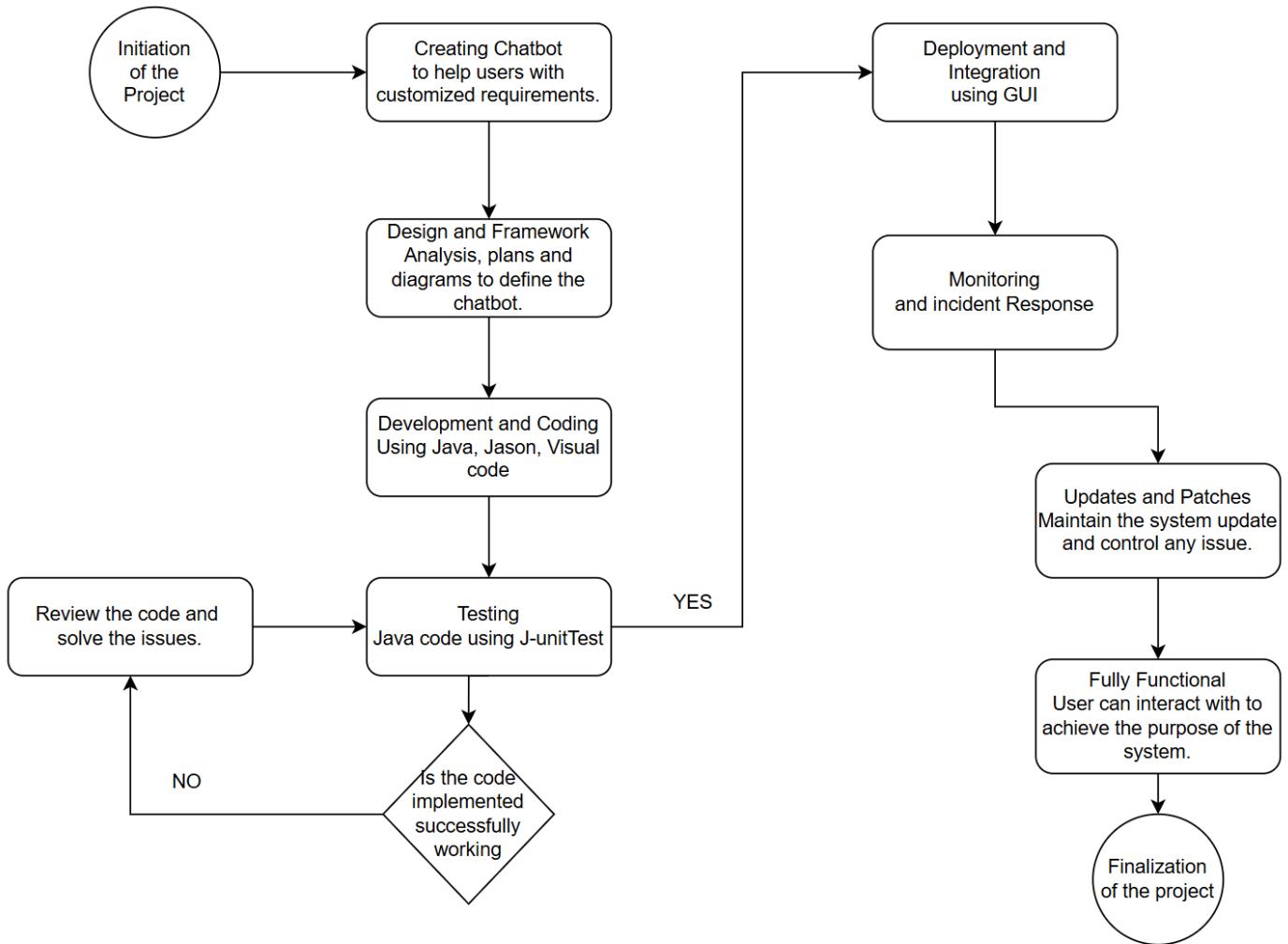
User: Athens, Corinto, Cavala, Pireu, Corfu. 12/10/2025 to 15/10/2025.

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

User: Thanks!

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

FlowChart Diagram (Cycle of the Project)



Written Use Case

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

Use case name: Plan Clothing for a Trip.

Actor: User.

System: Chatbot System.

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

Preconditions:

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

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

Main Flow:

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

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

Alternative Flows:

1. User enters invalid trip details.

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

2. The Weather API is unavailable.

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

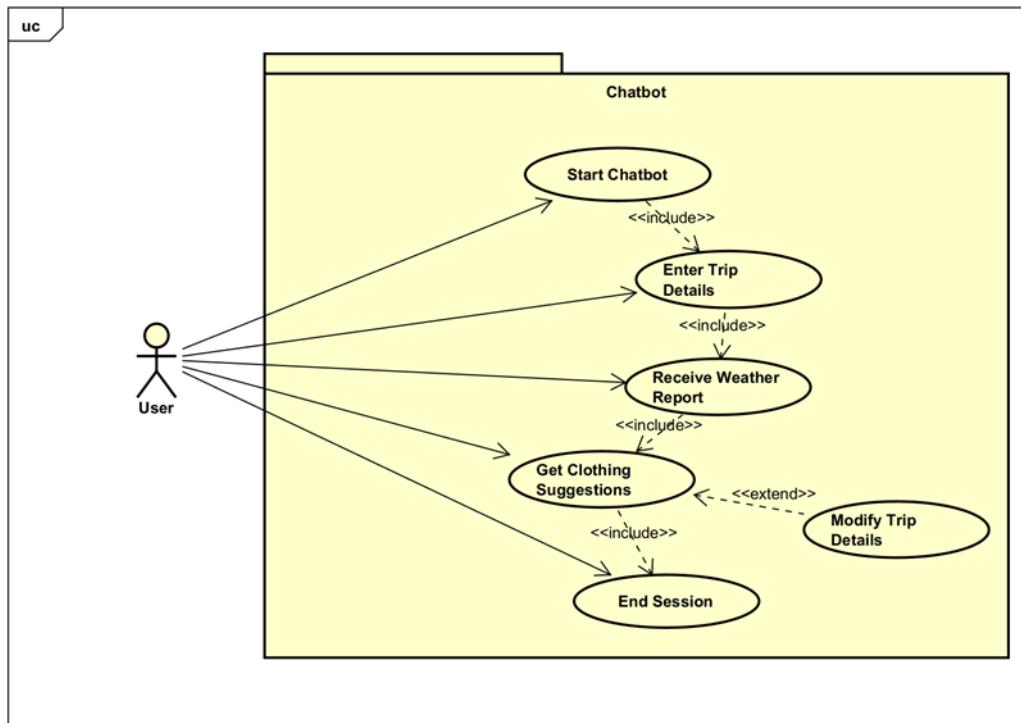
3. User exits before completing the process.

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

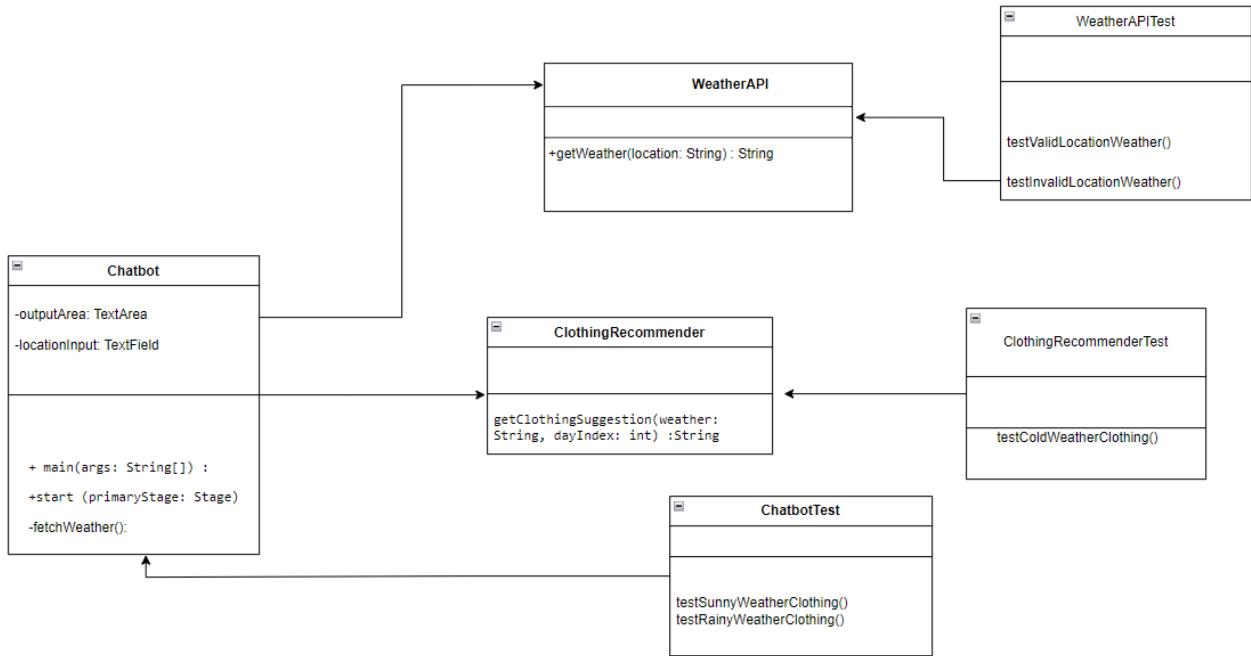
Postconditions:

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

Use Case Diagram



Class Diagram



8. Milestone 1

8.1 Goals

Implement basic weather-fetching functionality via the Weather API.

Provide basic clothing suggestions based on weather conditions.

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

→ The first milestone includes:

Fetching weather data via an API request.

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

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

8.2 Junit Tests

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

→ Tests:

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

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

8.3 Commit Logs

Aline – Commit logs

Cloning the repository:

```
MINGW64:/c/Users/lili/_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/'

Tili_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.

Tili_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 and returns it as a string.
- Status Bar:** Shows "2 warnings" and a message: "Commit and push checks failed 2 warnings Commit anyway and push More".

```
public class WeatherAPI { 1 usage a line andrade costa*
    public static String getWeather(String location) { 1 usage new *
        // Create a Scanner to read the response from the API
        Scanner scanner = new Scanner(url.openStream());
        StringBuilder inline = new StringBuilder();
        while (scanner.hasNext()) {
            inline.append(scanner.nextLine());
        }
        scanner.close();

        // Parse JSON response using the JSONObject class
        JSONObject data = new JSONObject(inline.toString()); // Convert the string response to a JSONObject
        JSONObject currentWeather = data.getJSONObject("current_weather");
        double temperature = currentWeather.getDouble("temperature");
        String description = currentWeather.getDouble("windspeed") > 20 ? "windy" : "clear";

        // Return the formatted weather information as a string
        return "Temperature: " + temperature + "\u00b0C, " + description;
    } catch (Exception e) {
        return "Error: " + e.getMessage();
    }
}
```

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

- File Explorer:** Shows a project structure with "src" containing "Chatbot", "ChatbotTest", "ClothingRecommender", "ClothingRecommenderTest", "WeatherAPI", "WeatherAPITest", ".gitignore", and "SD_ChatBot_GroupD.iml".
- Code Editor:** Displays the content of `WeatherAPITest.java`, which contains two test methods: `testGetWeather()` and `testErrorHandler()`.
- Run Interface:** Shows the results of running the tests. One test failed ("testErrorHandler()") and one passed ("testGetWeather()").

```
@Test
void testGetWeather() {
    // Test if weather information is returned correctly for a location
    String weather = WeatherAPI.getWeather("New York");
    assertTrue(weather.contains("Temperature"));
    assertTrue(weather.contains("\u00b0C"));
}

@Test
void testErrorHandler() {
    // Test if the API returns an error message for an invalid location
    String weather = WeatherAPI.getWeather("InvalidLocation");
    assertTrue(weather.contains("Error"));
}
```

Run Results:

Test	Status	Time
testErrorHandler()	Failed	783 ms
testGetWeather()	Passed	218 ms

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

- Top Bar:** Shows the project name "SD_ChatBot_GroupD" and 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 (1 modified: Implemented basic functions of the Chatbot.)
- Central Area:** Chatbot.java code editor.
- Code Content:**

```
1 import java.util.Scanner;
2
3 /**
4 * Cynthia Roque +1
5 */
6
7 public class Chatbot {
8
9     /**
10     * Cynthia Roque +1
11     */
12     public static void main(String[] args) {
13
14         Scanner scanner = new Scanner(System.in); // Create scanner object for user input
15
16         // Welcome message
17         System.out.println("Welcome to the Trip Clothing Planner Chatbot!");
18
19         // Ask for the number of locations
20         System.out.println("Enter the number of locations you will visit: ");
21         int numLocations = scanner.nextInt(); // Get the number of locations
22         scanner.nextLine(); // Clear the buffer
23
24         // Loop through each location
25         for (int i = 1; i <= numLocations; i++) {
26
27             // Ask for location name
28             System.out.println("Enter Location " + i + ":");
29             String location = scanner.nextLine();
30
31             // Fetch weather information for the location
32             String weatherInfo = WeatherAPI.getWeather(location); // Fetch weather data from the API
33
34             // Display the weather information
35             System.out.println("Weather in " + location + ": " + weatherInfo);
36
37             // Suggest clothing based on weather data
38             System.out.println("Clothing Suggestion: " + ClothingRecommender.getClothingSuggestion(weatherInfo));
39         }
40
41         scanner.close(); // Close the scanner to free up resources
42     }
43 }
```
- 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
- Command:** \$ git pull
- Output:**

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

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

- Commit:** A modal window titled "Commit" is open, showing changes made to "SD_ChatBot_GroupD". It lists "Changes 2 files" with "SD_ChatBot_GroupD.idea" and "SD_ChatBot_GroupD.iml" selected. A note says "Implement unit tests to verify clothing recommendations based on different weather conditions." with a checkmark.
- Code Editor:** The main editor window displays the file `ChatbotTest.java`. The code contains a test class `ChatbotTest` with a single test method `testGetClothingSuggestion` that asserts four different clothing suggestions based on weather conditions.
- Build Output:** The "Build" tab shows a failure for the project "SD_ChatBot_GroupD" at 21/03/2025 12 sec, 262 ms. The error is in `WeatherAPI.java`: "java: package org.json does not exist". Other errors include deprecated `URL` usage and missing `JSONObject` imports.
- Status Bar:** The bottom right shows the status "Pushed 2 commits to origin/main" and "1 file committed: Implement unit tests to verify clothing recommendations based on different weather conditions". The status bar also indicates the current time as 11:11, file encoding as CRLF, and other settings.

The screenshot shows the IntelliJ IDEA interface. The left sidebar displays the project structure for 'SD_ChatBot_GroupD' with 'src' expanded, showing files like Chatbot, ChatbotTest, ClothingRecommender, ClothingRecommenderTest, WeatherAPI, and WeatherAPITest. The right pane shows the code for 'ChatbotTest.java'. Below the code editor is the 'Run' tool window, which lists a single test 'testGetClothingSuggestion()' under the 'ChatbotTest' configuration. The test failed with the message: 'org.junit.ComparisonFailure: expected:<Wear a [good] jacket and warm clothing...> 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>'.

The terminal window shows the command 'git pull' being run. The output indicates that the repository is already up to date, with no changes made.

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

Sergio – Commit logs

Cloning the repository:

The terminal window shows the command 'git clone' being run to clone the repository from 'gitlab.griffith.ie'. The output shows the cloning process, including object enumeration, counting, compressing, and receiving objects, all completed successfully.

```
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.
```

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 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(+)

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:

The screenshot shows a Java project named "SD_ChatBot_GroupD" open in an IDE. The project structure on the left includes ".idea", "src" (containing "Chatbot", "ChatbotTest", "ClothingRecommender", "ClothingRecommenderTest", "WeatherAPI", "WeatherAPITest", ".gitignore", and "SD_ChatBot_GroupD.iml"), "External Libraries", and "Scratches and Consoles". The right pane displays the code for "ClothingRecommender.java". The code is a static method named "getClothingSuggestion" that takes a string "weatherInfo" and returns a clothing recommendation based on the weather conditions: Error, rain, snow, clear, cold, hot, windy, humid, and mild.

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

- Project View:** Shows the project structure under "SD ChatBot_GroupD".
- Code Editor:** Displays `ClothingRecommenderTest.java` with test cases for different weather conditions.
- Run Results:** Shows the output of the test run, indicating a failure for the rain condition test.
- Bottom Status Bar:** Shows the file path as "SD ChatBot_GroupD > src > ClothingRecommenderTest", and the status "31:2 CRLF UTF-8 4 spaces".

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

class ClothingRecommenderTest {
    @Test
    void testGetClothingSuggestion() {
        // Testing for rain condition.
        assertEquals("Carry an umbrella and wear waterproof clothing.", ClothingRecommender.getClothingSuggestion(weatherInfo: "Temperature: 15°C, rain"));

        // Testing for snow condition.
        assertEquals("Wear warm clothes, a coat, gloves, and a scarf.", ClothingRecommender.getClothingSuggestion(weatherInfo: "Temperature: -5°C, snow"));

        // Testing for clear weather condition.
        assertEquals("Wear light clothes and sunglasses.", ClothingRecommender.getClothingSuggestion(weatherInfo: "Temperature: 25°C, clear"));
    }
}
```

Run `ClothingRecommenderTest.testGetClothingSuggestion`

Tests failed: 1 of 1 test – 19 ms

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 the following details:

- Project View:** Shows the project structure under "SD ChatBot_GroupD".
- Code Editor:** Displays `ClothingRecommender.java` with the implementation of the `getClothingSuggestion` method.
- Run Results:** Shows the output of the test run, indicating a failure for the rain condition test.
- Bottom Status Bar:** Shows the file path as "SD ChatBot_GroupD > src > ClothingRecommender > getClothingSuggestion", and the status "19:45 CRLF UTF-8 4 spaces".

```
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

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:** Local main
- Remote:** origin main
- Commits:** 13 commits listed from newest to oldest.

Commit Message	Author	Date
Changes in class ClothingRecomender and ci	Sergio Alves Da Silva	53 minutes ago
Added the first unit tests for WeatherAPI to verify correct weat	aline andrade costa	Today 15:24
Implement unit tests to verify clothing recommendations basec	Cynthia Roque	Today 14:50
Implemented basic functions of the Chatbot.	Cynthia Roque	Today 14:45
Create class ClothingRecomender and implemented with a met	Sergio Alves Da Silva	Today 14:09
Added WeatherAPI class to fetch weather data using Open-Me	aline andrade costa	Today 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 documentation	aline andrade costa	11/03/2025 12:12
Added the initial java classes	aline andrade costa	11/03/2025 11:26
Initial commit	Aline Andradecosta	10/03/2025 11:16

9. Milestone 2

9.1 Goals

Planned Upgrades for Milestone 2

→ Improve Weather API Handling:

Add support for multiple locations and 3-day forecasts.

→ Enhance Clothing Suggestions:

Considering time of day and weather trends.

→ Basic GUI Implementation:

Using JavaFX for better user interaction.

→ Refine Version Control:

Implement branches, commits, and tags per versioning strategy.

→ More JUnit Tests:

Expand test coverage for API handling and UI components.

9.2 Junit Tests Integration

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

→ ChatbotTest:

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

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

→ ClothingRecommenderTest:

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

→ WeatherAPITest:

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

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

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

9.3 Commit List & Branches Tree

♦ Code Implementation

Cynthia – Weather API and Data Handling

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

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

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

Implement weather data integration and basic unit tests

The screenshot shows a GitHub commit interface for a Java project named SD_ChatBot_GroupD. The commit message is:

```
Added support for fetching weather by date.  
Improved location lookup and error handling.  
Created unit tests for ,  
valid/invalid locations, date ,  
formats, and future dates.
```

The commit history shows 11 commits pushed to the origin/cynthia branch. A tooltip indicates:

- Pushed 11 commits to origin/cynthia
- 2 files committed: Added support for fetching weather by date. Improved location lookup and error handling...

The code editor shows WeatherAPI.java and WeatherAPITest.java. WeatherAPI.java contains the following code:

```
49 // GeolocationAPI gets the latitude and longitude for a given location using Nominatim API.
50 class GeolocationAPI {
51     @
52         public static double[] getCoordinates(String location) {
53             try {
54                 // Encode the location name to ensure it is compatible with the URL
55                 String encodedLocation = URLEncoder.encode(location, "UTF-8");
56                 String urlString = "https://nominatim.openstreetmap.org/search?q=" + encodedLocation + "&format=json&limit=1";
57                 URL url = new URL(urlString);
58                 HttpURLConnection conn = (HttpURLConnection) url.openConnection();
59                 conn.setRequestMethod("GET");
60                 conn.connect(); // Connect to the server
61
62                 // If the response code is not successful, return null
63                 if (conn.getResponseCode() != 200) return null;
64                 Scanner scanner = new Scanner(url.openStream());
65                 StringBuilder inline = new StringBuilder();
66                 while (scanner.hasNext()) {
67                     inline.append(scanner.nextLine());
68                 }
69                 scanner.close();
70                 // Convert the API response from String to a JSON array
71                 JSONArray results = new JSONArray(inline.toString());
72                 if (results.length() == 0) return null;
73                 JSONObject locationData = results.getJSONObject(0);
74                 // Return the latitude and longitude as a double array
75                 return new double[]{locationData.getDouble("lat"), locationData.getDouble("lon")};
76             } catch (Exception e) {
77                 return null;
78             }
79         }
}
```

The screenshot shows a GitHub commit interface. On the left, there's a sidebar with 'Commit' and 'Shelf' tabs, and a 'Changes' section listing a single commit: 'Implement weather data integration and basic unit tests'. Below this is a 'Commits' section with a 'Commit' button. On the right, the main area shows the Java code for 'WeatherAPITest.java'. The code contains two test methods: 'testValidLocationWeather()' and 'testInvalidLocationWeather()'. The commit message is 'Pushed 1 commit to origin/cynthia' and the note says '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 /**
21 * Cynthia Roque
22 */
23    @Test
24    void testInvalidLocationWeather() {
25        /**
26         * This test checks if the app handles an invalid location
27         */
28        String response = WeatherAPI.getWeather( location: "");
29        assertEquals( expected: "Error: Invalid location.", response);
30    }
31 }
```

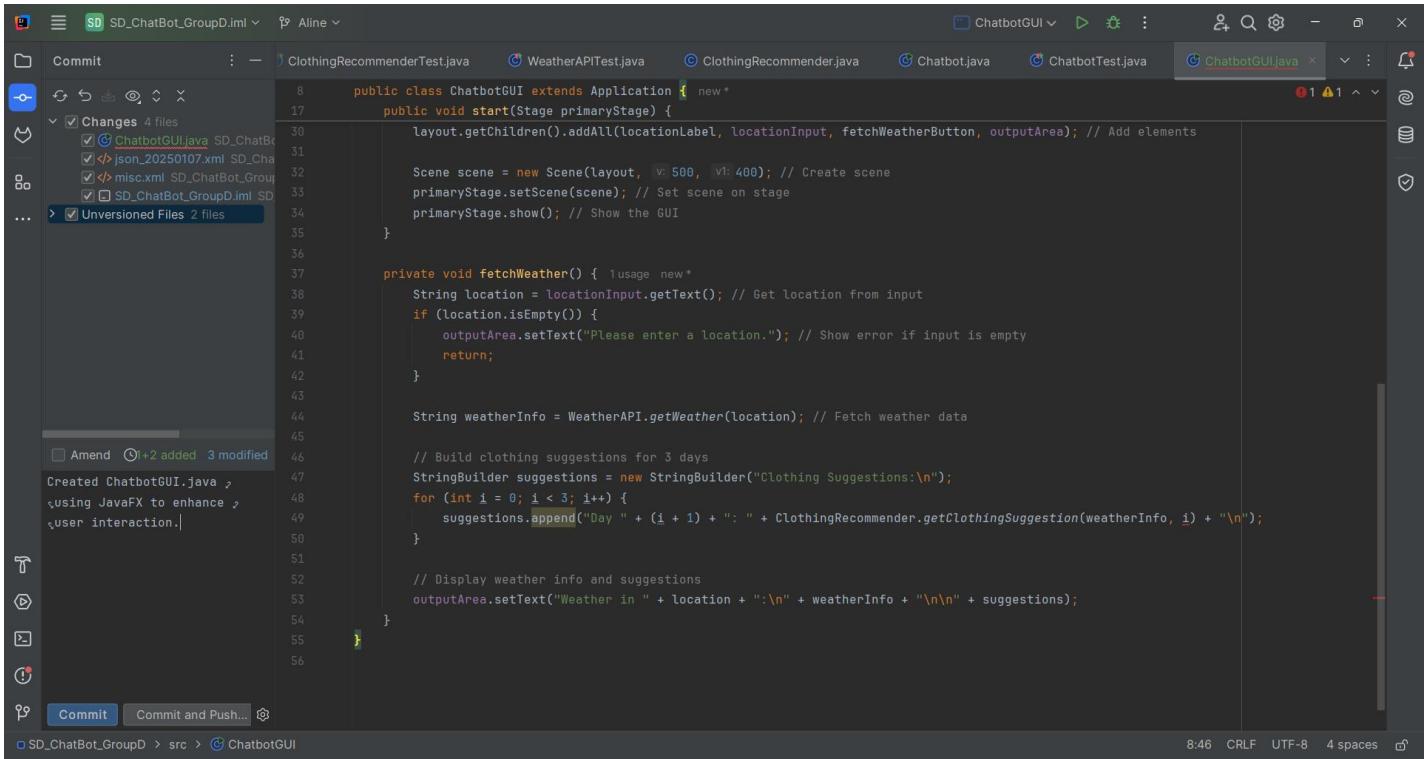
Commits

The screenshot shows a Git commit history for a repository named 'projectchatbot'. The sidebar on the left includes 'Project' (Manage, Plan, Code, Merge requests, Repository, Branches, Commits), 'Tags', 'Repository graph', 'Compare revisions', 'Snippets', 'Build', 'Secure', and 'Help'. The main area shows commits from 'cynthia' and 'Aline Andrade Costa'. The commits are grouped by date: April 06, 2025, and March 21, 2025. Each commit includes the author, message, timestamp, and a copy/paste icon.

Date	Author	Message	Timestamp	Copy/Paste
Apr 06, 2025	cynthia	Implement weather data integration and basic unit tests	38 minutes ago	[copy] [paste]
	cynthia	Implement weather data integration and basic unit tests	44 minutes ago	[copy] [paste]
	cynthia	Implement weather data integration and basic unit tests	50 minutes ago	[copy] [paste]
Mar 21, 2025	Aline Andrade Costa	Added the documentation with the first milestone	2 weeks ago	[copy] [paste]
	Sergio AlvesdaSilva	Changes in class ClothingRecommender and create a junit test for the same class where method failed	2 weeks ago	[copy] [paste]
	Aline Andrade Costa	Added the first unit tests for WeatherAPI to verify correct weather data... [more]	2 weeks ago	[copy] [paste]
	Cynthia Dasilvaroqe	Implemented unit tests to verify clothing recommendations based on different weather conditions.	2 weeks ago	[copy] [paste]
	Cynthia Dasilvaroqe	Implemented basic functions of the Chatbot.	2 weeks ago	[copy] [paste]

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() { // usage new*
        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

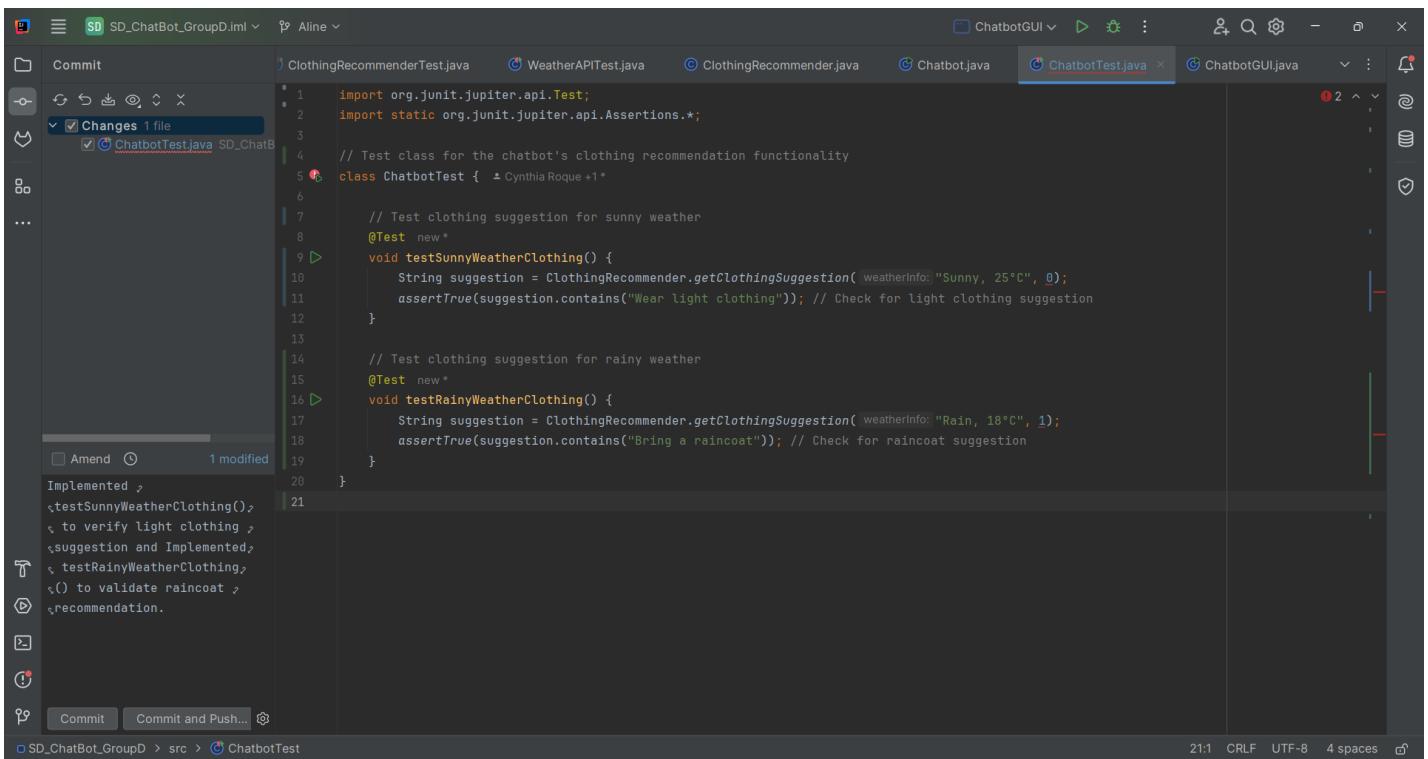
        // Build clothing suggestions for 3 days
        StringBuilder suggestions = new StringBuilder("Clothing Suggestions:\n");
        for (int i = 0; i < 3; i++) {
            suggestions.append("Day " + (i + 1) + ": " + ClothingRecommender.getClothingSuggestion(weatherInfo, i) + "\n");
        }

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

Commit Changes 4 files Unversioned Files 2 files

Amend Commit Commit and Push... 8:46 CRLF UTF-8 4 spaces

Add clothing suggestion tests for sunny and rainy weather



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

// Test class for the chatbot's clothing recommendation functionality
class ChatbotTest { // Cynthia Roque +1

    // Test clothing suggestion for sunny weather
    @Test new*
    void testSunnyWeatherClothing() {
        String suggestion = ClothingRecommender.getClothingSuggestion( weatherInfo: "Sunny, 25°C", 0 );
        assertTrue(suggestion.contains("Wear light clothing")); // Check for light clothing suggestion
    }

    // Test clothing suggestion for rainy weather
    @Test new*
    void testRainyWeatherClothing() {
        String suggestion = ClothingRecommender.getClothingSuggestion( weatherInfo: "Rain, 18°C", 1 );
        assertTrue(suggestion.contains("Bring a raincoat")); // Check for raincoat suggestion
    }
}
```

Commit Changes 1 file Implemented , testSunnyWeatherClothing(), to verify light clothing , suggestion and Implemented , testRainyWeatherClothing(), to validate raincoat , recommendation.

Commit Commit and Push... 21:1 CRLF UTF-8 4 spaces

Commits

The screenshot shows a Git commit history for a project named 'ProjectChatbot_Group_D'. The commits are listed by date:

- Apr 06, 2025**
 - Aline Andrade Costa implemented a test method `testSunnyWeatherClothing()` to verify light clothing suggestions.
 - Aline Andrade Costa created a JavaFX-based Chatbot GUI.
- Mar 21, 2025**
 - Sergio Alves dasilva added documentation for the first milestone.
 - Sergio Alves dasilva made changes to the ClothingRecommender class and added a unit test for it.
 - Aline Andrade Costa added the first unit tests for the WeatherAPI.
 - Cynthia Dasilvaroque implemented unit tests to verify clothing recommendations based on different weather conditions.
 - Cynthia Dasilvaroque implemented basic functions of the Chatbot.
 - Aline Andrade Costa created a class ClothingRecommender and implemented a method to return a string.

Sergio – Chatbot Logic and Clothing Recommendations

The class ClothingRecommender was implemented.

The screenshot shows an IDE interface with the following details:

- Project Structure:** The project is named 'SD_ChatBot_GroupD' located at 'C:\Users\Sgio\Desktop\projectchatbot\SD_ChatBot_GroupD'. It contains a 'src' folder with classes like Chatbot, ChatbotTest, ClothingRecommender, ClothingRecommenderTest, WeatherAPI, and WeatherAPITest.
- Code Editor:** The 'ClothingRecommender.java' file is open. The code implements a class 'ClothingRecommender' with methods to get clothing suggestions based on weather and day index, and to give personalized suggestions based on a user's wardrobe.
- Toolbars and Status:** The top bar shows tabs for 'ClothingRecommenderTest', 'ClothingRecommender.java', and 'ChatbotTest.java'. The status bar indicates the current file is 'ClothingRecommender.java'.

```

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.

The screenshot shows an IDE interface with a project named "SD_ChatBot_GroupD.iml". The left sidebar displays the project structure with a "src" folder containing "Chatbot", "ChatbotTest", "ClothingRecommender", "ClothingRecommenderTest", "WeatherAPI", and "WeatherAPITest". The right pane shows the code for "ClothingRecommenderTest.java". Below the code editor is the "Build" tab, which shows a build failure for "SD_ChatBot_GroupD.iml" with 6 errors. One specific error is highlighted: "method getClothingSuggestion in class ClothingRecommender cannot be applied to give 5 errors". The status bar at the bottom indicates the file is 1 sec, 270 ms, and the encoding is CRLF, UTF-8, 4 spaces.

```
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 new*
public void testSunnyMorningSuggestion() {
    String result = ClothingRecommender.getClothingSuggestion(weather: "Sunny", dayIndex: 0);
    assertTrue(result.contains("light clothing"));
    assertTrue(result.contains("sunglasses"));
    assertTrue(result.contains("morning"));
}

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

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

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

The screenshot shows the "ClothingRecommender.java" file in the code editor. The implementation of the "getClothingSuggestion" method is shown, which returns a default suggestion and then adds on suggestions based on the day of the week. The "getPersonalizedClothingSuggestion" method is also partially implemented. The status bar at the bottom indicates the file is 20:69 (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 a project tree on the left and a code editor on the right. The project tree includes a 'src' folder containing 'Chatbot', 'ChatbotTest', 'ClothingRecommender', 'ClothingRecommenderTest', 'WeatherAPI', and 'WeatherAPITest'. The code editor displays 'ClothingRecommenderTest.java' with the following content:

```

1 import org.junit.jupiter.api.Test;
2
3 import java.util.Arrays;
4 import java.util.List;
5
6 import static org.junit.jupiter.api.Assertions.assertTrue;
7
8 public class ClothingRecommenderTest { // Sergio Alves Da Silva +1
9
10     @Test new
11     public void testSunnyMorningSuggestion() {
12         String result = ClothingRecommender.getClothingSuggestion(weather: "Sunny", dayIndex: 0);
13         assertTrue(result.contains("light clothing"));
14         assertTrue(result.contains("sunglasses"));
15         assertTrue(result.contains("morning"));
16     }
17
18     @Test new
19     public void testRainyAfternoonSuggestion() {
20         String result = ClothingRecommender.getClothingSuggestion(weather: "Rain", dayIndex: 1);
21         assertTrue(result.contains("raincoat"));
22         assertTrue(result.contains("Afternoon"));
23     }
24
25     @Test new
26     public void testColdEveningSuggestion() {
27         String result = ClothingRecommender.getClothingSuggestion(weather: "Cold", dayIndex: 2);
28         assertTrue(result.contains("Dress warmly"));
    }

```

The 'Run' tab at the bottom shows the test results: 'Tests passed: 5 of 5 tests - 16 ms'. The command line output shows the tests were run from 'C:\Users\Sgio\jdks\openjdk-21.0.2\bin\java.exe ...' and 'Process finished with exit code 0'.

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

```

Already up to date.

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

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

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

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

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

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git commit -m "The Class 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(-)

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

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

```

A commit was sent to the repository of the project.

The screenshot shows a GitHub commit history for the 'ProjectChatbot_Group_D' repository. The commits are as follows:

- Apr 06, 2025**: The Class ClothingRecommender and the class Test related was implemented with... (sergio91) - c82a87c9
- 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) - f7dcbae7
- Create class ClothingRecommender and implemented with a method to return string (Sergio Alvesdasilva) - fa8d50ba
- Added WeatherAPI class to fetch weather data using Open-Meteo API (Aline Andrade Costa) - 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.

10.Milestone 3

10.1 Goals

- Complete and integrate all project features as per the project brief.
 - Add JUnit testing framework to validate chatbot components.
 - Finalize the GUI to interact with the chatbot visually.

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

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

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

10.2 Junit Tests Integration

- ClothingRecommenderTest:

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

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

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

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

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

→ ChatbotTest:

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

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

→ WeatherAPITest:

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

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

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

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

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

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

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

10.3 Commit List & Branches Tree

Aline -

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

```

40 }
41
42 class GeolocationAPI { 1 usage new*
43 @  public static double[] getCoordinates(String location) { 1 usage new*
44     try {
45         String encodedLocation = URLEncoder.encode(location, "enc: \"UTF-8\"");
46         String urlString = "https://nominatim.openstreetmap.org/search?q=" + encodedLocation + "&format=json&limit=1";
47         URL url = new URL(urlString);
48         HttpURLConnection conn = (HttpURLConnection) url.openConnection();
49         conn.setRequestMethod("GET");
50         conn.connect();
51
52         if (conn.getResponseCode() != 200) return null;
53
54         Scanner scanner = new Scanner(url.openStream());
55         StringBuilder inline = new StringBuilder();
56         while (scanner.hasNext()) {
57             inline.append(scanner.nextLine());
58         }
59         scanner.close();
60     }

```

Commit: 1 modified
Integrate live weather and geolocation APIs for dynamic location-based forecasts

Run: ChatbotGUI ×
WARNING: A terminally deprecated method in sun.misc.Unsafe has been called
WARNING: sun.misc.Unsafe::allocateMemory has been called by com.sun.martini.OffHeapArray (file:/C:/Users/lili/_OneDrive/Área%20de%20Trabalho/Software%20Development%202/Proj...)
WARNING: Please consider reporting this to the maintainers of class com.sun.martini.OffHeapArray
WARNING: sun.misc.Unsafe::allocateMemory will be removed in a future release

Process finished with exit code 0

SD_ChatBot_GroupD > src > WeatherAPI.java

17:32 CRLF UTF-8 4 spaces

Implement chatbot UI and CSS styling for the trip planner.

```

14 public class ChatbotGUI extends Application { ▲ aline andrade costa *
15     private TextField locationInput; // For location input 4 usages
16     private TextField dayInput; // For visit day (1-3) input 4 usages
17
18     public static void main(String[] args) { ▲ aline andrade costa *
19         launch(args); // Launch JavaFX application
20     }
21
22
23
24     @Override ▲ aline andrade costa *
25     public void start(Stage primaryStage) {
26         primaryStage.setTitle("Trip Clothing Planner Chatbot");
27
28         chatContainer = new VBox( v: 10 );
29         chatContainer.setPadding(new Insets( v: 20 ));
30         chatContainer.setPrefWidth(450);
31
32         ScrollPane scrollPane = new ScrollPane(chatContainer);
33         scrollPane.setFitToWidth(true);
34
35         // Input controls
36         startDatePicker = new DatePicker();
37         startDatePicker.setPromptText("Trip Start Date (YYYY-MM-DD)");
38         locationInput = new TextField();
39         locationInput.setPromptText("Enter location...");
40         dayInput = new TextField();
41         dayInput.setPromptText("Visit day (1-3)");
42
43         Button fetchWeatherButton = new Button( s: "Get Weather & Suggestion" );
44         fetchWeatherButton.setOnAction( ActionEvent e -> fetchWeather() );
45
46         HBox inputRow = new HBox( v: 10, startDatePicker, locationInput, dayInput, fetchWeatherButton );

```

Commit: 3 added 2 modified
Implement chatbot UI and CSS styling for trip planner.

Run: ChatbotGUI ×
WARNING: A terminally deprecated method in sun.misc.Unsafe has been called
WARNING: sun.misc.Unsafe::allocateMemory has been called by com.sun.martini.OffHeapArray (file:/C:/Users/lili/_OneDrive/Área%20de%20Trabalho/Software%20Development%202/Proj...)
WARNING: Please consider reporting this to the maintainers of class com.sun.martini.OffHeapArray
WARNING: sun.misc.Unsafe::allocateMemory will be removed in a future release

Build Build Output ×

SD_ChatBot_GroupD > src > ChatbotGUI > main

22:6 CRLF UTF-8 4 spaces

Align(ChatbotGUI): match console Chatbot flow and messaging

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

- Title Bar:** SD_ChatBot_GroupD.Impl > main > ChatbotGUI
- File List:** style.css, ChatbotGUI.java (selected), Chatbot.java, WeatherAPITest.java
- Toolbars:** Commit, Changes (highlighted), and various icons for file operations.
- Code Editor:** The ChatbotGUI.java file contains code for fetching weather and clothing suggestions. It includes imports for `Platform`, `Alert`, and `Optional`. The code uses `Platform.runLater` to update the UI with weather information and clothing suggestions. It also includes a `promptRestart` method to ask the user if they want another trip.
- Status Bar:** 181:8 CRLF UTF-8 4 spaces

```
public class ChatbotGUI extends Application {    private void fetchAll() {        new Thread(() -> {            int idx = i;            Platform.runLater(() -> {                addBotMessage( text: "Weather in " + locations[idx] + " on " + date + ": " + weatherInfo);                addBotMessage( text: "Clothing Suggestion for Day " + visitDays[idx] + ": " + clothingSuggestion);            });            Platform.runLater(() -> {                progressIndicator.setVisible(false);                promptRestart();            });        }).start();    }    // Ask user if they want another trip    private void promptRestart() {        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);        alert.setTitle("Plan Another Trip?");        alert.setHeaderText(null);        alert.setContentText("Would you like to plan another trip? (yes/no)");        ButtonType yes = new ButtonType( s: "yes");        ButtonType no = new ButtonType( s: "no");        alert.getButtonTypes().setAll(yes, no);        Optional<ButtonType> res = alert.showAndWait();        if (res.isPresent() && res.get() == yes) {            resetAll();        }    } }
```

Cynthia -

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

SD ChatBot_GroupD.iml cynthia

Commit Shelf : - WeatherAPI.java WeatherAPITest.java

Changes 1 file misc.xml SD_ChatBot_GroupD.i

Amend @

Added support for fetching weather by date.

Improved location lookup and error handling.

Created unit tests for valid/invalid locations, date formats, and future dates.

Commit Commit and Push... 79 }

// GeolocationAPI gets the latitude and longitude for a given location using Nominatim API.

```
class GeolocationAPI {
    public static double[] getCoordinates(String location) {
        try {
            // Encode the location name to ensure it is compatible with the URL
            String encodedLocation = URLEncoder.encode(location, "UTF-8");
            String urlString = "https://nominatim.openstreetmap.org/search?q=" + encodedLocation + "&format=json&limit=1";
            URL url = new URL(urlString);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            conn.connect(); // Connect to the server

            // If the response code is not successful, return null
            if (conn.getResponseCode() != 200) return null;
            Scanner scanner = new Scanner(url.openStream());
            StringBuilder inline = new StringBuilder();
            while (scanner.hasNext()) {
                inline.append(scanner.nextLine());
            }
            scanner.close();
            // Convert the API response from String to a JSON array
            JSONArray results = new JSONArray(inline.toString());
            if (results.length() == 0) return null;
            JSONObject locationData = results.getJSONObject(index: 0);
            // Return the latitude and longitude as a double array
            return new double[]{locationData.getDouble(key: "lat"), locationData.getDouble(key: "lon")};
        } catch (Exception e) {
            return null;
        }
    }
}
```

Pushed 11 commits to origin/cynthia

2 files committed: Added support for fetching weather by date. Improved location lookup and error handling....

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

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

- Project View:** Shows the project structure for "SD_ChatBot_GroupD". The "out" directory is selected.
- Code Editor:** Displays the "WeatherAPI.java" file. The code implements a static method `getWeather` that takes a location and date, determines a temperature range, and returns a weather description. It also includes a catch block for exceptions and a static method `getWeather` that returns the location directly.
- Run Tab:** Shows the "WeatherAPITest" run configuration.
- Tool Window:** The "Tests" tool window displays the test results:
 - Total tests: 7
 - Passed: 6
 - Failed: 1 (testValidLocationWeather())
- Status Bar:** Shows the file path "SD_ChatBot_GroupD.iml", the current file "WeatherAPITest.java", and the status "Tests passed: 6 of 7 tests".

chatbot now working properly

The screenshot shows the Android Studio interface. At the top, there's a navigation bar with tabs for WeatherAPITest, ChatbotTest, ClothingRecommender, ClothingRecommenderTest, WeatherAPI, and WeatherAPITest.java. Below the navigation bar is a code editor window titled 'WeatherAPITest'. The code contains several test methods for a Geolocation API, including assertions for invalid and empty location strings. A commit message is visible at the bottom of the code editor.

Commit Commit and Push...

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

Run Chatbot x WeatherAPITest x

C:\Users\cycyn\jdks\openjdk-24\bin\java.exe ...
Welcome to the Trip Clothing Planner Chatbot!
I'm Wadrobot! Packing can be tricky but I'm here to simplify your travel planning!
Please enter the start date of your trip (YYYY-MM-DD):

SD_ChatBot_GroupD > .idea > misc.xml

11°C Cloudy Search 43:90 CRLF UTF-8 4 spaces 20:16 ENG GA 25/04/2025

Sergio -

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

The screenshot shows the Android Studio interface with the project 'SD_ChatBot_GroupD' selected. The code editor displays 'ClothingRecommender.java'. The code defines a class 'ClothingRecommender' with a static method 'getClothingSuggestion'. This method takes a weather string and a day index as parameters. It first tries to extract a temperature value from the weather string. If successful, it parses the number between the start of the temperature and the degree Celsius symbol ('°C'). If parsing fails or if no temperature is found, it defaults to 20.0. Finally, it checks if the weather contains 'rain' and appends a rain alert message to the suggestion.

```

// Main class definition
public class ClothingRecommender { 19 usages ▲ Sergio Alves Da Silva +1 *

    // This method returns a clothing suggestion based on the weather description and time of day (dayIndex).
    public static String getClothingSuggestion(String weather, int dayIndex) { 19 usages ▲ Sergio Alves Da Silva *
        StringBuilder suggestion = new StringBuilder(); // Used to build the final suggestion string.
        weather = weather.toLowerCase(); // Normalize weather string to lowercase for easier matching.

        double temperature = 20.0; // Default temperature if parsing fails.

        // Try to extract a temperature value from the weather string.
        try {
            int index = weather.indexOf("°C"); // Look for "°C" in the string.
            if (index > 0) {
                int start = index;

                // Walk backwards to find the beginning of the temperature number (can include negative sign and decimal).
                while (start > 0 && (Character.isDigit(weather.charAt(start - 1)) ||
                    weather.charAt(start - 1) == '-' ||

                    weather.charAt(start - 1) == '.')) {
                    start--;
                }

                // Parse the number between start and "°C".
                temperature = Double.parseDouble(weather.substring(start, index));
            }
        } catch (Exception e) {
            // If anything goes wrong, stick with the default temperature of 20.0.
        }

        // Main clothing suggestion based on weather conditions and parsed temperature.
        if (weather.contains("rain")) {
            // If "rain" is mentioned in the forecast.
            suggestion.append("⚠️ Rain alert! Grab your cutest raincoat and waterproof boots - bonus points if your umbrella");
        }
    }
}

```

SD_ChatBot_GroupD > src > ClothingRecommender > getClothingSuggestion

57:33 CRLF UTF-8 4 spaces

All scenarios in ClothingRecommenderTest were successful tested.

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

- Project View:** Shows the project structure under "SD_ChatBot_GroupD".
- Code Editor:** The active file is `ClothingRecommenderTest.java`. The code defines two test methods: `testRainyMorning()` and `testSnowyEvening()`.
- Run Tool Window:** Displays the results of the "ChatbotTest" run. It shows 5 tests passed in 18ms. The output window shows the command used to run the test and several "WARNING" messages related to restricted Java methods and native access.
- Status Bar:** Shows the current time (15:1), file encoding (UTF-8), and code style settings (4 spaces).

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
// Test class to validate ClothingRecommender's logic across different weather scenarios.
public class ClothingRecommenderTest {
    // Test rainy morning suggestion.
    @Test
    public void testRainyMorning() {
        String result = ClothingRecommender.getClothingSuggestion(weather: "Rain, 15°C", dayIndex: 0);
        assertTrue(result.contains("☔ Rain alert!")); // Checks for main rain message.
        assertTrue(result.contains("morning")); // Ensures the time-specific advice is included.
    }
    // Test snowy evening suggestion.
    @Test
    public void testSnowyEvening() {
        String result = ClothingRecommender.getClothingSuggestion(weather: "Snow, -3°C", dayIndex: 2);
        assertTrue(result.contains("❄ Snowy vibes today")); // Main snowy suggestion.
        assertFalse(result.contains("Evening stroll")); // Ensure casual stroll advice is not shown in freezing.
    }
}
```

Process finished with exit code 0

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

```
MINGW64:/c/Users/Sgio/Desktop/projectchatbot/SD_ChatBot_GroupD
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/chatbot.java
  modified: src/chatbotTest.java
  modified: src/ClothingRecommender.java
  modified: src/ClothingRecommenderTest.java

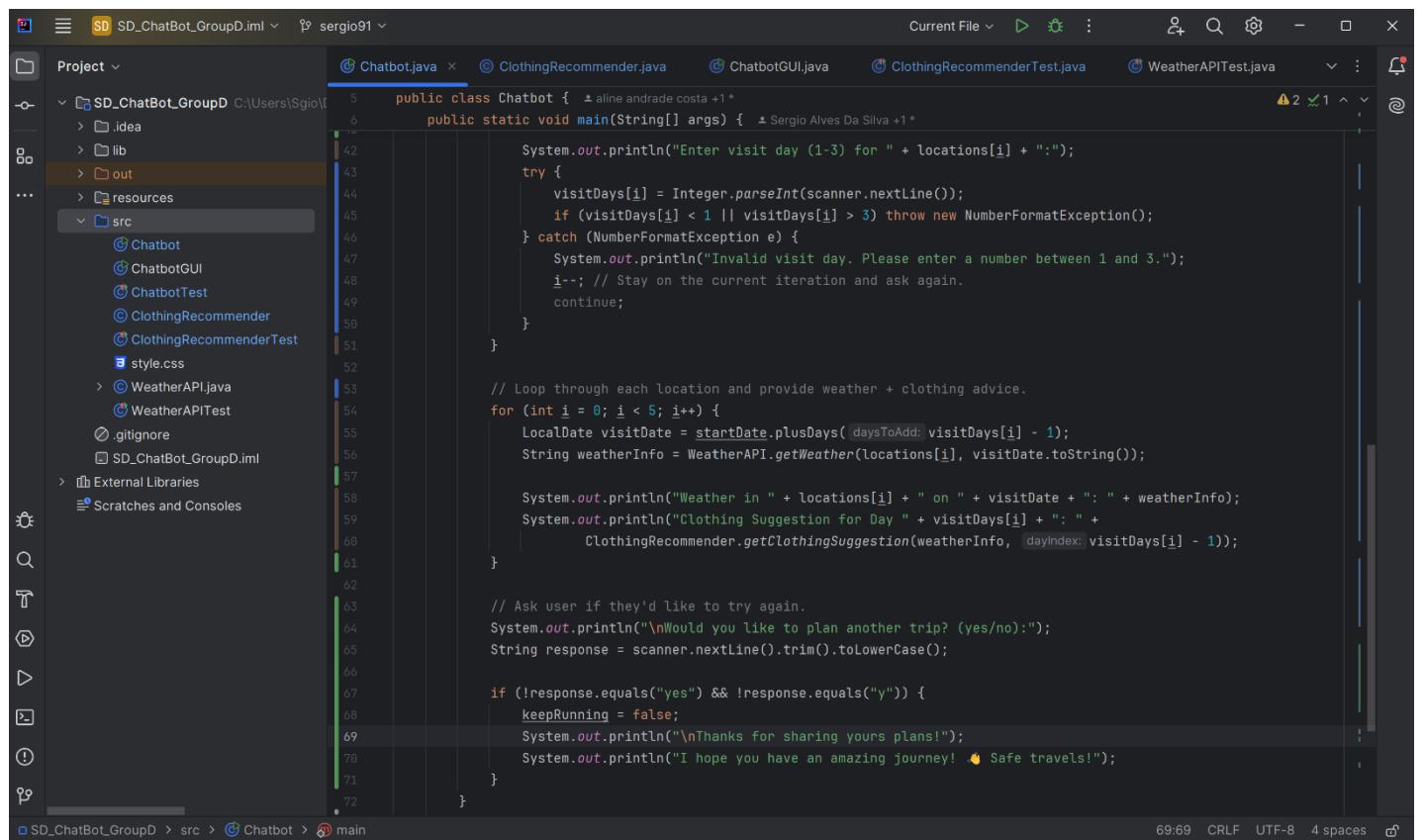
Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git add .

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git status
On branch sergio91
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
  modified: .idea/libraries/json_20250107.xml
  modified: src/chatbot.java
  modified: src/ChatbotTest.java
  modified: src/ClothingRecommender.java
  modified: src/ClothingRecommenderTest.java

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git commit -m "Implemented clothing reccomender and test for more accuracy and successful logic."
[sergio91 62caf12] Implemented clothing reccomender and test for more accuracy and successful logic.
  5 files changed, 163 insertions(+), 132 deletions(-)

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git push origin sergio91
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 20 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 3.56 KiB | 3.56 MiB/s, done.
Total 10 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
```

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



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

- Project View:** Shows the project structure under "SD_ChatBot_GroupD". The "src" folder is selected, containing files: Chatbot.java, ChatbotGUI.java, ChatbotTest.java, ClothingRecommender.java, ClothingRecommenderTest.java, and style.css.
- Editor View:** The main editor window displays the code for Chatbot.java. The code implements a chatbot that asks for visit days and provides weather and clothing advice for each day. It also asks if the user wants to plan another trip.
- Status Bar:** At the bottom right, it shows the file is 69:69, uses CRLF line endings, is in UTF-8 encoding, and has 4 spaces for indentation.

```
public class Chatbot { // aline andrade costa +1*
    public static void main(String[] args) { // Sergio Alves Da Silva +1*
        System.out.println("Enter visit day (1-3) for " + locations[i] + ":");
        try {
            visitDays[i] = Integer.parseInt(scanner.nextLine());
            if (visitDays[i] < 1 || visitDays[i] > 3) throw new NumberFormatException();
        } catch (NumberFormatException e) {
            System.out.println("Invalid visit day. Please enter a number between 1 and 3.");
            i--; // Stay on the current iteration and ask again.
            continue;
        }

        // Loop through each location and provide weather + clothing advice.
        for (int i = 0; i < 5; i++) {
            LocalDate visitDate = startDate.plusDays( daysToAdd: visitDays[i] - 1);
            String weatherInfo = WeatherAPI.getWeather(locations[i], visitDate.toString());

            System.out.println("Weather in " + locations[i] + " on " + visitDate + ": " + weatherInfo);
            System.out.println("Clothing Suggestion for Day " + visitDays[i] + ": " +
                ClothingRecommender.getClothingSuggestion(weatherInfo, dayIndex: visitDays[i] - 1));
        }

        // Ask user if they'd like to try again.
        System.out.println("\nWould you like to plan another trip? (yes/no):");
        String response = scanner.nextLine().trim().toLowerCase();

        if (!response.equals("yes") && !response.equals("y")) {
            keepRunning = false;
            System.out.println("\nThanks for sharing yours plans!");
            System.out.println("I hope you have an amazing journey! 🚀 Safe travels!");
        }
    }
}
```

All functionalities were successful in ChatbotTest.

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

- Project View:** Shows the project structure under "SD ChatBot_GroupD". The "src" folder contains "Chatbot", "ChatbotGUI", "ChatbotTest", "ClothingRecommender", "ClothingRecommenderTest", and "WeatherAPITest".
- Code Editor:** The "ChatbotTest.java" file is open, displaying Java code for testing the "ClothingRecommender" class. It includes tests for sunny, cold, snowy, and rainy weather conditions.
- Run Tab:** The "ChatbotTest" run configuration is selected, showing a green checkmark indicating success. The output window shows:
 - Tests passed: 5 of 5 tests - 20 ms
 - "C:\Program Files\Java\jdk-24\bin\java.exe" ...
 - WARNING: A restricted method in java.lang.System has been called
 - WARNING: java.lang.System::Load has been called by com.intellij.rt.execution.application.AppMainV2 in an unnamed module (file:/C:/Users/Sergio/OneDrive/Área de Trabalho/SD ChatBot_GroupD)
 - WARNING: Use --enable-native-access=ALL-UNNAMED to avoid a warning for callers in this module
 - WARNING: Restricted methods will be blocked in a future release unless native access is enabled
- Status Bar:** Shows "Process finished with exit code 0" and other system information like file path and encoding.

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

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

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

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

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

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git commit -m "Implemented chatbot and test with new functionalities for more accuracy and successful logic."
[sergio91 c63a8bc] Implemented chatbot and test with new functionalities for more accuracy and successful logic.
 2 files changed, 150 insertions(+)

Sgio@Sergio MINGW64 ~/Desktop/projectchatbot/SD_ChatBot_GroupD (sergio91)
$ git push origin sergio91
Enumerating objects: 11, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 20 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 2.48 KiB | 2.48 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_bra
```

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

◆ Branches Tree

→ main branch:

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

→ Aline's branch:

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

Files included in this branch:

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

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

→ Sergio's branch:

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

Files included in this branch:

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

→ Cynthia's t branch:

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

Files included in this branch:

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

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

10.4 Full Log Details

Cynthia – Weather API & Data Handling

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

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

- ◆ Ensured error handling for incorrect locations.

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

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

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

- ◆ Ensured API calls are optimized without exceeding rate limits.

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

Aline – JavaFX GUI Enhancements

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

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

- ◆ Added an icon/logo and improved UI responsiveness.

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

- ◆ Implemented a loading indicator while fetching data.

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

- ◆ Allowed users to save and review past weather reports.

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

Sergio – Chatbot Logic & Clothing Recommendations

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

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

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

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

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

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

- ◆ Wrote unit tests for all chatbot functionalities.

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

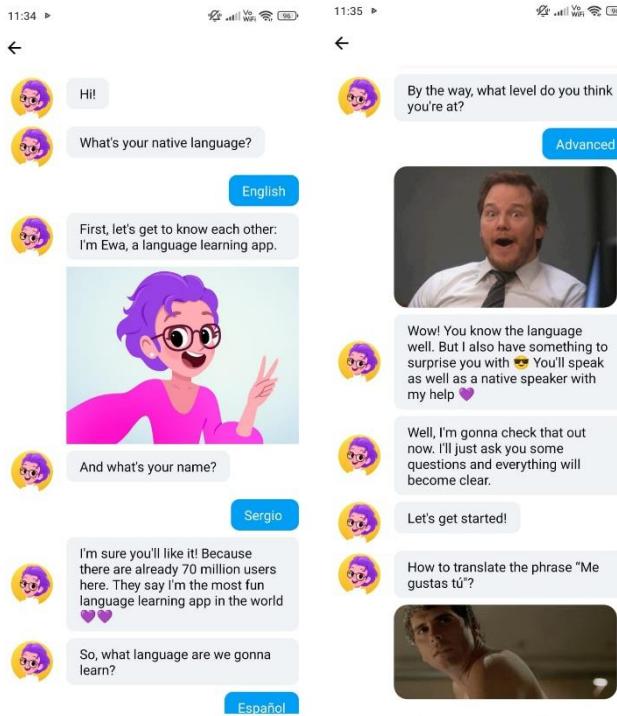
The screenshot shows a GitHub commit history for a project named 'SD_ChatBot_GroupD'. The commits are listed in chronological order from top to bottom. The commits include:

- align(ChatbotGUI); match console origin & main aline andrade costa 9 minutes ago
- Merge branch 'cynthia' aline andrade costa 44 minutes ago
- Resolved issue with future date tr origin & cynthia Cynthia Roque 50 minutes ago
- Merge branch 'sergio91' aline andrade costa Today 19:45
- Implemented chatbot and test w origin & sergio91 Sergio Alves Da Silva Today 03:25
- Implemented clothing recommender and test for mor Sergio Alves Da Silva Today 02:54
- Add JavaFX chatbot UI with input v origin & Aline aline andrade costa 08/04/2025 16:42
- Merge branch 'main' into sergio91 aline andrade costa 08/04/2025 14:31
- Fixing some output as response of the chatbot to in Sergio Alves Da Silva 08/04/2025 14:30
- Merge branch 'Aline' aline andrade costa 08/04/2025 14:22
- Merge branch 'sergio91' aline andrade costa 08/04/2025 14:21
- Implemented some output as response of the chatb: Sergio Alves Da Silva 08/04/2025 14:17
- Implemented some output as response of the chatb: Sergio Alves Da Silva 08/04/2025 13:27
- Implemented some output as response of the chatb: Sergio Alves Da Silva 08/04/2025 13:20
- Added support for fetching weather by date. Improv Cynthia Roque 08/04/2025 13:15
- Implemented some output as response of the chatb: Sergio Alves Da Silva 08/04/2025 13:15
- Implement chatbot GUI and CSS styling for trip plan aline andrade costa 08/04/2025 13:03
- Updating configurations. aline andrade costa 07/04/2025 15:01
- Added the javafx-sdk-24 file aline andrade costa 07/04/2025 14:13
- Integrate live weather and geolocation APIs for dyn:aline andrade costa 07/04/2025 14:00
- Added the documentation with the second milestone aline andrade costa 06/04/2025 22:50
- Updating the configurations. aline andrade costa 06/04/2025 22:17
- Merge remote-tracking branch 'origin/cynthia' aline andrade costa 06/04/2025 22:09
- Merge remote-tracking branch 'origin/sergio91' aline andrade costa 06/04/2025 22:08
- Implement weather data integration and basic unit t Cynthia Roque 06/04/2025 21:00

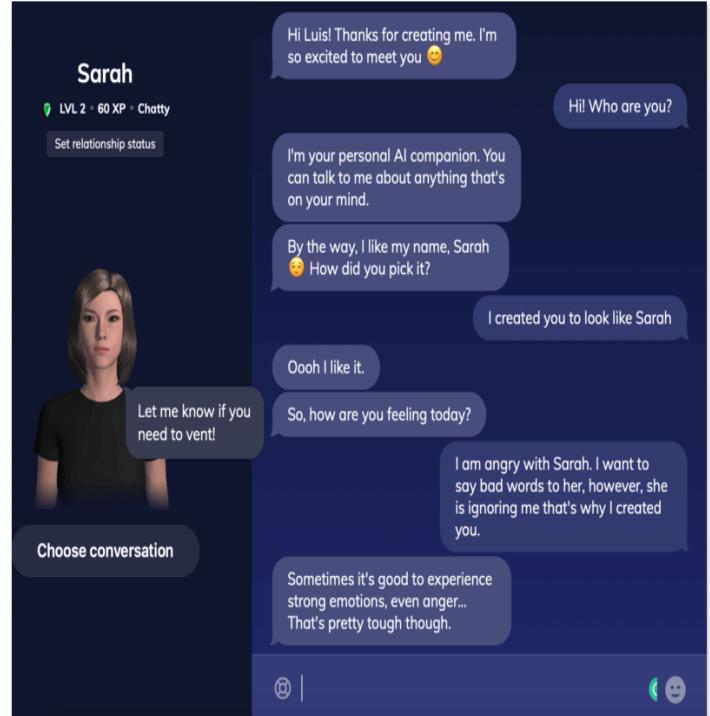
11. Appendix

Reference idea

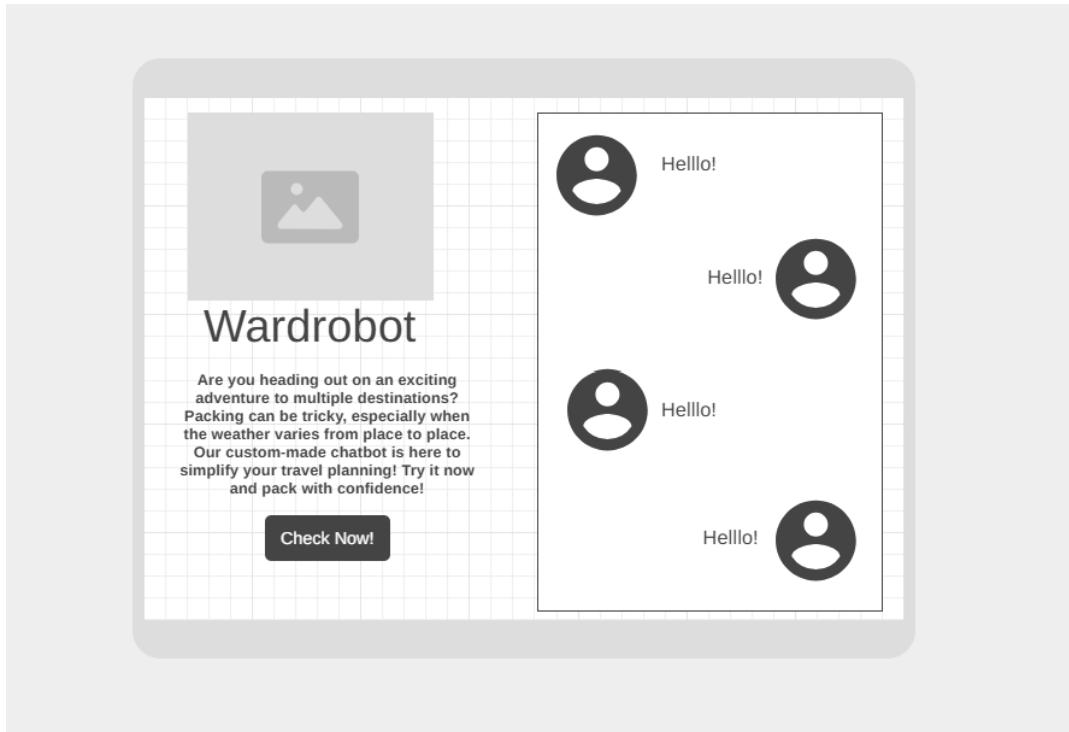
2.



3.



Wireframe (Layout)



12.Bibliography

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

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

Images References

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