

Student name:	Student 1: Oleksii Babii Student 2: Ihor Tryndey Student 3: Dezzy (Ruslan) Tleubergen					
Student number:	Student 1: 3104904 Student 2: 3105023 Student 3: 3091317					
Faculty:	Computing Science					
Course:	BSCH/BSCO/EXCH			Stage/year:	2	
Subject:	Software Development 2					
Study Mode:	Full time	<input checked="" type="checkbox"/>		Part-time	<input type="checkbox"/>	<input type="checkbox"/>
Lecturer Name:	Gemma Deery					
Assignment Title:	Final Documentation					
Date due:	23/05/2024					
Date submitted:	23/05/2024					

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: 

Date: 23/05/2024

Please note: Students **MUST** retain a hard / soft copy of **ALL** assignments as well as a receipt issued and signed by a member of Faculty as proof of submission.

Software Development 2

BSCH-SD2

SkyTalk

23/05/2024

Table of Contents

[Versioning Approach](#)

[Development Process](#)

[Chatbot Logic](#)

[UI Implementation](#)

[Weather API](#)

[External Packages](#)

[Project Setup](#)

[Milestone 1](#)

[Milestone 2](#)

[Goals](#)

[Collaboration](#)

[Milestone 3](#)

[Goals](#)

[Junit Tests integration](#)

[Collaboration](#)

[Bibliography](#)

Versioning Approach

For this project, we adopted the Git Flow workflow for version control, utilizing specific branches tailored to our development needs: ``main``, ``development``, ``feature``, ``API``, and ``GUI``.

Main Branch: The ``main`` branch holds the production-ready code. It is the most stable version of our application and only contains thoroughly tested and approved changes (Git-scm, 2023).

Development Branch: The ``development`` branch is our integration branch where all features and fixes are merged before going into ``main``. It is the basis for feature development and serves as the preparation ground for the next release (Haddad, 2022).

Feature Branches: These branches are created from ``development`` for developing individual features. Each feature branch is named descriptively to indicate the specific feature being worked on. Once the feature is complete, it is merged back into ``development`` after passing code review and testing (Atlassian, 2024).

API Branch: The ``API`` branch is dedicated to backend API development. It is branched from ``development`` and works similarly to a feature branch but focuses specifically on backend functionalities. Changes in the ``API`` branch are merged back into ``development`` when ready.

GUI Branch: The ``GUI`` branch is dedicated to front-end user interface development. Similar to the ``API`` branch, it is branched from ``development`` and focuses solely on the GUI aspects of the project. Once the GUI updates are complete and tested, they are merged back into ``development``.

By using these specific branches, Git Flow allows us to isolate different parts of the project effectively. The ``API`` and ``GUI`` branches enable focused development on backend and frontend features, respectively, while feature branches ensure individual features are developed and tested in isolation. This branching strategy protects the ``main`` branch, keeps the development process organized, and facilitates a smooth and efficient workflow (Haddad, 2022).

Development Process

1. In the beginning, before developing we gather information about the product and how it works, it involves stating all requirements for this project and identifying its business idea. To understand the business idea we discussed what exactly the chatbot has to do.
2. The next is creating the design. it involves creating a plane of development and identifying necessary branches. In this project at the design stage very important to select an API to proceed in developing. At the lower level of development, it includes identifying necessary classes.
3. The implementation is the developing stage where we use the gathered information from the previous stages and just code our chatbot.
4. Verification is one of the most important stages for developing an application because here tests are written and all errors arise. It includes fixing our product.
- 5.

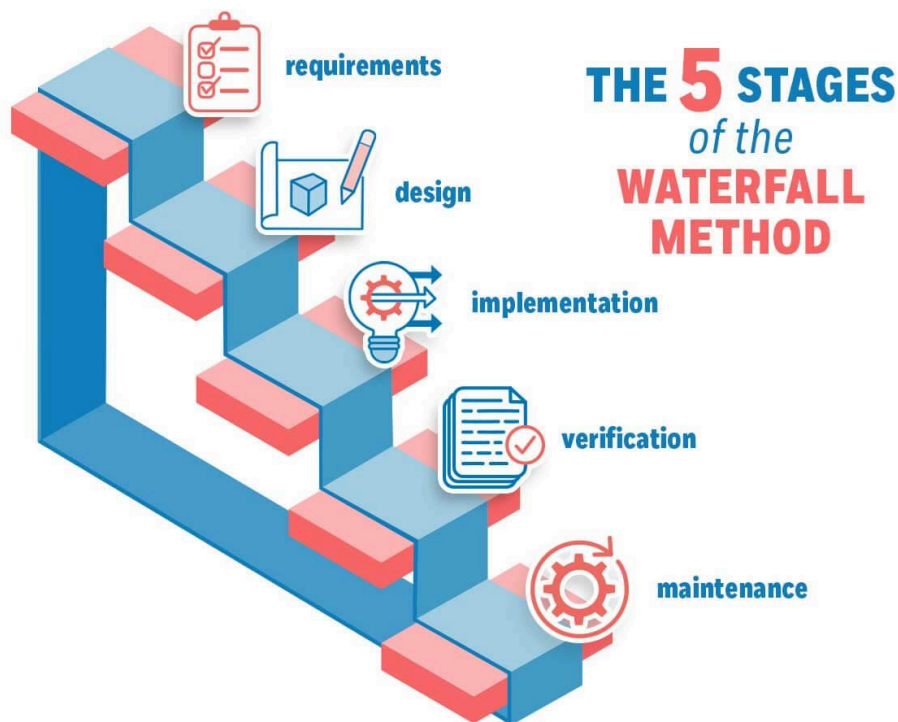
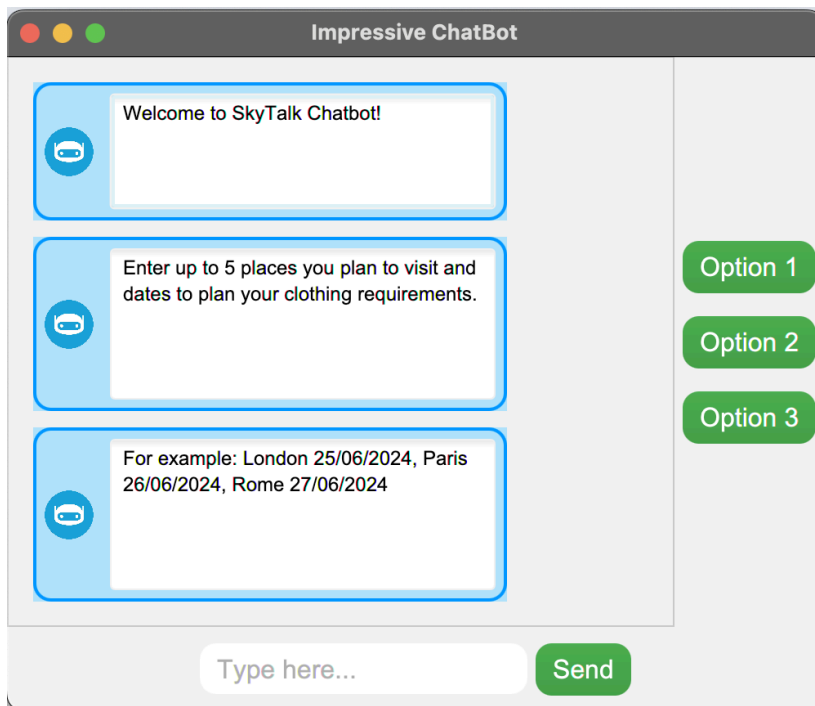


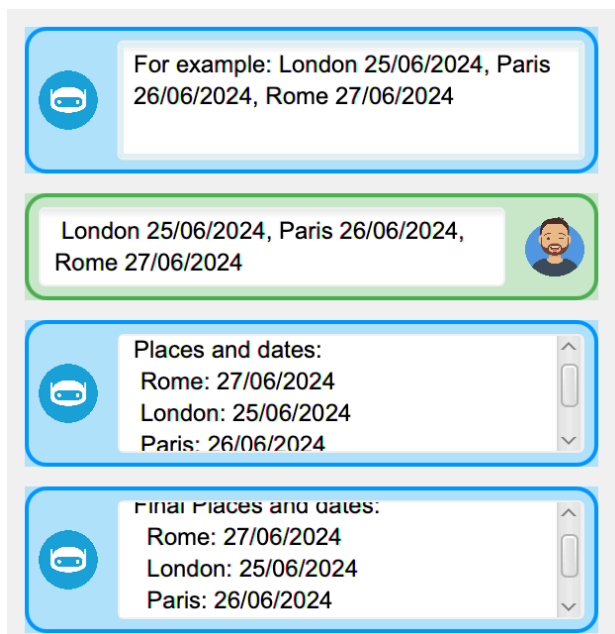
Figure 2 - Waterfall Methodology (<https://management.org/waterfall-methodology>)

Chatbot Logic

1. The application is launched and the chatbot prompts the user to enter up to 5 locations and dates, delimited by commas



2. The user enters the list of locations and dates, and the chatbot confirms them



3. The chatbot generates a list of recommendations for each city on the list

PLAN FOR: Rome temperature for Rome
at 2024-06-27:

minimum temperature: 19.92C |

maximum temperature: 21.1C

cloud cover: 81.74%

precipitation: 114.11cm

My suggestion:

Since the lowest temperature during the
entire trip will be 19.92 degrees
Celsius and the highest 21.1 degrees
Celsius.

Put on a T-Shirt and Light trousers.

There is a high chance of rain during
your trip,

so take an umbrella or a raincoat. ☔

PLAN FOR: London temperature for
London at 2024-06-25:

minimum temperature: 14.18C |

maximum temperature: 16.99C

cloud cover: 35.76%

precipitation: 5.19cm

My suggestion:

Since the lowest temperature during the
entire trip will be 14.18 degrees
Celsius and the highest 16.99 degrees
Celsius.

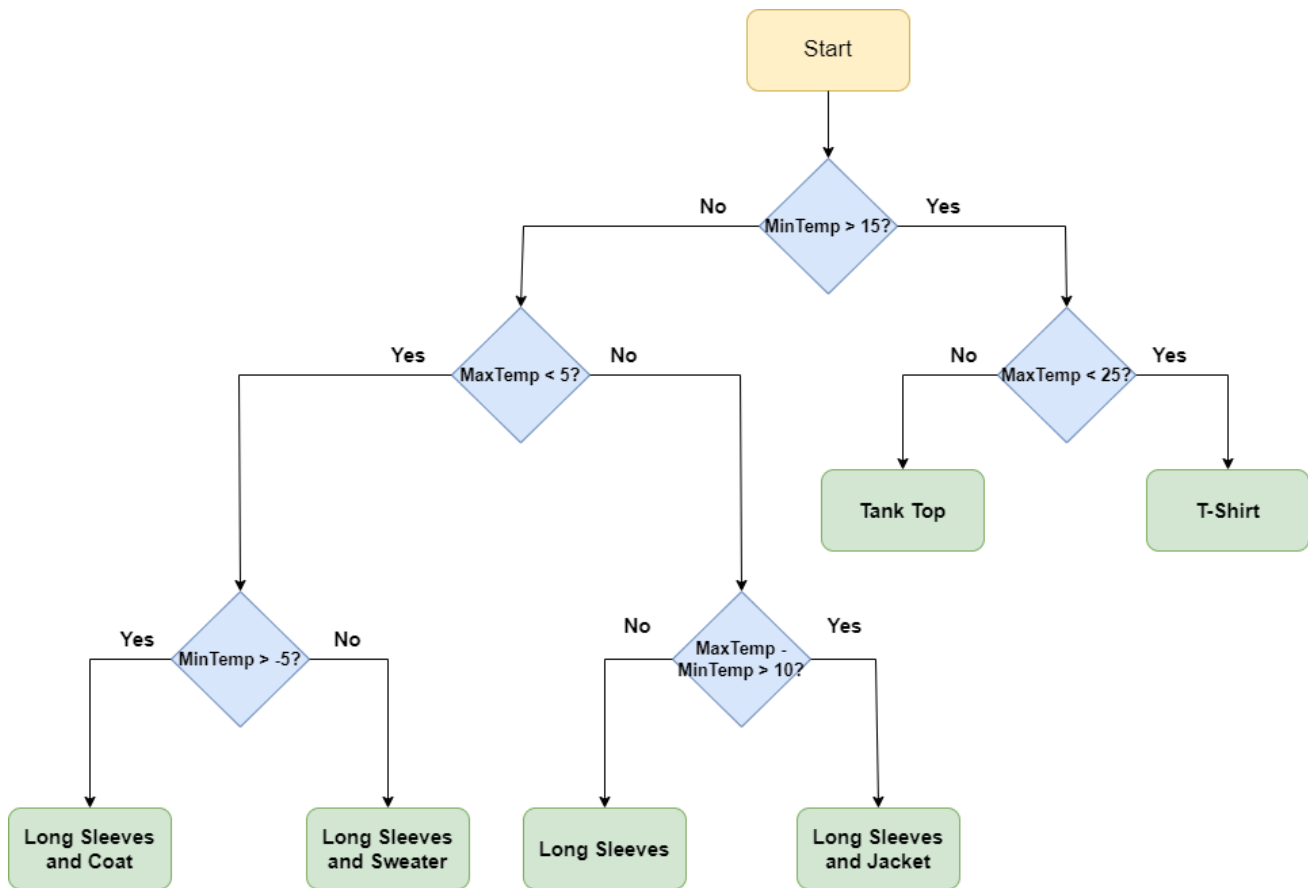
Put on a Long Sleeves and Jeans.

There is a high chance of rain during
your trip,

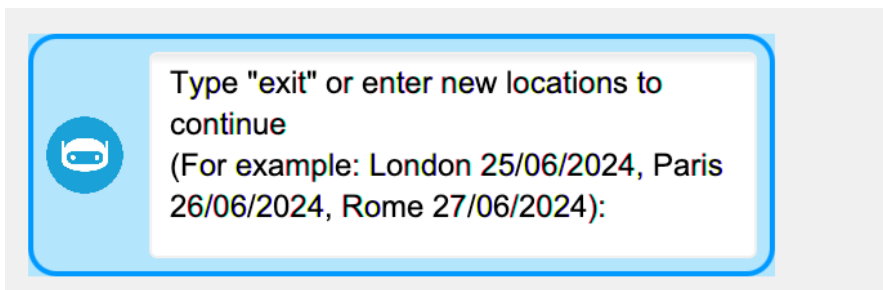
so take an umbrella or a raincoat. ☔

PLAN FOR: Paris temperature for Paris
at 2024-06-26:

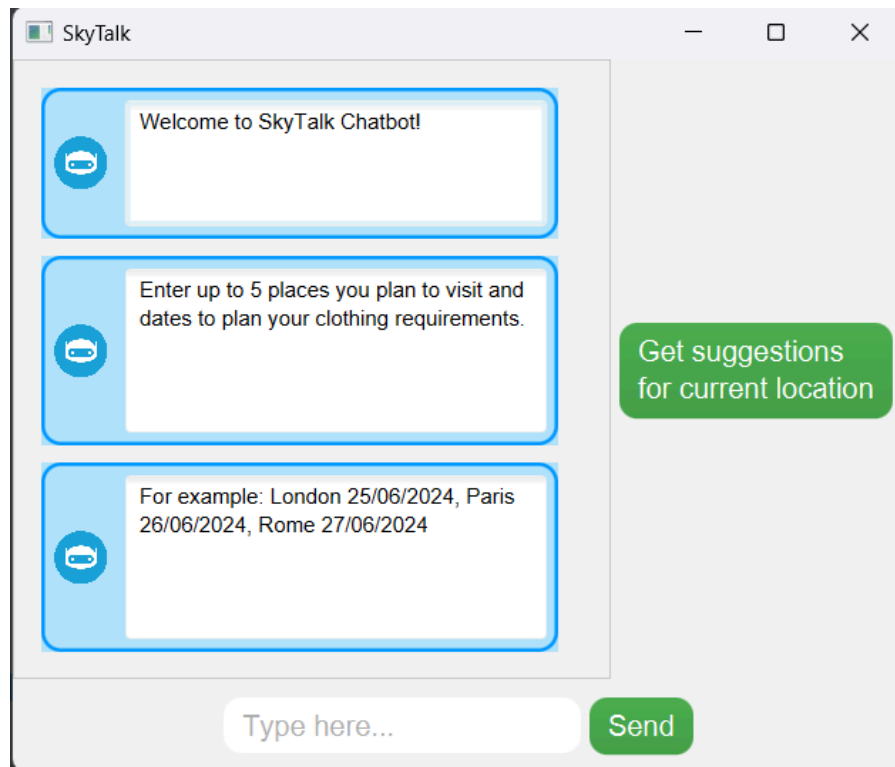




4. The chatbot prompts the user to either quit the program or to continue with a new list of cities



UI Implementation



Graphical user interface (GUI) chatbot application designed to provide users with an interactive and engaging chat experience. The application is built using JavaFX, which offers a rich set of tools for creating modern and responsive GUIs (FXdocs, 2022). SkyTalk helps users with travel-related queries, offering suggestions and information based on user input.

Key GUI Components

Chat Display Area

- **VBox chatPane:** Holds chat messages.
- **ScrollPane scrollPane:** Enables scrolling through messages.

Input Boxes

- **HBox inputBox:** Contains the text input field and send button.
- **TextField inputField:** For user text input.
- **Button sendButton:** Sends the typed message.

Predefined Options

- **VBox optionsBox:** Contains buttons for common user queries.

Event Handling

- **TextField inputField:** Processes user input when Enter is pressed.
- **Button sendButton:** Processes user input when clicked.
- **Option Buttons:** Sets input field to the selected option and sends the message.

Chat Message Handling

SendMessage Method: Manages message sending and receiving.

- Adds user messages to chatPane.
- Clears the input field and triggers chatbot response after a short delay.

AddMessage Method: Displays messages in chatPane

- Differentiates user and bot messages using color and alignment.
- Includes profile images and formatted text.
- Animates message appearance using translate and fade transitions.

By implementing clear and engaging interactions, the project meets its objective of providing a helpful travel assistant. This assignment showcases the integration of GUI design principles with practical event handling and animation techniques, resulting in a polished and user-friendly application.

Weather API

We used OpenWeatherMap's API (<https://openweathermap.org/api>) in order to fetch weather data for a given set of coordinates. OpenWeatherMap supports a variety of API calls, but we decided to use the One Call Daily Aggregation API as it suited our program's needs the most. The Daily Aggregation API is able to get weather data for up to 18 months into the future and returns a JSON object as shown below:

```
/* format: You, last month • update methods with Op
{
  "lat":33,
  "lon":35,
  "tz":"+02:00",
  "date":"2020-03-04",
  "units":"standard",
  "cloud_cover":{
    "afternoon":0
  },
  "humidity":{
    "afternoon":33
  },
  "precipitation":{
    "total":0
  },
  "temperature":{
    "min":286.48,
    "max":299.24,
    "afternoon":296.15,
    "night":289.56,
    "evening":295.93,
    "morning":287.59
  },
  "pressure":{
    "afternoon":1015
  },
  "wind":{
    "max":{
      "speed":8.7,
      "direction":120
    }
  }
}
```

and from this object we got the temperature, humidity, precipitation and wind speed data required in order to generate an appropriate clothing recommendation.

External Packages

We tried using Mockito for testing purposes, but in the end we couldn't get it to work with the tests we'd already written so it had to be abandoned.

The org.json dependency was used to process the weather API's JSON data. org.json includes the objects JSONObject and JSONParser. JSONParser takes a string as an input and converts it to a JSONObject, which allows the individual fields such as temperature or humidity to be accessed and stored into variables.

```
<dependency>
  <groupId>org.json</groupId>
  <artifactId>json</artifactId>
  <version>20240303</version>
</dependency>

<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.13.0</version>
</dependency>

<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-lang3</artifactId>
  <version>3.12.0</version>
</dependency>
```

```
<dependency>
  <groupId>edu.stanford.nlp</groupId>
  <artifactId>stanford-corenlp</artifactId>
  <version>4.2.0</version>
</dependency>
<dependency>
  <groupId>edu.stanford.nlp</groupId>
  <artifactId>stanford-corenlp</artifactId>
  <version>4.5.1</version>
</dependency>
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>1.2.6</version>
  <!-- Or latest version -->
</dependency>
```

```
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-swing</artifactId>
  <version>17.0.6</version>
</dependency>
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-media</artifactId>
  <version>17.0.6</version>
</dependency>
<dependency>
  <groupId>org.controlsfx</groupId>
  <artifactId>controlsfx</artifactId>
  <version>11.1.2</version>
</dependency>
```

```
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-controls</artifactId>
  <version>17.0.6</version>
</dependency>
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-fxml</artifactId>
  <version>17.0.6</version>
</dependency>
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-web</artifactId>
  <version>17.0.6</version>
</dependency>
```

```
<dependency>
  <groupId>net.synedra</groupId>
  <artifactId>validatorfx</artifactId>
  <version>0.4.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.openjfx</groupId>
      <artifactId>*</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

```
<dependency>
  <groupId>com.dlsc.formsfx</groupId>
  <artifactId>formsfx-core</artifactId>
  <version>11.6.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.openjfx</groupId>
      <artifactId>*</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

The setup of this project relies on several external packages to enhance its JavaFX application capabilities. Below is a list of the main dependencies used to implement SkyTalk’s GUI, along with a brief description of their roles and functionalities:

1) **org.openjfx** (<https://mvnrepository.com/artifact/org.openjfx/javafx-controls>)

Description: This is the official JavaFX library provided by the OpenJFX project. It includes the core JavaFX modules necessary for creating modern, graphical user interfaces in Java applications.

Usage: Essential for building the user interface, handling graphics, and UI components such as buttons, text fields, and charts.

2) **org.kordamp.ikonli** (<https://mvnrepository.com/artifact/org.kordamp.ikonli/ikonli-core>)

Description: Ikonli provides a wide range of icon packs for JavaFX, allowing the use of scalable vector icons from various popular icon sets (e.g., FontAwesome, Material Design).

Usage: Adds rich iconography to the application, enhancing the visual appeal and user experience with easily integrable icons in menus, buttons, and other UI components.

3) **org.kordamp.bootstrapfx**

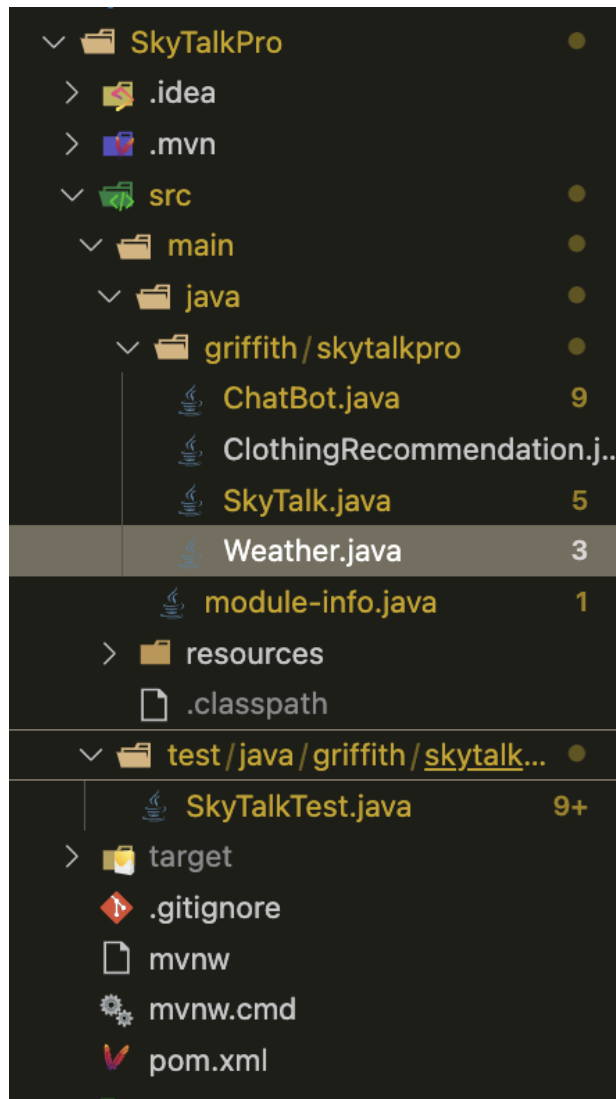
(<https://mvnrepository.com/artifact/org.kordamp.bootstrapfx/bootstrapfx-core>)

Description: BootstrapFX offers a CSS-based framework for JavaFX that emulates the look and feel of Bootstrap, a popular front-end component library for web development.

Usage: Enables developers to style JavaFX applications with a Bootstrap-like appearance, providing a familiar and modern UI design without extensive custom CSS.

Project Setup

The structure of our project is shown below. It was built using an archetype from the “Maven for Java” extension for VS code, which automatically generates a project along with the pom.xml file needed for dependencies. The src folder contains a main folder for the application and a test folder for the tests.

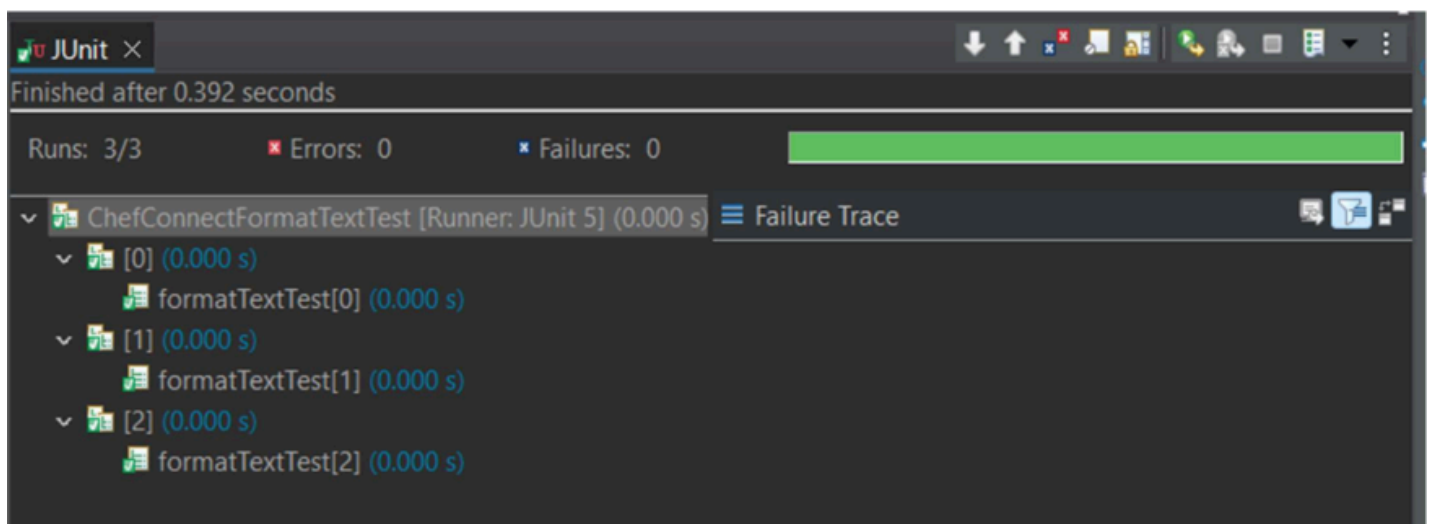
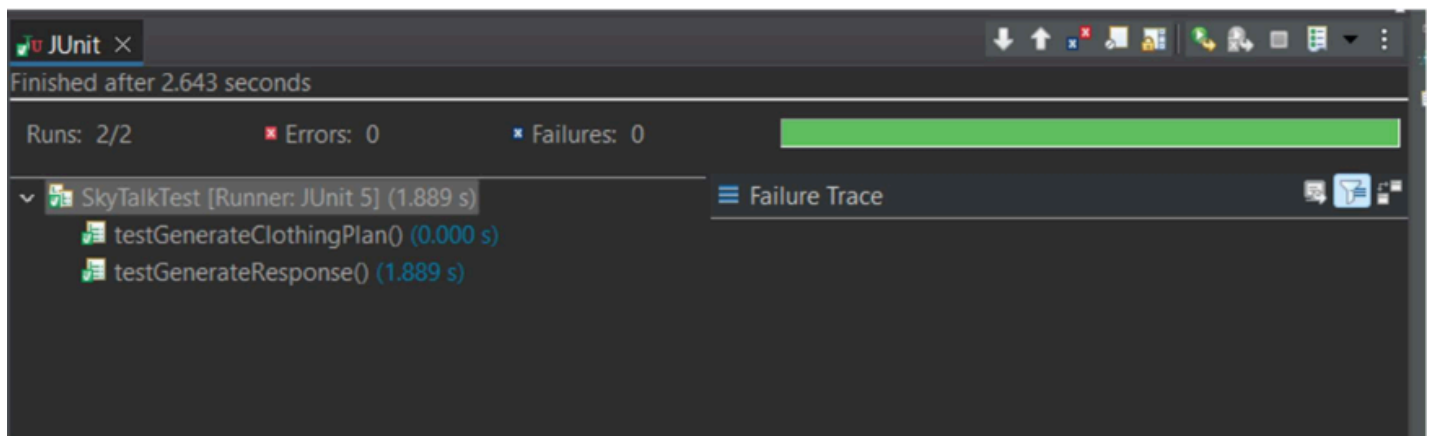


Milestone 1

Goals

- 1) **Functional Interface:** The code aims to provide a functional interface for a chatbot named “SkyTalk” that plans a user's clothing requirements for a trip that involves visiting up to 5 locations.
- 2) **Clothing Recommendation:** Analyze the weather forecast data to generate clothing recommendations for the specified locations and dates, taking into account factors like temperature and weather conditions (e.g., rain, sunshine).
- 3) **User Interaction:** Enable users to interact with the chatbot through a graphical user interface (GUI) implemented using Swing components.
- 4) **Answer generation:** Generate relevant answers based on user input, including suggestions for suitable clothing based on the weather forecast on a particular date, and the need to bring an umbrella or sunglasses. Information about the minimum and maximum temperature during the trip
- 5) **API Integration:** Integrate with external APIs, including WeatherAPI.com to find weather conditions at a specific location on a specified date.
- 6) **Text Formatting:** Format user input and responses for better readability, including text wrapping and proper alignment.
- 7) **Error Handling:** Implement basic error handling mechanisms, such as catching exceptions and displaying error messages when necessary.
- 8) **User Interface Design:** Design a visually appealing user interface with proper layout and styling using colors, fonts, and graphical elements.
- 9) **Ease of Use:** Ensure the chatbot is easy to use for both developers (maintainers) and end-users, with clear instructions and intuitive controls.

JUnit tests



Collaboration

Student 1 (Oleksii Babii):

In this project, I implemented the logic of the bot using an API that recommends clothes to the user and provides a strong connection between all aspects of the project. Also, I prepared to test some methods that check the output from the database. In the team, I worked with the API branch.

```
commit 40ce80192c7e05507710326e9a9a11e6220ca030
Merge: af5bd27 dabfbc8
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Sat Apr 13 16:23:43 2024 +0100

    SkyTalk 1.0

commit af5bd279fb8dd44ffaa736b3264ac241677145b6
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Sat Apr 13 15:20:27 2024 +0100

    feat: synchronize chat bot with GUI

commit f467548c84f368148d9197a30b8eedcc8fe10fe6
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Sat Apr 13 12:12:27 2024 +0100

    feat: create a GUI class

commit df1da596a4380239fa63fccda308ed18d1463bb4
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Sat Apr 13 08:59:51 2024 +0100

    feat: implement generateClothingPlan() method

commit f2a24f7081d7c2ef95849e646ff96d707fd1cb98
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Sat Apr 13 07:03:44 2024 +0100

    test: add a unit test for the generateClothingPlan() method

commit 748166b98915db4ea27f5302c0729bbc3c82cb19
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Thu Apr 11 17:29:21 2024 +0100

    fix: add error handling in takeUserInput() method

commit e1d3be56f4a61e66f3329ca34560a91ac4820139
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Thu Apr 11 11:22:36 2024 +0100

    feat: implement takeUserInput() method

commit 413f05dd3ae128d8dbd177940a9971eb793db4be
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Thu Apr 11 09:14:14 2024 +0100

    feat: create a new chatbot project, connect to the Weather API, add the main() method
```

Student 2 (Ihor Tryndey):

In this project, I was responsible for the graphical user interface and some of the tests that related to the interface and string operations. I tried to implement a parametrized constructor. I was expected to work on the GUI branch and implement the window interface and all other elements including buttons, text area, text field, and scroll bar. Furthermore, I created rounded button and text field classes where using graphical tools I painted them round and with extra features. Also, I was in charge of adjusting the application window, score bar and text area.

```
commit dabfbc8058f19f8ea8d14ab954a422271d375286
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Sat Apr 13 11:18:05 2024 +0100

    generete response implementation

commit 2ec94fdb941b39fff2906f96fd389d30fba21894
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Sat Apr 13 11:16:02 2024 +0100

    generate response unit test addition

commit 38697e22366699d3d03b48a8c12e08cfbcfba854
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Fri Apr 12 23:56:03 2024 +0100

    identify weather condition by appling API

commit 8c89ec118532a3a6da921b83362b6baff576fb4d
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Fri Apr 12 20:43:02 2024 +0100

    Parse JSON response

commit 45f63768e4d215bb3f83262041c8907e365e60aa
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Fri Apr 12 20:40:34 2024 +0100

    getForecast method implementation

commit 80074b05fffdb107b6b442edb09a9bbef638a378
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Mon Apr 1 23:44:47 2024 +0100

    recreation of class Chatbot

commit ddb33fca1e8079c828606abe79ddcf341a8ffdcc (origin/gui, gui)
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Thu Mar 28 21:35:41 2024 +0000

    formatTextTest parametrised test implementation

commit 55ae21f9fd9e5cc17163feeca790c70628a5212b
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Wed Mar 27 15:09:37 2024 +0000

    parametrized test template implementation
```

Milestone 2

Goals

The goals for this milestone were to:

1. Update the GUI to make it more intuitive for users by updating the visuals and providing an options menu, and
2. Simplify the process of getting a forecast for a given city and date. We rewrote the weather-related methods in Skytalk.java and added more comments so that there would be less confusion as to what each bit of the code does.

Collaboration

Student 1: Oleksii

In this review, my attention was focused on developing the design and functionality of the GUI using a modern approach that includes the JavaFX software platform. Interface design, colours and placement of fields and buttons. My work also includes the implementation of animations and the development of user-friendly designs. Setting up the project and adding dependencies, and plugins. Customizing buttons and popups for input and output. Creating a modern design using CSS.

```
Home@DESKTOP-QL2S8TT MINGW64 ~/Desktop/sky/SkyTalk (feature)
$ git log --author="oleksii.babii"
commit a2c75d3c413d5e36c8ff6180d1dcc1b0244034a1
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date:   Wed Apr 17 19:37:29 2024 +0100

    feat: add external css resources and photos

commit 9b7032e6a50304d9e32b493683f98d37bd0406d8
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date:   Wed Apr 17 04:03:00 2024 +0100

    fix: fix the bug in start() method

commit c4dd4e89007b23884989c11ae1c902245e88763f
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date:   Tue Apr 16 22:59:18 2024 +0100

    feat: add 'send' button in start() method

commit 0495bce7e48351ab4f57860fbff2f48014717397
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date:   Tue Apr 16 22:54:15 2024 +0100

    feat: implement start() method to display frame

commit 5976379e34cd520563621d77b5b47f162f9ec73b
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date:   Tue Apr 16 22:47:25 2024 +0100

    feat: add a ChatBot class to represent the logic

commit 3fda4312de5bddcf64a4efccd1909884b8820b2e
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date:   Tue Apr 16 20:24:55 2024 +0100

    feat: set up default javaFX Maven project
```

Student 2: Igor

In this project, I take part in the development of the GUI using the JavaFX framework. Especially I implemented the addMessage and sendMessage methods. Also I take part in the GUI design and make it more user friendly. The GUI creation is important to implement, because it allows people, who are not friendly with programming and command prompt to work with your application code. Also I made an animation for messages to appear in the chat.

Last but not least, I take a part in concatenation of collaborators work to make it look like as a whole project.

```
commit 3dd53f0e6bbc053f0078112b7d93bf8eefbfe522
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Thu Apr 18 02:54:08 2024 +0100

    troubles shooting

commit 3dd53f0e6bbc053f0078112b7d93bf8eefbfe522
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Thu Apr 18 00:30:07 2024 +0100

    worked version

commit db295687fe2eb7fd3c4df2262d03b1c27b056776
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Thu Apr 18 00:15:25 2024 +0100

    format changes

commit d6fd40743abf345cc9797751320e6066ecd8bfd0
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Thu Apr 18 00:11:44 2024 +0100

    implementation of API

commit e0f51e82c16c4fcc7b6bea7e24678a85fc66500a
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Thu Apr 18 00:10:23 2024 +0100

    welcome message and input method
```

```

Home@DESKTOP-QL2S8TT MINGW64 ~/Desktop/sky/SkyTalk (feature)
$ git log --author="Ihor Tryndey"
commit 08d578dfe92164e67cf034ddc677e70191b06045
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Wed Apr 17 18:18:25 2024 +0100

    comments for addMessage method

commit 0871fb7b91be9a37e6c2d0aca4568ad05409c8e9
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Wed Apr 17 18:14:15 2024 +0100

    animation optimization

commit 23acac9f42ecc6b5c4c75fcd25d71936693b65a2
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Wed Apr 17 18:07:49 2024 +0100

    addMessage method implementation

commit b21dc2fd03e1c5710e0e4240045b680ebc826ec8
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Wed Apr 17 14:20:29 2024 +0100

    fixing PauseTransition inside sendMessage

commit 543bfb2ad85e2aed76308764cdcfa9634f42c2fa
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Wed Apr 17 06:16:11 2024 +0100

    sendMessage method implementation

```

Student 3: Dezzy (Ruslan)

I wrote most of the documentation for this milestone report and worked on implementing OpenWeatherAPI into this project, in order to streamline the forecasting process, as the previous iteration of this chatbot relied on methods that were difficult to decipher. Using OpenWeatherAPI, it's very easy to get forecast data just from a pair of coordinates and a date. This forecast data can then be filtered to show only the most important information like the weather conditions and the temperature. To aid the forecast API, I made use of another API called GeoTimeZone that returns the timezone for a given set of coordinates, so that the date that's passed to the forecast method is offset accordingly. There were a lot of different coordinate-to-timezone APIs to choose from. I went with GeoTimeZone as it's very simple in its functionality and the only call it has is exactly what I needed.



Once the forecast data is ready, the temperature and weather conditions will be passed to a clothing recommendation method that'll return the appropriate string response.

Based on recommendations given on the previous review, I made sure to be more consistent in adding comments on what each part of the methods I wrote do.

```

update methods with OpenWeatherAPI, complete implementation of feature branch methods into development branch
desmondbergen committed 2 minutes ago

```

5fa821c  

```
MacBook-Air:Skytalk dessie$ git log
commit c4471b5eef5d4509a8bb710a838bc06c8d967122 (HEAD -> development, origin/development)
Author: dessie <lynxes.desmond@gmail.com>
Date: Thu Apr 18 10:05:44 2024 +0100

    fix issues with JSON dependency, rewrite methods
```

```
MacBook-Air:Skytalk dessie$ git log
commit a4de260c0f426a7e9f8c6bc5be4bfd93db7120cf (HEAD -> feature, origin/feature)
Author: dessie <143977574+desmondbergen@users.noreply.github.com>
Date: Wed Apr 17 22:02:55 2024 +0100

    Delete obsolete src directory

commit 47d62730ea7cf009aa081398b2d68b6e011f764c
Author: dessie <lynxes.desmond@gmail.com>
Date: Wed Apr 17 21:53:28 2024 +0100

    transfer new methods to new version of bot

commit 5140aba233348b9031386ada0a6c72341b2c7823
Author: dessie <lynxes.desmond@gmail.com>
Date: Wed Apr 17 21:48:12 2024 +0100

    implement new weather forecasting methods and tests for those methods
```

Links to APIs used

OpenWeatherAPI: <https://openweathermap.org/api>

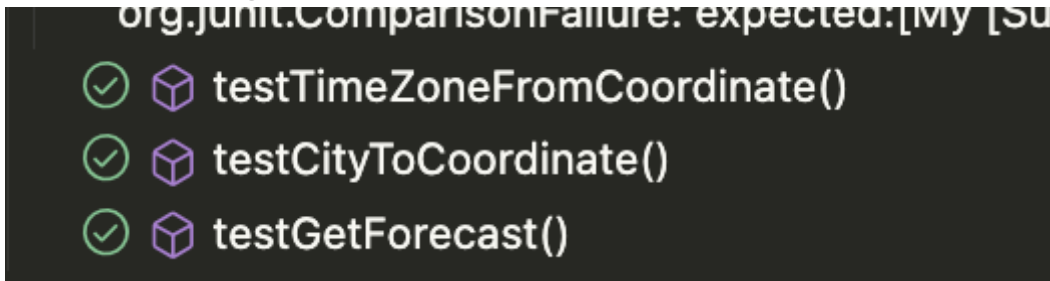
GeoTimeZone: <https://www.geotimezone.com/>

Milestone 3

Goals

Separate the big Skytalk.java class into individual components and write test methods.

Junit Tests integration



Collaboration

Student 1: Oleksii Babi

Taking into account the feedback from the previous review, I restructured the code using Object-Oriented Programming (OOP) principles by splitting one class into two separate classes for the logic and the GUI with an aggregation relationship. Additionally, I introduced a run() method to encapsulate the main functionality.

Moreover, I implemented a feature to provide clothing suggestions based on the current location and time. To achieve this, I created a CurrentLocation class and utilized the IPinfo API to retrieve the city name and the IPify API to determine the IP address. This allows you to receive dynamic recommendations adapted to the user's current location.

```
git
User@LAPTOP-OS0AHPB5 MINGW64 ~/chefConnect/Chef_Connector (development)
$ git log
commit 3701f5b1607f2e1a68338d9c5f3875eb9f742773 (HEAD -> development, origin/development)
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Fri May 10 22:47:52 2024 +0000

    refactor: determine the current location more precisely by using the API

commit 2024ca2d21844e06656841a2fcddea4dc26ae35a
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Fri May 10 22:08:19 2024 +0000

    fix: add getCurrentCity() method to find city name

commit c6267d6b5d77566d4aeebc34f48b3a3db130cf56
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Fri May 10 21:51:03 2024 +0000

    feat: implement the CurrentLocation class to get the user's current location

commit c7a10cde8d7b8b9a1cdaccc16f660ddb31688b51
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Fri May 10 20:29:52 2024 +0000

    feat: Implement an optional button to suggest clothes for the current location

commit 32ff68f69ac1b3f11fa838abfd43be9ba33fd012
Merge: 3abed34 7e74d35
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Fri May 10 20:00:32 2024 +0100

    Merge branch 'development' of https://github.com/ihortry/Chef_Connector into development

commit 3abed345d4344abe171cf7648552d1260ec711c9
Author: oleksii.babii <oleksii.babii@student.griffith.ie>
Date: Fri May 10 20:00:19 2024 +0100

    feat: use OOP techniques to divide the code into several classes with aggregation relationship
```


Student 2: Ihor Tryndey

In this project, I made some refactoring of the code to make it more readable and easy for other members of the team to read it. Also, I made some changes in the animation of messages to make it look more smooth. Additionally I helpt to make integration of code to make gui work with api. Last but not least, I made comments to SkyTalk class and try to separate some aspects of code in different classes.

```
Home@DESKTOP-QL2S8TT MINGW64 ~/Desktop/ChatBot/SkyTalk (development)
$ git log --author="Ihor Tryndey"
commit a895dc3437391c0a13f119bc3c3a24c868ed9633 (HEAD -> development, origin/development)
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Sat May 11 00:33:02 2024 +0100

    integration of code

commit 0b7aa64f50ce54ca1105d4aebf9adc9699f3a1ac
Merge: 545cd1e 3701f5b
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Sat May 11 00:24:32 2024 +0100

    code integration

commit 545cd1ed9c5080a9918114e3700a4acc09819869
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Sat May 11 00:02:07 2024 +0100











    make code more readable

commit 2be26e167d8785344ab04912574f9c8bdce0665c
Author: Ihor Tryndey <igortryndey@gmail.com>
Date: Fri May 10 23:57:36 2024 +0100

    removing unused code
```

Student 3:Dezzy

I made the clothing recommendation part of Skytalk.java into a separate class and wrote several test functions for testing Weather.java’s methods, and a method for testing the clothing recommendation method. I tried to also write tests for the Skytalk class, but there were errors that I wasn’t able to resolve and I’ve included them as a comment in the test file.

Commits on May 10, 2024		
filter strings to remove newlines in testGenerateClothingPlan()	7e74d35	 
desmondberg committed 1 minute ago		
add testGenerateClothingPlan(), test failed	8346288	 
desmondberg committed 12 minutes ago		
add testGetForecast() method, test passed	9869f58	 
desmondberg committed 4 hours ago		
add testCityToCoordinate() method, test passed	ca6ec3e	 
desmondberg committed 5 hours ago		
add testTimeZoneFromCoordinate() function, test passed	88d66e7	 
desmondberg committed 5 hours ago		

Commit Logs

Bibliography

OpenWeatherApi - One Call API 3.0 Daily Aggregation. Available at:

https://openweathermap.org/api/one-call-3#history_daily_aggregation

Atlassian, 2024. *Comparing Git workflows: What you should know.* Available at:

<https://www.atlassian.com/git/tutorials/comparing-workflows#:~:text=Feature%20branching&text=The%20core%20idea%20behind%20the,without%20disturbing%20the%20main%20codebase.>

Git-scm, 2023. *3.1 Git Branching - Branches in a Nutshell.* Available at:

<https://git-scm.com/book/en/v2/Git-Branching-Branched#:~:text=The%20default%20branch%20name%20in,branch%20pointer%20moves%20forward%20automatically.&text=The%20%E2%80%9Cmaster%E2%80%9D%20branch%20in%20Git%20is%20not%20a%20special%20branch.>

Haddad, R., 2022. *What Are the Best Git Branching Strategies.* Available at:

<https://www.abtasty.com/blog/git-branching-strategies/>

Haddad, R., 2022. *What Are the Best Git Branching Strategies.* Available at:

<https://www.abtasty.com/blog/git-branching-strategies/>

FXdocs, 2022. *JavaFX Documentation Project.* Available at: <https://fxdocs.github.io/docs/html5/>