**Status Summary:**
Title: Climate Explorer

Group Members:
1. Kaile Suoo
2. Manas Gupta

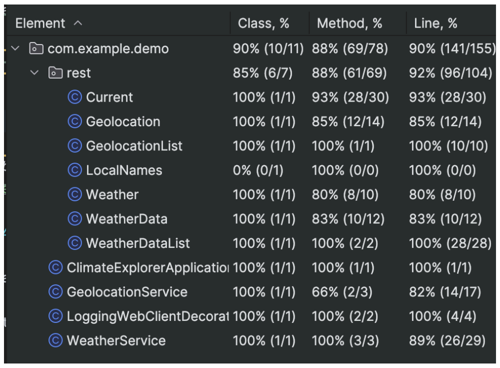
Work Done:
- Manas
    - Implemented the Geolocation API that returns the latitude and longitude and longitude of a given location. This information will be used to gather weather information for a particular location
    - Initialized the corresponding Java objects for our JSON results from the API calls.
    - Modified the API calls to implement the Decorator pattern.
- Kaile
    - Implemented weather API that returns a detailed report on the current weather.
        - This can be changed to return future weather conditions depending on user input. For example, if the user wanted to find predicted weather they can filter it by hourly, daily, etc.
    - Created and formatted page that displays weather information for a given location. Currently, the results page for a given location displays the description/type of weather and displays a detailed report on measurements for the weather below.
    - Extracted the coordinate data out of the geo API that can be used as a parameter for the weather API call.
    - Wrote test cases to test if the application is launching as expected and making sure that we are getting the expected data from the API calls.

Changes/Issues Encountered:
- The API for weather is only able to take coordinates (latitude and longitude) as parameters for the query, so we had to find a way to get the coordinates from the location that the user would enter
- We used another API call for geographic location to get this data, which users can use to find out more about the city they input.

Patterns:
- Singleton: In our application, the singleton pattern ensures that we only have one instance of the services. This makes it easy to manage each of the services and access them when we are using them for the API calls.
- Builder: The webclient builder ensures that instances of the WebClient class are constructed in a flexible and configurable manner, allowing developers to specify properties such as base URL, headers, timeouts, and other configurations needed for making HTTP requests.

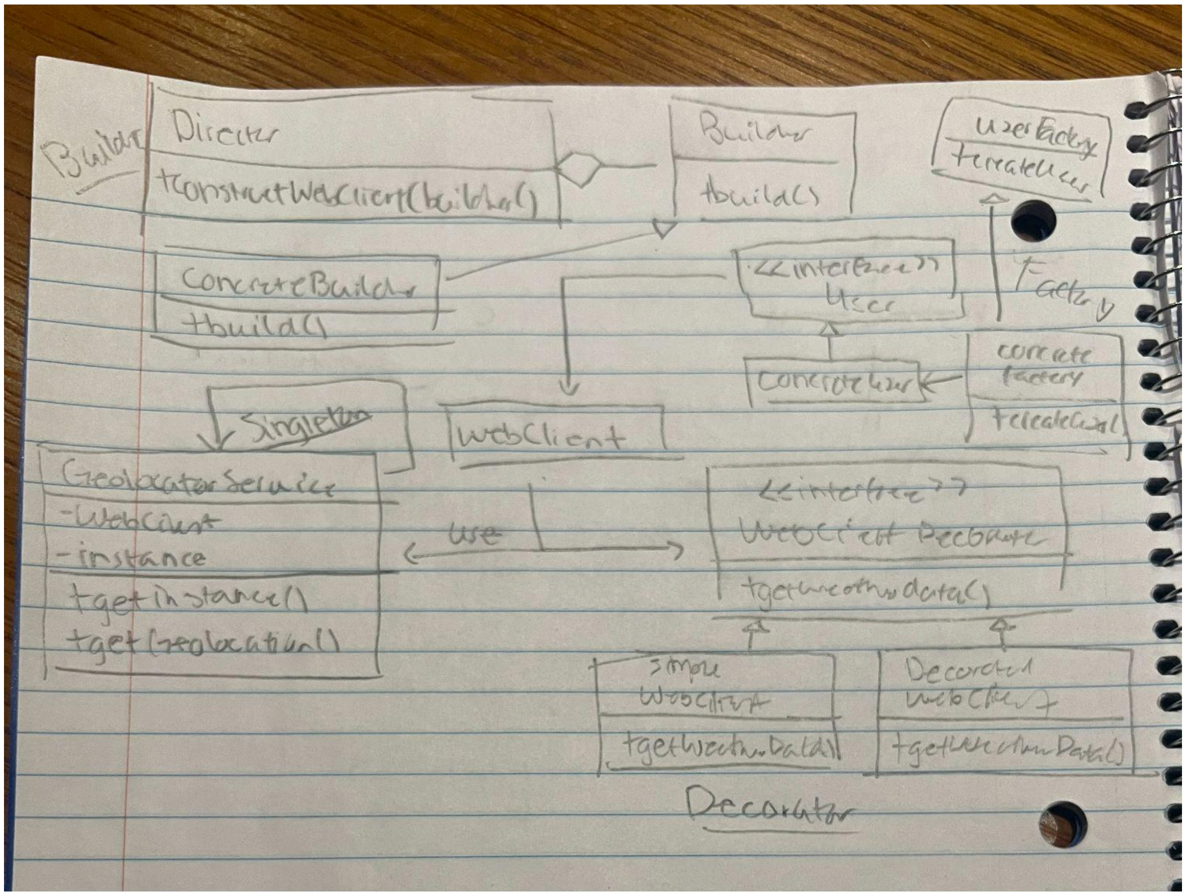
- Decorator - In our application, the Decorator Pattern is utilized to enhance the behavior of the WebClient class dynamically. Specifically, we've implemented a LoggingWebClientDecorator class that wraps around the original WebClient instance and adds logging functionality.

Coverage:



| Element | Class, % | Method, % | Line, % |
|---|---|---|---|
| com.example.demo | 90% (10/11) | 88% (69/78) | 90% (141/155) |
| rest | 85% (6/7) | 88% (61/69) | 92% (96/104) |
| Current | 100% (1/1) | 93% (28/30) | 93% (28/30) |
| Geolocation | 100% (1/1) | 85% (12/14) | 85% (12/14) |
| GeolocationList | 100% (1/1) | 100% (1/1) | 100% (10/10) |
| LocalNames | 0% (0/1) | 100% (0/0) | 100% (0/0) |
| Weather | 100% (1/1) | 80% (8/10) | 80% (8/10) |
| WeatherData | 100% (1/1) | 83% (10/12) | 83% (10/12) |
| WeatherDataList | 100% (1/1) | 100% (2/2) | 100% (28/28) |
| ClimateExplorerApplicatio | 100% (1/1) | 100% (1/1) | 100% (1/1) |
| GeolocationService | 100% (1/1) | 66% (2/3) | 82% (14/17) |
| LoggingWebClientDecorat | 100% (1/1) | 100% (2/2) | 100% (4/4) |
| WeatherService | 100% (1/1) | 100% (3/3) | 89% (26/29) |

**UML Class Diagram:**



All implemented except for Factory Pattern

**BDD Scenarios:**
As a user I want to be able to see information about a city's geographic location and current/upcoming weather when I
Scenario 1:
Given that I have logged in, when I input a city in the search bar and search for geographic location (geo route), then I will find the top matching results and their coordinates.
Scenario 2:
Given that I have logged in, when I input a city in the search bar and search for current weather, then I will see the current summary of the weather status and more information about temperature, wind, etc.

**Plan for Completion:**
We currently have the backend working and have the desired output. We also have three of the four design patterns that we planned to implement currently in. To complete the project, we would need to finish implementing functionality that allows users to access the features in our application and design patterns. The two main ones that we still have to implement are a login page and the search bar. When we create a login/register function, we plan to use the factory pattern to create the objects for an account when a new user registers. The search bar would allow users to input a city as a string and select different filters depending on the results that they want. Their input would then be passed into either the geolocator or weather functions to make an API call and redirect the user to the results page. After implementing these two features, the functionality of the application would be completed.

**Demonstration:**
https://drive.google.com/file/d/1xMoLzoFTHR7SLLMD4Isy765iEJ9MesCJ/view?usp=sharing