[Group3207]

Project document.

Project Name:

WeatherApp

Team Members:

Ayman Asad Khan | An Cao | Areeba Nadeem

Project Description:

The Java based desktop Weather App project aims to create a functional program that retrieves weather data from OpenWeatherMap and presents it through a JavaFX graphical user interface (GUI). This application assists users in accessing weather forecasts for different locations, saving favorite locations, visualizing and displaying weather information based on time intervals. The Model-View-Controller (MVC) architecture organized the software development into three interconnected components: Model, View, and Controller.

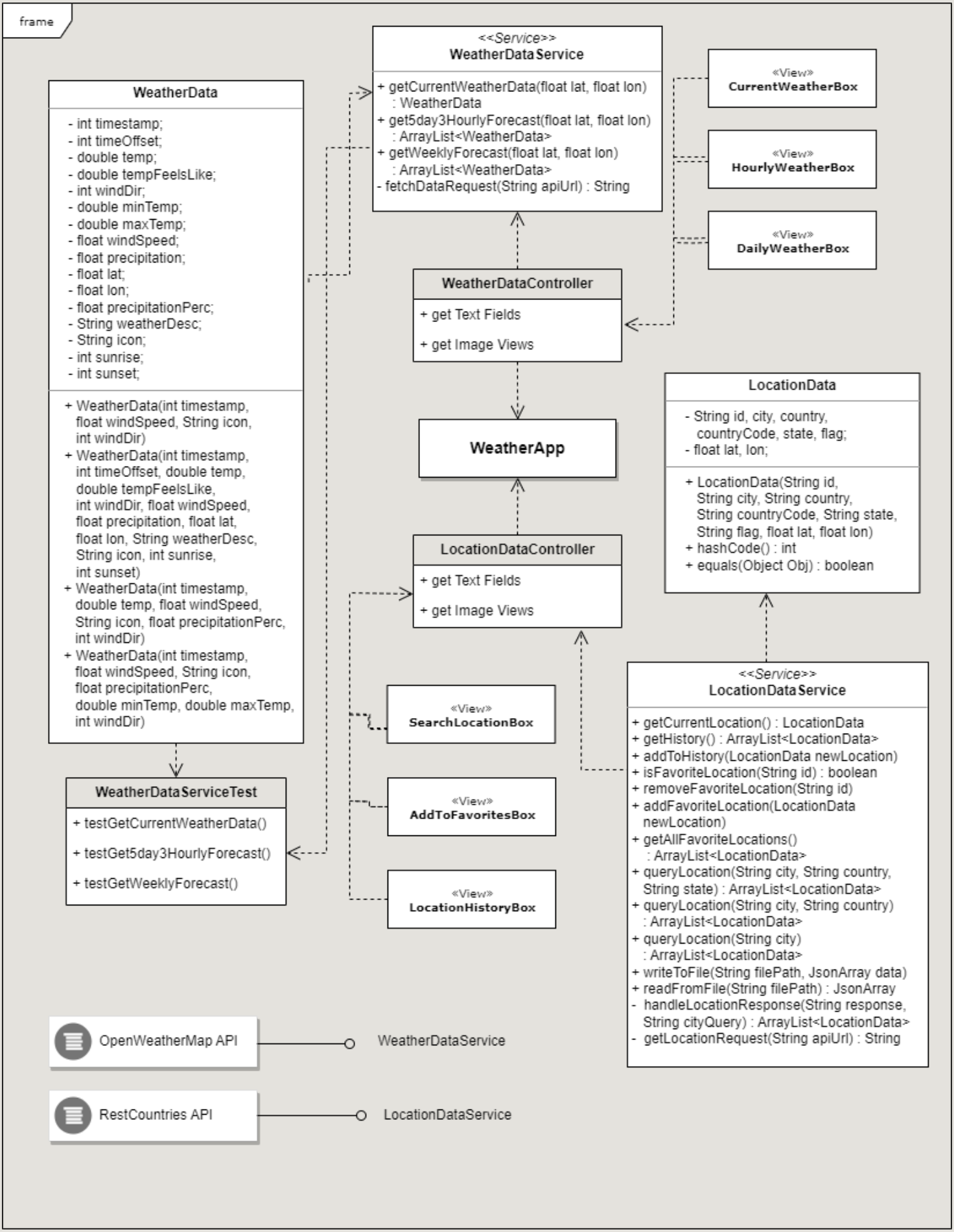
In the Weather App, the Models are responsible for fetching data, handling data storage, retrieval, and manipulation to support the application's functionalities. Views represent the presentation layer of the application - the GUI. It displays information to the user and captures their interactions. And the Controllers act as an intermediary between the Model and the View, handling user inputs and responding accordingly by manipulating the Model's data or updating the View.

Objectives:

- **Enable location searches and display current weather**
- **Implement at least one of the provided interfaces**
- **Develop a unique graphical user interface.**
- **Allow saving locations as favorites and restore program state on restart**
- **Present a detailed forecast (hourly and daily aggregate).**
- **Use custom weather icons, more than what OpenWeatherMap provides.**
- **Handle errors during file processing.**
- **Implement unit tests for the program.**
- **Support for Multiple Systems of Units.**
- **Create an additional feature not listed in the requirements but requiring coding effort.**

Action Plan:
MainView presents current weather and search button
iReadAndWriteToFile implemented in LocationDataService
Custom JavaFX User Interface rendered
Search history and program state maintained
Visualization of 3 Hourly and Daily weather forecasts
Weather icons for multiple weather data fields
Error Handling when reading from / writing to files
JUnit Tests implemented for Models and Services
Conversion to Imperial and Metric Temp Unit Systems
Rendering UI w.r.t Sunset/Sunrise.

UML Class Diagram



Key Classes & Responsibilities

The **WeatherApp** class is the main entry point of the Weather App JavaFX application. It extends **javafx.application.Application**, providing the **start** method to initialize and display the graphical user interface (GUI) for the Weather App.

- **Application Initialization**
- **Instantiates Data Services**
- **GUI Configuration**
- **User Interaction Handling**
- **Debugging/Test Interface**

The **WeatherDataController** class serves as a controller responsible for handling the presentation logic related to weather data in the Weather App's graphical user interface (GUI). It contains methods that format and provide textual and visual representations of weather information retrieved from the **WeatherData** model.

- **Data Formatting**
- **Textual Representations**
- **Visual Representations**
- **Time and Date Display**
  Utilizes utilities from **WeatherUtils**

The **WeatherDataService** class is responsible for fetching weather data from the **OpenWeatherMap** API for a specified location. It retrieves various weather information temperature ranges, wind speed, precipitation percentage, weather icons etc.

- **API Integration**
- **Weather Data Retrieval:** Fetches current, 3-hourly 7-day daily weather forecast
- **JSON Parsing**
- **API Request Handling**

The **LocationDataController** class manages the display and formatting of location-related information within the Weather App. It specifically deals with handling and presenting data related to a searched location.

- **City Name Display**
- **Country Flag Display**
- **Country and State Name Display**

The **LocationDataService** class primarily handles location-related functionalities within the Weather App. This service manages fetching, processing, and storing location data for the application.

- **Location Response Handling**
- **File Handling** (**Reading/Writing JSON**)
- **Favorite Locations Management**
- **History Management**
- **API Request Handling**

The **JUnit** test class **WeatherDataServiceTest** focuses on testing the functionalities within the **WeatherDataService** class. The tests aim to confirm that the WeatherDataService methods function correctly.

- **Checks retrieval of weather data**
- **Ensure the attributes fall within expected ranges or conditions**
- **Validates the functionality of the service class**

The **WeatherUtils** and **LocationUtils** classes are a utility class offering various methods for weather, time and location related operations.

- **Conversion to the specified unit system ("metric" or "imperial").**
- **Conversion of UTC timestamp into local.**

The **MainContent** class inclusive of other classes in **.\views** are a part of the view layer in the weather app. It constructs the content layout of the application, incorporating various data related panels, icons and buttons. They handle the organization and presentation of the data within the app.

Project Functionalities

1. UI Components Setup

- **Stage & Scene Configuration**: Sets up the initial stage with a scrollable tabbed interface.
- **Tabs**: Configures two tabs - ***'Weather Report'*** and ***'Search and History'.***
- **Components**: Initializes various UI elements like text fields, buttons, and dropdowns within the tabs.

2. **UI Views**

- **MainContent Class**: Constructs the main content view for weather data display. This class serves as a container to organize and display weather-related information within a graphical user interface.
- **SearchContent Class**: Handles the search interface and history display. Constructs a layout where the search box and search results are in one panel, and favorites and history are in another panel. Each location item within the history and favorites sections has a click event handler that triggers an action when clicked, to perform a search or display detailed information related to that location.

3. Main Content View

- **Unit System Conversion:** Allows toggling between metric and imperial units for temperature-related fields (like temperature, feels-like temperature, min and max temperatures) in a WeatherData object to a specified unit system.
- **Weather Data Retrieval:** Encapsulate different parameters of Weather Data such as timestamp and time offset, temperature data, wind information, precipitation details, geolocation data, ***icon representation***, sunrise and sunset times tailored for current, 3 hourly and daily weather data based on the location.
- **Time Zone Handling:** Displaying weather information in the local time zone of the selected location ensuring relevance and ease of understanding.

4. Search Interface

- **Search and History Management:** Encompasses the feature allowing users to explore and find locations within the application. This feature not only permits users to conduct searches but also manages a history of their previous searches. It retains a record of locations that users have explored or searched for, enabling them to quickly revisit these locations without having to re-enter the search details.
- **Add To Favorites:** Responsible for adding a user-selected location as a favorite within the application. When a user designates a location as a favorite, it handles the storage and retrieval of favorite locations using JSON files.
- **UI Interaction:** Involves the application's responsiveness to user actions and inputs. It listens and reacts to various events triggered by the user, such as clicking buttons or selecting options from dropdown menus. For instance, when a user clicks a button to initiate a search or selects a particular location from a dropdown list, the user interface responds by updating the displayed content accordingly.

5. Application User Interaction

- **Events Handling:** Manages user interaction events like clicking buttons or selecting dropdown options to trigger data updates and view changes.

6. **UI Styling**

- **UI Styling and Layout**: Defines UI elements' appearance, layout, and styles for a more user-friendly interface.
- **Weather Color Codes**: Background color appearance display according to the temperature of searched location.
- **Chart-like Temperature Display:** Generate chart-like hourly and daily temperature display for better visualization.

Division Of Work

Ayman Asad Khan	Xuan An Cao	Areeba Nadeem
Project Documentation	Internal Project Documentation	Location Search User Interface (Frontend)
README.md / User Manual	Retrieval of Location Data (Backend)	
Retrieval of Weather Data (Backend)	Processing and Presentation of Location Data	
Processing and Presentation Of Weather Data	Java FX implementation of User Interface Views	
Additional Functionalities	Additional Functionalities	

USER MANUAL

Comprehensive guidelines for project configuration and getting started provided in README.md