Weather Application Documentation

Tampere University

COMP.CS.140 - Programming 3

Author: Nguyen Quang Duc, Nguyen Viet Duy Quang

Date: 06/12/2023

Table of content

1. Installation instruction

The Weather App's JAR file is located in the 'Release' folder of the group's root directory. You can run it by using the command line, which is *'java -jar "weather_app_release.jar"'* at the 'Release' folder's directory.

2. User manual

Our Weather App is the one you can use to check your weather right now, weather forecast and past weather as well.

Different from other applications in the market, we have one of the most user-intuitive graphical user interfaces.

You can quickly search for your own place, add it to your favorite list and see how the weather there looks at any time.

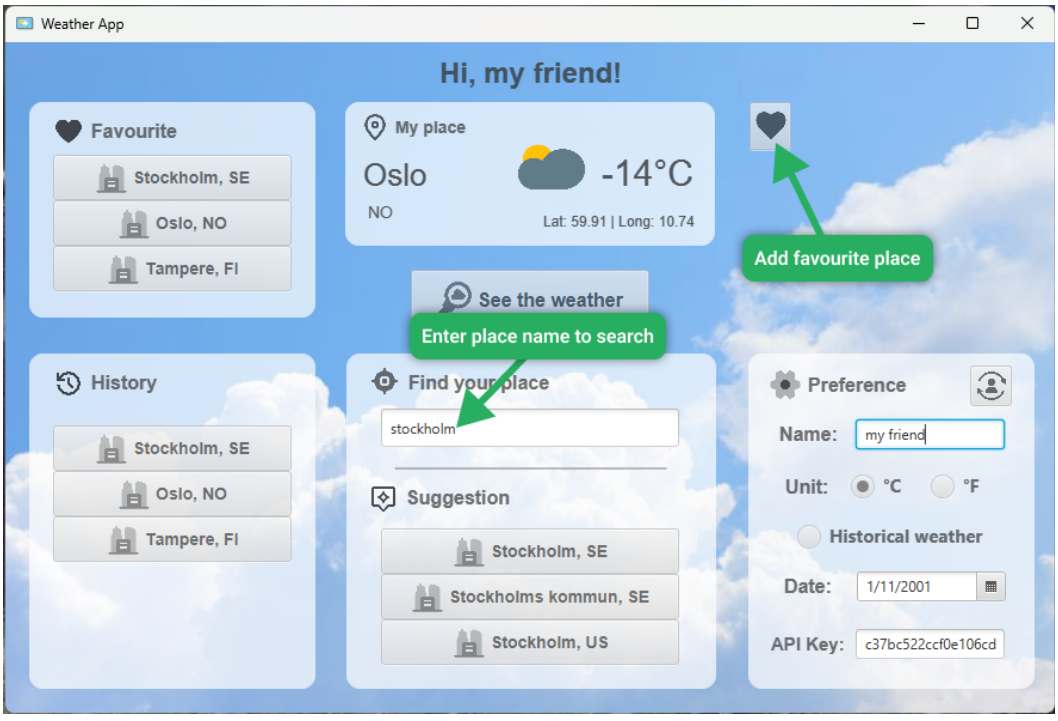

Figure 1: Guide on how to search for places and add favorite places.

You also can see the weather in the past by selecting the 'Historical weather' button and preferred date in the 'Preference' section.

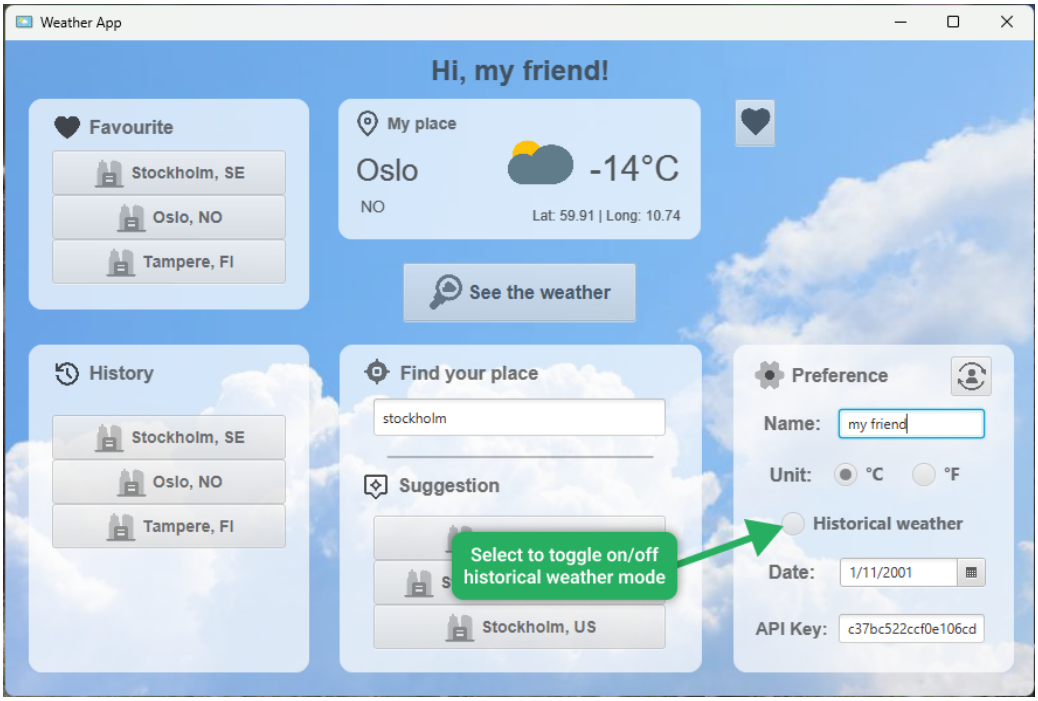

Figure 2: Guide on how to toggle on and off historical weather mode.

After that, all you need is to press the ‘See the weather’ button to find out how the weather looks there.

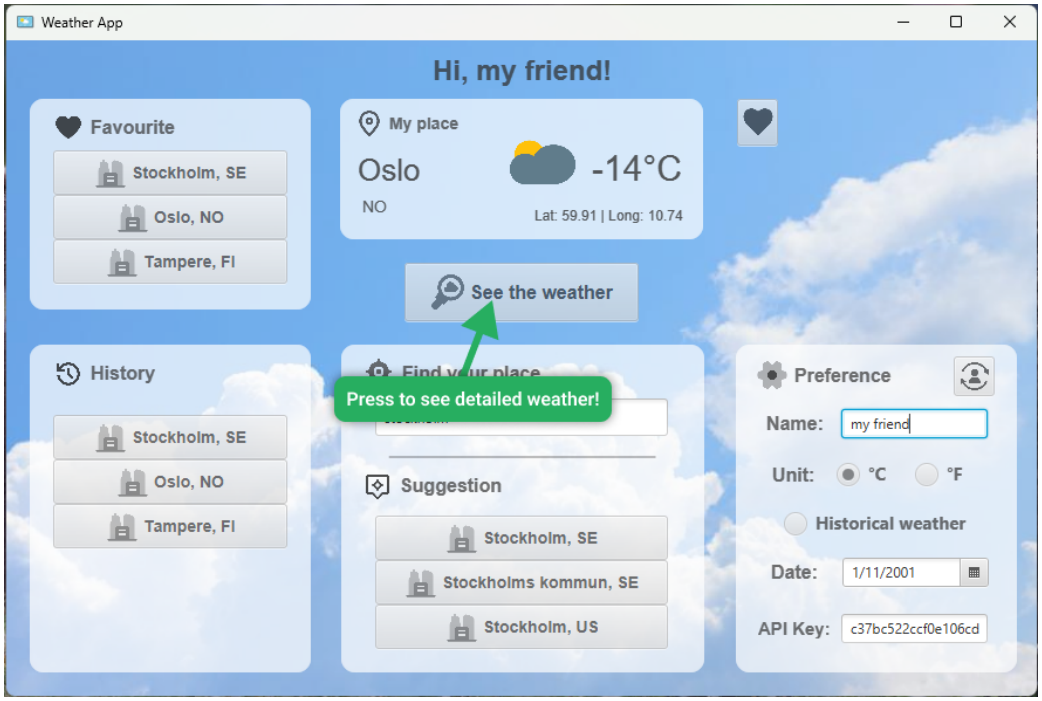

Figure 3: Guide on how to see the detailed weather information.

If you like, you can go back and play around without forgetting the old places saved in search history on the left side.

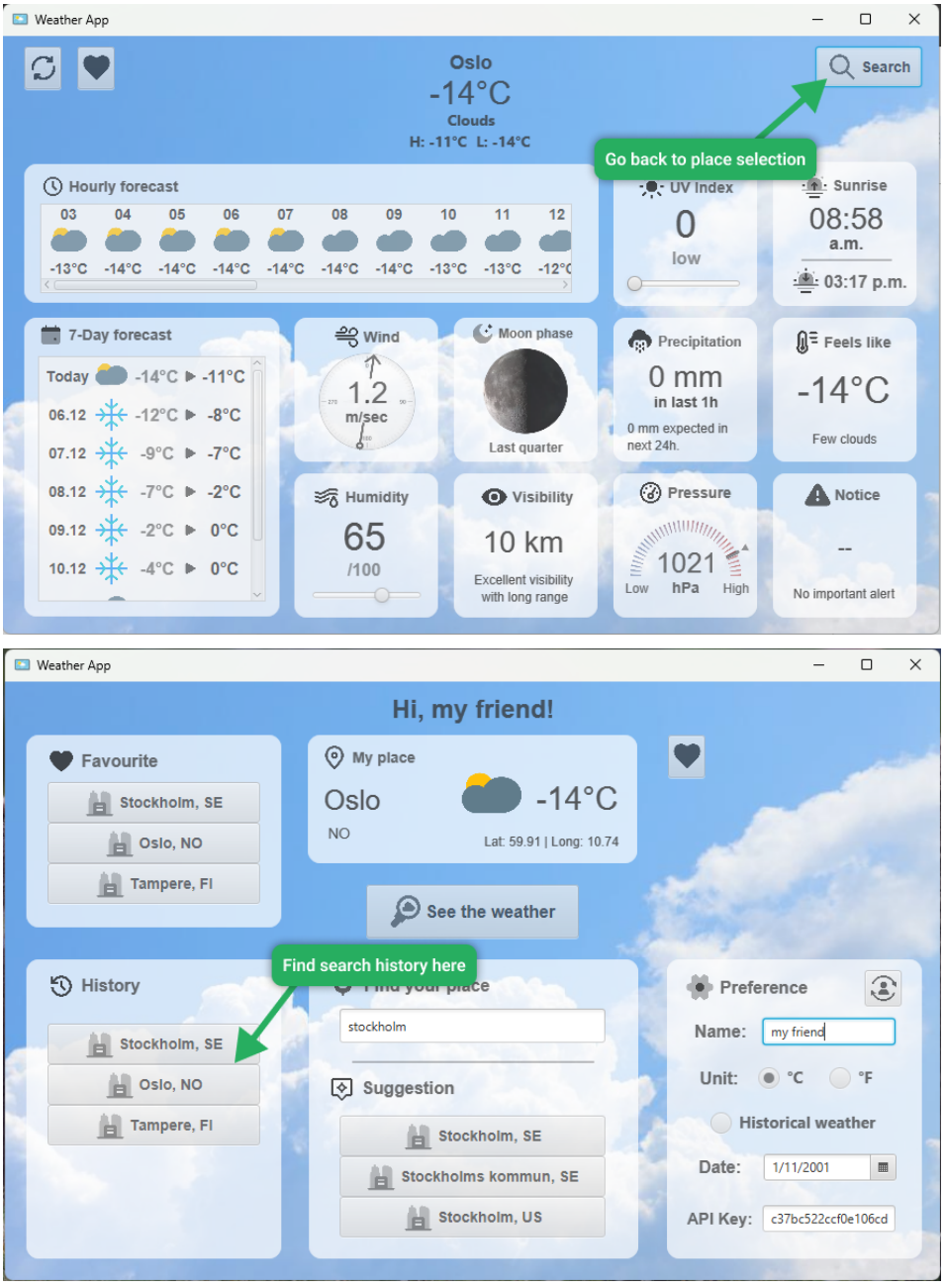

Figure 4, 5: Guide on how to navigate back to place selection and use history search.

Want to feel more familiar? You can change the displayed unit and greeting name in the ‘Preference’ section. For the nerds out there, the API key should be only changed with caution, and there is always a reset button to revert all the changes.

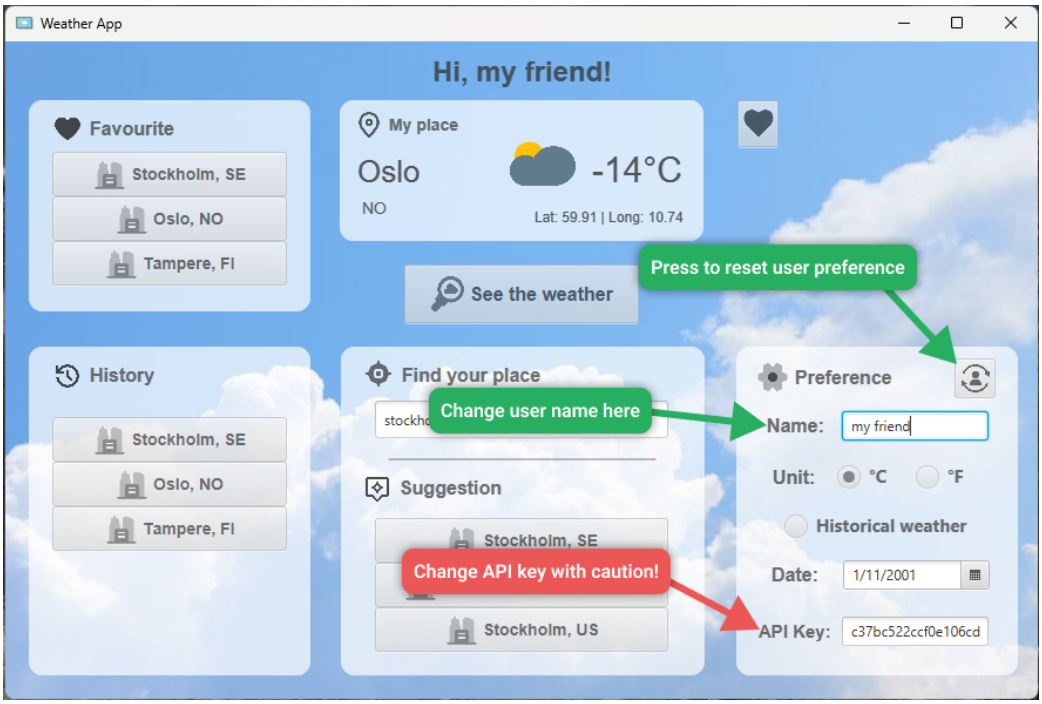

Figure 6: Guide on how to customize user preference.

3. UML map of the software

The UML map that represents the structure of the software implemented by our team is demonstrated as follows.

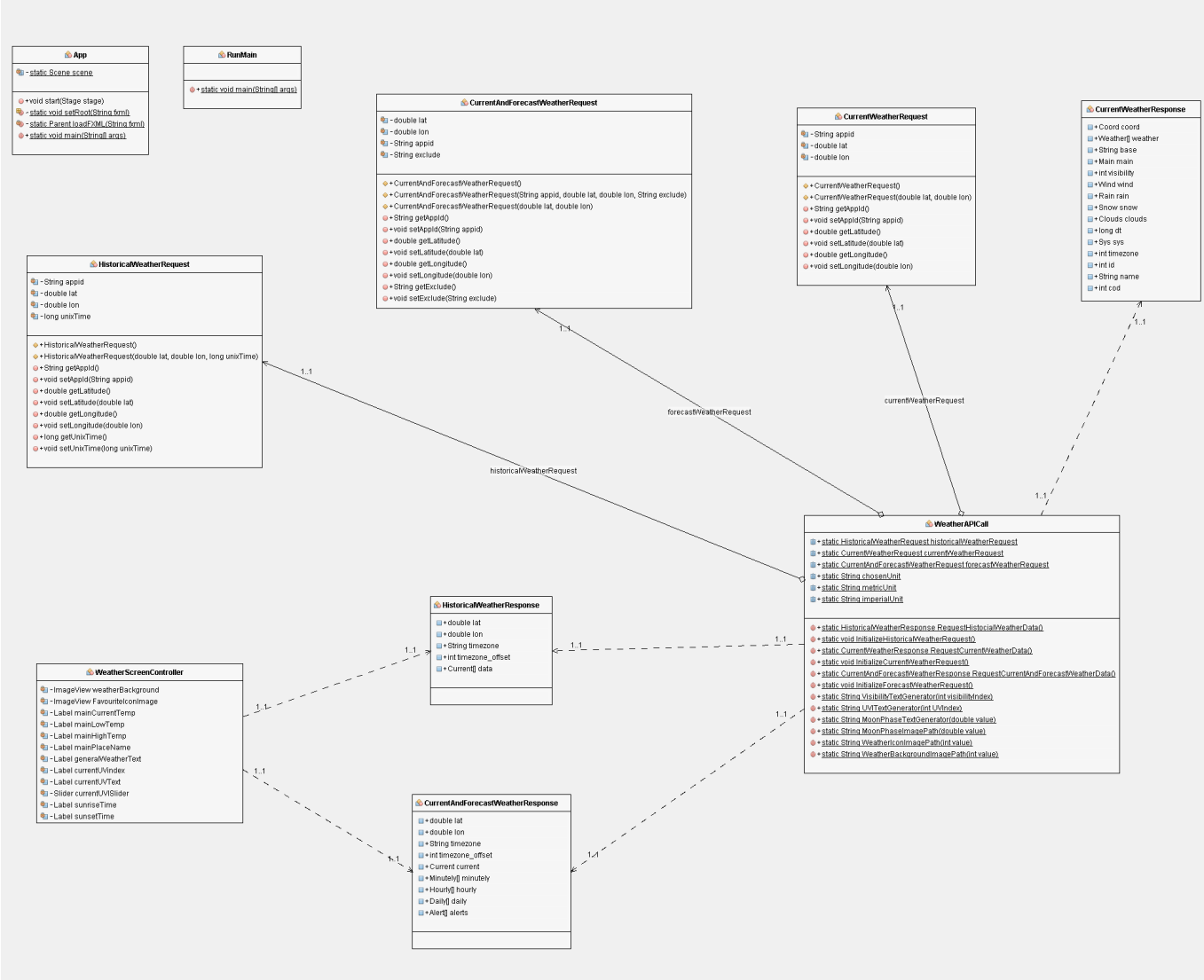

Figure 7: First part of the UML map of classes in the software

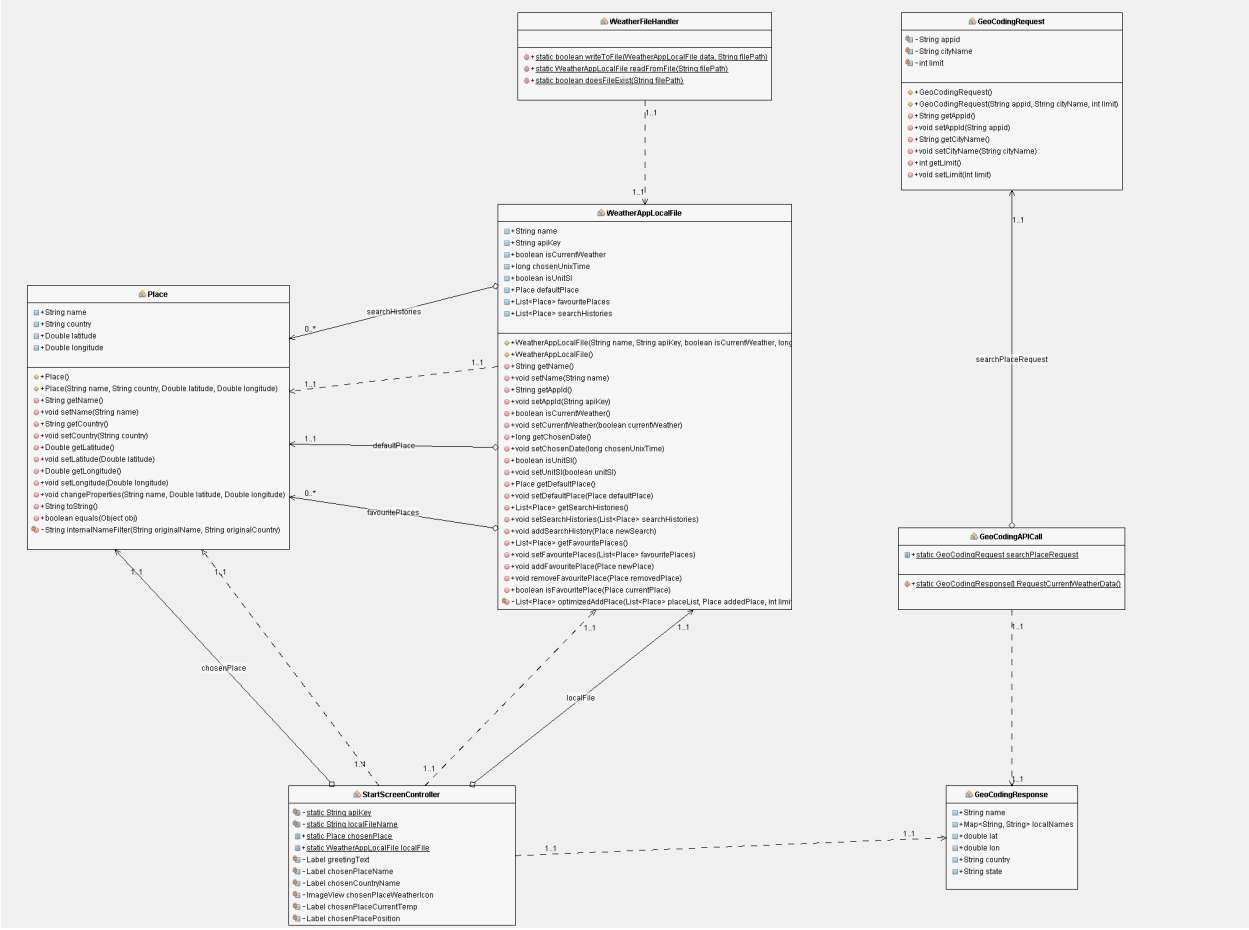

Figure 8: Second part of the UML map of classes in the software

4. JavaDoc documentation

Our team's project also has the Javadoc documentation from the source code in the directories: *"Documentation/apidocs/index.html"* or *"WeatherApp/target/site/apidocs/index.html"*. This document can be used with browsers to learn more details about the classes and its properties.

5. Important Java classes' responsibilities

The software includes different types of classes that play different roles. The table of all important classes with their description can be found as follows.

Table 1: Class name and description

Ordering number	Class name	Description
1	App	This class initializes the stage window as well as its properties (name and icon of stage window) and loads the FXML files for the corresponding scenes.
2	RunMain	This class serves as an entry point for the application, which initializes the execution of the JavaFX application.
3	StartScreenController	This class plays as a communicator between the FXML file named "StartScreen.fxml" and the underlying logic of GUI elements of the file.
4	WeatherScreenController	Similarly, the class acts as the intermediary between the UI components defined in the FXML file "WeatherScreen.fxml" and the application logic.
5	WeatherAPICall	This class contains all static functions to perform API call that request current weather data, forecast weather data and historical weather data.
6	CurrentWeatherRequest	This class contains properties needed for an API call requesting current weather data.
7	CurrentWeatherResponse	This class has all properties of a current

		weather data response to be parsed from the API call's response.
8	CurrentAndForecastWeatherRequest	This class contains properties needed for an API call requesting current and forecast weather data.
9	CurrentAndForecastWeatherResponse	This class has all properties of a current and forecast weather data response to be parsed from the API call's response.
10	HistoricalWeatherRequest	This class contains properties needed for an API call requesting historical weather data.
11	HistoricalWeatherResponse	This class has all properties of a historical weather data response to be parsed from the API call's response.
12	GeoCodingAPICall	This class contains all static functions to perform API calls that request a list of possible place objects based on an input string.
13	GeoCodingRequest	This class contains properties needed for an API call requesting a list of possible place objects.
14	GeoCodingResponse	This class has all properties of a possible place to be parsed from the API call's response.
15	Place	This class represents a place with necessary properties to be handled by other classes.
16	WeatherAppLocalFile	This class contains properties related to user preference setting and other utilities to be handled by other classes and saved as a JSON file in the user's local device.
17	WeatherFileHandler	This class encompasses static functions that write and read a specific object of "WeatherAppLocalFile" class in the user's device as well as check if such file exists.

Furthermore, preconditions and postconditions of each method inside the classes are well written in the Javadoc-standard comment section of each method.

6. Software functionality

Overall, our software includes two main scenes. The first scene is made for users to search for a specific place, choose a default place and save them into their favorite places. The second scene mainly displays weather-related data such as current weather, weather forecast, etc. The overviews of the first scenes can be found as follows.

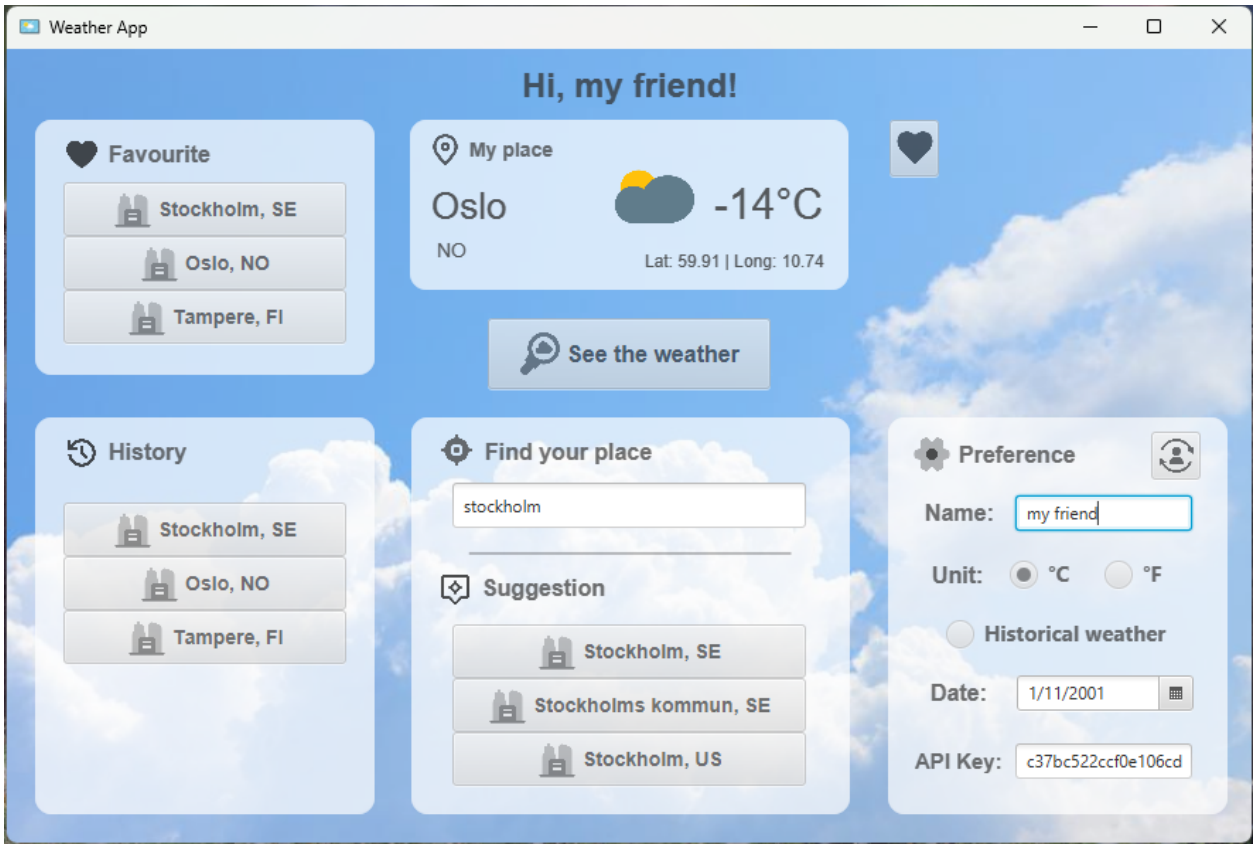

Figure 9: An overview of first scene

In the first scene, features we performed and their description is listed in the following table.

Table 2: Features and their description in first scene

Ordering number	Feature name	Feature description
1	Search for a new place	Users can enter a place name in the ‘Find your place’ section to search for the place.
2	Automatic place search	The place search is performed every time a user enters a new key in the textfield.
3	Automatic search history saving	Everytime user selects a new place, it will be added into the search history place list.
4	Add and remove favorite place	When selecting a favorite icon, the current chosen place will be added into the favorite place list, if such place is already in the list, it will be removed.
5	Quick selected history and favorite place saving	Users can quickly select a saved place in history and favorite place without re-enter place name in the textfield.
6	Automatic chosen place‘s information update	Currently chosen place’s name, location and current weather are updated in the ‘My place’ section.
7	Change user preference	Users can change their name, displayed unit (metric or imperial unit), general API key used to make requests.
8	Switch between current weather and historical weather mode	Users can select historical weather mode and select the historical past date in the preference section.
9	Automatic user preference saving	When users change any value in the preference section, the change will take effect immediately and the change will be saved into a JSON file in the device.
10	User preference reset	Users can reset user preference back to default one by pressing reset button.
11	Notifying request error	If the request is rejected due to invalid API key, an UI element will appear to notify users.
12	Apply saved preference on startup	When the application opens for the first time, the saved JSON file in the device is

		read and the changes in the file will be applied to the application.
13	Custom weather background and weather icon	The weather background and icon are highly customized depending on weather status.
14	See more detailed weather in chosen weather mode	When pressing the ‘See the weather’ button, users are moved to the second scene with chosen weather mode.
15	Automatic history place search and favorite places saving	A list of history place searches and favorite places are saved to the local device every time their values change.

Furthermore, the weather display of weather data in the second scene can be different depending on the fact that the user selects current weather mode or historical weather mode. Overview of the second scene in two different modes can be found in the following figures.

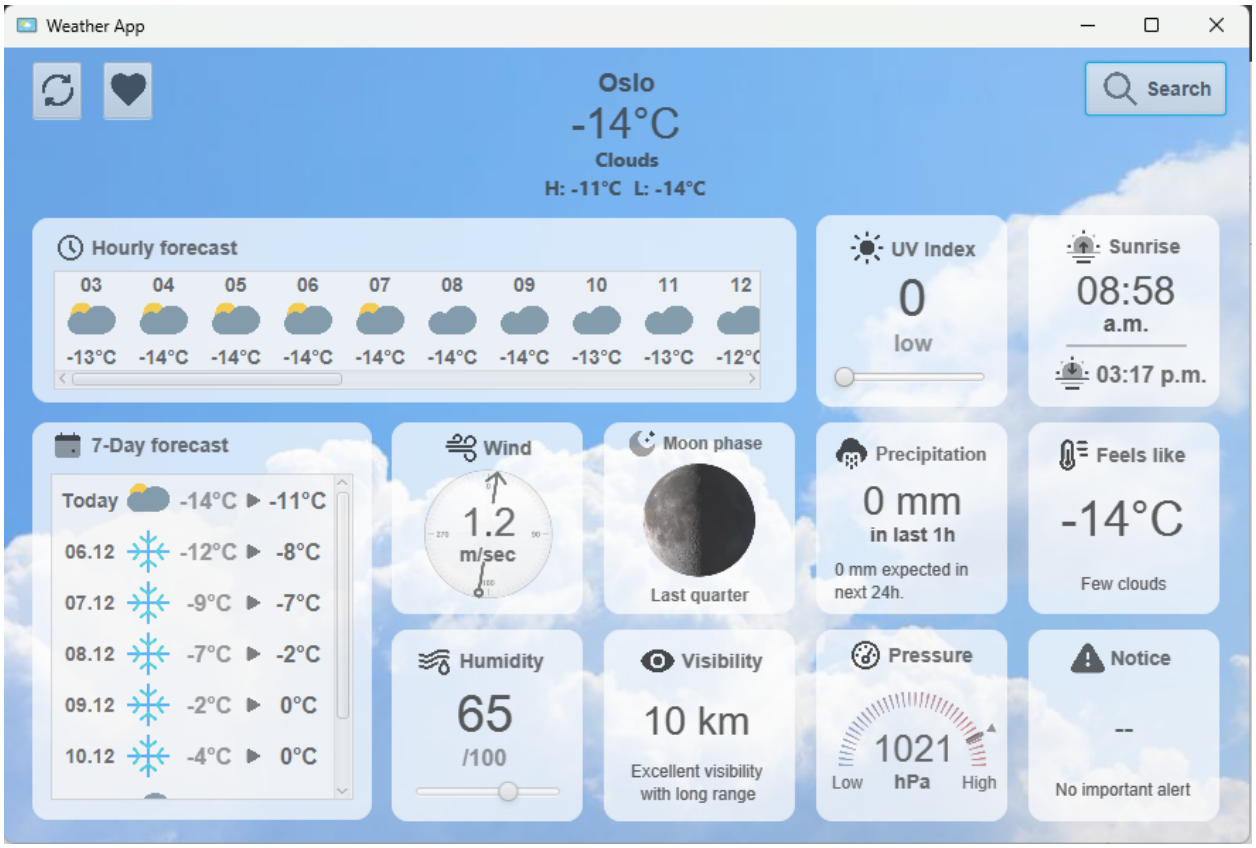

Figure 10: An overview of second scene in current weather mode

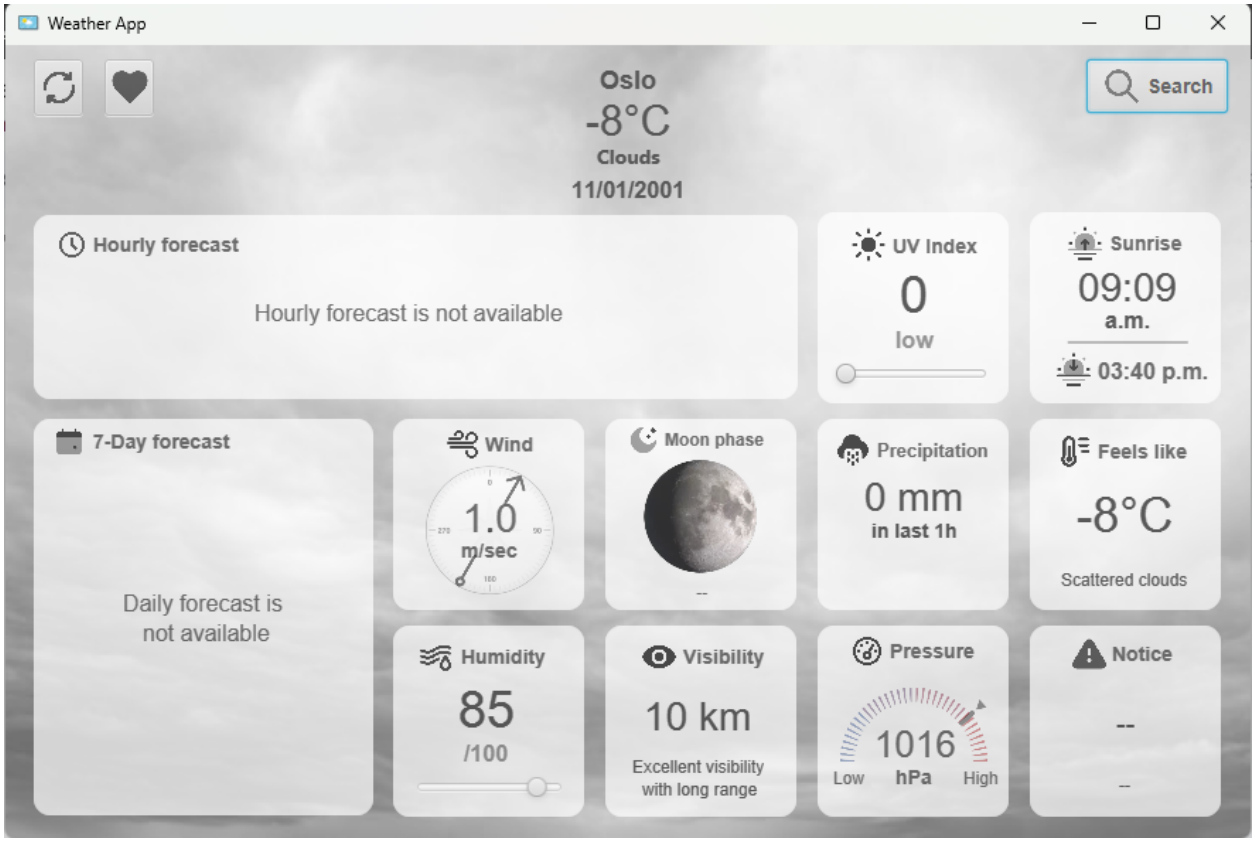

Figure 11: An overview of second scene in historical weather mode

The weather-related data displayed in the second scene during the current weather and historical weather modes are listed in the table below.

Table 3: Weather data displayed in the current weather and historical weather modes of second scene

Ordering number	Weather data	Current weather mode	Historical weather mode
1	(Historically) Current temperature	Yes	Yes
2	Chosen place name	Yes	Yes
3	Highest temperature during the day	Yes	No
4	Lowest temperature during the day	Yes	No
5	Short weather description	Yes	Yes
6	Next 24-hour forecast of temperature and weather status.	Yes	No
7	Next 7-day forecast of highest and lowest temperature as well as weather status.	Yes	No
8	Current wind speed and wind degree.	Yes	Yes
9	Current moon phase name.	Yes	No
10	Precipitation in last 1 hour	Yes	Yes
11	Precipitation in next 24 hours	Yes	No
12	Feel like temperature	Yes	Yes
13	Humidity percentage	Yes	Yes
14	Visibility range	Yes	Yes
15	Ground-level pressure in hPA	Yes	Yes
16	Important weather alert	Yes	No
17	Sunrise and sunset time	Yes	Yes
18	UV Index	Yes	Yes

Apart from details related to weather, the second scene also supports some fundamental features listed as follows.

Table 4: Features and their description in second scene

Ordering number	Feature name	Feature description
1	Refresh weather data	Users can press refresh button to refresh weather data from the API call
2	Notifying request error	If the request is rejected due to invalid API key, an UI element will appear to notify users.
3	Custom weather background and weather icon	The weather background and icon are highly customized depending on weather status.
4	Add and remove favorite place	When selecting a favorite icon, the current chosen place will be added into the favorite place list, if such place is already in the list, it will be removed.
5	Return back to first scene	When pressing the ‘Search’ button, users are moved to the first scene with saved preference.

7. Software testing

To ensure the software performs without encountering any issue, our team has implemented two different types of testing, which includes Unit testing and TestFX testing. The tables below will show different functions utilized in Unit testing and TestFX testing.

Table 5: Unit testing functions and description

Function name	Description
testDefaultConstructor	This function is used to test default initialization of various fields in the class WeatherAppLocalFile.
testConstructorWithParameters	This function is used to test the functionality of the parameterized constructor of the class WeatherAppLocalFile.
testGettersAndSetters	This function tests the functionality of get and set methods of the class WeatherAppLocalFile.
testAddFavouritePlace	This function tests the validity of a method named testAddFavouritePlace of the class WeatherAppLocalFile.

Table 6: TestFX testing functions and description

Function name	Description
clickOnUnknowButton	This function tests the scenario where an unknown button is clicked.
testButtonAddFavouritePlace	This function tests the functionality of adding a favorite place.
testDefaultPlaceDisplay	This function tests the display of the default place.
testAutomaticSavedUserPreference	This function tests the automatic saving of user preferences.
testNewPlaceSearch	This function tests the search functionality for a new place.

8. Work division in team

Our team size is quite small, which includes only two members. The information of the members is put in the following table.

Table 7: Team members' information

Name	Email	Short name
Nguyen Quang Duc	duc.3.nguyen@tuni.fi	Duc
Nguyen Viet Duy Quang	quang.nguyen@tuni.fi	Quang

During the implementation of the software, we decided to divide the tasks based on each member's strength as well as personal preference. For example, Duc has work preference as developing programs, while Quang prefers to work with user interface designing. Based on that we have adopted and finished the tasks of developing software listed in the following table.

Table 8: Software development tasks of each member in the team

Member name	Nguyen Quang Duc	Quang
Task lists	● Sketch and organize software structure. ● Read and design structure handling API calls. ● Implement important functions. ● Perform Unit testing and TestFX Testing.	● Sketch and design user interface. ● Implement FXML file for different scenes. ● Gather required graphic resources. ● Design implementation for Unit and TestFX Testing.

Moreover, to facilitate the work of developing software, we composed a workflow that includes a list of tasks needed to be done step by step to make a fully functional software. The workflow is represented as the following table.

Table 9: Workflow table with ordering tasks corresponding responsible members

Ordering step number	Task	Responsible member(s)
1	Gathering software requirements	Quang, Duc
2	Sketch a basic user interface	Quang
3	Fix and agree upon the user interface	Quang, Duc
4	Implement FXML file in Scene builder based on the design	Quang
5	Developing the API-handling functions	Duc
6	Perform unit testing for the functions	Quang
7	Developing FXML-handling functions	Duc
8	Perform TestFX testing for the functions	Quang
9	Organize code and enhance coding style	Duc
10	Writing documentation for the software	Quang, Duc

9. Known issues

Since our team's application heavily relies on the API Call to perform, there are a variety of issues related to API calls and responses from those calls accordingly. To bring that into awareness, every API call made by the application is handled with the try and catch structure to spot network-related issues. Some common issues can be found in the below table.

Table 10: Common issue arising in the application

Issue name	Issue description	Possible issue
Unauthorized request issue (Code 401)	This happens when a user changes the default API key with invalid one.	Reset back to default preference or use another valid API key.
Network issue	This happens when users have unstable network connections or the API call is not handled properly on the OpenWeatherMap server's side.	Establish a stable internet connection or try next time later when the OpenWeatherMap server is stable.
Library missing issue	We witnessed this issue happen when the application JAR file is run with a different java version than 20.0.2.	Run the application JAR file with the desired java version, which is 20.0.2.

Because the user can modify the API key, it makes the unauthorized request issue happen more frequently. To tackle this issue, we have developed a notification appearing when the unauthorized request issues are catched by the function. This notification will let users be aware of the error and change the current erroneous API key.

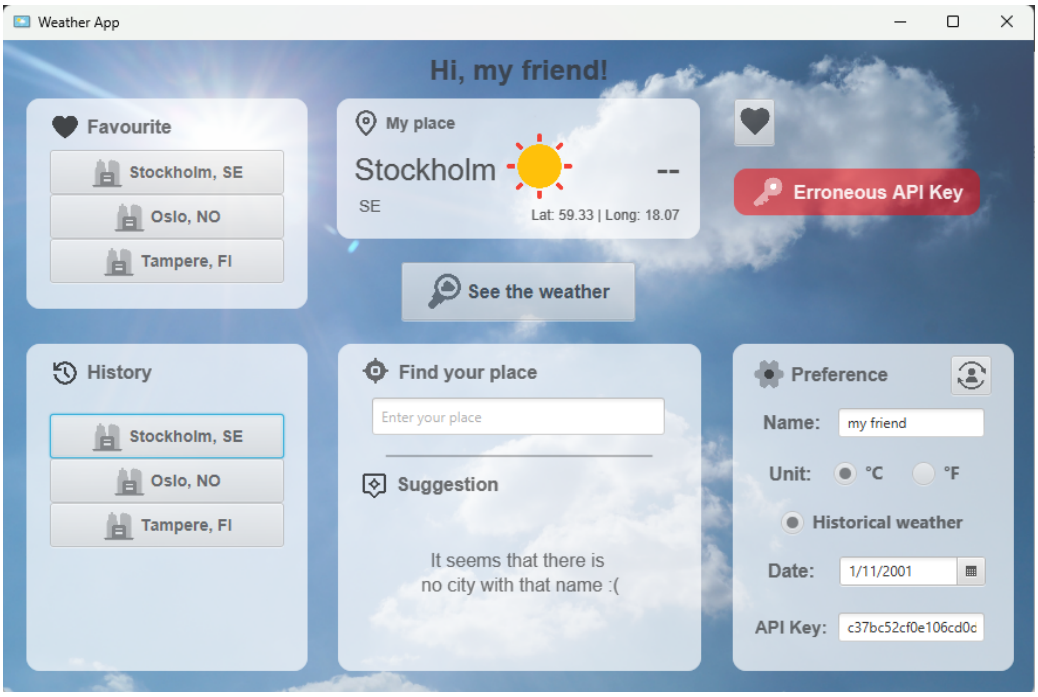

Figure 12: Notification appears when users use an erroneous API key in the first scene.

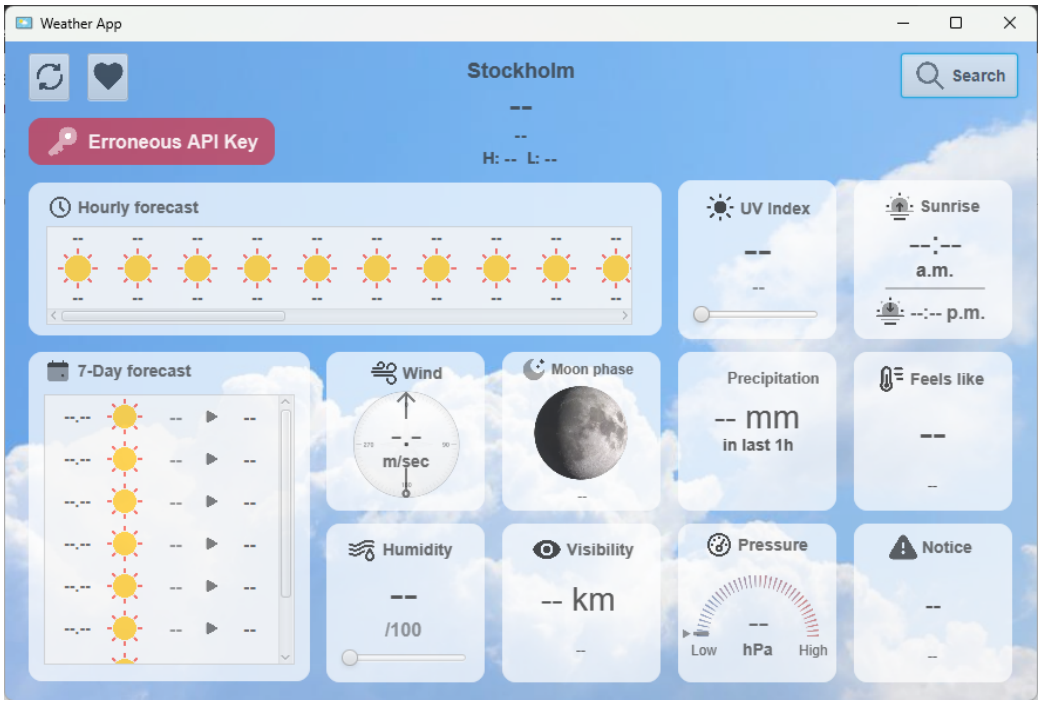

Figure 13: Notification appears when users use an erroneous API key in the second scene.