Documentation for WeatherApp

UML Diagram:

The UML diagram is present in the documentation folder as weatherapp_uml.svg.
Link to file.

Responsibilities of key classes:

- WeatherApp : Main class which extends the Application class from the javafx library. Creates the gui and adds components to it, while also having the ability to change the gui based on events.
- WeatherApi: Implements iApi and is responsible for making api calls to the OpenWeather api, and returns the json data as string.
- WeatherJsonProcessor: Reads and writes json data to the file to store user input
- LocationData: Gson class, but is used as the model to store search history and favourites.
- CurrentWeatherDisplay: Displays the current weather of the searched location
- DailyForecast: Stores the daily forecast for a location, using DailyForecastColumn. (5 days)
- HourlyForecastDisplay: Stores the hourly forecast for a location, using HourlyForecastColumn. (24 hours)
- ForecastChart: Present in its own scene, displays the forecast charts for temperature, rain and wind speed for the location.
- SearchBar: Present in its own scene, allows a user to search for locations and displays the favourites using LocationRow.
- SearchHistory: Present in its own scene, displays the search history of the user using LocationRow.

Project Functionality:

This project aims to meet the top-grade requirements. All the required functionality in the minimum and intermediate requirements have been implemented as well as five additional requirements.

The five additional requirements are:

- Usage of charts - ForecastChart displays the charts
- Location search history - SearchHistory displays the search history
- Multiple units of measurement - Units(class name) changes the units and the api makes the call with it.
- Continuous integration for unit tests - Test task present in the gitlab-ci.yml file
- Remove locations from favourites and search history. (Own feature) - SearchHistory, Favourite and SearchBar.

Division of Work:

Onur Sertgil: The design and implementation of the UI element which displays the current weather condition was done by him. For this task, the required classes and functions were coded to generate and update the UI element with current data. In addition, the research for a more comprehensive weather icon set was done by him. This task includes the process of choosing and labelling each icon with respect to the description of OpenWeatherMap.

Vasav Juyal: Implementing the api and creating the gson classes to handle json data, making the search functionality and displaying the searched location, feature to add and delete favourites and display them, feature to display search history and to delete it, creation and updating of the hourly forecast, creation and updating of the daily forecast, feature to change the unit of measurement, tests for the self-made non gui classes and adding the test task for CI/CD.

Ville Kivioja: Implementing Json reader and writer for saving search history, favourite locations and current location and tests for it, making exploratory version of saving searches and favourites that was later left out due to better implementation being found, Implementing tab for charts to show different measurements of the weather and creating class diagram for the program.

User Manual:

On the first opening of the program, the user is automatically directed to the weather display of Tampere, showing current weather, daily forecast and hourly forecast. Searches can be made by clicking the text field on the top right corner of the gui, which will display a new scene which contains a search bar and displays favourites. Searching a location will automatically transfer back to the main scene. Favourites can be added by clicking the star icon and can be removed from the search and favourites tab. You can press on any of the favourites to re-search for them

Search history is present on the top left, which displays a new scene where the locations are displayed based on recency. Similarly to the favourites, these locations can be deleted and can be pressed to re-search for them. Next to it is the unit changing button. It switches to the unit which is displayed on the button, i.e, if it says “Imperial”, clicking it will switch units to imperial.

The forecast charts can be viewed by pressing the button at centre left, where a new scene is displayed again with three different charts.

All the scenes have a back button, which can be used to go back to the main scene. In case of an empty file or an error in reading the file, the user will be directed back to just Tampere as the location and behave as if it's been opened for the first time.

Known bugs and missing features:

Although locations can be searched, some places do not work and states, countries can not be searched as a whole. It has to be a single location, like a city or a town.

Some locations do not have a state, so there are some locations in the search history which do not show any states, just their names.

If the data obtained from the api has a description of the weather which is not similar to the predefined options, an image of an error is used to display the data.

While getting the measured precipitation for a given time would enhance the clarity on top of probability of precipitation, this was dismissed due to API giving null values for the said statistic on multiple occasions.

AI:

AI use has not been marked in the source code, but all uses have been mentioned here.

- In the SearchBar, SearchHistory and Units classes, there is a use of a Task and Thread to make asynchronous execution of the program. Along with this, the use of Platform in SearchHistory and WeatherApp to update the ui in the javafx thread.
- Many uses of AI for styling, positioning and alignment, like asking for colour hex code, alignment advice, border width, rounded borders and fonts. Also some help with padding and margins.
- Animations used for loading: creating the arc and its rotation.
- Creating the star (Polygon) required to add/remove favourites
- Also used in test cases for the gson classes. As they are quite simple, basic tests were generated and worked on further to save a lot of time.

Tools used: ChatGPT
Own thoughts: The current version of AI works well as a tool rather than a complete replacement of work. It gets stuck a lot when you try to describe a situation that it does not have a lot of material to draw from. When it comes to common, well documented solutions to material that you are unfamiliar with, it is quite useful. It saved quite a lot of time by generating tests for the gson classes which were quite basic.

Although it could be used to generate an entire solution, though as the codebase increases in size, it gets worse. At no point did I describe the task at hand, just general questions like, “How do I make the borders rounded?” or “How do I make an asynchronous call to an api and display something at the same time?”. Generally, I feel like these are the best way to ask questions, as a good way to save time while also learning at the same time, and not just have it do everything for you.