Task 2

List of Requirements

Functional

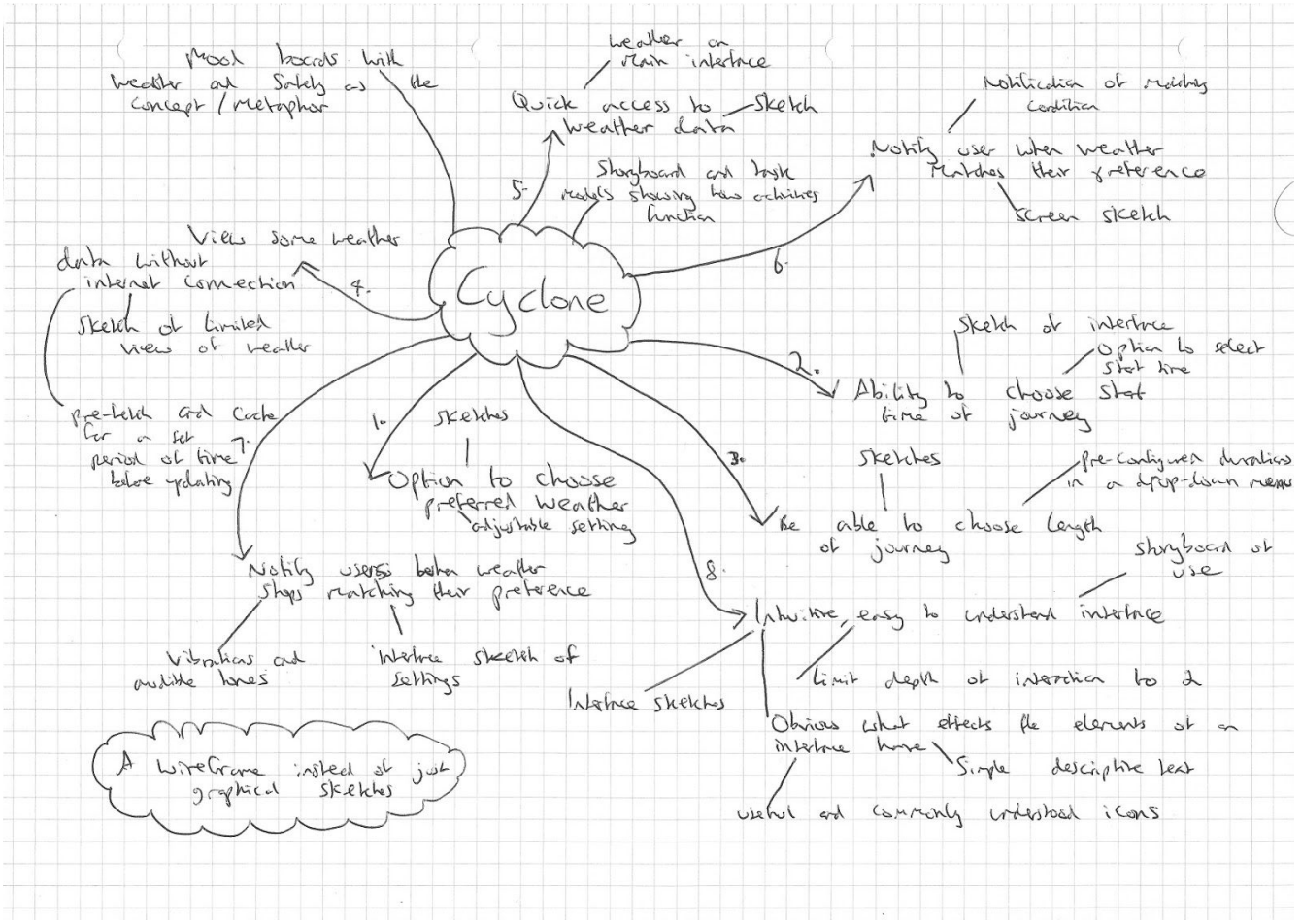
1. Ability to schedule cycling sessions with specific weather
2. Be able to enter preferences with prefered times
3. Be able to deal with varying lengths of times of cycling
4. Usable without internet connection
5. Reasonably quick to access data
6. Ability to notify users about appropriate times
7. Ability to notify users about changing weather while cycling
8. Intuitive and Powerful interface

Non Functional

9. Recent weather data
10. Usable for our target audience
11. Limited weather data cache capacity

Below our design process can be seen, from our initial brainstorm to the individual task models for certain activities. Also included are screen designs, wireframes and storyboards which complete the lo-fi prototype, each with their own descriptions and details relating them to our stakeholders' requirements. The brainstorm is centred around the requirements so that each design decision is relevant. From these requirements we note the concepts behind it that need to exist. There are also ideas for designs in terms of sketches/storyboards and along with the concepts are possible implementation details to help us build up our solutions.

Brainstorm



Moodboard for Weather:

Fog

Sun

Weather

Rain

Moodboard for Safety

Phone vs Tablet

From focus group discussions with our stakeholders, nobody once mentioned checking the weather on their tablet or using their tablet while cycling. Instead they immediately referred to their usage on mobile phones. Therefore the target platform of our design and application will be mobile phones. This is because it is in the best interest of our primary stakeholder which perhaps exists due to high levels of portability. In fact, requirement 4 talks about how it could be used when cycling, therefore in low signal areas. Requirement 7 also refers to usage while cycling, which would be much easier when using a phone - this is what the focus group members mentioned.

Wireframe - Main Screen

Wireframe - Daily View

Wireframe - Settings

Design - Main Screen

Design - Daily View

Design - Settings Screen

API Data Flow

Weather Iconography

These are the icons provided by the API as visual representations of the weather which are clear and intuitive to understand. As this conforms with requirement 8 we will use these icons as visual representations of the weather in locations defined in the designs.

Task 1: View Today's Weather

The process for displaying the current day's weather is as follows:

1) The default screen displays the current day's weather on a button structure
2) If tapped on, the weather is displayed more thoroughly on a new screen, with Highs, Lows, and hourly info.
3) If we click a particular time, we see wind speed and other useful information.

The idea is to proceed from low complexity to a higher level of complexity as the user requests it. This also agrees with requirement 8, being intuitive, and requirement 5, being quick.

Task 2: Show weekly weather forecast

Nowhere in the requirements do we directly describe the ideas with particular regards to accessing the weather forecast for the future period of time. However, it would be useful to consider the requirements which have any relationship to this task.

The first is requirement 8 - to have an intuitive and powerful interface. In order to make the system powerful, we want to find as much relevant information about the upcoming weather, with particular regards to the weather at particular hours for example. We specify in the further details of the requirement 1 that the useful weather for a cyclist is rain and wind, so it is important that we show the precipitation and the wind speed/direction for each period of time. The OpenWeatherAPI shows weather for three-hour periods for the next 5 days, so it appears ideal to choose these periods of time for the design. The other aspect of requirement 8 is having an intuitive interface. With regards to showing the weather, it means using easily recognisable 'stock images' for the different types of weather as well as easy readability of the important information, such as wind speed and direction, which can easily be represented with an arrow and a number for quick reading.

Given requirement 4. Which requires that the screen load quickly we want to ensure that the minimum data is on the page, as the more data, the longer it would take to load. However, it is clear that all the required data could be fetched quickly, so this is a very minor limitation.
Task Model:

In order to show the weather, we employ the previous main screen and add another screen for every day, hence having this **StoryBoard**:

Task 3: “Find today’s temperature in City X”
Task Model

StoryBoard

Task 4: Change time and duration settings
Task Model

StoryBoard

Lo-Fi to Hi-Fi Process
Pathway Diagram

Lo-fi to Hi-fi Design and Implementation
Using our designs shown above, we will implement hi-fi prototypes from our lo-fi prototype designs by implementing the screens shown in the form of sketches as interfaces. We will use the details on the designs which are backed up by the requirements of our stakeholder to ensure the implementation meets the demands.

The team will be largely split into two groups, with one group working on the data acquisition via the API (and from memory, when the internet connection is not working) as well as the settings interface and checking when cycling may be possible. The other will work on the main screen interface and the detailed day view interface shown in the design, given the assumption that the data would be passed from the other group's code to this teams. Therefore, it is important that these two teams also work together well to ensure there is a common method for the data, once found, to be given from the data acquisition code to the main interface. It is likely the first team will have two people, one of whom will work on the settings interface and the other of whom will work on the data acquisition, while the second team, with three people, will be split with two working on the main screen interface and the daily view interface respectively and one working on the population of the interfaces with correct data and ensuring that things like images can be accessed from the internet (or the device) in order to ensure the complete working of the design.

Afterwards, we will have these interfaces interact with each other by linking them together through buttons that allow movement between them. This will correspond with details outlined in storyboards and task analyses of how activities will be completed and how the user can move between screens. Here the members of the team will work together by sharing their individual work with the rest of the group to add the interaction implementation, as well as integrating the code that enables use of the API to display weather data.

Since the hi-fi prototype will be implemented in Java, we will use the likes of Swing and JavaFX to create our interactive interfaces in accordance to interface details detailed on the designs such as layout. Additionally, we will use a free weather API, OpenWeatherMap, to get the data we need to display.

There are potential risks involved, one of them being deviating away from the requirements when implementing the designs. To avoid this, it may be useful not to spend long periods of time on the implementation so as to not lose focus of the true goal, and to keep the requirements nearby so they are always our focus. Another risk is not being able to deliver a final implementation as it is not in working condition. One way to tackle this is to perhaps prioritise the requirements, getting the essential functionality out of the way, and then worrying about details such as having pixel perfect positioning of our interface elements. In such a case, the priority list is:

1. Requirements no.: (1, 2, 3)
2. Requirements no.: (4, 5, 8)
3. Requirements no.: (6, 7)

The requirements are grouped because it is difficult to prioritise them so precisely, but these groups 1 - 3 are enough to highlight which are most important to the hi-fi implementation. This is because, for example, requirement 6 is to do with mobile phone notifications which aren't possible to implement while we are developing on desktop so will be left to the end where we perhaps consider emulating such a notification **if there is time.**

Meeting the Requirements

1. Ability to schedule cycling sessions with specific weather
    a. In the settings part of the application, there is an ability to deal with different preferred weather scenarios, where there are a few simple choices, sunny, cloudy and rainy. Though we had considered offering more options, the simplicity of this fed into the intuitive requirement of requirement 4.
    b. Then, the system will base the recommended timings for the cycling based on this selected preferred weather option.
2. Be able to enter preferences with prefered times
    a. As with requirement 1, this has been met in the settings screen, with the user being able to select the preferred time.
    b. Again, it would have been easy to overcomplicate the system, by allowing the user to select a number of period of time where they are free, or even allowing us to look at their google calendar or suchlike and see when they are free.
    c. However, the much simpler system ensures that the system is as simple and easy to use as possible.
3. Be able to deal with varying lengths of times of cycling
    a. The varying lengths of time of cycling are selectable using a drop down box in the settings menu.

4. Usable without internet connection
    a. All the accessing of data (see task models) first check whether there is any internet connection, before accessing data from the API if there is and accessing data from memory if not.
    b. Therefore, by caching the most recently updated data in memory means that the system is usable even without an internet connection, with the people being able to see upcoming weather and even change the preferred timings and allowing the application to plan a session.
5. Reasonably quick to access data
    a. By limiting the data required as all coming from a single API - OpenWeatherMap, we are ensuring that the data should be reasonably quick to access. Additionally, since we require the user to press a button to view the available sessions, there should not be too long periods where the screen is loading because the session has not been found yet - in fact this planning will likely be done in the background and shown to the user when they press the button.
6. Ability to notify users about appropriate times
    a. This is impossible to accomplish in any kind of prototype for this Interaction Design course due to the inability to show ‘phone-like’ notifications on a desktop Java application - as we are attempting to implement it.
7. Ability to notify users about changing weather while cycling
    a. As for Requirement 6, we have chosen to largely ignore this requirement. However, if it appears that we have more time, it may be possible to emulate notifications, though this is a very low priority.
8. Intuitive and Powerful interface
    a. For each screen, we have considered these two words;
    b. For the main screen, the most important information is shown most clearly to the user, therefore they can access it quickly and easily. Everything is clearly labelled, so the system is clearly intuitive. It however, packs lots of information, specific to our stakeholder, including wind speed and direction of that day, as well as the weather for the next 5 days. It also has intuitive designs for the settings screen, with the universally recognisable gear symbol. Additionally, it is very intuitive that tapping on any day will yield more information about that day’s weather.
    c. For the settings window, the focus was been on intuitive design, with all fields being labelled. Wherever possible, drop down menus have been used to reduce the probability of user error. However, the high customizability of the application to one’s own needs is clearly a powerful feature. Finally, the back button is easily recognisable (universally used) and is therefore very intuitive.
    d. The daily view has a striking balance between intuitiveness and power. The most important information is shown at the top of the screen, with the weather (using the commonly used symbols for clarity) as well as high, low and wind speed and directions being listed. Then for power, more information is shown for each 3 hour period, though this is much smaller, with the weather, wind speed, direction, temperature and the probability of precipitation.