# 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.

**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.

**Lo-fi to Hi-fi Design and Implementation**
Using our designs shown below, 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 split into two groups where one group will work on the settings interface and data acquisition via the API, and the other will work on the main screen interface and the detailed day view interface shown in the designs. This will give us stand-alone interfaces and methods for retrieving data for the time being.

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.**

## Brainstorm

## Wireframe - Main Screen

## Wireframe - Daily View

## Wireframe - Settings

## Design - Main Screen

## Design - Daily View

## Design - Settings Screen

## API Data Flow

## Weather Iconography

TODO: Write something here!

**Task: 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 **Story Board**:

**Task: 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.

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

**StoryBoard -**

**Task: Change time and duration settings**
**Task Model -**

**StoryBoard -**

**Lo-Fi to Hi-Fi Pathway -**

**Meeting the Requirements**

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