Task 2

Below our design process can be seen, from our initial brainstorm to the individual task models for certain activities. Also included are screen designs, a wireframe 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.

From discussions with our stakeholders which occurred while we gathered data, 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.

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 three groups where two groups will work on interfaces and the third will work to get the data we need using our chosen API detailed below. This will give us stand-alone interfaces and methods for retrieving data for the time being.

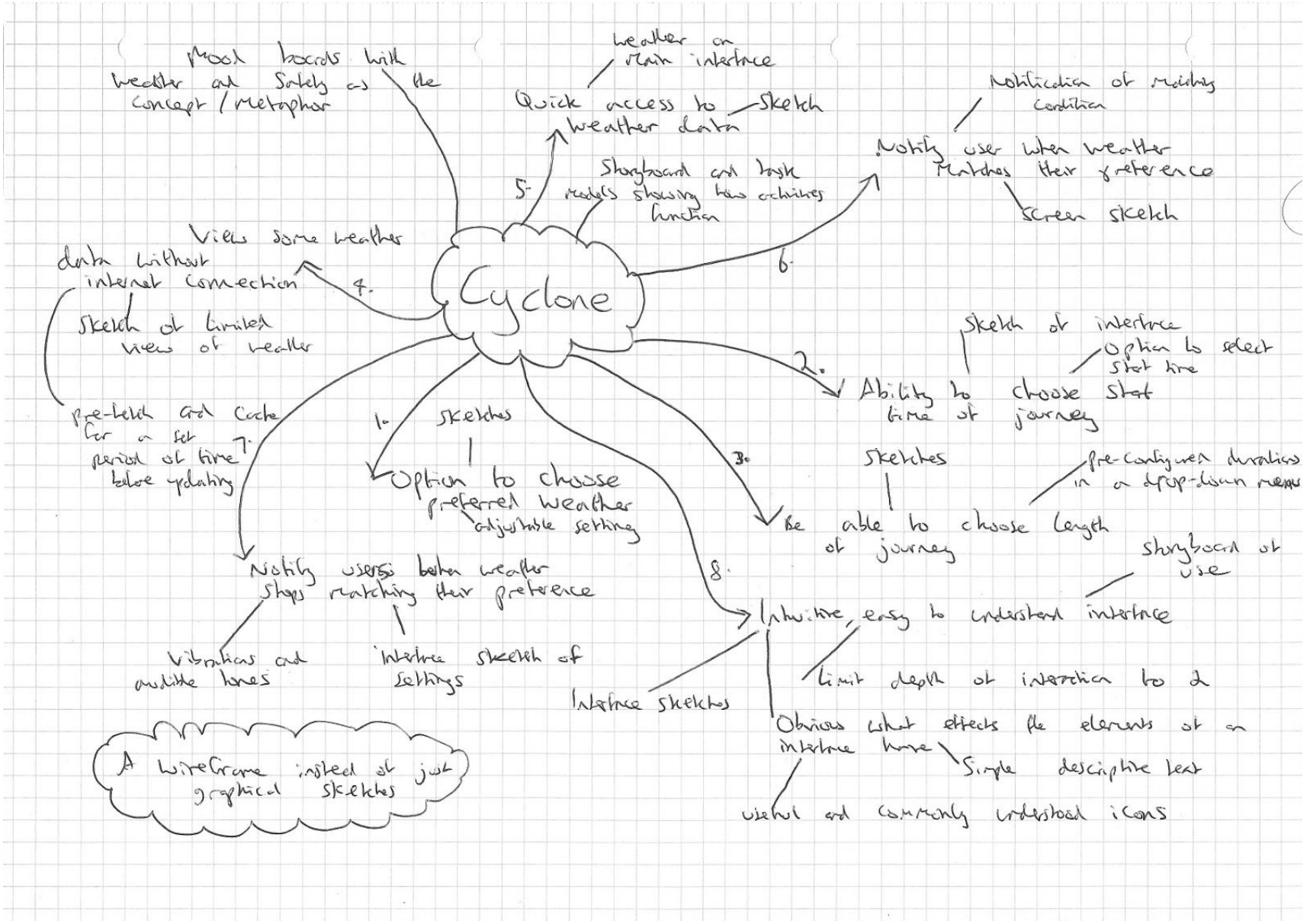
Afterwards, we will have these interfaces interface and interact with each other by linking them together through buttons that allow movement between them. This will correspond with details outlined in storyboards and task analyses of how activities will be completed and how the user can move between 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.

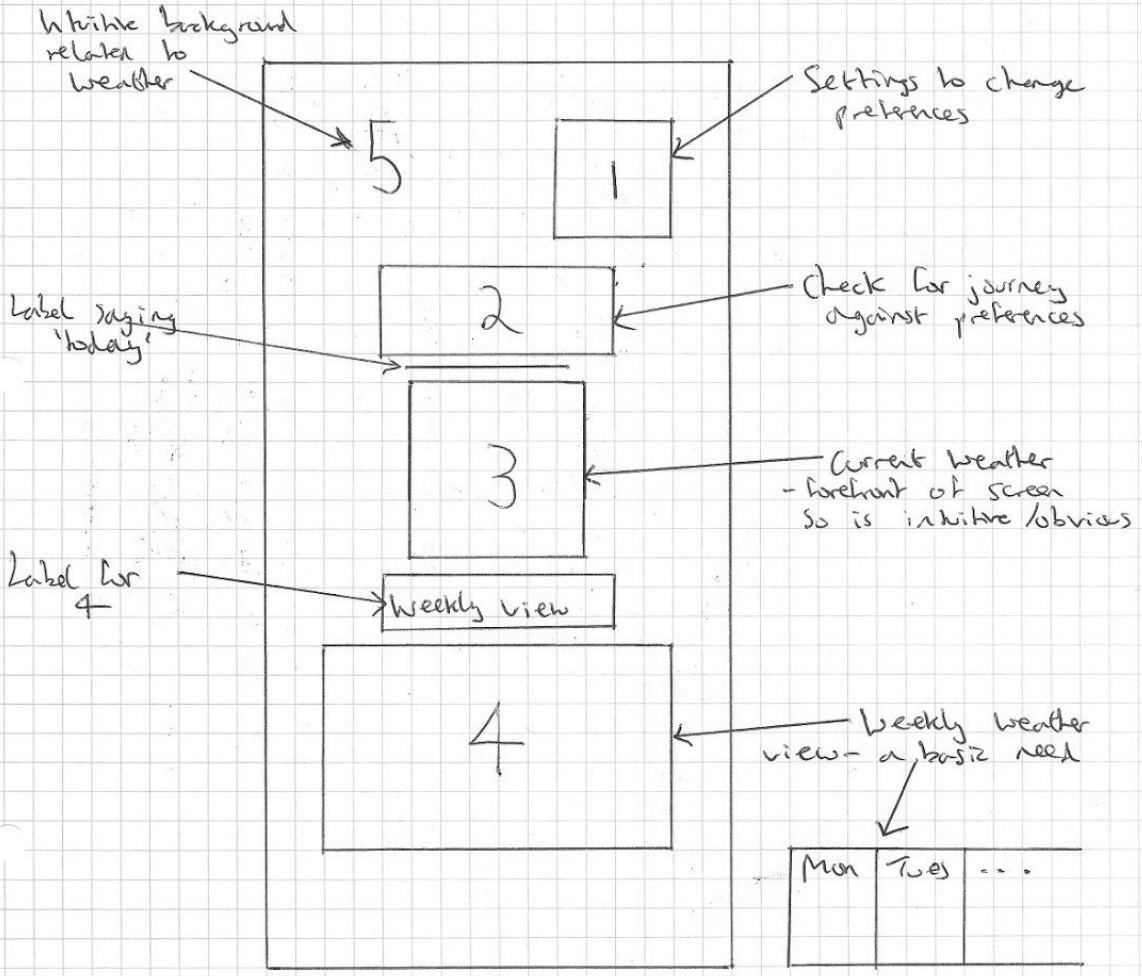
There are potential risks involved, one of them being deviating away from the requirements when implementing the designs. To avoid this, it may be useful not to spend long periods of time on the implementation so as to not lose focus of the true goal, and to keep the requirements nearby so they are always 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. (1, 2, 3)
2. (4, 5, 8)
3. (6, 7)

Where the numbers correspond to the requirements in our list from task 1. 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.**


Cyclone
Mock boards with weather art Solely as concept / metaphor
Weather on main interface
Quick access to weather data
Sketch
5.
Storyboard and task models showing how activities function
Notification of raining condition
Notify user when weather matches their preference
Screen sketch
6.
View some weather data without internet connection
4.
Sketch of limited view of weather
Pre-fetch and Cache for a set period of time before updating
7.
1.
Sketches
Option to choose preferred weather adjustable setting
Notify users when weather stops matching their preference
Vibrations and audible tones
Interface sketch of settings
2.
Sketch of interface
Option to select start time
Ability to choose start time of journey
3.
Sketches
Pre-configured durations in a drop-down menu
Be able to choose length of journey
8.
Storyboard of use
Intuitive, easy to understand interface
Limit depth of interaction to 2
Interface sketches
Obvious what effects the elements of an interface have
Simple descriptive text
Useful and commonly understood icons
A wireframe instead of just graphical sketches

Wireframe - Main Screen



1 - Settings button, opens settings screen

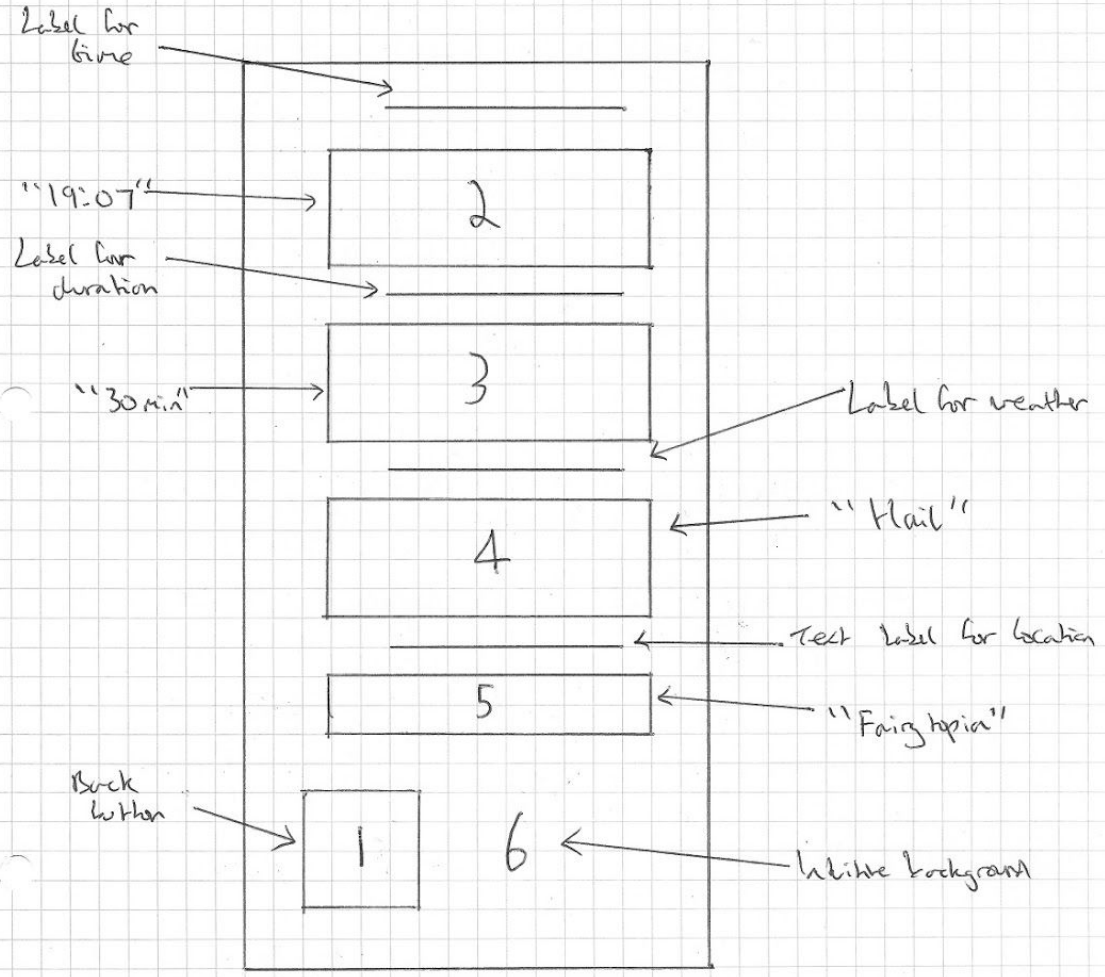
2 - Check for journey button, determines if weather matches preferred settings for a journey

3 - Current weather, possibly an image with temperature

4 - Weather for next week, icons and temperature

5 - Background, possibly an image to reflect weather, static

Wireframe - Settings



1 - Button to go back to main screen, icon with arrow pointing left (commonly seen as back).

2 - Drop down menu - Select preferred start time

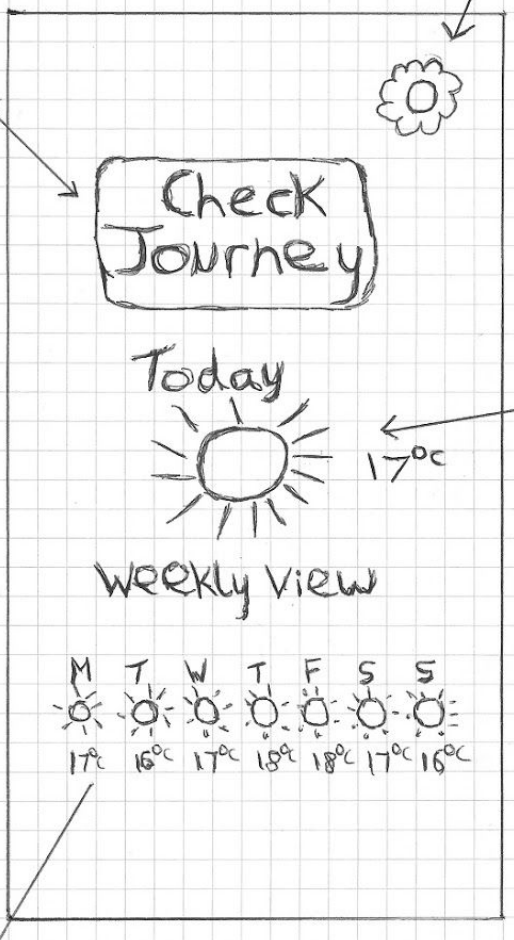
3 - Drop down - Select preferred journey length

4 - Drop down - Set preferred weather

5 - Drop down - Choose location

6 - Background - carries over from main screen

Screen Design – Main Screen



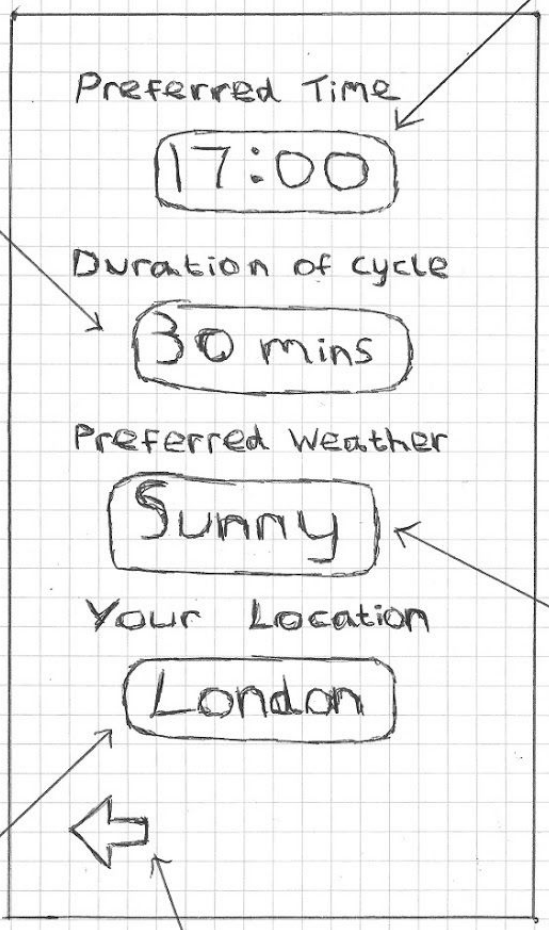
This is the settings button used to manage the preferences as per the requirements by transitioning between interfaces. Pressing it will switch interfaces. It will have the well-known gear icon so it is intuitive. It will be implemented via a button with a press event to transition.

With the preferences set, the journey checker can report back if the conditions are suitable for a cycle or not. This is done by pressing the button, which then checks preferences against weather data, as per requirements, and can be implemented via a button with appropriate methods to check the weather.

Central is the current weather data, adhering to the requirement of being easy to spot the weather data. It displays an appropriate icon and temperature according to the weather, and can be implemented with an icon container and code to detect the weather using the API.

Here we display predictions on a weekly basis, a basic weather app need, by retrieving data from the API similar to how is done for the current weather but on a weekly basis. (Assume in the design that today is Monday so it shows next 7 days including today). The design highlights that to maintain aesthetics we need to ensure adequate spacing.

Screen Design – Settings



Here, at on all settings, we have a label to tell the user what preference they are setting - this makes it easy and intuitive. The time can be changed by pressing on it and choosing, which affects how we look for valid journeys according to start time. This can be implemented with a label and a menu box allowing times to choose from.

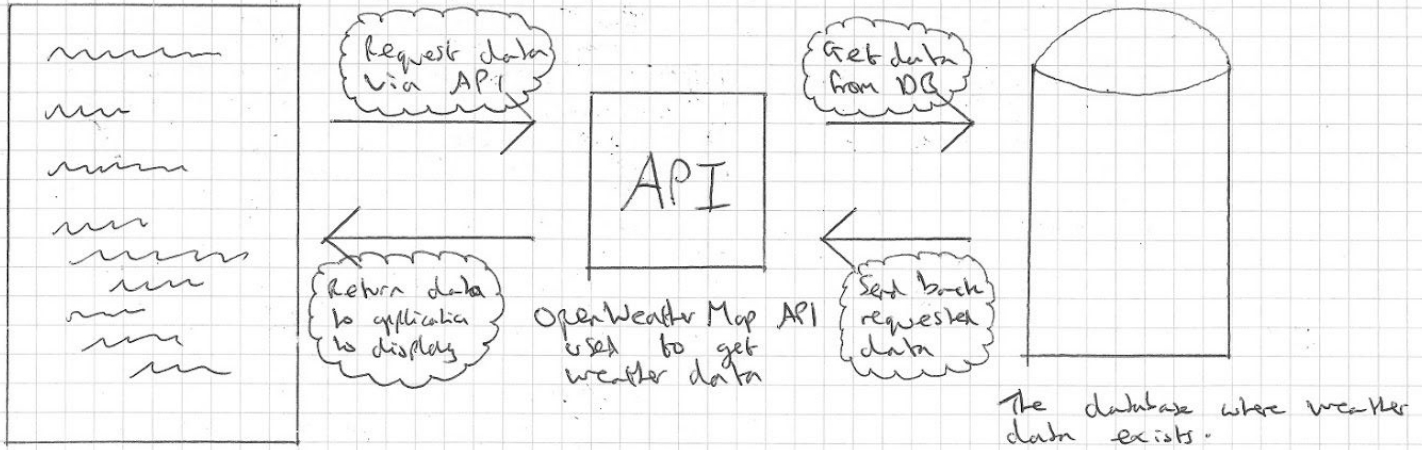
Our users cycle for different durations and the requirement is setting this value. By pressing on the one can choose a time ranging from 30 minutes to 5 hours, implemented with a drop-down menu.

The user can set their preferred cycling weather as required by pressing and choosing. This is implemented by a drop-down menu.

A location can be set as required by pressing and choosing from a drop-down menu

This is a back button, given the intuitive and commonly known backwards icon, to transition back to the main screen to see the data based on new preferences. It can be implemented with a button and a press event.

Weather API Data Flow



Java code which will take preferences set in the settings and make use of the API to get relevant data.