

GUI Deliverable 3 – Evaluation - Group 41

Section 1 - Summary of the Evaluation

Firstly, we decided to carry out heuristic evaluation of our interface. We did this by asking other evaluators to use our interface and detect any issues with them, according to the ten heuristics we were assessing our interface on. We recorded their responses and were able to summarise them, allowing us to see what kind of issues our interface had, and also prioritise them by giving them severity ratings.

We found that a few issues needed to be addressed. The issues that were found were related to the following heuristics - familiarity of symbols language, error prevention, efficiency of the interface, helping users to recognise and recover from errors, and help documentation. Following this, we designed how the interface would look with the added improvements. The improvements suggested include creating weather symbols for night time, having loading screens to prevent sluggishness, adding a message to show no conditions matching and a help page to ensure users can navigate their way through the application.

This evaluation was very helpful to us as we were able to identify issues with our interface with the help from evaluators. This allowed the identification of issues that we may not have found if we had evaluated it ourselves, and also removes a fair amount of bias as it was evaluated by people who were not involved in the project.

Section 2 – Evaluation Process

We decided to use heuristic evaluation to evaluate our application and interface. We used the usability principles (the ten heuristics, as identified by Nielsen and Molich) to measure the usability of our interface.

We presented the interface to people who were completely new to our application, and so they had no prior knowledge as to how it worked. The people who were involved in the evaluation had a varying but above average level of experience with using different interfaces. For this reason, they may not be regarded as experts, but were very well-versed in many types of interfaces. We involved 5 evaluators in our evaluation as different people would be able to identify different issues with the interface, and they would be able to analyse the interface from different perspectives, which could in turn allow for more feedback to be obtained.

Although those that were involved were adept at using different interfaces (including other weather applications), our application presented additional features (that are not present in other applications) that would specifically assist our primary stakeholder group with their activities. This meant that at the very least, the way these features functioned were completely new to the users.

We evaluated our interface based on these ten heuristics:

1. Keeping the user informed with helpful messages and headings, so the user is aware of what is happening during each interaction. Headings were relevant to the function of that section, such as being able to time the run, selecting temperature conditions for the run, etc.
2. The familiarity of icons, symbols and language to users, e.g. an icon of rain droplets to indicate rain.
3. The method of reversal if the users performed an incorrect interaction. Although our interface did not have an explicit undo button, it did provide reversal methods.
4. Consistency in language, aesthetics and symbols throughout the interface.
5. Helpful error messages in the case that something goes wrong.
6. Making sure the user would not have to remember things from other screens or pages. This meant that any information that is needed would be present on the page.
7. The efficiency of the interface.
8. Designing a simple yet attractive interface that allowed users to be comfortable with the interface, encouraging them to use it (no extraneous information).
9. Helping users to recognise errors and recover from them, e.g. if the user presses the 'Go' button without selecting the start time and duration of the run.
10. The presence of help documentation.

We asked the 5 evaluators to assess the interface based on the heuristics - after using the application, we questioned them about each heuristic. We were then able to transcribe their responses. Evaluators carried out their evaluations of the interface separately, as we wanted to ensure independent and unbiased evaluations from each evaluator. After collecting the results, we were able to summarise their statements. We also assigned each issue with a severity rating, based on the frequency, impact and persistence of them.

Section 3 – Findings

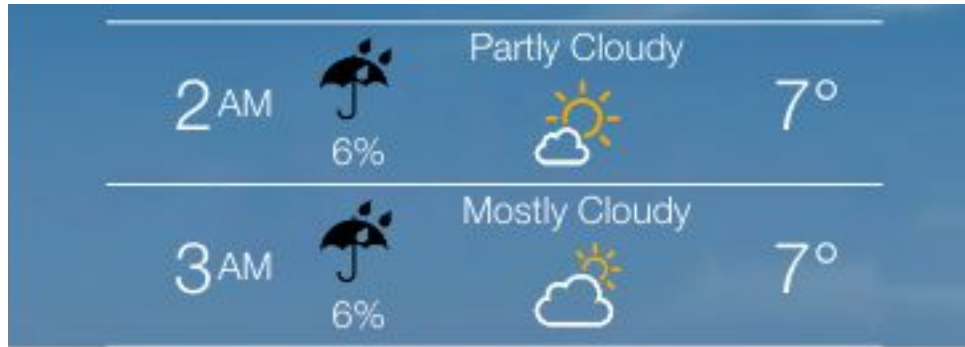
After conducting the evaluation, we found that our application had many positives but also had a few weaknesses. Our findings were based on the ten heuristics we assessed the application on. The following table summarises the responses from our evaluators.

Heuristic	Overall Findings	Severity Rating
Keeping the user well-informed	All users were well-informed by the interface, and knew what to do during each interaction, along with how to perform these actions. This was attributed to helpful headings that were placed appropriately throughout the interface.	N/A
Familiarity of symbols and language	<p>The icons were among those that were familiar to the user – all participants were aware of what each icon meant, along with the specific terms used in the interface, such as humidity and precipitation.</p> <p>One thing that was pointed out was that the some of the icons do not change according to the time of day (e.g. a cloud with a sun was used to represent cloudy weather even during night time).</p>	<p>N/A</p> <p>1 – Cosmetic issue.</p>
Reversal	Although our application did not have an explicit undo button, it did provide a means of reversal when necessary. For example, when a user pressed the 'Go' button on the homepage without selecting a starting time and duration, no weather data would be displayed. After this event, a 'Return' button would appear to allow users to go back to the original screen. This would also be the case if the user wanted to change their starting or duration. This was well-received by evaluators and sufficed as a method of reversal.	N/A
Consistency in language, aesthetics and icons	The consistency of elements in our interface was also praised. We used the same background in all separate pages of the interface, so users did not need to familiarise themselves with new information. The same icons were used in the appropriate places – for example, a cloud with rain was used to represent rainy conditions throughout the interface.	N/A
Error prevention	Evaluators pointed out that there were some cases where the API information was not	3 – a major problem, albeit

	loaded in the correct order, when displaying the hourly forecast. This error did not provide an appropriate message indicating that it was not working correctly, and the users in our evaluation pointed this out.	not happening often.
Recognition rather than recall	<p>Evaluators had stated that it was very helpful that they could select options rather than enter them. This was a positive, as one of our goals was to minimise the application of user memory.</p> <p>Also, it was mentioned that everything that the evaluators needed to perform a certain task was always present on the screen, also reducing memory load.</p>	N/A
Efficiency of the interface	<p>All users stated that it was an efficient interface, with almost no wait times for all tasks. Although, some users pointed out that the data in the hourly forecast appeared sequentially (one after the other) instead of all at once.</p> <p>Another efficiency error was that was noticed by evaluators was that the weather data in the homepage would not load immediately, resulting in the alternate message being displayed.</p>	<p>2 – a minor error that needed to be addressed, but it is a low priority.</p> <p>2 – minor problem as it was a response problem – the data would load just after a split second, just not immediately.</p>
Keeping everything simple (no extraneous information)	Users stated that there was no information that distracted them from the task at hand.	N/A
Help users recognize, diagnose, and recover from errors	An error that did not provide an error message was in the case that the user entered weather conditions they desired to run in, but there were no hourly conditions that matched their query. This meant that some users were not sure if the application was slow in response, or if the times with the conditions the user selected were not found.	2 – a minor error that needed to be addressed, but the interface function without the change.
The presence of help documentation	Our interface did not include a help page, and this was pointed out by users.	2 – a minor error that needed to be addressed.

Evidence of Violated Heuristics

Icons



The image above illustrates the second heuristic (familiarity of icons and language) being violated. Although the icons used are familiar to the user (showing an icon representing clouds when the weather is cloudy) and they were able to understand them, the usage of the icons are not always appropriate. For example, in the above screenshot, the icon that is used to represent the weather shows a cloud with a sun, even though the time of day is 2AM and 3AM. This does not make the most sense as the part of the icon that shows the sun is not representative of the actual time of day (as it is almost always dark during this time). This needed to be sorted so that users were informed of what was happening, through the usage of appropriate icons.

Error Prevention

Home	Forecast	Optimise
3 PM	Overcast 3%	12°
5 PM	Mostly Cloudy 1%	13°
7 PM	Mostly Cloudy 1%	11°
4 PM	Mostly Cloudy 1%	12°
9 PM	Partly Cloudy 2%	9°
12 AM	Partly Cloudy 4%	8°
6 PM	Overcast 0%	12°
8 PM	Mostly Cloudy 1%	11°
1 AM	Partly Cloudy 5%	7°

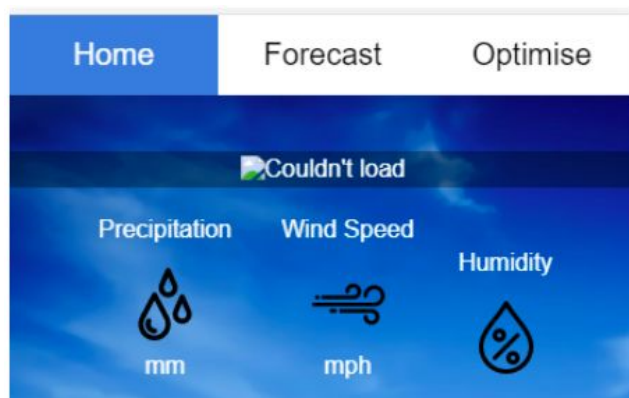
This diagram illustrated another heuristic that was violated (error prevention). We designed our application to not suffer from any major errors in the first place, however this was a big error that occurred. Sometimes, when loading up the hourly forecast in the Forecast page, the times would be shown in the wrong order.

This error did not occur all the time, but the few times it did, it was pointed out by users. Since this was an error that actually affected the usability of the application, we decided to label it as a major error that needed fixing.

Efficiency

Another heuristic that was violated was the efficiency. The picture on the left shows the forecast page, which is supposed to load the hourly forecast for the next 12 hours. However, since our application retrieves API data, this was somewhat slow to do and each hour was loaded after each other, instead of all at once. This was difficult to show in the screenshot; however, it does show that only information for the next 8 hours is displayed, rather than 12 – meaning that all the data was not loaded all at once.

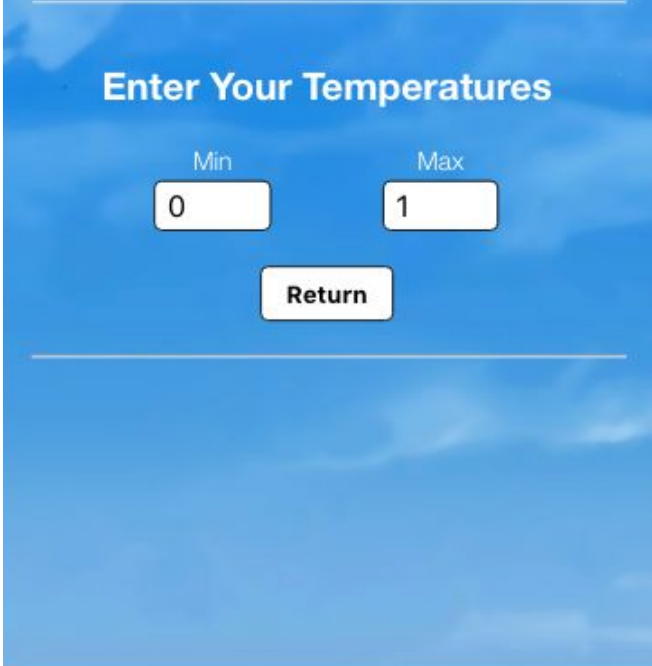
The blank area at the bottom of the screen represents the part of the hourly forecast that has not been loaded yet.



This screenshot also shows the efficiency heuristic being violated. Although it was only for a split second, the weather data in the home page would not load immediately, resulting in the alternate message being displayed. This was a very minor error but we still feel that it needed to be corrected.

Recognising and Recovering from Errors

The screenshot on the right shows another heuristic being violated (being able to recognise errors and recover from them). For this functionality, users select the conditions they would like to run in (minimum and maximum temperatures) and the interface would retrieve the hours in which those conditions match (would display 12 hourly conditions from the next 36 hours). The error here is that, when users select temperatures that do not match, nothing would be displayed. This was a negative point as users were not able to recognise whether it was an error on their part, or if the interface was just taking a lot of time to respond.



The screenshot shows a form titled "Enter Your Temperatures" on a blue background with a cloud pattern. It features two input fields: "Min" with the value "0" and "Max" with the value "1". Below these fields is a "Return" button. A horizontal line is positioned below the button.

Help Documentation



The screenshot shows a weather application interface. At the top, there are three tabs: "Home", "Forecast", and "Optimise". The "Home" tab is selected. Below the tabs, the location "London" is displayed. The current temperature is "14.1°" with a sun and cloud icon and the text "Mostly Cloudy". Below this, there are three weather metrics: "Precipitation" (5 mm), "Wind Speed" (0 mph), and "Humidity" (78%). At the bottom, there is a section titled "Enter Run Time" with two input fields: "Start" (16:00) and "Duration" (1 hour). Below these fields is a "Go" button. A horizontal line is positioned below the "Go" button.

The screenshot on the left shows the help documentation heuristic being violated. We originally did not include a help icon/page as we wanted to provide meaningful headers that would prevent a need for help pages. Although users were aware of what to do at each stage of each interaction, a help page should have been included.

Section 4 - Improvements

Improving Icons



The picture on the left shows how the usage of icons could be improved. As stated earlier, in some cases, the icon would show a sun during night time, which was not suitable. Hence, we decided to use another icon to represent certain weather conditions during night time, so the user is more informed and satisfied with their usage of the interface.

This change could help our stakeholders such that if their run extends into the night (or whenever it becomes dark), users will be aware of the specific hour of when it becomes dark - hence, they will be able to modify their run appropriately if needed.

This icon would be present in all pages where necessary. It would be present in the homepage where the weather is shown at the top of the screen, and also would be used in the Forecast page when showing the hourly forecast, where appropriate (where the position of the icon is currently). It would also be present in the Optimise page in the same fashion, as described above.

Improving Errors

The image on the right shows how the error prevention aspect could be improved. Earlier, some of the hourly conditions would not appear in the right order. However, the updated picture shows how the interface would look if all of the data in the hourly forecast loaded in the correct order. In this improvement, each 'bar' representing each hour of the hourly forecast is displayed in the correct consecutive order.

This would be helpful for our primary stakeholders (runners) and also anyone else who would like to use the interface as a general weather application, as it shows all of the times in the correct order. This would allow users to easily see what the weather would be like for the next few hours, rather than looking at a jumbled forecast, and trying find the times that are not in their sequential order.

Home	Forecast	Optimise
	2 PM	Chance of Rain 5°
	96%	
	3 PM	Chance of Rain 7°
	62%	
	4 PM	Chance of Rain 8°
	46%	
	5 PM	Chance of Rain 8°
	48%	
	6 PM	Chance of Rain 8°
	45%	
	7 PM	Chance of Rain 8°
	42%	
	8 PM	Chance of Rain 7°
	43%	
	9 PM	Chance of Rain 6°
	41%	
	10 PM	Chance of Rain 6°
	37%	

Improving Efficiency

Home	Forecast	Optimise
3 PM	Chance of Rain 80%	6°
4 PM	Chance of Rain 35%	7°
5 PM	Chance of Rain 35%	8°
6 PM	Chance of Rain 37%	8°
7 PM	Chance of Rain 39%	8°
8 PM	Chance of Rain 43%	7°
9 PM	Chance of Rain 43%	6°
10 PM	Chance of Rain 37%	6°
11 PM	Partly Cloudy 12%	6°
12 AM	Partly Cloudy 11%	5°
1 AM	Partly Cloudy 11%	4°
2 AM	Partly Cloudy 11%	4°

This image shows how the efficiency could be improved. Earlier, the weather data in the forecast page would load in one 'bar' at a time. This image shows how the page would look straight away - all of the data would be loaded in immediately, allowing users to view all 12 hours without waiting a few seconds.

This is beneficial to our stakeholders as they can see how the weather conditions will be for the next 12 hours of the day. The efficiency aspect is very helpful to our primary stakeholders (runners) as they may be busy on a run at the time they are viewing this page. Therefore, the fact that they would be able to view this data immediately would prevent any delays from occurring in their run.

Instead of loading the 'bars' one at a time, all of the data would be displayed at once. If this would take some time, an appropriate loading icon would be displayed indicating that the data will be displayed soon.



This image also shows how the efficiency could be improved (in the homepage). One of the issues that was pointed out by evaluators was that the weather data that was retrieved from the API would not load immediately (instead would take a split second to load). This resulting in an alternate message being shown temporarily. The image here shows how that interface would be if this data would loaded in straight away, so the users would not have to wait to see the weather data on the homepage.

Also, if it does so happen that there are weather data retrieval issues, instead of showing the alternate text, a loading icon (as on the right) could be shown instead.

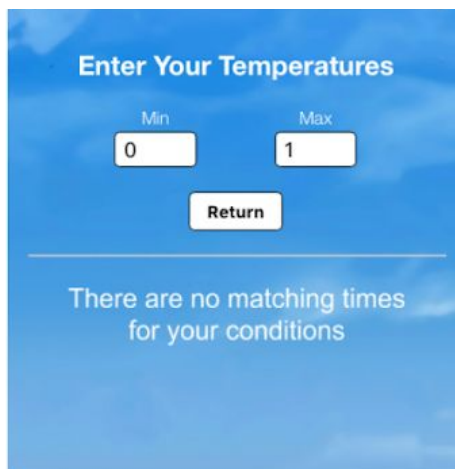


The weather data being loaded in immediately would be beneficial to our stakeholders, as they may be busy with their run while using the application. Hence, it would be much more preferable if they could see the weather in the homepage as soon they open the application so all delays would be avoided.



The loading icon is more suitable than alternate text in case the weather data cannot be retrieved straight away, as it abstracts away from the underlying problem and allows users to see a smoother user interface (with a familiar icon) when encountering a problem.

Improving Recognition and Recovery from Errors



This image shows how recognition and recovery from errors could be improved. When the user enters minimum and maximum temperatures, there is now a suitable message informing the user that there are no matching times, in the case that their temperatures do not match any within the next 36 hours.

This is beneficial to our stakeholders as they can now be informed that for the conditions they entered, times could not be found. This means that they can go back and edit their minimum and maximum temperatures, as they would know that the error is a user-oriented one, rather than being an interface-oriented one (they would know that the interface is

not just taking long to respond).

Improving Help Documentation



The screenshot on the left shows how the help documentation could be improved. We learned that regardless of how easy an interface is to use, help documentation should be provided in the case that it is needed.

Hence, we solved this by showing how our interface would look with a help icon (located in the top right of the page). This icon would be present on each page and would display the relevant information, depending on the page in which it is clicked.

This would be beneficial to our stakeholders as they would be provided with a help screen for each page in the case they are unsure of what to do, or if they forget how to use the interface. This is helpful as runners (our primary stakeholders) come in all sorts of technical skill and have varying levels of technical literacy - this means that if they are ever in need of the functionalities our application provides and are uncertain of how they should use the interface, the help documentation would provide simple explanations of what each page does and how to do it.

The Home help screen would inform the user that the purpose of this page is for entering when the user is starting their run, and its duration. After the 'Go' button is clicked, the times for the next few hours would be shown.

The Forecast help screen would inform the user that it shows the next 12 hours of hourly conditions, allowing them to see a more general view of the weather.

The Optimise help screen would inform the user that the purpose of this page is for selecting the conditions of the run - the application would then provide information of 12 hours from the next 36 hours, when the conditions match.