Task 1 OSC Design Docs

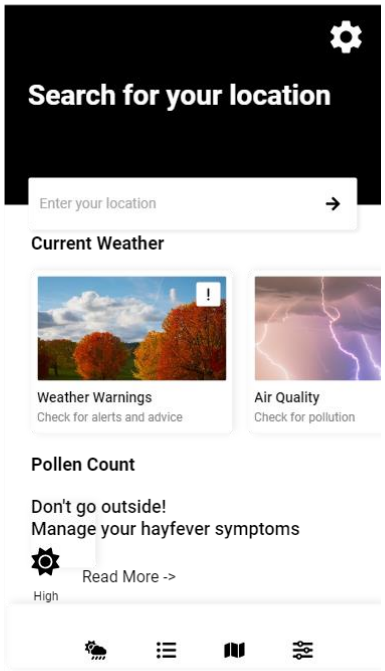
Carl Baines

Contents

Visual/Interface Design 2
Homescreen 2
Settings page 3
Weather Forecast page 4
Weather Map page 5
Entity Relationship Diagram 6
Data Dictionary 6
Weather 6
UserAccount 7
UserOPTIONS 7
Weatherinfo 7
Air quality data 8
Algorithm/Flowchart 9
Main Process 9
Home page sub Process 10
Settings page sub Process 11
Weather Forecast page sub Process 12
Weather Map page sub Process 13
Test Strategy 14

Visual/Interface Design

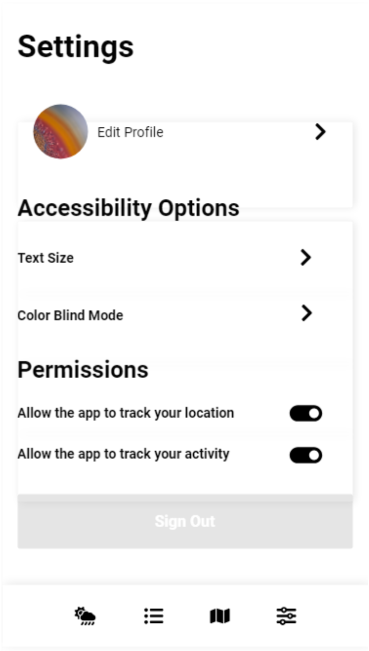
HOMESCREEN



Don't go outside!
Manage your hayfever symptoms

My home screen has a clean and simple design, which allows the content to be easily understood. As well as this, the color scheme of the page is user-friendly as there is a clear contrast between colors. The page has a main input field to allow the user to enter their location in which they would like to find their weather forecasts and the conditions of their environment. It also has widgets for weather warnings and to provide information about air quality. The homes screen also gives advice to the user based on the environmental conditions. A lot of the interactivity on the home screen is icon-based, which makes it easy for the user to understand exactly how to interact with it. The user can easily navigate between this page and the other pages on the app using the icons in the footer.

SETTINGS PAGE



Settings

Edit Profile

Accessibility Options

Text Size

Color Blind Mode

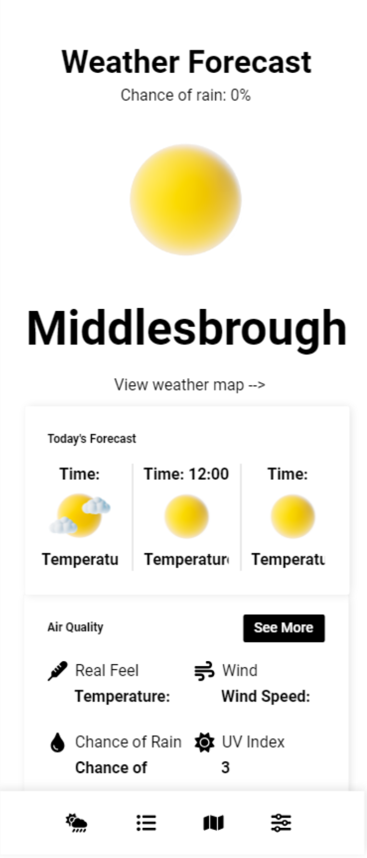
Permissions

Allow the app to track your location

Allow the app to track your activity

My settings screen has a modern design which makes it simple and easy to use. The color scheme of the page is user-friendly as the white background has a clear contrast to the black text and icons, meaning it can be viewed easily. The page allows the user to edit their profile and change the display of the app via the text size, color blind mode options. The app appeals to a wide range of user needs as some of the users may be more visually impaired than others so they can change the text size; accordingly, some users may be colorblind so they can change the display of the app to be appropriate for their deficiency. The user can easily navigate between this page and the other pages on the app using the icons in the footer.

WEATHER FORECAST PAGE



Weather Forecast

Chance of rain: 0%

Middlesbrough

View weather map -->

Today's Forecast

Time:	Time: 12:00	Time:
Temperatu	Temperatur	Temperatu

Air Quality

Real Feel
Temperature:

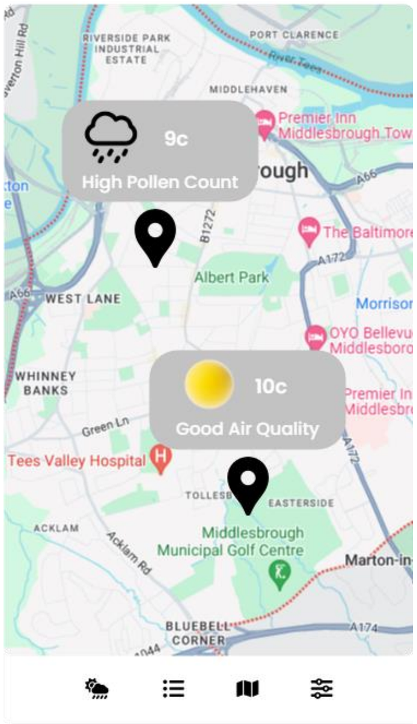
Wind
Wind Speed:

Chance of Rain
Chance of

UV Index
3

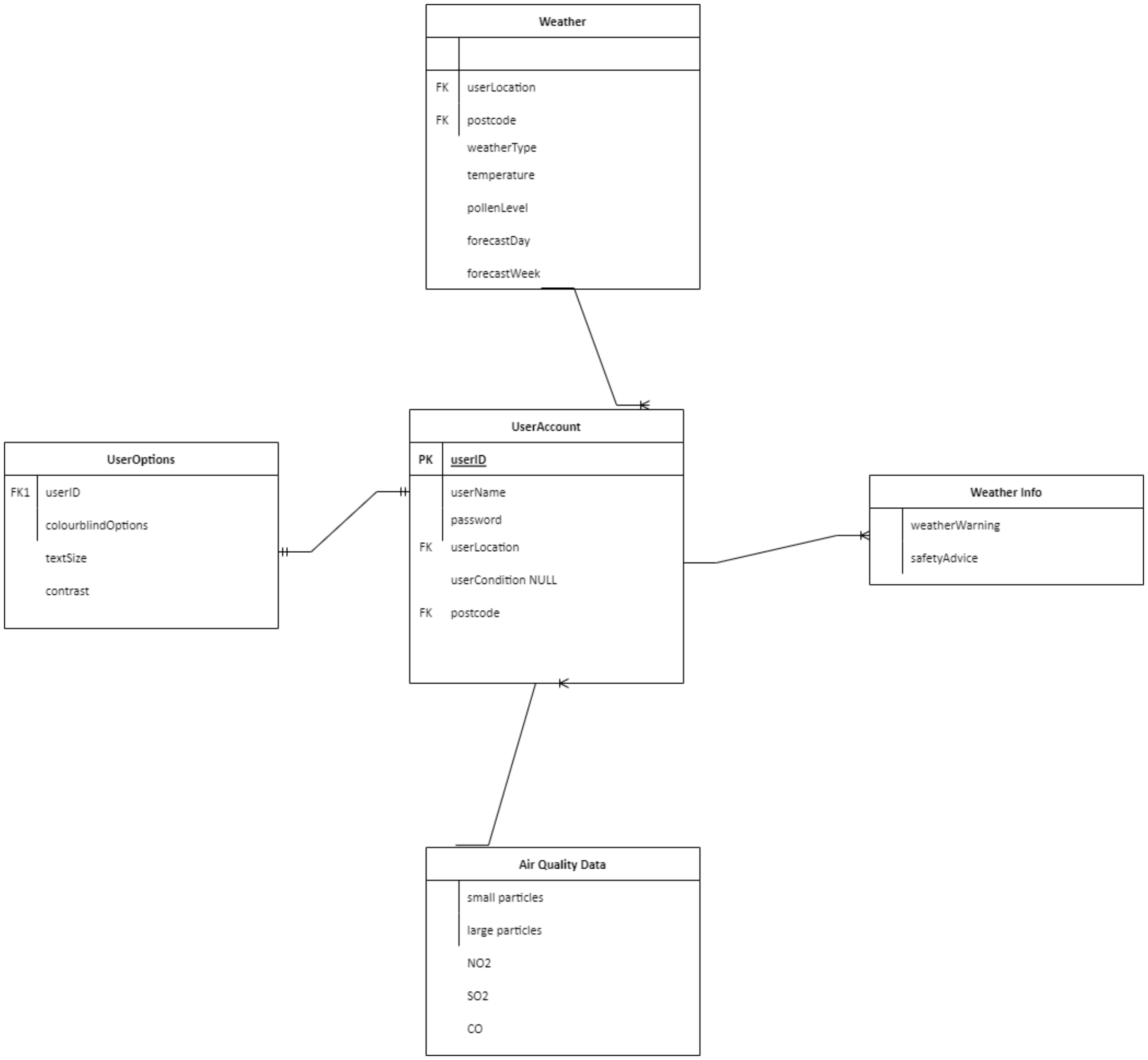
My weather forecast screen has a simple and modern design, which makes it easy to understand. The page displays the weather forecast for the user's location and other things such as air quality, UV index etc. It is user-friendly as most of the interactivity comes from the form of clicking on icons. The user can easily navigate between this page and the other pages on the app using the icons in the footer.

WEATHER MAP PAGE



The weather map page has a simple but effective design. The page displays an app of the user's location and has pinpoints of areas highlighting the air quality and pollen count. The user can easily navigate between this page and the other pages on the app using the icons in the footer.

Entity Relationship Diagram



Data Dictionary

WEATHER

Field Name	Data Type	Field Size	Example of stored data	Primary/Foreign Key
userLocation	String	nVarchar (50))	‘Middlesbrough’	Foreign key

postcode	String	Nvarchar(50))	‘TS10 1A9’	Foreign key
weatherType	String	Nvarchar(50))	‘Windy’	N/A
temperature	Integer	Fixed length	‘19c’	N/A
pollenLevel	String	Nvarchar(25))	‘HIGH’	N/A
forecastDay	String	Nvarchar(255))	‘Sunny with small showers’	N/A
forecastWeek	String	Nvarchar(255))	‘Sunny with patches of rain across the week’	N/A

USERACCOUNT

Field Name	Data Type	Field Size	Example of stored data	Primary/Foreign Key
userID	Integer	Fixed length	‘012345’	Primary key
userName	String	Nvarchar(25))	‘Carl’	N/A
password	String	Fixed length	‘Carl123’	N/A
userLocation	String	Fixed length	‘Middlesbrough’	Foreign key
userCondition	String	Nvarchar(25))	‘Hay fever’	N/A
postcode	String	Nvarchar(255))	‘S10 8YN’	Foreign key

USEROPTIONS

Field Name	Data Type	Field Size	Example of stored data	Primary/Foreign Key
userID	Integer	Fixed length	‘012345’	Primary key
colourblindMode	Boolean	Fixed length	‘TRUE’	N/A
textSize	Integer	Two-digit number	‘16’	N/A
contrastMode	Boolean	Fixed length	‘FALSE’	N/A

WEATHERINFO

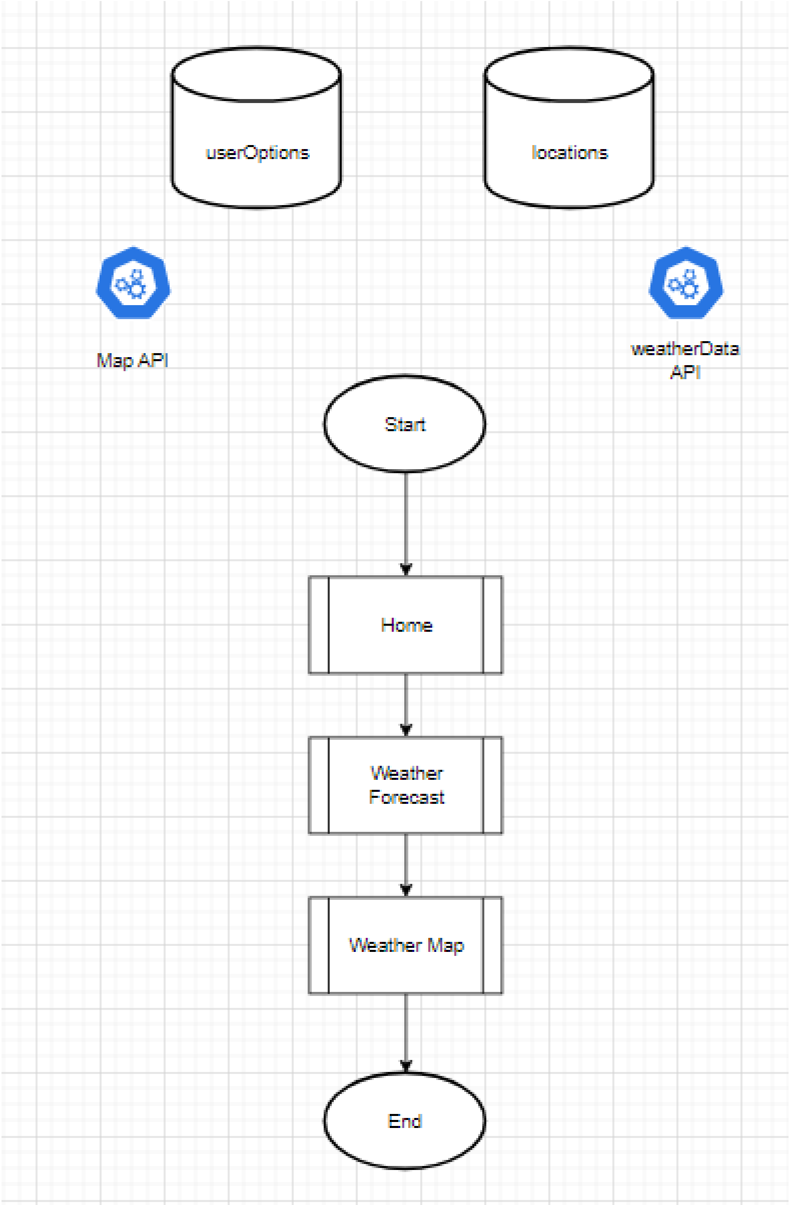
Field Name	Data Type	Field Size	Example of stored data	Primary/Foreign Key
weatherWarning	String	Nvarchar(255))	‘Flash flood warning’	N/A
safetyAdvice	String	Nvarchar(255))	‘Stay inside’	N/A

AIR QUALITY DATA

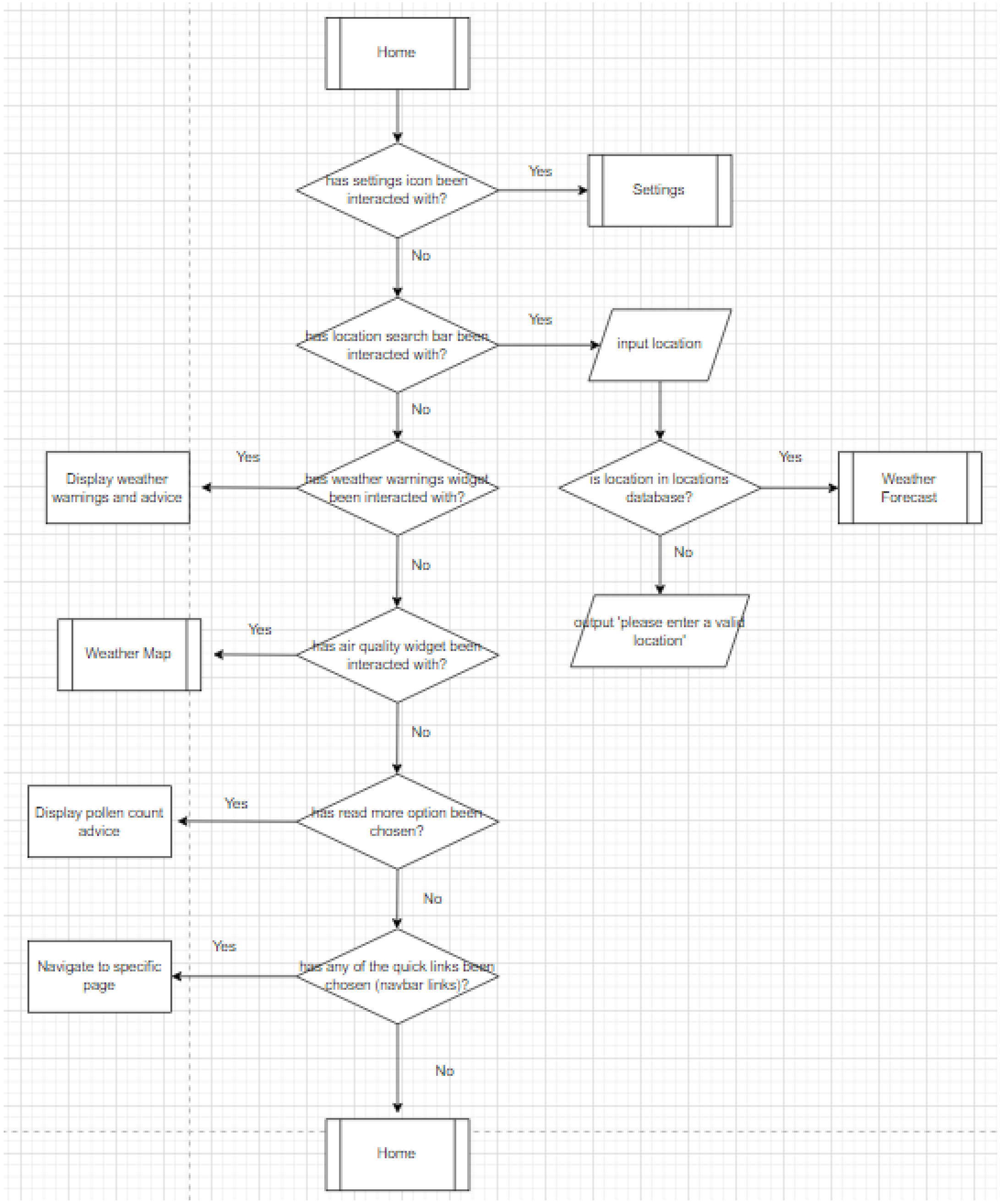
Field Name	**Data Type**	**Field Size**	**Example of stored data**	**Primary/Foreign Key**
Small particles	Integer	Fixed length	‘25’	N/A
Large particles	Integer	Fixed length	‘50’	N/A
NO2	Float	Fixed length	’24.2’	N/A
SO2	Float	Fixed length	’12.5’	N/A
CO	Float	Fixed length	‘14’	N/A

Algorithm/Flowchart

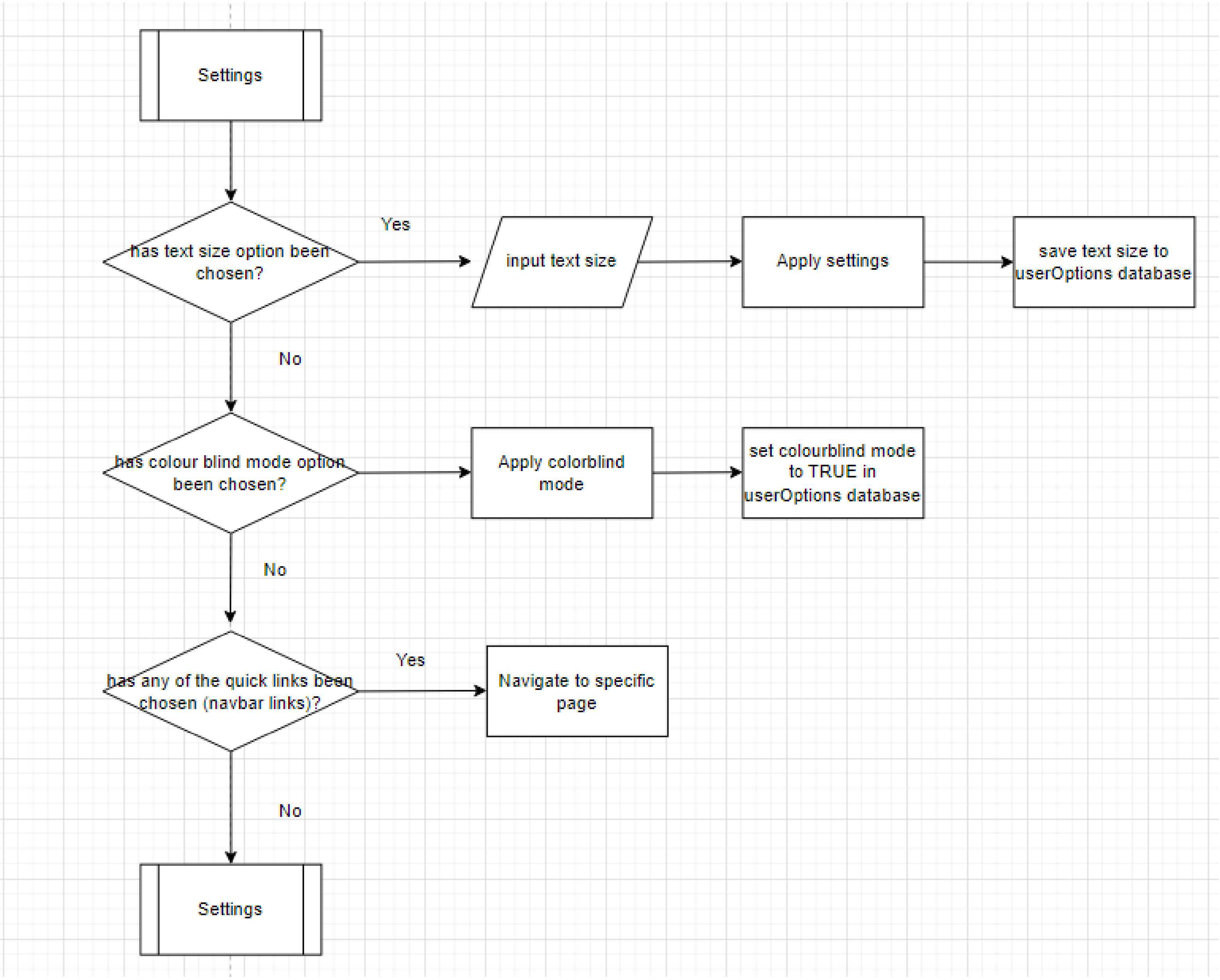
MAIN PROCESS



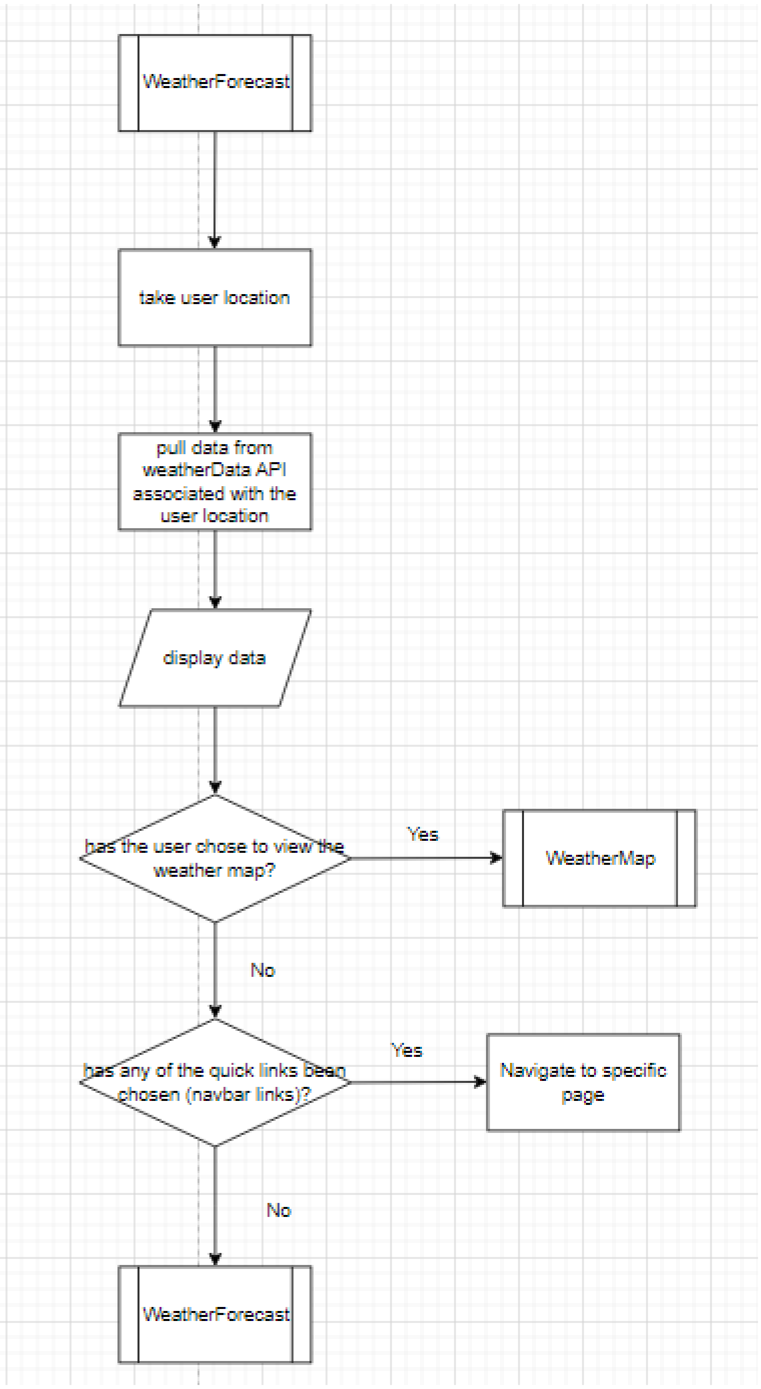
HOME PAGE SUB PROCESS


Home
has settings icon been interacted with?
Yes
Settings
No
has location search bar been interacted with?
Yes
input location
No
Display weather warnings and advice
Yes
has weather warnings widget been interacted with?
is location in locations database?
Yes
Weather Forecast
No
No
output 'please enter a valid location'
Weather Map
Yes
has air quality widget been interacted with?
No
Display pollen count advice
Yes
has read more option been chosen?
No
Navigate to specific page
Yes
has any of the quick links been chosen (navbar links)?
No
Home

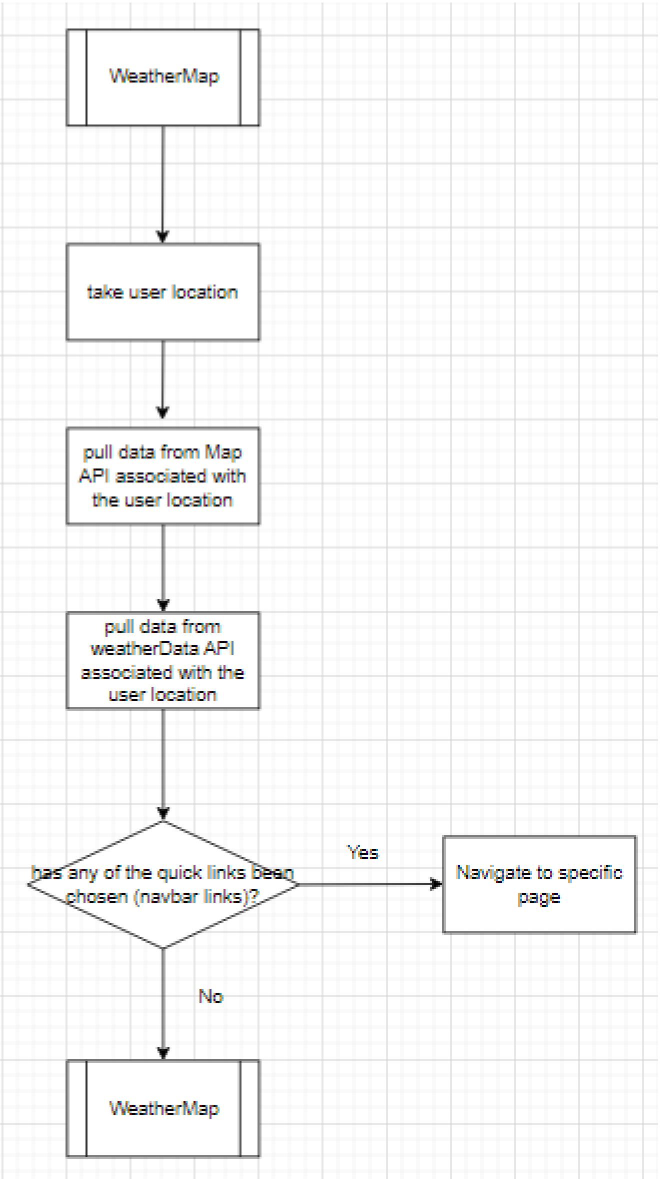
SETTINGS PAGE SUB PROCESS


Settings
has text size option been chosen?
Yes
input text size
Apply settings
save text size to userOptions database
No
has colour blind mode option been chosen?
Apply colorblind mode
set colourblind mode to TRUE in userOptions database
No
has any of the quick links been chosen (navbar links)?
Yes
Navigate to specific page
No
Settings

WEATHER FORECAST PAGE SUB PROCESS


WeatherForecast
take user location
pull data from weatherData API associated with the user location
display data
has the user chose to view the weather map?
Yes
WeatherMap
No
has any of the quick links been chosen (navbar links)?
Yes
Navigate to specific page
No
WeatherForecast

WEATHER MAP PAGE SUB PROCESS


WeatherMap
take user location
pull data from Map API associated with the user location
pull data from weatherData API associated with the user location
has any of the quick links been chosen (navbar links)?
Yes
Navigate to specific page
No
WeatherMap

Test Strategy

Component to be tested	Type of test to be carried out	Prerequisites and dependencies
Home screen	The home page needs to be tested to make sure it successfully navigates to other pages on the site. The home page needs to be stress tested to see if the page runs smoothly when there is a lot of traffic. Stress testing could be carried out by refreshing the page regularly and making repeated inputs. The home page needs to be tested to see if it functions as intended. This can be tested by seeing if the buttons, links and input fields work as intended to see if the user can interact with them. The home page needs to be tested across a range of different devices to see if it is responsive.	The page should allow the user to search for their location and it should redirect the user to the weather forecast page upon input. To do this, the page should query a backend database of locations to search for the user's input. The page should then use this input to pull weather data associated with the user's location, from an API, to then be displayed on the weather forecast page.
Settings page	The settings page needs to be navigation tested as it should be able to return to the pages on the app after settings are applied. The settings page needs to be stress tested to see if the page runs smoothly when there is a lot of traffic. Stress testing could be carried out by refreshing the page regularly and making repeated inputs. The settings page needs to be tested to see if it functions as intended. This can be tested by seeing if you can change the text size of the app and its	The page should allow the user to change the text size and display of the app. The page should save the user's settings in a database and apply the changes to the other pages on the site.

	display through the page. As well as this, all buttons and links need to be tested to see if they work as intended. The settings page needs to be tested across a range of different devices to see if it is responsive.	
Weather forecast page	The weather forecast page needs to be navigation tested as the user should be able to navigate to the other pages on the app from the page. The weather forecast page needs to be stress tested to see if the page runs smoothly when there is a lot of traffic. Stress testing could be carried out by refreshing the page regularly and making repeated inputs. The weather forecast page needs to be tested to see if it functions as intended. The page should interact with a backend database to get the user's location. The page should then pull weather data based off the user's location from an API to be displayed on the page.	The page should interact with the home page to take the user's inputted location and pull weather data that is associated with that location from an API to be displayed on the page.
Weather map page	The weather map page needs to be navigation tested as the user should be able to navigate to the other pages on the app from the page. The weather map page needs to be stress tested to see if the page runs smoothly when there is a lot of traffic. Stress testing could be carried out by refreshing the page regularly and making repeated inputs.	The weather map page should interact with the weather forecast page to get the desired location. It should then pull data from two separate APIs – data from a map API and the weather data already used on the weather forecast page.

	The weather map page needs to be tested to see if it functions as intended. It can be tested by seeing if the map takes the user's location stored in a backend database and whether it pulls the correct data from an API.	