6COSSC004W Mobile Native Development Coursework 2 – SwiftUI Weather Application (2022/23)	
Module leader	PHILIP TRWOGA
Unit	Coursework 2: Real-time data app in SwiftUI
Weighting:	**6**0%
Qualifying mark	30%
Description	A SwiftUI weather&air-quality app using OpenWeather APIs
Learning Outcomes Covered in this Assignment:	This assignment contributes towards the following Learning Outcomes (LOs): LO1 understand language features and programming practice required for native development LO2 apply industry standard tools for design and development LO3 communicate and defend work by both written and oral means
Handed Out:	20th March 2023
Due Date	08th May 2022: 1:00 pm
Expected deliverables	Submit on Blackboard a zip file containing: **Complete XCode 14.X solution - <u>must</u> be in Swift 5.7**
Method of Submission:	Electronic submission on BB via a provided link close to the submission time. The file you upload should have the following naming format: E.g. **6COSC004W**_StudentNumber_firstName_lastName.zip
Type of Feedback and Due Date:	Formative feedback will be provided during tutorial sessions. Verbal feedback on the submitted CW will be provided during an online viva. Students are encouraged to record this feedback at this time. Feedback and marks are due by the TBD. Feedback shall also be given on the Blackboard. Note: All marks will remain provisional until formally agreed by an Assessment Board.

Assessment regulations

See the Assessment guidelines https://www.westminster.ac.uk/current-students/guides-and-policies/assessment-guidelines

for a clarification of how you are assessed, penalties and late submissions, **what constitutes plagiarism etc.**

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website http://www.westminster.ac.uk/study/current-students/resources/academic-regulations

Note: By submitting the work through Blackboard you are acknowledging that this is solely your own work. Any code which is not created by you MUST be clearly commented as such. Any code discovered to not have been created by you will mean that the work will be submitted to academic standards for a potential assessment offence, which may result in a zero mark in the component or whole module.

6COSC004W Native Programming Coursework 2
Individual coursework

Introduction

This coursework is about using **SwiftUI** Framework to build a weather & air-quality application. The weather and air-quality data must be fetched using the following two apis:

Weather-data:
https://api.openweathermap.org/data/3.0/onecall?lat={lat}&lon={lon}&exclude={part}&appid={API key}

Air-quality-data:
https://api.openweathermap.org/data/2.5/air_pollution?lat={lat}&lon={lon}&appid={API key}

You will need to register with openweather.org to obtain API key.

Note that this is a subscription-based API and payment details will have to be added, however, you are allowed 1000 free calls per day, and you can set a limit to number of calls (900) so as not incur any charges.

If for any reasons you are unable to set this up, please contact the module team and a key will be provided for your use.

Very Important Note

The application must be built using SwiftUI and CoreLocation framework.

Frameworks such as Pods, Lottie, SDWebImage, WeatherKit, UIKit are NOT ALLOWED, and any use of these frameworks will result in your mark being capped to a maximum of 30.

You must use CWK2Starter template that has been provided and add where necessary additional models, view-models and views to render air-quality data.

You will be expected to defend and demonstrate your code during viva, failure to do so will be deemed an academic offence and be reported as such.

The application design is as depicted in the images shown below and **no changes** are allowed regarding background images nor navigation between different views. The information presented in each view must be the same as shown in the images. Weather icons must be fetched from openweather.
Background images and other images are in the starter template project.

There are many ways to manage an API call and you are only allowed to use the ones that will be covered and used in the seminar sessions and included in the starter template project.

The functional requirements are:

- When the app launches, all data will be loaded from json files that has London data.
- Navigation between views is tab-based.
- Tab City (**Figure 1**) view gives an overview of the weather with location details.
- Tab WeatherNow (**Figure 2**) gives a detailed weather for the location.
- Tab Hourly Summary (**Figure 3**) gives hourly forecast for 48 hours.
- Tab Forecast (**Figure 4**) gives forecast for 8 days.
- Tab Pollution (**Figure 5**) gives some weather information and air-quality information in one view.
- User can input a new place name by activating “Change Location” button shown in the city view.
- Change location to be managed using a sheet (**Figure 6**) that will be dismissed when the user has entered a new location as shown in the images.
- All views will be updated with real-time data when the location changes.
- The app must safely handle non-existent locations and non-alphabetic entries.
- Location names with spaces must be correctly processed, e.g., San Francisco, etc

The non-functional requirements are:

- The app interface follows Apple guidelines.
- The app interaction is consistent.
- The app data should be correctly formatted.

Design Images

The images are based on iPhone 14 Pro, OS16.1 Simulator.

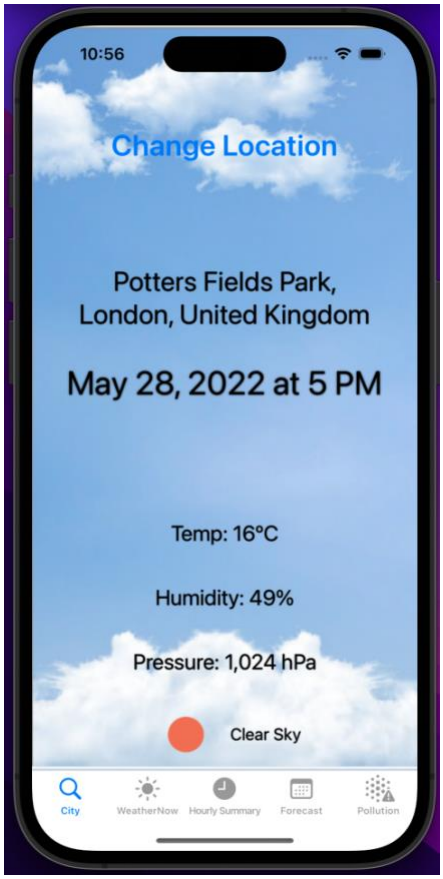


Figure 1

City Tab

Figure 2

WeatherNow Tab

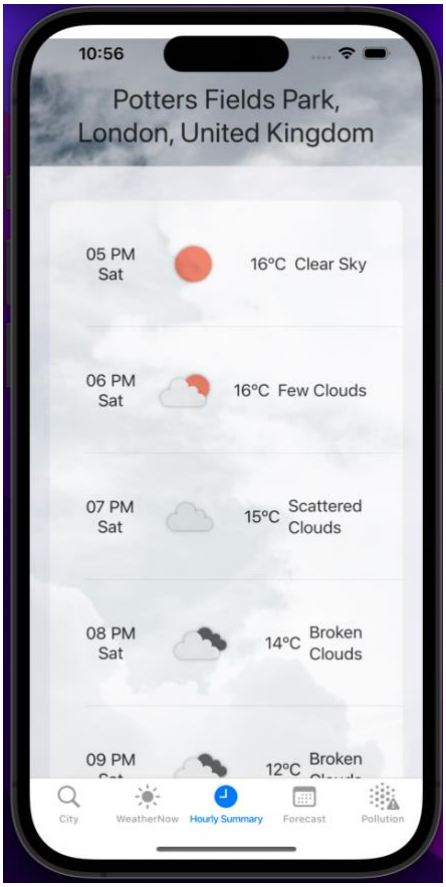


Figure 3

Hourly Summary Tab

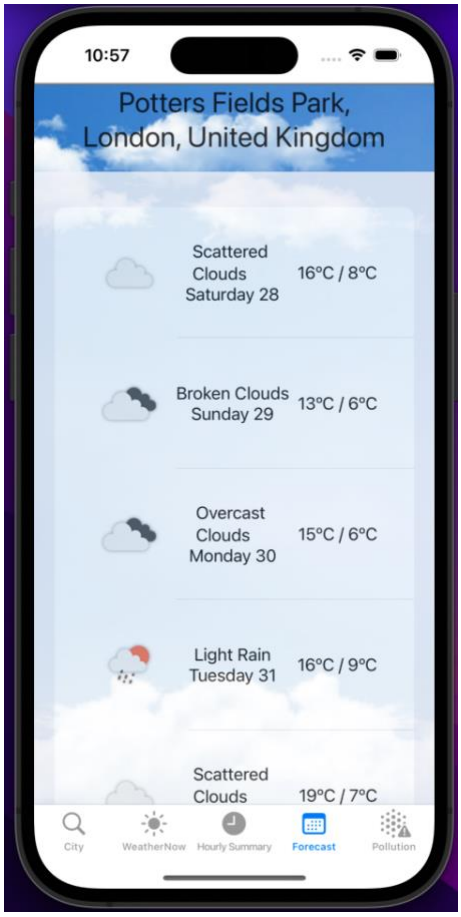


Figure 4

Forecast Tab

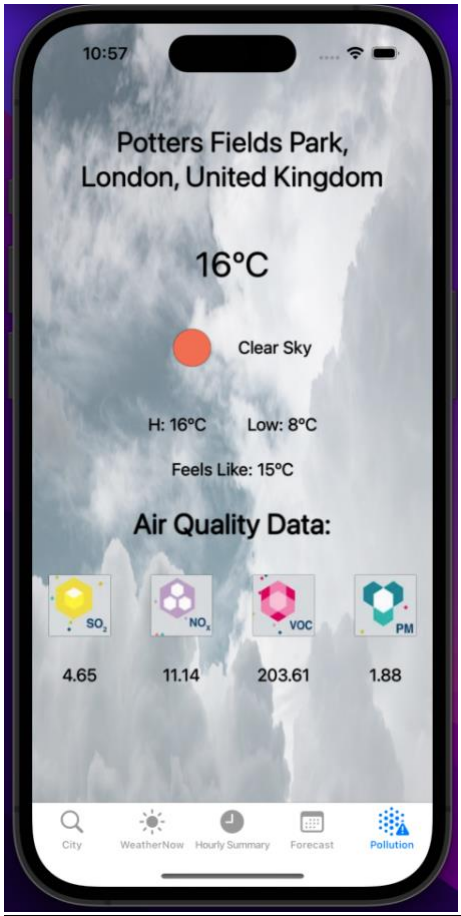


Figure 5

Pollution Tab

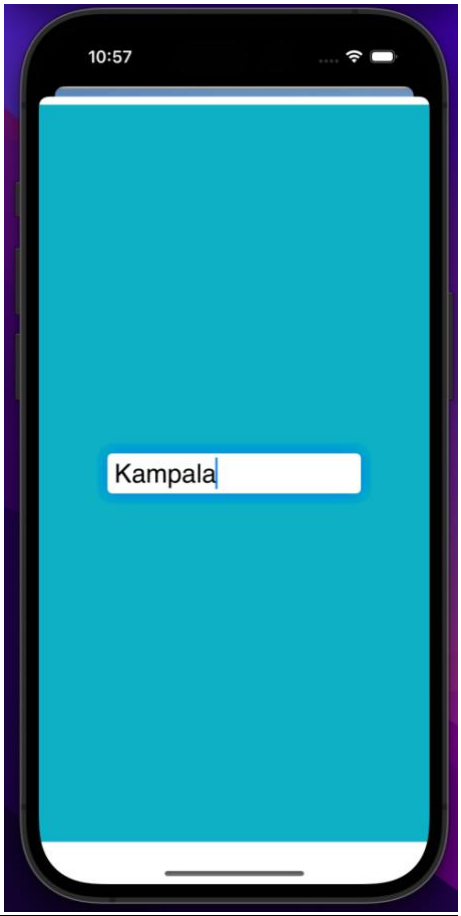
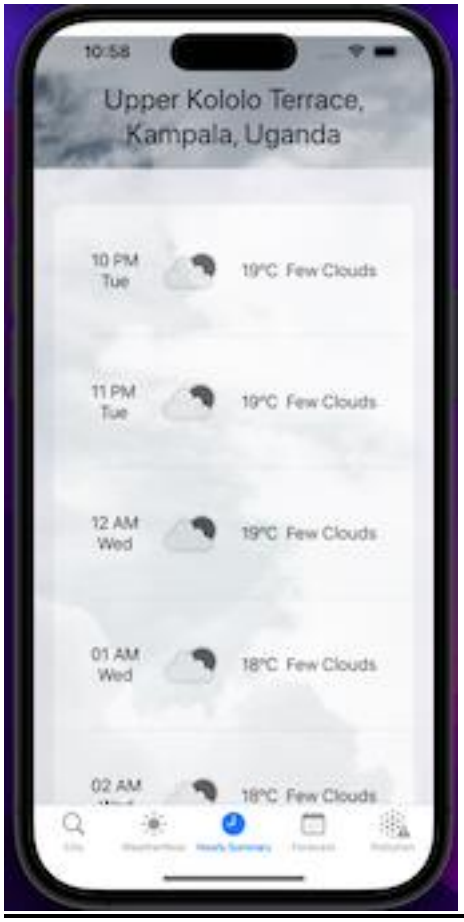
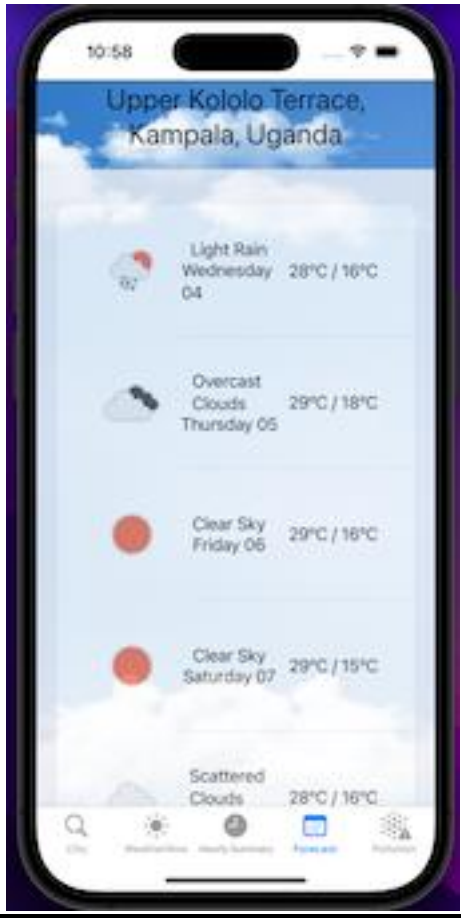


Figure 6

Sheet View

Updated views after new location - Kampala (Uganda) for example


10:58
Upper Kololo Terrace,
Kampala, Uganda
10 PM
Tue
19°C Few Clouds
11 PM
Tue
19°C Few Clouds
12 AM
Wed
19°C Few Clouds
01 AM
Wed
18°C Few Clouds
02 AM
18°C Few Clouds


10:58
Upper Kololo Terrace,
Kampala, Uganda
Light Rain
Wednesday
04
28°C / 16°C
Overcast
Clouds
Thursday 05
29°C / 18°C
Clear Sky
Friday 06
29°C / 16°C
Clear Sky
Saturday 07
29°C / 15°C
Scattered
Clouds
28°C / 16°C


10:59
Upper Kololo Terrace,
Kampala, Uganda
19°C
Few Clouds
H: 28°C
Low: 16°C
Feels Like: 20°C
Air Quality Data:
4.65
11.14
203.61
1.88

Marking Scheme (Draft)

- When app is launched, initial view rendered with data from a file:
    a. json weather data successfully loaded from a file. (5 marks)
    b. data correctly decoded into a swift a weather struct. (5 marks)
    c. a button to enable change of location (5 marks)
    d. Initial view correctly rendered showing location name, date and time, temperature, humidity, pressure and conditions with an icon (10 marks)

- Tab Navigation to load alternative views with different information (15 marks)
- Weather views updated after an API call (10 marks)
- Pollution view updated after an API call (10 marks)
- Icons rendered correctly from OpenWeather Image call. (10 marks)
- Interface design demonstrating good use of stacks and modifiers (10 marks)
- API(s) construction and management demonstrating completion handlers (8 marks)
- Error management in network connections and safely unwrapping data (6 marks)
- Coding standards and naming conventions (6 marks)

End of document