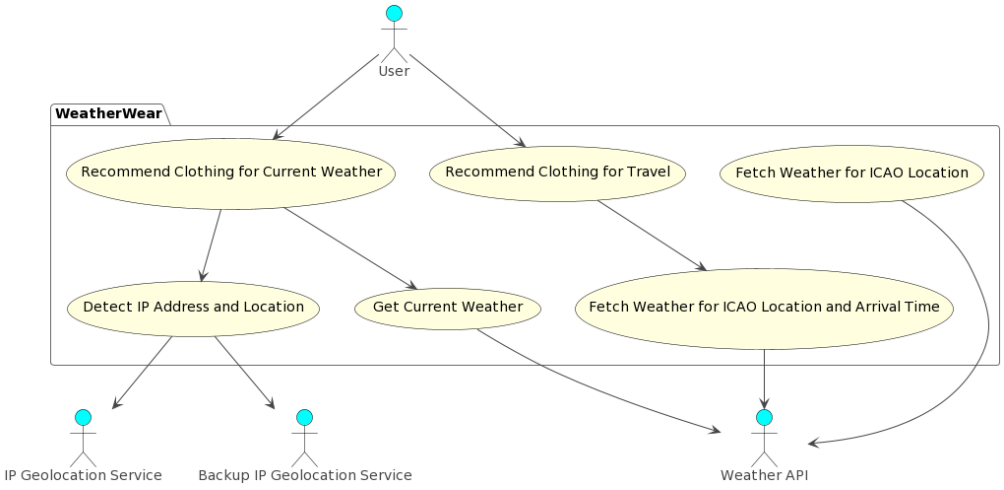
1 Instructions

- This is the first part of a three-part assignment.
- This assignment will be marked out of 35 and accounts for 35% of the study unit mark. It is estimated that it will take you around 16-20 hours to complete this assignment.
- The work is to be done on an *INDIVIDUAL BASIS*, not in groups.
- The assignment is to be submitted via VLE by noon of 20th November 2023 but it is highly recommended that you submit by 10th November, when we commence Part 2 of the course.
- A random sample of students will be interviewed about their assignment to reduce the risk of collusion and/or plagiarism.
- Your submission should be a PDF file documenting your efforts and decisions. Code should be made available via an appropriate repository (e.g. Github).
- **IMPORTANT:** All questions regarding the assignment should be asked on the assignment forum in the VLE. Questions sent by email will be ignored. Before you ask a question, check if it has already been asked on the forum.

2 WeatherWear.com

You form part of a team that is designing and developing *WeatherWear.com*, a website which recommends clothing items based on the weather in your current location or destination. Before creating a fancy website, your Chief Technical Officer (CTO) has asked you to develop the backend to ensure that the concept works. You are to consider the following feature specifications as depicted in the use-case diagram below.



2.1 Feature 1: Recommend clothing for current weather

This feature checks the user's current location by fetching IP GeoLocation data and then making a call to a weather API to check the weather. The initial recommendation algorithm will be simple and will simply provide a string in the following format:

```
It is <cold|warm> so you should wear <warm|light> clothing.
It is <not|currently> raining so you <do|don't> need an umbrella.
```

The weather is deemed to be cold if it is at or below 15 degrees celsius, and is considered warm otherwise. It is considered to be raining if the precipitation is greater than 0.

2.2 Feature 2: Recommend clothing for weather at another destination in the future

This feature asks the user to provide two inputs:

Airport ICAO Code - A 4-character code which uniquely identifies an airport. For example, LMML refers to Malta International Airport whilst EGLL refers to London Heathrow Airport.

Date of arrival - The date when the user expects to arrive at the destination airport in `YYYY-MM-DD` format. The date cannot be more than 10 days in the future.

The system will then check what the weather forecast at the destination airport on the day of arrival and provide clothing recommendations.

2.3 Feature 3: IP Service fail over

Your CTO is worried that if the IP Geolocation service fails, the whole system will collapse. Therefore, you are to implement a fail over service, whereby if the system does not receive a reply from the primary IP Service provider within 3 seconds, it will try a different service.

2.4 Feature 4: Console User Interface

Since this is a proof-of-concept, the CTO requires a simple console application as a user interface. It should display a top-level menu as follows:

```
WeatherWear.com
---------------
1. Recommend clothing for current location
2. Recommend clothing for future location
3. Exit

Enter choice: __
```

Although the user interface is simple, appropriate data validation needs to be carried out on all user input and error messages displayed where required.

3 Third-party APIs

You are free to use any freely available third-party REST APIs to construct your system but the following are being provided as starting points:

Weather API - Weather API is a fully-fledged weather API **and** IP Geolocation service with a free tier subscription. **URL**: `https://rapidapi.com/weatherapi/api/weatherapi-com`

ip-api - This is an IP Geolocation service that can be used as your backup service for Feature 4. **URL:** `https://www.ip-api.com`

4 Task: Develop and Unit-Test the WeatherWear System

Develop and unit-test the proof-of-concept `WeatherWear` app as described above. Ensure that you:

1. Develop all 4 features as described.
2. Validate all direct inputs from users.
3. Validate all indirect inputs from third-party services such as error codes and service unavailability.
4. Develop unit tests to achieve as high a level of coverage as possible.

Your report should include:

1. A high-level design of your system.
2. A discussion on important decisions you made, including any test patterns that you may have used to improve the testability of your system.
3. A discussion of the code coverage achieved, and if you did not achieve 100% coverage, identify reasons as to why this was not pragmatically possible.

5 Marking Scheme

This part of the assignment accounts for 35% of the marks for this study unit. This will be allocated as follows.

- Construction of functioning system - **10 marks**
- Demonstration of testability concepts and test patterns - **10 marks**
- Appropriate unit testing, including use of appropriate test doubles - **10 marks**
- Discussion of coverage - **5 marks**

Please be aware that an elaborate report is not necessary. Prioritise clear communication of the information over the aesthetics of the report. I anticipate that the reports will range between 5-7 pages, contingent on your writing approach and inclusion of diagrams. This is merely a guideline; feel free to adjust the page count as needed without concern.