Homework Week 1

# 2)

## Project

We are building a symptom checker website. If a user is feeling unwell our online symptom checker can help you identify whether you need to seek immediate medical attention

This is to help patients utilise their consultation time with health care professionals more efficiently, as they will have a better understanding of their symptoms and potential diagnoses.

The website will allow users to input the symptoms they are experiencing and additional information about their medical history. This user will then receive a list of differential diagnoses and potential tests they can take to confirm the diagnoses.

## Functionality / Problem Solving

Our online symptom checker will ask users to input details about the symptoms. Using an API comprised of computerised algorithms, the self-diagnosis tool will then give a range of conditions that might fit the problems a user is experiencing.

**Functionality**

* User inputs symptoms into a search field
* A server sends symptoms to an API
* API that will return a list of differential diagnoses and potential tests to the user can see
* Users can make an informed decision based on results
* Users can opt to log in or register and save their search results to an account
* When a user account is created, their personal information will be saved to the database
* If the user saves the result, the result will be added to the database

**Problem Solving**

* Our API is different to what is currently available, as it will allow users to search with their test results in addition to symptoms
* Giving users an understanding of their symptoms and potential diagnoses before they go to a health care professional
* Reducing the load on health care professionals as users with less urgent symptoms will be less likely to go to the health care services

## Key Features

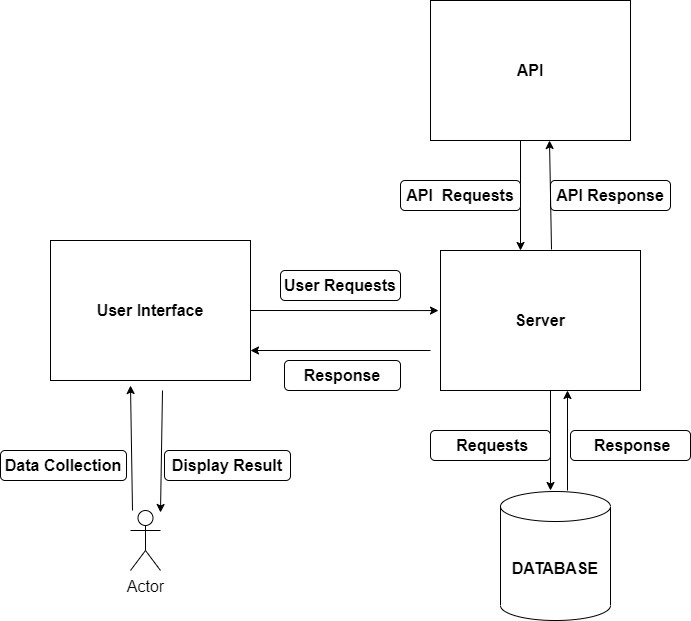
**Performance**

* Users should be able to type words into the search field and be presented with possible diagnostics based on those words.
* In case of misspelt words, unknown characters, or numbers, it should display clear messages so the user understands the mistake and can correct it.
* All links must take the user to the correct page for example the link attached to the logo should take the user back to the homepage no matter on which page she/he is at that moment.
* If a page cannot display, there should be a 404 not Found page displayed with a clear message for the user.
* If there are any coding errors, the user should see clear messages and not the errors displayed.
* Logout button should take the user back to the homepage and not allow user to go back to the account page without logging in first.
* Optimised for mobile

**UI/UX**

* Create obvious, logical navigation with clear hierarchy.
* Use consistent layouts and visual cues for functionality across the site.
* Design and flow need to be simple and clear, without clutching information, and build with complementary colours.
* Typography used must follow the rules. Text must be easily legible. Headings must follow the correct order flow on the page.
* Footer should display minimum information and links that are normally found on a footer.
* Contact page should display concise information on how we can be contacted. These details must be correct.
* About us should display correct information about each of us.
* Account page should display basic information about the user. If the user added any search results to her/his account, we should display this on her/his page correctly.
* Login button should take user to the Login page. The form should have 2 input fields and a submit button. The form should display correct messages if there are any errors. Upon successful login, the user should be redirected to the correct account.
* Logout button should take the user back to the homepage and not allow user to go back to the account page without logging first.

## Sample Architectural Diagram



## Team Approach

**Distribution of Workload**

* Product Backlog Refinement: We will use Jira to create a backlog of all tasks.
* We will have a sprint planning meeting on Google Meet to assign and prioritise what tasks in the backlog can be completed by the deadline
* We will have daily scrum meetings to establish everyone’s progress and issues

**Managing Code**

* Github - manage version control and collaborate
* Using effective class, method, and variable names
* Prettier (VSCode extension) - helps to make code layout should be readable and consistent
* Using DRY (Don’t Repeat Yourself) to avoid elongated code
* Following Clean Coding Best Practices for all languages (ex. Airbnb React/JSX Style Guide)
* Organising folders using a source tree directory structure
* Use correct exception handling to display a correct error message to the user and avoid exiting the programme prematurely

**Testing System**

* Create Unit tests
* User Experience Testing - Survey
* Browser Compatibility (End-to-end Test)
* Mobile Compatibility (OS, Size, Display Type)
* Website Tester Tool