Stage 1: Propose a dynamic web application case study that requires the design and development of a simple online comparison system that allows customers to view the same products from different companies. Include justification of your case study and the need for a dynamic web application. (Maximum word count: 500, 10 points)

# 1. Case study

Online shopping nirvana: type the name of the product that you wish to buy and see the supplier that offers the lowest price. This is what comparison websites are all about. Some years ago, the buyer was spending a significant amount of time searching the web for his product on different e-commerce websites, aiming to find the best possible deal.

From a user’s perspective, comparison websites can look straightforward, but they are not after merging all the challenges and technology. One of the biggest challenges is how to collect data from suppliers. Diving a bit deeper into this matter I discovered three ways of doing this:

1. Feeds from vendors: websites for price comparison enter into an agreement with stores or businesses to collect data directly from the web feeds of a retailer. Retailers will provide lists of goods and prices of their own.

2. Product feeds from third-party API: Some services provide e-commerce data via an API request when integrated with shopping carts. These businesses regularly crawl the most popular retailer websites and provide this information for price comparison websites in a database for easy access.

3. Web Scraping: to obtain pricing data, you can either build your own in-house web scraping system or employ a data-as-a-service (DaaS) provider who will provide the necessary data as per your requirement. A web scraping system will use crawling bots to extract data from websites for further processing on a regular basis.

Another big challenge is the variation in the pricing data structures of various retailers. For example, the biggest challenge in API integration is that when creating multiple integrations adhering to the requirements of each shopping platform, it is a difficult and costly process. Shopping platform leaders such as Shopify, Magento and WooCommerce can be integrated through API2Cart, which provides a single unified API to connect all shopping carts simultaneously.

For any of these three cases a dynamic web application will be needed because in a “normal” web application or a simple website everything is just a client side with some HTML and CSS or maybe JavaScript or any other extension of JavaScript (with animation or interaction purposes). There is no server side.

In a dynamic web application (in our case the comparison website), all the product details and pricing data received from the suppliers will be stored in a database and the website pages will need to be generated in real time, as per the request. A response will trigger from the server side and will reach the client side. Depending upon the response the client side code will take action.

For the purpose of this assignment I will assume that I have an agreement with different retailers(10 as per requested in the assignment scenario) and the products and pricing data will be provided directly by these retailers. My chosen niche will be the smartphone industry because in the last decade this niche literally exploded, and there is plenty of inspiration and examples out there .

Word count: 499.

Stage 2: Plan and design between five to eight web pages for a given website that includes comparison process components. Provide recommendations with regards to a markup language, database, scripting (both client side and server side), multimedia components, and layout design. (Maximum word count: 1000, 10 points)

# 2. Web Planning and Design

The key to develop and create a good, on-time website is a website project plan. A web developer may escape all the common difficulties that make a project too complicated with the right organisation and plan in place. Linking this to my project, this is my path:

The purpose of the website - I concluded from the assignment scenario that two key goals would have to be accomplished by building this website.

//2

In order to make an informed decision in regard to a satisfactory selection of programming languages and technologies I had to conduct extensive research.

The technologies I decided to use are **MySQL** along with the **PHP** scripting language for implementing the database aspect of the application. When talking about backend I did chose to use Facebook’s **React** framework and for implementing the structure and styling of the website I used **HTML** and **CSS**.

Below I will try to explain what made me use those over other similar technologies and how their usage allowed me to write a well optimized and fully working comparison website.

According to Russel, **MySQL** seems to be a very well-known database implementation which also has the benefit of being open source. It seems to be fast and stable and for the purpose of this assignment this database technology might prove as an excellent choice.

In order for the **MySQL** database to communicate with the website, I used **PHP** which, according to **PHP**’s official website, seems to be a popular open-source language which is optimized for usage in web development.

The reason I used this scripting language is because it seems to be very easy to implement and has a wide set of features which might prove very useful in the case that the website needs to have the ability to scale in the future.

For implementing the dynamic aspect of the website which is needed for showing the fetched data from the database in a flexible manner; I chose to use the **React** framework which, according to **React’s** official website, looks to be a fast and flexible library based on the **JavaScript** programming language. It seems to be very well suited for creating web interfaces. According to a **StackOverflow** survey, ranked the 3rd place on the list of the most used web frameworks. The reason I used **React** for creating this website is because of its’ intuitive approach, ease of use, speed and powerful features like dynamic rendering based on state, ability to pass data among multiple pages in an easy manner using context **hooks** and easy integration with different other libraries and languages which in turn might make websites made in **React** very scalable.

In order to implement the structure of the website, I decided to use **HTML** which seems to be a markup language used to define the basic structure of a website and enable elements like images and text to be shown on the screen of the user. **CSS** is a web technology used to style the resulting elements from **HTML** which seems to improve readability, allows the developer to show effects like animations and provides the ability to define the positioning of the elements.

Those technologies provide the means of creating a fully working and responsive website which can fully serve the needs of the user.

Stage 3: Recommend improvements that would enhance the browser compatibility of the website as well as web accessibility to meet the diverse audience. (Maximum word count: 1000, 20 points) L.O.2.

# 3. Cross browser compatibility and accessibility

Stage 4: Implement between five to eight web pages for a given case study using dynamic web development language (ASP, PHP, or JSP). Provide screenshots of the document and code in the appendix Provide a table that shows the testing logs of each page. (Maximum word count: 1500, 40 points) L.O.3 and L.O.4

# 4. Implementation

//4

For the purpose of this assignment, I created multiple web pages which can be directly accessed using a **<nav></nav>** element inside a **React** stateful component. Because of the way **React** works, the website is served as a singular page and based on the URL path, the server renders the matching component which is defined on that respective path. A **React** library is used to route to the respective component on the path and a render call is triggered each time another component is shown. This in turn seems to give the impression of fluid behaviour.

When talking about the webpages that the website will contain, I had to properly think about what is expected and what could improve the life of the user accessing it.

Because of the way **React** works, I decided to save every webpage as a separate **React** component in order to facilitate order and code reusability.

After careful consideration, I decided I needed a page which would act as a home page that contains general information about the service and a call to action (CTA) button which redirects the user to the phone comparison page. This page is containing an input field and 3 radio buttons which enables sorting functionality. The way the phones are shown on display is via a process in which the user inputs the name of a phone and selects from the radio buttons to sort by the lowest price, highest price or from the best rating to the worst. The user can then click on the **‘Add to basket’** button which will introduce the selected phone to the user’s personal basket.

In the navbar there exists a basket icon which, when clicked, sends the user to a checkout page. This page is comprised of a form where the user can input data like their name, email, telephone and means to pay and a list with all the phones that were put in the basket. In the list relevant data about the phone like phone brand, model, rating and price is shown. When the user is satisfied a **‘Buy Now’** button can be clicked which will send the user’s data to an **PHP** file which then will process the data and store it in the local **MySQL** database.

The website also contains an **About** page which tells more about the website, its’ mission, general legal information and a **CTA** button which redirects to the comparison page.

A **Contact** page also exists which is the host of relevant data about the business, like opening times, phone numbers and a form which takes the user’s name, email address and a message which is then supposed to be handled by an **SMTP** server and send an internal email for the customer support department to read. Unfortunately, this feature is not implemented as there is not a server that can handle those requests and this webpage is used only for presentation purposes.

The **Footer** component is also very essential to the webpage. This component is split into three sections: The first section contains relevant links for the user like the **‘Compare Phones’** section which redirects the user to the **Searchbar**, our terms and conditions, the privacy policy of our website and links to the ‘**Contact**’and ‘**About**’ pages. The second section encompasses links to the social media accounts of the website in the form of animated icons which change colour. The third section is the container of a ‘**Subscribe**’ form which enables the user to input their email address and receive notifications about any changes happening to the site. Due to not being specifically requested, the form has the purpose of serving as a decoration.

In order for the website to be functional, backend components had to created and configured in such a way that optimal communication can be achieved without sacrificing the speed of the website or the user experience. The database is encompassing multiple tables that hold data about the phones, the suppliers and the customers that choose to shop with our website. In order for the phones to be shown on the screen, I decided to send all the phones in the database from all suppliers by fetching and storing the data in objects which are then stored in an array. This array is then acting as the central source from which the data will be retrieved when the user searches for a phone. This might not be the most efficient way of doing this, but I decided to create this simple mechanism because of the fact that this might not put as much stress on the database like having users run multiple search queries for every search would. Also, the data has to be retrieved from the database anyway at the moment the website loads so the user can see all the phones in the list.

The **Searchbar** component is the component that contains this object array. It also encompasses code which has the role of sorting the phones, displaying the data and images of the phones and suppliers. This is done via the usage of switch statements that read the different values of the object and based on those values different code is returned from the functions in the form of **HTML** and **JSX** code. According to the React’s website, **JSX** seems to be an extension to **JavaScript’s** language and might be used in order to create the webpage’s structure and display it on the screen. It seems to keep track of different inherent **React** behaviours like state change which seems to be extremely important when dealing with any application that has to display in a dynamic manner.Without **JSX**, I would not have been able to allow the website to make show the list of phones in such an efficient and quick way.

Stage 5: Evaluate the implementation and how you would rate your own application against current accessibility guidelines. Provide logs and testing records to justify any claims. (Maximum word count: 1000, 20 points) L.O.5.

# 5. Evaluation

References:

[**https://books.google.co.uk/books?hl=en&lr=&id=lBHE8xdSGwQC&oi=fnd&pg=PR7&dq=mysql+explained&ots=9FSnc4zcBA&sig=ZJrQQ0xRCXhjh7Mjhv1ChOMN568&redir\_esc=y#v=onepage&q=mysql%20explained&f=false**](https://books.google.co.uk/books?hl=en&lr=&id=lBHE8xdSGwQC&oi=fnd&pg=PR7&dq=mysql+explained&ots=9FSnc4zcBA&sig=ZJrQQ0xRCXhjh7Mjhv1ChOMN568&redir_esc=y#v=onepage&q=mysql%20explained&f=false)

[**https://www.php.net/manual/en/intro-whatis.php**](https://www.php.net/manual/en/intro-whatis.php)

[**https://reactjs.org/tutorial/tutorial.html**](https://reactjs.org/tutorial/tutorial.html)

[**https://insights.stackoverflow.com/survey/2018/#most-loved-dreaded-and-wanted**](https://insights.stackoverflow.com/survey/2018/#most-loved-dreaded-and-wanted)

[**https://reactjs.org/docs/hooks-intro.html**](https://reactjs.org/docs/hooks-intro.html)

[**https://www.w3.org/standards/webdesign/htmlcss.html**](https://www.w3.org/standards/webdesign/htmlcss.html)

[**https://developer.mozilla.org/en-US/docs/Learn/Getting\_started\_with\_the\_web/CSS\_basics**](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics)

[**https://reactrouter.com**](https://reactrouter.com)

[**https://reactjs.org/docs/components-and-props.html**](https://reactjs.org/docs/components-and-props.html)

[**https://reactjs.org/docs/introducing-jsx.html**](https://reactjs.org/docs/introducing-jsx.html)