**Alternative Assessment - 6CS028**

**Kim Mir – 1824409**

Contents

[Web Frameworks 2](#_Toc71289909)

[1a) 2](#_Toc71289910)

[1b) 2](#_Toc71289911)

[Mobile Websites 4](#_Toc71289912)

[2a) 4](#_Toc71289913)

[2b) 5](#_Toc71289914)

[Rich Internet Applications 6](#_Toc71289915)

[3a) 6](#_Toc71289916)

[3b) 6](#_Toc71289917)

[3c) 7](#_Toc71289918)

[HTML5 and the Semantic Web 7](#_Toc71289919)

[4a) 8](#_Toc71289920)

[4b) 8](#_Toc71289921)

[4c) 8](#_Toc71289922)

[References 9](#_Toc71289923)

# Web Frameworks

## 1a)

Web frameworks provide a foundation to build web applications upon. This foundation is there to provide developers a means to develop web applications in a more simplified manner. For this reason, web frameworks are designed with certain concepts in mind. For example, easier development, easier maintenance, security, scalability, built-in features, libraries, form validation, and debugging.

Easier development and maintenance go hand in hand, often web frameworks are designed for particular use cases and are organized in a modular format to improve the development experience. Moreover, framework libraries are hugely beneficial. They often include the addition of commonly needed features that will speed up development as the developer no longer needs to develop such features. Examples include validation, security, routing, URLs etc. (Ivanova & Georgiev, 2019) discusses different approaches using different technologies and how they are suitable for different use cases. It is the perfect example as to why web frameworks help facilitate the development of web applications. Flexibility of usage is valuable. Furthermore, design patterns such as MVC are mentioned. These patterns help facilitate good web development as they provide a prebuilt scaffold like structure that developers can use to build web applications.

To visually explain the benefits of web frameworks, (María del Pilar, et al., 2015) includes the best practices for web development as well as comparison tables to see which frameworks include such features. This includes anything from testing, security, HTML5 support, JavaScript framework support, validation, debugging, documentation, etc. Without these features, web development would not be as easy or as popular as it is today. They are essential to development.

## 1b)

The MVC acronym refers to Model, View, and Controller. MVC is a design pattern that separates application logic from views. This separation of concerns means that there is clear distinction between client side and server side. In practice, the server side can contain most of the scripting and the client side can have minimal scripting while remaining as an interface between the user and the controller.

Using the CodeIgniter web framework as an example, the individual web MVC components are defined as (Codeigniter, 2019):

* The **Model** represents data structures, typically database related
* The **View** is the information presented to the user
* The **Controller** serves as an intermediary between the Model, the View, and any other required resources

The model aspect of web MVC, usually written in a mixture of SQL and the framework language, typically refers to the addition, removal, retrieval, or update of database information. In most cases the usage of models occurs after a user has made a request to the controller that requires some information to be included in a particular view. In these circumstances, this occurs before the view has been returned.

Web MVC views describe the actual pages that users will interact with. This interface will determine what controller functions the user can use. View pages, mostly consisting of HTML, CSS, and JavaScript, can be made into anything the developer wants the user to see. For this reason, they are very malleable and accommodating for user requirements.

Controllers, written in the framework’s language, are the middlemen between the view and the model. Web MVC controllers are designed to handle user requests. User requests can include anything from basic navigation, login/registration, forms, buttons, etc. Controllers also allow the usage of logic before a view is returned; this includes the usage of models. An example of this would be data validation from a form submission, or the retrieval of a record from the database. Separated from the view, this means the application logic is hidden from the user.

Unlike traditional MVC, web MVC views and models do not interact with each other. Web MVC views only interact with the controller, whereas traditional MVC views can update themselves by using the model directly.

Here is an example flowchart of Web MVC in practice. As you can see, the controller is the entry point to the application.

Diagram

Description automatically generated

In modern times, companies hire multiple developers to work on the same web application. A lot of this would not be possible without the implementation of web MVC. The reasons are as follows (Singh, et al., 2018); developer specialization, and parallel development. Developer specialization means that different developers that specialize in different development fields can work according to their specialty. For example, a developer that is not UI oriented can distance themselves from that area and only work on business logic areas, e.g., controllers and models. Secondly, parallel development means that multiple developers can work on the same project without clashing. This is due to the separation of concerns, otherwise known as the modularity of web MVC. Tasks can be performed simultaneously.

# Mobile Websites

## 2a)

Modern web enabled smartphones hold their position in the market due to a variety of unique features. For example, GPRS, touch screen, cameras, fingerprint sensors, microphones, size, and mobility.

GPRS (General Packet Radio Service) permits mobile communication. Without GPRS smartphone communication would be non-existent. This goes hand in hand with size (able to carry easily) and mobility. Without this, smartphones would become more akin to desktops and would be significantly less attractive due to the decrease of convenience. For web developers, GPRS allows users to use your product anywhere with a connection. This means there is more interaction with your user base. More interaction means it is more likely that the users will spend more money for your product.

Cameras and microphones can be used by web developers in a multitude of ways. It can be designed to handle anything from interaction, communication to security purposes. While it is true that some desktops have cameras and microphones, the way which it is handled for smartphones is usually very different. The handheld usage that smartphones offer allows for versatile usage. This means cameras and microphones can be used almost anywhere, unlike their counterpart, the desktop. For web developers, this means hardware can be used in a three-dimensional format rather than fixed position format.

Moreover, since hardware such as cameras, microphones, and to a lesser extent fingerprint sensors (due to their newer implementation) are available; developers can use these devices for security purposes. For example, facial recognition, voice recognition, and fingerprint locks. Lastly, linking back to connectivity, cameras and microphones permit more interaction between users as well as user and business, e.g., users with disabilities may find it easier to use a microphone. The benefits of hardware like this are extensive and will only improve as time goes on.

Touch screen inputs are highly beneficial due to the highly adaptable methods of usage. To name some examples, they can be used for, selecting items, normal typing, drawing and gestures. Desktops on the other hand are limited to mouse and keyboard. Different input types allow for different interaction. Touch screen means that physical keyboards do not take up screen space. For developers, this means your web application can show more on the same sized phone.

Using voice control, gesture, and touch screen, the authors of (Khadilkar & Wagdarikar, 2015) were able to develop an android phone based controlled smart wheelchair. It highlights the benefits of modern web enabled smartphones as without them, this would not be possible.

On the other hand, smartphones do have trade-offs for all the benefits that they offer; battery life, technology obsoletion, and software obsoletion are a few examples.

Smartphones, unlike desktops, have limited battery life. The connectivity and mobility of smartphones is dependent on the battery life. Smartphone devices are constrained as they rely on limited battery supply that has not been increased at the same pace to support the power demands (Zaman & Almusalli, 2017). Web developers need to take this into consideration. When possible, they must try to reduce the amount of power needed to run their application. This usually means that developers resort to using lower quality application features so that battery life may be extended.

Technology and software obsoletion often occur together. When a user buys a new smartphone, over time less and less applications are made available to them. This is due to the hardware usually being outdated, which then in turn leads to software not being updated. For the applications that do remain available, web developers must build their application to suit both older and newer phones. This is added work for the developers and is another limitation to smart phones. Desktops for comparison use the latest operating systems and will receive updates regardless for the foreseeable future.

## 2b)

No mobile support entails that users have no access to mobile versions of your product. The pros of no support are that the developers have less to make and to maintain. The cons of no support are that your mobile users are cut off or will have a limited experience with your product. This leads to less traffic followed by less business.

Dedicated mobile websites refer to having two sites per page, one for desktop and one for mobile. Usually, the mobile site URL starts with “m.”. The pros of dedicated mobile websites are that desktop and mobile versions are kept separate. This means that content can be tailored on a device basis. If implemented correctly, dedicated mobile websites are faster as they are optimized and streamlined for the device. If a website already has a desktop site, creating a separate mobile site will be relatively quick. On the other hand, multiple URLs are required with dedicated mobile sites, this can increase costs. Next, mobile devices that access the desktop URL need to be redirected to the mobile URL and vice versa for desktop devices. This redirection adds to load time. Lastly, two sets of content for each page means that there is more management and maintenance needed. This can become difficult to manage as what may be suitable for one, may not be suitable for the other (Jogoo, et al., 2019).

Several mobile websites refers to multiple versions of the same website offered for different resolutions, interfaces, browsers, etc. It is akin to dedicated mobile websites. Multiple versions of the same site tailored for different devices/screens/technologies is a positive. If done correctly, it should mean the user has a better user experience. On the other hand, only large companies can do something like this. Large amounts of manpower, money, and time is required to develop and then maintain said websites.

Responsive websites are websites where the design can correctly respond to display content that matches the device display. In practice this means that the same websites can be displayed on desktop and mobile without changes to the design. Responsive websites unlike dedicated mobile websites only need to have one site developed for all devices. This lowers the cost of maintenance. Responsive sites have the same content across multiple platforms. This is a pro as it provides consistency for the user. Lastly only a single URL is required as it is a single website. No redirection is needed unlike dedicated mobile websites. The drawbacks of this type of design are worse performance as the site will usually take more storage, usability problems due to clashes in how desktop and mobile users want to interact with the website, and the content of the site is not optimized for either device.

Progressive web applications are the closest to native applications except are still in a browser (Behl & Raj, 2018). They can be installed on the mobile home screen, be accessed offline, and use push notifications. The benefits to these are that they behave like native applications while requiring no storage. Although it behaves like a native application, it does not have the same visibility as a native application as no app store is involved. Moreover, they have less functionalities than native applications meaning the user experience will be worse than a native application. Lastly technologies such as Bluetooth and NFC cannot be applied to progressive web applications. This means progressive web applications are limited in what they can do.

Native applications offer the most functionality possible to the user. They are installed on to the device meaning they work the fastest (benefit) and take storage space (drawback). Native applications offer the best user experience (benefit) as it works exactly like a default application (Jobe, 2013). Lastly, native applications can be used offline (benefit). On the flip side, native applications do not offer platform flexibility (drawback). If developing a native application, an application for both android and iOS must be developed separately (drawback). This adds to development time, maintenance, cost, etc.

# Rich Internet Applications

## 3a)

AJAX is described as Asynchronous JavaScript and XML. AJAX improves website responsiveness by exchanging small volumes of data with the server without needing to reload the page. AJAX utilizes JavaScript by using it as the logic to start the process of sending/receiving data. XML/JSON is used as the medium of sending data to and from the server. HTML/CSS in the context of AJAX usually describes the section of the webpage that will execute or be influenced by AJAX. For example, a search bar may execute the JavaScript, and the result of the influence could appear as a table in HTML/CSS form.

## 3b)

Autocomplete searches can be implemented via the use of AJAX. They are beneficial as they do not require a page reload to get results which consequently means less stress on the server, it improves usability, it helps users with hard to spell searches, and it can make searches faster (Yao & Sen, 2013).

Information pop-up boxes are another option. These pop-up boxes can be beneficial as they can be implemented in a such a way that the content only loads when it is needed. Furthermore, the content that is in the pop-up boxes will be more relevant to the user.

Form filling assistance is another AJAX use case. If implemented it can speed up form filling, show context specific messages, and improves usability as it works without needed to reload the page (Zhang, et al., 2008). It is an overall more seamless experience.

AJAX can also be used to update database information. It is beneficial as it permits real time communication with the database, updates can be sent without reloading, and it provides an overall quicker and more usable experience.

## 3c)

AJAX is not perfect and does come with some problems of its own. For example, bookmarking, navigation, referencing, response time, client support, accessibility, and performance. When using AJAX to update page content the URL does not change, therefore, if booking the page and returning to the page later, then the updated content is not saved. A workaround would be URL hashing, save buttons linked to accounts, or provide a full URL to users. For navigation, when pressing the back button, the same issue can occur. Using the HTML History API (MDN contributors, 2021) can solve this problem by manipulating the browser history and changing URLs.

The speed at which AJAX requests are processed can be an issue. Traditionally, there is some sort of animated logo to inform the user that something is happening. This is not the case for AJAX. A solution for this would be implementing some form of message, animated picture, progress bar, etc.

Not all browsers support AJAX. This is a client support issue. Without support, certain features may not work at all for some users. The solution to this would be to check support tables before implementing features as well as when possible, keeping Ajax features to a minimum. Moreover, device/client detection can be used to adapt your website to the circumstances. This includes using browser specific workarounds.

Accessibility can be an issue when using AJAX as not all AJAX features are available to disabled users. It is illegal to discriminate against people with disabilities when providing a service, including websites. An example would be that users with screen readers find it difficult to know when a page has updated, making it inaccessible for them. To combat this, the WAI-ARIA website (Cooper, 2020) should be used as it helps explain how to make rich internet applications accessible to people with disabilities. Informing users of dynamic updates, highlighting areas of change, not changing the page focus, offering the option to disable automatic updates, and ensuring the site works if JavaScript is not available are solutions to help make pages more accessible (Sandhya & Devi, 2011).

Lastly, AJAX was developed with technology not intended for rich internet application usage. AJAX messages are bloated and communication to the server is mono-directional. Mono-directional communication means that unnecessary polling can occur. Both points lead to more than required data usage as well as more pressure on the server. Reducing the number of AJAX requests, reducing the amount of data transmitted, and optimizing code are possible solutions to help improve performance.

# HTML5 and the Semantic Web

## 4a)

Semantic elements clearly describe their meaning to both the browser and the developer. They help structure the code created into a more readable and easier to maintain format (Fulanovic, et al., 2012). They add intuitive meaning to the elements by adding context. Adding context to elements allows search engines such as Google and Bing to understand the roles and importance of the different parts of the page. Additionally, semantic elements help with accessibility as devices such as screen readers are better able to understand the webpage.

## 4b)

The goal of the semantic web is to make internet data machine readable (Berners-Lee, et al., 2001). Due to the rapid development and expansion of the web, it expanded as a medium of documents for people rather than of information that can be manipulated automatically. The argument against the web today is that information is not connected. The semantic web is a web of linked data that acts like a large database (W3C, 2015). This database allows computers to do more useful work.

There is a growing need for data integration via the development of the semantic web. The IEEE journal is a driver for the semantic web (Shadbolt, et al., 2006). Scientific research requires the integration of diverse scientific backgrounds. Without the semantic web, these scientific communities are more isolated. Collaboration using the semantic web will integrate these components and make for better research.

Healthcare is an example of where the semantic web can be useful. Large quantities of online resources lead to internet users seeking health related information. Incorporating various medical concepts and vocabularies, and mapping each concept to semantic type, the semantic web can infer meaning of search queries (Panahiazar, et al., 2014). Without the semantic web, web users can be left with results that are not relevant to the type of result they wanted.

## 4c)

WebSockets offer bidirectional full duplex communication between the server and the client. On the other hand, AJAX is monodirectional and can only send requests to the server from the client using POST/GET methods. AJAX is asynchronous, meaning that the user can continue with what they are doing while the request is waiting for a response (Garrett, 2007). WebSockets use HTTP for handshaking and will keep TCP for sending messages between client and server, whereas AJAX only uses the HTTP protocol. Since WebSockets are full duplex, the connection between client and server remains alive, meaning that you can use it as much as you like until the client or server decides to close the connection. When AJAX sends a request to the server, the connection will end once the response has returned.

Based on experimental results from (Puranik, et al., 2013), it is suggested that WebSockets are a more suitable alternative to AJAX for real-time web applications due to the higher throughput and lower bandwidth requirements. For example, this could be ecommerce or stock-trading related web applications because of their need for accurate real-time updates.

Some other examples would be chat apps and collaboration apps. WebSockets are more suitable when you need to push data to the server as well as retrieve data. WebSockets for chat apps will allow you to send messages instantly, while receiving messages that other people have sent. This all happens in real time. Moreover, collaboration apps, e.g., google docs. If multiple people are working on the same project and are using the same document at the same time, WebSockets permit changes to occur and be viewed by others and yourself in real-time. AJAX does not have such functionality and would not be suitable in these situations.

# References

Behl, K. & Raj, G., 2018. Architectural Pattern of Progressive Web and Background Synchronization. *{2018 International Conference on Advances in Computing and Communication Engineering (ICACCE),* pp. 366-371.

Berners-Lee, T., Hendler, J. & Lassila, O., 2001. THE SEMANTIC WEB. *Scientific American,* 284(5), pp. 34-43.

Codeigniter, 2019. *Model-View-Controller.* [Online]   
Available at: https://www.codeigniter.com/userguide3/overview/mvc.html  
[Accessed 2 May 2021].

Cooper, M., 2020. *WAI-ARIA.* [Online]   
Available at: https://www.w3.org/WAI/standards-guidelines/aria/  
[Accessed 04 May 2021].

Fulanovic, B., Kucak, D. & Djambic, G., 2012. Structuring documents with new HTML5 semantic elements. *23rd DAAAM International Symposium on Intelligent Manufacturing and Automation 2012,* II(1), pp. 723-726.

Garrett, J. J., 2007. Ajax: A New Approach to Web Applications.

Ivanova, S. & Georgiev, G., 2019. Using modern web frameworks when developing an education application: a practical approach. *{2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO),* pp. 1485-1491.

Jobe, W., 2013. Native Apps vs Mobile Web Apps. *Internation Journal of Interactive Mobile Technologies,* VII(4).

Jogoo, P., Golamgous, I. S. & Chuttur, Y., 2019. From Desktop to Mobile View: A Simplified Approach to Mobile Website Development. *2019 Conference on Next Generation Computing Applications (NextComp),* pp. 1-6.

Khadilkar, S. U. & Wagdarikar, N., 2015. Android phone controlled voice, gesture and touch screen operated smart wheelchair. *2015 International Conference on Pervasive Computing (ICPC),* pp. 1-4.

María del Pilar, S.-Z.et al., 2015. Analyzing best practices on Web development frameworks: The lift approach. *Science Of Computer Programming,* Volume 102, pp. 1-19.

MDN contributors, 2021. *History API.* [Online]   
Available at: https://developer.mozilla.org/en-US/docs/Web/API/History\_API  
[Accessed 4 May 2021].

Panahiazar, M., Taslimitehrani, V., Jadhav, A. & Pathak, J., 2014. Empowering personalized medicine with big data and semantic web technology: Promises, challenges, and use cases. *2014 IEEE International Conference on Big Data (Big Data),* pp. 790-795.

Puranik, D. G., Feiock, D. C. & Hill, J. H., 2013. Real-Time Monitoring using AJAX and WebSockets. *{2013 20th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS),* pp. 110-118.

Sandhya, S. & Devi, K. S., 2011. Accessibility evaluation of websites using screen reader. *2011 7th International Conference on Next Generation Web Services Practices,* pp. 338-341.

Shadbolt, N., Beners-Lee, T. & Hall, W., 2006. The Semantic Web Revisited. *IEEE Intelligent Systems,* 21(3), pp. 96-101.

Singh, A., Chawla, P., Singh, K. & Singh, A. K., 2018. Formulating an MVC Framework for Web Development in JAVA. *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI),* pp. 926-929.

W3C, 2015. *SEMANTIC WEB.* [Online]   
Available at: https://www.w3.org/standards/semanticweb/#w3c\_overview  
[Accessed 7 May 2021].

Yao, Z. & Sen, A., 2013. Implementation of the AutoComplete Feature of the Textbox Based on Ajax and Web Service. *J. Comput.,* Volume VIII, pp. 2197-2203.

Zaman, N. & Almusalli, F. A., 2017. Review: Smartphones power consumption & energy saving techniques. *2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT),* pp. 1-7.

Zhang, E., Liu, C. & Yang, Z., 2008. Use Ajax to Prevent Repeat Form Submission. *2008 International Symposium on Computer Science and Computational Technology,* pp. 261-264.