130089872

Fundamentals of web app development

Contents

[Table of Figures 2](#_Toc95778126)

[Introduction 3](#_Toc95778127)

[History of web app 4](#_Toc95778128)

[Java 4](#_Toc95778129)

[Popular frameworks 4](#_Toc95778130)

[Svelte 4](#_Toc95778131)

[React 5](#_Toc95778132)

[Frontend 5](#_Toc95778133)

[Backend 5](#_Toc95778134)

[Process of a web application 5](#_Toc95778135)

[Http & Https 5](#_Toc95778136)

[Host 6](#_Toc95778137)

[Path 6](#_Toc95778138)

[Query string 6](#_Toc95778139)

[Databases 6](#_Toc95778140)

[SQL 6](#_Toc95778141)

[NoSQL 7](#_Toc95778142)

[Legislation 8](#_Toc95778143)

[Infrastructure and networking 8](#_Toc95778144)

[Cookies 8](#_Toc95778145)

[Bibliography 9](#_Toc95778146)

[Appendix/Appendices 10](#_Toc95778147)

# Table of Figures

[Figure 1 Use Case Diagram example for frontend and backend 5](file:///C:\Users\dales\Dropbox\My%20PC%20(LAPTOP-D0497OJ7)\Desktop\Web%20app%20fundamentals.docx#_Toc95959890)

[Figure 2 Example of hashing search (Peng Xiang, 2010) 8](#_Toc95959891)

# Introduction

Inside this document contains the fundamentals of web app development topics such as the following. The creation and the purpose of web app development, what technologies are used when creating a web application comparing their strengths and limitations and the data protection laws that have to be followed.

## History of web app

Web app development has adapted over the last 20 years.

It all started with web 1.0, which were static web pages that just displayed information and didn’t track or store data. Then in the mid-2000s, web 2.0 dynamic web apps were created. These are normally user-generated content, push notifications and storing account data for signing up to a website. The internet has now adapted to web 3.0. This is very similar to web 2.0 but it is running by an AI that is focusing on edge computing, decentralization, artificial intelligence, machine learning and the blockchain. An example of this is Youtube’s recommendation algorithm system for videos, Netflix’s our picks videos or an online shopping cart such as amazon tracking items you normally buy or have an interest in. This is done by using cookies. There are many methods to gather data from a web server. Some are more in-depth than others and maybe more secure.

## Java

Script tag allows users to load files from our server or elsewhere on the internet. These are known as web servers. Once sent over, Java can use user events to track data. For example, login when clicking the button. This updates the HTML to show loader so Java would update the appearance of the page. The login information is sent to the server for validation to prevent errors and attacks. Java allows this to happen without having to reload the webpage. These requests that send data or send data are known as Ajax, Async, Js and XML requests.

## Popular frameworks

In present-day, front-end developers use frameworks to link different coding languages together. Inside this section, is a list of different frameworks and what their primarily used for. These are listed as difficult of learning. These frameworks function the same as java but provide a better developer experience, which makes it a lot easier when changing code and bug fixing.

## Svelte

The Svelte framework uses java similar to Vue, React and Angular but this framework is designed for building user interface components. Svelte functions differently as it is a compiler when deploying code. It is exported into an imperative format which is more accessible to other API’s. Doing so keeps the application fluent while running as well as being downloaded (Scherer, 2020, p. 121) states “Svelte framework decided to shift the focus from a runtime-based system to a compiler-based system.”

## React

The React framework is more frequently used than Svelte and Vue by companies. React has been used to build many widely used programs today such as Discord, Instagram, Shopify and most importantly Facebook which originally developed the software. This overtime has granted a lot of popularity for communities with similar interests granting the ability for programmers to seek solutions to broken code with ease on sites such as git hub, just answer and other forums. The framework itself is easy to come to grasps and provides simple ways to debug code but there are some drawbacks to using react as (Vasilev, 2021, p. 18) states “One of the main downsides of choosing to react is the freedom that it gives to developers, as opposite to Angular, with already predefined architecture and development patterns, React developers should find their custom solution for different areas of the application”

## Frontend

The frontend communicates to the backend using APIS and data transfers shown in (Figure1). These can be in different languages as an API links the two together. Most commonly rest APIs are used. Typically, this is a verb and a location such as ‘GET accounts’ or ‘POST accounts’ with adding parameters. When using these it can find specific account information meaning you can ‘GET account id’ information or ‘DELETE account id’. With this being explanatory, it is very easy to understand and use when building a web application. But variables are required to complete the request. Without these, the application will not know what exact data is required and where it is stored. An example of this would be ‘$username = $\_POST['username'];’.

Figure 1 Use Case Diagram example for frontend and backend

## Backend

Backend, you can write with any language but the most popular are python, java, PHP and c#. Backend frameworks are also used such as flask for python, express for java, Laravel for PHP and asp.net for c#, backends respond to APIs. They also do batch processes, such as web scraping to talk to a database. They use Object Relation Mapping which are more detailed queries and validation, so the program can add to the database.

## Process of a web application

When viewing the web application’s, the application receives a request in the form of a URL. For example, ‘display musflix URL for add account here’. With this URL it can gather all the information required for the request. This can be carried through the entire application with a connection to the database throughout each page where it is required. This could be used for access levels to require options like removing or adding content.

## Http & Https

The HTTP:// is the protocol this determines if the request is encrypted HTTP is non-encrypted. Whilst HTTPS is encrypted, all this protocol does is decide if the request is encrypted. But this doesn’t change how the request is handled.

## Host

The host is the domain name. This tells the server where to send the response. Each server has its host for example google.com, musflix.com.

## Path

The path tells the server what the client wants and determines what code on the server should be executed. To achieve this response, the server is broken down into different sections, which relate to a different path ‘/usernames’ for example

## Query string

The query string is used to alter the response received. This is broken into different parameters; an example would be when an individual is looking for a movie on Netflix. The input is gathered from a search bar which tells the server what they have searched for so it can respond with the correct information requested if available. But this would require more than the information provided in the URL as it can tell the server what section to look for or to alter the section. For this to progress and action is required such as [POST] [GET] [PUT] [DELETE]. Using these it can determine the correct section is required and then use the query parameters to alter the response of the particular part and section. Normally the response would be an HTML page generated from the client, but the content displayed because of the same path and action was used but the content displayed will be different because of the query parameter from the search. The server acts as a barrier to hidden information such as a database and only uses the paths that are intended for access.

## Databases

When creating a web application, a database is normally required. This can be for many purposes. From a login system that keeps track of users, admins, and guests, to keep track of settings users, require or have requested such as profile pictures and layouts, or for storing content. But the database needs to be secure (Saleem, 2020)“Data breaches are among the most important computer security and privacy problems. It is routine for the attackers to steal millions or even billions of records.” There have been numerous breaches since the start of storing data. From the AOL breach in 2004 to Red Cross in 2022. Over the years precautions have been put in place to slow down and minimalize these breaches such as hashing passwords, adding parameters to prevent code injection with more methods being updated constantly.

## SQL

SQL is in a strict Colum format that must follow the set of rules, or the command inputted by the developer will not work and cause errors. These strict rules allow for more complicated commands when gathering information from the database. Again, this can be for content details.

## NoSQL

Unlike SQL and other relational databases, NoSQL databases aren’t as consistent with storing data efficiently but are designed to scale with ease. This is done by containing different rules and requirements. In NoSQL data is stored differently without relationships and uses key-value stores instead of keeping each part of data separate, meaning if the amount of data stored is causing delays and slows when being retrieved it is possible to split the load of the database between another server lowering congestion times. The user can choose and determine where which part of data is stored where using the database technique known as hashing for each key. For example, Server one could store all data between hash 0 to 25, server two could store between 25 to 50 and so on, but ‘consistent hashing’ should be considered as without this finding the data could be a lengthy task but can be easily reduced. In 2010 a study was conducted by (Peng Xiang, 2010) that explained using node split and consistent hashing techniques will reduce the load on a server the example that was given was in the style of a football game as shown in (Figure2). The downside of NoSQL is that data can only be retrieved by primary key, it can be a very slow process when trying to find data over a set number value but relational database’s do not have this issue so it may be beneficial to decide what type of database is required before creation. Another major concern is the security of a NoSQL database as it does not contain SQL queries. It allows access via injection as (Ron, 2015) states “One of the common reasons for a SQL injection, vulnerability is building the query from string literals which include user input without using proper encoding.”

Diagram, schematic

Description automatically generated

Figure Example of hashing search (Peng Xiang, 2010)

## Legislation

Any app with a login must consider authorization and authentication. This is normally the login that provides a string-based token to the account when inputted, which is a random sequence of charters, which is used on the server-side to validate the account and maintain security from hacks and data breaches, which validates each request. Authorization on the other hand uses defines user roles such as admin’s granting access, authorization to edit information or adding content such as videos.

## Infrastructure and networking

Data centres are where web servers exist. Unless there’s a dedicated server, these servers are controlled with SSH, which is a direct way to control the machine from the command line but can be automated via scripts. This can be also grouped with a private network. Each server in a network is called a node and in private networks, they can communicate quickly and securely. Servers that talk to the outside world have more to worry about they are set up with a reverse proxy such as NGINX, which is a gateway that provides ways to specific ports to the outside world such as exposing the API but not the database.

## Cookies

One of the important parts of a web application is the cookies that work alongside It, cookies are (Mozilla, 2021)“ a small piece of data that a server sends to a user's web browser. The browser may store the cookie and send it back to the same server with later requests.” Some examples of the use of cookies would be keeping users logged in, storing the remaining amount of content that has not been finished from a previous session this is normally used by streaming platforms. The method behind this is using can either be using session cookies or a permanent cookie, as seen on online banking websites session cookies can be more secure for a user’s account some instances users may forget to log out after using the application without session cookies in place someone else could gain access to the account if they can use the same system.

## Conclusion

In conclusion large, scaled web applications should use NoSQL to maintain a smooth quick professional website but should consider the safety of its users and could possibly use session cookies reduce breaches.

# Bibliography

Dabbs, M. (2019, July 29th). *The Fundamentals of Web Application Architecture*. Retrieved Febuary 15th, 2022, from reinvently: https://reinvently.com/blog/fundamentals-web-application-architecture/

Mozilla. (2021, March 8th). *Using HTTP cookies*. Retrieved February 15th, 2022, from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

Peng Xiang, R. H. (2010). *Cache and Consistency in NOSQL.* Information Engineering Center, Ocean University of China. China: Institute of Electrical and Electronics Engineers. doi:10.1109/ICCSIT.2010.5563525

Ron, A. (2015). *No SQL, No Injection? Examining NoSQL Security.* Israel: Cornell University. doi:arXiv preprint arXiv:1506.04082

Saleem, H. (2020). *SoK: Anatomy of Data Breaches.* California: sciendo. doi:DOI 10.2478/popets-2020-0067

Scherer, J. (2020). *Hands On JavaScript High Performance: Build faster web apps using Node.js, Svelte.js and WebAssembly.* Packt Publishing . doi:9781838821098; 1838821090

Vasilev, A. (2021). *Comparison of React components.* Metropolia. Retrieved January 18th, 2022, from https://www.theseus.fi/bitstream/handle/10024/510460/Vasilev\_Andrei.pdf?sequence=2