Devclub Assignment

Vinamr Jain

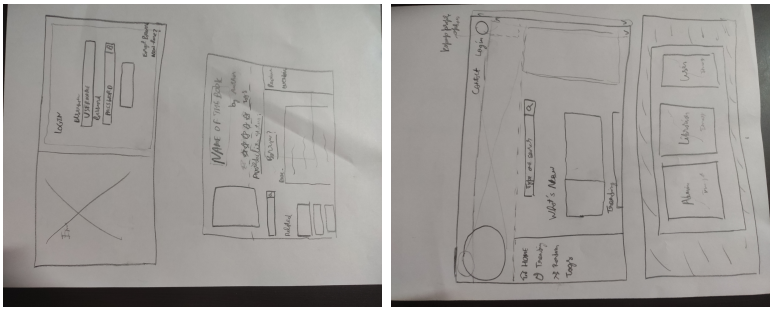
April 2021

Introduction

The following is my Devclub assignment. I decided to do the Design part (ii) Designing the user interface for a library management system
Along with the compulsory Research Assignment (iii)

DESIGN PART (ii)

Honestly speaking I am a complete noob at this I Hadn't the slightest clue of how all this works so I started by Watching some figma tutorials on youtube.(Didn't even know what figma was before this!)
I carefully went through the problem statements time and again.
So I googled how professionals create wireframes to pen down their thought processes and then create the UI interface based on that.
Initially I tried creating some rough wireframes on my desktop (https://wireframe.cc/) But it was pretty frustrating to begin with, no ideas were popping up so I switched to drawing with pencil and paper and I found it more convenient. The following are my rough sketches (I know my drawing is terrible :"C)



In the process I went through a few websites to gain some inspiration but frankly speaking what I ended up doing was completely unrelated(I Just started working on it thinking about all the things I would want in such a website and accordingly Designed and the end product just found it's way somehow, My workflow was completely unplanned)
Here are a few links which I went through but ended up ignoring-

1. https://opac.app/en/
2. https://openlibrary.org/ (this I still found a bit useful)
3. https://www.culturepk.org.uk/libraries/e-library/your-library-app/web-app/
4. I also went through w3schools how to page
   https://www.w3schools.com/howto/default_page3.asp
   This Was extremely helpful, I was able to find what components I can Include in the UI interface.

Finally, I started working on figma, although I was slow at the beginning but I gradually got the hang of it.
I made 5 frames in total. (It was getting really taxing) The Home page, An example of A book page, Type of user page, The login page and the signup page. I'll include the file along with the protypes and a pdf on a github repo. I was suprised that I didn't make many changes (a few insignificant ones but nothing major). The main blueprint remained the same. If I'm being honest I am satisfied with what I created and It is decent although I'll admit there is scope of improvement and some minor adjustments are to be made but still pretty good given that it's my first time.
I plan on bringing this to life but I'm new to html css and javascript. I already watched some tutorials on front-end coding. I'm gonna continue working on it after the assignment deadline.

Research Assignment part (iii)

1 What is HTTP? How does it exchange messages between servers and clients?

The Hyper Text Transfer Protocol (HTTP) is basically a client-server network protocol which is used by the world wide web to deliver to us request messages like HTML pages, images, scripts etc. When you visit a website, your browser makes an HTTP request to a server. Then that server responds with a resource (an image, video, or the HTML of a web page) - which your browser then displays for you.
HTTP messages are how data is exchanged between a server and a client. There are two types of messages: requests sent by the client to trigger an action on the server, and responses, the answer from the server.

2 What are the different HTTP methods that can be specified in a request?

An HTTP method is a verb (like GET, PUT or POST) or a noun (like HEAD or OPTIONS), that describes the action to be performed.
GET indicates that a resource should be fetched
POST means that data is pushed to the server (creating or modifying a resource, or generating a temporary document to send back).
PUT method creates a new resource or replaces a representation of the target resource with the request payload.
HEAD method requests the headers that would be returned if the HEAD request's URL was instead requested with the GET method.
OPTIONS method requests permitted communication options for a given URL or server.
examples-
POST / HTTP/1.1
GET /background.png HTTP/1.0
HEAD /test.html?query=alibaba HTTP/1.1
OPTIONS /index.html HTTP/1.1

3 HTTP HEADERS

HTTP headers let the client and the server pass additional information with an HTTP request or response. A few common headers are-

1. The **Host** request header specifies the host and port number of the server to which the request is being sent.

2. The **User-Agent** request header is a characteristic string that lets servers and network peers identify the application, operating system, vendor, and/or version of the requesting user agent.

3. The **Cookie** HTTP request header contains stored HTTP cookies associated with the server

4. The **Allow** header lists the set of methods supported by a resource.

5. The **From** request header contains an Internet email address for a human user who controls the requesting user agent.

6. The HTTP **Upgrade** header can be used to upgrade an already established client/server connection to a different protocol (over the same transport protocol).

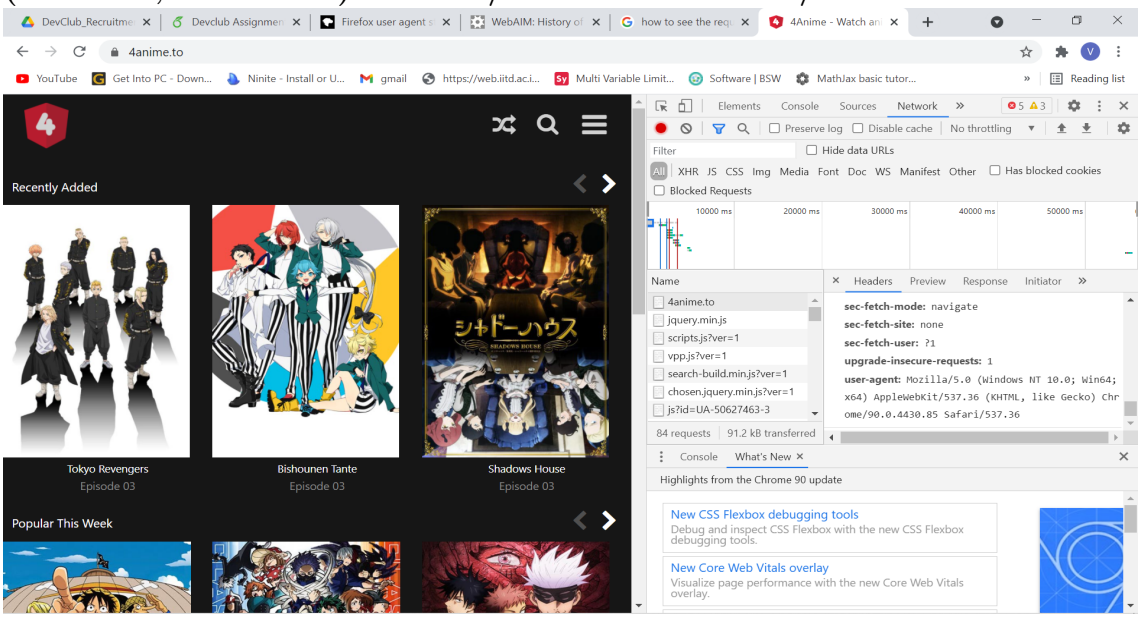
4 History of the User-Agent string

(User-Agent Strings And Gecko versions are basically A token which tells us about the web-browser/renderer(Basically the version of mozilla it pretended to be), And Gecko version indicating the version of the web-browser of sorts.)

So basically the User-Agent String emerged from the war between the Web-browsers. Mosaic was one of the earliest web-browsers which rendered only images and texts.
And then netscape emerged and called itself the 'Mosaic Killer' AKA 'Mozilla'.It was better than Mosaic, and it supported frames which Mosaic did not. And then Microsoft came in along with others each calling themselves an upgraded version of Mozilla.
Internet explorer (Microsoft) killed netscape which then re-emerged as Mozilla/5.0 and built *Gecko* it's rendering engine.

Then came along other Web-browsers/Search engines each with their own developments AND IT GOT REALLY CHAOTIC
Linux users developed KHTML(GECKO WANNABE)
Apple Built safari which used KHTML
GOOGLE Built chrome which Used WebKit,(Chrome is safari wannabe)
And thus Chrome used WebKit, and pretended to be Safari, and WebKit pretended to be KHTML, and KHTML pretended to be Gecko, and all browsers pretended to be Mozilla
the user agent string was a complete mess, and near useless, and everyone pretended to be everyone else, and confusion abounded.

References- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent/Firefox
https://webaim.org/blog/user-agent-string-history/#:~:text=And%20behold%2C%20then%20came%20a,and%20there%20was%20more%20rejoicing.
On Chrome I visited a URL and the following is the User-Agent String value which it returned.
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36



5 What are cookies? How does a website set a cookie and why would it do that?

Cookies are text files with small pieces of data — like a username and password — that are used to identify your computer as you use a computer network. Specific cookies known as HTTP cookies are used to identify specific users and improve your web browsing experience. Cookies let websites remember you, your website logins, shopping carts and more.
Cookies are set using the **Set-Cookie** HTTP header, sent in an HTTP response from the web server. This header instructs the web browser to store the cookie and send it back in future requests to the server (the browser will ignore this header if it does not support cookies or has disabled cookies). then the cookie is sent with requests made to the same server inside a **Cookie** HTTP header.

6 What are the different attributes that can be applied while setting a cookie? What do they achieve?

Few attributes which can be applied while setting a cookie are-

1. Permanent cookies are deleted at a date specified by the **Expires** attribute, or after a period of time specified by the **Max-Age** attribute.

2. A cookie with the **Secure** attribute is sent to the server only with an encrypted request over the HTTPS protocol

3. The **Domain** attribute specifies which hosts are allowed to receive the cookie.

4. The **Path** attribute indicates a URL path that must exist in the requested URL in order to send the Cookie header.

7 How do cookies assist in advertising companies (such as Google) being able to track users across different sites?

Tracking cookies are cookies that either a website or a third-party stores on web browsers. Most often they are third-party cookies.

Such cookies track the user's online behavior. In other words, tracking cookies collect their data, such as clicks, shopping preferences, device specifications, locations, and search history. This data helps in targeted advertising or gathering website analytics.

8 When is the CORS mechanism required?

Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served.
CORS defines a way in which a browser and server can interact to determine whether it is safe to allow the cross-origin request. It allows for more freedom and functionality than purely same-origin requests, but is more secure than simply allowing all cross-origin requests.

9 What are the headers set in CORS and how do their values affect things?

The Cross-Origin Resource Sharing standard works by adding new HTTP headers that let servers describe which origins are permitted to read that information from a web browser.

10 What are CORS preflight requests? When are these requests sent? How does a CORS preflight request look like?

In Preflighted requests the browser first sends an HTTP request using the OPTIONS method to the resource on the other origin, in order to determine if the actual request is safe to send. Cross-site requests are preflighted like this since they may have implications to user data.
The following is an example of a preflighted request:
OPTIONS /doc HTTP/1.1
Host: bar.other
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:71.0) Gecko/20100101 Firefox/71.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Connection: keep-alive
Origin: http://foo.example
Access-Control-Request-Method: POST
Access-Control-Request-Headers: X-PINGOTHER, Content-Type

HTTP/1.1 204 No Content
Date: Mon, 01 Dec 2008 01:15:39 GMT
Server: Apache/2
Access-Control-Allow-Origin: https://foo.example
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-PINGOTHER, Content-Type
Access-Control-Max-Age: 86400
Vary: Accept-Encoding, Origin
Keep-Alive: timeout=2, max=100
Connection: Keep-Alive
Lines 1 - 10 above represent the preflight request with the OPTIONS method.