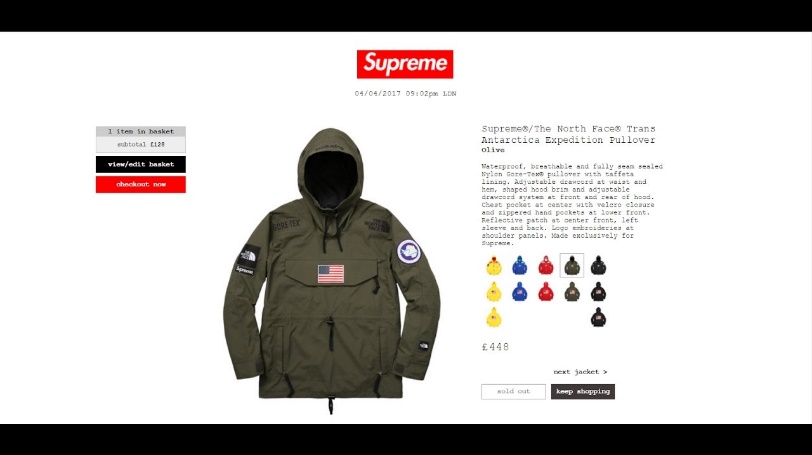


Supreme Bot Project

By Chris Abraham





Supreme

Contents

[**Project Proposal** 3](#_Toc5184221)

[**Market Research** 4](#_Toc5184222)

[**Research Conclusions** 5](#_Toc5184223)

[Research ctd 6](#_Toc5184224)

[**Gantt Charts** 7](#_Toc5184225)

[**Initial Requirement Specifications** 8](#_Toc5184226)

[Scope 8](#_Toc5184227)

[Boundaries 8](#_Toc5184228)

[Target User 8](#_Toc5184229)

[Functional Requirements 8](#_Toc5184230)

[User requirements 8](#_Toc5184231)

[Feasibility 9](#_Toc5184232)

[Technical Requirements 9](#_Toc5184233)

[Requirement Changes: 9](#_Toc5184234)

[**Test Plan** 10](#_Toc5184235)

[Test Plan Changes 10](#_Toc5184236)

[**Interface** 11](#_Toc5184237)

[Interface Design Changes 11](#_Toc5184238)

[**Extension Structure** 12](#_Toc5184239)

[**Database** 14](#_Toc5184240)

[Database Changes 14](#_Toc5184241)

[**Linking the extension to server** 16](#_Toc5184242)

[**Registering a New User** 19](#_Toc5184243)

[**Edit** **Details** 20](#_Toc5184244)

[**Login Validation** 21](#_Toc5184245)

[**Profile Page** 22](#_Toc5184246)

[**Order Form** 22](#_Toc5184247)

[**Final Testing** 23](#_Toc5184248)

[**End User Testing** 24](#_Toc5184249)

[**Evaluation** 25](#_Toc5184250)

[Appendix 2: Designs 26](#_Toc5184251)

[2.1 User Details Table 26](#_Toc5184252)

[2.2 Order Details Table 26](#_Toc5184253)

[2.3 SignUp 27](#_Toc5184254)

[2.4 Edit Details php script 28](#_Toc5184255)

[Appendix 3: Implementation 29](#_Toc5184256)

[3.1 Popup 29](#_Toc5184257)

[3.2 Database Screenshots 29](#_Toc5184258)

[3.3 SignUp code 29](#_Toc5184259)

[3.4 Edit Details Code 29](#_Toc5184260)

[3.5 Login Code 29](#_Toc5184261)

[3.6 Profile Code 29](#_Toc5184262)

[3.7 Order Form code 29](#_Toc5184263)

[Appendix 4: Testing 30](#_Toc5184264)

[4.2 Login Validation 31](#_Toc5184265)

# **Project Proposal**

For my advanced higher computing project, I plan to make a chrome extension which is able to automatically checkout on a desired item specifically from the website: [www.supremeneyork.com/shop](http://www.supremeneyork.com/shop), this extension will make use of chrome APIs to manipulate the DOM of the webpage, to buy the customers item.

As a part of this I will make a platform using HTML, CSS and JavaScript in which customers can securely login to store their checkout details:

* Full name
* Email address
* Address Lines
* Postcode
* Telephone
* Card number
* Security Number
* Expiry Date

These details are saved to a user profile in **a database**, making use of **server-side scripting**, this will allow the data to be accessed when needed for the final purchase. The main function of the program is to be able to enter the following details of an item:

* Category of the Item
* Exact Name of item
* Colour(s)
* Size(s)

This will allow the program to **search** for the item on the supreme page when it appears and initiate the buying process.

As a part of designing a user-oriented program, I will research about my target market, by messaging people on social media who show an interest in the hype culture of supreme. I also can ask people I see in Edinburgh that are wearing hyped brands to further gain an understanding of what the needs of end users would be.

The items sold by supreme sell out very quickly, due to the hype of an item they can sell out in under 5 seconds, for any chance of getting it, people use programs called ‘bots’ to attempt to buy the item in the quickest time possible eliminating human error. This is what I hope to achieve in this project.

# **Market Research**

To do my market research I talked to people that fit the target market of hype beast and roughly fit these questions into the conversation. I didn’t want to bore them with a list of survey questions as a conversation allows a more open discussion to get better and more useful answers.

1. Have you ever used a supreme bot?
   1. If yes, was it a positive experience?
   2. If no, would you want to?

what’s stopping you? (cost, reliability, security, other)

1. Would you trust a bot with all your credit card details?
2. How much would you honestly pay for a bot with 100% cop rate?

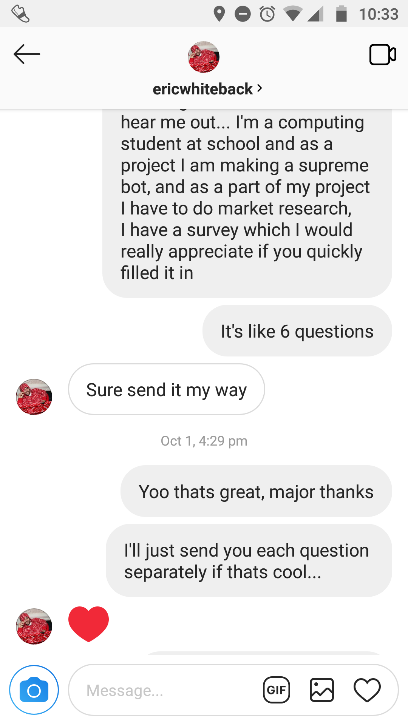
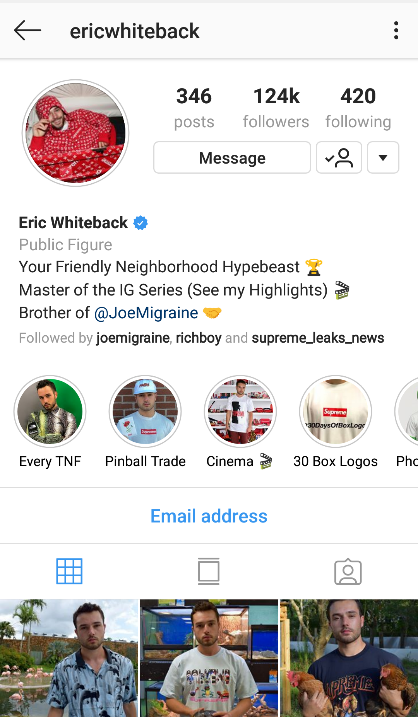
(£0-£10) (£11-£50) (£51-£100)(£100+)

1. Name a feature you would want to add to a “dream supreme bot”.
2. Do you tend to buy multiple items in a drop or just focus on one?
3. Bonus Question, out of interest…

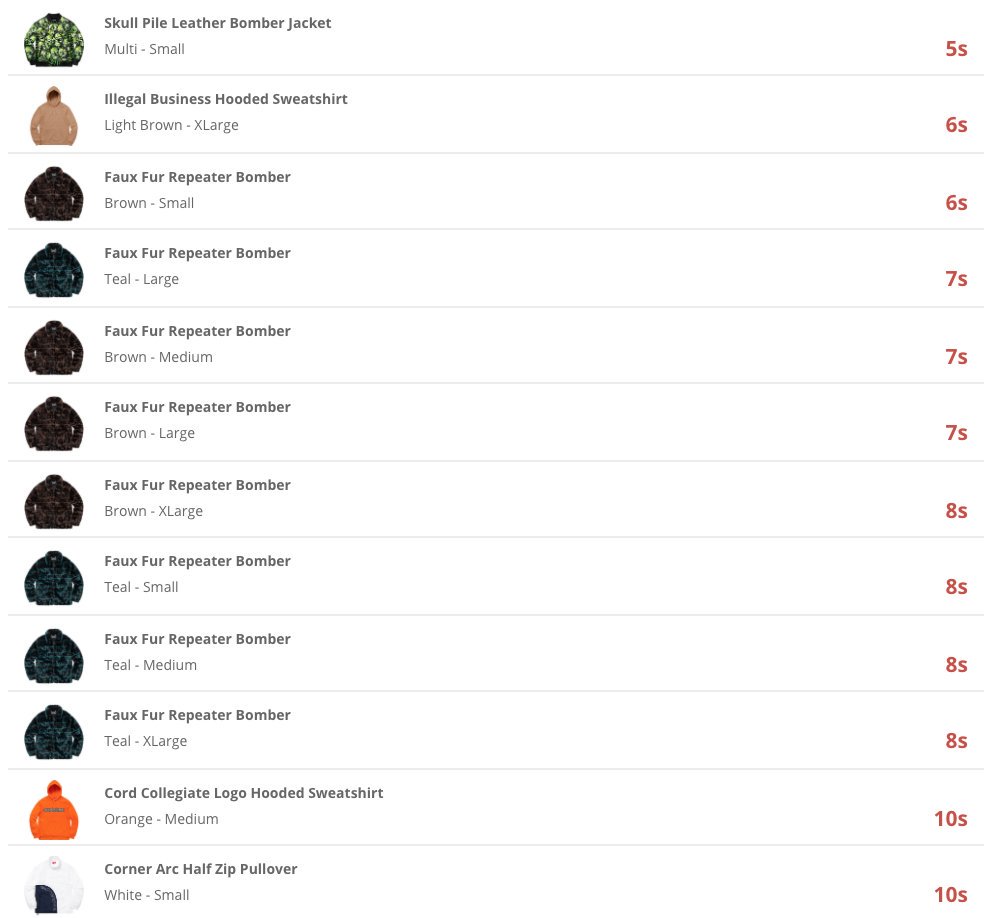
What is the most hyped item you own?

# **Research Conclusions**

My market research went very well as I was able to get a much better idea on people’s thoughts about supreme bots buying items automatically. I was able to speak to 10 people in total, through social media and in person. One of the people I talked to is a famous hype beast on Instagram @ericwhiteback

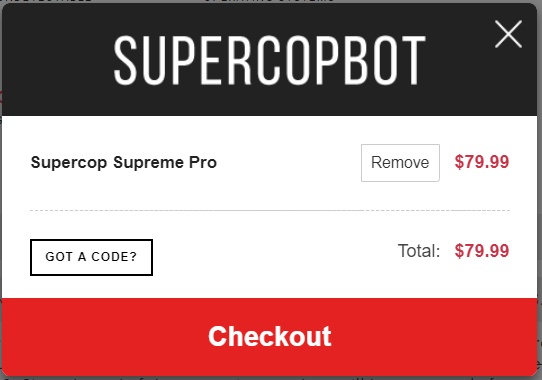


He was very helpful, as he has been following and buying supreme for a long time and has plenty experience with bots.

From chatting with Eric, others on social media and people I know I have a much better idea of the opinions of the target market. Out of the 10 I spoke to, 7 had never used a bot before and 3 had. Out of the 7 people who hadn’t used a bot, it was mainly due to the price of them, and they didn’t trust that they would be reliable enough for the price paid. The 3 people who had used them previously had mixed experiences, some saying very positive and some not so good, from further investigation I found this is because bots are never 100% consistent at getting items, they can’t ever guarantee the order of an item. With further conversation I found that this could be due to very small external changes such as internet transmission speed and the speed of the hardware. This can affect the order by seconds which can be devastating if you are buying supreme. If the item is very hyped it can easily sell out in under 5 seconds.

## Research ctd

To further my research, I have looked at the bots that my target market said worked best. These are also the most popular bots to be used:



The rough price of $80 for the whole season converts to about £62, this is a stupid price considering the reliability of bots, it should be priced a lot lower, and is clearly shown in the market research as 9/10 think that bots are “unnecessarily expensive”. The more appropriate price range was said to be around £20.

From 9/10 people I spoke to, they didn’t mind how elaborate the bot was, they only cared that it “attempted to buy the item quickly” and is priced much cheaper than the rest of the market. Most people have accepted that “a bot will never have a 100% success rate” but believe it is still much better than buying it manually.

# **Gantt Charts**

I have many versions of my Gantt chart as I made many changes which resulted in a change in the project plan:

I initially made a rough plan with estimates to fill the time to the deadline appendix 1.1

This was quickly changed to be a development strategy, where I went through three parts of the project simultaneously: interface design (HTML&CSS), server-side scripting, java scripting (bot) Appendix 1.2

During the project I made a decision to approach the development slightly different, this way I would tackle each problem as its own project, designing, implementing and testing it along the way, this proved to be very efficient and helpful for the course of the project as it helped me set achievable goals and gave more sense of satisfaction completing a part of the project. Appendix 1.3

Then after a large delay in development as specified in my record of progress, I had to remove the supreme bot functionality, to account for the missing time. Appendix 1.4

# **Initial Requirement Specifications**

## Scope

1. Open the extension on any page and login
   1. Validate a username and password
      1. Ensure secure transmission of data
      2. Allow a new user to make an account
2. Once logged in, allow user to edit personal details in a profile tab
   1. These details must be very secure since it is card details.
3. Show a record of previous attempted purchase and if they succeeded or not.
4. New order tab which has a form to identify an item to buy in an upcoming drop.
5. Once activated it will redirect the user to the category of the chosen item and start to refresh searching for the desired item
6. It will continue to refresh until the item is found
   1. If not found by 11:05 it will automatically deactivate.
7. Once found it will click the item taking the use to that page
8. The colour will be selected
9. One of the desired sizes will be selected
10. The bot will try to add the item to basket
    1. If failed return the stage failed at and deactivate process.
11. Once at the checkout page, the bot will autofill all the respective details, and leave the final click to the user to confirm the order.
12. After the whole process allow an order report to be filled out, if it was successful and times of sell-out.

## Boundaries

1. The bot will not guarantee the successful purchase of an item since orders are affected by many external factors.

## Target User

This program will be targeted at ‘hype beasts’ the term given to people that chase popular brands such as supreme, specifically this will include 17-25 age range and mostly male.

## Functional Requirements

To keep this project at an advanced higher level I need to create a platform where the user can register with personal details (with validation throughout), the details are stored in a database and can be accessed when the user can login to the profile. Allow the user to edit the details on login.

## User requirements

1. Login to an interface and be able to fill out checkout details before an order
2. Be able to specify an item name, size and colour which will be selected on the drop
3. The extension will select that item and add to basket taking the user through the checkout process
4. Allow the user to record the success rate of orders in a log

## Feasibility

* Cost- there is no cost involved in the project as all the software needed is free and the hardware needed is all accessible for free.
* Time- I have until the Easter holidays, I should be able to do it in time, however I cannot be sure as I am learning new concepts and don’t know exactly how long it will take
* Legal – Supreme have not made any statement against the use of bots and there are many other commercial bots on the market, however the one legality is that any item purchased using the bot must be for person use and not to be sold on.

## Technical Requirements

* An extension interface hosted locally on the chrome browser
* A database to store the data of the user and be able to access it when needed in the order.
* Be able to interact with the chrome page, to click links and enter search data

# Requirement Changes:

Over the course of the project I was forced to make changes to the requirements due to various reasons:

9/11/18: Realising I am unaware and unable to learn how to store a photo in the database and pass the photo as data between server and extension, this must be removed from the spec.

8/11/18: Making my data dictionary I realised that storing users card details will require significant security associated with it, however since I had already accounted for the time I had, I didn’t have extra time to learn and implement these security precautions. Which meant I added the below statement to the project’s boundaries.

“The bot will have minimal security, I am aware of the presence of customers credit card details being part of the data stored, however since integrating security into this bot would take too much time to learn and implement, I have assumed a world of no hackers or malicious attacks.”

28/11/18: While implementing the link between the extension and the php which manipulates the server, I came across a barrier to do with the practicalities and had to take more time to address the issue. Consequently, I had to remove time at the end of the project for the “supreme bot” functions. At the time I was unaware how much functionality this would result in losing, however nearing the end of the project It was evident that I couldn’t do any of it in the time scale. So, I had to remove steps 5)-12), this is all I can remove from the project without losing the advanced higher-level content mark.

# **Test Plan**

If my project is to work according to projectScope1, I will test it with an example case:

1. Set up a new user with:
   * 1. username – coreyHamilton
     2. password – hampton101
2. Add details of checkout: (randomly generated details)
   * 1. Email – [coreyHam@gmail.com](mailto:coreyHam@gmail.com)
     2. Name – Corey Hamilton
     3. Profile Pic – 🡪
     4. Telephone – 07793426789
     5. Address1 – 65 Eden Walk
     6. Address2 – Kilbury
     7. Address3 - Wisting
     8. City – Polington
     9. Postcode – PL34 0DR
     10. Card Number – 1969924629617723
     11. Expiry Date – 06/22
     12. SCN – 674
3. Exit the extension and attempt to log back in to the users profile.
4. From the profile page, attempt to edit the user details to change the telephone to:
5. 07987654321
6. Try to order an item that hasn’t sold out on supreme website (order without time constraints)
7. Attempt to order an item on the drop (test against Supremes’ quick sell-out times)
8. Record these orders (successful or not) in the order log

Success in doing all these things would mean the program is fully working, however if it does all of them apart from step 6, It is still a fully working program as I specified in the scope the bot need only to attempt to buy the item quickly and it can never guarantee a 100% success rate.

The input validation will be tested separately with normal exceptional and extreme data, this will also be recorded. I plan to take screenshots of the program along the way to record the testing and any errors that come up.

After completing the testing above I plan to give the program to 2 people of the target audience to complete the following tasks:

1. Register a new user account with details.
2. Once registered, restart the program and login with details.
3. Update 2 items of personal details
4. Fill out a form for a desired item and submit it
5. Process an order
6. Record the order in order history.

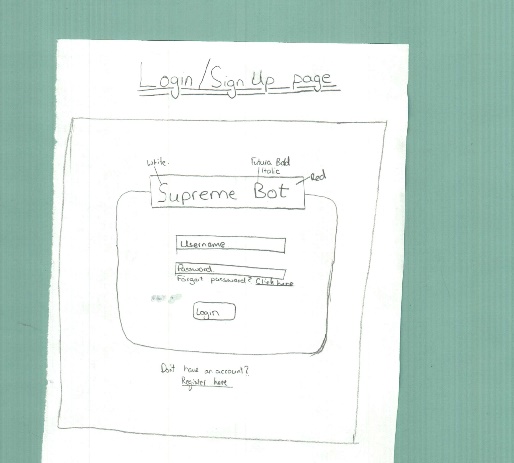
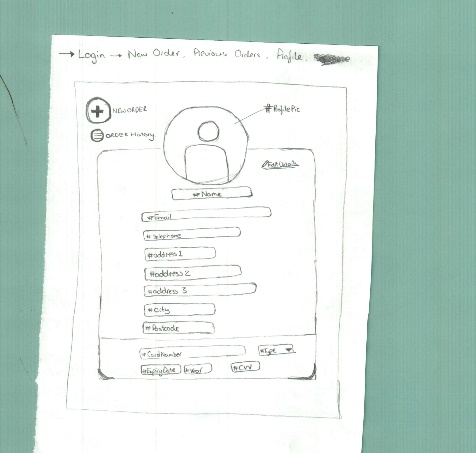
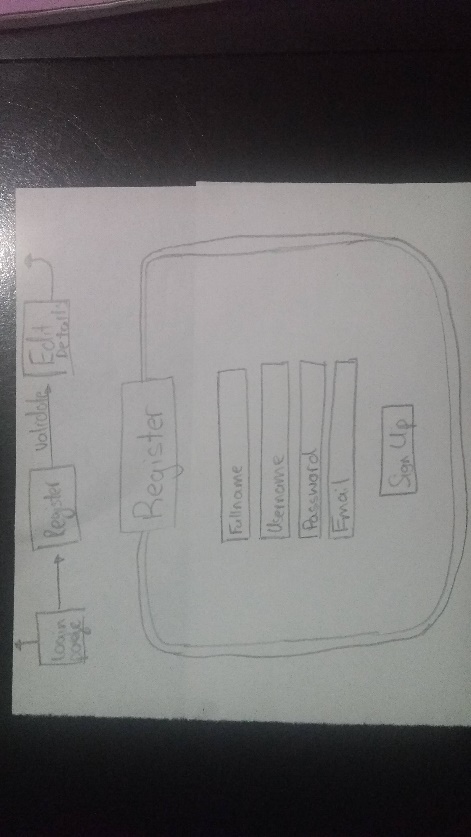
## Test Plan Changes

9/11/18: Since the program will not store a profile photo, step 2)iii) is removed.

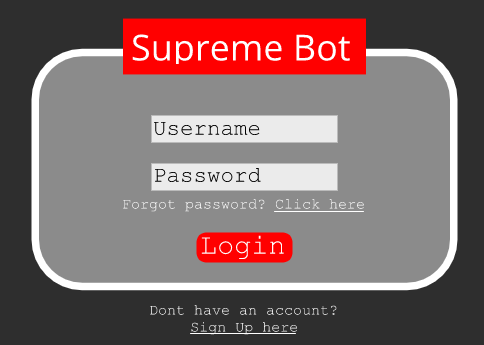
28/12/18: After changes to the project outline the respective changes were also made to the test plan, meaning in alpha testing step 5,6,7 are removed and in beta testing, step 5,6 are removed

# **Interface**

To design the interface of the extension I first drew possible layout for the login page, profile page and register page, being an extension, the window size is flexible but is ideally a small box in the corner

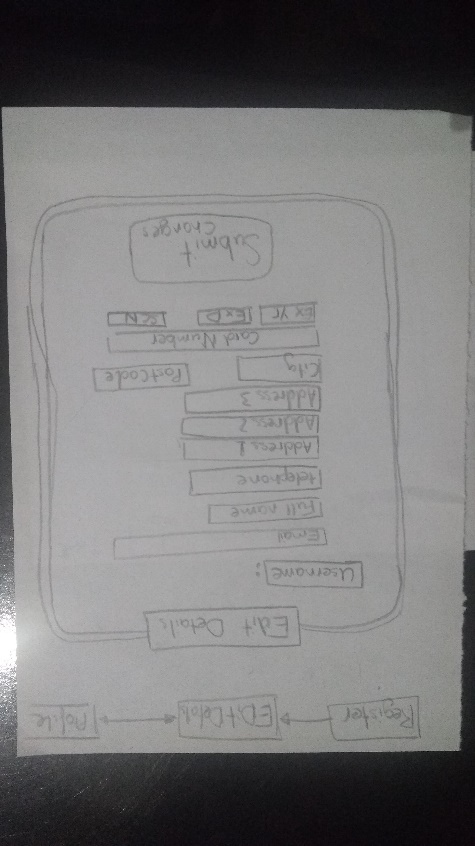
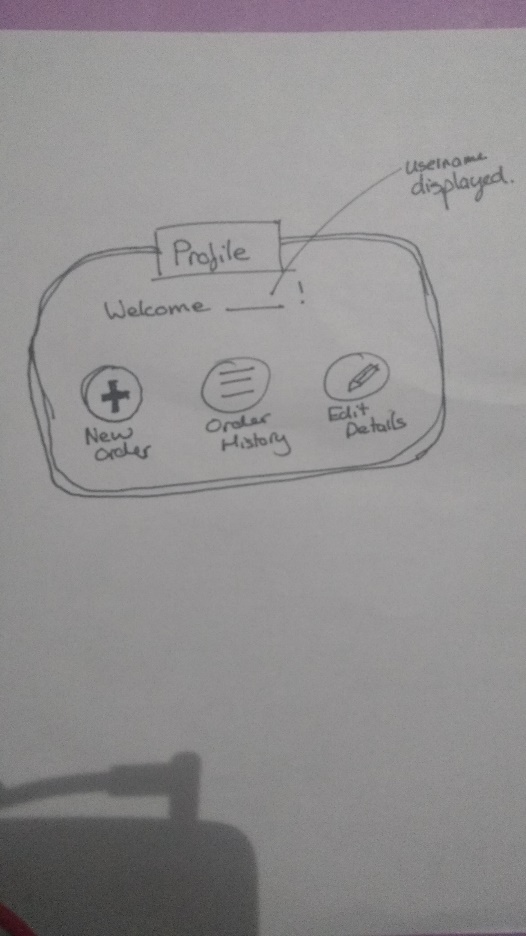
Based on the initial requirements I designed these wireframes:

This was a simple and practical design for the extension, I tried to keep it small and compact to contain it to a corner of the user’s window. I then used a graphics software called Gravit to make the login page with colours, to showcase the colour scheme and ensure it works.



I only made a digital design for the login page to show the colour scheme. The colours, style and overall theme is transferable to other pages through classes in CSS

### Interface Design Changes

9/11/18: After removing the profile picture from the details that are stored for the user the design also needed to be changed, to remove the space for it. While doing this I rethought the pages completely to be a bit simpler for the user, as in hindsight the first profile page was a bit cluttered. The profile page was reduced to three buttons leading to other pages.

# **Extension Structure**

First thing to design for the project was the basic backbone structure of the extension, made up of a manifest.json file which acts as a contents page directing the browser to the location of the html that is run when the extension is called. I designed the manifest according to the template given on chrome developer tools, which shows how simple it is.

03/11/18:

*“Name: supremeExtension*

*Version: 1.0*

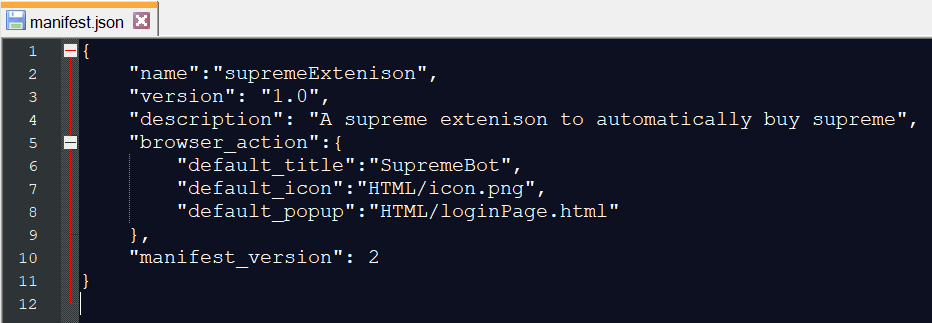
*Description: A supreme extension to automatically buy supreme*

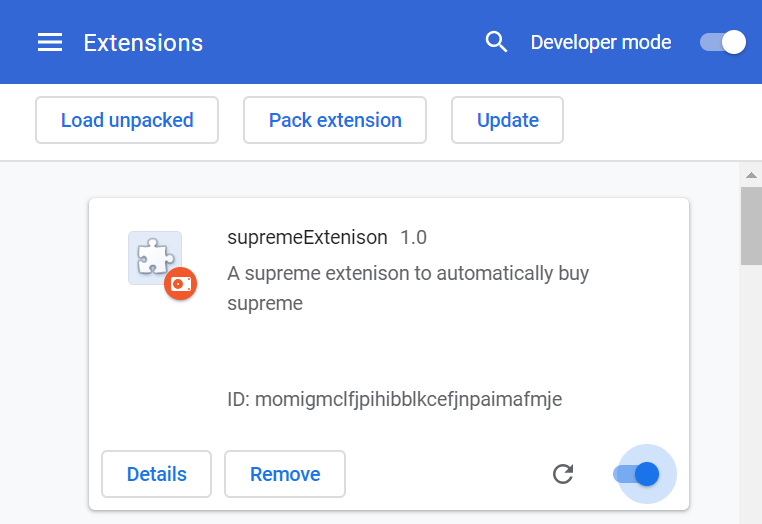
*Permissions: to be confirmed (through research)*

*Default popup: popup.html (\*name of the html file)*

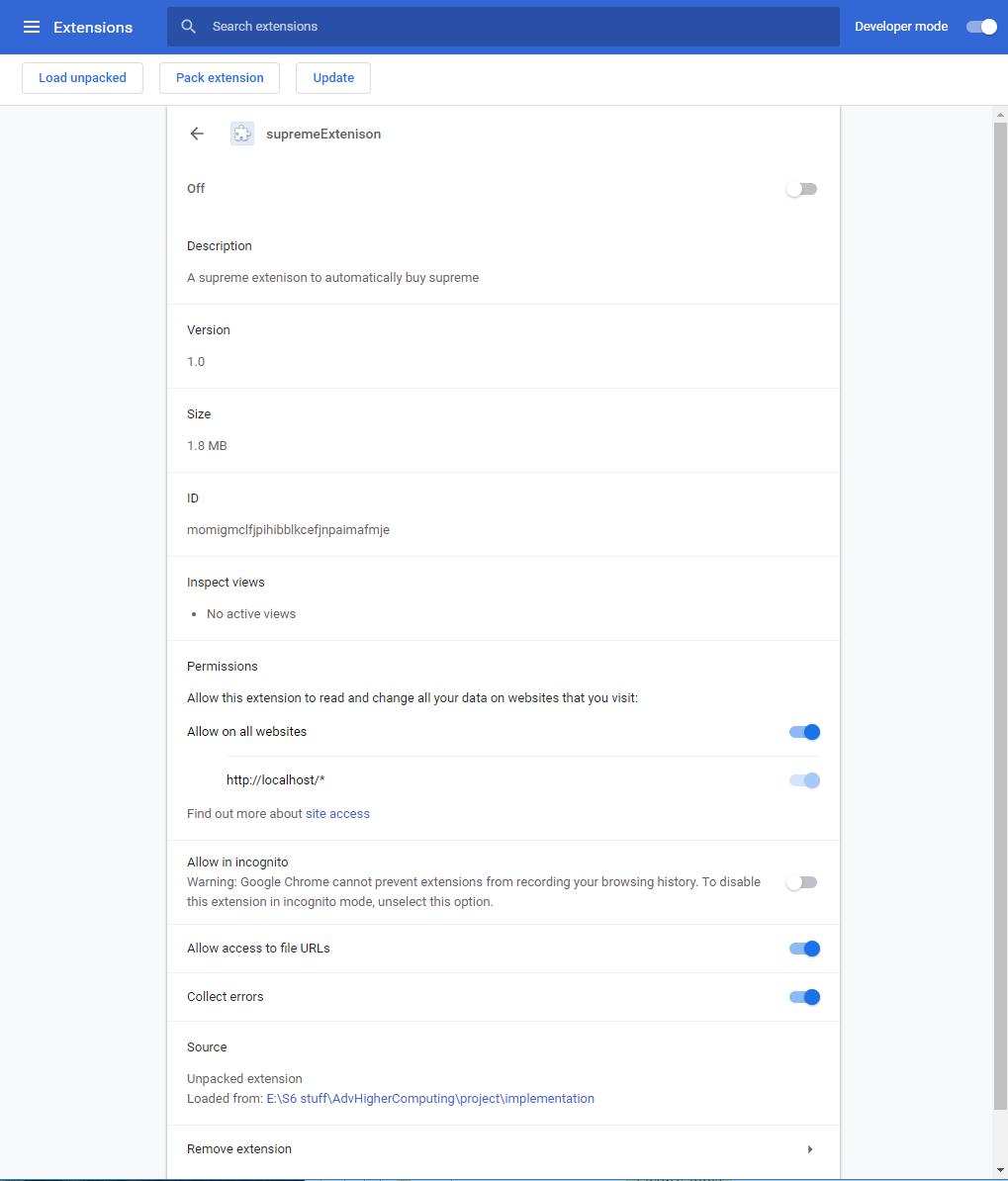
*Default icon: icon.png (\*name of icon file)”*

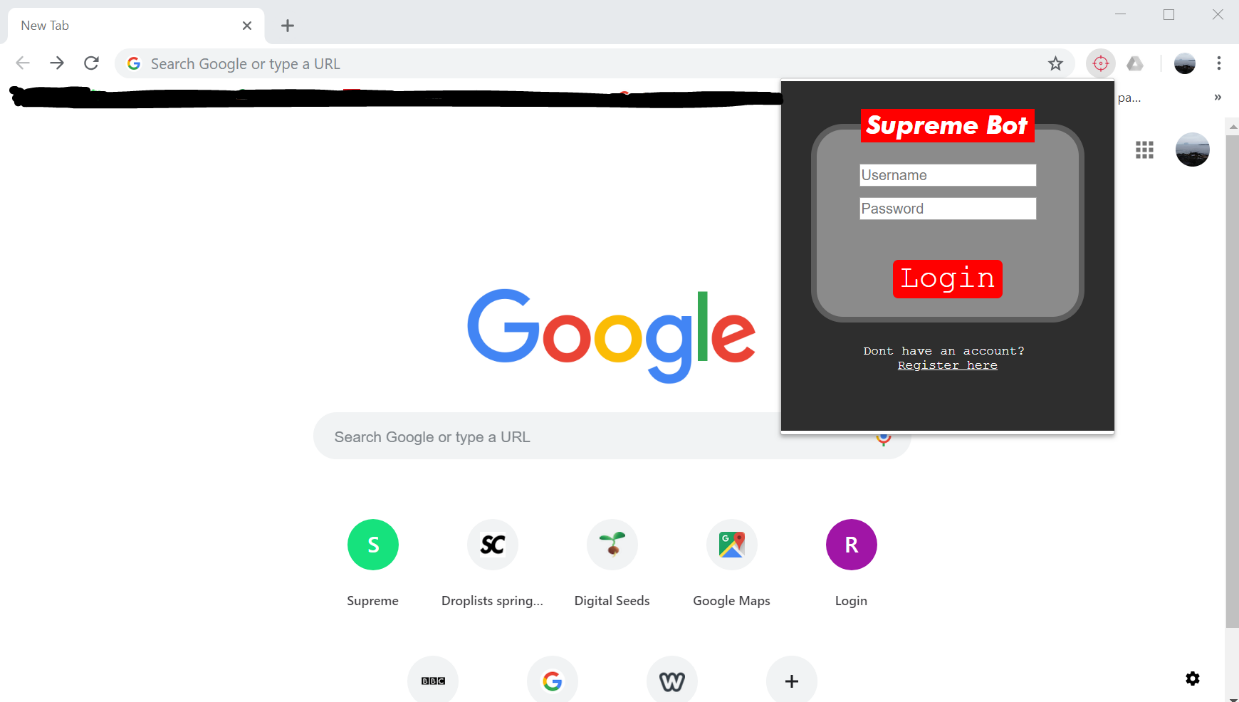
This is to be tested by simply clicking the icon and it displays the login screen to match the interface design made earlier.

After a few simple syntax errors in the manifest caused by missing commas or speech marks, I was able to display the page as hoped and passed testing. \*see Appendix 3.1 for the manifest and the loginPage HTML with associated CSS

To run this, I loaded the manifest with the location of the popup into chrome://extensions

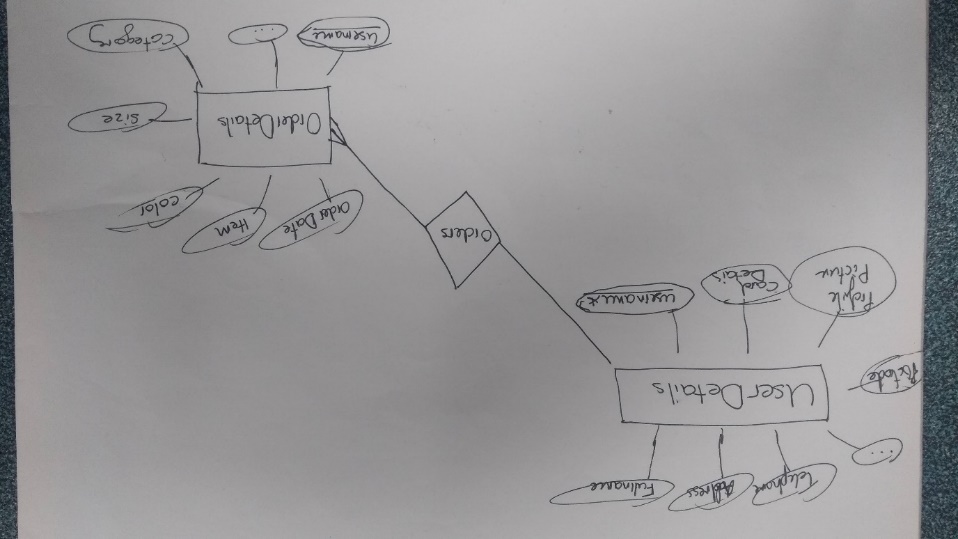
On the developer tools in chrome I can access the details of the extension



When activated the icon appears in the chrome browser toolbar, revealing the popup (see Appendix3.1 for code)

# **Database**

Next on the list, was to create the data structure to hold the details of the user for the extension to access. First made an entity relationship diagram to roughly show the structure of how the tables link, using username as a primary/foreign key, the specific data that needs to be stored will be further specified and detailed in the data dictionary.



I used a data dictionary (see Appendix 2.1 & 2.2) to design both tables in more detail specifying the validation on each field. To test the structure, I designed a data set to be entered to the database and a query that tested whether the user details are linked to the order details:

### Database Changes

09/11/18: I am not implementing a profile picture, so it will be removed from any tables from now

10/11/18: Considering that I am already going to have to do validation on the form (scripting side) it makes sense that there is no need for validation on the database, when all data entered is already validated. In an ideal world I would implement validation on both, to ensure security, since scripting can be overridden by malicious attacks. To test my database, I am just going to test the relationship between the tables.

To test the relationship these are the users that I will add into the “userDetails” table:

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Sample 1 | Sample 2 | Sample 3 |
| Username | davidKing119 | jamesNelson | susanWilson |
| Password | preme\_King34 | JellieN\_956 | susie101 |
| Email | davidKing@gmail.com | jamesNelson@gmail.com | susanWilson@gmail.com |
| Full name | David King | James Nelson | Susan Wilson |
| Telephone | 07123456789 | 07123498756 | 07987654321 |
| Address1 | 61 Bolven Walk | 48 Grove St | 34 Park St |
| Address2 | Lotherton | Firlington | Kirbington |
| Address3 | Witlockery | Readingly | Harrow |
| City | Bradbury | Gogelbury | Thistle |
| Postcode | BT93 0YR | GH45 0HY | TR23 0FR |
| Card Number | 5159927655617746 | 1749927652967746 | 512392764517746 |
| Expiry date | 01 | 27 | 05 |
| Expiry Year | 2021 | 2017 | 2019 |
| SCN | 162 | 698 | 409 |

And these orderDetails:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Sample 1 | Sample 2 | Sample 3 | Sample 4 |
| User ID | davidKing119 | davidKing119 | davidKing119 | jamesNelson |
| Order Date | 2018-11-08 | 2018-10-25 | 2018-10-18 | 2018-11-01 |
| Item Category | Shoes | Sweatshirt | Jacket | Accessories |
| Item Keywords | Supreme Nike Air Force 1’s | Raglan Sweater | North Face Pullover | Thermal Crew Neck |
| Item Sell-out time | 20 | 40 | 10 | N/A |
| Order-success? | Card Decline | Yes | Sold Out | Yes |
| Retail Price | 130 | 120 | 120 | 20 |
| Resale Price | 230 | 150 | 250 | N/A |

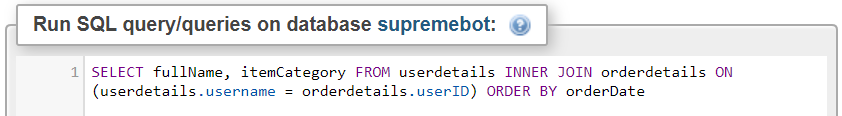
Finally, I will query the database to ensure the data of the orders link to the respective user.

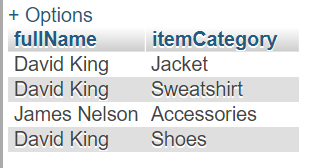
|  |  |
| --- | --- |
| Fields and calculations | userDetails.fullName , orderDetails.itemCategory |
| Tables and queries | userDetails, orderDetails |
| Search criteria | none |
| Grouping | none |
| Sort Order | orderDate |

|  |  |
| --- | --- |
| David King | Jacket |
| David King | Sweatshirt |
| James Nelson | Accessories |
| David King | Shoes |

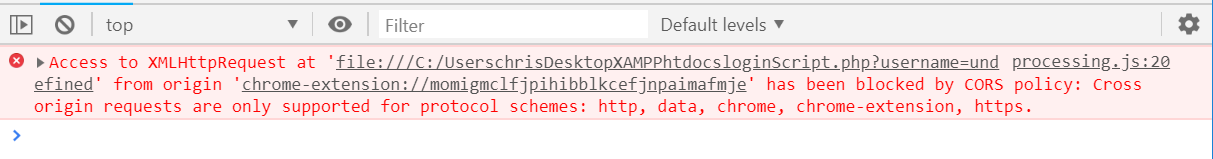
Expected result (show orders in chronological order with the full name associated):

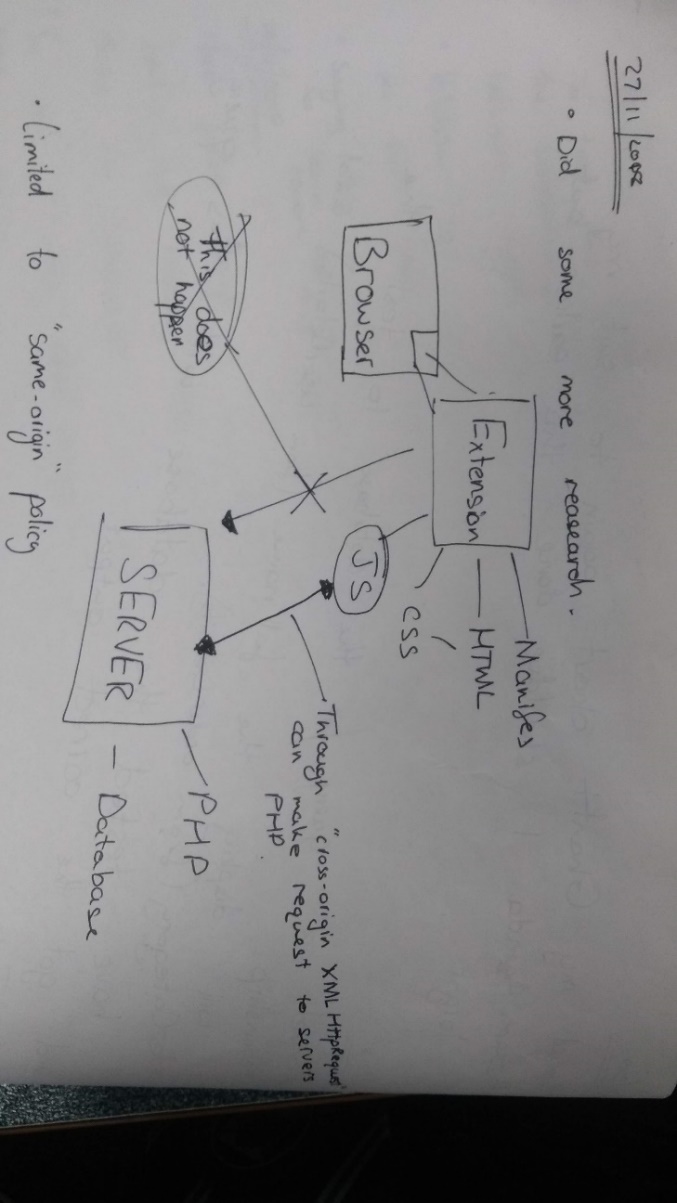
The evidence of implementation is all in Appendix 3.2

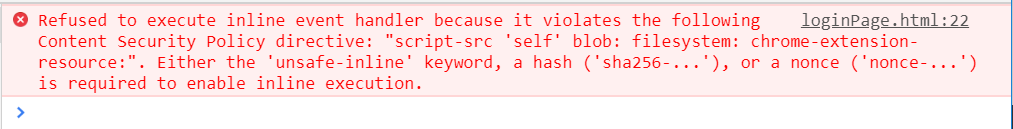
And when tested with the above query it produced the expected result:



# **Linking the extension to server**

Next on the agenda was to establish a connection between the server and the extension, this was a vital part of the project as all the advanced higher level content rests on this working. Initially I had a rough idea of how the code would work, however this was soon thrown out the window as I never considered that being an extension, it should be treated differently. The extension file itself was restricted to a local file on my laptop for my browser to access it, and the server side php was restricted to a file on my C:drive, trying to access each other directly using the root address brought the error:

Cross-origin resource sharing (CORS), is the policy restricting a site requesting data from external servers, for security reason, for extensions this policy is very strict in such that data can only be accessed from the “same origin” (same server). As I am trying to access data from http:// from a local file on my laptop, that is breaching the policy. The only way to overcome this is an external API “Cross-Origin XMLHttpRequest” (XHR)

When I started implementing this, I came across another error.

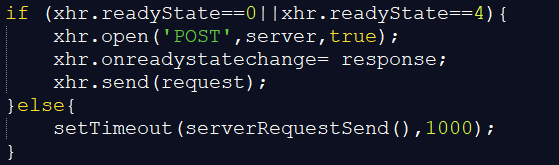
This was a lot easier to fix as the error was rather explicit with what was wrong. Another restriction that extensions have, is that they do not allow any inline JavaScript, meaning it all must be purely external. This error came as a result of me using onclick() in a button element, instead I had to identify the element in JavaScript with its ID.

To test this aspect of the program I plan to just make sure that I can send and receive variables to the server from the extension, so for this test I will send a simple query:

*“Enter the username of a user and return that users fullName”*

This step of the project took much longer than expected as I was trying to learn and use the external API without any previous knowledge, I used YouTube explanations and support forums to aid the process

The basis of the code was:

1. Initial creating a XMLHttpRequest() object for use throughout the program.
2. If the server is in a “State” where it can receive a new request.
3. “open” a new request with the type of request and the address of the php script on the server.
4. State that when the server changes state (has finished processing request) call another function to deal with response
5. Finally send the request with data to be sent as the parameter.
6. The timeout is in place if the server is not in a ready state, it will call the function again in 1 second, at which point the server should be free.

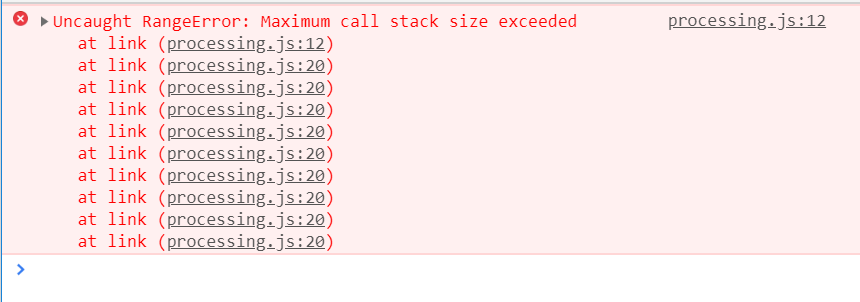
On the server side, the php script will be called which then takes the data that is sent and does its processing to send back its response to the extension. Data is sent back in the form of an xml object, which is very similar to a HTML parent object with various children, to access the specific data in the xml object, I used selectors for the known location.

One difficulty I repeatedly experienced was the lack of detail when coming across errors in my php, compared to JavaScript errors. If there is any error in the php (syntax), the script will return:

To overcome this problem, instead of directly attempting to access the results I printed the raw xml object returned by the php to the console, which normally contained a more detailed error message.



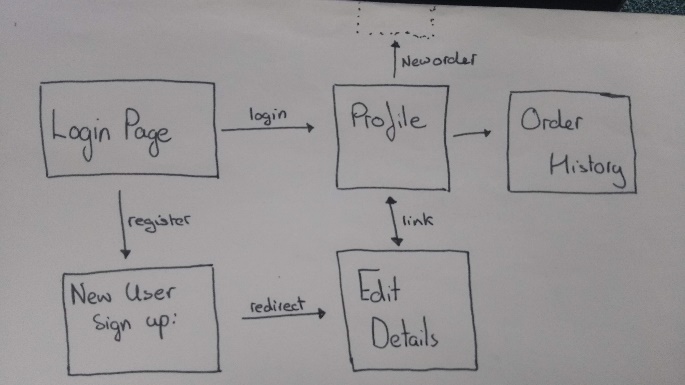
After a lot of time and effort I was able to produce the expected result, and I was familiar with using XMLHttpRequest to process requests.

In further development I came across another error which was interesting:

At the time I was unsure why it was occurring, however after doing some of the computing course learning about stacks and recursion. I realised that the function was being called infinitely causing an overflow in the task stack. This was because the server was not in an available state, which meant the script ran over and over. Once sorting the settings, the program ran fine. That is why I added a time out function which gives the server some time before it tries again to query it.

# **Registering a New User**

Having made the link to the server, it was time to put that to use registering a new user. To do this I initially planned to do 1 large sign up form where the user puts all the details. After reconsideration at this stage I had a more efficient idea.

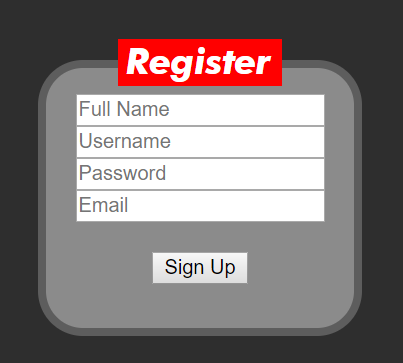
In the project specifications I have mentioned that I will allow users to edit their details when logged on. It makes sense to reuse the sign-up form to edit details. When a user wants to register they enter their:

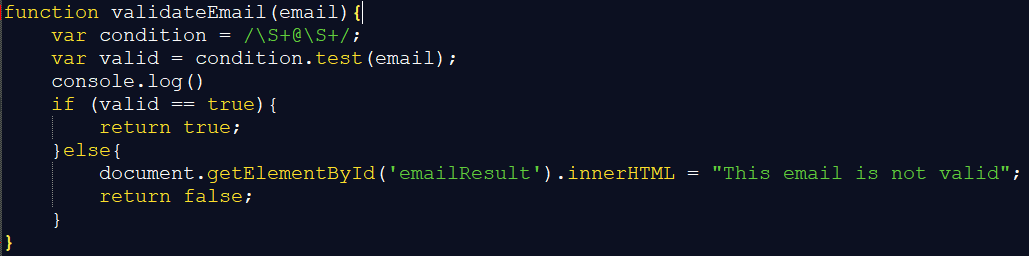
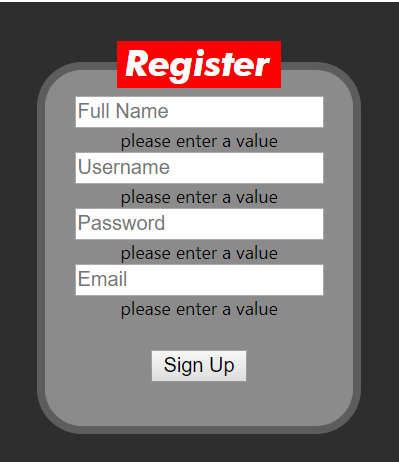
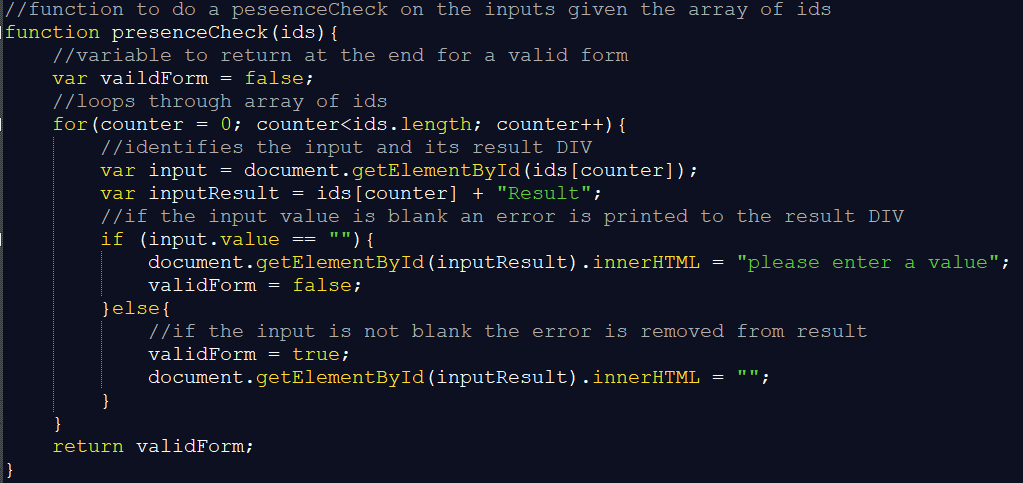
* Full Name
* Desired username
* Desired Password
* Email

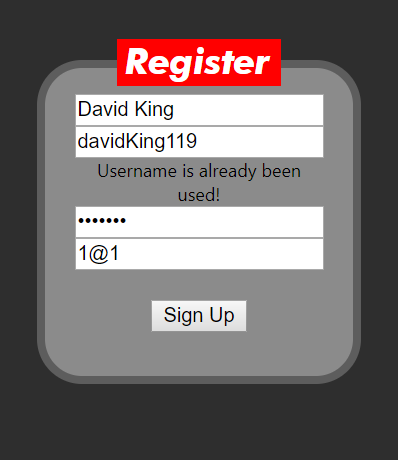
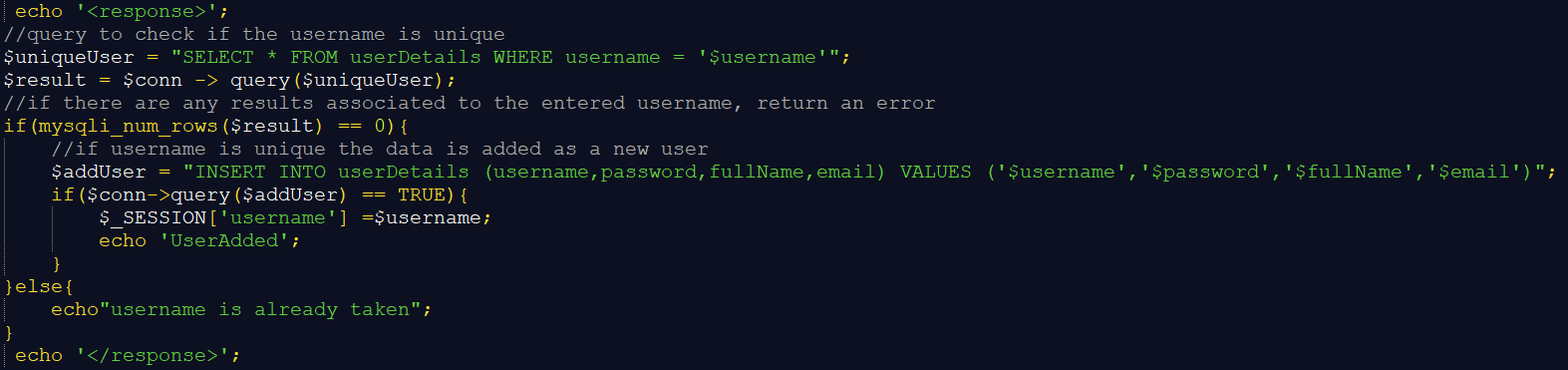
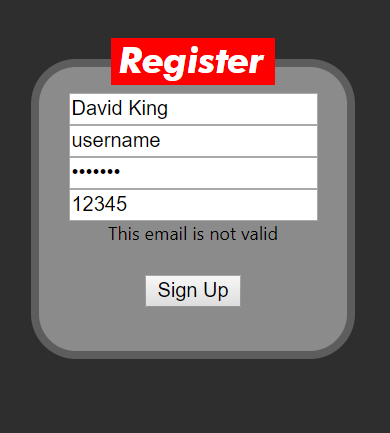
The program will then validate the data, and check whether the username is available, if yes, then redirect the user to edit details page where they can fill in the rest of there details, it is the same form which an existing user can login and go to edit their existing stored details.

\*See Appendix 2.3 for more in-depth design (pseudocode functions) of the sign up process.

To test the sign up process of the program I planned to register a new user and for each input, enter a normal, extreme, exceptional value, to ensure the data sent to the server is valid, check if the username is not already taken, and when sent the data should be added to the database, to the respective records. It should also display clear and informative error messages if data is invalid.

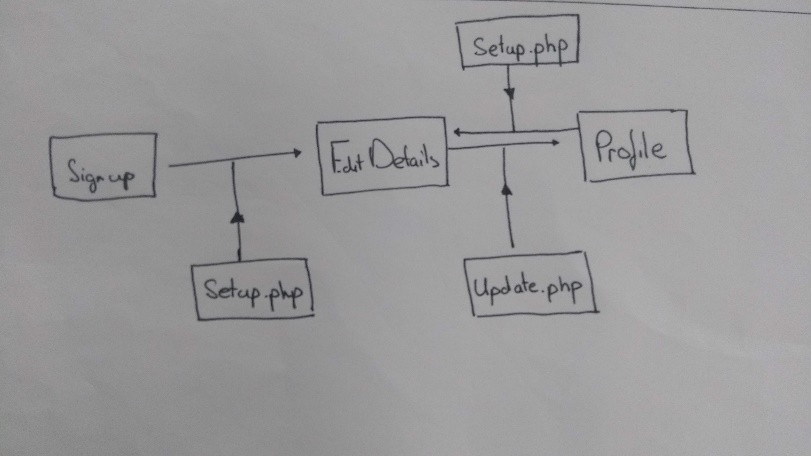




 \*See Appendix 3.3 for all code

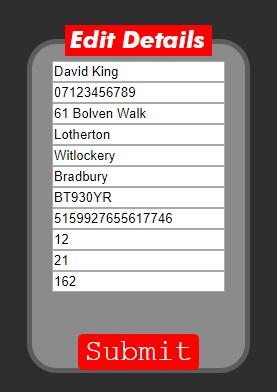
Finally, when valid data is entered it is saved on the database and the user is redirected to the edit details page…

# **Edit** **Details**

Instead of having the sign-up form used once, I plan to reuse it as an edit details page, this means loading all the details on that user into the form for them to see and edit if needed. To do this, I will need to use SESSION variables. Adding session\_start() at the top of my php and setting the session variable to be the currently logged in user’s username, allows the program to identify the user and attain any related details that are needed. This will enable the edit details page to display the existing information on that user, by simply requesting for the fields of data associated with the $SESSION\_[‘username’].

I have designed it so when edit details is called, a php script will be run first, to retrieve that user’s data and fill it into the form, and when leaving the edit details page (after entering valid data) another script will send the new details to the database to be updated. See pseudocode designs of the php script at Appendix 2.4

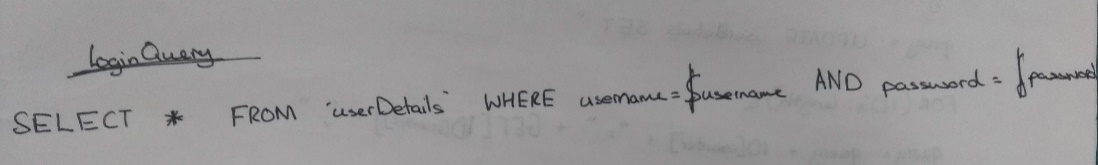
For this to work I will ensure it can display a whole profile of data of a user and be able to edit it, so I will access the page with the session username set as an existing user, then their details should appear, and finally I will change two fields of data and check if it has been updated in the database. I also will have to do the standard testing of the validation, normal, extreme and exceptional.

\*See code at Appendix 3.4

As shown in the picture to the right, the form works, as soon as it loads the details are loaded into the inputs, ready to be edited. Changing two items and clicking submit, triggers the update.php which sent the data to the database to be changed, and the user is directed to the profile page.

# **Login Validation**

Now that the user can register, its time to allow them to then login using a username and password. This is easy considering I already have the login page structure it is just a case of implementing the server side and I now am used to the coding processes to access the server and the database. The login form will take in username and password without any validation, then send it to the database, if the username and password match a successful message is sent back, the session variable is set to the username and the user is directed to their profile page. This is the query design for login,



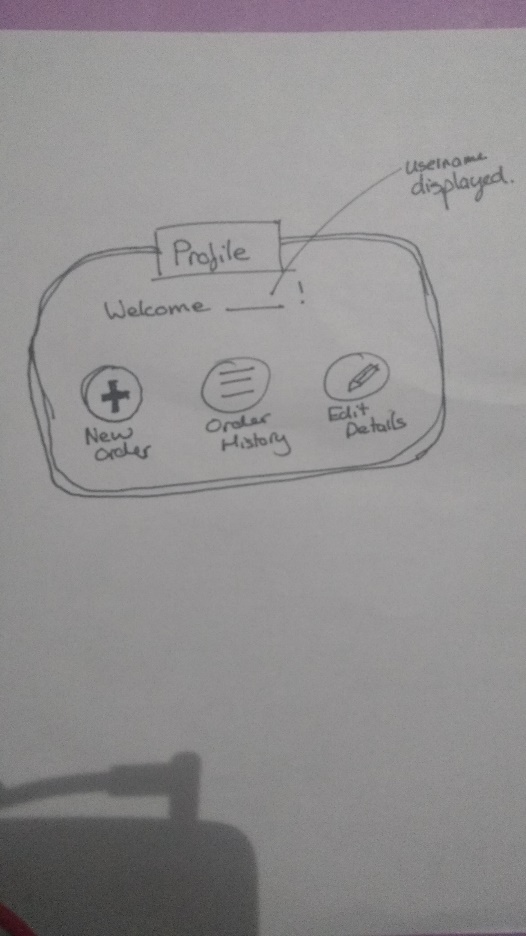
If this returns any results the login is valid, else login is failed.

To test this, I will simply login once with correct details, once with a correct username but wrong password and vice versa.

See Appendix 3.5 for the implementation of the login php script.

See Appendix 4.2 for the testing screenshots

# **Profile Page**

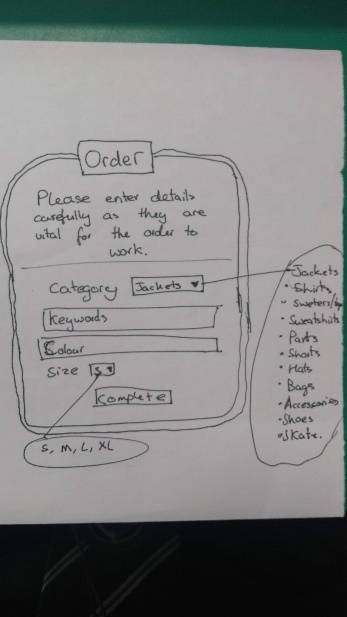
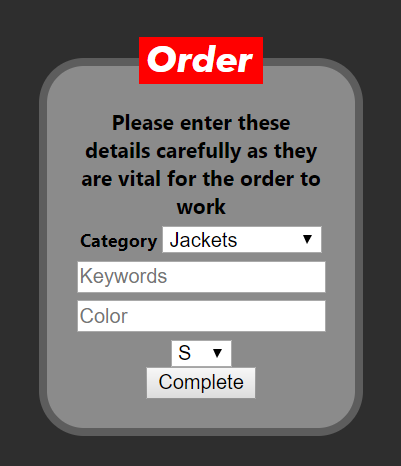
I made the initial design to display all the user details on the profile page, however I quickly changed that, to be easier to understand and navigate. Following this design:

I implemented this interface quickly due to my familiarity with HTML and CSS, I got the icons for free off google, without any copyright restrictions. To display the users full name in the welcome, I had to make a script to find the full name matching the session username.

See code at Appendix 3.6 for full code.

I linked the edit details page and it worked perfectly first time, no problems, I was surprised to be honest. It displayed all the users details immediately and I was able to edit them.

# **Order Form**

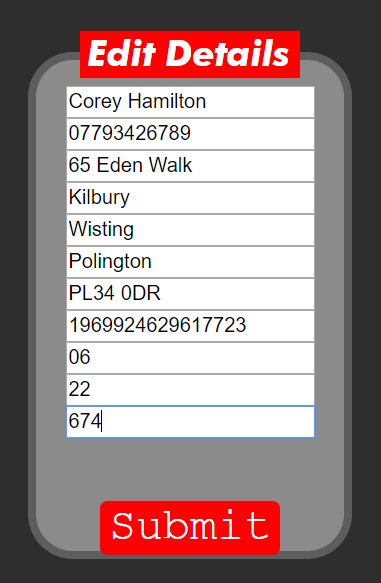
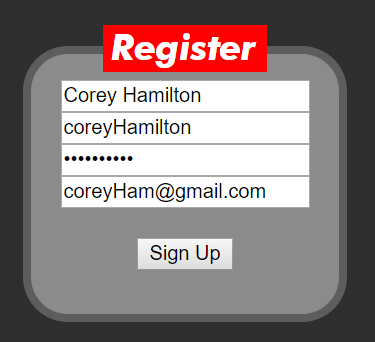
This was another simple task to code a html interface to match the design and to continue the existing colour scheme. See Appendix 3.7 for order form code.

I didn’t implement any validation or processing for the form to send the details to the database, due to time constraints I needed to start the write up and evaluation of the project.

# **Final Testing**

To test the program, I will follow the initial test plan however according to the changes I will not follow steps 5) 6) and 7)

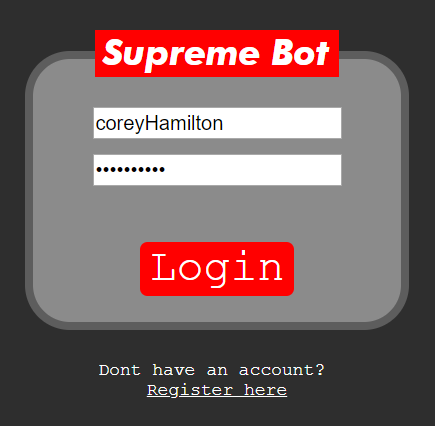
**Step 1:** **Step 2:**

****On loading the extension, I clicked “Register Here” to make a new account for this user.

After entering the initial details, I moved on to enter address and card details.

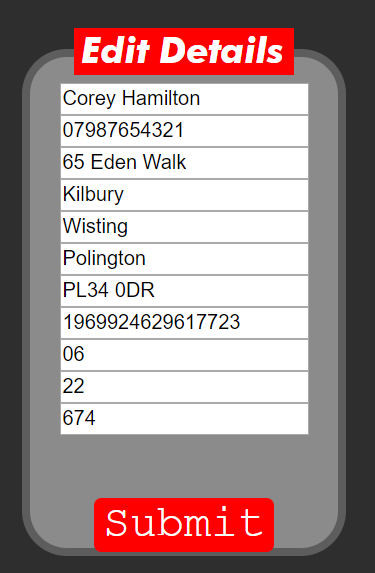
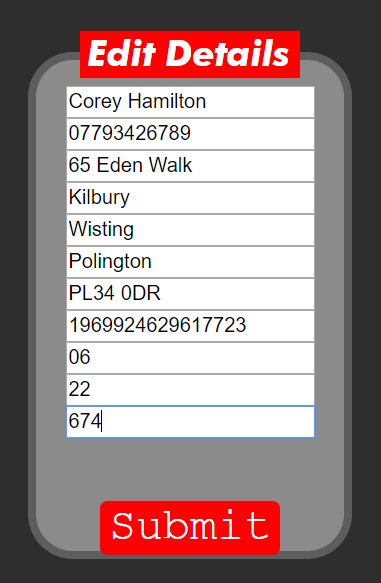
Validation was not tested in here as It was done previously, this is merely checking the register process is functional.

 All the details were added to the database as shown below.

**Step 3**

Once completed I was directed to the profile page, I exited the extension (by clicking off it). Then starting it up again I logged in with the newly made account , which successfully sent me to the profile page again





**Step 4:**

From the profile I clicked on “Edit Details” to show me the details I entered in step 2. I then went on to change the phone number as shown, and this was updated on the database after I pressed submit.



# **End User Testing**

I followed the initial plan to give the program to 2 people of the target audience with the following tasks:

1. Register a new user account with details.
2. Once registered, restart the program and login with details.
3. Update 2 items of personal details
4. Fill out a form for a desired item and submit it

Feedback from the 1st user was (more experienced user)…

* The interface was clear
* Inputs clearly marked
* However, did not like the username being case sensitive
* Commented that the sign up process was rather long, but acknowledged that it is needed for the program.

Feedback from 2nd user (beginner user)…

* 2nd user struggled with the inputs, misunderstanding the error messages and which input they are referring to.
* Also struggled when fitting the input to the range, error message did not specify the range.

# **Evaluation**

Looking back, this project has been fairly successful, considering the hurdles I had to overcome. At the start of the project I had aimed to make a fully functional supreme bot. With more time I believe this would be possible but limited to this time scale I was restricted when I had to take time to learn how to communicate between server and extension. At the time I was unsure whether I would be able to execute the process, I even tried to look for examples of other extensions on the store that access an external database, I couldn’t find any. This left me thinking it wasn’t possible, however I pursued the idea using google developer tools and support forums. Looking back, I can realise the reason that there are no extensions that access external servers, is because doing so is very insecure and is vulnerable to malicious attacks, but for the purpose of this project it would be appropriate as I have implemented minimal security precautions. So due to this loss of time I was forced to reduce the program to an interface that a user could register with details, login and edit details. I am happy with the program fulfilling the modified requirements, and it meets all the specifications.

The testing proved that the program was functional, even though it didn’t fulfil the initial project requirements, after the mentioned adjustments, it does its job. Giving the program to end users, some things came out that I would not have considered as a problem, or worth changing, however after talking to those users, certain things stood out. The error messages were not clear, to me they seemed clear as I was aware of the whole program, but to a complete beginner it was confusing. I would make changes to the error messages if I was to continue with the project.

One thing that I will do different is to check my Gantt chart templates, are printable before implementing them, as this time I realised when I went to print the Gantt charts that they did not appear as a full chart, instead just a current screen. This meant I had to screen shot every new screen , scrolling a bit more every time.

At the start of the project I planned to follow a large waterfall method, however this showed to be impractical as I used to much time designing components of the project, I was unsure if I was able to implement and test. I soon changed this to an AGILE methodology, where I applied the waterfall method to smaller chunks of the project, this proved to be much more practical, primarily because I was more motivated to meet my short-term goals and it is easier to set these short-term goals in time.

Over the course of the project I have waivered in effort levels, mainly due to lack of motivation at stages, but when faced with problem and a time goal, I was more than able to work to that deadline, with positive outcomes. I am proud of my persistence in the project as the extension caused problems, but I took time and effort to understand and overcome these problems without confirmation that it was possible.

# Appendix 2: Designs

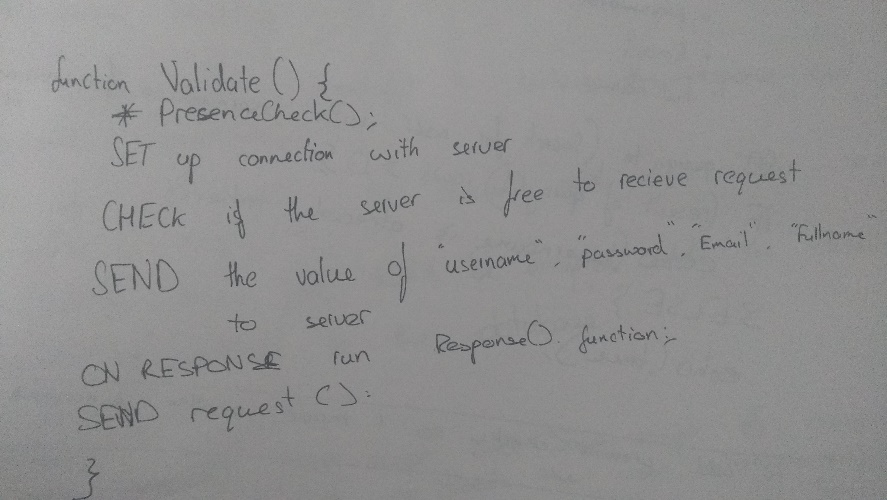
## 2.1 User Details Table

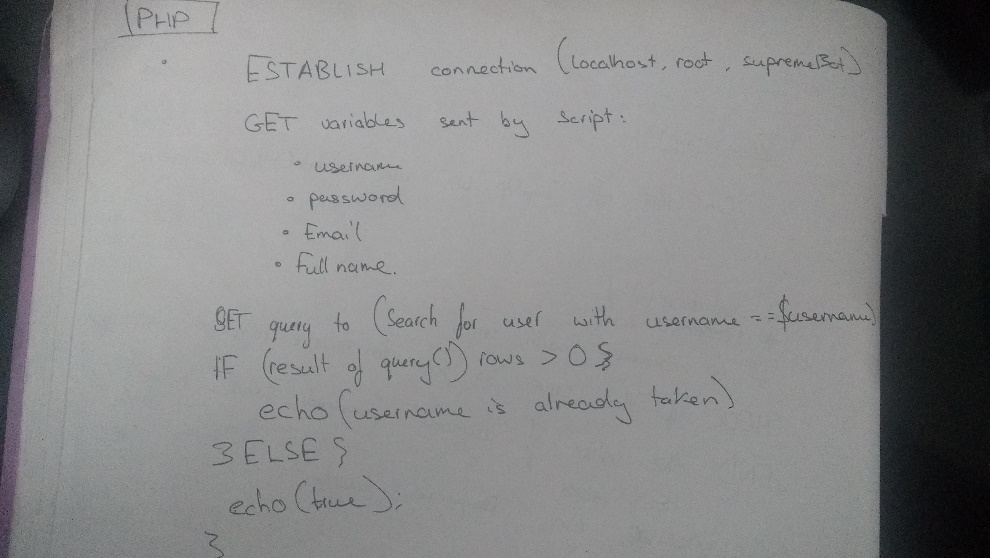
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Description | Data Type | Length | Required? | Validation |
| Username | Unique identifier of the user (Primary Key) | Text | 20 | Yes | Unique |
| Password | A secure password | Text | 30 | Yes | Length Check |
| Email | Needed for supreme checkout | Text | 30 | No | [\*@\*](mailto:*@*) |
| Full Name | Needed for supreme checkout | Text | 30 | No | Length Check |
| Telephone | Needed for supreme checkout | Text | 11 | No | Length Check |
| Address1 | Needed for supreme checkout | Text | 20 | No | Length Check |
| Address2 | Needed for supreme checkout | Text | 20 | No | Length Check |
| Address3 | Needed for supreme checkout | Text | 20 | No | Length Check |
| City | Needed for supreme checkout | Text | 20 | No | Length Check |
| Postcode | Needed for supreme checkout | Text | 7 | No | Length Check |
| Profile Picture | To display on profile page | Image |  | No | None |
| Card Number | All card details Encrypted storage? | Text | 16 | No | Length Check |
| Expiry Date | Encrypted? | Integer | 2 | No | Length Check |
| Expiry Year | Encrypted? | Integer | 4 | No | Length Check |
| Security code | Encrypted? | Integer | 3 | No | Length Check |

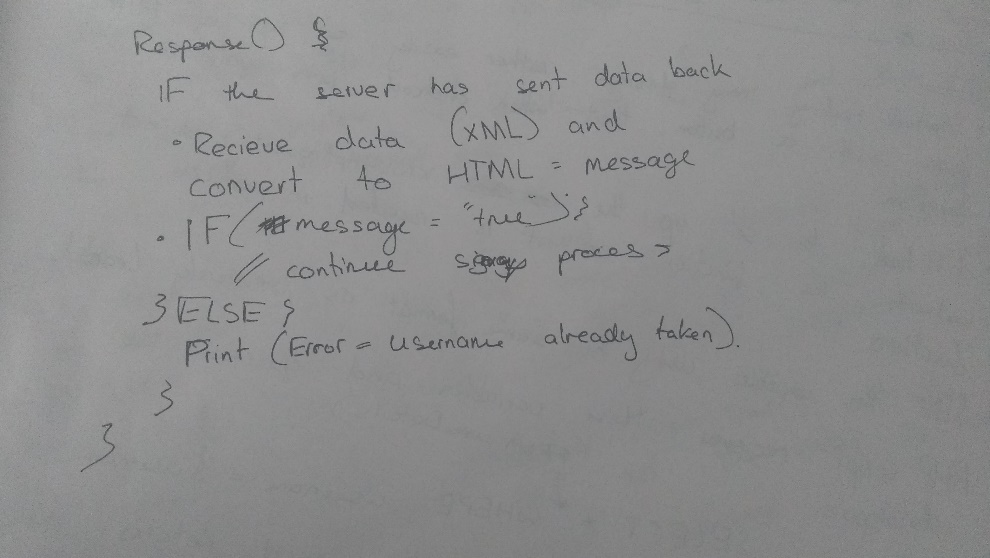
## 2.2 Order Details Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Description | Data Type | Length | Required? | Validation |
| Username | Foreign Key-from user details |  |  | Yes |  |
| Order Date | Will be made automatically after an order | Date |  |  | None |
| Item category | Will have been entered before the order | Value list |  | Yes | Set Values |
| Item keywords | Will have been entered before the order | Text | 30 | Yes | None |
| Item sell-out time | User can enter from Supreme community | Integer |  | No | None |
| Order success? | Successful, no checkout, Order Blocked, Other | Value list |  | No | Set Values |
| Retail Price | The price of item at retail, user enter… | Integer | 3 | No | None |
| Resale Price | Rough estimate of resale price of item | Integer | 3 | No | None |

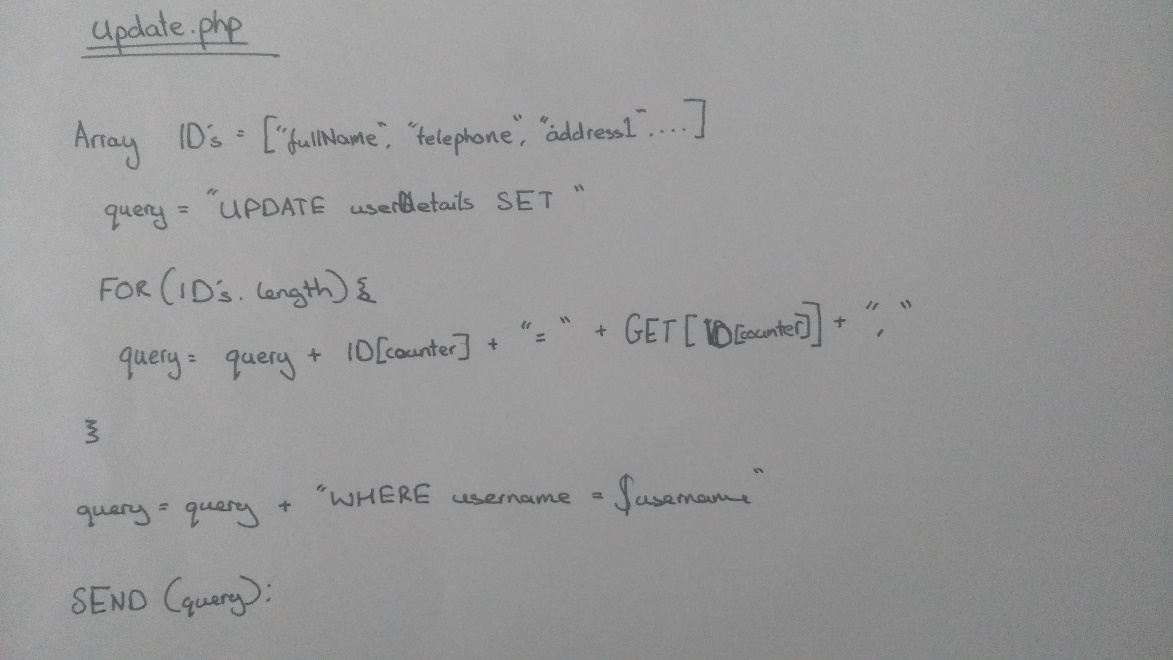
## 2.3 SignUp







## 2.4 Edit Details php script



# Appendix 3: Implementation

## 3.1 Popup

## 3.2 Database Screenshots

## 3.3 SignUp code

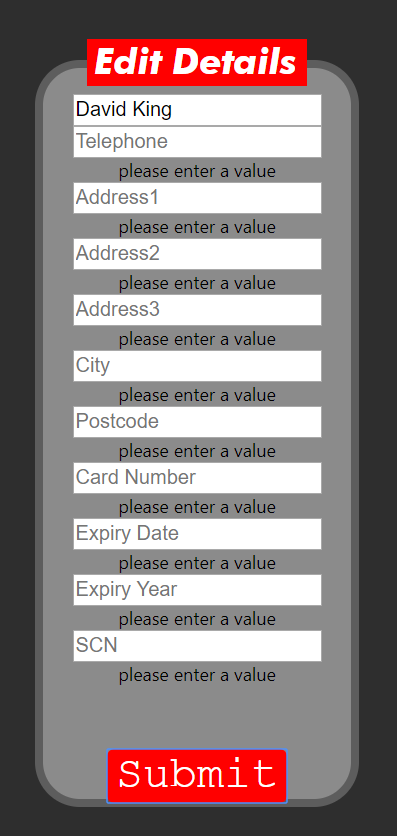
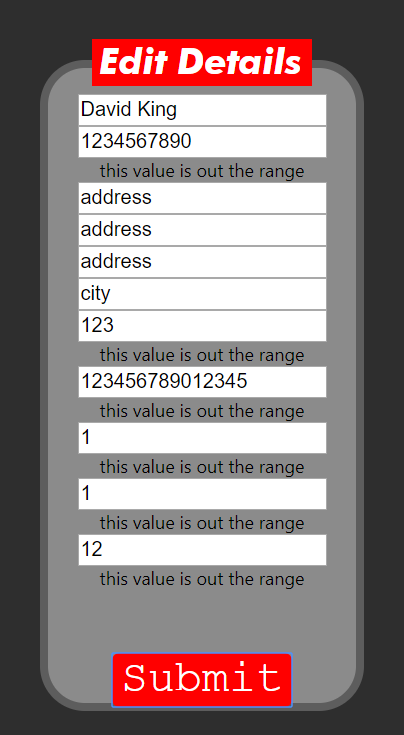
## 3.4 Edit Details Code

## 3.5 Login Code

## 3.6 Profile Code

## 3.7 Order Form code

# Appendix 4: Testing

EditDetails validation testing screenshots

## 4.2 Login Validation

