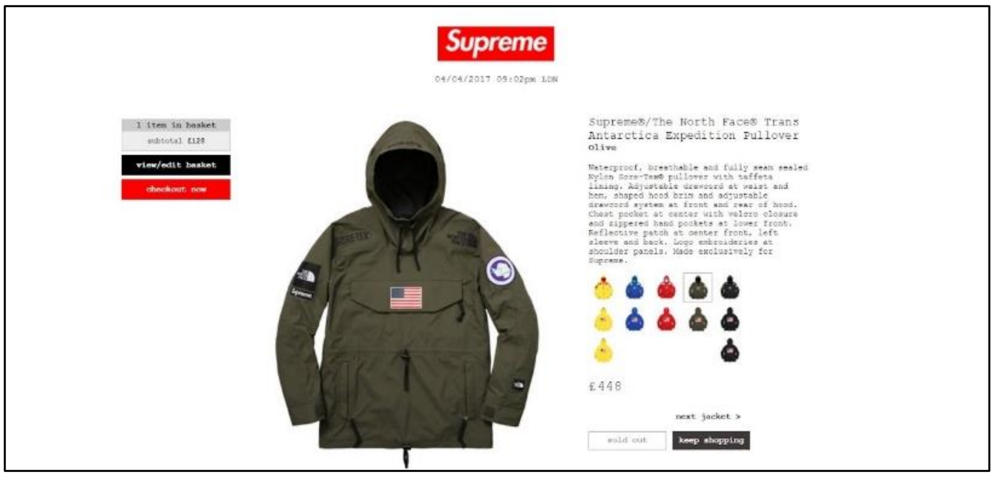
SUPREME BOT PROJECT

By Chris Abraham


04/04/2017 09:02pm LDN

Supreme®/The North Face® Trans Antarctica Expedition Pullover
olive

Waterproof, breathable and fully seam sealed Nylon Gore-Tex® pullover with taffeta lining. Adjustable drawcord at waist and hem, shaped hood brim and adjustable drawcord system at front and rear of hood. Chest pocket at center with velcro closure and zippered hand pockets at lower front. Reflective patch at center front, left sleeve and back. Logo embroideries at shoulder panels. Made exclusively for Supreme.

£448

[DATE]

[COMPANY NAME]

[Company address]

Contents

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, 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?

   1.1) If yes, was it a positive experience?

   1.2) If no, would you want to?
   what’s stopping you? (cost, reliability, security, other)

2. Would you trust a bot with all your credit card details?

3. How much would you honestly pay for a bot with 100% cop rate?
   (£0-£10) (£11-£50) (£51-£100)(£100+)

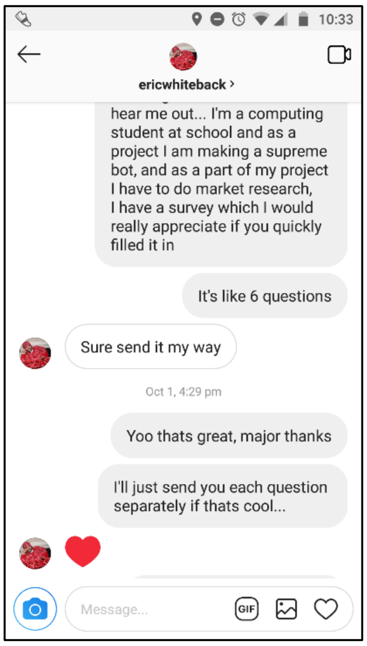
4. Name a feature you would want to add to a “dream supreme bot”.

5. Do you tend to buy multiple items in a drop or just focus on one?

6. 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



10:33

ericwhiteback ›

hear me out... I'm a computing student at school and as a project I am making a supreme bot, and as a part of my project I have to do market research, I have a survey which I would really appreciate if you quickly filled it in

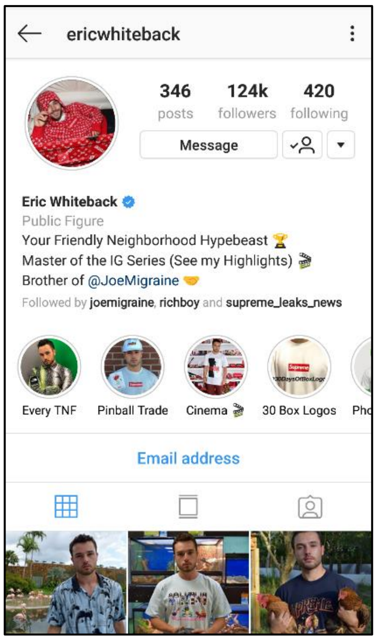
It's like 6 questions

Sure send it my way

Oct 1, 4:29 pm

Yoo thats great, major thanks

I'll just send you each question separately if thats cool...



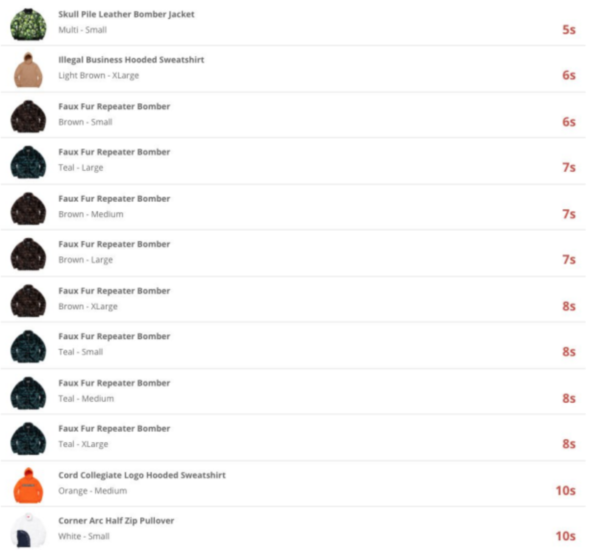
ericwhiteback

346	124k	420
posts	followers	following

Eric Whiteback
Public Figure
Your Friendly Neighborhood Hypebeast
Master of the IG Series (See my Highlights)
Brother of @JoeMigraine
Followed by joemigraine, richboy and supreme_leaks_news

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

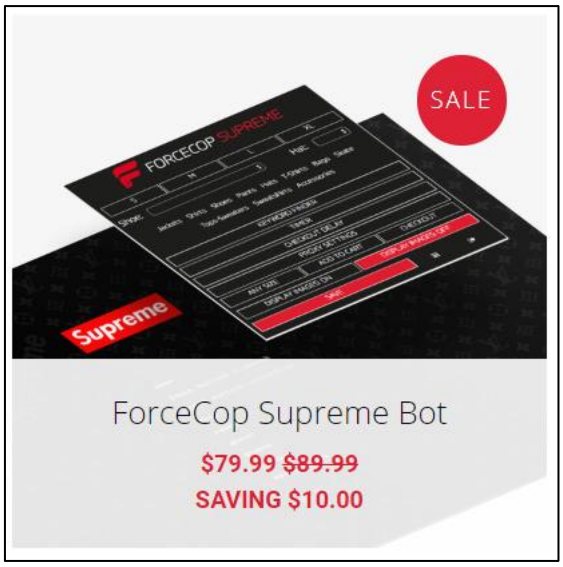
From chatting with Eric, others on social media and people I know I have a much better idea of the opinions of the target market. Out of the 10 I spoke to, 7 had never used a bot before and 3 had. Out of the 7 people who hadn't used a bot, it was mainly due to the price of them, and they didn't trust 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.



Skull Pile Leather Bomber Jacket Multi - Small	5s
Illegal Business Hooded Sweatshirt Light Brown - XLarge	6s
Faux Fur Repeater Bomber Brown - Small	6s
Faux Fur Repeater Bomber Teal - Large	7s
Faux Fur Repeater Bomber Brown - Medium	7s
Faux Fur Repeater Bomber Brown - Large	7s
Faux Fur Repeater Bomber Brown - XLarge	8s
Faux Fur Repeater Bomber Teal - Small	8s
Faux Fur Repeater Bomber Teal - Medium	8s
Faux Fur Repeater Bomber Teal - XLarge	8s
Cord Collegiate Logo Hooded Sweatshirt Orange - Medium	10s
Corner Arc Half Zip Pullover White - Small	10s

Research ctd

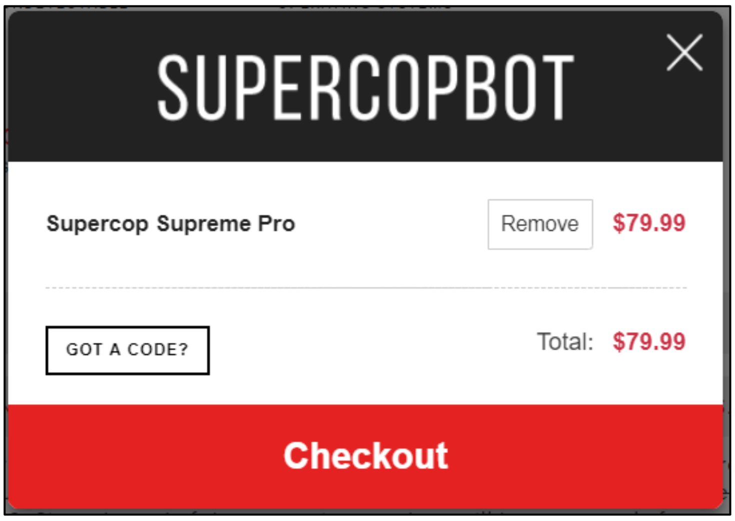
To further my research, I have looked at the bots that my target market said worked best. These are also the most popular bots to be used:



ForceCop Supreme Bot

$79.99 ~~$89.99~~

SAVING $10.00



Supercop Supreme Pro $79.99

Total: $79.99

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

See appendix 1*

Initial Requirement Specifications

Scope

1) Open the extension on any page and login
    a) Validate a username and password
        i) Ensure secure transmission of data
        ii) Allow a new user to make an account
2) Once logged in, allow user to edit personal details in a profile tab
    a) 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
    a) 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
    a) 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 ‘hypebeasts’ 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:
    i) username – davidKing119
    ii) password – preme_King34
2) Add details of checkout: (randomly generated details)
    i) Email – davidKing@gmail.com
    ii) Name – David King
    iii) Profile Pic – →
    iv) Telephone – 07123456789
    v) Address1 – 61 Bolven Walk
    vi) Address2 – Lotherton
    vii) Address3 - Tillington
    viii) City – Bradbury
    ix) Postcode – BT93 0YR
    x) Card Number – 5159927655617746
    xi) Expiry Date – 1/2021
    xii) SCN – 162

3) From the profile page, attempt to edit the user details to change the telephone to:
    i. 07987654321
4) Try to order an item that hasn't sold out on supreme website (order without time constraints)
5) Attempt to order an item on the drop (test against Supremes' quick sell-out times)
6) 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 5, 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.

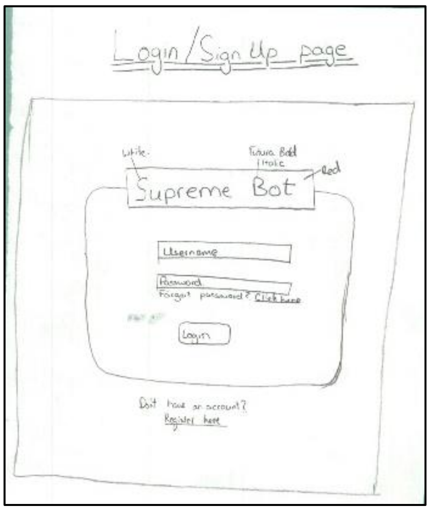
28/12/18: After changes to the project outline the respective changes were also made to the test plan, meaning in alpha testing step 4,5,6 are removed and in beta testing, step 5,6 are removed

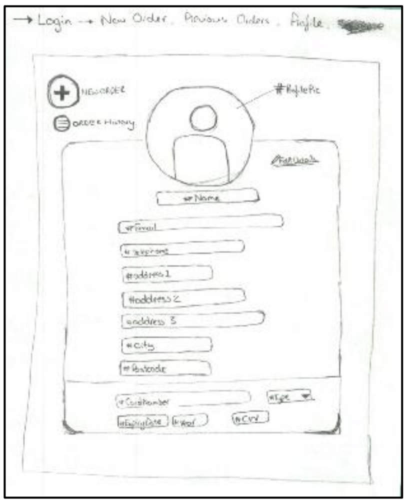
Design

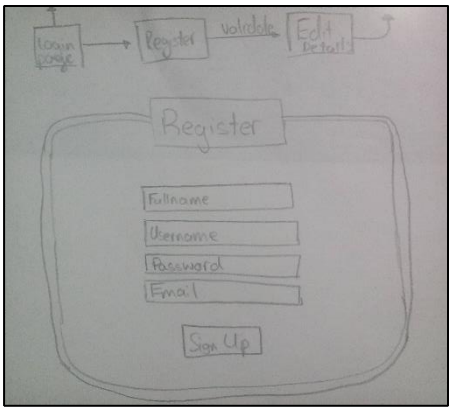
Interface

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

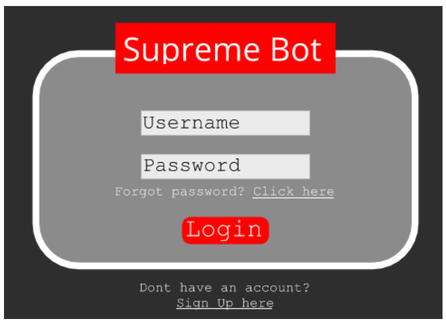
Based on the initial requirements I designed these wireframes:







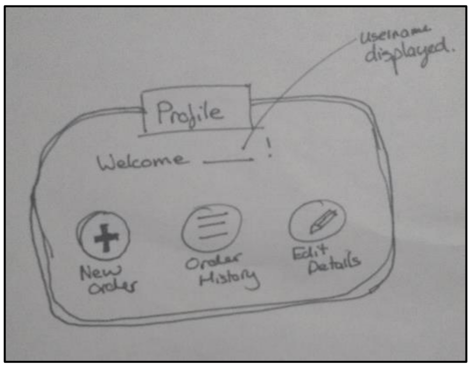
This was a simple and practical design for the extension, I tried to keep it small and compact to contain it to a corner of the user's window. I then used a graphics software called Gravit to make the login page with colours, to showcase the colour scheme and ensure it works.

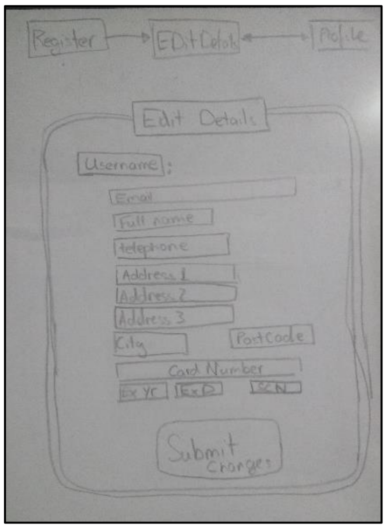


I only made a digital design for the login page to show the colour scheme. The colours, style and overall theme is transferable to other pages through classes in CSS

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

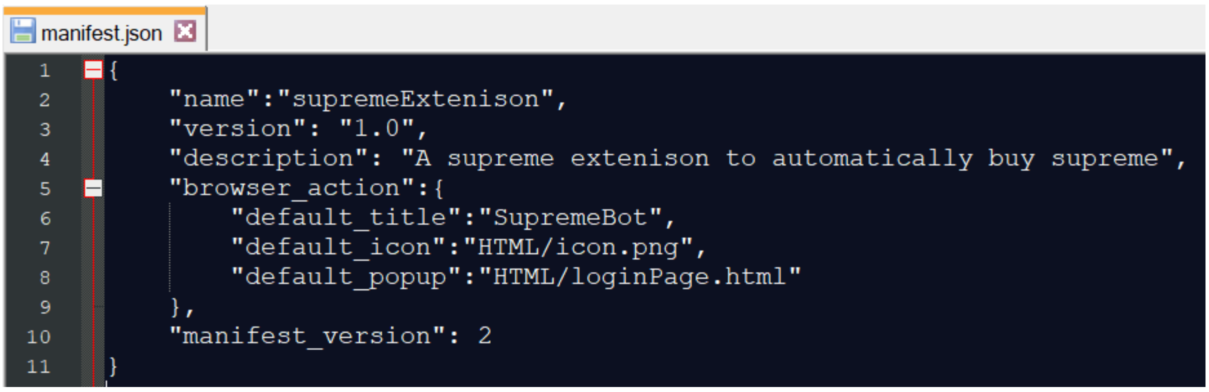
Permissions: to be confirmed (through research)

*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.

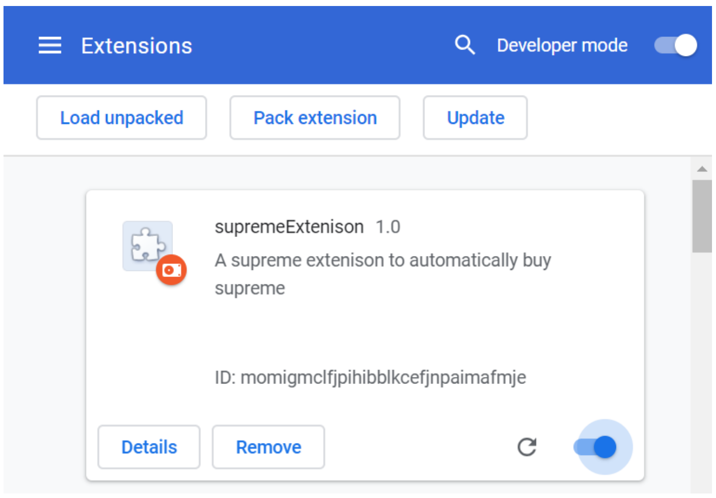
After a few simple syntax errors in the manifest caused by missing commas or speech marks, I was able to display the page as hoped and passed testing. *see Appendix 2.1 and 2.2 for the loginPage HTML and the associated CSS



manifest.json

```
{
    "name":"supremeExtenison",
    "version": "1.0",
    "description": "A supreme extenison to automatically buy supreme",
    "browser_action":{
        "default_title":"SupremeBot",
        "default_icon":"HTML/icon.png",
        "default_popup":"HTML/loginPage.html"
    },
    "manifest_version": 2
}
```

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

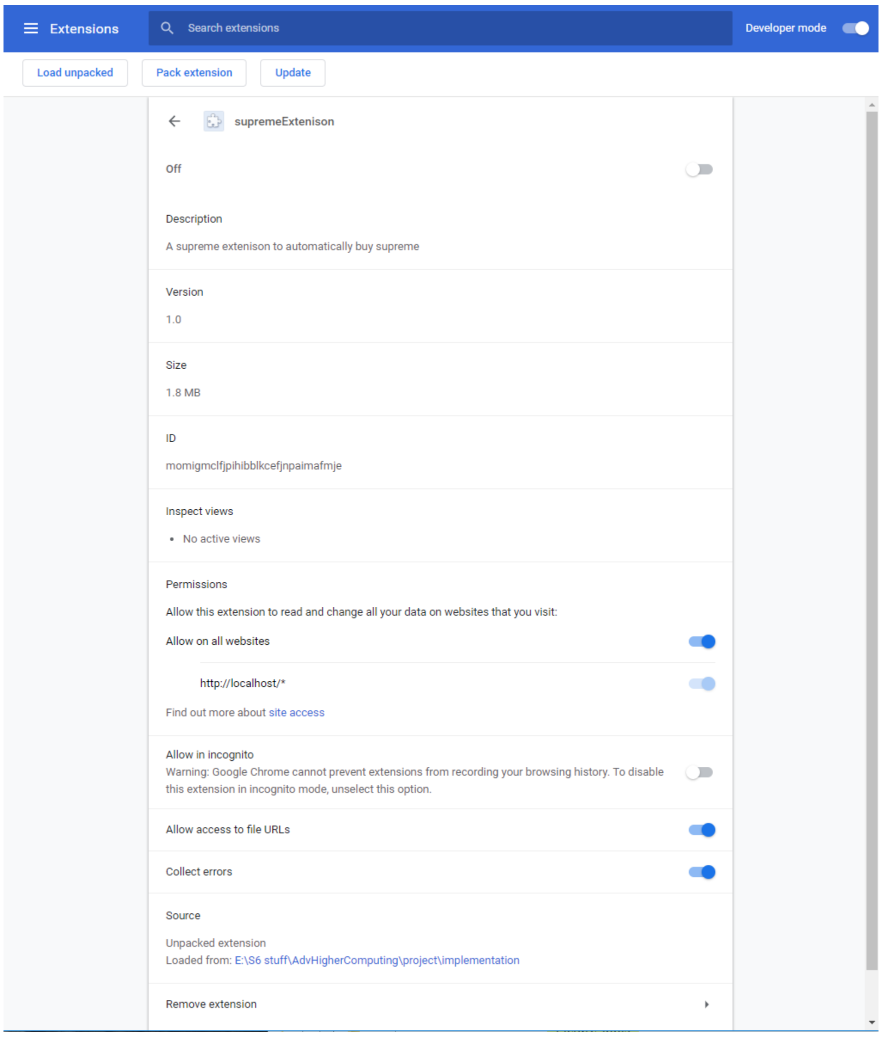


supremeExtenison 1.0

A supreme extenison to automatically buy supreme

ID: momigmclfjpihibblkcefjnpaimafmje

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



Extensions

Search extensions

Developer mode

Load unpacked | Pack extension | Update

supremeExtenison

Off

Description

A supreme extenison to automatically buy supreme

Version

1.0

Size

1.8 MB

ID

momigmclfjpihibblkcefjnpaimafmje

Inspect views

- No active views

Permissions

Allow this extension to read and change all your data on websites that you visit:

Allow on all websites

http://localhost/*

Find out more about site access

Allow in incognito

Warning: Google Chrome cannot prevent extensions from recording your browsing history. To disable this extension in incognito mode, unselect this option.

Allow access to file URLs

Collect errors

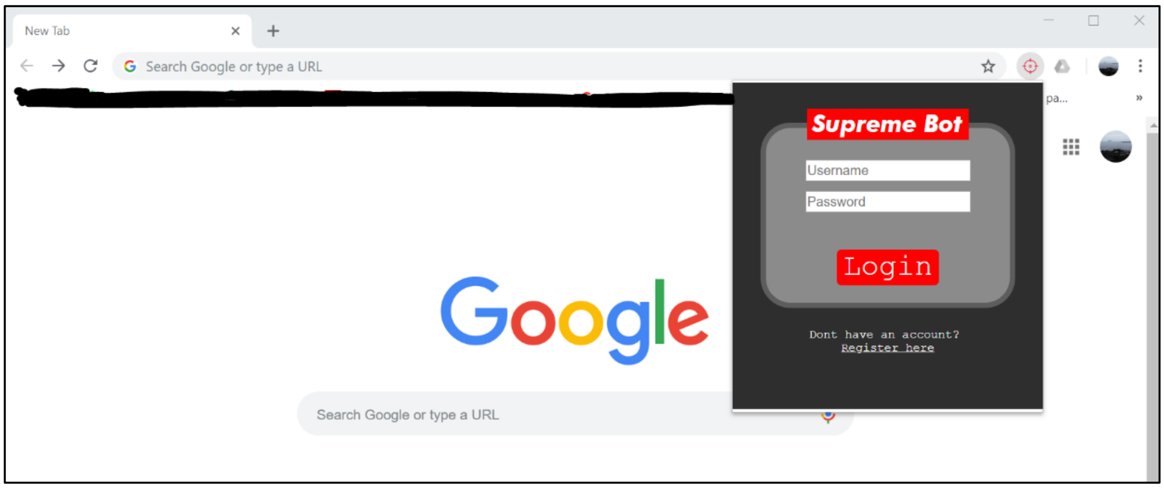
Source

Unpacked extension

Loaded from: E:\S6 stuff\AdvHigherComputing\project\implementation

Remove extension

When activated the icon appears in the chrome browser toolbar, revealing the popup



[redacted]

Supreme Bot

Username

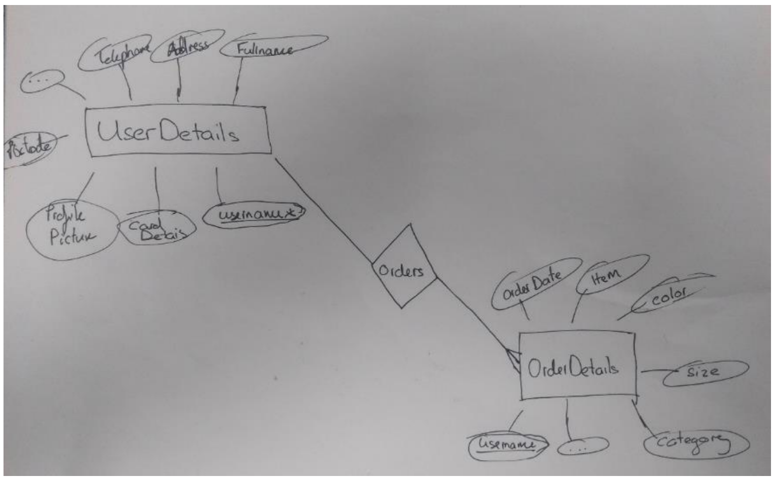
Password

Login

Dont have an account?
Register here

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 3) 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.

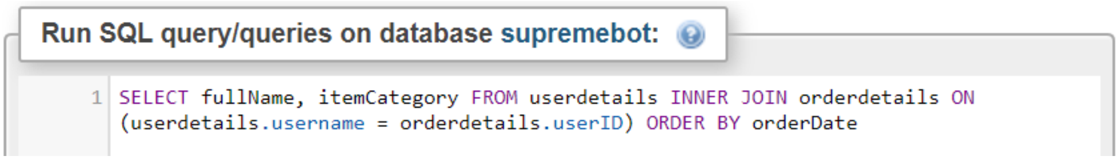
Fields and calculations	userDetails.fullName , orderDetails.itemCategory
Tables and queries	userDetails, orderDetails
Search criteria	none
Grouping	none
Sort Order	orderDate

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

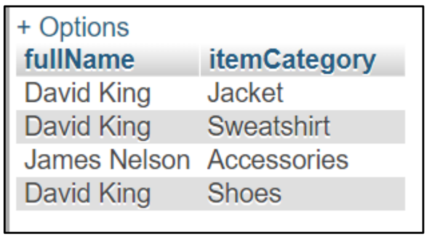
David King	Jacket
David King	Sweatshirt
James Nelson	Accessories
David King	Shoes

The evidence of implementation is all in Appendix 4

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



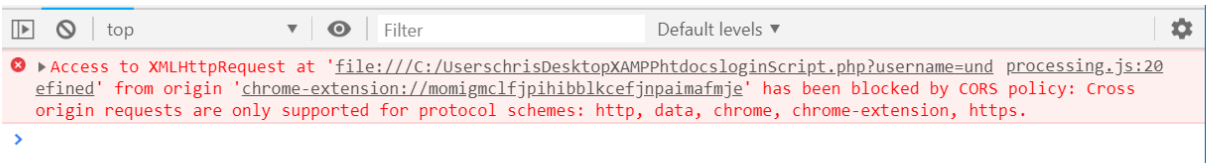
```
SELECT fullName, itemCategory FROM userdetails INNER JOIN orderdetails ON
(userdetails.username = orderdetails.userID) ORDER BY orderDate
```



fullName	itemCategory
David King	Jacket
David King	Sweatshirt
James Nelson	Accessories
David King	Shoes

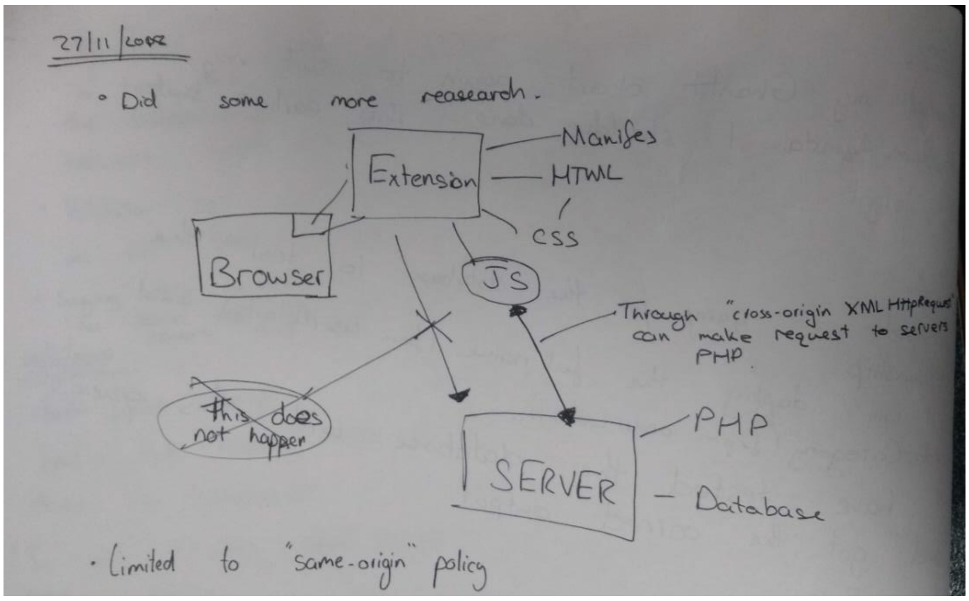
Linking the extension to server

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:

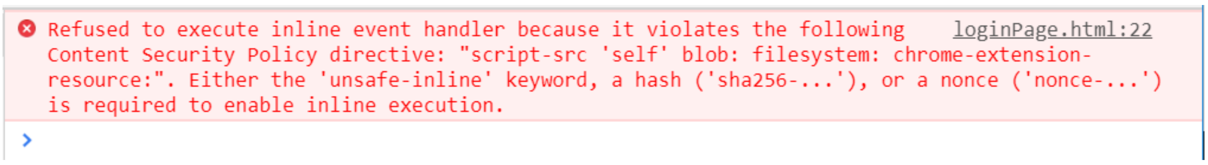


```
Access to XMLHttpRequest at 'file:///C:/UserschrisDesktopXAMPPhtdocsloginScript.php?username=und   processing.js:20
efined' from origin 'chrome-extension://momigmclfjpihibblkcefjnpaimafmje' has been blocked by CORS policy: Cross
origin requests are only supported for protocol schemes: http, data, chrome, chrome-extension, https.
```

Cross-origin resource sharing (CORS), is the policy restricting a site requesting data from external servers, for security reason, for extensions this policy 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)


27/11/2018
• Did some more reasearch.
Extension
Manifes
HTML
CSS
Browser
JS
Through "cross-origin XML HttpRequest" can make request to servers PHP.
~~This does not happen~~
SERVER
PHP
Database
• Limited to "same-origin" policy

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



```
Refused to execute inline event handler because it violates the following    loginPage.html:22
Content Security Policy directive: "script-src 'self' blob: filesystem: chrome-extension-
resource:". Either the 'unsafe-inline' keyword, a hash ('sha256-...'), or a nonce ('nonce-...')
is required to enable inline execution.
```

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.

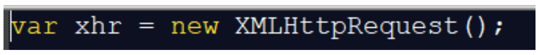
To test this aspect of the program I plan to just make sure that I can send and receive variables to the server from the extension, so for this test I will send a simple query:

"Enter the <u>username</u> of a user and return that users <u>fullName</u>"

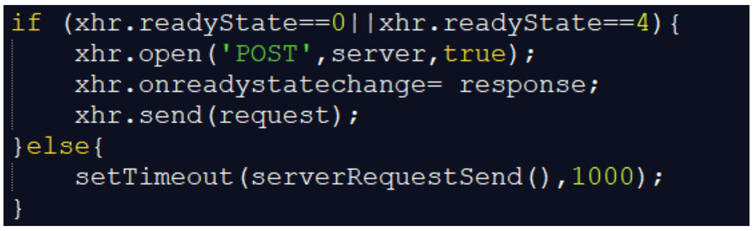
This step of the project took much longer than expected as I was trying to learn and use the external API without any previous knowledge, I used YouTube explanations and support forums to aid the process

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.

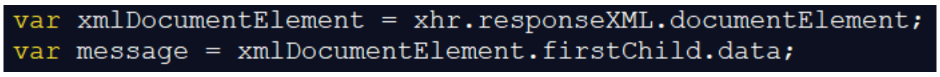


```
var xhr = new XMLHttpRequest();
```



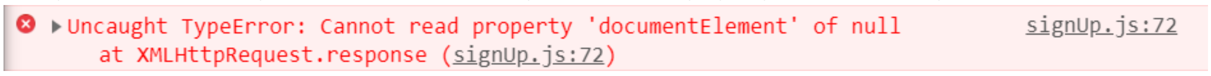
```
if (xhr.readyState==0||xhr.readyState==4){
    xhr.open('POST',server,true);
    xhr.onreadystatechange= response;
    xhr.send(request);
}else{
    setTimeout(serverRequestSend(),1000);
}
```

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.



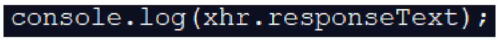
```
var xmlDocumentElement = xhr.responseXML.documentElement;
var message = xmlDocumentElement.firstChild.data;
```

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:



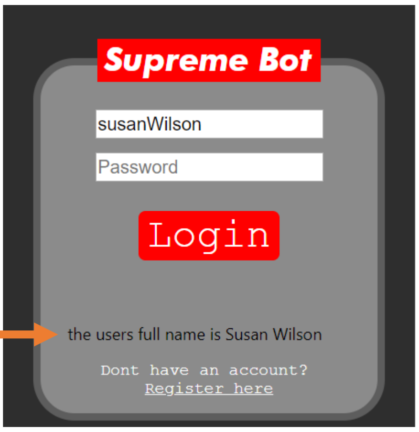
```
✖ ▶Uncaught TypeError: Cannot read property 'documentElement' of null        signUp.js:72
      at XMLHttpRequest.response (signUp.js:72)
```

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.



```
console.log(xhr.responseText);
```

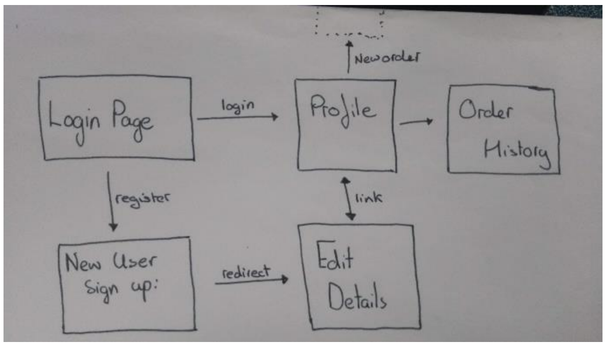
After a lot of time and effort I was able to produce the expected result, and I was familiar with using XMLHttpRequest to process requests.



the users full name is Susan Wilson

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.



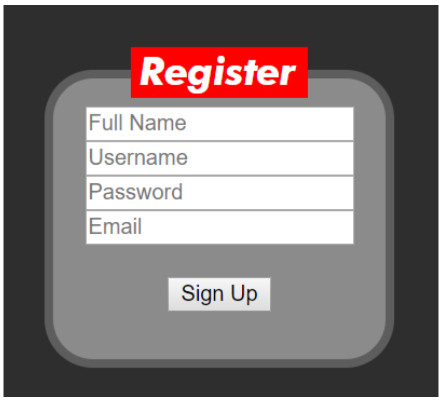
In the project specifications I have mentioned that I will allow users to edit their details when logged on. It makes sense to reuse the sign-up form to edit details. When a user wants to register they enter their:

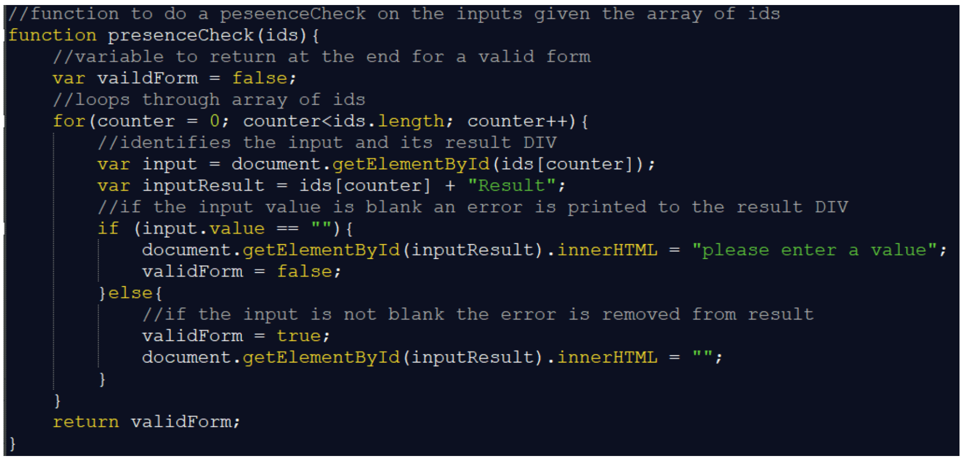
- Full Name
- Desired username
- Desired Password
- Email

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.

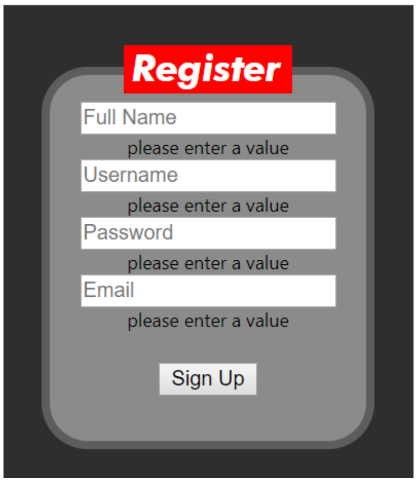
*See 10/12/18 - 12/12/18 for more in-depth design (pseudocode functions) of the sign up process.

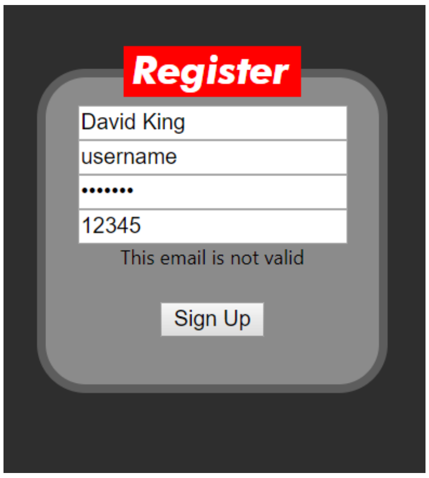
To test the sign up process of the program I planned to register a new user and for each input, enter a normal, extreme, exceptional value, to ensure the data sent to the server is valid, check if the username is not already taken, and when sent the data should be added to the database, to the respective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



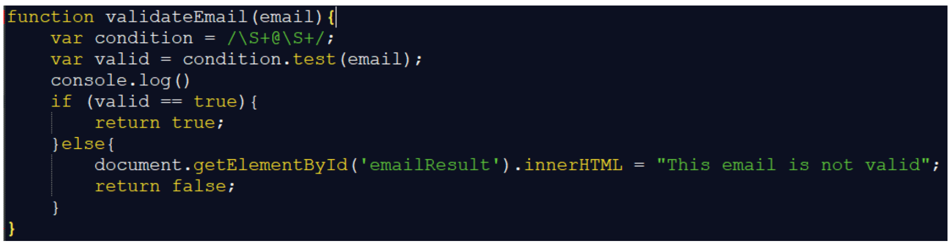


```
//function to do a peseenceCheck on the inputs given the array of ids
function presenceCheck(ids){
    //variable to return at the end for a valid form
    var vaildForm = false;
    //loops through array of ids
    for(counter = 0; counter<ids.length; counter++){
        //identifies the input and its result DIV
        var input = document.getElementById(ids[counter]);
        var inputResult = ids[counter] + "Result";
        //if the input value is blank an error is printed to the result DIV
        if (input.value == ""){
            document.getElementById(inputResult).innerHTML = "please enter a value";
            validForm = false;
        }else{
            //if the input is not blank the error is removed from result
            validForm = true;
            document.getElementById(inputResult).innerHTML = "";
        }
    }
    return validForm;
}
```

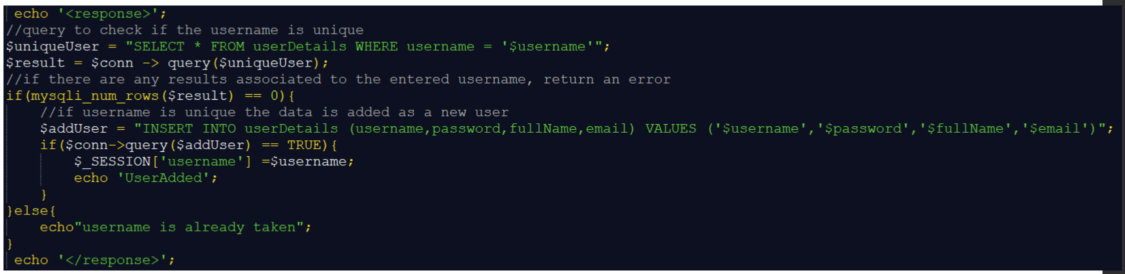




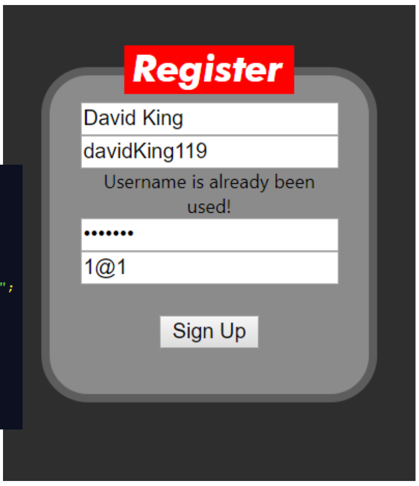
David King
username
•••••••
12345
This email is not valid



```
function validateEmail(email){
    var condition = /\S+@\S+/;
    var valid = condition.test(email);
    console.log()
    if (valid == true){
        return true;
    }else{
        document.getElementById('emailResult').innerHTML = "This email is not valid";
        return false;
    }
}
```



```
 echo '<response>';
//query to check if the username is unique
$uniqueUser = "SELECT * FROM userDetails WHERE username = '$username'";
$result = $conn -> query($uniqueUser);
//if there are any results associated to the entered username, return an error
if(mysqli_num_rows($result) == 0){
    //if username is unique the data is added as a new user
    $addUser = "INSERT INTO userDetails (username,password,fullName,email) VALUES ('$username','$password','$fullName','$email')";
    if($conn->query($addUser) == TRUE){
        $_SESSION['username'] =$username;
        echo 'UserAdded';
    }
}else{
    echo"username is already taken";
}
 echo '</response>';
```



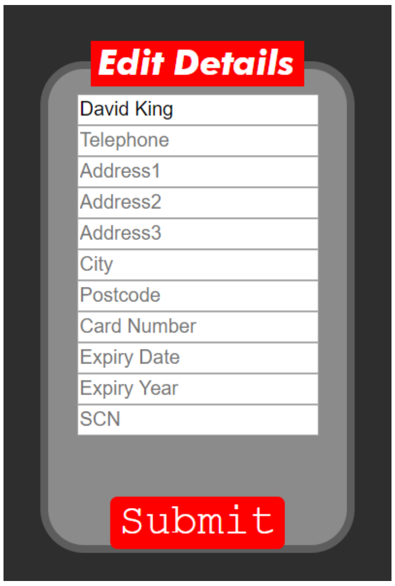
David King
davidKing119
Username is already been used!
•••••••
1@1

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 users username, allows the program to

where all changeable stored details on the user will be displayed, at this point it is only the full name, and the rest is under more validation.


Edit Details

David King

Telephone

Address1

Address2

Address3

City

Postcode

Card Number

Expiry Date

Expiry Year

SCN

See Appendix 5 for more testing screenshots

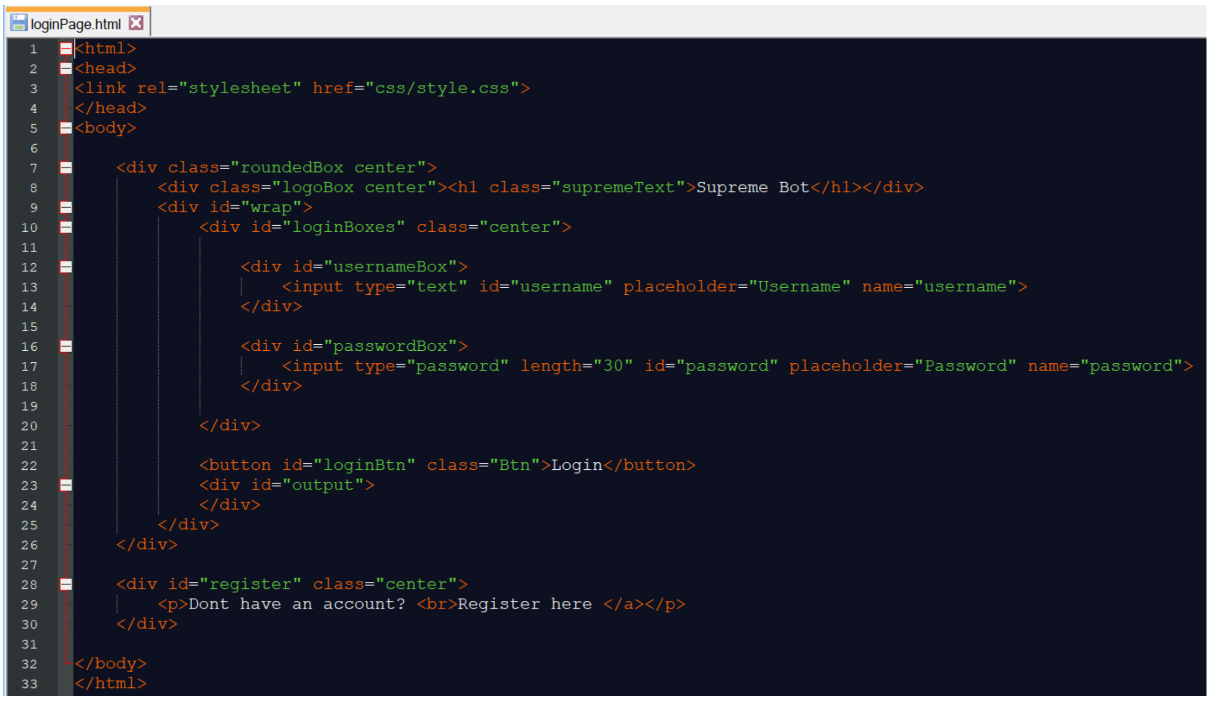
Using the

Appendix 1

*Gantt Charts

Appendix 2

2.1



loginPage.html

```html
<html>
<head>
<link rel="stylesheet" href="css/style.css">
</head>
<body>

    <div class="roundedBox center">
        <div class="logoBox center"><h1 class="supremeText">Supreme Bot</h1></div>
        <div id="wrap">
            <div id="loginBoxes" class="center">

                <div id="usernameBox">
                    <input type="text" id="username" placeholder="Username" name="username">
                </div>

                <div id="passwordBox">
                    <input type="password" length="30" id="password" placeholder="Password" name="password">
                </div>

            </div>

            <button id="loginBtn" class="Btn">Login</button>
            <div id="output">
            </div>
        </div>
    </div>

    <div id="register" class="center">
        <p>Dont have an account? <br>Register here </a></p>
    </div>

</body>
</html>
```

Appendix 3

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	*@*
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 Encryted 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

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?	Succesfull, 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

Appendix 4

Database implementation evidence (screen shots)

Appendix 5

EditDetails validation testing screenshots