The scenario:

Mr. Xue owns a medium sized business that primarily sells products such as LED lights, sockets, multi socket extension and a range of other electrical products which he imports overseas and distributes them to other smaller businesses around Indonesia.

Currently, the client works with many businesses and the process of ordering to delivery is done manually. The process begins with customers placing an order, then the personnel in charge of taking orders will check availability from the warehouse and contact the customer to confirm the order. The policy of the business is that delivery will only be processed once the customers have transferred the payment. The office branch then has to contact the warehouse once again to send the stock quantity and product type so that the order can be processed.

The problem with the system as described by the client, "*As the number of our clients grows, it is becoming much more difficult to keep track and organize the clients and their orders.*" The client also mentioned that he works with many customers, as a result, when there are a lot of orders being placed, the possibility of mistakes, errors or even miscommunication between all parties involved in the process of ordering to delivery increases and is very inefficient.

Note: *this section was written based on consultation with the client, refer to the appendix A for the transcript of the first interview*)

Scenario word count: 210

Rationale for the proposed solution:

I proposed th idea of a web application to my client that allows different access to the users depending on their status. This allows the customers to place orders and the workers (office/warehous) to check the details of the purchases so that it can be processed. The website will be linked to a database that stores all the necessary information requested by the client.

After consulting my client, I discovered that he works with a lot of smaller businesses which led me to decide on a web application as a more suitable solution. As opposed to an app which requires downloading and is dependent on compatibility with the device, a website is much more convenient and flexible which allows my client's customers and workers to access the platform using different devices.

The website will be structured and styled using HTML and CSS to build the basic framework for the website and to create an interface that is easy to navigate and understand. It is also suitable for websites as they are compatible with most browsers that are used today[1]. Paired

[1] "Handling Common HTML and CSS Problems - Learn Web Development: MDN," Learn web development | MDN, accessed June 28, 2021, https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/HTML_and_CSS.

with Javascript to create an interactive website that enhances user experiences. As well as, backend development which primarily uses node.js as it doesn't require conversion between JSON and binary models[2] and it is asynchronous event driven improving concurrent request handling[3]. MySQL will be the choice of database due to its excellent data security[4] compared to other database softwares which is essential as my client handles confidential information.

Note: *this section was written based on consultation with the client, refer to the appendix A for the transcript of the second interview*)

Rationale word count: 249

Success criteria:

1. Authentication features that limits access of the users based on their roles.
2. All passwords should be hashed
3. User is able to reset password.
4. Admin should be able to view and manipulate the analytics.
5. Admin should be able to accept an order that is waiting for approval.
6. Allow admin to sort and filter the table records.
7. View relevant details about each record/order
8. Admin can search the users and filter records based on names and email.
9. Admin can create a new user
10. Admin can edit and delete users
11. Admin can view and update all the pending transactions that need to be reviewed.
12. Admin can add product and edit the details.
13. View and update relevant records to the logistic department.
14. View all the available products and search them by category and name.
15. Users should be able to view a product and change the variant/model of the product during browsing.
16. Allow the user to add items to cart and create a new order.
17. Users should be able to upload proof of payment for certain orders.

---

[2] Editor, "The Good and the Bad of Node.js Web App Development," AltexSoft (AltexSoft, February 27, 2020), https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/.
[3] Voidcanvas, "Node.js - Reasons to Use, Pros and Cons, Best Practices!," voidcanvas, accessed June 28, 2021, https://www.voidcanvas.com/describing-node-js/.
[4] MySQL Advantages and disadvantages, accessed June 28, 2021, https://www.techstrikers.com/MySQL/advantages-and-disadvantages-of-mysql.php.

Bibliography

Editor. “The Good and the Bad of Node.js Web App Development.” AltexSoft. AltexSoft, February 27, 2020. https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/.

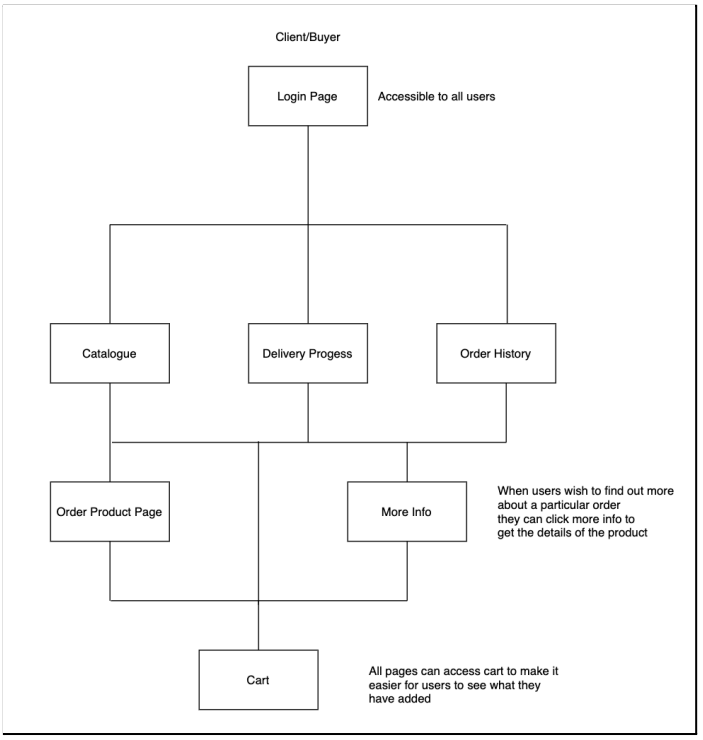
“Handling Common HTML and CSS Problems - Learn Web Development: MDN.” Learn web development | MDN. Accessed June 28, 2021. https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/HTML_and_CSS.

MySQL Advantages and disadvantages. Accessed June 28, 2021. https://www.techstrikers.com/MySQL/advantages-and-disadvantages-of-mysql.php.

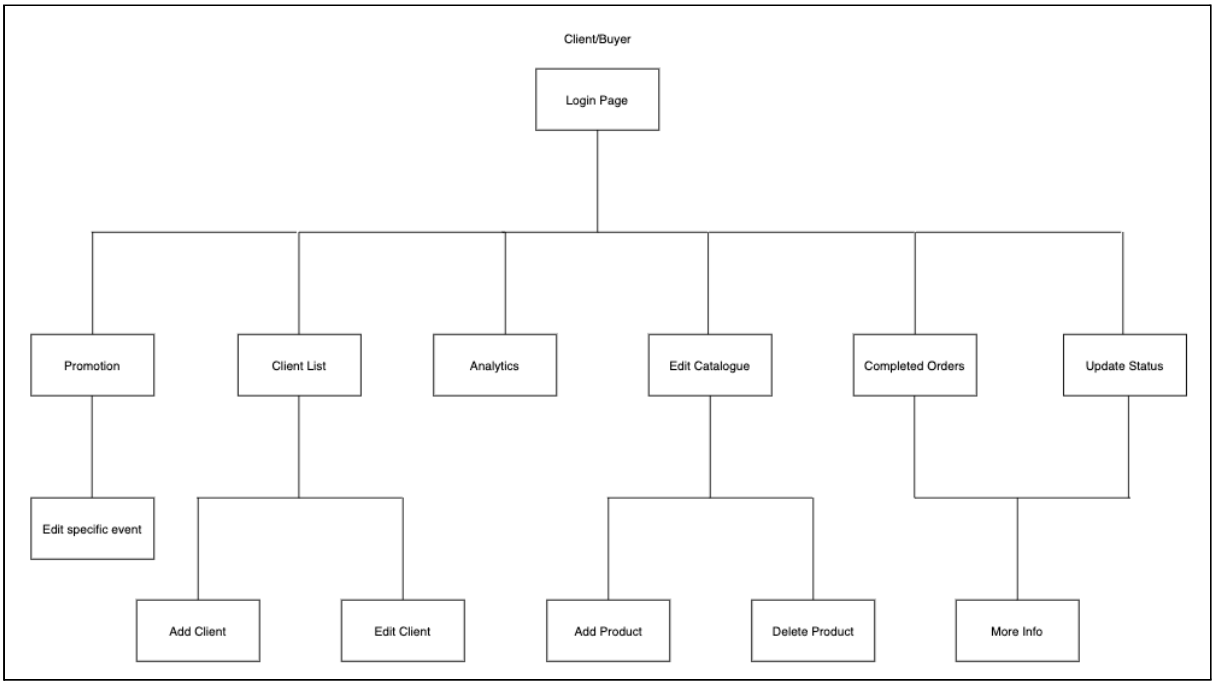
Voidcanvas. “Node.js - Reasons to Use, Pros and Cons, Best Practices!” voidcanvas. Accessed June 28, 2021. https://www.voidcanvas.com/describing-node-js/.

Pages Overview

Client Side:



Employees:

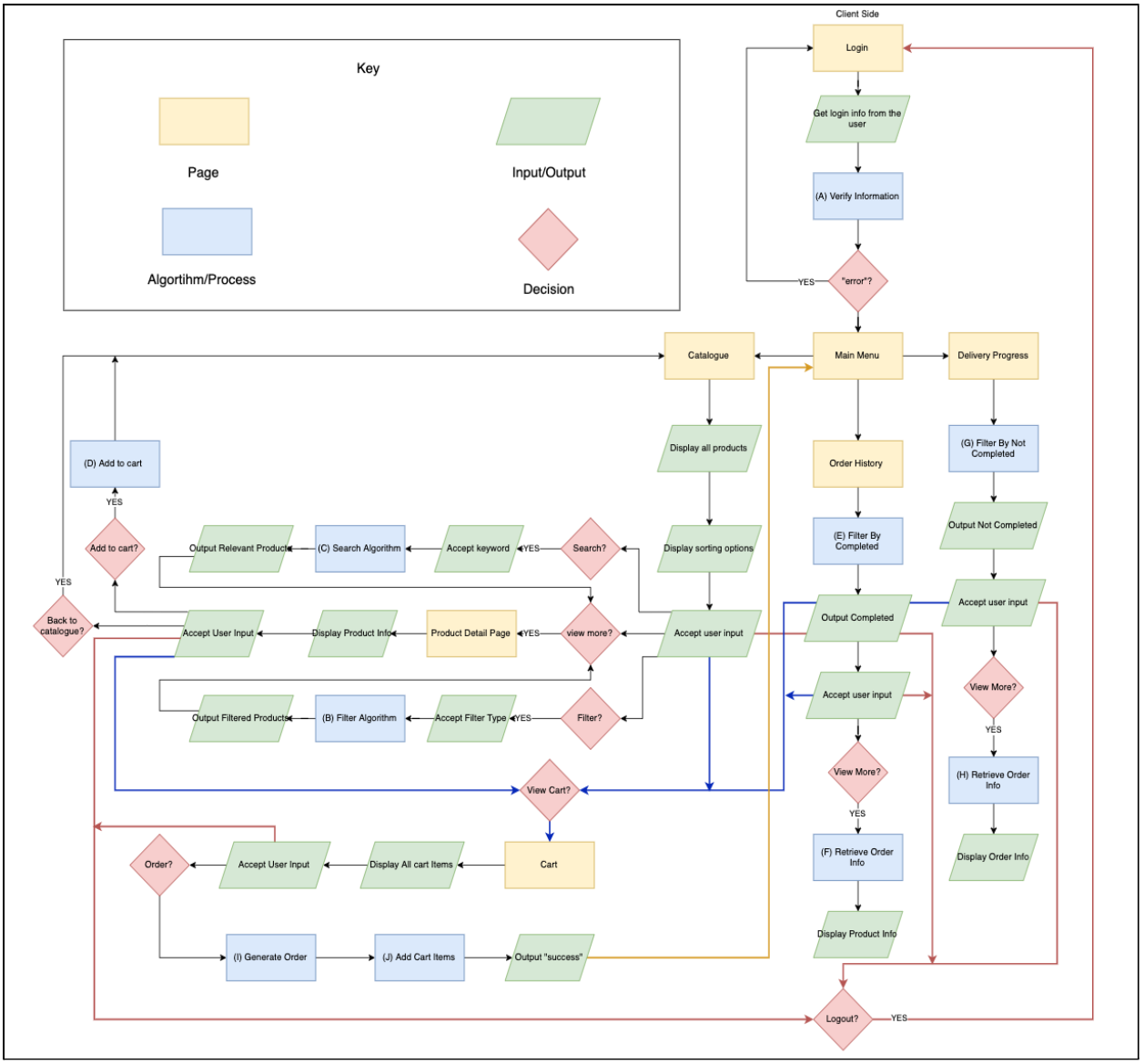


System Flowchart

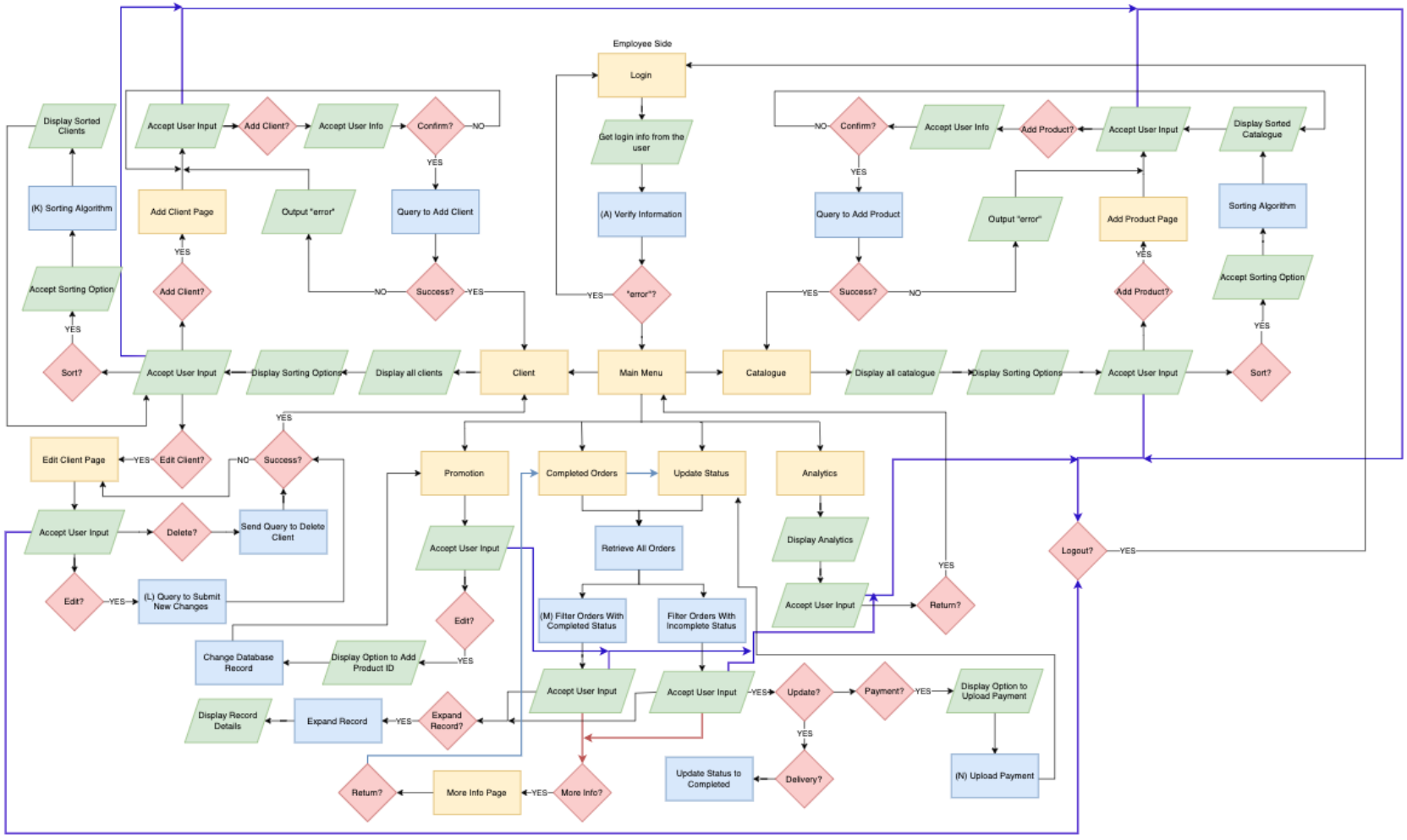
*Subjected to vary from final product

Client/Buyer Side:

*Some boxes are annotated with a letter as it will be described in depth

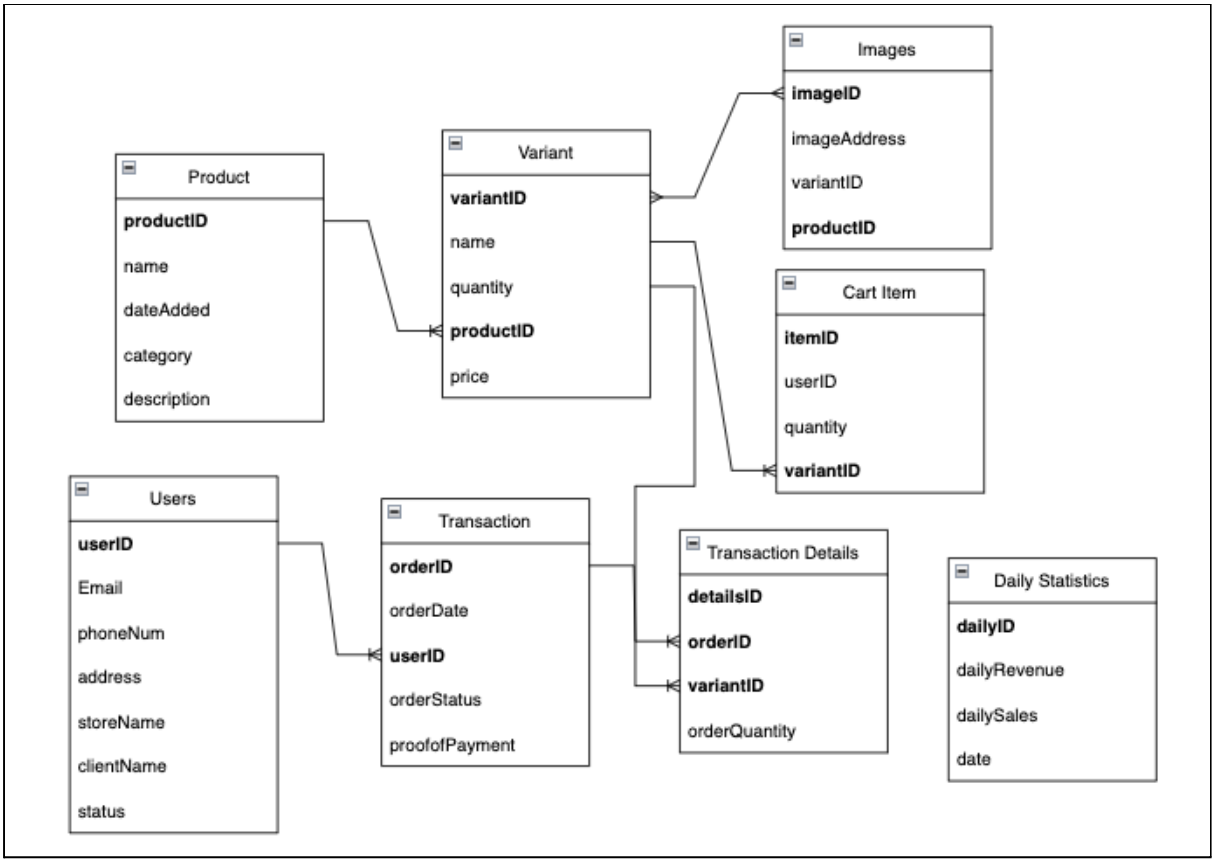


Employee Side



Database Design

Entity-relationship overview/diagram:



Product table:

The database includes information about the products provided by the business

Field name	Data type	Description
productID (Primary Key)	Integers	To identify the name and type of product
Name	String	Includes the name of the product
Date added to the database	Date format	Identifies when the product was introduced/added to the company in mm/dd/yy format
Category	Integer	Cateogry of the product
Description	Integer	Description of the product

Sample				
Product ID	name	dateAdded	category	description
291	LED light	23/02/2021	Light	-
292	Glue gun	25/05/2021	Adhesive	-

Variant table:

For each products there are different models, such as sizes or colours and a separate table is needed.

Field name	Data type	Description
variantID (Primary Key)	Integers	To identify the record for each variant registered
Name	String	Includes the name of the product
Quantity	Integer	How much quantity of the variant is available.
productID (Foreign Key)	Integer	Identify which product the variant belongs to.
price	integer	Stores the price of the variant /pcs

Sample				
Variant ID	Name	Quantity	productID	price (Rp)
34	5W	503	291	32000
35	10W	234	291	35000
36	small	3556	292	13000
37	large	4324	292	16000

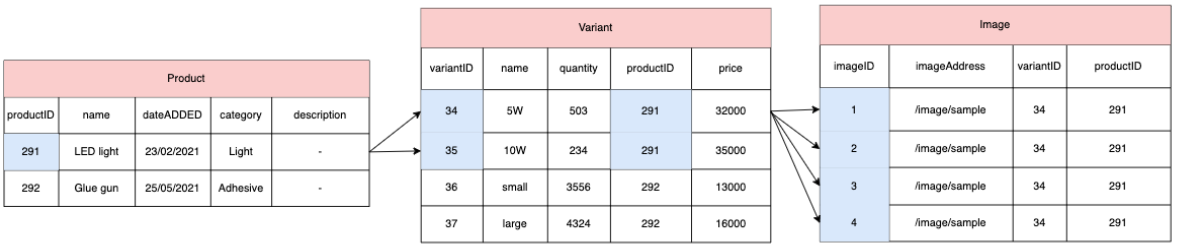
Images table:

Each variant will include multiple images, thus another table is required.

Field name	Data type	Description
imageID (Primary Key)	Integers	Identify the record of the image
Image address	String	Stores the address of the image or the image itself depending on development
variantID (Foreign Key)	Integer	Identifies which variant the image belongs to.
productID (Foreign Key)	Integer	Identifies which product the image belongs to.

Sample			
Image ID	imageAddress or image	variantID	productID
1	/image/sample	34	291
2	/image/sample	34	291
3	/image/sample	34	291
4	/image/sample	35	291

Visulation for product, variant, and image table:



User table:

Stores information about the clients of the business to successfully carry out the delivery process, authentication, recording transcripts. Certain details are left blank for logistics or admins.

Field name	Data type	Description
userID (Primary Key)	Integers	Uniquely identifies the client
Email address	String	Email address of the client
Phone number	Integer	Contact info/phone number of the client
Address	String	The warehouse can identify where to deliver the products
Name of store	String	To identify the client's store in real life
Name of client	String	Determines the recipient of the delivery
Status	Boolean	Determines whether the user is a client, employee or admin during the login process.

Sample						
userID	Email	Phone	Address	Store Name	Client Name	Status
45	johnsmith@gmail.com	08182921	...	Terang Jaya	John Smith	User
46	willroger@gmail.com	08123792	...	Pertama Jaya	Will Roger	User
75	admin@gmail.com	-	-	-	-	Admin
86	logistics@gmail.com	-	-	-	-	Logistics

Transactions Database:

Transactions and products have many to many relationships. Where one transaction can contain multiple products and a product can be included in many transactions. To design the database, a bridge table is used to clearly identify the relationships between the two.

Transactions Table: This table gives the details of the transaction.

Field name	Data type	Description
orderID (Primary Key)	Integer	Uniquely identifies the order
orderedDate	Date format	Identifies when the order was placed or requested by the user
userID (Foreign Key)	Integer	Identify who is placing the order so necessary information can be retrieved
Order status	String	The order can either be “payment”, “delivering”, “completed”
proofOfPayment	String	Contains the image/address of the image of the payment evidence

Sample				
orderID	orderDate	userID	orderStatus	proofOfPayment
46	12/12/21	1	Waiting Approval	-
47	17/12/21	1	Waiting Payment	-
48	20/12/21	1	Payment Uploaded	Address of payment proof

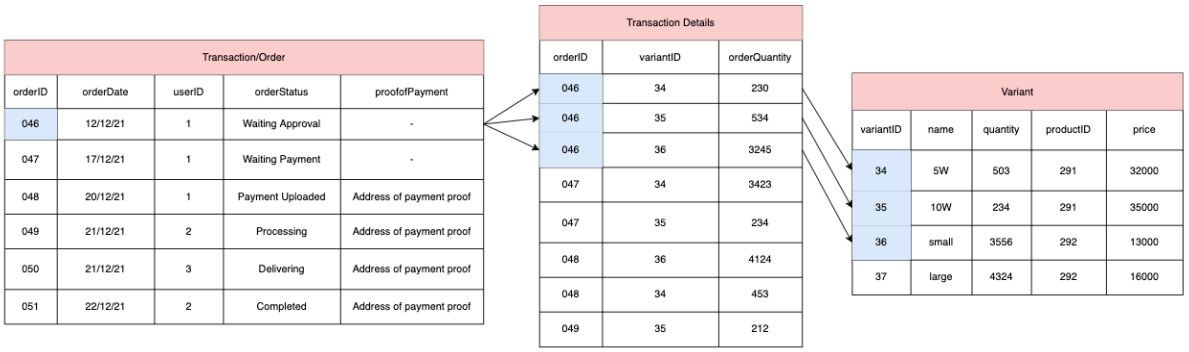
49	21/12/21	2	Processing	Address of payment proof
50	21/12/21	3	Delivering	Address of payment proof
51	22/12/21	2	Completed	Address of payment proof

Transaction_Details Table:

Field name	Data type	Description
orderID (Foreign Key)	Integer	Uniquely identifies the order
variantID (Foreign Key)	Integer	Identifies what type of product it is.
orderQuantity	Integer	Determines how the quantity of the products ordered

Sample		
orderID	variantID	orderQuantity
46	34	230
46	35	534
46	36	3245
47	34	3423
47	35	234
48	36	4124
48	34	453
49	35	212

Transaction Database Overview:



Cart Database:

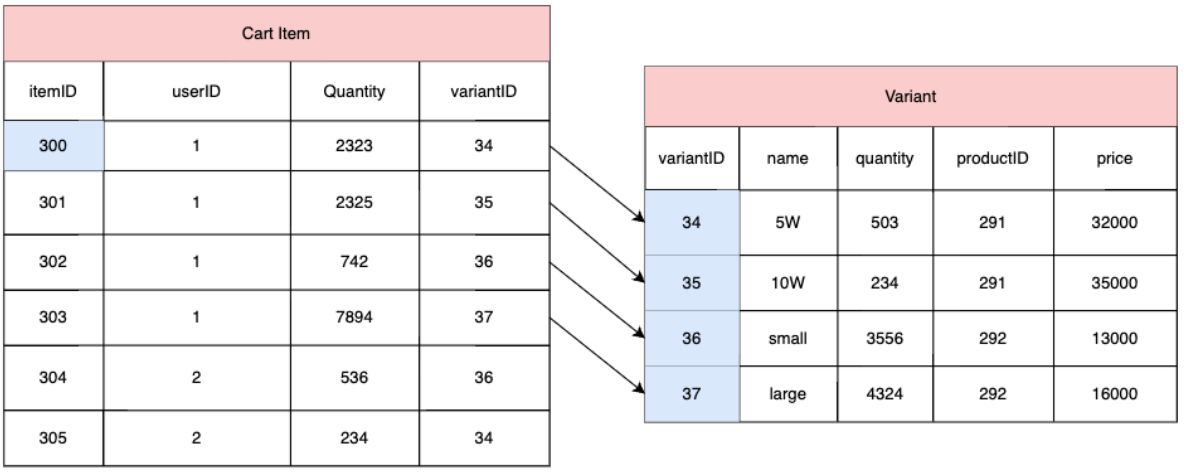
The records within cart item will be deleted when the user places an order.

Cart_Item Table:

Field name	Data type	Description
cartItemID (Primary Key)	Integer	Uniquely identifies the cart
variantID (Foreign Key)	Integer	One to one relationship between cart and user
userID	Integer	Identifies the user that placed an item to cart
Quantity	Integer	Determines how many of the items are ordered

Sample			
itemID	userID	quantity	variantID
300	1	2323	34
301	1	2325	35
302	1	742	36
303	1	7894	37
304	2	536	36
305	2	234	34

Visualisation of cart and variant tables



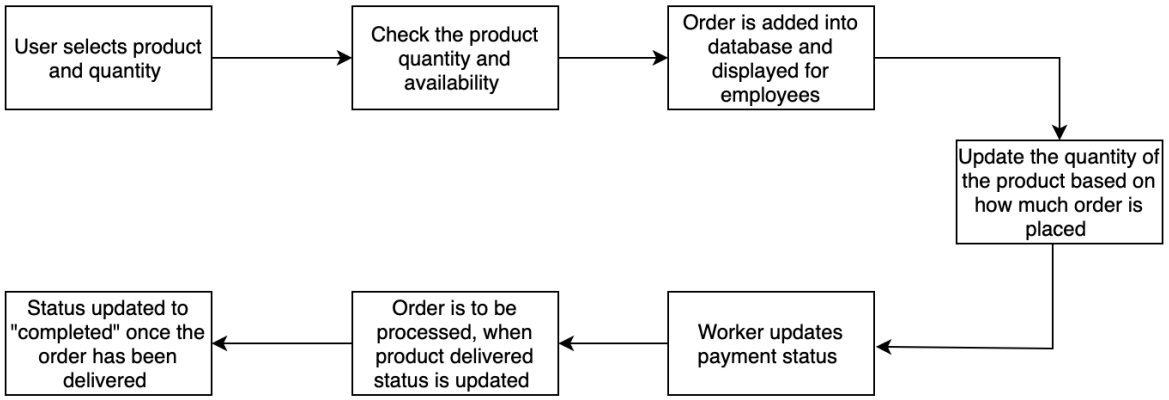
Daily Statistics Table:

Stores the total sales and revenue for each day.

Field name	Data type	Description
dailyID	Integer	Uniquely identifies the month
dailyRevenue	Integer	This record gets added every time a new order is placed
dailySales	Integer	Quantity is added here when an order is placed
date	Date format (DD/MM/YY)	Identifies the date of the record

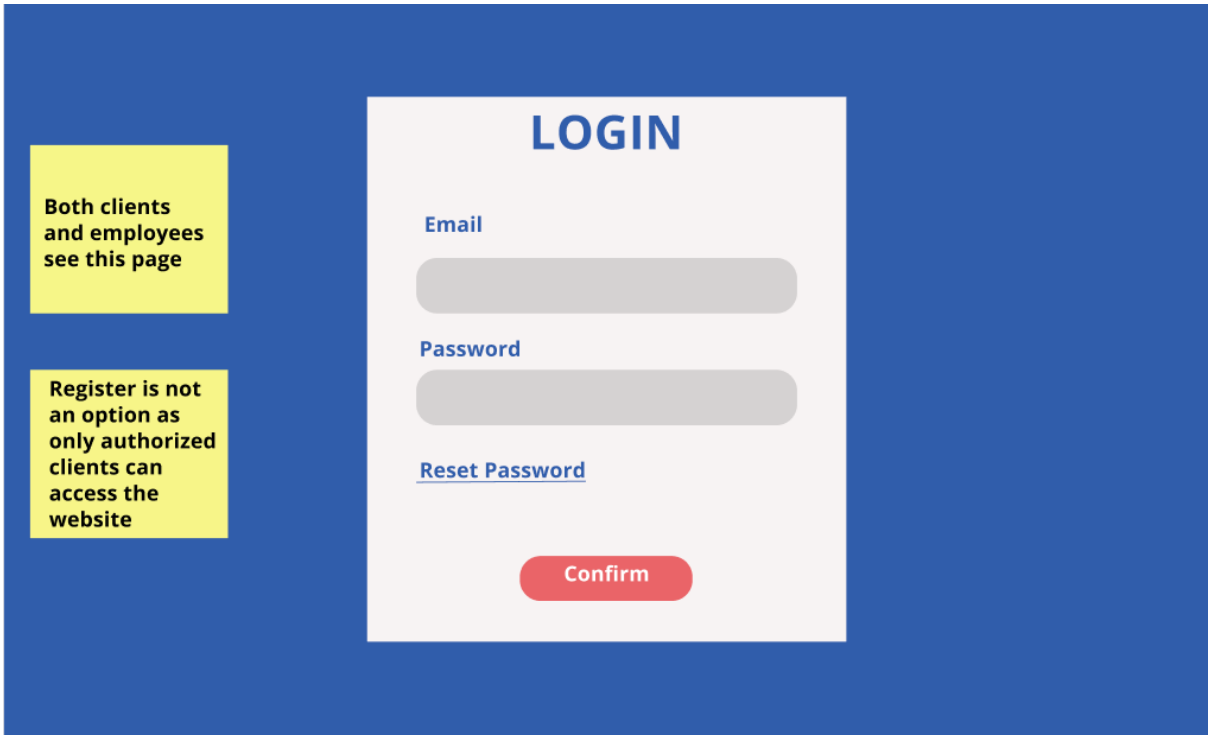
General ordering process:

The diagram below shows the process of ordering from start to completion.

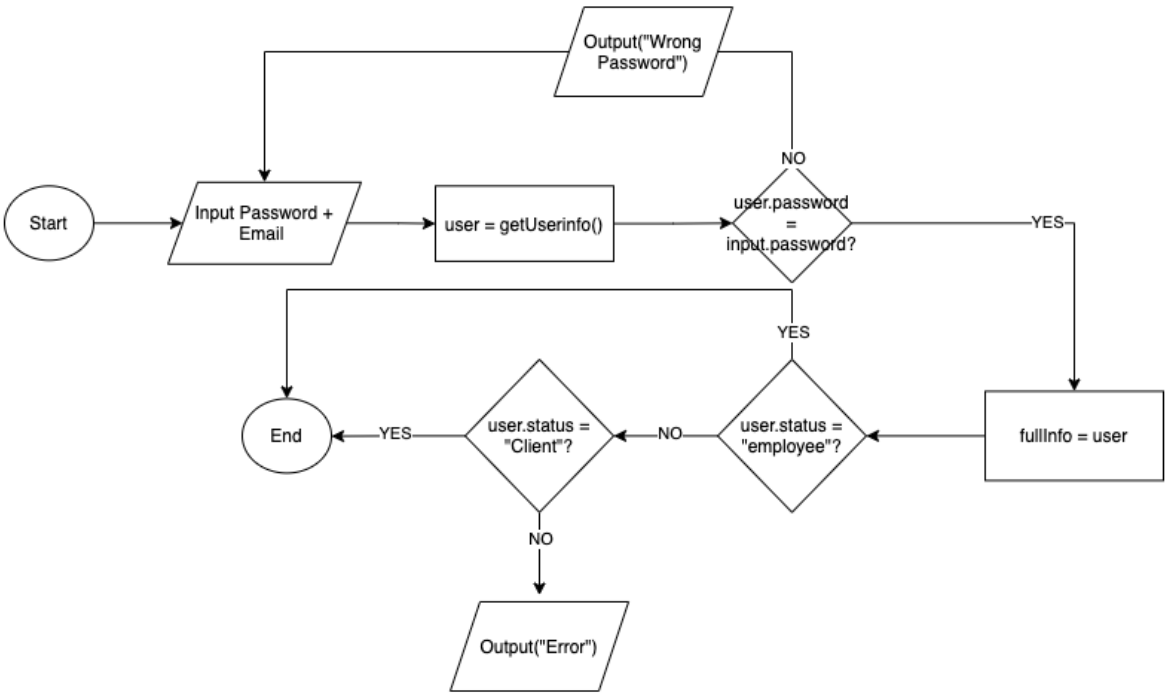


User Interfaces + Flowchart of all diagrams

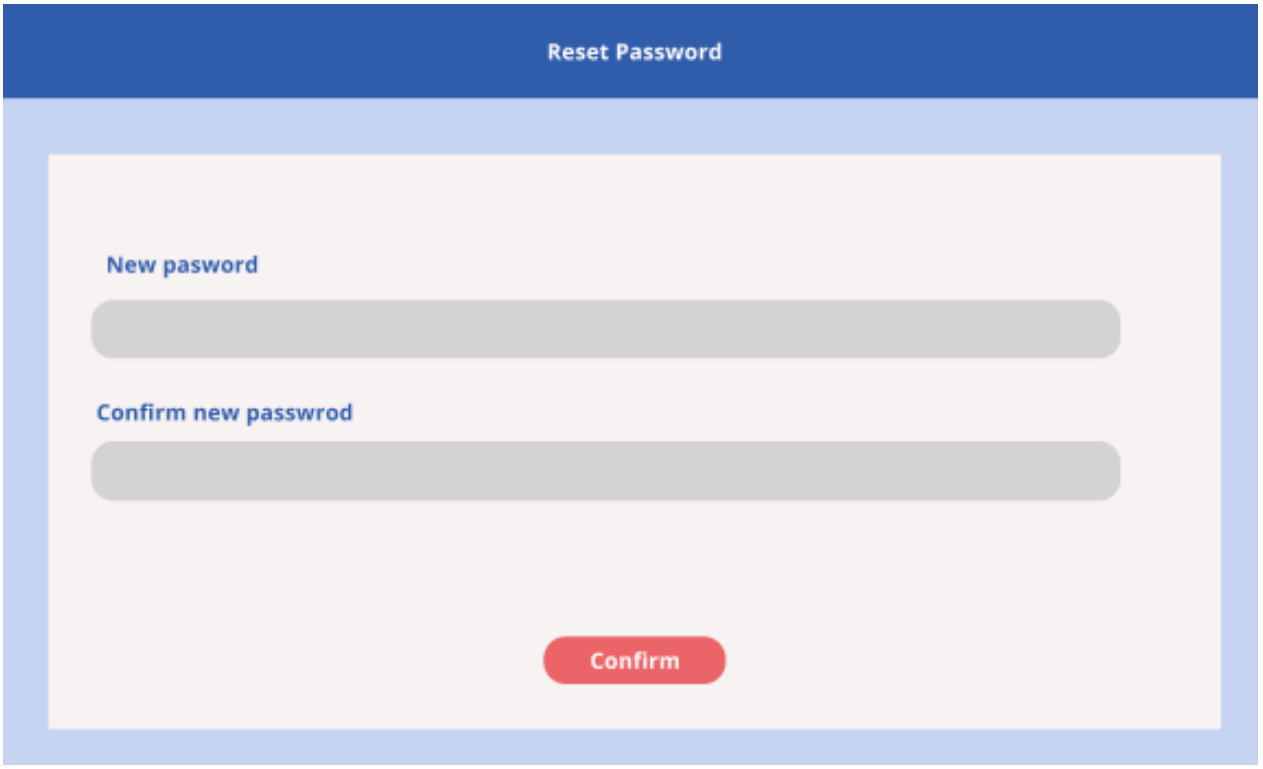
All parties (login Page)



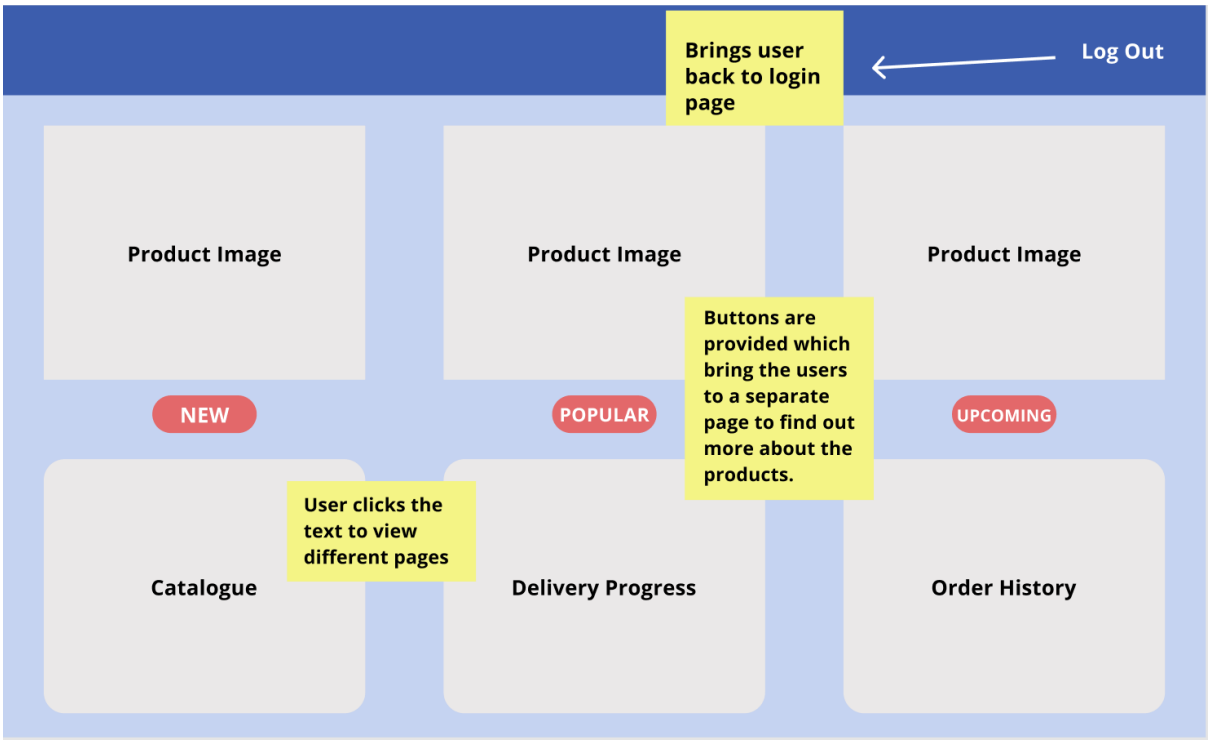
Login process:



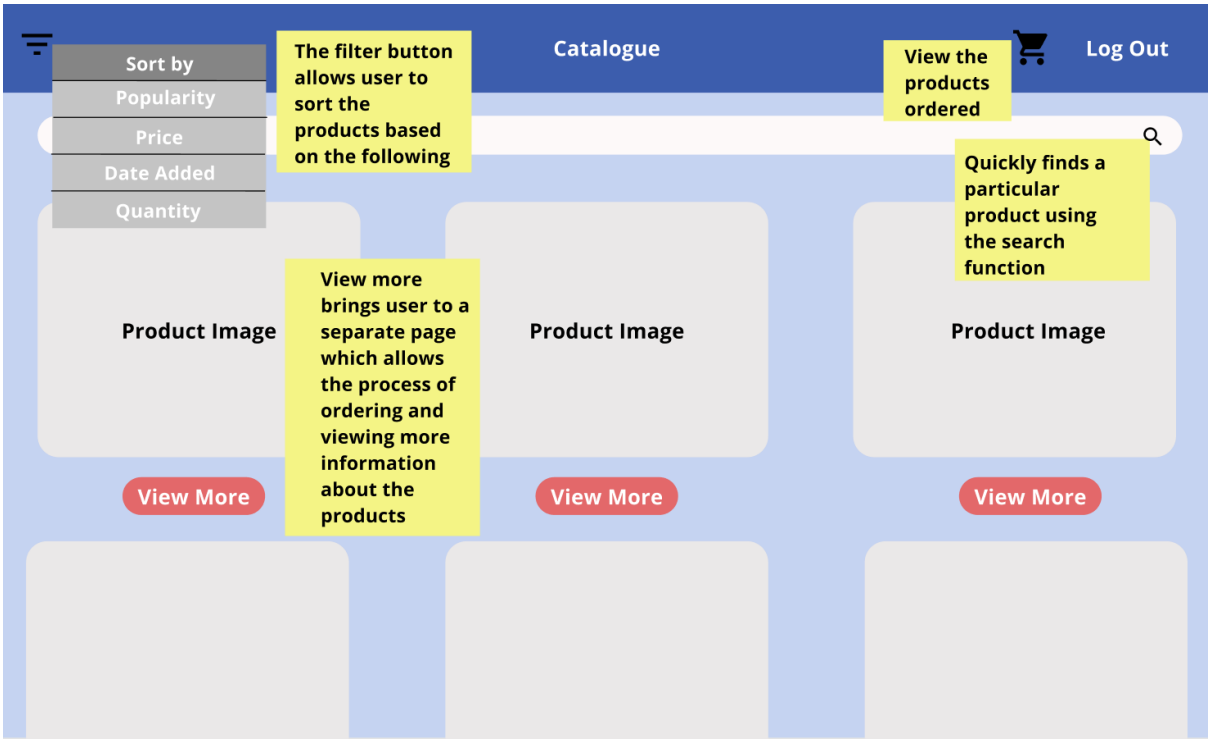
Reset password:



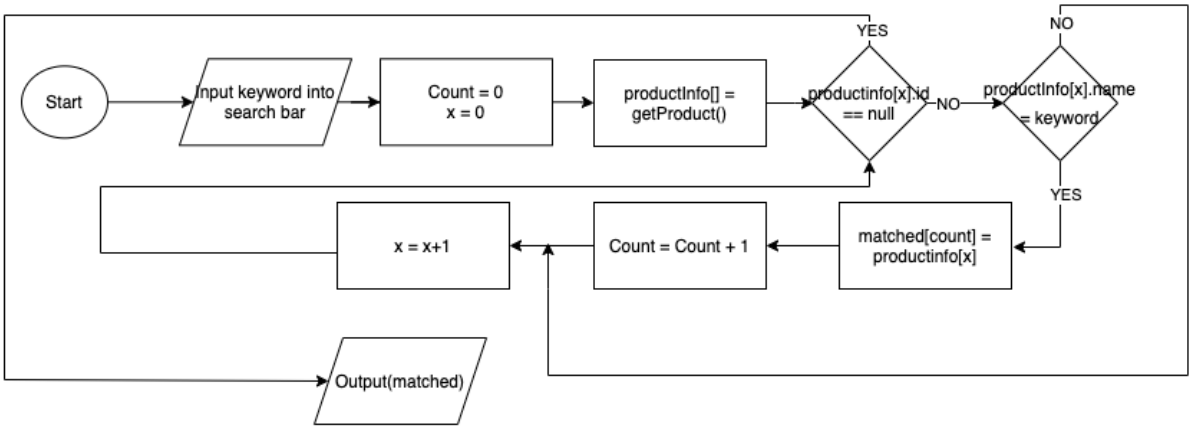
Clients
Main menu:



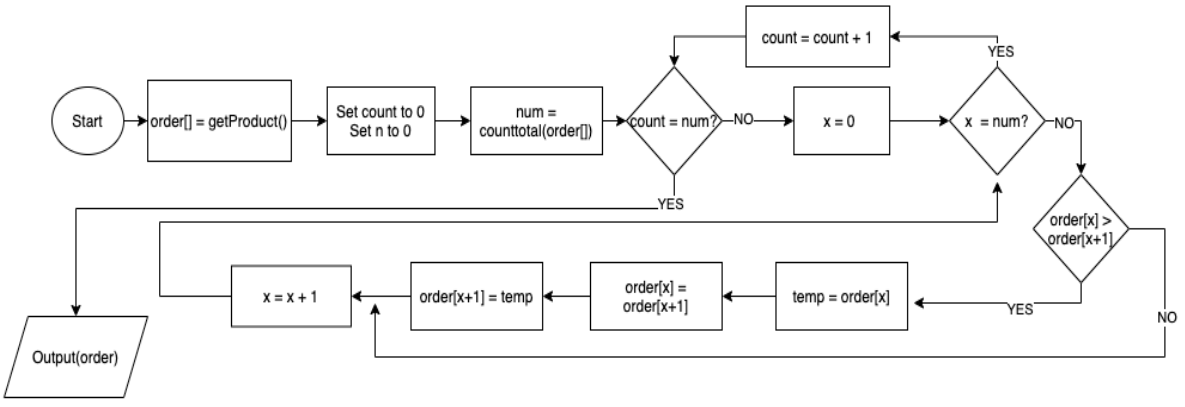
Catalogue Page:



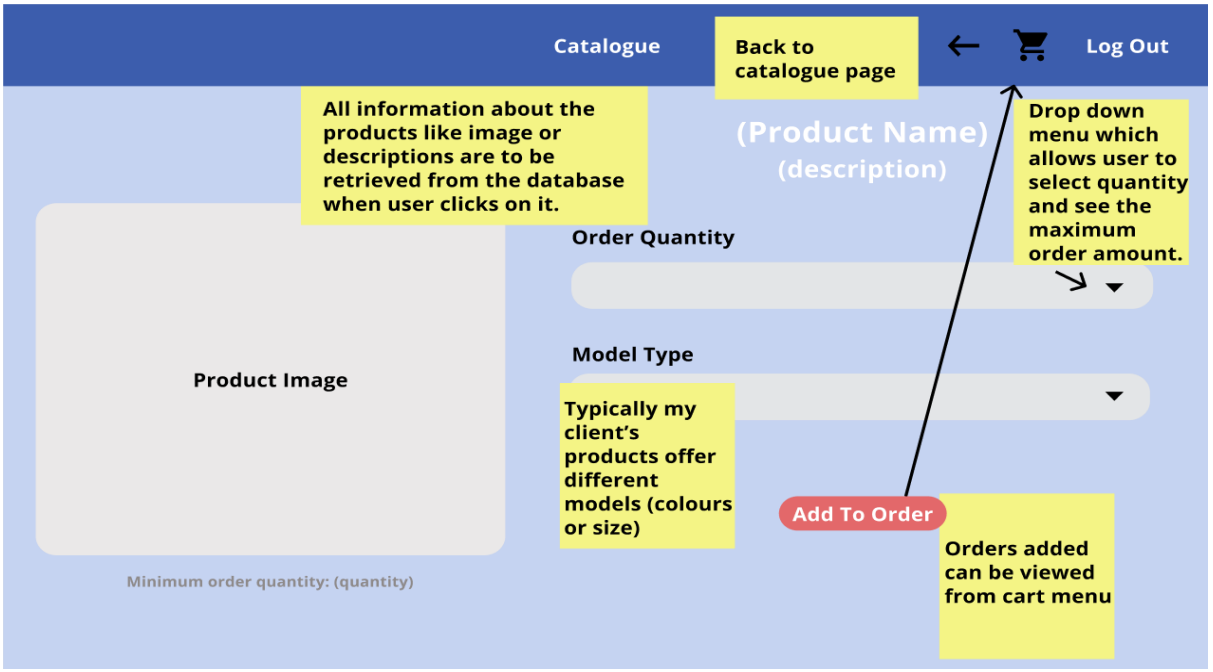
Searching process:



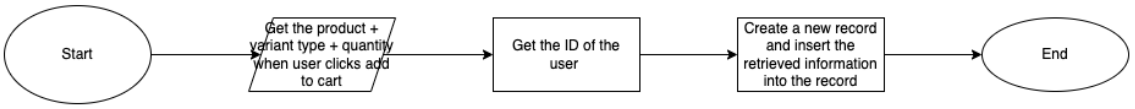
General sorting process (sorting by popularity as a sample):



Product More Info/Order Page:

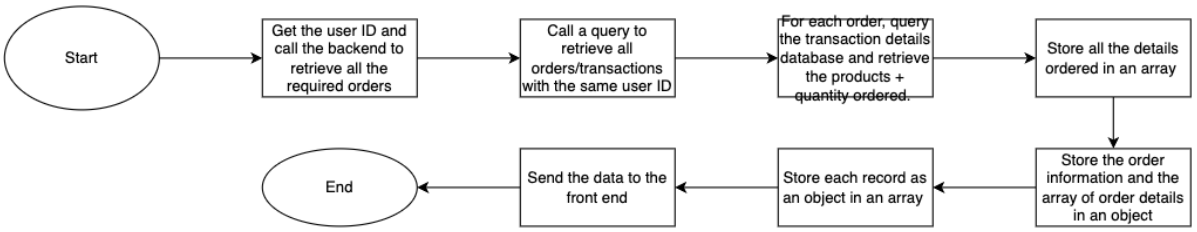


Add to cart process:



Order History:

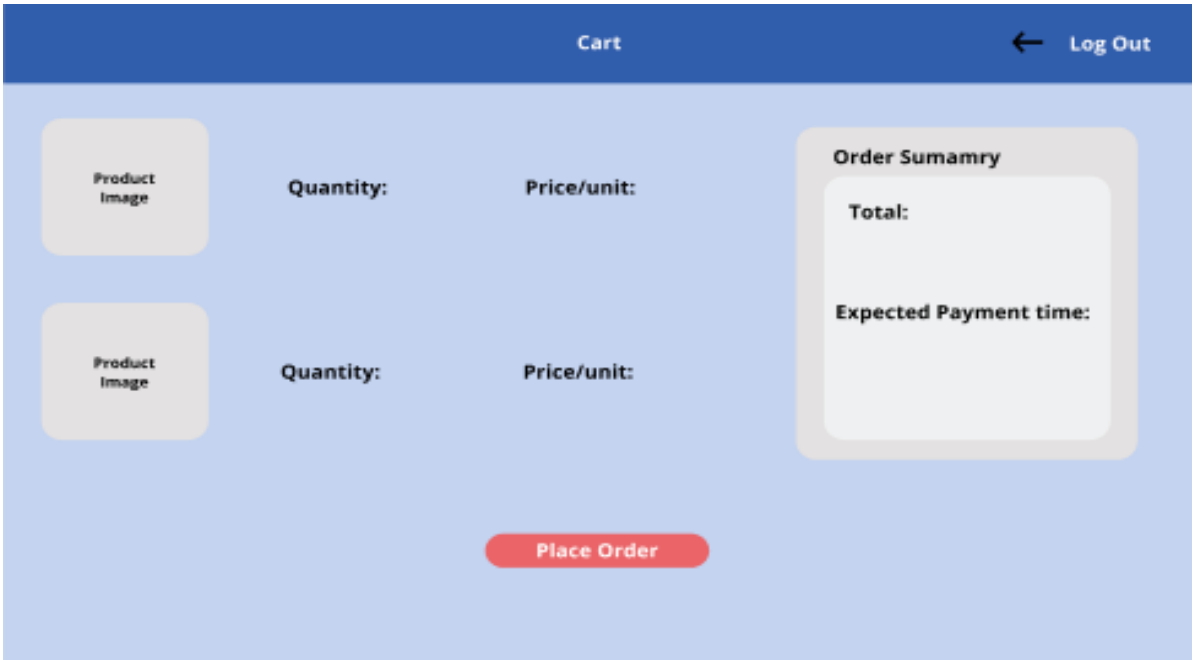
The following sequence is called when page loads.



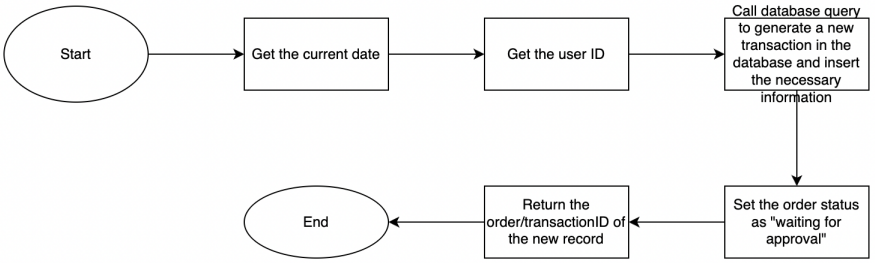
Delivery progress uses the same flow of algorithm to display transactions (only orders other than completed are shown).

Delivery Progress Page:

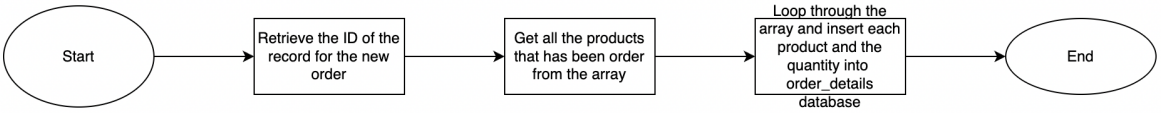
Cart Page:



The flowchart below outlines the process to generate a new order:

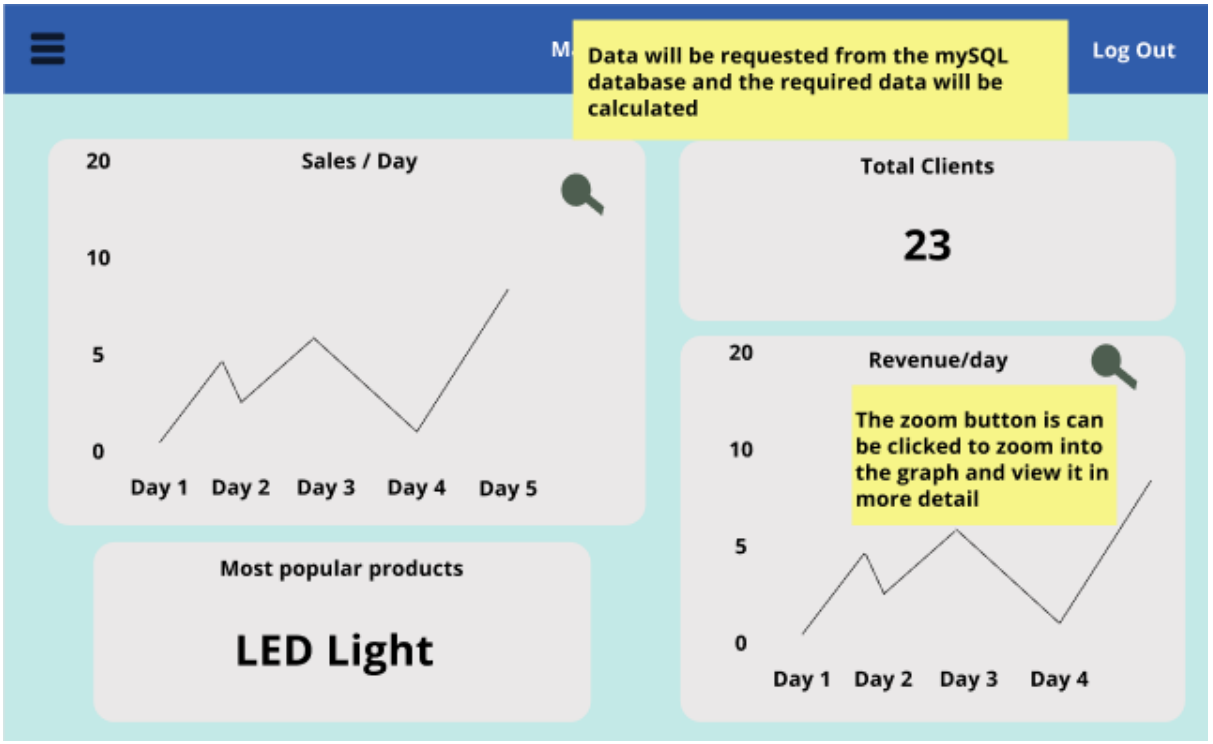


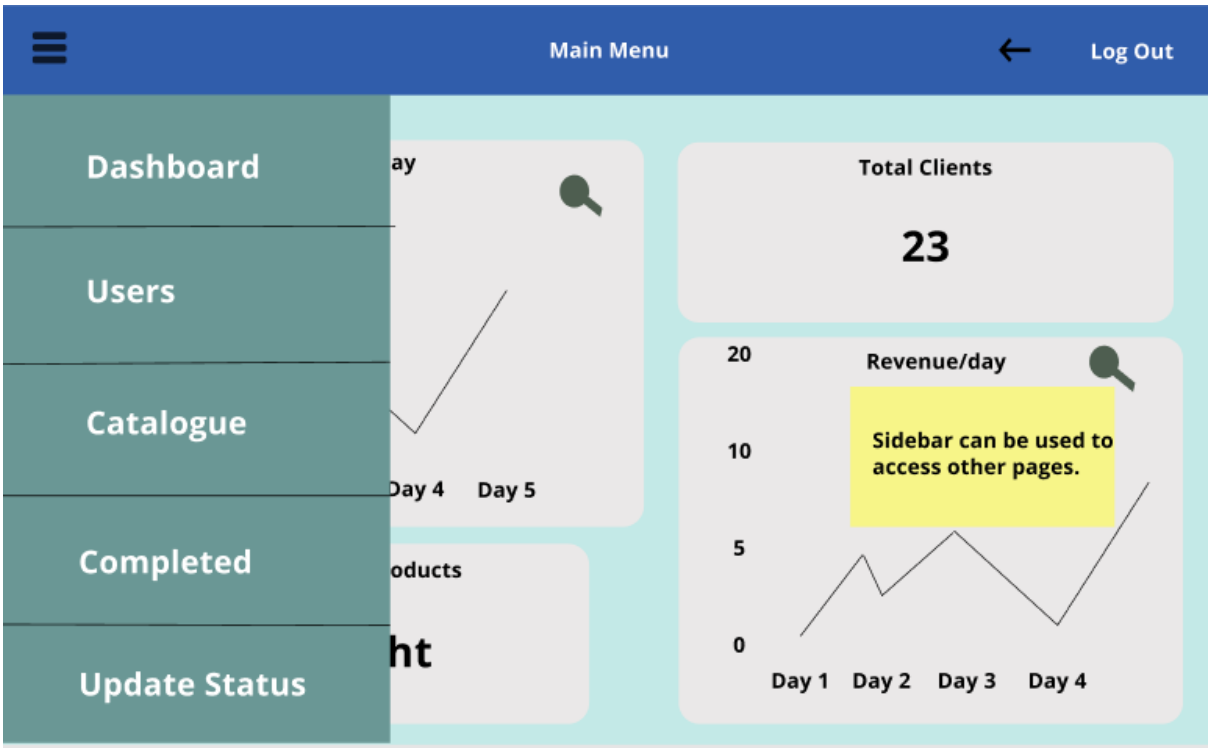
The following displays the use of the asynchronous function (can only be executed once the previous function returned a value) as it requires the orderID to be generated so that the orders can be linked to the products and quantity.



Employees

Main Menu:

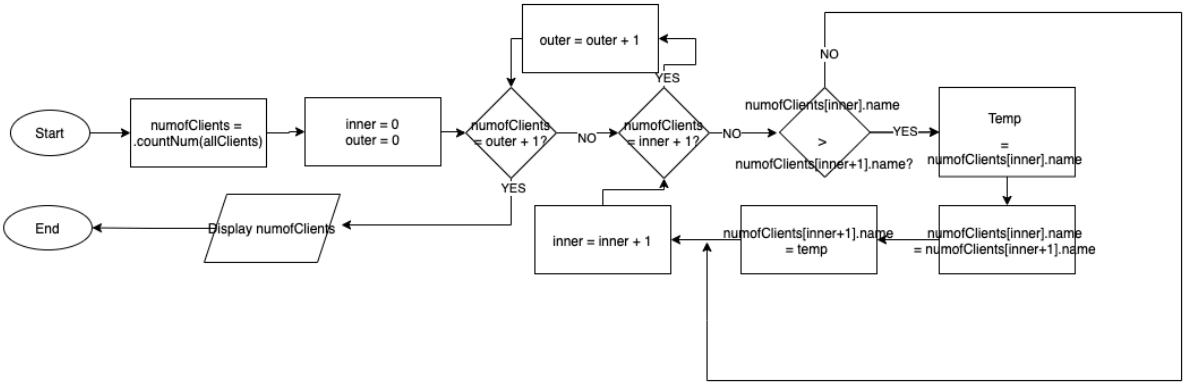




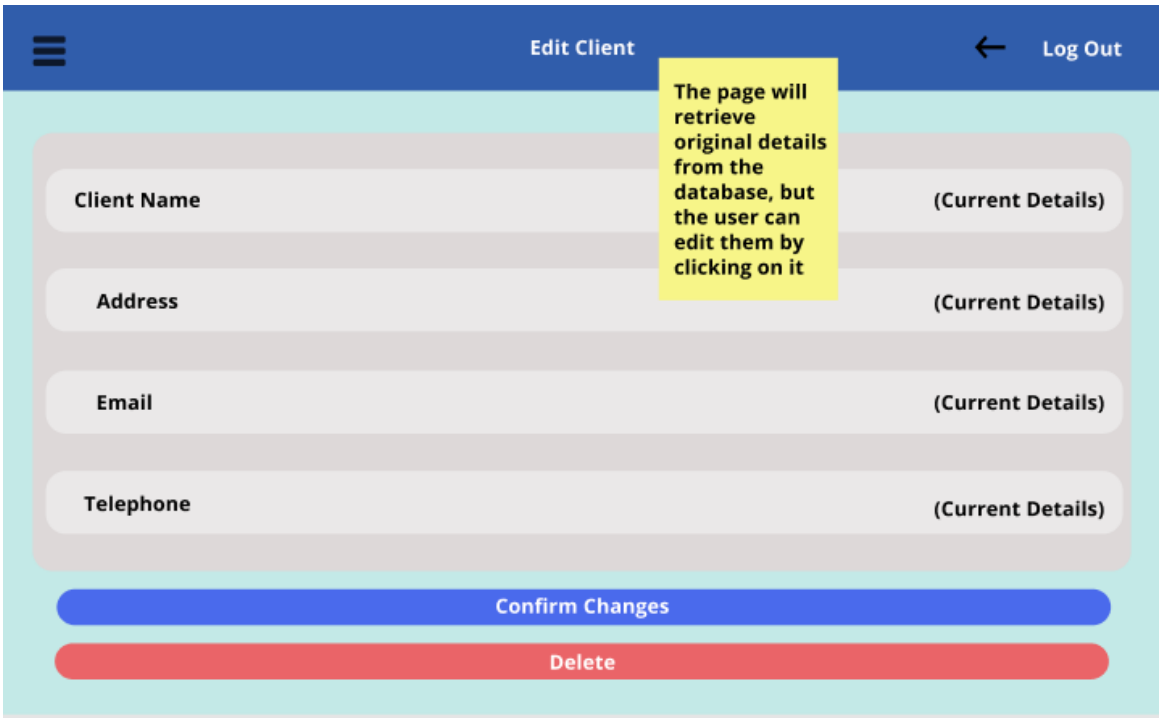
Client/customer list:

Sort the client list by alphabet:

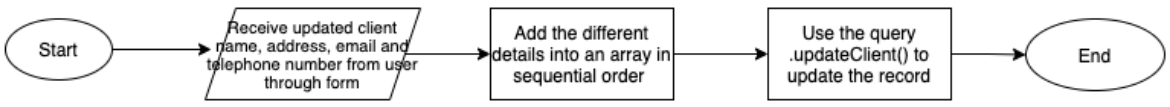
Other sorting processes are quite similar so they won't be shown



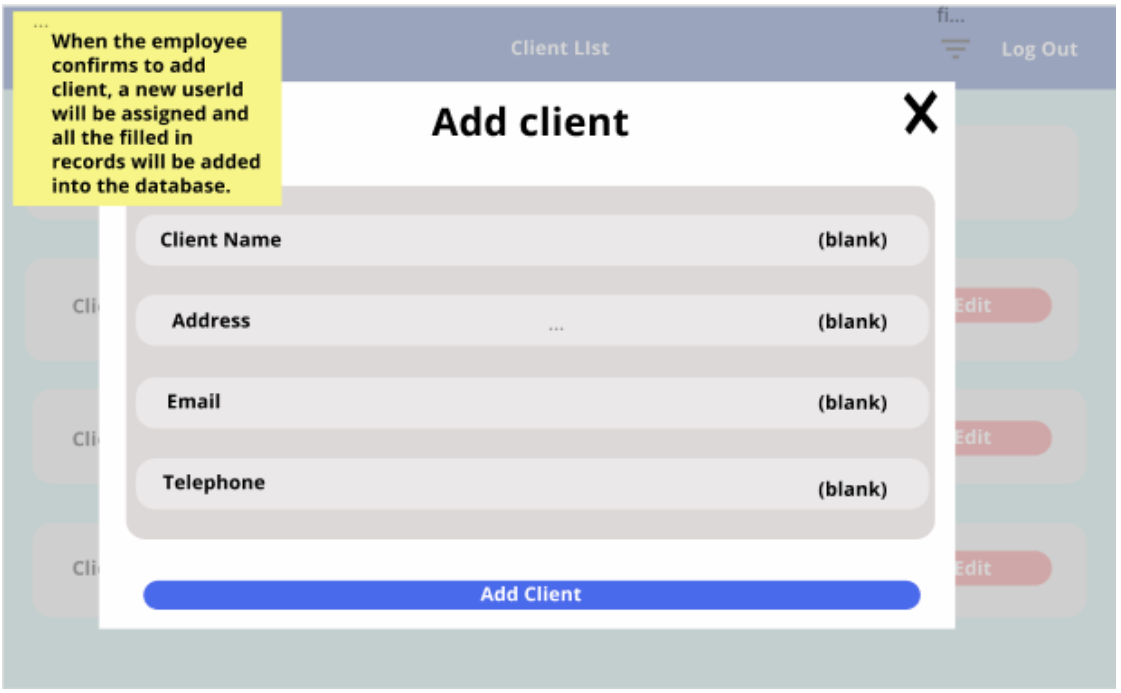
Edit Client Page:



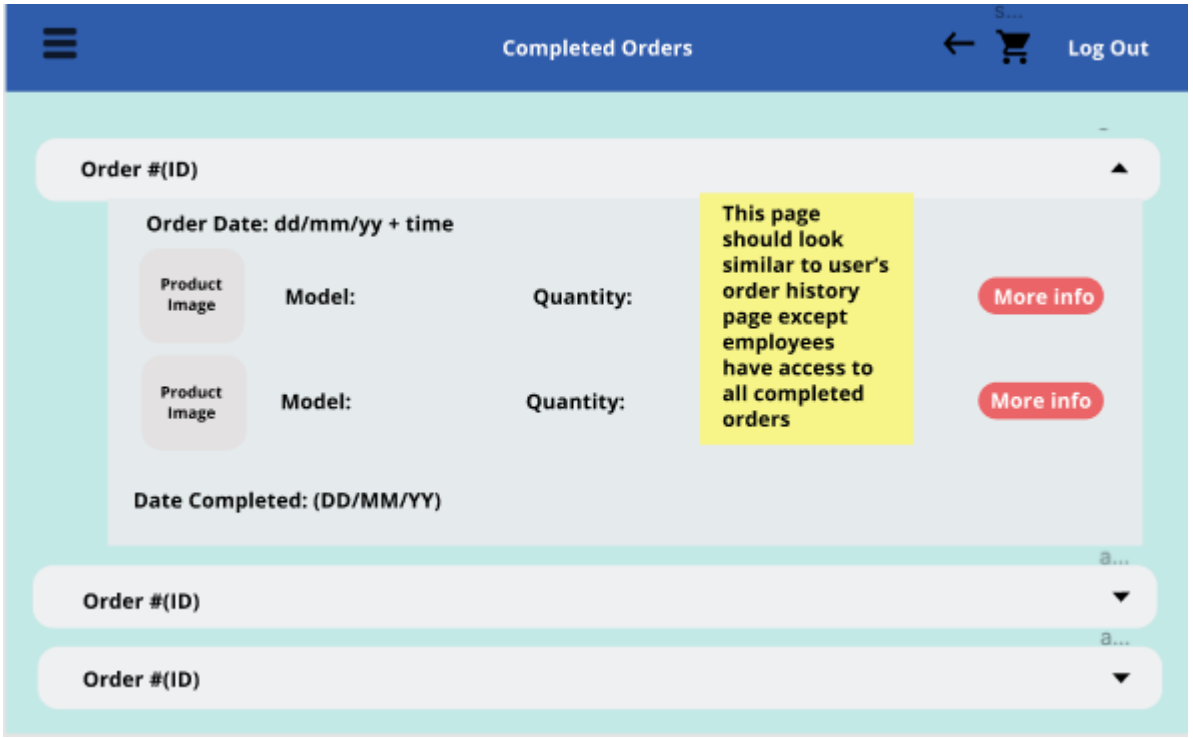
submitChange():



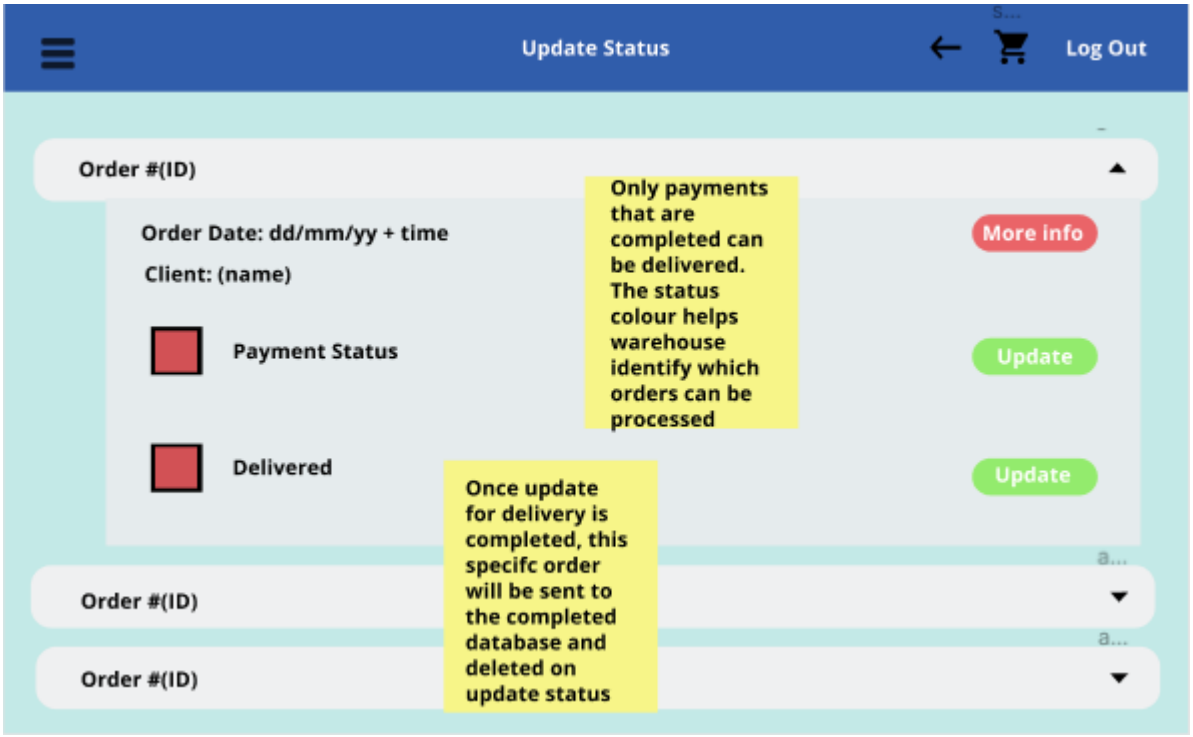
Add client popup:

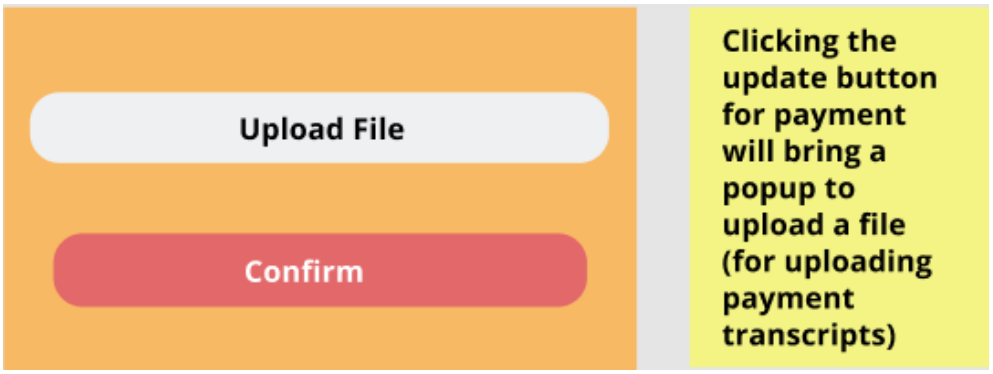


Completed Orders:

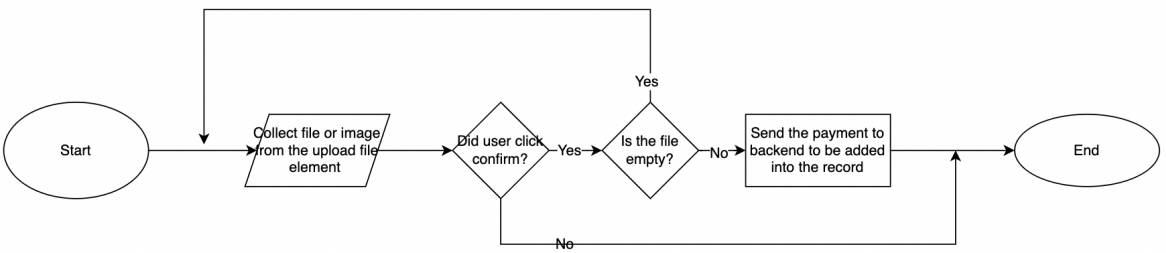


Update Status Page:

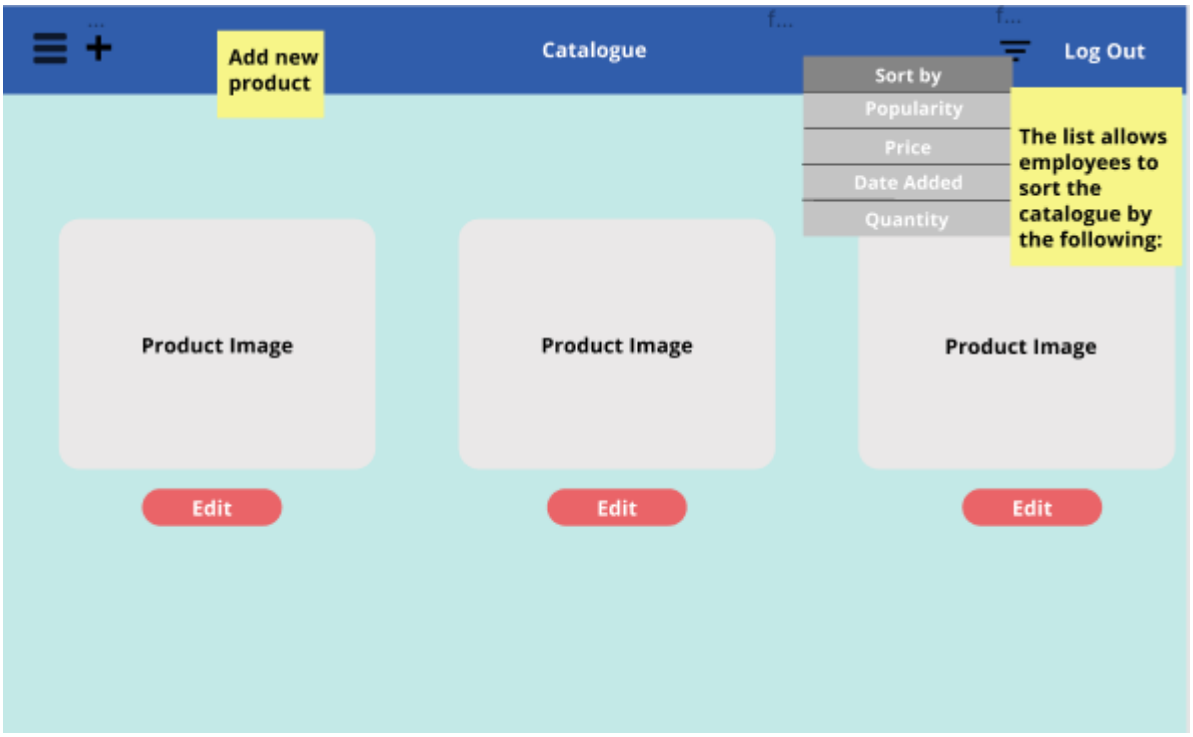




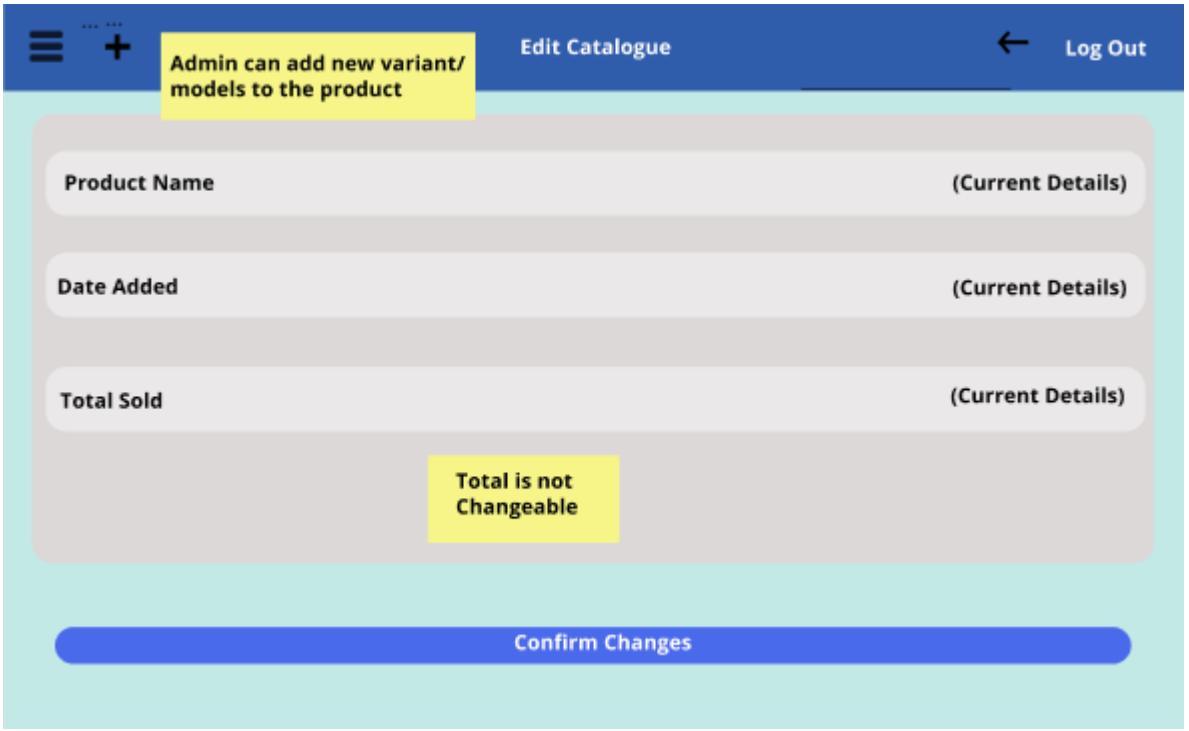
Confirm To Upload Payment:

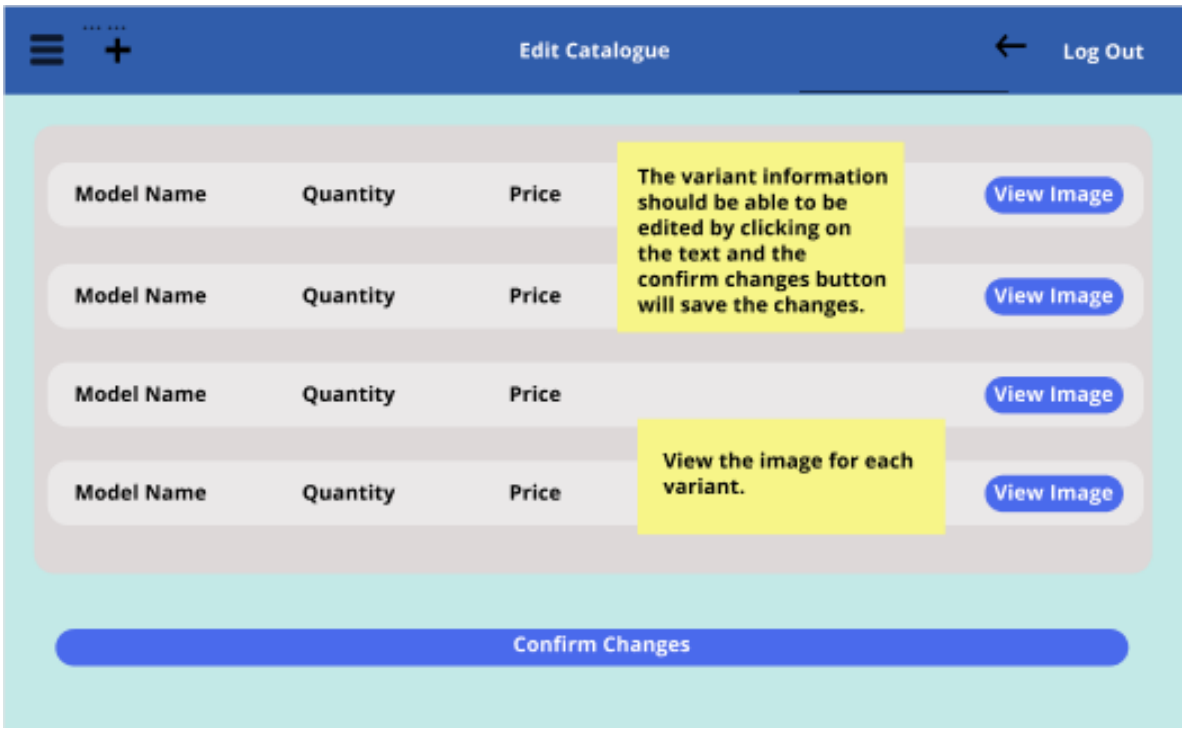


Catalogue Page:

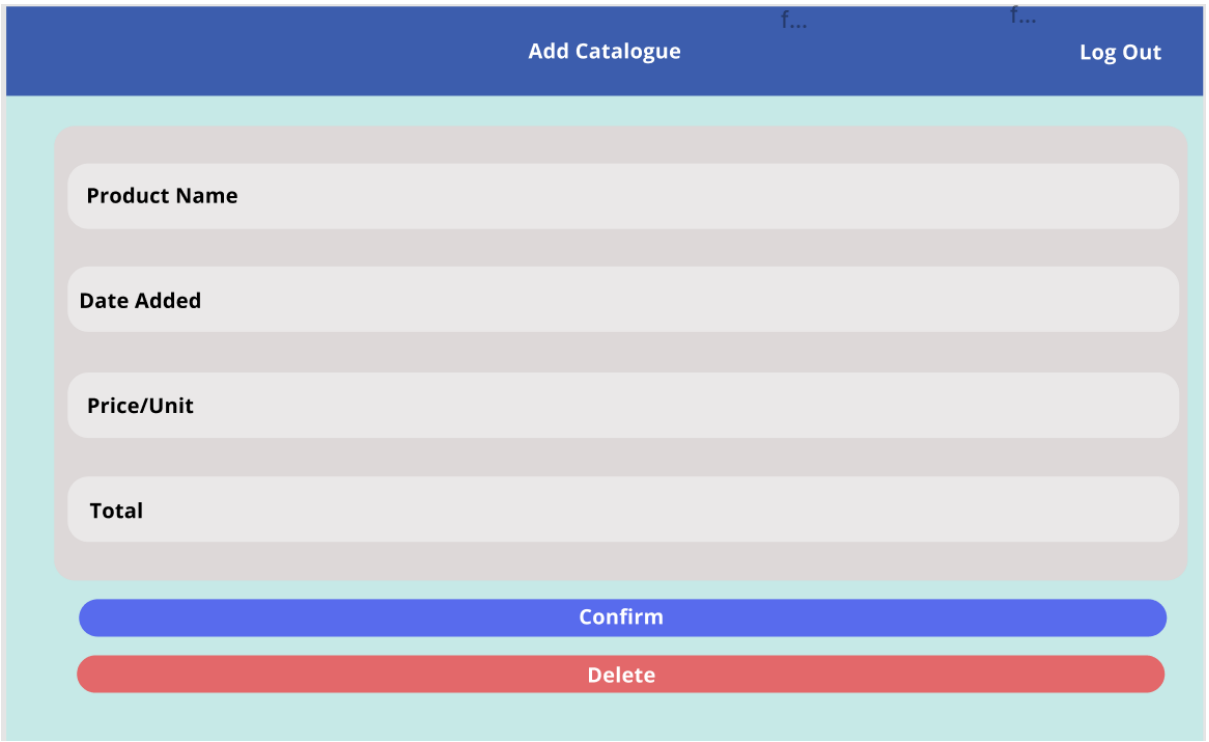


Edit Catalogue Page:

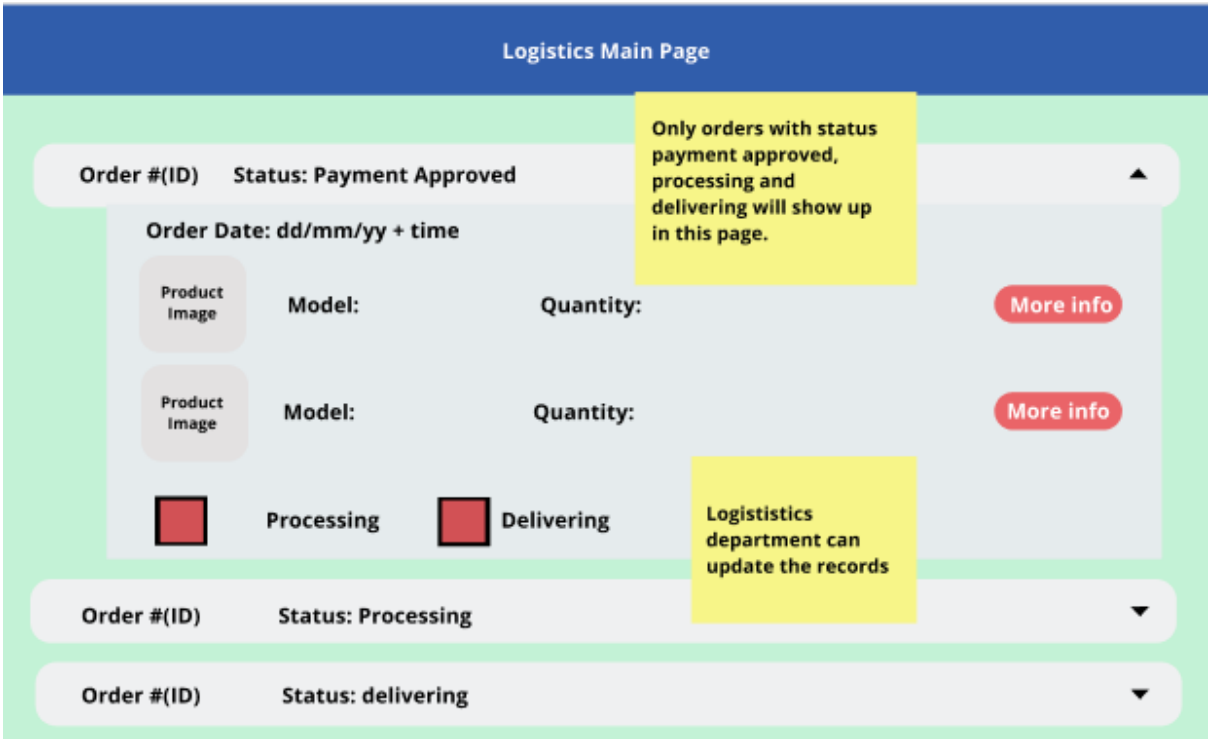




Add Catalogue Page:



Logistics main page:



Test Plan

Test Plan	Criterion Number	Criterion for Success	Test to be done	Expected outcome
	Authentication & Validation			
1	1	Authentication features that limit access of the users based on their roles.	Email: admin@gmail.com Password: 1234 Email: logistics@gmail.com Password: 1234 Email: user1@gmail.com Password: 1234	Brings the users to their respective main page based on their status (logistics, admins, users) (If the details are wrong, asks the user to re-input the details)
2	1	Authentication features that limit access of the users based on their roles.	Open the admin menu while logged in as the USER role.	If the user doesn't have access to a webpage or is not logged in, reject them by directing the page to the login menu.
3	1	Authentication features that limit access of the	Access certain pages in incognito browser mode to model	If the user doesn't have access to a webpage or is not logged in, reject them by directing the page

		users based on their roles.	someone who hasn't logged in.	to the login menu.
4	1	Any form of authentication (JWT tokens) details must be deleted when the user logouts	Log out.	Tokens should be deleted when the user logs out.
5	2	All passwords should be hashed	Create a new user: Email: (use temporary email)	The password should be different from the default value as it is hashed.
	Resetting Password			
6	3	User is able to reset password.	Click forget password and check email.	If the email exists: An email should be sent prompting the user to reset the password with a hyperlink. If the email doesn't exist: Asks the user to input a valid email.
7	3	User is able to reset password.	Click a hyperlink that is contained in the email.	Redirects the user to a password reset page.
8	3	Only valid for emails that exist within the database.	Use an email that doesn't exist within the database.	Alerts the users.
	Admin Role - Main Menu			
9	4	Admin should be able to view and manipulate the analytics.	Open the menu page and click the buttons to ensure that they are functional in manipulating the graphs.	All the mentioned statistics will be displayed in the correct format.
	Admin Role - Managing Orders			
10	5	Admin should be able to accept an order that is waiting for approval.	Admin clicks the accept order button.	Status is updated and an email is sent that includes the invoice when an order has been accepted.
11	6	Allow admin to sort and filter the table records.	Select the filter options available.	The records should be displayed in the requested format.

12	7	View relevant details about each record/order	Click the “view more” button.	The details of the transaction is displayed according to the selected row of record.
	Admin Role - Managing Users			
13	8	Admin can search the users and filter records based on names and email.	Input a selected name into the search bar.	Only the relevant records are shown.
14	9	Create a new user	Input the necessary information into the records. As well as, determine the role of the new user.	If any of the records are blank, notify the admin. The new user should be available in a new record within the database. An email should also be sent with the relevant login details.
15	10	Edit and delete users	Change user information and click confirm	When the window is refreshed, display the new/updated information.
16	10	Edit and delete users	Click the delete user button	User is removed from the database.
	Admin Role - Managing Transactions			
17	11	Allow the admin to view and update all the pending transactions that need to be reviewed.	Admin can view the uploaded payment proof and accept/reject the payment.	The status of the order should be updated based on whether it was rejected or accepted. The value should be added to daily sales and revenue.
	Admin Role - Managing catalogue			
18	12	Admin can add product and edit the details.	Input sample product information to add new products	Refresh the page and display the recently added product.
19	12	Admin can add products and edit the details.	Edit the product information.	The edited product details are displayed.
20	12	Admin can add products and edit the details.	Add a new product variant/model.	New variants/models can be added to an existing product.
21	12	Admin can add products and edit the details.	Allow the admin to add pictures for each variant/mode.	Multiple images can be uploaded to each variant.
22	12	Admin can add products and edit the details.	Edit the variant details.	The edited variant details should be displayed.

23	12	Admin can add products and edit the details.	Edit and replace an image.	The image should only change for the selected variant and image.
	Logistics Role - Updating order status			
24	13	View and update relevant records to the logistic department.	Click all the buttons to update the transactions.	The status is updated.
	User Role - Ordering System			
25	14	View all the available products and search them by category and name.	Input the name of one of the products.	Filter the catalogue list to only show the relevant products.
26	15	Users should be able to view a product and change the variant/model of the product during browsing.	Select a different variant/model for the product.	The images and price should change depending on what the user selected.
27	16	Allow the user to add items to cart and create a new order.	Add a selected item and product to the cart.	When the cart page is viewed, all the inputted items should be available.
28	16	Allow the user to add items to the cart and create a new order.	View cart.	The cart should display all the items added with the previously selected quantities.
29	16	Allow the user to add items to the cart and create a new order.	Click a button to change the quantity value.	The value/price should change based on the quantity that the user selects.
30	16	Allow the user to add items to the cart and create a new order.	Delete an item from cart	When the order is placed, the cart items should be deleted and a new order is generated.
	User Role - Managing orders			
31	17	Users should be able to upload proof of payment for certain orders.	Upload a random image and click upload payment.	The same order from the admin side to check that the correct image has been placed and status is updated on the admin side.

Criterion B: Record of tasks

Task number	Planned action	Planned outcome	Time estimated	Target completion date	Crit.
1	Brainstorming of potential clients	Find the client for the project and set up a potential meeting	2 days	18th June 2021	A
2	Initial interview with the client	Get a full description of the client's scenario and the description of the problem	1 hour	20th June 2021	A
3	Discussion and approval of the project with a supervisor	Discuss potential project improvements and receive approval of the project	30 minutes	25th June 2021	A
4	The second interview with the supervisor	Consult the client of potential solutions and changes	30 minutes	27th June 2021	A
5	User side Interface design	Finish the interface of the website for user side	2 days	7th September	B
6	Employee Side Interface Design	Finish the interface of the website for user side	1 day	9th September	B
7	Single database design	Create an outline of the database table	2 days	11th September 2021	B
8	Multiple relationship database designs	Create an outline of the multiple relationship databases	4 hours	11th September	
7	Logical Flowchart Design	Create a logical flowchart for all the processes (searching, filtering, sorting)	2 days	13th September 2021	B
8	Systematic Flowchart Design	Create systematic flowcharts outlining the flow of processes (retrieving data from the database, storing, delete)	2 days	14th September 2021	B
9	Backend setup + database connection	A new database is created and is connected to the backend side of the web application	3 days	9th November 2021	C
10	Front-end design and setup (without the functionality)	All the raw front-end components are completed according to the design plan.	7 days	17th November 2021	C

11	Page routing and functionality (i.e. logging out/ logging in)	(Criterion 1-3) Login functionalities is working and authentication for valid access for each of the pages. Resetting password functionality should also be completed.	6 days	25th November 2021	C
12	Complete admin-side functionalities (success criteria)	Criterion 4 - 12 is completed	15 days	15th December 2021	C
13	Complete logistics functionalities (success criteria)	(Criterion 13) Logistics side should be able to: View and manipulate orders: - Processing - Delivering - Payment Uploaded	1 day	18th December 2021	C
14	Complete user side functionalities (success criteria)	Criterion 14-17 should be completed	10 days	28th December 2021	C
15	Emailing functionalities	Able to send emails in the following functionalities: - Resetting password - Creating new user - Sending invoice	2 days	30th December	C
16	Recording the demonstration videos	Have all the demonstration videos ready for the next plan	3 days	10th January 2021	D
17	Narrating + editing	Videos are narrated and edited with the displayed success criteria	2 days	16th January	D
17	Final interview with the client	Discuss feedback with the client to find out points for improvement, suggestions and what they liked about the final product.	2 Hours	28th January 2021	E

Techniques, frameworks and, libraries used

<table>
<tr><td colspan="2">Techniques used
- Saving and retrieving data from the database
- Asynchronous function (as multithreading is not supported in javaScript)
  - Event handles, callbacks, promises
- Authentication and validation using JWT
- Sending Email using nodemailer
- Chart manipulation using chartJS</td></tr>
<tr><td>Front-end library/frameworks</td><td>Back-end library/frameworks</td></tr>
<tr><td>HTML
- Chart JS
- XHR HTTP requests (for connecting with the backend to send and receive data)</td><td>Node JS
- Multer
- JWT
- Nodemailer
- cookieParser
- Moment
- Express (framework)
- Easy INVOICE</td></tr>
<tr><td>CSS</td><td>MySQL - Managed using MySQL workbench</td></tr>
<tr><td>javaScript</td><td></td></tr>
</table>

Use of Asynchronous functions

Javascript is a single-threaded programming language where statements are executed one at a time and running **synchronous functions can create thread blocking which is inefficient when there are several requests[1] occuring at the same time as it can't perform tasks concurrently[2].This makes asynchronous functions suitable for web development as it continues running the thread and executes other tasks/functions, even while waiting for time-consuming processes like querying or reading files to finish.** Asynchronous functions allow the use of *await* keyword which tells the compiler to hold off on executing the line/async function until a value has been returned in the form of a *promise*. Synchronous function is faster when it performs one request, but **asynchronous functions improve web app scalability as time-consuming processes such as database queries can be held off to perform other tasks.**

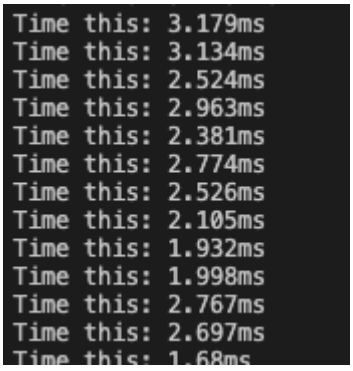


Figure 1: Execution time using async functions

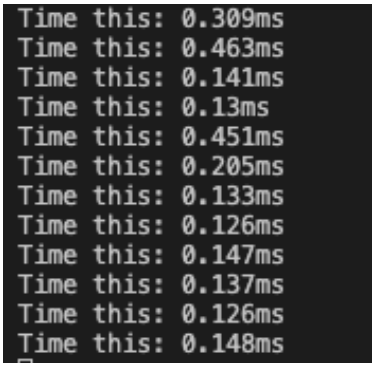


Figure 2: Execution time using synchronous functions

[1] "Entity Framework Why EF Async Methods Are Slow," zzz projects, accessed January 2, 2022, https://entityframework.net/why-ef-async-methods-are-slow.
[2] Frederik Banke, "Improving Scalability in C# Using Async and Await," Medium (Medium, March 2, 2019), https://medium.com/@frederikbanke/improving-scalability-in-c-using-async-and-await-f97af1466922.

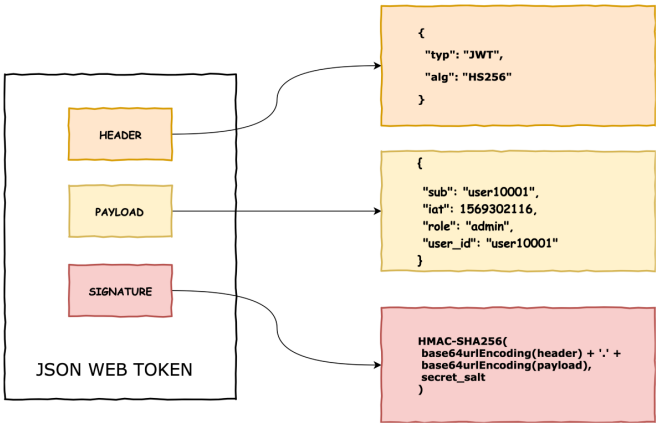
The following API makes use of a function that returns the query result in the form of a promise and the await keyword pauses the function until the database query has been completed.

```
//Function that takes in a query and returns a promise
function queryDb(query) {
    return new Promise((resolve, reject) => {
        //Creates a connection to the database using connection.
        connection.query(query, (err, result) => {
            if (err) {
                return reject(err);
            }
            //Returns the result by resolving in the form of a promise
            resolve(result);
        });
    })
}
```

```
app.get('/getallcatinfo/:catID', authenticateToken, async (req, res) => {
    try {
        //Makes sure that the user is an ADMIN
        if (req.userInformation.status == "ADMIN") {
            //Creates two queries, 1: querying the product info, 2: querying the all the other product types
            var queryProduct = 'SELECT * FROM `catalogue` where  productID = ' + req.params.catID
            var queryVariant = 'SELECT * FROM `variant` where  productID = ' + req.params.catID
            //Using promise.all allows both functions to run in parallel and holds off on execution until a promise is returned
            const [allProducts, allVariants] = await Promise.all([queryDb(queryProduct), queryDb(queryVariant)]);
            //The returned values are parsed into javaScript object
            var productINFO = JSON.parse(JSON.stringify(allProducts));
            var variantINFO = JSON.parse(JSON.stringify(allVariants));
```

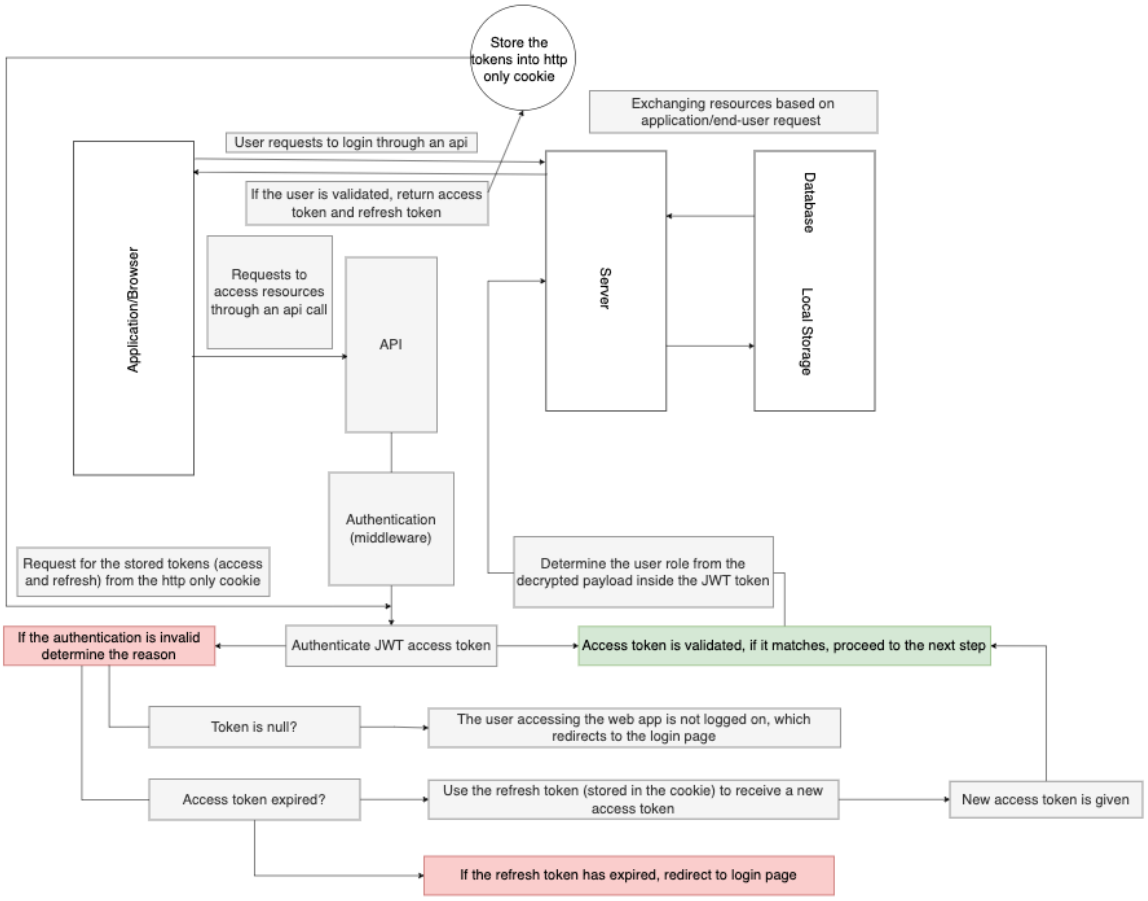
Authentication and user validation - JWT token and cookie

The following shows the structure of the JWT token primarily used for sending data across sessions and verifying the eligibility of the users to open a web page or access data.



Structure of a JWT token[3]

[3] Suresh Kumar, “How JSON Web Token(Jwt) Authentication Works?,” Medium (Medium, March 20, 2021), https://medium.com/@sureshdsk/how-json-web-token-jwt-authentication-works-585c4f076033.



The flow of authentication and validation of the web application

The JWT token will be stored in HTTP only cookie as opposed to a front-end session or local storage in the browser as it supports back-end validation and authentication of users. **This prevents the possibility of a cross-site scripting (xss) attack which allows attackers to retrieve any data stored in the front-end local storage by running a malicious javascript code[4]. Server-side sessions such as HTTP only cookie are suitable for storing JWT tokens as they are impossible to be accessed through front-end javascript due to its settings.**

When a user first logs in, the following API route is called to authenticate the user and generate an access/refresh token if successful.

[4] map[name:Randall Degges email:r@rdegges.com], “Randall Degges,” Randall Degges - Please Stop Using Local Storage, accessed January 3, 2022, https://www.rdegges.com/2018/please-stop-using-local-storage/.

```
//A route that calls a function calls generateStatusToken() and stores the retrieved value inside the http only cookie
//The user status (admin, user, logistics) is returned to the user for page routing.
app.post('/loginusers', async (req, res) => {
    try {
        //Pass email and password into the function
        const tokenStatus = await generateStatusToken(req.body.email, req.body.password)
        //Store the values inside http only cookies with the following name
        //            Name      Passing the returned token values   Expiry duration   Can't be accessed through Javscript
        res.cookie('accessToken', tokenStatus.accessToken, { maxAge: 60000000000000, httpOnly: true })
        res.cookie('refreshToken', tokenStatus.refreshToken, { maxAge: 100000000000000, httpOnly: true })
        //Respond the user status to the user
        res.json({
            status: tokenStatus.status
        })
    }
    //If erorr send the error message
    catch (err) {
        res.send(err)
    }
})
```

generatestatustoken(), calls another function to generate new access and refresh tokens if the password and email match.

```
    for (specificUser in allUsers) {
        //Condition: if password matches with the email
        //source: https://www.youtube.com/watch?v=Ud5xKCYQTjM&ab_channel=WebDevSimplified
        if (requestedEmail == allUsers[specificUser].email && await bcrypt.compare(requestedPassword, allUsers[specificUser].password)) {
            //Create a user object container the ID and status
            const user = {
                userID: allUsers[specificUser].userID,
                status: allUsers[specificUser].status
            }
            //Determine if the user's role/status
            if (allUsers[specificUser].status == "USER") {
                status = "USER"
            }
            else if (allUsers[specificUser].status == "ADMIN") {
                status = "ADMIN"
            }
            else if (allUsers[specificUser].status == "LOGISTICS") {
                status = "LOGISTICS"
            }
            //Calls a function to generate the token
            // Source: https://www.youtube.com/watch?v=mbsmsi7l3r4&t=1052s&ab_channel=WebDevSimplified - setting up jwt token functionality
            var accessToken = generateAccessToken(user);
            var refreshToken = generateRefreshToken(user)
        }
    }
```

JWT is the keyword that is used to access all the libraries to perform verification and token generation.

```
const jwt = require('jsonwebtoken');
```

```
// Source: https://www.youtube.com/watch?v=mbsmsi7l3r4&t=1052s&ab_channel=WebDevSimplified - setting up jwt token functionality
function generateAccessToken(userInformation) {
    return jwt.sign(userInformation, process.env.ACCESS_TOKEN_SECRET, { expiresIn: '10s' })
}

// Source: https://www.youtube.com/watch?v=mbsmsi7l3r4&t=1052s&ab_channel=WebDevSimplified - setting up jwt token functionality
function generateRefreshToken(userInformation) {
    return jwt.sign(userInformation, process.env.REFRESH_TOKEN_SECRET, { expiresIn: '7d' })

}
```

Jwt.sign() is a function that generates new tokens that takes in the payload (user information) and the secret code as parameters.

The following showcases the use of the middleware, *authtoken()*, that is called before API proceeds with any tasks to make sure that the access token is valid and retrieve the payload inside the JWT token to find out the user role.

```
//Middleware used to authenticate the user's access token or refresh token
function authenticateToken(req, res, next) {
    //Retrieve the access token from the http only cookie
    const accessToken = req.cookies.accessToken
    //If there is no access token then the user is not logged in --> redirects to the login page
    if (accessToken == null) {
        return res.redirect('/login')
```

The token is checked to make sure it hasn't expired or else it will generate a new one, with the condition that the refresh token is still valid. Otherwise, the users will be logged out as they have been logged in for too long.

```
    } else {

        //Verify that the token is valid by passsing in the token secret and the accessToken
        jwt.verify(accessToken, process.env.ACCESS_TOKEN_SECRET, (err, userInformation) => {
            if (err) {
                //If there is an error, call a function to generates a new access token as there is a token but it is invalid
                var newAccessToken = generateNewAccessToken(req.cookies.refreshToken)
                //If there is no refresh token it might have been expired --> redirect to the login page so that users can
                //receive a new token
                if (newAccessToken.newToken == null) {
                    return res.redirect('/login')
                }
                //Otherwise store the decoded userInformation within the JWT token into req.userInformation which can be access by the requesting api
                else {
                    req.userInformation = newAccessToken.user
                    next()
                    //Also storest the new access token
                    return res.cookie('accessToken', newAccessToken.newToken, { maxAge: 100000000000, httpOnly: true })
                }
                //This is called if the jwt access token is valid
            } else if (userInformation) {
                req.userInformation = userInformation
                next()
            }
        })
    }
}
```

```
//Authtoken to validate the user through tokens is called before the api proceeds with any code
app.get('/getalldashboard', authenticateToken, async (req, res) => {
```

Function called before it proceeds with the rest of the code

Sending and receiving data between backend and frontend (CRUD operations)

Backend

The backend needs to connect to the database to retrieve or edit any form of data through the use of MySQL dependency to have access to the library. The database password is stored in a local file called *.env* as it shouldn't be shared.

```
const mysql = require('mysql');

var connection = mysql.createConnection({
    host: "localhost",
    connectionLimit:10,
    user: "root",
    password: process.env.DATABASE_PASSWORD,
    database: "order",
    port: 3306
})
```

The backend end node JS makes use of a web framework called Express JS which helps create HTTP methods and middleware, as well as, connecting to the database to retrieve and send information.

The following shows a route that requests to GET a certain resource and the middleware *authtoken()* is called to authenticate the user role to be an admin.

```
app.get('/getalllogistics', authenticateToken, async (req, res) => {
    try {
        if (req.userInformation.status == "ADMIN" || req.userInformation.status == "LOGISTICS") {
            var allLogistics = await getLogistics()
            res.send(JSON.stringify(allLogistics))
        }
    } catch (err) {
        res.send(err)
    }
})
```

Most of the data processing is done on the backend and grouped into one single data strcuture to minimize the number of HTTP request as it improves user experience, but this could result in the page loading slower initially. For example, the following retrieves all the necessary orders/transactions for the logistics team, as well as their details (products/quantities) so that fewer HTTP requests will be needed when the details need to be viewed.

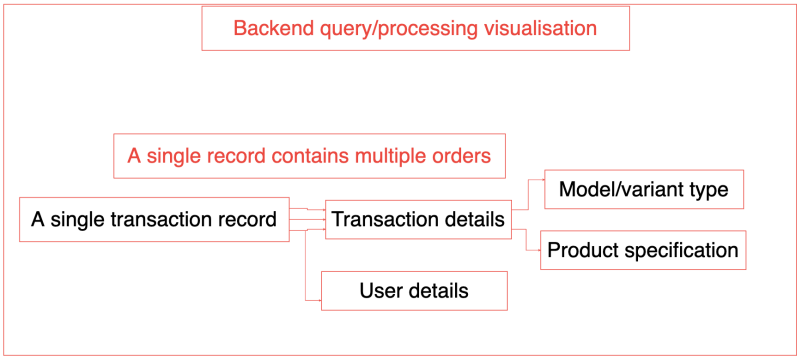
The function is called by the API above that initializes a database query and an array to store the data.

```
async function getLogistics() {
    return new Promise(async (resolve, reject) => {
        //Calls the function queryDb and continues executing other lines while waiting to get a response
        const allTransactions = await queryDb('SELECT * FROM `transaction` ORDER BY dateOrdered')
        var transactions = JSON.parse(JSON.stringify(allTransactions))
        var allTransactionsArray = [];
```

Orders/transactions are filtered to those that are relevant to the logistics department as following:

```
//Cycles through all the transaction records
for (transaction in transactions) {
    var allTransactionDetails = []
    //Filters the transaction to only records with the following status
    if (transactions[transaction].orderStatus == "Payment Approved" || transactions[transaction].orderStatus == "Processing" || transactions[transaction].orderStat
        var queryDetails = 'SELECT * FROM `transaction_details` where  idTransaction = ' + transactions[transaction].idTransaction
        var queryClient = 'SELECT * FROM `user_info` where  userID = ' + transactions[transaction].userID
        //Retrieve the details of the transactions + the details of the user so that logistics know where and who to send to
        const [clientInfo, details] = await Promise.all([queryDb(queryClient), queryDb(queryDetails)]);
        //parse the results
        var allClient = JSON.parse(JSON.stringify(clientInfo))
        var allDetails = JSON.parse(JSON.stringify(details))
```

For each of the transaction records that are retrieved, the details (product ordered, quantity, etc) are requested from the database.



```
    //For all the details retrieve the necessary product and variant information
    for (specificDetails in allDetails) {
        var queryProduct = 'SELECT * FROM `catalogue` where  productID = ' + allDetails[specificDetails].productID
        var queryVariant = 'SELECT * FROM `variant` where  idvariant = ' + allDetails[specificDetails].variantID
        const [productInformation, variantInformation] = await Promise.all([queryDb(queryProduct), queryDb(queryVariant)]);
        var allVariantInformation = JSON.parse(JSON.stringify(variantInformation))
        //store the query result in an object and and push it into the array of transaction details
        var completeInformation = {
            details: details[specificDetails],
            variant: allVariantInformation[0].variantName,
            products: {
                category: productInformation[0].category,
                name: productInformation[0].name
            }
        }
        allTransactionDetails.push(completeInformation)
    }
```

The retrieved data is returned in the form of a promise using resolve as it is asynchronous.

```
                //Final object of all the data required, transaction --> transaction details + product + variant + quantity
                var collectiveTransactions = {
                    transaction: transactions[transaction],
                    transactionDetails: allTransactionDetails,
                    user: {
                        name: allClient[0].name,
                        storename: allClient[0].storename,
                        address: allClient[0].address,
                        phonenum: allClient[0].phonenum
                    }
                }
                //For each record, there are multiple details and such --> stored in an array for each record
                allTransactionsArray.push(collectiveTransactions)
            }

        }
    }
    //Return the final result
    resolve(allTransactionsArray)
});
```

Frontend

XMLHttpRequest is a built-in browser library that can make HTTP requests using javascript. For the web application[5]. The requests are called based on the route defined in the backend.

```
var createXhrRequest = function (httpMethod, url, callback) {
    //Create a XML HTTP Request
    var xhr = new XMLHttpRequest();
    //The method accepts the http method (post,put,delete, get...) and the URL to be requested
    xhr.open(httpMethod, url);
    //When the request is complete uses a callback so the process can come back to retrieve the response
    xhr.onload = function () {
        callback(null, xhr.response);
    };
    //Called when a request wasnt successful
    xhr.onerror = function () {
        callback(xhr.response);
    };
    //Not necessary for get requests, but POST/PUT requires this
    xhr.send();
}
```

The following calls the previous function and passes the values (URL and HTTP method) into the createXHR request method so that the required data can be retrieved from the backend.

[5] “Using Xmlhttprequest - Web Apis: MDN,” Web APIs | MDN, accessed January 2, 2022, https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest.

```
createXhrRequest("GET", 'http://localhost:5001/getalllogistics' , function (err, response) {
    if (err) { console.log("Error!"); }
    allTransactions = JSON.parse(response);
    alltransactions = allTransactions
```

Nodemailer[6]

Sending email is essential in the web application and this is achieved by using a node js module called nodemailer which helps the communication between the email service and the application/browser. **Nodemailer has been used as it is a library integrated into nodeJS which makes it easy to set up and there is more control over every aspect of the emails to be sent as opposed to using third-party services to manage it.** However, third party services are more suitable for high volume sending as they are more specialized.

The following displays several credentials (refresh token, client ID, and client secret) that are required to access and interact with Gmail services.

```
let transporter = nodemailer.createTransport({
    service: "gmail",
    //Pass retrieved credentials into the auth section for authenticationto use the GMAIL service
    auth: {
        type: "OAuth2",
        user: process.env.EMAIL,
        pass: process.env.WORD,
        clientId: process.env.OAUTH_CLIENTID,
        clientSecret: process.env.OAUTH_CLIENT_SECRET,
        refreshToken: process.env.OAUTH_REFRESH_TOKEN,
    },
});
```

(Credentials are received by setting up a project in google cloud platform)

Once the query to insert a new record for the new user is completed, the object *mailOptions* is created which contains all the necessary information (email address, subject, text) for the email to be sent using the nodemailer library.

```
//Connect to the database and insert the following information
var dateCreated = moment().format("YYYY-MM-DD")
connection.query("INSERT INTO `user_info` SET `password`=?, `profilepic`=?,`email`=?, `phonenum`=?, `storename`=?, `address`=?, `status`=?, `
    if (error) throw error;
    //Define the email options so that the api knows where to send the email as well the content of the email
    let mailOptions = {
        from: process.env.EMAIL,
        to: req.body.email,
        subject: "New Account",
        text: req.body.text,
        //The email content which informs the user of their new account credentials
        html: `
    <p> Hi ${req.body.name},<p>  <br>
    <p>The following includes the login information for your new account. <p><br>
    <p> Email: ${req.body.email} <p> <br>
    <p> Password: ${password} <p> <br>
    <p> Thanks, <p> <br>
    <p> Torch ID <p>
    `
    };
    //Use the pre-defined transported module to send an email to the specified address using the gmail service
    transporter.sendMail(mailOptions, function (err, data) {
        if (err) {
            console.log(err)
        } else {
        }
    });
    res.send(JSON.stringify(results));
```

[6] Tomer Ben Rachel, “How to Use Nodemailer to Send Emails from Your Node.js Server,” freeCodeCamp.org (freeCodeCamp.org, April 29, 2021), https://www.freecodecamp.org/news/use-nodemailer-to-send-emails-from-your-node-js-server/.

Charts and analytics (using chart JS)[7]

To create dynamic analytic graphs and charts, **the chart JS library is used to represent the data as it supports animation and is compatible with most browsers**.

Backend operation

An API is called which retrieves an ordered list of daily statistics which includes daily total sales and total revenue in the last 30 days

```
var statsquery = 'SELECT * FROM `daily`  WHERE date >= current_date - 30 ORDER BY date'
```

iddaily	dailySales	dailyRevenue	date
1	34	2323	2021-11-26
2	3434	345	2021-11-27
3	43	345	2021-11-28
4	234	3453	2021-11-29
5	234	4545	2021-11-30
6	234	23	2021-12-01
7	234	345	2021-12-03
8	234	345	2021-12-02

```
//Initialize three arrays which contains the following information
var revenue = [];
var sales = [];
var date = [];

//Push all the retrieved data into the respective array
for (stat in parsedStatistics) {
    revenue.push(parsedStatistics[stat].dailyRevenue)
    sales.push(parsedStatistics[stat].dailySales)
    //Convert the date format into (YYYY-MM-DD)
    date.push(parsedStatistics[stat].date.split('T')[0])
}

var collectionOfAllStatistics = {
    revenue: revenue,
    sales: sales,
    date: date,
    customers: numberOfCustomers,
    products: numberOfProducts,
    transactions: numberOfTransactions,
    orderedProducts: sortedproducts,
    orderedCustomers: sortedCustomers
}
//Send the data as a response
res.send(collectionOfAllStatistics)
```

[7] Chartjs, “Introduction,” Introduction · Chart.js documentation, accessed January 2, 2022, https://www.chartjs.org/docs/2.9.4/.

Front-end javascript operation

The following is used to set up the graph options.

```
//Get the HTML element of a specific chart
var revenue = document.getElementById('chartrevenue').getContext('2d');
revenueChart = new Chart(revenue, {
    //Specify the type of chart
    type: 'line',
    //Pass the arrays into the data object
    data: {
        //Date array (x-axis)
        labels: allstats.date,
        //Revenue array (y-axis)
        datasets: [{
            label: "Revenue",
            backgroundColor: 'rgb(65, 240, 225)',
            borderColor: 'rgb(255, 99, 132)',
            //Pass the revenue array into the data section
            data: allstats.revenue,
        }]
    },
    options: {
        title: {
            display: true,
            text: 'Revenue chart (30 days)'
        },
        scales: {
            yAxes: [{
                scaleLabel: {
                    display: true,
                    labelString: 'Revenue'
                }
            }],
            xAxes: [{
                scaleLabel: {
                    display: true,
                    labelString: 'Date'
                }
            }]
        }
    }
});
```

The graph includes a zoom button that zooms into the graph by removing a certain number of elements from the data array, the options are 30 days, 15 days, and 7 days.

```
//clicked = 1 indicates that the user chooses the 15 days options and the array is selected between the index 14-29.
else if (clicked == 1) {
    //Uses a temporary data so that it doens't alter the original value
    var temporaryArray = allstats.revenue
    var temporaryDateArray = allstats.date
    revenueChart.data.labels = temporaryDateArray.slice(14, 29)
    revenueChart.options.title.text = "Revenue chart (15 days)"
    //Get the 15 most recent data (days)
    revenueChart.data.datasets[0].data = temporaryArray.slice(14, 29)
    revenueChart.update()
    //add click so that if the user clicks the zoom button again, the next option is selected for the graph
    clicked++;
}
```

Word count: 1022

Bibliography

Banke, Frederik. “Improving Scalability in C# Using Async and Await.” Medium. Medium, March 2, 2019. https://medium.com/@frederikbanke/improving-scalability-in-c-using-async-and-await-f97af1466922.

Chartjs. “Introduction.” Introduction · Chart.js documentation. Accessed January 2, 2022. https://www.chartjs.org/docs/2.9.4/.

email:r@rdegges.com], map[name:Randall Degges. “Randall Degges.” Randall Degges - Please Stop Using Local Storage. Accessed February 21, 2022. https://www.rdegges.com/2018/please-stop-using-local-storage/.

“Entity Framework Why EF Async Methods Are Slow.” zzz projects. Accessed January 2, 2022. https://entityframework.net/why-ef-async-methods-are-slow.

Kumar, Suresh. “How JSON Web Token(Jwt) Authentication Works?” Medium. Medium, March 20, 2021. https://medium.com/@sureshdsk/how-json-web-token-jwt-authentication-works-585c4f076033.

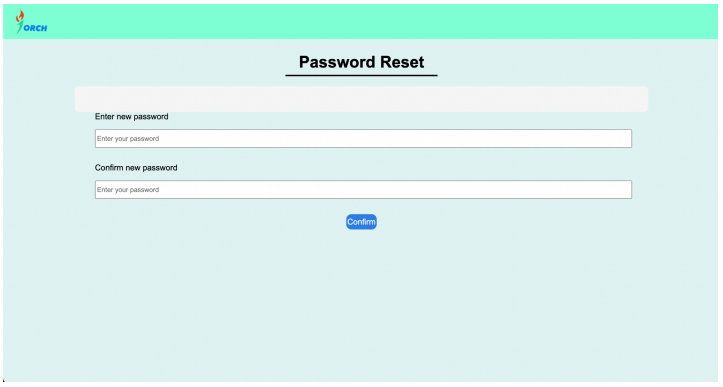
Rachel, Tomer Ben. “How to Use Nodemailer to Send Emails from Your Node.js Server.” freeCodeCamp.org. freeCodeCamp.org, April 29, 2021. https://www.freecodecamp.org/news/use-nodemailer-to-send-emails-from-your-node-js-server/.

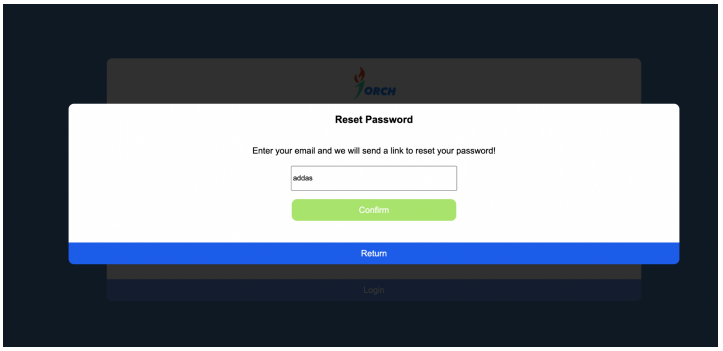
“Using Xmlhttprequest - Web Apis: MDN.” Web APIs | MDN. Accessed January 2, 2022. https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest.

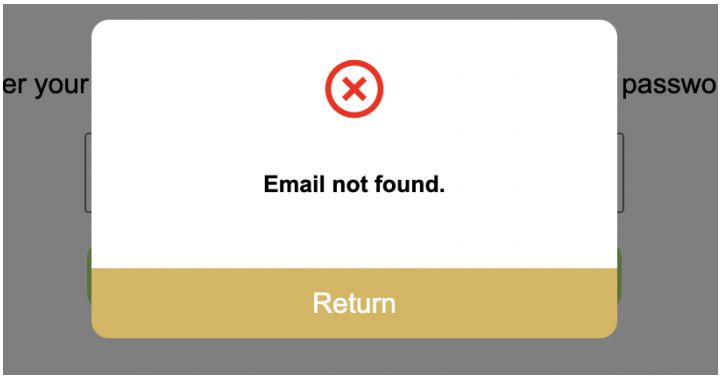
Evaluation

Refer to <u>Appendix B</u> for detailed client feedback)

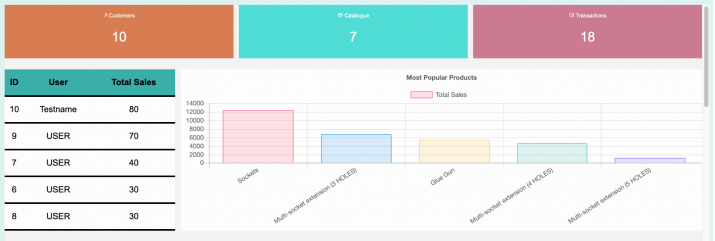
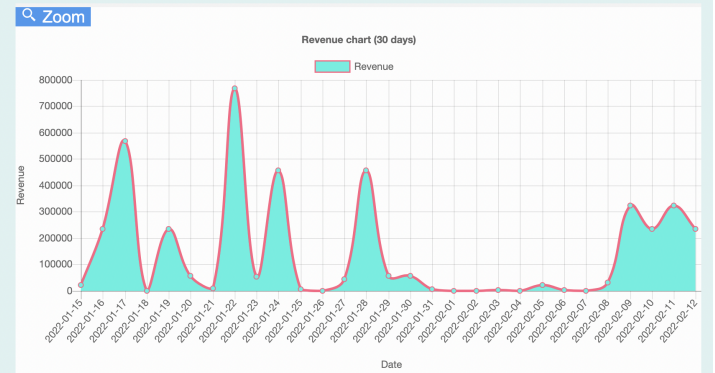
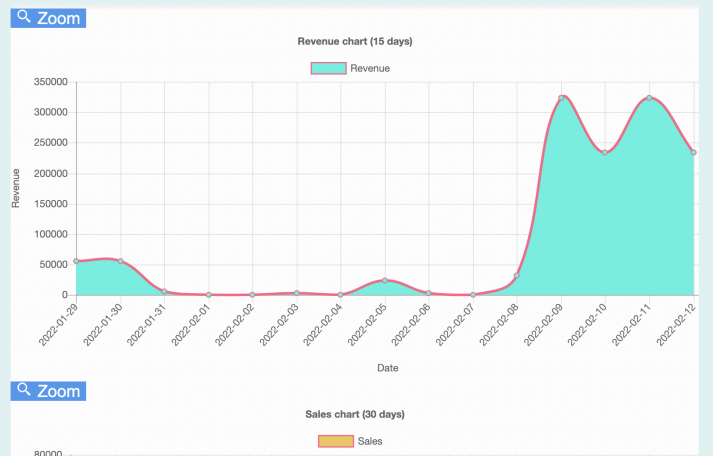
<table>
<tr><th>Criterion Number</th><th>Criterion for Success</th><th>Met?</th><th>Evaluation</th></tr>
<tr><td colspan="4">Authentication & Validation</td></tr>
<tr><td>1</td><td>Authentication features that limit access of the users based on their roles.</td><td>YES</td><td>Successful as users are granted different levels of access depending on their roles. Users could be notified of the reason they were prompted to the login page to avoid confusion.</td></tr>
<tr><td>2</td><td>All passwords should be hashed</td><td>YES</td><td> 

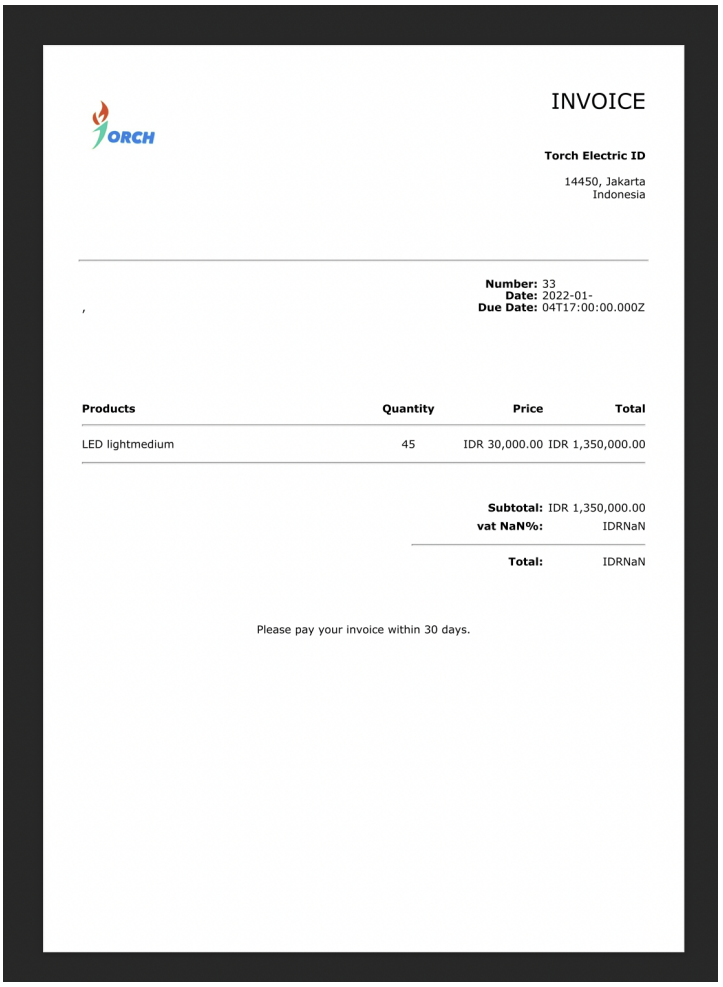
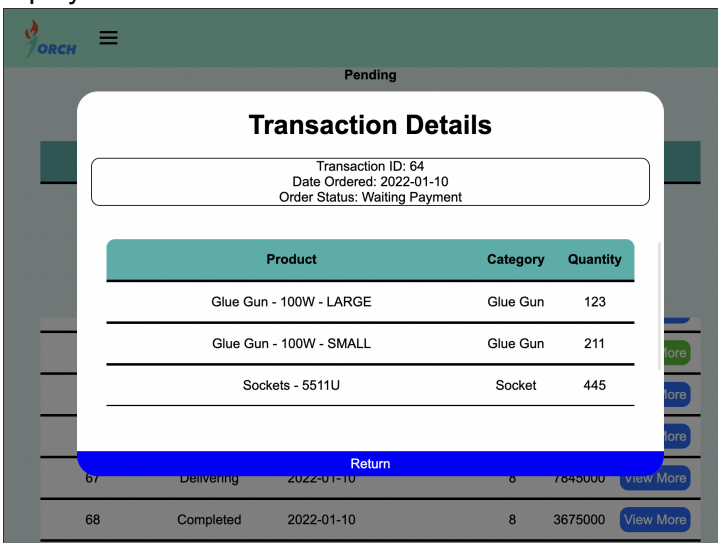
All passwords within the database/system are hashed to improve security.</td></tr>
<tr><td colspan="4">Resetting Password</td></tr>
<tr><td>3</td><td>User is able to reset password.</td><td>YES</td><td> 

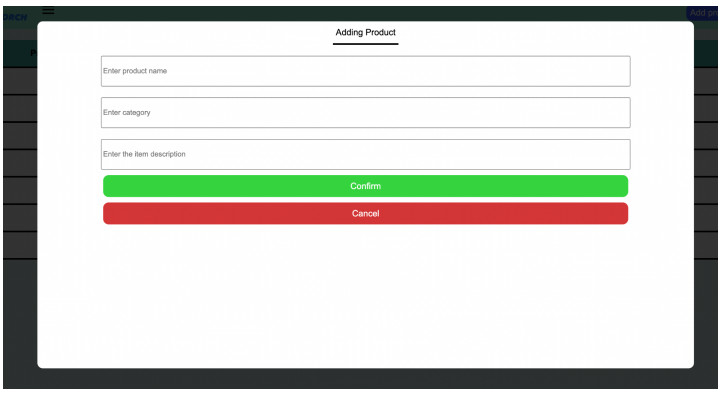


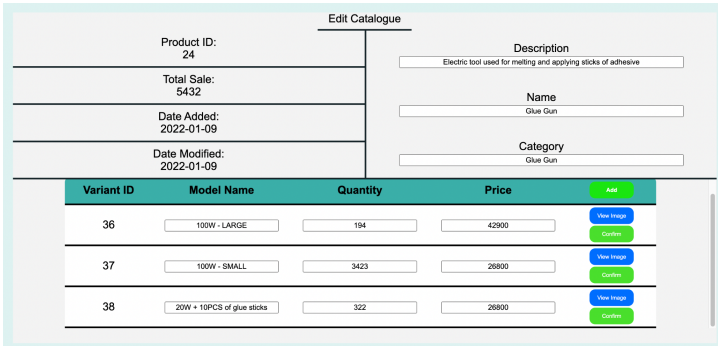
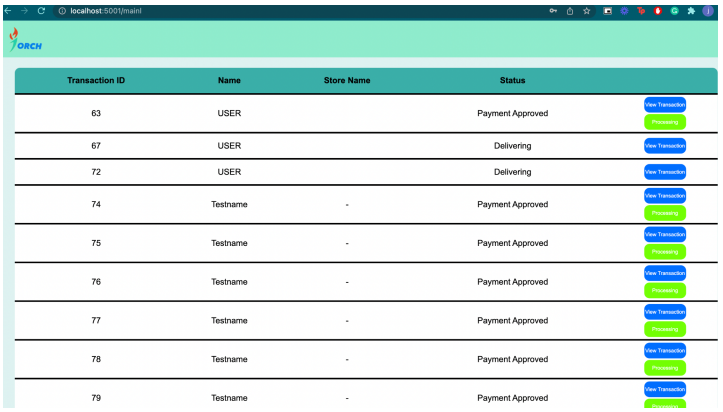


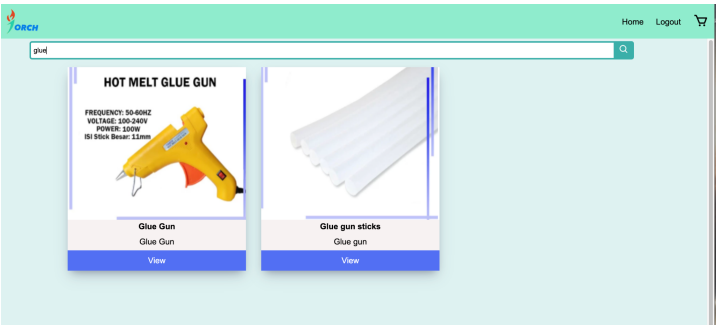
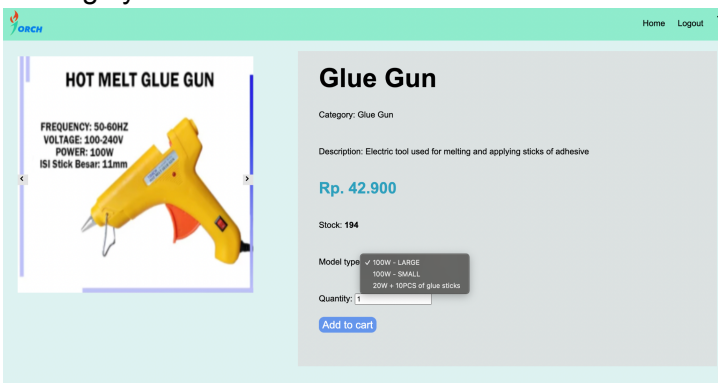
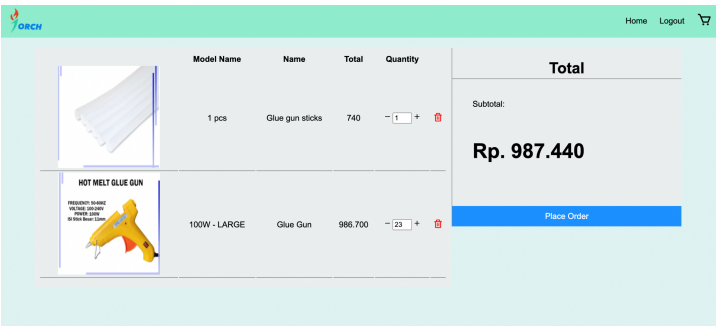
Client was satisfied as users can easily change</td></tr>
</table>

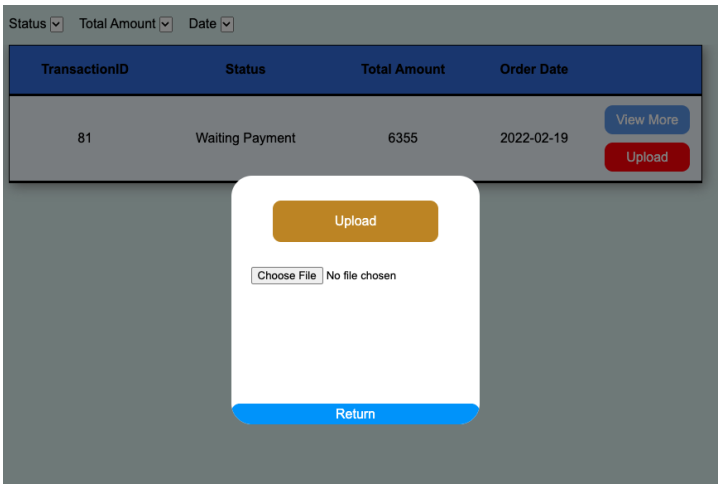
			passwords. Emails are sent containing a link to redirect the page.
Admin Role - Main Menu			
4	Admin should be able to view and manipulate the analytics.	YES	   The analytics was implemented well, but the client suggested that the zoom should allow the selection of a specific time period, rather than the most recent data values.
Admin Role - Managing Orders			
5	Admin should be able to accept an order that is waiting for approval.	YES	An email containing the invoice can be successfully sent when an order is accepted and updated by an admin.

			
6	Allow admin to sort and filter the table records.	YES	The feature improves record organization and the client didn't notice any noticeable lag or time delay.
7	View relevant details about each record/order		Client was satisfied as relevant transaction details can be displayed for each record. 
Admin Role - Managing Users			
8	Admin can search the users and filter	YES	The client noted that filtering and searching by name was very useful and no further suggestion was given.

	records based on names and email.		
9	Create a new user	YES	New users can be added easily, however, the client suggested that the address form can be more specific. 
10	Edit and delete users	YES	The admin is able to edit and delete users, but a confirmation popup can be included to prevent accidental changes.
Admin Role - Managing Transactions			
11	Admin can view and update all the pending transactions that need to be reviewed.	YES	The client liked the implementation of the transaction system and suggested that email notifications can be added to inform the clients of rejected and accepted payments.
Admin Role - Managing catalogue			
12	Admin can add product and edit the details.	YES	The client was satisfied with this feature where admins can add and edit product details as it closely follows their system. 

			 
Logistics Role - Updating order status			
13	View and update relevant records to the logistic department.	YES	The logistics department can view and update relevant orders. The client was satisfied as it resolves the communication problems between the employees. 
User Role - Ordering System			
14	View all the available products and search them by category and name.	YES	Functional and very little delay which improves usability.

			
15	Users should be able to view a product and change the variant/model of the product during browsing.	YES	Prices and images are changed when the user selects a different model for the same product. The client commented that it was really easy to navigate the ordering system. 
16	Allow the user to add items to cart and create a new order.	YES	Items can be added/deleted to cart with the selected quantity and users can create order with the selected products. 
User Role - Managing orders			
17	Users should be able to upload proof of payment for certain orders.	YES	Client was content with the features and no further suggestions were given.

Evaluation Word Count: 330

Recommendations

During the final interview (Refer to **Appendix B**), the client gave various recommendations which are expanded to create coherent additional features that can be used to improve the overall system.

1. **user-side product recommendation feature that primarily uses machine learning.**

   This feature can improve the web application as customers are recommended products that they are interested in, creating a passive promotion. However, python is more suitable for such projects due to its libraries and frameworks. Hence, something known as “child processes” within nodeJS can be used to run scripts from other languages asynchronously[1]. The python code itself feeds on past purchasing trends of the overall system and the specific user data to predict the type of products that the client’s clients might purchase. The difficulty lies in the fact that it is a business-business model and purchasing trends will differ from individual customers.

2. **Include stock re-ordering system involving suppliers**

   The client has expressed interest in an automatic stock replenishment system while he while giving suggestions on the catalogue system. This is will be very efficient for the business as they wouldn’t have to communicate directly with the suppliers overseas and stocks can be reordered automatically which is very convenient. Although an external stakeholder, the suppliers, have to be integrated into the system, this is highly achievable as no new technology is introduced.

Recommendation word count: 176

[1]“Run Python Script from Node.js Using Child Process Spawn() Method,” GeeksforGeeks, February 19, 2019, https://www.geeksforgeeks.org/run-python-script-node-js-using-child-process-spawn-method/.

Bibliography

“Run Python Script from Node.js Using Child Process Spawn() Method.” GeeksforGeeks, February 19, 2019. https://www.geeksforgeeks.org/run-python-script-node-js-using-child-process-spawn-method/.

Transcript - first interview (The Scenario) - Translated
Client Name: YouYu Xue
Contact details: xueyouyu168@gmail.com
Date of interview: 20th of june 2021
Place of interview: Client's office

Hello Mr. Xue, for our first interview we will be discussing the problem you are having.

Client: ok.

Mr. Xue, could you explain your job and briefly describe your business?

Client: Yes, I am an owner of a small to medium sized business that provides electrical goods like LED lights, Sockets, multi socket extension cord and a lot of other products to other businesses and I work with multiple manufacturers from china. My business deals with many clients around Indonesia and distributes those products to them.

I see, could you give a rough estimate as to how many clients you work with and are they large or small businesses?

Client: I would say around 100 clients and they are mostly small to medium resale shops.

Do you have any direct contact with your customers?

Client: No, most of them are smaller businesses.

You mentioned that you work with suppliers and manufacturers from overseas, how many do you work with at the moment?

Client: 7-8 suppliers and they are based in china, I also work with one that is based in Indonesia.

Could you explain in detail as to the processes that take place from your clients taking orders to the products being delivered?

Client: First, the clients contact us to place down orders and we check for availability from the warehouse, then we contact the clients to confirm their order, only if the product is still available. Once the order is confirmed, we pass the information to the warehouse so that the products can be processed and delivered. Our company policy only allows products to be sent when payments have been made.

You mentioned about information being sent to the warehouse, what kind of information do the personnel at the warehouse receive?

Client: Usually, the stock type and quantity are sent, including the customer/client name. The address is also sent so that the warehouse knows the destination of the delivery.

What problems are you facing with this system?

Client: Sometimes I am also in charge of taking orders from customers along with 4 to 5 other people in the office. As the number of our clients grows, it is becoming much more difficult to keep track and organize the clients and their orders. It is also very inefficient as the personnel at the warehouse have no clue about all these orders until we send the orders to them and we have to also wait for client payments which add up to the complexity of the system and the time it takes to process the orders.

How does it affect your business?

Client: Well, when the business is doing well, things will get hectic and there is a rare chance that we miss an order or forget to process it. Reputation is very important so if something like that occurs, it can be very damaging to the business. Also, a lot of the personnel in the office who are in charge of taking orders have other priorities in the office.

Are there any softwares or programs available on the market that might be able to solve this problem?

Client: Yes, there is a similar software from what I envision, however, I am looking for one that is suitable and specific to my needs.

So in summary, the current ordering and delivering system that you have is fully manual which is relatively inefficient and time consuming. Are you looking for a solution that will remove some of the processes that you just described and a system that will keep track of all the ordering records?

Client: Yes, that is what I am looking for.

Okay thank you for your time, I think there is sufficient information for me to start thinking about the solution. I will let you know of some ideas I have in the next interview.

Client: Thank you.

Date Signed:

13/02/2022

Transcript - second interview (The Solution) - Translated
Client Name: YouYu Xue
Contact details: xueyouyu168@gmail.com
Date of interview: 27th of June 2021
Place of interview: Client's office

Hi Mr. Xue, thank you for agreeing to another interview. Today we will be discussing the details for your solution. In the last interview, you mentioned that you are looking for a way to make your ordering system more efficient, is that correct?

Client: Yes, that is right.

After reviewing your situation and processes, I think a website would be more suitable as opposed to an app. Mr. Xue, how many clients do you have at the moment and how often do you add new clients?

Client: I have about 100 clients and every so often I will get new clients, maybe two to three every month.

I have taken that into consideration and requesting all 100 of your clients to download an app might be slightly extensive. I decided that a website is much more suitable because it is compatible with more devices like phones, computers and laptops. A website will allow them to switch devices without downloading the app on both devices which might be slightly more convenient for your clients. What devices are most commonly used in your business?

Client: I usually work on my laptop, but for contacting customers, I rely on phones and emails.

Is that also true for all your workers?

Client: Yeah, most of them use a mix of both laptops and mobile phones.

I see, so in your case an app might be too inconvenient. I was thinking about a website which will grant different types of access depending on their status, such as customers or workers. Once the customer orders, the purchase details are sent to the office branch to be processed and once the payment is completed, the order can be ticked and it will be accessible to the warehouse to be processed. That is just a rough idea, how does that sound to you?

Client: I might request for additional features, but at the moment this sounds good to me.

Okay now that we have established the platform for the solution, I need a little more information so that I can construct a basic framework for how the website will work. I will be configuring the website to a specific database which will store data about clients, history of orders, stocks and their quantities. Starting with the clients, what

data do you think will be important to be stored?

Client: it's important that their name is stored along with their addresses so that we can identify the clients and where they are located.

Anything else that will be useful?

Client: Maybe their contact information, like email and phone numbers.

Okay, since the website should allow customers to order products, the database needs to store data about the products. What information do you think is necessary?

Client: We identify products by name and a code behind it, so that will be useful for us to identify the goods during each stage of the process. I think a stock quantity is definitely necessary, maybe we can get some sort of alert when the product needs to be restocked. I think it is important that the stock quantity is hidden from clients as information like that can influence their decision to purchase rather than focusing on the product itself

Noted. As for the orders themselves, what information do you typically send between the warehouse and your office?

Client: Date of order, stock quantity, the product types, and usually the customer information that I mentioned.

Perfect. I think I have some basic idea of how your website will operate, is there anything else that you would like to be included in the website?

Client: Maybe some sort of searching or filtering that can help us find ordering records quicker. Because we currently don't have a way to quickly find information like that as we store them both digitally and in paper form. I might need a little more time to think about it.

Sure. Please let me know when you think of any additional features that will be useful for the solution. Thank you for your time, Mr. Xue. That will be all for today's consultation, I will be contacting you in regards to any problems I encounter, is that alright with you?

Client: Yes, that is fine with me.

Okay, thank you.

Client: No problem.

Date Signed:

13/02/2022

Transcript - Third interview (Final Interview) - Translated
Client Name: YouYu Xue
Contact details: xueyouyu168@gmail.com
Date of interview: 28th of January 2021
Place of interview: Client's office

The client was presented with the success criteria and the test plan for each of the criteria. His feedback was asked for each of the success criteria (Some of them were conducted with my personal help)

Testing page routing upon login

Criterion 1: Authentication features that limit access of the users based on their roles.

So you can see that when you login as different users with various user roles, they are brought to different main pages, any thoughts on this?

Client: I don't see any further improvements needed, this was done very well and I can see that they are brought to different pages.

As you have just seen, some of the web pages are restricted so that it is only available to certain user roles, what was your overall impression of this feature?

Client: I see, so some of the pages are only available based on their role in the web application. However, I think this can be improved by notifying the person of why they are brought to the login page so that they understand why they aren't able to access the pages. If I wasn't informed of this functionality, I might have been confused.

Resetting Password

Criterion 2: User is able to reset password.

Client: It works fine, I like that the web application makes sure that an email exists in the database before sending an email.

Main Menu (ADMIN) - functionality

Criterion 4: Admin should be able to view and manipulate the analytics

Client: So where does the system get all these data?

Essentially, whenever one of your clients fulfils a payment, it is counted towards the daily sales and revenue which can be found within the database. Then, whenever the admin main page loads, the data is retrieved from the database.

Client: I see. Well, I really like that you can zoom in to see specific graphs, it is also very colourful which makes it very pleasing to look at. I do have some suggestions to improve it,

for example, maybe you could allow the users (admin) to select specific time periods, rather than the most recent ones. Recently, I saw a feature on a website that allowed me to select the time periods dynamically, but overall this is sufficient and I don't see any problems.

Managing orders (ADMIN)

Criterion 5: Admin should be able to accept an order that is waiting for approval.

Once the order has been approved by an admin or employee, an email is sent. What are your thoughts on this feature?

Client: The button is working and I can see that the order has been updated, but I think this is a bit too slow. You could try to shorten the time difference between pressing the button and the execution of updating the records.

What did you think of the email and invoice functionality?

Client: The emailing of the invoice is exactly what I was looking for — I particularly like the structure of the invoice.

Criterion 6: Allow admin to sort and filter the table records.

What did you think of the sorting and filtering functionality, specifically the speed and performance?

Client: The sorting and filtering of records are fast and I don't see any noticeable lag or delay.

Criterion 7: View relevant details about each record/order

Are all the necessary information displayed when you click to view more for each record?

Client: Yes, I like that you can also view each and every product ordered, I don't see any other details that need to be added. This is fine.

Overall Impression:

What do you think about the tables for storing records and the way the page is structured?

Client: This page is very easy to follow, I like that the tables are split into two sections which makes it organized and very intuitive to use. The best thing about these tables and functionality is that we can easily keep track of all the processes and orders, which will make communication between clients and our business, as well as internally, much easier.

What is your impression with the flow of the ordering system? Does it follow the system which your business uses?

Client: I am very impressed with this as it closely follows the system which we employ in the business, some possible suggestions would be adding functionality which can notify the users if they are late in paying what is required.

Managing users (ADMIN)

Criterion 8: Admin can search the users and filter records based on names and email.

What are your thoughts on filtering and sorting users? Any performance or speed issues?

Client: Filtering the users by name and email is very useful as we can have up to hundreds of clients with different names and emails. Similar to my previous comment, the functionality works as intended and I don't see any noticeable delay.

Criterion 9: Create a new user

I designed the creation of users to be as intuitive and as easy to learn as possible, what are your thoughts on this? Any suggestions that can help improve the process for your employees?

Client: As for creating the users, I don't see any major improvements needed, but the address form could perhaps be improved so that it includes the specific street and city, which would be very helpful for the logistics department.

The password you see here was generated randomly, rather than using the same uniform passwords for every new user created, which would not be ideal for security reasons.

Client: I don't have much to say about this, I can see that the email contains the login details. I particularly like that you generated unique passwords for every new user.

Criterion 10: Edit user and delete user

What did you think about the functionality of deleting and editing users?

Client: Deleting and editing users is also very quick and easy to navigate, so I think the employees wouldn't have a hard time getting used to the features. I think it would very helpful to also add a confirmation button, just in case the button was accidentally pressed.

Managing transactions (ADMIN)

Criterion 11: Admin can view and update all the pending transactions that need to be reviewed.

I tried to model your transaction system and made the system more convenient as direct contact with the clients are not needed, do you think that it achieved the intended purpose and would it fit well into your system?

Client: This works as intended, I like that you included the reject and accept payments option which could make our system a lot more efficient as we don't need the clients to be available to inform them of rejected payments. I think you could add a feature where it sends the client an email when a payment has been rejected or accepted, so if they don't log on to the system, they are still notified of this.

Managing catalogue (ADMIN)

Criterion 12: Admin can add product and edit the details.

Client: I think you have done a very excellent job following our product classification, adding a new model is very easy and I like that you can change prices very easily which reflects our business system where we often change the prices of products very frequently. A suggestion that I can give is sending an alert when the stock amount falls below a certain amount, so we can place an order from the suppliers. But, this system is well implemented and the page is very easy to navigate.

I know that very often your business has several variations for the same product so I added a feature that allows your employees to add a new product variation, such as colour or size, what are your overall thoughts on this?

Client: This is what I was looking for, I think adding new product variation is very easy and I am sure the employees would find it very easy to navigate around this. Overall, adding new product and their variations is an excellent feature of your final product and it follows our product classification system very well.

View the necessary orders (LOGISTICS)

Criterion 13: View and update relevant records to the logistic department.

Client: Everything works fine, I don't have any further improvements or suggestions to give. I think this should solve the connection problems between our departments as they are at different locations. I think you have done well as this will remove a layer of communication in terms of receiving new orders.

Ordering System (USER)

Criterion 14: View all the available products and search them by category and name.

Client: The filtering of products by searching is almost instantaneous, so I don't have a lot of issues with that. I like that you didn't just limit it to name, but also the category which should be very useful.

Criterion 15: Users should be able to view a product and change the variant/model of the product during browsing.

When the users click on a specific product, they can change the model type which should make browsing a lot easier for your clients. Rather than having a bunch of products with each of their variants shown on the front page which should take up a lot of space. What are your thoughts on this?

Client: I really like that the users can change the model type of the product they are viewing, as well as the images, which makes it a lot easier for the users to navigate the products while purchasing.

Criterion 16: Allow the user to add items to the cart and create a new order.

Client: Not a lot of comments to give here. I can see that the product is successfully added to the cart when I click the button. Changing the total price based on the quantity selected is an excellent feature of the cart system, everything works very smoothly and my clients should be very happy with this type of ordering system.

What is your overall impression of the product ordering feature?

Client: Overall, I think you captured our system from the business to client sides very well and this web application should make the processes a lot easier and more efficient compared to our current system.

Managing Orders (USER)

Criterion 17: Users should be able to upload proof of payment for certain orders.

Client: This works as expected, I don't have any suggestions to give.

Date Signed:

13/02/2022

Appendix C - code (CSS files not included - can be found in product folder)

Database.js - Establish a connection between nodeJs and mysql

```
const mysql = require('mysql');

//Source: youtube.com/watch?v=hGZX_SA7lYg - setting up database and connection to
nodeJS
var connection = mysql.createConnection({
   host: "localhost",
   connectionLimit:10,
   user: "root",
   //.env indicates that the password was stored locally
   password: process.env.DATABASE_PASSWORD,
   database: "order",
   port: 3306
})

module.exports = connection
```

Server.js - stores all the available routes and functions; essentially connects front-end and back-end and coordinates activities within the web application

```
//Initializations + dependencies to access certain libraries in nodeJs
const express = require('express');

var app = express();
var bodyParser = require('body-parser');

require("dotenv").config();



const pt = process.env.PORT || 5001;
var connection = require('./database');
const path = require('path');



var multer = require('multer')
const nodemailer = require("nodemailer");
const jwt = require('jsonwebtoken');
const cookieParser = require('cookie-parser');
var moment = require('moment');
const cron = require('node-cron');
const bcrypt = require('bcryptjs');
const Crypto = require('crypto')
var easyinvoice = require('easyinvoice');
var fs = require('fs');
const { reset } = require('browser-sync');
const e = require('express');
const { resolve } = require('path');
```

```
const { query } = require('express');


app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.use(cookieParser())
app.use(express.static('public'))

//Relative route to css/javascript/images
app.use('/css', express.static(__dirname + 'public/css'))
app.use('/js', express.static(__dirname + 'public/js'))
app.use('/images', express.static(__dirname + 'public/images'))

//All general functions/queries needed in the routes

//Function that takes in a query and returns a promise
//source:
https://stackoverflow.com/questions/60928216/how-can-i-make-my-node-js-mysql-connec
tion-as-a-promise-work
function queryDb(query) {
   return new Promise((resolve, reject) => {
       //Creates a connection to the database using connection.
       connection.query(query, (err, result) => {
           if (err) {
               return reject(err);
           }
           //Returns the result by resolving in the form of a promise
           resolve(result);
       });
   })
}


function queryDbWithId(query, newProduct) {
   return new Promise((resolve, reject) => {
       connection.query(query, newProduct, (err, result) => {
           if (err) {
               return reject(err);
           }
           resolve(result.insertId);
       });
   })
}

// Source:
https://www.youtube.com/watch?v=mbsmsi7l3r4&t=1052s&ab_channel=WebDevSimplified -
setting up jwt token functionality
```

```
// Source:
https://www.youtube.com/watch?v=894seNhONF8&ab_channel=AlextheEntreprenerd - why
use http only cookies
//
Source:https://keikaavousi.medium.com/nodejs-authentication-with-jwt-and-cookies-3f
b1c8c739ba auth cookie setup

//Middleware used to authenticate the user's access token or refresh token
function authenticateToken(req, res, next) {
   //Retrieve the access token from the http only cookie
   const accessToken = req.cookies.accessToken
   //If there is no access token then the user is not logged in --> redirects to
the login page
   if (accessToken == null) {
       return res.redirect('/login')
   } else {

       //Verify that the token is valid by passsing in the token secret and the
accessToken
       jwt.verify(accessToken, process.env.ACCESS_TOKEN_SECRET, (err,
userInformation) => {
           if (err) {
               //If there is an error, call a function to generates a new access
token as there is a token but it is invalid
               var newAccessToken =
generateNewAccessToken(req.cookies.refreshToken)
               //If there is no refresh token it might have been expired -->
redirect to the login page so that users can
               //receive a new token
               if (newAccessToken.newToken == null) {
                   return res.redirect('/login')
               }
               //Otherwise store the decoded userInformation within the JWT token
into req.userInformation which can be access by the requesting api
               else {
                   req.userInformation = newAccessToken.user
                   next()
                   //Also storest the new access token
                   return res.cookie('accessToken', newAccessToken.newToken, {
maxAge: 100000000000, httpOnly: true })
               }
               //This is called if the jwt access token is valid
           } else if (userInformation) {
               req.userInformation = userInformation
               next()
           }
       })
   }
}
```

```
// Source:
https://www.youtube.com/watch?v=mbsmsi7l3r4&t=1052s&ab_channel=WebDevSimplified -
setting up jwt token functionality
function generateNewAccessToken(refreshToken) {
   if (refreshToken == null) {
       return null
   }
   var newToken
   var userInformation
   jwt.verify(refreshToken, process.env.REFRESH_TOKEN_SECRET, (err, returnedResult)
=> {
       if (err) return res.sendStatus(403)
       const user = {
           userID: returnedResult.userID,
           status: returnedResult.status
       }
       var newaccess = generateAccessToken(user)
       newToken = newaccess
       userInformation = returnedResult
   })
   return {
       newToken: newToken,
       user: userInformation
   }
}

// Source:
https://www.youtube.com/watch?v=mbsmsi7l3r4&t=1052s&ab_channel=WebDevSimplified -
setting up jwt token functionality
function generateAccessToken(userInformation) {
   return jwt.sign(userInformation, process.env.ACCESS_TOKEN_SECRET, { expiresIn:
'10s' })
}

// Source:
https://www.youtube.com/watch?v=mbsmsi7l3r4&t=1052s&ab_channel=WebDevSimplified -
setting up jwt token functionality
function generateRefreshToken(userInformation) {
   return jwt.sign(userInformation, process.env.REFRESH_TOKEN_SECRET, { expiresIn:
'7d' })

}

function generateStatusToken(requestedEmail, requestedPassword) {
   //Use a promise keyword to indicate that it is used for async function
   //Return keyword for an async function
   return new Promise(async (resolve, reject) => {
       //QueryDb is a wrapper that takes in a query and resolves the results stored
in the following variable
       const users = await queryDb('SELECT * FROM user_info')
```

```
        //Convert the returned value in a readable format
        var allUsers = JSON.parse(JSON.stringify(users));
        //Initialize a status as not found
        var status = "Not Found"
        //Cycle through all the users and find one that matches with the requested
password/email
        for (specificUser in allUsers) {
            //Condition: if password matches with the email
            //source:
https://www.youtube.com/watch?v=Ud5xKCYQTjM&ab_channel=WebDevSimplified
            if (requestedEmail == allUsers[specificUser].email && await
bcrypt.compare(requestedPassword, allUsers[specificUser].password)) {
                //Create a user object container the ID and status
                const user = {
                    userID: allUsers[specificUser].userID,
                    status: allUsers[specificUser].status
                }
                //Determine if the user's role/status
                if (allUsers[specificUser].status == "USER") {
                    status = "USER"
                }
                else if (allUsers[specificUser].status == "ADMIN") {
                    status = "ADMIN"
                }
                else if (allUsers[specificUser].status == "LOGISTICS") {
                    status = "LOGISTICS"
                }
                //Calls a function to generate the token
                // Source:
https://www.youtube.com/watch?v=mbsmsi7l3r4&t=1052s&ab_channel=WebDevSimplified -
setting up jwt token functionality
                var accessToken = generateAccessToken(user);
                var refreshToken = generateRefreshToken(user)
            }
        }
        //Resolve results in the form of an object
        resolve({
            status: status,
            accessToken: accessToken,
            refreshToken: refreshToken
        });
    });
}


function editmodel(variantName, quantity, price, variantID) {
    return new Promise((resolve, reject) => {
        connection.query("UPDATE `variant` SET
`variantname`=?,`quantity`=?,`price`=? where `idvariant`=?", [variantName,
quantity, price, variantID], function (error, results, fields) {
```

```
            resolve(JSON.parse(JSON.stringify(results)));
        });
    });
}

//Source:
https://medium.com/swlh/how-to-upload-image-using-multer-in-node-js-f3aeffb90657  -
uploading files into a server
var storage = multer.diskStorage({
    destination: function (req, file, cb) {
        cb(null, './public/images')
    },
    filename: function (req, file, cb) {
        cb(null, file.originalname)
    }
})

const upload = multer({ storage: storage });


//Automatically insert a new date every 12 AM
cron.schedule('0 0 0 * * *', async () => {

    var query = 'INSERT INTO `daily` SET date=?' + moment().format("YYYY-MM-DD")
    var empty = await queryDb(query)

});

//Routes to retrieve HTML files
app.get('/login', (req, res) => {
    res.sendFile(path.join(__dirname, './HTML', 'login.html'))
})

app.get('/main', authenticateToken, (req, res) => {
    res.sendFile(path.join(__dirname, './HTML', 'mainmenu.html'))
})

app.get('/maina', authenticateToken, (req, res) => {
    if (req.userInformation.status == "ADMIN") {
        res.sendFile(path.join(__dirname, './HTML', 'mainmenua.html'))
    }
    else {
        res.redirect('/login')
    }
})

app.get('/mainl', authenticateToken, (req, res) => {
    if (req.userInformation.status == "LOGISTICS" || req.userInformation.status ==
"ADMIN") {
        res.sendFile(path.join(__dirname, './HTML', 'mainl.html'))
```

```
    }
    else {
        res.redirect('/login')
    }
})

app.get('/orderhist', authenticateToken, (req, res) => {
    res.sendFile(path.join(__dirname, './HTML', 'orderhist.html'))
})

app.get('/delivery', authenticateToken, (req, res) => {
    res.sendFile(path.join(__dirname, './HTML', 'delivery.html'))
})


app.get('/usersdash/admin', authenticateToken, (req, res) => {
    if (req.userInformation.status == "ADMIN") {
        res.sendFile(path.join(__dirname, './HTML', 'usersa.html'))
    }
    else {
        res.redirect('/login')
    }
})

app.get('/usersdash/admin/:userID', authenticateToken, (req, res) => {
    if (req.userInformation.status == "ADMIN") {
        res.sendFile(path.join(__dirname, './HTML', 'edituser.html'))
    }
    else {
        res.redirect('/login')
    }
})

app.get('/catalogue/admin', authenticateToken, (req, res) => {
    if (req.userInformation.status == "ADMIN") {
        res.sendFile(path.join(__dirname, './HTML', 'cataloguea.html'))
    }

})

app.get('/orders/admin', authenticateToken, (req, res) => {
    if (req.userInformation.status == "ADMIN") {
        res.sendFile(path.join(__dirname, './HTML', 'ordersa.html'))
    }
    else {
        res.redirect('/login')
    }
})

app.get('/catalogue/admin/:catID', authenticateToken, (req, res) => {
```

```
    if (req.userInformation.status == "ADMIN") {
        res.sendFile(path.join(__dirname, './HTML', 'editcatalogue.html'))
    }
    else {
        res.redirect('/login')
    }
})

app.get('/transactions/admin', authenticateToken, (req, res) => {
    if (req.userInformation.status == "ADMIN") {
        res.sendFile(path.join(__dirname, './HTML', 'transactionsa.html'))
    }
    else {
        res.redirect('/login')
    }
})

app.get('/cart', authenticateToken, (req, res) => {
    res.sendFile(path.join(__dirname, './HTML', 'cart.html'))
})

app.get('/catalogue', authenticateToken, (req, res) => {
    res.sendFile(path.join(__dirname, './HTML', 'catalogue.html'))
})

app.get('/reset/:userID', (req, res) => {
    res.sendFile(path.join(__dirname, './HTML', 'resetpass.html'))
})

app.get('/productinfo/:pID', authenticateToken, (req, res) => {

    res.sendFile(path.join(__dirname, './HTML', 'product.html'))
})

app.get('/edituser', authenticateToken, (req, res) => {
    res.sendFile(path.join(__dirname, './HTML', 'useredit.html'))
})

//Routes to get all the users from database
app.get('/users', authenticateToken, async (req, res) => {
    if (req.userInformation.status == "ADMIN") {
        var userInformation = await queryDb('SELECT * FROM user_info')
        var allUsersInformation = JSON.parse(JSON.stringify(userInformation))
        var userList = []
        for (user in allUsersInformation) {
            //only retrieve the necessary information
            var necessaryUserInformation = {
                userID: allUsersInformation[user].userID,
                name: allUsersInformation[user].name,
                email: allUsersInformation[user].email,
```

```
                sales: allUsersInformation[user].totalSales,
                status: allUsersInformation[user].status,
            }
            userList.push(necessaryUserInformation)
        }
        res.send(userList)
    }
    else {
        res.send("Not Available")
    }
})

//Route to retrieve the user profile picture
app.get('/getuserprofilepic', authenticateToken, async (req, res) => {
    try {
        var specificUser = await queryDb('SELECT * FROM `user_info` where  userID =
' + req.userInformation.userID)
        var alluserInformation = JSON.parse(JSON.stringify(specificUser))

        var userInformation = {
            imageAddress: alluserInformation[0].profilepic
        }
        res.send(userInformation)
    }
    catch (err) {
        res.send(err)
    }
})

//Route to retrieve the user information of a specific user by ID
app.get('/getuserinfo', authenticateToken, async (req, res) => {
    try {
        var specificUserInformation = queryDb('SELECT * FROM `user_info` where
userID = ' + req.userInformation.userID)

        var allUserInformation = JSON.parse(JSON.stringify(specificUserInformation))
        var userInformation = {
            name: allUserInformation[0].name,
            email: allUserInformation[0].email,
            image: allUserInformation[0].profilepic,
            address: allUserInformation[0].address,
            storename: allUserInformation[0].storename,
            phonenum: allUserInformation[0].phonenum,
        }
        res.send(userInformation)
    }
    catch (err) {
        res.send(err)
    }
})
```

```
//Route to retrieve the user information and their transaction records for admins
app.get('/users/:userID', authenticateToken, async (req, res) => {
   try {
       if (req.userInformation.status == "ADMIN") {
           const [specificUser, allTransactions] = await
Promise.all([queryDb('SELECT * FROM `user_info` where  userID = ' +
req.params.userID), queryDb('SELECT * FROM `transaction` where  userID = ' +
req.params.userID)]);
           var userAndTransactions = {
               specificUser: specificUser,
               allTransactions: allTransactions
           }
           res.send(userAndTransactions)
       }
       else {
           res.send('No access')
       }
   }
   catch {
       res.send('something went worng')
   }
})

//Route to determine the most popular product and recently released product for
user front page.
app.get('/promotion', authenticateToken, async (req, res) => {
   const [catalogueDate, catalogueSales] = await Promise.all([queryDb('SELECT *
FROM `catalogue` ORDER BY dateAdded DESC'), queryDb('SELECT * FROM `catalogue`
ORDER BY totalSales DESC')]);
   //Product is ordered by the date that they were added in
   var parsedDate = JSON.parse(JSON.stringify(catalogueDate))
   var parsedSales = JSON.parse(JSON.stringify(catalogueSales))
   var promotionList = [];
   var promotion = [];
   promotionList.push(parsedDate[0])
   promotionList.push(parsedSales[0])
   for (var x = 0; x < 2; x++) {
       //Select a random image of the variant to be displayed
       var image = await queryDb('SELECT * FROM `images` where productID=' +
promotionList[x].productID)
       var imageDetails = JSON.parse(JSON.stringify(image))
       var imageInformation = {
           product: promotionList[x],
           image: imageDetails[0]
       }
       promotion.push(imageInformation)
   }
   res.send(promotion);
})
```

```
//The getUserTransactionDetails function is also marked async as it uses functions
that returns promises
async function getUserTransactionDetails(requestedUserId) {
   //Resolves is used to return a value in the form of a promise
   return new Promise(async (resolve, reject) => {
       //Calls the function queryDb and continues executing other lines while
waiting to get a response
       const allTransactions = await queryDb('SELECT * FROM `transaction` where
userID = ' + requestedUserId)
       var transactions = JSON.parse(JSON.stringify(allTransactions))
       //Cycles through all the transaction records
       var collectionOfTransactions = []
       for (transaction in transactions) {
           //Filters the transaction to only records with the following status
           var details = await queryDb('SELECT * FROM `transaction_details` where
idTransaction = ' + transactions[transaction].idTransaction)
           var allTransactionDetails = JSON.parse(JSON.stringify(details))
           var collectionOfTransactionsDetails = [];

           for (specificDetails in allTransactionDetails) {

               const [productInformation, variantInformation] = await
Promise.all([queryDb('SELECT * FROM `catalogue` where  productID = ' +
allTransactionDetails[specificDetails].productID), queryDb('SELECT * FROM `variant`
where  idvariant = ' + allTransactionDetails[specificDetails].variantID)]);

               var fullinfo = {
                   details: allTransactionDetails[specificDetails],
                   products: productInformation[0],
                   variant: variantInformation[0]
               }
               collectionOfTransactionsDetails.push(fullinfo)
           }
           var alltransc = {
               transaction: transactions[transaction],
               details: collectionOfTransactionsDetails
           }
           collectionOfTransactions.push(alltransc)
       }
       resolve(collectionOfTransactions)
   });
}

//using async can allow returning of value outside of the function
//Source:
https://stackoverflow.com/questions/63103436/cannot-push-object-returned-from-mysql
-in-array

//route to retrieve all the onoging transactions/orders from a specific user
```

```
app.get('/getdelivery', authenticateToken, async (req, res) => {
   try {
       //Use the await keyword to indicate to the compiler to hold off on the
execution until a promise is returned
       //Results are stored in the variable alltranscdet
       var allTransactionDetails = await
getUserTransactionDetails(req.userInformation.userID)
       //Send the converted reesponse
       res.send(JSON.parse(JSON.stringify(allTransactionDetails)))
   } catch (err) {
       res.send('something went wrong')
   }
})

//Authtoken to validate the user through tokens is called before the api proceeds
with any code
app.get('/getalldashboard', authenticateToken, async (req, res) => {
   try {
       //Use the decoded userInformation to access the user role and determine
whether they have access to it
       if (req.userInformation.status == "ADMIN") {
           //get the stats for the last 7 days
           var statsQuery = 'SELECT * FROM `daily`  WHERE date >= (CURDATE() -
INTERVAL 1 MONTH )'
           var customerQuery = 'SELECT * FROM `user_info`  WHERE `status` =  "user"
ORDER BY totalSales desc'
           var productQuery = 'SELECT * FROM `catalogue` ORDER BY totalSales desc'
           var transactionQuery = 'SELECT * FROM `transaction`'
           //Query to gather all the necessary statistics
           const [allStats, allCustomers, allProducts, allTransactions] = await
Promise.all([queryDb(statsQuery), queryDb(customerQuery), queryDb(productQuery),
queryDb(transactionQuery)]);
           var parsedStatistics = JSON.parse(JSON.stringify(allStats))
           var numberOfCustomers = allCustomers.length
           var orderedCustomers = JSON.parse(JSON.stringify(allCustomers))
           var sortedCustomers = []
           for (customer in orderedCustomers) {
               var customerInformation = {
                   userID: orderedCustomers[customer].userID,
                   name: orderedCustomers[customer].name,
                   totalsales: orderedCustomers[customer].totalSales
               }
               sortedCustomers.push(customerInformation)
           }
           var numberOfProducts = allProducts.length
           var orderedProducts = JSON.parse(JSON.stringify(allProducts))
           var sortedproducts = []
           for (product in orderedProducts) {
               var productinfo = {
                   name: orderedProducts[product].name,
```

```
                    totalsales: orderedProducts[product].totalSales
                }
                sortedproducts.push(productinfo)
            }

            var parsedStatistics = JSON.parse(JSON.stringify(allStats))

            var numberOfTransactions = allTransactions.length

            //Initialize three arrays which contains the following information
            var revenue = [];
            var sales = [];
            var date = [];

            //Push all the retrieved data into the respective array
            for (stat in parsedStatistics) {
                revenue.push(parsedStatistics[stat].dailyRevenue)
                sales.push(parsedStatistics[stat].dailySales)
                //Convert the date format into (YYYY-MM-DD)
                date.push(parsedStatistics[stat].date.split('T')[0])
            }

            var collectionOfAllStatistics = {
                revenue: revenue,
                sales: sales,
                date: date,
                customers: numberOfCustomers,
                products: numberOfProducts,
                transactions: numberOfTransactions,
                orderedProducts: sortedproducts,
                orderedCustomers: sortedCustomers
            }
            //Send the data as a response
            res.send(collectionOfAllStatistics)
        }
    } catch (err) {
        res.send(err)
    }
})

//The getUserTransactionDetails function is also marked async as it uses functions
that returns promises
async function getAllTransactionDetails() {
    //Resolves is used to return a value in the form of a promise
    return new Promise(async (resolve, reject) => {
        //Calls the function queryDb and continues executing other lines while
waiting to get a response
        const allTransactions = await queryDb('SELECT * FROM `transaction`')
        var transactions = JSON.parse(JSON.stringify(allTransactions))
        //Cycles through all the transaction records
```

```
        var allTransactionsArray = []
        for (transaction in transactions) {
            //Filters the transaction to only records with the following status
            var details = await queryDb('SELECT * FROM `transaction_details` where
idTransaction = ' + transactions[transaction].idTransaction)
            var allDetails = JSON.parse(JSON.stringify(details))
            var allTransactionDetailsArray = [];

            for (specdetails in allDetails) {
                var productQuery = 'SELECT * FROM `catalogue` where  productID = ' +
allDetails[specdetails].productID
                var variantQuery = 'SELECT * FROM `variant` where  idvariant = ' +
allDetails[specdetails].variantID

                const [productInformation, variantInformation] = await
Promise.all([queryDb(productQuery), queryDb(variantQuery)]);
                var fullInformation = {
                    details: details[specdetails],
                    products: productInformation[0],
                    variant: variantInformation[0]
                }
                allTransactionDetailsArray.push(fullInformation)
            }
            var completeTransactionInformation = {
                transaction: transactions[transaction],
                details: allTransactionDetailsArray
            }
            allTransactionsArray.push(completeTransactionInformation)
        }
        resolve(allTransactionsArray)
    });
}

app.get('/getalltransactions', authenticateToken, async (req, res) => {
    try {
        if (req.userInformation.status == "ADMIN") {
            var allTransactions = await getAllTransactionDetails()
            res.send(JSON.parse(JSON.stringify(allTransactions)))
        }
    } catch (err) {
        res.send(err)
    }
})


async function getLogistics() {
    return new Promise(async (resolve, reject) => {
        //Calls the function queryDb and continues executing other lines while
waiting to get a response
```

```
        const allTransactions = await queryDb('SELECT * FROM `transaction` ORDER BY
dateOrdered')
        var transactions = JSON.parse(JSON.stringify(allTransactions))
        var allTransactionsArray = [];
        //Cycles through all the transaction records
        for (transaction in transactions) {
            var allTransactionDetails = []
            //Filters the transaction to only records with the following status
            if (transactions[transaction].orderStatus == "Payment Approved" ||
transactions[transaction].orderStatus == "Processing" ||
transactions[transaction].orderStatus == "Delivering") {
                var queryDetails = 'SELECT * FROM `transaction_details` where
idTransaction = ' + transactions[transaction].idTransaction
                var queryClient = 'SELECT * FROM `user_info` where  userID = ' +
transactions[transaction].userID
                //Retrieve the details of the transactions + the details of the user
so that logistics know where and who to send to
                const [clientInfo, details] = await
Promise.all([queryDb(queryClient), queryDb(queryDetails)]);
                //parse the results
                var allClient = JSON.parse(JSON.stringify(clientInfo))
                var allDetails = JSON.parse(JSON.stringify(details))
                //For all the details retrieve the necessary product and variant
information
                for (specificDetails in allDetails) {
                    var queryProduct = 'SELECT * FROM `catalogue` where  productID =
' + allDetails[specificDetails].productID
                    var queryVariant = 'SELECT * FROM `variant` where  idvariant = '
+ allDetails[specificDetails].variantID
                    const [productInformation, variantInformation] = await
Promise.all([queryDb(queryProduct), queryDb(queryVariant)]);
                    var allVariantInformation =
JSON.parse(JSON.stringify(variantInformation))
                    //store the query result in an object and and push it into the
array of transaction details
                    var completeInformation = {
                        details: details[specificDetails],
                        variant: allVariantInformation[0].variantName,
                        products: {
                            category: productInformation[0].category,
                            name: productInformation[0].name
                        }
                    }
                    allTransactionDetails.push(completeInformation)
                }

                //Create another object for each of the transactions and all their
details --> push into an array
                if (allClient.length == 0) {
                }
```

```
            else {
                //Final object of all the data required, transaction -->
transaction details + product + variant + quantity
                var collectiveTransactions = {
                    transaction: transactions[transaction],
                    transactionDetails: allTransactionDetails,
                    user: {
                        name: allClient[0].name,
                        storename: allClient[0].storename,
                        address: allClient[0].address,
                        phonenum: allClient[0].phonenum
                    }
                }
                //For each record, there are multiple details and such -->
stored in an array for each record
                allTransactionsArray.push(collectiveTransactions)
            }

        }
    }
    //Return the final result
    resolve(allTransactionsArray)
  });
}

app.get('/getalllogistics', authenticateToken, async (req, res) => {
  try {
    if (req.userInformation.status == "ADMIN" || req.userInformation.status ==
"LOGISTICS") {
        var allLogistics = await getLogistics()
        res.send(JSON.stringify(allLogistics))
    }
  } catch (err) {
    res.send(err)
  }
})

app.get('/product/:productID', authenticateToken, async (req, res) => {
  try {
    var specificProductInformation = await queryDb('SELECT * FROM `catalogue`
where  productID =' + req.params.productID)
    var specificProductInformation = JSON.parse(JSON.stringify(results));
    res.send(specificProductInformation)
  }
  catch {
    res.send('something went wrong')
  }
})

//Creates a router that returns all information about a product and the variants
```

```
app.get('/getallcatinfo/:catID', authenticateToken, async (req, res) => {
   try {
       //Makes sure that the user is an ADMIN
       if (req.userInformation.status == "ADMIN") {
           //Creates two queries, 1: querying the product info, 2: querying the all
the other product types
           var queryProduct = 'SELECT * FROM `catalogue` where  productID = ' +
req.params.catID
           var queryVariant = 'SELECT * FROM `variant` where  productID = ' +
req.params.catID
           //Using promise.all allows both functions to run in parallel and holds
off on execution until a promise is returned
           const [allProducts, allVariants] = await
Promise.all([queryDb(queryProduct), queryDb(queryVariant)]);
           //The returned values are parsed into javaScript object
           var productINFO = JSON.parse(JSON.stringify(allProducts));
           var variantINFO = JSON.parse(JSON.stringify(allVariants));
           var finalvariantinfo = []
           for (variant in variantINFO) {
               const imageaddresses = await queryDb('SELECT * FROM `images` where
variantID = ' + variantINFO[variant].idvariant)
               var fullvariant = {
                   variant: variantINFO[variant],
                   imageInformation: imageaddresses
               }
               finalvariantinfo.push(fullvariant)
           }
           //Creates an object that marks the parsed values with the following
labels
           var allcat = {
               product: productINFO,
               variantInformation: finalvariantinfo
           }
           //Sends all the data
           res.send(allcat)
       }
   }
   catch (err) {
       res.send(err)
   }
})

//Route to retrieve all the images concerned with a particular product variant
app.get('/imageaddress/:variantID', authenticateToken, async (req, res) => {
   try {
       if (req.userInformation.status == "ADMIN") {
           const allAddresses = await queryDb('SELECT * FROM `images` where
variantID =' + req.params.variantID)
           res.send(allAddresses)
```

```
        }
    }
    catch (err) {
        res.send(err)
    }
})

async function getVariantAndProductInformation(requestedUser) {
    return new Promise(async (resolve, reject) => {
        //Select all the cart items from a requested user
        const allCartItems = await queryDb('SELECT * FROM `cart_item` where  cartID
=' + requestedUser)
        const allItemsInformation = JSON.parse(JSON.stringify(allCartItems))
        var cartCollectionsArray = [];
        //For each of those items, retrieve the product, variant, and images to be
displayed
        for (item in allItemsInformation) {
            var queryCatalogue = 'SELECT * FROM `catalogue` where  productID =' +
allItemsInformation[item].productID
            var queryVariant = 'SELECT * FROM `variant` where  idvariant =' +
allItemsInformation[item].variantID
            var queryImage = 'SELECT * FROM `images` where  variantID =' +
allItemsInformation[item].variantID

            //Run the asynchronous functions in parallel to save time.
            const [allProduct, allVariant, allImage] = await
Promise.all([queryDb(queryCatalogue), queryDb(queryVariant), queryDb(queryImage)]);

            var variant = JSON.parse(JSON.stringify(allVariant))
            var product = JSON.parse(JSON.stringify(allProduct))
            var image = JSON.parse(JSON.stringify(allImage))

            //Collect all the information in one giant packet to be sent to the
front-end
            var productInformation = {
                cartItem: allItemsInformation[item],
                product: {
                    name: product[0].name,
                    productID: product[0].productID
                },
                variantInformation: {
                    variantName: variant[0].variantName,
                    price: variant[0].price,
                    stock: variant[0].quantity,
                },
                image: image[0].imageaddress
            }
            cartCollectionsArray.push(productInformation)
        }
        //Return the value in the form of a promise using resolve
```

```
        resolve(cartCollectionsArray)
    })
}

//Route to retrieve all the cart items to be displayed
app.get('/cartitem', authenticateToken, async (req, res) => {
    try {
        const allCartItems = await
getVariantAndProductInformation(req.userInformation.userID)
        res.send(allCartItems)

    }
    catch (err) {
        res.send(err)
    }
})

//Route to get all the products from the catalogue
app.get('/catalogue/all', authenticateToken, async (req, res) => {
    try {
        var catalogue = await queryDb('SELECT * FROM catalogue')
        var allCatalogue = JSON.parse(JSON.stringify(catalogue));
        var allProducts = []
        for (product in allCatalogue) {
            var allImages = await queryDb('SELECT * FROM `images` where  productID =
' + allCatalogue[product].productID)
            var randomImage = JSON.parse(JSON.stringify(allImages));
            //Choose a random image to represent the product for the user to see
            if (randomImage.length == 0) {
                var productInformation = {
                    productID: allCatalogue[product].productID,
                    name: allCatalogue[product].name,
                    category: allCatalogue[product].category,
                    totalSales: allCatalogue[product].totalSales,
                    description: allCatalogue[product].description,
                    randomImage: "/images/Screen Shot 2021-12-31 at 14.39.23.png"
                }
                allProducts.push(productInformation)
            }
            else {
                var productInformation = {
                    productID: allCatalogue[product].productID,
                    name: allCatalogue[product].name,
                    category: allCatalogue[product].category,
                    totalSales: allCatalogue[product].totalSales,
                    description: allCatalogue[product].description,
                    randomImage: randomImage[0].imageaddress
                }
                allProducts.push(productInformation)
            }
```

```
        }
        var product = {
            product: allProducts
        }
        //Sends all the data
        res.send(JSON.stringify(product))
    }
    catch (err) {
        res.send(err)
    }
})

//Route to retrieve all information of a specific product when the user clicks to
view a specific image during browsing
app.get('/getspecific/:pID', authenticateToken, async (req, res) => {
    try {
        var queryVariant = 'SELECT * FROM variant where productID =' +
req.params.pID
        var queryCatalogue = 'SELECT * FROM catalogue where productID =' +
req.params.pID
        const [catalogue, variant] = await Promise.all([queryDb(queryCatalogue),
queryDb(queryVariant)]);
        var productInformation = JSON.parse(JSON.stringify(catalogue));
        var variantInformation = JSON.parse(JSON.stringify(variant));
        var allVariant = []
        //For each of the variants gather their images and store them in one object
        for (specificVariant in variantInformation) {
            var allimages = await queryDb('SELECT * FROM `images` where  variantID =
' + variantInformation[specificVariant].idvariant)
            var variantAndImages = {
                variant: variantInformation[specificVariant],
                images: allimages
            }
            allVariant.push(variantAndImages)
        }
        var product = {
            product: productInformation,
            allVariant: allVariant
        }
        //Sends all the data
        res.send(JSON.stringify(product))
    }
    catch (err) {
        res.send(err)
    }
})

app.get('/cart/:cartID', (req, res) => {
```

```
    connection.query("SELECT * FROM `cart_item` where  cartID = ?",
[req.params.cartID], function (err, results) {
        if (err) throw err;
        var allCartItems = JSON.parse(JSON.stringify(results));
        res.send(allCartItems)
    })
})

app.get('/getcartitems', authenticateToken, async (req, res) => {
    try {
        var cartItems = await queryDb('SELECT * FROM `cart_item` where  cartID= ' +
req.userInformation.userID.toString())
        res.send(cartItems)
    }
    catch (err) {
        res.send(err)
    }
})



app.get('/details/:idTransaction', (req, res) => {
    connection.query("SELECT * FROM `transaction_details` where  idTransaction = ?",
[req.params.idTransaction], function (err, results) {
        if (err) throw err;
        var allTransactionDetails = JSON.parse(JSON.stringify(results));
        res.send(allTransactionDetails)
    })
})

//Source: https://stackoverflow.com/questions/27978868/destroy-cookie-nodejs - how
to destroy cookie when the user logouts
app.get('/logout', (req, res) => {
    cookie = req.cookies;
    for (var prop in cookie) {
        //remove the cookie items: tokens
        if (!cookie.hasOwnProperty(prop)) {
            continue;
        }
        res.cookie(prop, '', { expires: new Date(0) });
    }
    res.redirect('/login');

})

//Routes to post data into the database

//Source:
https://dev.to/jlong4223/how-to-implement-email-functionality-with-node-js-react-js
-nodemailer-and-oauth2-2h7m - sending email using nodemailer
```

```
let transporter = nodemailer.createTransport({
    service: "gmail",
    //Pass retrieved credentials into the auth section for authenticationto use the
GMAIL service
    auth: {
        type: "OAuth2",
        user: process.env.EMAIL,
        pass: process.env.WORD,
        clientId: process.env.OAUTH_CLIENTID,
        clientSecret: process.env.OAUTH_CLIENT_SECRET,
        refreshToken: process.env.OAUTH_REFRESH_TOKEN,
    },
});

//Source: https://sebhastian.com/bcrypt-node/ - hasing/encrypting passwords
//Hashing and encrypting passwords using Bcrypt
async function hashing(password) {
    return new Promise(async (resolve, reject) => {
        const salt = await bcrypt.genSalt(7);
        const hashPassword = await bcrypt.hash(password, salt);
        resolve(hashPassword)
    })
}

//Source:
https://futurestud.io/tutorials/generate-a-random-string-in-node-js-or-javascript
//Function used to generate random passwords.
function randomString(size = 21) {
    return Crypto
        .randomBytes(size)
        .toString('base64')
        .slice(0, size)
}

//Route to create a new user and send an email to inform them of their credentials
app.post('/createUsers', authenticateToken, async function (req, res) {
    //Only admin has access to creating a new user
    if (req.userInformation.status == "ADMIN") {
        //Generate a password of random strings
        password = randomString()
        //Hash the password using BCRYPT library
        var hashedPassword = await hashing(password)
        //Connect to the database and insert the following information
        var dateCreated = moment().format("YYYY-MM-DD")
        connection.query("INSERT INTO `user_info` SET `password`=?,
`profilepic`=?,`email`=?, `phonenum`=?, `storename`=?, `address`=?, `status`=?,
`name`=?, `created`=?", [hashedPassword,
"/images/blank-profile-picture-973460_1280.png", req.body.email, req.body.phonenum,
req.body.storename, req.body.address, req.body.status, req.body.name, dateCreated],
function (error, results, fields) {
```

```
            if (error) throw error;
            //Define the email options so that the api knows where to send the email
as well the content of the email
            let mailOptions = {
                from: process.env.EMAIL,
                to: req.body.email,
                subject: "New Account",
                text: req.body.text,
                //The email content which informs the user of their new account
credentials
                html: `
            <p> Hi ${req.body.name},<p>  <br>
            <p>The following includes the login information for your new account.
<p><br>
            <p> Email: ${req.body.email} <p> <br>
            <p> Password: ${password} <p> <br>
            <p> Thanks, <p> <br>
            <p> Torch ID <p>
            `
            };
            //Use the pre-defined transported module to send an email to the
specified address using the gmail service
            transporter.sendMail(mailOptions, function (err, data) {
                if (err) {
                    console.log(err)
                } else {
                }
            });
            res.send(JSON.stringify(results));
        });
    }
    else {
        res.redirect('/login')
    }
});



//front end
https://stackoverflow.com/questions/48859546/upload-image-to-server-with-xmlhttpreq
uest-and-formdata-in-react-native
//Source: https://www.geeksforgeeks.org/file-uploading-in-node-js/

//Route to add a new model to an exisiting product
app.post('/addmodel/:productID', authenticateToken, upload.array('uploaded_file',
5), async function (req, res) {
    try {
        if (req.userInformation.status == "ADMIN") {
            var modelInformation = JSON.parse(req.body.newModel)
            var newModel = {
```

```
                variantName: modelInformation.name,
                quantity: parseInt(modelInformation.quantity),
                price: parseInt(modelInformation.price),
                productID: req.params.productID
            }
            //Inser the new variant into the table
            connection.query('INSERT INTO `variant` SET ?', newModel, async function
(err, results) {
                if (err) throw err;
                for (var i = 0; i < req.files.length; i++) {
                    var newimage = {
                        variantID: results.insertId,
                        productID: req.params.productID,
                        imageaddress: req.files[i].path.replace('public', '')
                    }
                    var newImageID = await queryDbWithId('INSERT INTO images SET ?',
newimage)
                }
            })
            return res.send("success")
        }
        else {
            res.send('Not Available')
        }
    }
    catch {
        res.send('something went wrong')
    }
})

//Upload a new profile picture
app.post('/uploadimage', authenticateToken, upload.single('uploaded_file'),
function (req, res) {
    try {
        var imageAddress = req.file.path.replace('public', '')
        connection.query('UPDATE `user_info` SET  `profilepic` = ? where
`userID`=?', [imageAddress, req.userInformation.userID], async function (err,
results) {
            if (err) throw err;
            res.send(results)
        })
    }
    catch (err) {
        res.send(err)
    }
})

//Update the picture of a variant
app.post('/updateimage/:iID', authenticateToken, upload.single('uploaded_file'),
function (req, res) {
```

```
    try {
        var imageAddress = req.file.path.replace('public', '')
        connection.query('UPDATE `images` SET  `imageaddress` = ? where
`idimages`=?', [imageAddress, req.params.iID], async function (err, results) {
            if (err) throw err;
            res.send(results)
        })
    }
    catch (err) {
        res.send(err)
    }
})

//Upload a new payment proof for the users
app.post('/uploadpayment/:transactionID', authenticateToken,
upload.single('uploaded_file'), function (req, res) {
    try {
        var imageAddress = req.file.path.replace('public', '')
        connection.query('UPDATE `transaction` SET  `paymentproof` = ? where
`idTransaction`=?', [imageAddress, req.params.transactionID], async function (err,
results) {
            if (err) throw err;
        })
    }
    catch (err) {
        res.send(err)
    }
})

//Add a new product in the database
app.post('/addproduct', authenticateToken, function (req, res) {
    try {
        if (req.userInformation.status == "ADMIN") {
            var newProduct = {
                name: req.body.name,
                category: req.body.category,
                description: req.body.description,
                totalSales: 0,
                dateAdded: moment().format("YYYY-MM-DD"),
                dateModified: moment().format("YYYY-MM-DD"),
            }
            connection.query('INSERT INTO `catalogue` SET ?', newProduct, function
(err, results) {
                if (err) throw err;
            })

            res.send("success")
        }
        else {
            res.send('Not Available')
```

```
        }
    }
    catch {
        res.send('something went wrong')
    }
})

//A route that calls a function calls generateStatusToken() and stores the
retrieved value inside the http only cookie
//The user status (admin, user, logistics) is returned to the user for page
routing.
app.post('/loginusers', async (req, res) => {
    try {
        //Pass email and password into the function
        const tokenStatus = await generateStatusToken(req.body.email,
req.body.password)
        //Store the values inside http only cookies with the following name
        //             Name      Passing the returned token values   Expiry duration
Can't be accessed through Javscript
        res.cookie('accessToken', tokenStatus.accessToken, { maxAge: 60000000000000,
httpOnly: true })
        res.cookie('refreshToken', tokenStatus.refreshToken, { maxAge:
100000000000000, httpOnly: true })
        //Respond the user status to the user
        res.json({
            status: tokenStatus.status
        })
    }
    //If erorr send the error message
    catch (err) {
        res.send(err)
    }
})

//Route to allow users to add an item to acart
app.post('/addcartitem', authenticateToken, async (req, res) => {
    try {
        var newCartItems = {
            variantID: req.body.variantID,
            quantity: req.body.quantity,
            cartID: req.userInformation.userID,
            productID: req.body.productID
        }
        //Query to insert it into the database
        var newcartid = await queryDbWithId('INSERT INTO cart_item SET ?',
newCartItems)

        res.send('success')
    }
    catch (err) {
```

```
        res.send(err)
    }
})

//When the user checksout and places an order
async function checkOut(allTransactions, requesteduser) {
    return new Promise(async (resolve, reject) => {
        var fullTransaction = {
            orderStatus: "Waiting Approval",
            userID: requesteduser,
            dateOrdered: moment().format("YYYY-MM-DD"),
            totalUnits: allTransactions.transaction.quantity,
            totalValue: allTransactions.transaction.totalValue
        }
        //Insert the items into new records
        var newTransactionId = await queryDbWithId('INSERT INTO transaction SET ?',
fullTransaction)
        for (detail in allTransactions.details) {
            var completeTransactionDetails = {
                productID: allTransactions.details[detail].productID,
                idTransaction: newTransactionId,
                quantity: allTransactions.details[detail].quantity,
                variantID: allTransactions.details[detail].variantID
            }
            var response = await queryDbWithId('INSERT INTO transaction_details SET
?', completeTransactionDetails)
        }

        var status = await queryDb('DELETE FROM `cart_item` WHERE cartID =' +
requesteduser)
        resolve(status)
    })
}

app.post('/checkout', authenticateToken, async (req, res) => {
    try {
        var allTransactions = req.body
        var response = await checkOut(allTransactions, req.userInformation.userID)
    }
    catch (err) {
        res.send(err)
    }
})


//Sends an email when a new user is created
app.post('/sendemail', async (req, res) => {
    try {
        var specificUser = await queryDb('SELECT * FROM `user_info`')
        var parsedUserInformation = JSON.parse(JSON.stringify(specificUser))
```

```
        var status = "NOT FOUND"
        for (user in parsedUserInformation) {
            if (parsedUserInformation[user].email == req.body.email) {
                status = "FOUND"
                let mailOptions = {
                    from: process.env.EMAIL,
                    to: req.body.email,
                    subject: "Password Reset",
                    html: `
                            <p> Hi ${parsedUserInformation[user].name},<p>  <br>
                            <p>We received 1 request to reset your web application
password.<p><br>
                            <p> Click the following link to reset your password <p>
<br>
                            <a href=
http://localhost:5001/reset/${parsedUserInformation[user].userID}> Reset Password
</a>
                            <p> Thanks, <p> <br>
                            <p> Torch ID <p>
                            `
                };
                transporter.sendMail(mailOptions, function (err, data) {
                    if (err) {
                        console.log(err)
                    } else {
                        res.send(JSON.stringify(mailOptions))
                    }
                });
                break;
            }
        }
        if (status == "NOT FOUND") {
            res.send("NOT FOUND")
        }
    }
    catch {
        res.send('something went worng')
    }
})

//Routes to delete data from the database
app.delete('/deleteUsers/:userID', function (req, res) {
    connection.query("DELETE FROM `user_info` WHERE userID =?;",
[req.params.userID], function (error, results, fields) {
        if (error) throw error;
        res.send(JSON.stringify(results));
    });
});

//Route to delete a cart item for the user
```

```
app.delete('/deletecartItem/:cartID', authenticateToken, async function (req, res)
{
    try {
        var deleteQuery = "DELETE FROM `cart_item` WHERE idcart_item =" +
req.params.cartID
        var response = await queryDb(deleteQuery)
        res.send("DELETED")
    }
    catch (err) {
        res.send(err)
    }
});

//Rotues to update data in the database
//Update the user information
app.put('/uploaduserInformation', authenticateToken, function (req, res) {
    try {
        connection.query('UPDATE `user_info` SET  `name` = ?, `storename` = ?,
`address` = ? , `phonenum` = ?, `email` = ? where `userID`=?', [req.body.name,
req.body.storename, req.body.address, req.body.phonenum, req.body.email,
req.userInformation.userID], async function (err, results) {
            if (err) throw err;
            res.send(results)
        })
    }
    catch (err) {
        res.send(err)
    }
})

//Edit the information of a model/variant of a product
app.put('/editModel/:vID', authenticateToken, async function (req, res) {
    try {
        if (req.userInformation.status == "ADMIN") {
            var status = await editmodel(req.body.name, req.body.quantity,
req.body.price, req.params.vID)
            res.send("edited")
        }
    }
    catch (err) {
        res.send(err)
    }
});

//Route to edit the product information
app.put('/editCatalogue/:productID', authenticateToken, async function (req, res) {
    try {
        if (req.userInformation.status == "ADMIN") {
            connection.query("UPDATE `catalogue` SET
`category`=?,`name`=?,`description`=?,`dateModified`=? where `productID`=?",
```

```
[req.body.category, req.body.name, req.body.description,
moment().format("YYYY-MM-DD"), req.params.productID], function (error, results,
fields) {
            });
        }
    }
    catch (err) {
        res.send(err)
    }
});

//User can update a transaction as completed when it arrives
app.put('/updateComplete/:idTransaction', function (req, res) {
    connection.query("UPDATE `transaction` SET `orderStatus`=? where
`idTransaction`=?", [req.body.orderStatus, req.params.idTransaction], function
(error, results, fields) {
        if (error) throw error;
        res.send(JSON.stringify(results));
    });
});

//Gather all the transaction details of a particular record so that an invoice can
be created
async function gettransactiondetails(requestedTransaction) {
    return new Promise(async (resolve, reject) => {
        var transactionQuery = "SELECT * FROM transaction where idTransaction =" +
requestedTransaction
        var transactionDetailQuery = "SELECT * FROM transaction_details where
idTransaction =" + requestedTransaction
        const [allTransactionsInformation, allTransactionDetails] = await
Promise.all([queryDb(transactionQuery), queryDb(transactionDetailQuery)]);
        var parsedTransactions =
JSON.parse(JSON.stringify(allTransactionsInformation))
        var parsedDetails = JSON.parse(JSON.stringify(allTransactionDetails))
        var collectionOfTransactionDetailsArray = []
        //For all the details query for more product and variant information
        for (details in parsedDetails) {
            var productQuery = "SELECT * FROM catalogue where productID =" +
parsedDetails[details].productID
            var variantquery = "SELECT * FROM variant where idvariant =" +
parsedDetails[details].variantID
            const [productInformation, variantInformation] = await
Promise.all([queryDb(productQuery), queryDb(variantquery)]);
            var parsedProduct = JSON.parse(JSON.stringify(productInformation))
            var parsedVariant = JSON.parse(JSON.stringify(variantInformation))
            var transcinfo = {
                quantity: parsedDetails[details].quantity,
                description: parsedProduct[0].name + " " +
parsedvariant[0].variantName,
                price: parsedVariant[0].price,
```

```
                "tax-rate": 0
            }
            collectionOfTransactionDetailsArray.push(transcinfo)
        }
        var fullTransactionDetails = {
            transc: parsedTransactions,
            alldetails: collectionOfTransactionDetailsArray
        }
        resolve(fullTransactionDetails)
    })
}

//Generate a new invoice using the library easyinvocie
async function generateinvoice(data, reqinfo) {
    const result = await easyinvoice.createInvoice(data);
    fs.writeFileSync(path.join(__dirname, "./public/invoices", "Transaction" +
reqinfo + ".pdf"), result.pdf, 'base64');
}

//Source: https://easyinvoice.com/
async function getuserbyidtransaction(reqid) {
    return new Promise(async (resolve, reject) => {
        var response = await gettransactiondetails(reqid)
        var data = {
            "images": {
                // The logo on top of your invoice
                "logo": fs.readFileSync('output-onlinepngtools.jpg', 'base64')
                // The invoice background
            },
            // Your own data
            "sender": {
                "company": "Torch Electric ID",
                "zip": "14450",
                "city": "Jakarta",
                "country": "Indonesia"
            },
            "information": {
                // Invoice number
                "number": response.transc[0].idTransaction,
                // Invoice data
                "date": response.transc[0].dateOrdered.split('T')[0],
                "due-date": "-"
            },
            "products": response.alldetails,
            "bottom-notice": "Please pay your invoice within 30 days.",
            "settings": {
                "currency": "IDR"
            }
        };
        generateinvoice(data, response.transc[0].idTransaction, async () => {
```

```
            try {
            }
            catch (err) {
                console.log(err)
            }
        })
        resolve(response.transc[0].userID)
    })
}

async function getUserByTransactionId(reqID) {
    return new Promise(async (resolve, reject) => {
        var response = await getuserbyidtransaction(reqID)
        var userInformation = await queryDb('SELECT * FROM user_info where userID ='
+ response)
        resolve(JSON.parse(JSON.stringify(userInformation)))
    })
}

//Set an interal delay to provide clashing of tasks or requests
function delay(time) {
    return new Promise(function (resolve) {
        setTimeout(resolve, time)
    });
}

//When the user places an order, reduce the stock quantity
async function updateStockQuantity(newstock) {
    return new Promise(async function (resolve) {
        for (stock in newstock) {
            var updatestockquery = "UPDATE `variant` SET `quantity`= '" +
newstock[stock].newstock + "' where `idvariant`=" + newstock[stock].variantID
            var response = await queryDb(updatestockquery)
        }
        resolve('.')
    })
}

//Add the total sales and revenue when a new order/payment has been acccpeted by an
employee
async function updateSalesAndRevenue(alldata) {
    return new Promise(async function (resolve) {
        var currentdate = moment().format("YYYY-MM-DD")
        var addsalesrev = "UPDATE `daily` SET `dailySales`= dailySales +  " +
alldata.totalSales + " , `dailyRevenue`= dailyRevenue+" + alldata.totalRevenue + "
WHERE date = CURDATE()"
        var response = await queryDb(addsalesrev)
        resolve(response)
    })
}
```

```
//Updates the status of a record based on its current status
app.put('/updateStatus/:tID', authenticateToken, async function (req, res) {
   try {
       if (req.body.orderStatus == "Waiting Payment") {

           const [response, newstock] = await
Promise.all([getUserByTransactionId(req.params.tID),
updateStockQuantity(req.body.newstock)]);

           let mailOptions = {
               from: process.env.EMAIL,
               to: response[0].email,
               subject: "INVOICE",
               html: `
                   <p> Hi ${response[0].name},<p>  <br>
                   <p>We have successfully received your order. Please find
attached for your payment details and invoice. <p><br>
                   <p> Thanks, <p> <br>
                   <p> Torch ID <p>
                   `,
               attachments: [
                   {
                       filename: 'Transaction' + req.params.tID + '.pdf',
                       path: path.join(__dirname, './public/invoices',
'Transaction' + req.params.tID + '.pdf'),
                       contentType: 'application/pdf'
                   }
               ]
           };
           await delay(4000);
           transporter.sendMail(mailOptions, function (err, data) {
               if (err) {
                   console.log(err)
               } else {
               }
           });
       }
       else if (req.body.orderStatus == "Payment Approved") {
           var response = await updateSalesAndRevenue(req.body)
       }
       var updatequery = "UPDATE `transaction` SET `orderStatus`= '" +
req.body.orderStatus.toString() + "' where `idTransaction`=" + req.params.tID
       var response = await queryDb(updatequery)
       res.send(response)
   }
   catch (err) {
       res.send(err)
   }
});
```

```
app.put('/updateUsers/:userID', function (req, res) {
   connection.query("UPDATE `user_info` SET
`phonenum`=?,`storename`=?,`address`=?,`name`=? where `userID`=?",
[req.body.phonenum, req.body.storename, req.body.address, req.body.name,
req.params.userID], function (error, results, fields) {
       if (error) throw error;
       res.send(JSON.stringify(results));
   });
});

//Used for resetting password
app.put('/updatePassword/:userID', async function (req, res) {
   try {
       var hashedPassword = await hashing(req.body.password)
       connection.query("UPDATE `user_info` SET `password`=? where `userID`=?",
[hashedPassword, req.params.userID], function (error, results, fields) {
           if (error) throw error;
       });
   }

   catch {

   }
});


app.listen(pt)
```

User-Side Codes

Main manu - USER

mainmenu.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Main Menu</title>
   <script src="/js/mainmenu.js"></script>
   <script src="/js/client.js"></script>
   <link rel="stylesheet" href="/css/mainmenu.css">
   <link rel="stylesheet" href="/css/client.css">

   <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
```

```
</head>

<body>
   <div class="header" colspan="2">
       <div id="header-left">
           <img class="logo" src =
"/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">

       </div>

       <div class="header-right">

         <a onclick="logOut();">Logout</a>
         <a><i onclick="goCart();" id ="cart" class='bx bx-cart' ></i></a>

       </div>
   </div>


   <div class = "wrap1">
       <div id ="new">
       </div>

       <div id ="popular">
       </div>
   </div>

   <br>
   <div class = "wrap">

       <div onclick="catalogue();" class="item">
           <ul class="lists" style="list-style-type:none">
               <li id="icons"><i class='bx bxs-package'></i></li>
               <li>Catalogue</li>
           </ul>
       </div>
       <div onclick="delivery();" class="item">
           <ul class="lists" style="list-style-type:none">
               <li id="icons"><i class='bx bxs-truck' ></i></li>
               <li>Delivery Progress</li>
           </ul>

       </div>
       <div onclick="orderHistory();" class="item">
           <ul class="lists" style="list-style-type:none">
               <li id="icons"><i class='bx bx-history' ></i></li>
               <li>Order History</li>
           </ul>
       </div>
   </div>
```

```
</div>
</body>
</html>
```

Mainmenu.js

```
var ne;
var upcoming;
var popular;


window.onload = function() {
   retrievePromotion();
 };

 //Calls the query to retrieve the promotion of products (popular and newly
released)
function retrievePromotion(){
   createXhrRequest( "GET", 'http://localhost:5001/promotion', function( err,
response ) {
       if( err ) { console.log( "Error!" ); }
       var promotion = JSON.parse(response);
       document.getElementById('new').innerHTML='<img class="image"
src="'+promotion[0].image.imageaddress+'"> <br> <div
class="promotiontext">NEW</div> <div
onclick="viewproduct('+promotion[0].product.productID+')"
class="viewmorebutton">View More </div> '
       document.getElementById('popular').innerHTML='<img class="image"
src="'+promotion[1].image.imageaddress+'"> <br> <div
class="promotiontext">POPULAR</div> <div
onclick="viewproduct('+promotion[1].product.productID+')"
class="viewmorebutton">View More </div>'

   });

   //Get the profile picture of the user to be displayed in the front page
   createXhrRequest( "GET", 'http://localhost:5001/getuserprofilepic', function(
err, response ) {
       if( err ) { console.log( "Error!" ); }
       var userInformation = JSON.parse(response);
       document.getElementById('header-left').innerHTML +='<img
onclick="changePage()" class="profilepic" src="'+userInformation.imageAddress+ '"
class="image"/> '
   });

}

function changePage(){
   window.location.replace('http://localhost:5001/edituserinfo')
}
```

```
function catalogue(){
   window.location.replace('http://localhost:5001/catalogue')
}

function delivery(){
   window.location.replace('http://localhost:5001/delivery')
}

function orderHistory(){
   window.location.replace('http://localhost:5001/orderhist')
}

function productpage(){
   window.location.replace('http://localhost:5001/product')
}

function viewproduct(id){
   window.location.replace('http://localhost:5001/productinfo/'+id)
}
```

Login Page

login.html

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <script src="/js/login.js"></script>
   <link rel="stylesheet" href="/css/login.css">
   <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>

   <title>CS IA</title>
</head>

<body>
   <div class="container">
       <img id="logo"
src="/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">

           <div class = "emailtext">
               <div class="text">Email</div> <br>
               <input type="text" id="email" placeholder="Enter your email">

           </div>
           <div class="passwordtext">
               <div class="text">Password</div><br>
```

```
            <input id="password" type="password" placeholder="Enter your
password" id="password">
            <div id="forgot" onclick="openForgot();">Forgot Password?</div>

        </div>
            <div id="loginbutton" onclick="login();"> Login</div>
    </div>

    <div class="forgot-password">
        <div class="actual-forgot">
            <h3 id="forgotText">Reset Password </h3><br>
            Enter your email and we will send a link to reset your password!
            <input id="reset" type="text" placeholder="Enter your email"><br>
            <div id="confirmbutton" onclick="confirm();"> Confirm </div>
            <div id="returnbuttonf" onclick="closeForgot();"> Return </div>
        </div>
    </div>

    <div class="modal-container">
        <div class="actualModal">
            <ul style="list-style-type:none;" class="lists">
                <li><i class='bx bx-x-circle' id="xbutton"></i></li>
                <li>
                    <h5 id="modalText">Wrong password or email!</h5>
                </li>
                <li>
                    <div id="returnbuttonw" onclick="back();">Return</div>
                </li>
            </ul>
        </div>
    </div>

    <div class="email-not-found">
        <div class="actualModal">
            <ul style="list-style-type:none;" class="lists">
                <li><i class='bx bx-x-circle' id="xbutton"></i></li>
                <li>
                    <h5 id="modalText">Email not found.</h5>
                </li>
                <li>
                    <div id="returnbuttonw" onclick="closeNotfound();">Return</div>
                </li>
            </ul>
        </div>
    </div>

    <div class="password-sent">
        <div class="actual-sent">
            <ul style="list-style-type:none;" class="lists">
                <i class='bx bx-check-circle' id="cbutton"></i>
```

```
                <h3 id="senttext-success">Success! </h3>
                <p>Your new password has been set!</p>
            </ul>
        </div>
    </div>

</body>

</html>
```

login.js

```
var createXhrRequest = function (httpMethod, url, callback) {
    var xhr = new XMLHttpRequest();
    xhr.open(httpMethod, url, false);

    xhr.onload = function () {
        callback(null, xhr.response);
    };

    xhr.onerror = function () {
        callback(xhr.response);
    };
    xhr.send();
}

function login() {
    var email = document.getElementById("email").value;
    var password = document.getElementById("password").value;
    var response = {}
    response.email = email
    response.password = password
    //Collect the email and password into an object
    var JSONformatdata = JSON.stringify(response);

    var modal = document.querySelector('.modal-container');
    var xhr = new XMLHttpRequest();
    xhr.open('POST', 'http://localhost:5001/loginusers');
    xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');

    xhr.onload = function () {
        //Once a response has been loaded, the following is executed
        if (xhr.readyState == 4 && xhr.status == "200") {
            var userResponse = JSON.parse(xhr.responseText)
            //Find out the user role of the person
            if (userResponse.status == "ADMIN") {
                window.location.replace('http://localhost:5001/maina')
            }
            else if (userResponse.status == "USER") {
                window.location.replace('http://localhost:5001/main')
```

```
            }
            else if (userResponse.status == "LOGISTICS") {
                window.location.replace('http://localhost:5001/mainl')
            }
            //Login not successful
            else if (userResponse.status == "Not Found") {
                modal.classList.add('container-active');
            }

        } else {
            console.error(users);
        }
    }
    //Send email and password to the backend to be authenticated
    xhr.send(JSONformatdata);
}


function back() {
    var modal = document.querySelector('.modal-container');
    modal.classList.remove('container-active');
}

function openForgot() {
    var modal = document.querySelector('.forgot-password');
    modal.classList.add('forgot-password-active');
}

//Triggers when the user confirms to reset the password
function confirm() {

    var email = document.getElementById("reset").value;

    var xhr2 = new XMLHttpRequest();
    xhr2.open("POST", "http://localhost:5001/sendemail");
    xhr2.setRequestHeader('Content-type', 'application/json; charset=utf-8');
    xhr2.onload = function () {
        //Checks that the user/email exists within the database
        if (xhr2.readyState == 4 && xhr2.status == "200") {
            if (xhr2.responseText=="NOT FOUND"){
                var modal = document.querySelector('.email-not-found');
                modal.classList.add('password-sent-active');
            }
            else{
            var modal = document.querySelector('.password-sent');
            modal.classList.add('password-sent-active');


//https://stackoverflow.com/questions/17883692/how-to-set-time-delay-in-javascript
                setTimeout(function () {
```

```
                    location.reload();
                }, 1000);
            }


        } else {
            console.error(users);
        }
    }
    var confirmEmail = {};
    confirmEmail.email = email;

    var JSONformatdata = JSON.stringify(confirmEmail);
    xhr2.send(JSONformatdata);
}

function closeForgot() {
    var modal = document.querySelector('.forgot-password');
    modal.classList.remove('forgot-password-active');
}

function closeSent() {
    var modal = document.querySelector('.password-sent');
    modal.classList.remove('password-sent-active');
}

function closeNotfound() {
    var modal = document.querySelector('.email-not-found');
    modal.classList.remove('password-sent-active');
}
```

Passwor Resetting Page

resetpass.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Reset Password</title>
    <script src="/js/resetpass.js"></script>
    <link rel="stylesheet" href="/css/resetpass.css">
    <link rel="stylesheet" href="/css/admin.css">
    <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
```

```
</head>

<body>
   <div class="header" colspan="2">
       <img class="logo"
src="/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
   </div>
   </div>
   <div class="maincontainer">
       <div class="text">
           <h1 id="bigtext">Password Reset</h1>
       </div>

       <div class="form">
           <ul class="list" style="list-style-type:none;">
               <li>
                   <p>Enter new password</p>
                   <input class="passwordinput" id ="password1"type="password"
placeholder="Enter your password">
               </li>
               <li>
                   <p>Confirm new password</p>
                   <input class="passwordinput" id ="password2" type="password"
placeholder="Enter your password">
               </li>
               <li>
                   <div class="button" onclick = "confirm();"> Confirm</div>
               </li>
           </ul>
       </div>
   </div>


   <div class="success">
       <div class="actual-success">
           <ul style="list-style-type:none;" class="groupofforms">
               <i class='bx bx-check-circle' id = "cbutton"></i>
               <h3 id="senttext-success">Success! </h3>
               <p>Check your email for a link to reset your password.</p>
           </ul>
       </div>
   </div>

   <div class="modal-container">
       <div class="actualModal">
           <ul style="list-style-type:none;" class="lists">
               <li><i class='bx bx-x-circle' id="xbutton"></i></li>
               <li><h5 id="modalText">Password doesn't match. Try Again.</h5></li>
               <li><div id="returnbuttonw" onclick="back();">Return</div></li>
```

```
            </ul>
        </div>
    </div>


</body>

</html>
```

resetpass.js

```
var currentLocation = window.location.pathname;
//Takes the current url and findout the userid
userID = currentLocation.replace("/reset/", "");

var createXhrRequest = function (httpMethod, url, callback) {

    var xhr = new XMLHttpRequest();
    xhr.open(httpMethod, url, false);

    xhr.onload = function () {
        callback(null, xhr.response);
    };

    xhr.onerror = function () {
        callback(xhr.response);
    };

    xhr.send();

}

//Make sure that both passwords match
function confirm(){
    var password1 = document.getElementById("password1").value;
    var password2 = document.getElementById("password2").value;
    if (password1==password2){
        var xhr = new XMLHttpRequest();
            xhr.open("PUT", "http://localhost:5001/updatePassword/"+userID, true);
            xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
            xhr.onload = function () {
                var users = JSON.parse(xhr.responseText);
                if (xhr.readyState == 4 && xhr.status == "200") {

                } else {
                    console.error(users);
                }
            }
            //Send the updated password to the backend to be updated
```

```
            var updatedPassword = {};
            updatedPassword.password = password1;
            var JSONformatdata = JSON.stringify(updatedPassword);
            xhr.send(JSONformatdata);
            var modal = document.querySelector('.success');
            modal.classList.add('success-active');
            setTimeout(function() {
                window.location.replace('http://localhost:5001/login')
            }, 2000);
    }
    else{
        var modal = document.querySelector('.modal-container');
        modal.classList.add('container-active');
    }
}

//close a confirmation popup
function closeSent(){
    var modal = document.querySelector('.password-sent');
    modal.classList.remove('password-sent-active');
}

//Allow user to close a popup
function back() {
    var modal = document.querySelector('.modal-container');
    modal.classList.remove('container-active');
}
```

Collection of functions used throughout the admin pages (repeated functions)

admin.js

```
function openSidebar(){
    let sidebar = document.querySelector(".sidebar")
    sidebar.classList.toggle("active")
}

//logout
function logOut(){
    window.localStorage.removeItem('userID');
    window.location.replace('http://localhost:5001/login')
}

//Each of the functions below is used to open and redirect to a specific page
function users(){
    window.location.replace('http://localhost:5001/usersdash/admin')
}

function orders(){
    window.location.replace('http://localhost:5001/orders/admin')
```

```
}

function catalogue(){
       window.location.replace('http://localhost:5001/catalogue/admin')
}

function transactions(){
   window.location.replace('http://localhost:5001/transactions/admin')
}

function mainMenu(){
   window.location.replace('http://localhost:5001/maina')
}


function remove(){
   var modal = document.querySelector('.create-form');
           modal.classList.remove('create-form-active');
}

//Used to call queries and api from the backend
var createXhrRequest = function (httpMethod, url, callback) {
   var xhr = new XMLHttpRequest();
   xhr.open(httpMethod, url, false);
   xhr.onload = function () {
       callback(null, xhr.response);
   };
   xhr.onerror = function () {
       callback(xhr.response);
   };
   xhr.send();
}
```

Collection of functions used throughout the client pages (repeated functions)

client.js

```
function logOut() {

   var xhr = new XMLHttpRequest();
   //Calls the api to logout as it has to delete the tokens within the http only
cookie
   xhr.open("GET", 'http://localhost:5001/logout', false);

   xhr.onload = function () {
       window.location.replace('http://localhost:5001/login')

   };
   xhr.onerror = function () {
```

```
    };
    xhr.send();

}

function home() {
    window.location.replace('http://localhost:5001/main')
}

//Source:
https://newbedev.com/javascript-number-format-javascript-rupiah-code-example
//convert numerical values into the currency
function convertRup(numb) {
    const format = numb.toString().split('').reverse().join('');
    const convert = format.match(/\d{1,3}/g);
    const rupiah = convert.join('.').split('').reverse().join('')

    return rupiah
}

function goCart() {
    window.location.replace('http://localhost:5001/cart')
}

//Filter the table certain status
function filterByStatus(){
    var table = document.getElementById('tables')
    var tr = table.getElementsByTagName("tr");
    var tablelength = tr.length-1

    for(var x =1;x<=tablelength;x++){
        if(document.getElementById('status').value==""){
            table.getElementsByTagName("tr")[x].style.display =""
        }else{

if(tr[x].getElementsByTagName("td")[1].innerHTML.toString()==document.getElementById
d('status').value.toString()){
                table.getElementsByTagName("tr")[x].style.display =""
            }
            else{
                table.getElementsByTagName("tr")[x].style.display ="none"
            }
        }
    }
}

//Sort the table by amount, typically total sales or total revenue
function sortByAmount(){
    var table = document.getElementById('tables')
    var tr = table.getElementsByTagName("tr");
```

```
   var tablelength = tr.length-1
   //standard bubble sorting
   for(var x =1;x<=tablelength;x++){
       for(var y =1;y<=tablelength-1;y++){
           if(document.getElementById('amount').value=="Descending"){

if(parseInt(tr[y].getElementsByTagName("td")[2].innerHTML)>parseInt(tr[y+1].getElem
entsByTagName("td")[2].innerHTML)){
                   tr[y].parentNode.insertBefore(tr[y+1],tr[y])
               }
           }
           else if(document.getElementById('amount').value=="Ascending"){

if(parseInt(tr[y].getElementsByTagName("td")[2].innerHTML)<parseInt(tr[y+1].getElem
entsByTagName("td")[2].innerHTML)){
                   tr[y].parentNode.insertBefore(tr[y+1],tr[y])
               }
           }

       }
   }
}

//Sort records by date - recent to latest/latest - recent
function sortByDate(){
   var table = document.getElementById('tables')
   var tr = table.getElementsByTagName("tr");
   var tablelength = tr.length-1

   for(var x =1;x<=tablelength;x++){
       for(var y =1;y<=tablelength-1;y++){
           if(document.getElementById('date').value=="Recent"){

if(tr[y].getElementsByTagName("td")[3].innerHTML<tr[y+1].getElementsByTagName("td")
[3].innerHTML){
                   tr[y].parentNode.insertBefore(tr[y+1],tr[y])
               }
           }
           else if(document.getElementById('date').value=="Latest"){

if(tr[y].getElementsByTagName("td")[3].innerHTML>tr[y+1].getElementsByTagName("td")
[3].innerHTML){
                   tr[y].parentNode.insertBefore(tr[y+1],tr[y])
               }
           }

       }
   }
}
```

```
//Calls the backend server api to retrieve and send data
var createXhrRequest = function (httpMethod, url, callback) {
   var xhr = new XMLHttpRequest();
   xhr.open(httpMethod, url, false);
   xhr.onload = function () {
       callback(null, xhr.response);
   };
   xhr.onerror = function () {
       callback(xhr.response);
   };
   xhr.send();
}
```

Cart page

cart.html

```
<!DOCTYPE html>
<html lang="en">

<head>
 <meta charset="UTF-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Cart</title>
 <link rel="stylesheet" href="/css/cart.css">
 <script src="/js/cart.js"></script>
 <script src="/js/client.js"></script>
 <link rel="stylesheet" href="/css/client.css">
 <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
</head>

<body>
 <div class="header" colspan="2">
   <img class="logo"
src="/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
   <div class="header-right">
     <a class="active" onclick="home();" href="#home">Home</a>

     <a onclick="logOut();">Logout</a>
     <a><i onclick="goCart();" id="cart" class='bx bx-cart'></i></a>

   </div>
 </div>
 <div class="maincontent">
   <div class="container">

     <div class="itemlist">
```

```
      <table id="maintable">
        <tr id="tableheader">
          <th id="first">

          </th>
          <th id="column">
            Model Name
          </th>
          <th id="column">
            Name
          </th>
          <th id="column">
            Total
          </th>
          <th id="column">
            Quantity
          </th>
          <th id="last">

          </th>
        </tr>
      </table>

    </div>
    <div class="total">
      <div class="header-total"><b>Total</b></div>
      <div class="maintotal">
        <div id="subtotal"> Subtotal: </div>
      </div>
      <div onclick="checkOut()" class="checkoutbutton"> Place Order</div>

    </div>
  </div>
 </div>
</body>

</html>
```

cart.js

```
window.onload = function () {
   getCartItemById();
};

var allCartItems

var totalValue = 0;


//Loads all the cart items based on the user ID
```

```
function getCartItemById() {
   createXhrRequest("GET", 'http://localhost:5001/cartItem', function (err,
response) {
       if (err) { console.log("Error!"); }
       var userCartItems = JSON.parse(response);
       allCartItems = userCartItems
       var table = document.getElementById('maintable')
       var count = 1;
       for (cart in userCartItems) {
           //Insert the data into the respective rows/columns
           var row = table.insertRow(count);
           var image = row.insertCell(0);
           var variantName = row.insertCell(1);
           var name = row.insertCell(2);
           var total = row.insertCell(3);
           var quantityUpdate = row.insertCell(4);
           var deleteButton = row.insertCell(5);
           image.innerHTML = '<img class="images" src="' +
userCartItems[cart].image + '">';
           variantName.innerHTML =
userCartItems[cart].variantInformation.variantName;
           name.innerHTML = userCartItems[cart].product.name;
           total.innerHTML =
convertRup(parseInt(userCartItems[cart].variantInformation.price) *
userCartItems[cart].cartItem.quantity);
           deleteButton.innerHTML= '<i
onclick="deleteCartItem('+userCartItems[cart].cartItem.idcart_item+')"
id="trashbutton" class="bx bx-trash"></i>'
           quantityUpdate.innerHTML = '<div class="container"> <div
onclick="subtractItemQuantity(this)"class="minusbutton"> <i class="bx bx-minus"
></i> </div> <input class="quantity" id="quantity' + cart + '" type="number"
value="' + userCartItems[cart].cartItem.quantity + '">  <div
onclick="addItemQuantity(this)"class="plusbutton"> <i class="bx bx-plus" ></i>
</div>'
           totalValue += (parseInt(userCartItems[cart].variantInformation.price) *
parseInt(userCartItems[cart].cartItem.quantity))
           count++
       }
       //Calculates the subtotal
       document.getElementById('subtotal').innerHTML += '<br> <div
id="totalnumber"><b>Rp. ' + convertRup(totalValue) + '</b> </div>'
   });
}

//Delete the cart item from user database
function deleteCartItem(itemID){
   var xhr = new XMLHttpRequest();

   xhr.open("DELETE", "http://localhost:5001/deletecartItem/"+itemID, true);
```

```
    xhr.send();

    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
        }
    }
    location.reload();

}

//Triggers when the user adds quantity to the select cart item
function addItemQuantity(row) {
    var table = document.getElementById('maintable')
    var rowNumber = row.parentElement.parentElement.parentElement.rowIndex
    var variantName = table.rows[rowNumber].cells[1].innerHTML
    for (specificCartItem in allCartItems) {
        if (allCartItems[specificCartItem].variantInformation.variantName ==
variantName) {
            //Increase quantity by 1 everytime the button is clicked
            var newQuantity = parseInt(document.getElementById('quantity' +
specificCartItem.toString()).value) + 1
            document.getElementById('quantity' + specificCartItem.toString()).value
= newQuantity
            table.rows[rowNumber].cells[3].innerHTML = convertRup(newQuantity *
parseInt(allCartItems[specificCartItem].variantInformation.price))
            totalValue +=
parseInt(allCartItems[specificCartItem].variantInformation.price)
            document.getElementById('subtotal').innerHTML = 'Subtotal: <br> <div
id="totalnumber"><b>Rp. ' + convertRup(totalValue) + '</b> </div>'
        }
    }
}

//Triggers when the user subtracts quantity to the select cart item
function subtractItemQuantity(row) {
    var table = document.getElementById('maintable')
    var rowNumber = row.parentElement.parentElement.parentElement.rowIndex
    var variantName = table.rows[rowNumber].cells[1].innerHTML
    for (specificCartItem in allCartItems) {
        if (allCartItems[specificCartItem].variantInformation.variantName ==
variantName) {
            var newQuantity = parseInt(document.getElementById('quantity' +
specificCartItem.toString()).value) - 1
            document.getElementById('quantity' + specificCartItem.toString()).value
= newQuantity
            table.rows[rowNumber].cells[3].innerHTML = convertRup(newQuantity *
parseInt(allCartItems[specificCartItem].variantInformation.price))
            totalValue -=
parseInt(allCartItems[specificCartItem].variantInformation.price)
```

```
            document.getElementById('subtotal').innerHTML = 'Subtotal: <br> <div
id="totalnumber"><b>Rp. ' + convertRup(totalValue) + '</b> </div>'
        }
    }
}

//User clicks checkout the trigger the button
function checkOut() {
    //Empty cart
   if (allCartItems.length==0){
    alert("You need to place an item in the cart")
   }else{
    var totalQuantity = 0;
    var totalValue = 0;
    var cartAndDetails = []
    //Loops through all the cart items and group them into one array
    for (var x = 0; x < allCartItems.length; x++) {
        totalQuantity += parseInt(document.getElementById('quantity' +
x.toString()).value)
        totalValue += parseInt(document.getElementById('quantity' +
x.toString()).value) * parseInt(allCartItems[x].variantInformation.price)
        var details = {
            productID: allCartItems[x].product.productID,
            quantity: parseInt(document.getElementById('quantity' +
x.toString()).value),
            variantID: allCartItems[x].cartItem.variantID
        }
        cartAndDetails.push(details)
    }

    var transactionOverview = {
        quantity: totalQuantity,
        totalValue: totalValue,
        productID: allCartItems[0].product.productID
    }

    var xhr = new XMLHttpRequest();

    //Calls the query to create a new order with all the cart items and quantities
    xhr.open("POST", "http://localhost:5001/checkout", true);
    xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');


    var allTransaction = {
        transaction: transactionOverview,
        details: cartAndDetails
    }
    var JSONformatdata = JSON.stringify(allTransaction);

    xhr.send(JSONformatdata);
```

```
    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
        }
    }
    window.location.replace('http://localhost:5001/main')
  }
}
```

User catalogue page (browsing)

catalogue.html

```
<!DOCTYPE html>
<html lang="en">

<head>
 <meta charset="UTF-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Main Menu</title>
 <script src="/js/catalogue.js"></script>
 <script src="/js/client.js"></script>
 <link rel="stylesheet" href="/css/catalogue.css">
 <link rel="stylesheet" href="/css/client.css">
   <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
</head>

<body>
 <div class="header" colspan="2">
   <img class="logo" src =
"/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
   <div class="header-right">
     <a class="active" onclick="home();" href="#home">Home</a>

     <a onclick="logOut();">Logout</a>
     <a><i onclick="goCart();" id ="cart" class='bx bx-cart' ></i></a>

   </div>
 </div>
   <div class="maincontent">
     <div class="maincontent">
       <div class="wrap">
           <div class="search">
              <input onkeyup="filterByNameOrCategory()" type="text"
id="searchbar"class="searchTerm" placeholder="Search product by category or name">
              <button type="submit" class="searchButton">
               <i class='bx bx-search'></i>
             </button>
           </div>
```

```
        </div>
     <div class="productcontainers" id="container">

   </div>
 </div>
</body>

</html>
```

catalogue.js

```
window.onload = function () {
   loadCatalogue();
};

//Retrieves all the catalogue in the database
function loadCatalogue() {
   createXhrRequest("GET", 'http://localhost:5001/catalogue/all', function (err,
response) {
       if (err) { console.log("Error!"); }
       allCatalogue = JSON.parse(response);

       for (var item in allCatalogue.product) {
           document.getElementById('container').innerHTML += '<div class="card">
<img class="productimage" src = "' + allCatalogue.product[item].randomImage + '">
<div class="productname"><b>' + allCatalogue.product[item].name + '</b></div> <div
class="categoryname">' + allCatalogue.product[item].category + '</div> <div
class="viewbutton" onclick = "viewMore(' + allCatalogue.product[item].productID +
');"> View</div> </div>'
       }
   });
}

function viewMore(productID) {
   window.location.replace('http://localhost:5001/productinfo/' + productID)
}

//Filters the product list by the inputted keyword
//Source:
https://www.youtube.com/watch?v=RVrHC__Tkx0&t=934s&ab_channel=ABNationProgrammers
function filterByNameOrCategory() {
   var value = document.getElementById('searchbar').value
   var cardContainer = document.getElementById('container')
   //Cycles through all the products in the list
   for (var y = 0; y < cardContainer.getElementsByClassName('card').length; y++) {
       if
(document.getElementsByClassName('card')[y].querySelector(".categoryname").innerTex
t.toLowerCase().includes(value) ||
document.getElementsByClassName('card')[y].querySelector(".productname").innerText.
toLowerCase().includes(value)) {
```

```
            //if it matches, do nothing
            cardContainer.getElementsByClassName('card')[y].style.display = ""
        }
        else {
            //if it doesnt match, hide the product listing
            cardContainer.getElementsByClassName('card')[y].style.display = "none"

        }
    }

}
```

User Delivery Page - check status of ongoing orders

delivery.html

```
<!DOCTYPE html>
<html lang="en">

<head>
 <meta charset="UTF-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Main Menu</title>
 <script src="/js/delivery.js"></script>
 <link rel="stylesheet" href="/css/delivery.css">
 <script src="/js/client.js"></script>
 <link rel="stylesheet" href="/css/client.css">
 <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
</head>

<body>
 <div class="header" colspan="2">
   <div id="header-left">
     <img class="logo"
src="/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
   </div>
   <div class="header-right">
     <a class="active" onclick="home();" href="#home">Home</a>
     <a onclick="logOut();">Logout</a>
     <a><i onclick="goCart();" id="cart" class='bx bx-cart'></i></a>
   </div>
 </div>
 <div class="maincontent">
   <div class="filtermenu">
       <div class="allfilters"> Status <select onchange="filterByStatus()"
id="status">
           <option value=""> </option>
           <option value="Payment Uploaded">Payment Uploaded</option>
           <option value="Waiting Payment">Waiting Payment</option>
```

```
          <option value="Delivering">Delivering</option>
          <option value="Waiting Approval">Waiting Approval</option>
        </select>
        Total Amount <select onchange="sortByAmount()" id="amount">
          <option value=""> </option>
          <option value="Ascending">Ascending </option>
          <option value="Descending">Descending</option>
        </select>
        Date <select onchange="sortByDate()"id="date">
          <option value="1"> </option>
          <option value="Recent">Recent </option>
          <option value="Latest">Latest</option>
        </select>
      </div>
  </div>
  <div class="tablecontainer">

    <table id="tables">
      <tr id="header">
        <th id="first">
          TransactionID
        </th>
        <th id="column">
          Status
        </th>
        <th id="column">
          Total Amount
        </th>
        <th id="column">
          Order Date
        </th>
        <th id="last">
        </th>
      </tr>
    </table>
  </div>

<div class="modal-container">
  <div class="actualModal">
    <h1>Transacton Details</h1>
    <div class="transcdet">
      <div id="transcID">
      </div>
      <div id="ordered">
      </div>
      <div id="orderStatus">
      </div>
    </div>
    <table id="table-list">
      <tr id="header">
```

```
          <th id="first-list">
            Product
          </th>
          <th id="column">
            Category
          </th>
          <th colspan="2" id="last-list">
            Quantity
          </th>
        </tr>
      </table>
      <div id="retbutton" onclick="back();"> Return</div>
    </div>
  </div>
<div class="modal-container-upload">
    <div class="actualModal-upload">
      <div onclick="uploadImage()" class="uploadbutton"> Upload</div>
      <input type="file" id="myFile" accept="image/*" name="profile-files">
      <div id="retbutton" onclick="closeUpload();"> Return</div>

    </div>
  </div>
</div>

</body>

</html>
```

delivery.js

```
var allcatalogue;

window.onload = function () {
    loadDelivery();
};

let transactionKey;
var allTransactions

//Load all the transactions/orders from the specific user
function loadDelivery() {
    createXhrRequest("GET", 'http://localhost:5001/getdelivery', function (err,
response) {
        if (err) { console.log("Error!"); }
        alldelivery = JSON.parse(response);
        allTransactions = alldelivery
    });
    var table = document.getElementById('tables')
    var count = 1;
    for (delivery in allTransactions) {
```

```
        //Take all the transactions that are not completed - which means it is
ongoing
        if (allTransactions[delivery].transaction.orderStatus != "Completed") {
            //Insert into the table
            transactionKey = allTransactions[delivery].transaction.idTransaction
            var row = table.insertRow(count);
            var transactionID = row.insertCell(0);
            var status = row.insertCell(1);
            var total = row.insertCell(2);
            var dateOrdered = row.insertCell(3);
            var button = row.insertCell(4);
            transactionID.innerHTML = transactionKey.toString();
            status.innerHTML =
allTransactions[delivery].transaction.orderStatus.toString();
            total.innerHTML =
parseInt(allTransactions[delivery].transaction.totalValue.toString());
            dateOrdered.innerHTML =
allTransactions[delivery].transaction.dateOrdered.split('T')[0];
            button.innerHTML = '<div onclick ="viewMore(this);" class="button">View
More</div>'
            if (allTransactions[delivery].transaction.orderStatus == "Waiting
Payment" ||allTransactions[delivery].transaction.orderStatus == "Payment Rejected")
{
                button.innerHTML += '<div onclick ="viewUpload(this);"
class="button-waiting">Upload</div>'
            }
            else if (allTransactions[delivery].transaction.orderStatus ==
"Delivering") {
                button.innerHTML += '<div onclick ="completed(this);"
class="button-complete">Completed</div>'
            }
            count++
        }
    }
}

//View a record in better detail
function viewMore(selectedRow) {
    var modal = document.querySelector('.modal-container');
    var table = document.getElementById('tables')
    var rownumber = selectedRow.parentElement.parentElement.rowIndex
    modal.classList.add('container-active');
    var idRow = document.getElementById('transcID');
    var dateOrdered = document.getElementById('ordered');
    var orderStatus = document.getElementById('orderStatus')
    var transactionID = table.rows[rownumber].cells[0].innerHTML;
    //Takes all the transaction details - products, variants, quantities ordered...
    for (transaction in allTransactions) {
        if (allTransactions[transaction].transaction.idTransaction == transactionID)
{
```

```
            idRow.innerHTML += 'Transaction ID: ' +
table.rows[rownumber].cells[0].innerHTML;
            dateOrdered.innerHTML += 'Date Ordered: ' +
table.rows[rownumber].cells[3].innerHTML;
            orderStatus.innerHTML += 'Status: ' +
table.rows[rownumber].cells[1].innerHTML;
        }
    }
    addProductList(transactionID)
}

//Add all the product ordered into the popup menu in the form of a table
function addProductList(transactionID) {
    var count = 1;
    var table = document.getElementById('table-list')
    for (transaction in allTransactions){
        if (allTransactions[transaction].transaction.idTransaction == transactionID)
{
            var allDetails = allTransactions[transaction].details
            for (detail in allDetails) {

                var row = table.insertRow(count);
                var product = row.insertCell(0);
                var category = row.insertCell(1);
                var quantity = row.insertCell(2);
                product.innerHTML = allDetails[detail].products.name +
allDetails[detail].variant.variantName
                category.innerHTML = allDetails[detail].products.category;
                quantity.innerHTML = allDetails[detail].details.quantity;
                count++
            }
        }
    }

}

//Close the popup menu for viewing transaction/order details
function back() {
    var modal = document.querySelector('.modal-container');
    modal.classList.remove('container-active');
    var transcID = document.getElementById('transcID');
    var dateOrdered = document.getElementById('ordered');
    var status = document.getElementById('orderStatus')
    transcID.removeChild(transcID.firstChild);
    dateOrdered.removeChild(ordered.firstChild);
    status.removeChild(status.firstChild);
    deleteRows()
}
```

```
//Close window for uploading proof of payment if the status of the order is
"waiting payment"
function closeUpload() {
   var modal = document.querySelector('.modal-container-upload');
   modal.classList.remove('container-active');
}

function deleteRows() {
   var tablelists = document.getElementById('table-list');
   var numofrows = tablelists.rows.length;
   for (let x = numofrows - 1; x > 0; x--) {
       tablelists.deleteRow(x)
   }
}

//Update a record into completed once it has been delivered
function completed(selectedRow) {
   var table = document.getElementById('tables')
   var rownumber = selectedRow.parentElement.parentElement.rowIndex
   var transactionID = table.rows[rownumber].cells[0].innerHTML;
   for (delivery in alldelivery) {
       if (alldelivery[delivery].idTransaction = transactionID) {
           var userID = alldelivery[delivery].userID
           var dateOrdered = alldelivery[delivery].dateOrdered
           var total = alldelivery[delivery].Total
       }
   }
   var data = {};
   data.orderStatus = "Completed";
   var JSONformatdata = JSON.stringify(data);
   var xhr = new XMLHttpRequest();
   xhr.open("PUT", "http://localhost:5001/updateComplete/" +
transactionID.toString(), true);
   xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
   xhr.onload = function () {
       var users = JSON.parse(xhr.responseText);
       if (xhr.readyState == 4 && xhr.status == "200") {
       } else {
           console.error(users);
       }
   }
   xhr.send(JSONformatdata);
   location.reload();
}

var transactionID

function viewUpload(selectedRow) {
   var modal = document.querySelector('.modal-container-upload');
   var table = document.getElementById('tables')
```

```
    var rownumber = selectedRow.parentElement.parentElement.rowIndex
    var idTransaction = table.rows[rownumber].cells[0].innerHTML;
    transactionID = idTransaction
    modal.classList.add('container-active');
}

//Function that is called when the user uploads an image
function uploadImage() {
    //If it is empty dont proceed with the process
    if (document.getElementById('myFile').files[0] == undefined) {
        alert("You need to upload an image")
    }
    else {
        //Create a formdata to store the images
        var formData = new FormData()
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "http://localhost:5001/uploadpayment/" + transactionID,
true);
        //Retrieve the uploaded images and store in "uploadedfile"
        formData.append("uploaded_file",
document.getElementById('myFile').files[0]);
        xhr.send(formData);
        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
            }
        }
        var newstatus = {
            orderStatus: "Payment Uploaded"
        }
        var JSONformatdata = JSON.stringify(newstatus);
        var xhr = new XMLHttpRequest();
        xhr.open("PUT", "http://localhost:5001/updateStatus/" + transactionID,
true);
        xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
            }
        }
        //Send the collected data
        xhr.send(JSONformatdata);
        window.location.reload()
    }
}

//Source:
https://newbedev.com/javascript-number-format-javascript-rupiah-code-example
function convertRup(numb) {
    const format = numb.toString().split('').reverse().join('');
    const convert = format.match(/\d{1,3}/g);
    const rupiah = convert.join('.').split('').reverse().join('')
```

```
    return rupiah
}
```

Editing user information - user side

useredit.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Edit Information</title>
        <script src="/js/useredit.js"></script>
        <script src="/js/client.js"></script>

        <link rel="stylesheet" href="/css/useredit.css">
        <link rel="stylesheet" href="/css/client.css">

        <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
    </head>
<body>
    <div class="header" colspan="2">
        <img class="logo" src =
"/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
        <div class="header-right">
          <a onclick="home()">Home</a>

          <a onclick="logOut();">Logout</a>
          <a><i onclick="goCart();" id ="cart" class='bx bx-cart' ></i></a>

        </div>
    </div>
    <div class="maincontent" >
        <ul class="editformlist" style="list-style-type:none">
            <li id="image">
            </li>
            <li id="name">
                Name
            </li>
            <li id="storename">
                Store Name
            </li>
            <li id="address">
                Address
            </li>
            <li id="phonenum">
```

```
                Phone number
            </li>
            <li id="email">
                Email
            </li>
            <li>
                <div class="confirmbutton" onclick="updateUser()"> Update </div>
            </li>
        </ul>
        <div class="success">
            <div class="actual-success">
                <ul style="list-style-type:none;" class="groupofforms">
                    <i class='bx bx-check-circle' id = "cbutton"></i>
                    <h3 id="senttext-success">Success! </h3>
                    <p>You have edited your information</p>
                </ul>
            </div>
        </div>
    </div>
</div>
</html>
```

useredit.js

```
window.onload = function () {
    loadUserInformation();
};

var userInformation

function loadUserInformation() {
    //Load the specific user information
    createXhrRequest("GET", 'http://localhost:5001/getuserinfo', function (err,
response) {
        if (err) { console.log("Error!"); }
        var allUserInformation = JSON.parse(response);
        userInformation =allUserInformation;

        document.getElementById('image').innerHTML += '<img  class="profilepic"
src="' + allUserInformation.image + '" /> <br><input type="file"
id="myFile"accept="image/*"   name="profile-files" > <div onclick = "uploadImage()"
class="uploadbutton"> Upload</div>'
        document.getElementById('name').innerHTML += '<input type="text"
id="inputname"  value="' + allUserInformation.name + '">'
        document.getElementById('storeName').innerHTML += '<input type="text"
id="inputstore"  value="' + allUserInformation.storeName + '">'
        document.getElementById('address').innerHTML += '<input type="text"
id="inputaddress"  value="' + allUserInformation.address + '">'
        document.getElementById('phonenum').innerHTML += '<input type="text"
id="inputphone"  value="' + allUserInformation.phonenum + '">'
```

```
        document.getElementById('email').innerHTML += '<input type="text"
id="inputemail"  value="' + allUserInformation.email + '"> '
    });
}

function updateUser() {

    var formData = new FormData()
    //collect all the inputted information whether it has been updated or not
    var name = document.getElementById("inputname").value;
    var storeName = document.getElementById("inputstore").value;
    var address = document.getElementById("inputaddress").value;
    var phonenum = document.getElementById("inputphone").value;
    var email = document.getElementById("inputemail").value;
    //none of the fields can be empty
    if (name == "" || storeName == "" || address == "" || phonenum == "" || email ==
"") {
        alert("Cant be empty")
    } else {

        var newUserInformation = {
            name: name,
            storeName: storeName,
            address: address,
            phonenum: phonenum,
            email: email,
        }

        var JSONformatdata = JSON.stringify(newUserInformation);
        var xhr = new XMLHttpRequest();
        xhr.open("PUT", "http://localhost:5001/uploaduserinfo", true);
        xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');

        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
            }

        }
        //send the new data
        xhr.send(JSONformatdata);
        window.location.reload()
    }
}

//Upload a new profile picture
function uploadImage(){

        if (document.getElementById('myFile').files[0]==undefined){
            alert("You need to upload an image")
        }
```

```
        else{
            var formData = new FormData()
            var xhr = new XMLHttpRequest();
            xhr.open("POST", "http://localhost:5001/uploadimage", true);
            formData.append("uploaded_file",
document.getElementById('myFile').files[0]);
            xhr.send(formData);

            xhr.onreadystatechange = function () {
                if (xhr.readyState == 4 && xhr.status == 200) {
                }
            }
            window.location.reload()
        }
}
```

Order history page - viewing completed transactions

orderhist.html

```
<!DOCTYPE html>
<html lang="en">

<head>
 <meta charset="UTF-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Main Menu</title>
 <script src="/js/orderhist.js"></script>
 <script src="/js/client.js"></script>
 <link rel="stylesheet" href="/css/client.css">
 <link rel="stylesheet" href="/css/orderhist.css">
 <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
</head>

<body>
 <div class="header" colspan="2">
   <div id="header-left">
     <img class="logo"
src="/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
   </div>
   <div class="header-right">
     <a class="active" onclick="home();" href="#home">Home</a>
     <a onclick="logOut();">Logout</a>
     <a><i onclick="goCart();" id="cart" class='bx bx-cart'></i></a>
   </div>
 </div>
 <div class="filtermenu">
   <div class="allfilters">
```

```
    Total Amount <select onchange="sortByAmount()" id="amount">
      <option value=""> </option>
      <option value="Ascending">Ascending </option>
      <option value="Descending">Descending</option>
    </select>
    Date <select onchange="sortByDate()"id="date">
      <option value=""> </option>
      <option value="Recent">Recent </option>
      <option value="Latest">Latest</option>
    </select>
  </div>
</div>
 <div class="tablecontainer">

  <table id="tables">
    <tr id="header">
      <th id="first">
        TransactionID
      </th>
      <th id="column">
        Status
      </th>
      <th id="column">
        Total Amount
      </th>
      <th id="column">
        Order Date
      </th>
      <th id="last">
      </th>
    </tr>
  </table>
 </div>
 <div class="modal-container">
  <div class="actualModal">
    <h1>Transacton Details</h1>
    <div class="transcdet">
      <div id="transcID">
      </div>
      <div id="ordered">
      </div>
      <div id="status">
      </div>
    </div>

    <table id="table-list">
      <tr id="header">
        <th id="first-list">
          Product
        </th>
```

```
        <th id="column">
          Category
        </th>
        <th colspan="2"  id="column">
          Quantity
        </th>
      </tr>

    </table>

    <div id="retbutton" onclick="back();"> Return</div>

  </div>
 </div>


</body>

</html>
```

orderhist.js

```
var allDelivery;

window.onload = function () {
   loadCompletedTransactions();
};

function loadCompletedTransactions() {
   createXhrRequest("GET", 'http://localhost:5001/getdelivery', function (err,
response) {
       if (err) { console.log("Error!"); }
       allDelivery = JSON.parse(response);
       allTransactions = allDelivery
   });
   var table = document.getElementById('tables')
   var count = 1;
   for (transaction in allTransactions) {
       if (allTransactions[transaction].transaction.orderStatus == "Completed" ) {
           transactionKey = allTransactions[transaction].transaction.idTransaction
           var row = table.insertRow(count);
           var transactionID = row.insertCell(0);
           var status = row.insertCell(1);
           var total = row.insertCell(2);
           var dateOrdered = row.insertCell(3);
           var button = row.insertCell(4);
           transactionID.innerHTML = transactionKey.toString();
           status.innerHTML =
allTransactions[transaction].transaction.orderStatus.toString();
```

```
            total.innerHTML =
parseInt(allTransactions[transaction].transaction.totalValue.toString());
            dateOrdered.innerHTML =
allTransactions[transaction].transaction.dateOrdered.split('T')[0];
            button.innerHTML = '<div onclick ="viewMore(this);" class="button">View
More</div>'
            if (allTransactions[transaction].transaction.orderStatus == "Waiting
Payment" ||allTransactions[transaction].transaction.orderStatus == "Payment
Rejected") {
                button.innerHTML += '<div onclick ="viewUpload(this);"
class="button-waiting">Upload</div>'
            }
            else if (allTransactions[transaction].transaction.orderStatus ==
"Delivering") {
                button.innerHTML += '<div onclick ="completed(this);"
class="button-complete">Completed</div>'
            }
            count++
        }
    }
}

function viewMore(selectedRow) {
    var modal = document.querySelector('.modal-container');
    var table = document.getElementById('tables')
    var rownumber = selectedRow.parentElement.parentElement.rowIndex
    modal.classList.add('container-active');
    var transacationID = document.getElementById('transcID');
    var dateOrdered = document.getElementById('ordered');
    var status = document.getElementById('status')
    var idTransaction = table.rows[rownumber].cells[0].innerHTML;
    transacationID.innerHTML += 'Transaction ID: ' + idTransaction;
    dateOrdered.innerHTML += 'Date Ordered: ' +
table.rows[rownumber].cells[3].innerHTML;
    status.innerHTML += 'Status: ' + table.rows[rownumber].cells[1].innerHTML;
    addProducts(idTransaction)

}

function addProducts(transactionID) {
    var count = 1;
    var table = document.getElementById('table-list')
    for (transaction in allTransactions){
        if (allTransactions[transaction].transaction.idTransaction == transactionID)
{
            var allDetails = allTransactions[transaction].details
            for (detail in allDetails) {
                var row = table.insertRow(count);
                var product = row.insertCell(0);
                var category = row.insertCell(1);
```

```
                var quantity = row.insertCell(2);
                product.innerHTML = allDetails[detail].products.name +
allDetails[detail].variant.variantName
                category.innerHTML = allDetails[detail].products.category;
                quantity.innerHTML = allDetails[detail].details.quantity;
                count++
            }
        }
    }
}

function back() {
    var modal = document.querySelector('.modal-container');
    modal.classList.remove('container-active');
    var transcID = document.getElementById('transcID');
    var dateO = document.getElementById('ordered');
    var status = document.getElementById('status')
    transcID.removeChild(transcID.firstChild);
    dateO.removeChild(dateO.firstChild);
    status.removeChild(status.firstChild);
    deleteRows()
}

function deleteRows() {
    var tablelists = document.getElementById('table-list');
    var numofrows = tablelists.rows.length;
    for (let x = numofrows - 1; x > 0; x--) {
        tablelists.deleteRow(x)
    }
}
```

Product page - browse a specific product in detail

product.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Main Menu</title>
    <script src="/js/product.js"></script>
    <script src="/js/client.js"></script>

    <link rel="stylesheet" href="/css/product.css">
    <link rel="stylesheet" href="/css/client.css">
    <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
```

```
</head>

<body>
 <div class="header" colspan="2">
   <img class="logo" src =
"/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
   <div class="header-right">
     <a class="active" onclick="home();" href="#home">Home</a>

     <a onclick="logOut();">Logout</a>
     <a><i onclick="goCart();" id ="cart" class='bx bx-cart' ></i></a>

   </div>
 </div>
   <div id = "maincontent"class="maincontent">
    <div id="image">
     <div class="imagebutton" onclick ="move(-1)" id="leftbutton"><i class='bx
bxs-chevron-left' ></i></div>
     <div class="imagebutton" id="rightbutton" onclick ="move(1)"><i class='bx
bxs-chevron-right' ></i></div>
    </div>
    <div id="payment">
      <ul class="productdetails" style="list-style-type:none"
style="line-height:1.4;">
        <li id = "name"></li>
        <li  id = "category"> </li>
        <li id = "description"></li>
        <li id = "price"> </li>
        <li id = "stock"> </li>
        <li id = "model"> Model type:
          <select id="modeltype" onchange="changeValues();" >
          </select>
        </li>
        <li id = "quantity">Quantity: </li>
        <li ><div onclick =  "addCart()"id="addtocart">Add to cart<div> </li>
      </ul>
    </div>
    <div class="success">
      <div class="actual-success">
          <ul style="list-style-type:none;" class="groupofforms">
              <i class='bx bx-check-circle' id = "cbutton"></i>
              <h3 id="senttext-success">Success! </h3>
              <p>You have added an item to your cart</p>
          </ul>
      </div>
   </div>
 </div>
</body>
</html>
```

product.js

```
var allcatalogue;

window.onload = function () {
   loadProductInformation();
}

var allProductInformation

var count = 0;
var position;

function loadProductInformation() {
   var currentLocation = window.location.pathname;
   productID = currentLocation.replace("/productinfo/", "");
   createXhrRequest("GET", 'http://localhost:5001/getspecific/' + productID,
function (err, response) {
       if (err) { console.log("Error!"); }
       var products = JSON.parse(response);
       allProductInformation = products

       document.getElementById('image').innerHTML += '<img id="images" src="' +
products.allVariant[0].images[0].imageaddress + '"> '
       document.getElementById('name').innerHTML = '<b>' + products.product[0].name
+ '</b>'
       document.getElementById('description').innerHTML = 'Description: ' +
products.product[0].description

       document.getElementById('category').innerHTML += 'Category: ' +
products.product[0].category
       document.getElementById('price').innerHTML = '<b>Rp.</b> <b>' +
convertRup(products.allVariant[0].variant.price) + '</b>'
       document.getElementById('stock').innerHTML = 'Stock: <b>' +
products.allVariant[0].variant.quantity + '</b>'
       document.getElementById('quantity').innerHTML += ' <input type="number"
id="inputquantity" value=1> '
       position = 0;

       for (variant in products.allVariant) {
           document.getElementById('modeltype').innerHTML += '<option value="' +
products.allVariant[variant].variant.variantName + '">' +
products.allVariant[variant].variant.variantName + '</option>'
       }
   });
}

//Changes the images by cycling through the array of images
function move(num) {
```

```
    var child = document.getElementById("images");
    var parent = document.getElementById("image");
    parent.removeChild(child);

    document.getElementById('image').innerHTML += '<img id="images" src="' +
allProductInformation.allVariant[position].images[count].imageaddress + '"> '
    count = count + num
    if (count > allProductInformation.allVariant[position].images.length - 1) {
        count = 0;
    }
}


var index=0;

//Updates the information such as images, variant name, quantities when the user
selects a differet model or variant of the product
function changeValues() {
    for (variant in allProductInformation.allVariant) {
        if (allProductInformation.allVariant[variant].variant.variantName ==
document.getElementById('modeltype').value) {
            index = variant
            position = variant;
            var child = document.getElementById("images");
            var parent = document.getElementById("image");
            parent.removeChild(child);
            //Update variant information
            document.getElementById('image').innerHTML += '<img id="images" src="' +
allProductInformation.allVariant[variant].images[0].imageaddress + '"> '
            document.getElementById('price').innerHTML = '<b>Rp.</b> <b>' +
convertRup(allProductInformation.allVariant[variant].variant.price) + '</b>'
            document.getElementById('stock').innerHTML = 'Stock: <b>' +
allProductInformation.allVariant[variant].variant.quantity + '</b>'

        }
    }
}

//Add an item to cart
function addCart(){
    var exist=false;
    createXhrRequest("GET", 'http://localhost:5001/getcartitems', function (err,
response) {
        if (err) { console.log("Error!"); }
        var allCartItems = JSON.parse(response);
        for (item in allCartItems){

if(allCartItems[item].variantID==allProductInformation.allVariant[index].variant.id
variant){
                exist=true;
```

```
            }
        }
    })
    if(exist==false){
        //If the quantity is less than the available stock, notify the user
        if
(document.getElementById('inputquantity').value>allProductInformation.allVariant[in
dex].variant.quantity){
            alert("Input value greater than stock value")
        }else{
            var newcart = {
                variantID:
allProductInformation.allVariant[index].variant.idvariant,
                quantity: document.getElementById('inputquantity').value,
                productID:
allProductInformation.allVariant[index].variant.productID,
            }
            var xhr = new XMLHttpRequest();
            //else add the item into user cart
            xhr.open("POST","http://localhost:5001/addcartitem",true);
            xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');

            var JSONformatdata = JSON.stringify(newcart);

            xhr.send(JSONformatdata);

            xhr.onreadystatechange = function(){
            if(xhr.readyState == 4 && xhr.status == 200)
            {
            }
            var modal = document.querySelector('.success');
            modal.classList.add('success-active');
            setTimeout(function () {
                window.location.replace('http://localhost:5001/catalogue')
            }, 2000);
            }
        }
    }
    else{
        alert("You have already placed it in cart!")
    }
}
```

Logistics Page Main Menu

mainl.html

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Logistics</title>
   <script src="/js/mainl.js"></script>
   <script src="/js/admin.js"></script>
   <link rel="stylesheet" href="/css/mainl.css">
   <link rel="stylesheet" href="/css/admin.css">

       <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
</head>
<body>
   <div id="header">
       <img id="logo" src =
"/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">


   </div>
   <div class="maincontent">

       <table id="maintable-log">
           <tr id="tableheader">
             <th id="first">
               Transaction ID
             </th>
             <th id="column">
               Name
           </th>
             <th id="column-address">
                 Store Name
             </th>

             <th id="column">
               Status
             </th>
             <th id="last">
             </th>
           </tr>
         </table>

         <div class="create-form">
           <div class="actual-form-create">
               <div class="view-form-header">
                   View Transaction
               </div>
               <div id="addressform">  </div>
               <table id="maintable-det">
                 <tr id="tableheader">
                   <th id="first">
```

```
                    Product Name
                  </th>
                  <th id="column">
                    Category
                </th>
                  <th id="column">
                    Model
                </th>
                  <th id="last">
                      Quantity
                  </th>
                </tr>
              </table>

              <div class="cancelbutton" onclick="removeRows()"> Cancel</div>
          </div>
      </div>
  </div>
</body>
</html>
```

mainl.js

```
//Source:
https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpReques
t
var createXhrRequest = function (httpMethod, url, callback) {
   //Create a XML HTTP Request
   var xhr = new XMLHttpRequest();
   //The method accepts the http method (post,put,delete, get...) and the URL to be
requested
   xhr.open(httpMethod, url);
   //When the request is complete uses a callback so the process can come back to
retrieve the response
   xhr.onload = function () {
       callback(null, xhr.response);
   };
   //Called when a request wasnt successful
   xhr.onerror = function () {
       callback(xhr.response);
   };
   //Not necessary for get requests, but POST/PUT requires this
   xhr.send();
}

window.onload = function () {
   loadTransactions();
};

var alltransactions
```

```
//Load the transactions that relevant to the logistics department
function loadTransactions() {
   var table = document.getElementById('maintable-log')
   var count = 1;
   //
   createXhrRequest("GET", 'http://localhost:5001/getalllogistics' , function (err,
response) {
       if (err) { console.log("Error!"); }
       allTransactions = JSON.parse(response);
       alltransactions = allTransactions
       for (var x = 0; x < allTransactions.length; x++) {
           //insert all the values into a table
           var row = table.insertRow(count);
           var transactionID = row.insertCell(0);
           var name = row.insertCell(1);
           var storeName = row.insertCell(2);
           var status = row.insertCell(3);
           var button = row.insertCell(4);
           transactionID.innerHTML =
allTransactions[x].transaction.idTransaction.toString();
           name.innerHTML =  allTransactions[x].user.name
           storeName.innerHTML = allTransactions[x].user.storeName
           status.innerHTML =  allTransactions[x].transaction.orderStatus
           //Include different buttons for different types of statuses
           if ( allTransactions[x].transaction.orderStatus=="Processing"){
               button.innerHTML += '<div onclick ="view(this);"
class="viewtransc">View Transaction</div>  <div onclick ="confirmChange(this);"
class="update">Delivering</div>'
           }
           else if (allTransactions[x].transaction.orderStatus=="Payment
Approved"){
               button.innerHTML += '<div onclick ="view(this);"
class="viewtransc">View Transaction</div>  <div onclick ="confirmChange(this);"
class="update">Processing</div>'
           }
           else{
               button.innerHTML += '<div onclick ="view(this);"
class="viewtransc">View Transaction</div> '

           }
           count++
       }

   });
}

function view(row){
   var table = document.getElementById('maintable-log')
   var count = 1;
```

```
    var rowNumber = row.parentElement.parentElement.rowIndex
    var tID = table.rows[rowNumber].cells[0].innerHTML;
    for(transaction in allTransactions){
        if (allTransactions[transaction].transaction.idTransaction ==
parseInt(tID)){
            document.getElementById('addressform').innerHTML+= "Address:
"+allTransactions[transaction].user.address
            var table1 = document.getElementById('maintable-det')
            for (var x = 0; x <
allTransactions[transaction].transactionDetails.length; x++) {
                //When user clicks view, load all the transaction details which
include - products ordered, quantity, model name, date ordered...
                var row = table1.insertRow(count);
                var productName = row.insertCell(0);
                var category = row.insertCell(1);
                var totalSales = row.insertCell(2);
                var quantity = row.insertCell(3);
                productName.innerHTML =
allTransactions[transaction].transactionDetails[x].products.name.toString();
                category.innerHTML =
allTransactions[transaction].transactionDetails[x].products.category
                totalSales.innerHTML =
allTransactions[transaction].transactionDetails[x].variant
                quantity.innerHTML =
allTransactions[transaction].transactionDetails[x].details.quantity
                count++
            }
        }
    }
    var modal = document.querySelector('.create-form');
    modal.classList.add('create-form-active');
}

//Remove the rows and details in html once the user closes the popup
function removeRows(){
    var address = document.getElementById('addressform');
    address.removeChild(address.firstChild);
    var tableLists = document.getElementById('maintable-det');
    var numofrows = tableLists.rows.length;
    for (let x = numofrows - 1; x > 0; x--) {
        tableLists.deleteRow(x)
    }
    var modal = document.querySelector('.create-form');
    modal.classList.remove('create-form-active');
}

//This function is mainly used to update records
function confirmChange(row){
    var table = document.getElementById('maintable-log')
    var rowNumber = row.parentElement.parentElement.rowIndex
```

```
   var transactionID = table.rows[rowNumber].cells[0].innerHTML;
   for(transaction in allTransactions){
       if (allTransactions[transaction].transaction.idTransaction ==
parseInt(transactionID)){
           if (allTransactions[transaction].transaction.orderStatus == "Payment
Approved")
           var newStatus = {
               orderStatus: "Processing"
           }
           else if(allTransactions[transaction].transaction.orderStatus ==
"Processing"){
               var newStatus = {
                   orderStatus: "Delivering"
               }
           }
           var JSONformatdata = JSON.stringify(newStatus);
           var xhr = new XMLHttpRequest();
           //Calls the api to update the record
           xhr.open("PUT",
"http://localhost:5001/updateStatus/"+transactionID.toString(), true);
           xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
           xhr.onload = function () {
               if (xhr.readyState == 4 && xhr.status == "200") {
               } else {
                   console.error(users);
               }
           }
           xhr.send(JSONformatdata);
           location.reload()

       }
}}
```

Admin main menu

mainmenua.html

```
<!DOCTYPE html>
<html lang="en">
   <head>
       <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <script src="/js/mainmenua.js"></script>
   <script src="/js/admin.js"></script>
   <link rel="stylesheet" href="/css/mainmenua.css">
   <link rel="stylesheet" href="/css/admin.css">
       <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
```

```
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.4.0/Chart.min.js"></script>

    </head>
<body>
    <div id="header">
        <img id="logo" src =
"/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
            <i class='bx bx-menu' onclick="openSidebar();" id ="iconmenu" ></i>

    </div>


    <div class= "sidebar">

        <ul id = "dashboardlist" style="list-style-type:none">
            <li class="lists">
                <i class='bx bxs-dashboard' ></i> Dashboard
            </li>
            <li class="lists" onclick="orders();">
                <i class='bx bx-cart'></i>
                Orders
            </li>
            <li class="lists" onclick="users();">
                <i class='bx bx-user' ></i> Users

            </li>
            <li class="lists" onclick="transactions();">
                <i class='bx bx-wallet-alt' ></i> Transactions

            <li class="lists" onclick="catalogue();">
                <i class='bx bx-package' ></i> Catalogue
            </li>
            <li class="lists" onclick="logOut();">
                <i class='bx bx-log-out'  ></i>  Logout
            </li>
        </ul>
        <div class="profile">
            profile
        </div>
    </div>
    <div class="maincontent">
        <div class="dashboard">
            <div id ="statheader">
                <div id="customers">
                    <ul id = "customers-list" style="list-style-type:none">
                            <li><i class='bx bx-user' ></i>Customers </li>
                    </ul>

                </div>
```

```
            <div id="products">
                <ul id = "products-list" style="list-style-type:none">
                    <i class='bx bx-package' ></i> Catalogue
            </ul>

            </div>
            <div id="transactions">
                <ul id = "transactions-list" style="list-style-type:none">
                    <i class='bx bx-wallet-alt' ></i> Transactions

            </ul>

            </div>
        </div>

    <div class="tables">
        <div id="topcustomers" >
            <table  id="maintable-customers" >
                <tr id="tableheader">
                  <th id="column">
                   ID
                  </th>
                  <th id="column">
                    User
                   </th>
                  <th id="column">
                    Total Sales
                </th>

                </tr>
              </table>
        </div>
        <div id = "topproducts">
            <canvas id="chartproducts" ></canvas>
        </div>
    </div>
        <div id ="graph">
            <div class="zoom" onclick ="changeRevenue(this)"> <i class='bx
bx-search-alt-2'></i> Zoom</div>
            <canvas class="charts" id="chartrevenue"></canvas>
            <div  class = "zoom" onclick ="changeSales(this)"> <i class='bx
bx-search-alt-2'></i> Zoom</div>

            <canvas class="charts" id="chartsales"></canvas>

        </div>

    </div>
  </div>
</body>
```

```
</html>
```

mainmenua.js

```
window.onload = function () {
   loadStatistics()
};


var allStatistics
var revenueChart
var salesChart

function loadStatistics() {
   createXhrRequest("GET", 'http://localhost:5001/getalldashboard', function (err,
response) {
       if (err) { console.log("Error!"); }
       var data = JSON.parse(response);
       allStatistics = data
       //Load all the required statistics upon the page opening by calling the
above api
       document.getElementById('customers-list').innerHTML += '<li
class="statnumbers">' + allStatistics.customers.toString() + '</li>'
       document.getElementById('products-list').innerHTML += '<li
class="statnumbers">' + allStatistics.products.toString() + '</li>'
       document.getElementById('transactions-list').innerHTML += '<li
class="statnumbers">' + allStatistics.transactions.toString() + '</li>'

       var userTable = document.getElementById('maintable-customers')
       var count = 1;

       var productNames = []
       var productSales = []

       for (var x = 0; x < 5; x++) {
           //Create an arra for the most popular products by total sales
           productNames.push(allStatistics.orderedProducts[x].name)
           productSales.push(allStatistics.orderedProducts[x].totalsales)
           var row = userTable.insertRow(count);
           var userID = row.insertCell(0);
           var userName = row.insertCell(1);
           var userTotalSales = row.insertCell(2);
           userID.innerHTML = allStatistics.orderedCustomers[x].userID.toString();
           userName.innerHTML = allStatistics.orderedCustomers[x].name.toString();
           userTotalSales.innerHTML =
allStatistics.orderedCustomers[x].totalsales.toString();
           count++
       }
```

```
        //Settigns to create the bar chart using the sales and products information
above
        var barchart = document.getElementById('chartproducts').getContext('2d');
        var myChart = new Chart(barchart, {
            type: 'bar',
            data: {
                labels: productNames,
                datasets: [{
                    label: 'Total Sales',
                    data: productSales,
                    backgroundColor: [
                        'rgba(255, 99, 132, 0.2)',
                        'rgba(54, 162, 235, 0.2)',
                        'rgba(255, 206, 86, 0.2)',
                        'rgba(75, 192, 192, 0.2)',
                        'rgba(153, 102, 255, 0.2)',
                        'rgba(255, 159, 64, 0.2)'
                    ],
                    borderColor: [
                        'rgba(255, 99, 132, 1)',
                        'rgba(54, 162, 235, 1)',
                        'rgba(255, 206, 86, 1)',
                        'rgba(75, 192, 192, 1)',
                        'rgba(153, 102, 255, 1)',
                        'rgba(255, 159, 64, 1)'
                    ],
                    borderWidth: 1
                }]
            },
            options: {
                title: {
                    display: true,
                    text: 'Most Popular Products'
                },
                scales: {
                    y: {
                        beginAtZero: true
                    }
                },
                responsive: true,
                maintainAspectRatio: false

            }
        });
        //Get the HTML element of a specific chart
        var revenue = document.getElementById('chartrevenue').getContext('2d');
        revenueChart = new Chart(revenue, {
            //Specify the type of chart
            type: 'line',
            //Pass the arrays into the data object
```

```
        data: {
            //Date array (x-axis)
            labels: allStatistics.date,
            //Revenue array (y-axis)
            datasets: [{
                label: "Revenue",
                backgroundColor: 'rgb(65, 240, 225)',
                borderColor: 'rgb(255, 99, 132)',
                //Pass the revenue array into the data section
                data: allStatistics.revenue,
            }]
        },
        options: {
            title: {
                display: true,
                text: 'Revenue chart (30 days)'
            },
            scales: {
                yAxes: [{
                    scaleLabel: {
                        display: true,
                        labelString: 'Revenue'
                    }
                }],
                xAxes: [{
                    scaleLabel: {
                        display: true,
                        labelString: 'Date'
                    }
                }]
            }
        }
    });
    var sales = document.getElementById('chartsales').getContext('2d');
    salesChart = new Chart(sales, {
        // The type of chart we want to create
        type: 'line',

        // The data for our dataset
        data: {
            labels: allStatistics.date,
            datasets: [{
                label: "Sales",
                backgroundColor: 'rgb(240, 198, 85)',
                borderColor: 'rgb(255, 99, 132)',
                data: allStatistics.sales,
            }]
        },

        // Configuration options go here
```

```
            options: {
                title: {
                    display: true,
                    text: 'Sales chart (30 days)'
                },
                scales: {
                    yAxes: [{
                        scaleLabel: {
                            display: true,
                            labelString: 'Sales'
                        }
                    }],
                    xAxes: [{
                        scaleLabel: {
                            display: true,
                            labelString: 'Date'
                        }
                    }]
                }
            }
        });
    });

}

//Clicked is initalized as 1 as the page automatically displays the first option
(30 days)
var clicked = 1
//A button calls this function when clicked
function changeRevenue(x) {
    //Clicked = 0 indicates going back to the first option (30 days) --> use the
original file
    if (clicked == 0) {
        revenueChart.data.labels = allStatistics.date
        revenueChart.options.title.text = "Revenue chart (30 days)"
        revenueChart.data.datasets[0].data = allStatistics.revenue
        //.update() changes the dataset
        revenueChart.update()
        //Add clicked so that the next button click moves to the second option
        clicked++
    }
    //clicked = 1 indicates that the user chooses the 15 days options and the array
is selected between the index 14-29.
    else if (clicked == 1) {
        //Uses a temporary data so that it doens't alter the original value
        var temporaryArray = allStatistics.revenue
        var temporaryDateArray = allStatistics.date
        revenueChart.data.labels = temporaryDateArray.slice(14, 29)
        revenueChart.options.title.text = "Revenue chart (15 days)"
        //Get the 15 most recent data (days)
```

```
        revenueChart.data.datasets[0].data = temporaryArray.slice(14, 29)
        revenueChart.update()
        //add click so that if the user clicks the zoom button again, the next
option is selected for the graph
        clicked++;
    }
    else if (clicked == 2) {
        var temporaryArray = allStatistics.revenue
        var temporaryDateArray = allStatistics.date
        revenueChart.data.labels = temporaryDateArray.slice(22, 29)
        revenueChart.options.title.text = "Revenue chart (7 days)"
        revenueChart.data.datasets[0].data = temporaryArray.slice(22, 29)
        revenueChart.update()
        clicked = 0;
    }
}


//Similar approach applies to the previous graph except it is changing sales graph
rather than revenue
function changeSales(x) {
    if (clicked == 0) {
        salesChart.data.labels = allStatistics.date
        salesChart.options.title.text = "Sales chart (30 days)"
        salesChart.data.datasets[0].data = allStatistics.sales
        salesChart.update()
        clicked++
    }
    else if (clicked == 1) {
        var temporaryArray = allStatistics.sales
        var temporaryDateArray = allStatistics.date

        salesChart.options.title.text = "Sales chart (15 days)"
        salesChart.data.labels = temporaryDateArray.slice(14, 29)

        salesChart.data.datasets[0].data = temporaryArray.slice(14, 29)
        salesChart.update()
        clicked++;
    }
    else if (clicked == 2) {
        var temporaryArray = allStatistics.sales
        var temporaryDateArray = allStatistics.date
        salesChart.data.labels = temporaryDateArray.slice(22, 29)
        salesChart.options.title.text = "Sales chart (7 days)"
        salesChart.data.datasets[0].data = temporaryArray.slice(22, 29)
        salesChart.update()
        clicked = 0;
    }
}
```

Admin page that displays the catalogue

cataloguea.html

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Catalogue</title>
   <link rel="stylesheet" href="/css/cataloguea.css">
   <link rel="stylesheet" href="/css/admin.css">
   <script src="/js/cataloguea.js"></script>
   <script src="/js/admin.js"></script>
   <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>

</head>

<body>
   <div id="header">
       <img id="logo"
src="/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
       <i class='bx bx-menu' onclick="openSidebar();" id="iconmenu"></i>
       <div class = "createbutton" onclick="openAddMenu()"> Add product </div>

   </div>

   <div class="sidebar">

       <ul id="dashboardlist" style="list-style-type:none">
           <li class="lists" onclick="mainmenu();">
               <i class='bx bxs-dashboard' ></i> Dashboard
           </li>
           <li class="lists" onclick="orders();">
               <i class='bx bx-cart'></i>
               Orders
           </li>
           <li class="lists" onclick="users();">
               <i class='bx bx-user'></i> Users

           </li>
           <li class="lists" onclick="transactions();">
               <i class='bx bx-wallet-alt'></i> Transactions

           <li class="lists" onclick="catalogue();">
```

```
                <i class='bx bx-package'></i> Catalogue
            </li>
            <li class="lists" onclick="logOut();">
                <i class='bx bx-log-out'></i> Logout
            </li>
        </ul>
        <div class="profile">
            profile
        </div>
    </div>
    <div class="maincontent">

        <table id="maintable">
            <tr id="tableheader">
              <th id="first">
                Product ID
              </th>
              <th id="column">
                Name
            </th>
              <th id="column">
                  Total Sales
              </th>
              <th id="column">
                Category
              </th>

              <th id="last">
              </th>
            </tr>
          </table>
    </div>
    <div class="create-form">
        <div class="actual-form-create">
            <div class="create-form-header">
                Adding Product
            </div>
            <ul style="list-style-type:none;" class="groupofforms">
                <input type="text" class="form" id="inputName" placeholder = "Enter
product name" ><br>
                <input type="text" class="form" id="inputCategory" placeholder =
"Enter category"  ><br>
                <input type="text" class="form" id="inputDescription" placeholder =
"Enter the item description"  ><br>
                <!--<p>Upload Image</p><input type="file"
id="myFile"accept="image/*"  name="profile-files" multiple="multiple">-->


                <div class="confirmbutton" onclick="addProduct()"> Confirm</div>
            </ul>
```

```
            <div class="cancelbutton" onclick="remove()"> Cancel</div>
        </div>
    </div>
    <div class="success">
        <div class="actual-success">
            <ul style="list-style-type:none;" class="groupofforms">
                <i class='bx bx-check-circle' id = "cbutton"></i>
                <h3 id="senttext-success">Success! </h3>
                <p>You have created a new product.</p>
            </ul>
        </div>
    </div>
</body>

</html>
```

cataloguea.js

```
window.onload = function () {
    loadCatalogue()
};

function loadCatalogue() {
    var xhr = new XMLHttpRequest();
    xhr.open('get', 'http://localhost:5001/catalogue/all');
    xhr.send();
    xhr.onload = function () {
        allProducts = JSON.parse(xhr.response);
        var table = document.getElementById('maintable')
        var count = 1;
        for (product in allProducts.product ) {
            //once the catalogue has been loaded, display them into table format
            var row = table.insertRow(count);
            var productID = row.insertCell(0);
            var name = row.insertCell(1);
            var totalSales = row.insertCell(2)
            var category = row.insertCell(3);
            var button = row.insertCell(4);
            productID.innerHTML = allProducts.product[product].productID
            name.innerHTML = allProducts.product[product].name;
            totalSales.innerHTML = allProducts.product[product].totalSales;
            category.innerHTML = allProducts.product[product].category;
            button.innerHTML += '<div onclick ="view(this);" class="viewbutton">View
More</div>'
            count++
        }
    }
}

//open popup to add a new product
```

```
function openAddMenu(){
   var modal = document.querySelector('.create-form');
   modal.classList.add('create-form-active');
   let sidebar = document.querySelector(".sidebar")
   sidebar.classList.remove("active");
}

//Add a new product to the catalogue
function addProduct(){
   var xhr = new XMLHttpRequest();

   xhr.open("POST","http://localhost:5001/addproduct",true);
   xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');

  //var formdata = new FormData();
  var name = document.getElementById("inputName").value;
  var category = document.getElementById("inputCategory").value;
  var description = document.getElementById("inputDescription").value;

  //Collect the information into one object
  var newProductInformation = {
      name:name,
      category: category,
      description: description
  }

   var JSONformatdata = JSON.stringify(newProductInformation);
   //Send the new data into the backend for the new record to be added into the
database
   xhr.send(JSONformatdata);

   xhr.onreadystatechange = function(){
   if(xhr.readyState == 4 && xhr.status == 200)
   {
   }
   var modal = document.querySelector('.success');
   modal.classList.add('success-active');
   setTimeout(function () {
       window.location.replace('http://localhost:5001/catalogue/admin')
   }, 2000);
   }
}

function view(row) {
   var table = document.getElementById('maintable')
   var rowNumber = row.parentElement.parentElement.rowIndex
   var productID = table.rows[rowNumber].cells[0].innerHTML;
   window.location.replace('http://localhost:5001/catalogue/admin/' +
productID.toString())
```

```
}
```

Admin page to select a specific product to be edited

editcatalogue.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <script src="/js/editcatalogue.js"></script>
   <script src="/js/admin.js"></script>
   <link rel="stylesheet" href="/css/editcatalogue.css">
   <link rel="stylesheet" href="/css/admin.css">
   <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
   <title>Edit Cat</title>
</head>
<body>
   <div id="header">
       <img id="logo"
src="/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
       <i class='bx bx-menu' onclick="openSidebar();" id="iconmenu"></i>
       <div class="cornerbutton" onclick="saveChanges()">Confirm</div>
   </div>


   <div class="sidebar">

       <ul id="dashboardlist" style="list-style-type:none">
           <li class="lists" onclick="mainMenu();">
               <i class='bx bxs-dashboard'></i> Dashboard
           </li>
           <li class="lists" onclick="orders();">
               <i class='bx bx-cart'></i>
               Orders
           </li>
           <li class="lists" onclick="users();">
               <i class='bx bx-user'></i> Users

           </li>
           <li class="lists" onclick="transactions();">
               <i class='bx bx-wallet-alt'></i> Transactions

           <li class="lists" onclick="catalogue();">
               <i class='bx bx-package'></i> Catalogue
           </li>
           <li class="lists" onclick="logOut();">
               <i class='bx bx-log-out'></i> Logout
```

```
            </li>
        </ul>
        <div class="profile">
            profile
        </div>
    </div>
    <div class="editform">
        <div class="formheader">
            Edit Catalogue
        </div>
        <div class="formgrid">
            <div id="editinfo">
                <div class ="rows"id="prodID"> Product ID:<br> </div>
                <div class ="rows"id="totalSales">Total Sale:<br> </div>
                <div class ="rows"id="dateAdded">Date Added:<br> </div>
                <div class ="rows" id="dateModified">Date Modified:<br></div>
            </div>
            <div id="stats">
                <div id="description">
                    <ul id="desc" style="list-style-type:none">
                        <li >Description </li>
                    </ul>
                </div>
                <div id="name">
                    <ul id="nameul" style="list-style-type:none">
                        <li >Name </li>
                    </ul>
                </div>
                <div id="category">
                    <ul id="cat" style="list-style-type:none">
                        <li >Category </li>
                    </ul>
                </div>
            </div>

            <div id="imagesliders">

        <table id="varianttable">
            <tr id="tableheader">
              <th id="first">
                Variant ID
              </th>
              <th id="column">
                Model Name
            </th>
              <th id="column">
                  Quantity
              </th>
              <th id="column">
                Price
```

```
            </th>

            <th id="last">
              <div class="addmodel" onclick="addMenu()">Add</div>
            </th>
          </tr>
        </table>
          </div>
      </div>
   </div>

   <div class="image-container">
       <div class="actual-image-container" id="imageid">
           <div  id="image-carousel">
               <div class="imagebutton" id="leftbutton" onclick ="move(-1)" ><i
class='bx bxs-chevron-left' ></i></div>
               <div class="imagebutton" id="rightbutton" onclick ="move(1)"><i
class='bx bxs-chevron-right' ></i></div>
           </div>
           <div id="returnbutton" onclick="closeContainer()">Return</div>
           <input type="file" id="imagefile"accept="image/*"  name="profile-files"
>

       </div>
   </div>

   <div class="delete">
       <div class="actual-success">
           <ul style="list-style-type:none;" class="groupofforms">
               <i class='bx bx-check-circle' id = "cbutton"></i>
               <h3 id="senttext-success">Deleted</h3>
               <p>You have deleted a product.</p>
           </ul>
       </div>
   </div>
   <div class="editproduct">
       <div class="actual-success">
           <ul style="list-style-type:none;" class="groupofforms">
               <i class='bx bx-check-circle' id = "cbutton"></i>
               <h3 id="senttext-success">Edited</h3>
               <p>You have edited a product.</p>
           </ul>
       </div>
   </div>
   <div class="deletemodel">
       <div class="actual-success">
           <ul style="list-style-type:none;" class="groupofforms">
               <i class='bx bx-check-circle' id = "cbutton"></i>
               <h3 id="senttext-success">Deleted</h3>
               <p>You have deleted a model.</p>
```

```
            </ul>
        </div>
    </div>
    <div class="addedmodel">
        <div class="actual-success">
            <ul style="list-style-type:none;" class="groupofforms">
                <i class='bx bx-check-circle' id = "cbutton"></i>
                <h3 id="senttext-success">Success! </h3>
                <p>You have created a new model.</p>
            </ul>
        </div>
    </div>
    <div class="create-form">
        <div class="actual-form-create">
            <div class="create-form-header">
                Adding Model
            </div>
            <ul style="list-style-type:none;" class="groupofforms">
                <input type="text" class="form" id="inputmodel" placeholder = "Enter
model type" ><br>
                <input type="number" class="form" id="inputprice" placeholder =
"Enter price"  ><br>
                <input type="number" class="form" id="inputstock" placeholder =
"Enter initial stock"  ><br>
                <p>Upload Image</p><input type="file" id="myFile"accept="image/*"
name="profile-files" multiple="multiple">

                <div class="confirmbutton" onclick="addModel()"> Confirm</div>
            </ul>
            <div class="cancelbutton" onclick="remove()"> Cancel</div>
        </div>
    </div>
</body>
</html>
```

editcatalogue.js

```
window.onload = function () {
    loadCatalogue();
};

var allProductAndModelInformation

function loadCatalogue() {
    var currentLocation = window.location.pathname;
    productID = currentLocation.replace("/catalogue/admin/", "");
    createXhrRequest("GET", 'http://localhost:5001/getallcatinfo/' + productID,
function (err, response) {
        if (err) { console.log("Error!"); }
        allProductInformation = JSON.parse(response);
```

```
        allProductAndModelInformation =allProductInformation
        //Inserting all the specific product information into either forms or static
information as it cant be edited
        document.getElementById("prodID").innerHTML +=
allProductInformation.product[0].productID
        document.getElementById("totalSales").innerHTML +=
allProductInformation.product[0].totalSales
        document.getElementById("dateAdded").innerHTML +=
allProductInformation.product[0].dateAdded.split('T')[0]
        document.getElementById("dateModified").innerHTML +=
allProductInformation.product[0].dateModified.split('T')[0]
        document.getElementById("desc").innerHTML += '<li><input class ="inputform"
id = "descriptionform" type="text"  value="' +
allProductInformation.product[0].description + '" </li>'
        document.getElementById("nameul").innerHTML += '<li><input class
="inputform" id = "newnameform" type="text"  value="' +
allProductInformation.product[0].name + '" </li>'
        document.getElementById("cat").innerHTML += '<li><input class ="inputform"
id = "categoryform" type="text"  value="' +
allProductInformation.product[0].category + '" </li>'

        var table = document.getElementById('varianttable')
        var count = 1;

        //Loop that generates a table filled with all the variants of a specific
product
        for (var x = 0; x < allProductInformation.variantInformation.length; x++) {
            var row = table.insertRow(count);
            var variantID = row.insertCell(0);
            var modelName = row.insertCell(1);
            var quantity = row.insertCell(2);
            var price = row.insertCell(3);
            var button = row.insertCell(4);
            variantID.innerHTML =
allProductInformation.variantInformation[x].variant.idvariant.toString();
            modelName.innerHTML = '<input class ="inputform" type="text"  value= "'
+ allProductInformation.variantInformation[x].variant.variantName + '" >';
            quantity.innerHTML = '<input class ="inputform" type="text"  value= "' +
allProductInformation.variantInformation[x].variant.quantity + '" >';
            price.innerHTML = '<input class ="inputform" type="text"  value= "' +
allProductInformation.variantInformation[x].variant.price + '" >';
            button.innerHTML += '<div onclick ="view(this);" class="viewimage">View
Image</div>  <div onclick ="confirmChange(this);" class="editthis">Confirm</div>'
            count++
        }

    });

}
```

```
//when the admin confirms the change fo information - this function is triggered
function confirmChange(row) {
   var table = document.getElementById('varianttable')
   var rowNumber = row.parentElement.parentElement.rowIndex
   var variantID = table.rows[rowNumber].cells[0].innerHTML;
   var xhr = new XMLHttpRequest();
   xhr.open("PUT", "http://localhost:5001/editModel/" + variantID, true);
   xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
   xhr.onload = function () {
       if (xhr.readyState == 4 && xhr.status == "200") {

       } else {
       }
   }
   var updatedModel = {};

   updatedModel.name = table.rows[rowNumber].cells[1].firstChild.value;
   updatedModel.quantity =
parseInt(table.rows[rowNumber].cells[2].firstChild.value);
   updatedModel.price = parseInt(table.rows[rowNumber].cells[3].firstChild.value);

   var JSONformatdata = JSON.stringify(updatedModel);
   xhr.send(JSONformatdata);
   window.location.reload();
}

function addMenu() {
   var modal = document.querySelector('.create-form');
   modal.classList.add('create-form-active');
   let sidebar = document.querySelector(".sidebar")
   sidebar.classList.remove("active");
}

//https://stackoverflow.com/questions/5587973/javascript-upload-file
//Add a new model or variant to the product
function addModel() {
   var xhr = new XMLHttpRequest();
   var formData = new FormData()
   var currentLocation = window.location.pathname;
   productID = currentLocation.replace("/catalogue/admin/", "");
   var modelName = document.getElementById("inputmodel").value;
   var price = document.getElementById("inputprice").value;
   var quantity = document.getElementById("inputstock").value;
   if (modelName == "" || price == "" || quantity == "") {
       alert("Can't be empty");
   } else {
       xhr.open("POST", "http://localhost:5001/addmodel/" + productID, true);
       var ins = document.getElementById('myFile').files.length
       for (var x = 0; x < ins; x++) {
```

```
            formData.append("uploaded_file",
document.getElementById('myFile').files[x]);
        }
        var newModel = {
            name: modelName,
            price: price,
            quantity: quantity
        }
        formData.append("newModel", JSON.stringify(newModel));
        xhr.send(formData);

        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
            }
        }
        var modal = document.querySelector('.addedmodel');
        modal.classList.add('addedmodel-active');
        setTimeout(function () {
            window.location.reload();
        }, 2000);
    }


}

//Save changes for the product information if edited
function saveChanges() {
    var newcat = {};
    newcat.description = document.getElementById("descriptionform").value;
    newcat.name = document.getElementById("newnameform").value;
    newcat.category = document.getElementById("categoryform").value;
    var currentLocation = window.location.pathname;
    productID = currentLocation.replace("/catalogue/admin/", "");
    var xhr = new XMLHttpRequest();
    xhr.open("PUT", "http://localhost:5001/editCatalogue/" + productID, true);
    xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');

    var JSONformatdata = JSON.stringify(newcat);

    xhr.onload = function () {
        var users = JSON.parse(xhr.responseText);
        if (xhr.readyState == 4 && xhr.status == "200") {
        } else {
            console.error(users);
        }
    }
    xhr.send(JSONformatdata);
    var modal = document.querySelector('.editproduct');
    modal.classList.add('success-active');
    setTimeout(function () {
```

```
        window.location.reload();
    }, 2000);
}

var imageList
var count
var numlist

//View the images of a particular variant
function view(row) {
    var table = document.getElementById('varianttable')
    var rowNumber = row.parentElement.parentElement.rowIndex
    var variantID = table.rows[rowNumber].cells[0].innerHTML;
    var allImage = []
    for (info in allProductAndModelInformation.variantInformation) {
        if (allProductAndModelInformation.variantInformation[info].variant.idvariant
=== parseInt(variantID)) {
            for (image in
allProductAndModelInformation.variantInformation[info].imageInformation ){

allImage.push(allProductAndModelInformation.variantInformation[info].imageInformati
on[image].imageaddress)
                numlist = info
            }
        }
        var modal = document.querySelector('.image-container');
        modal.classList.add('image-container-active');
    }
    document.getElementById('imageid').innerHTML += '<div id="changeimage"
onclick="editimage('+allProductAndModelInformation.variantInformation[numlist].imag
eInformation[0].idimages+')">Edit Image</div>'
    document.getElementById('image-carousel').innerHTML +=  '<img id="images"
src="'+ allImage[0]+'"> '
    count =0
    imageList = allImage
}

function move(num){
    //Triggers when the user clicks the "next" button to view the next images
    var child = document.getElementById("images");
    var file = document.getElementById("changeimage");
    var parent = document.getElementById("image-carousel");
    var parentID = document.getElementById("imageid");
    parent.removeChild(child);
    parentID.removeChild(file);
    //Count is added to keep track of the image that is being viewed
    count = count + num
    if (count>imageList.length-1){
        count=0;
    }
```

```
    else if(count<0){
        count = imageList.length-1
    }
    document.getElementById('imageid').innerHTML += '<div id="changeimage"
onclick="editimage('+allProductAndModelInformation.variantInformation[numlist].imag
eInformation[count].idimages+')">Edit Image</div>'
    document.getElementById('image-carousel').innerHTML +=  '<img id="images"
src="'+ imageList[count]+'"> '
}

function closeContainer() {
    var child = document.getElementById("images");
    var file = document.getElementById("changeimage");
    var parent = document.getElementById("image-carousel");
    var parentID = document.getElementById("imageid");
    parent.removeChild(child);
    parentID.removeChild(file);
    var modal = document.querySelector('.image-container');
    modal.classList.remove('image-container-active');
}

//Source:
https://stackoverflow.com/questions/48859546/upload-image-to-server-with-xmlhttpreq
uest-and-formdata-in-react-native
function editimage(imageID){
    var xhr = new XMLHttpRequest();

    var formData = new FormData()

    if (document.getElementById('imagefile').files[0]==undefined){
        alert("You need to upload an image")
    }else{
        xhr.open("POST", "http://localhost:5001/updateimage/" + imageID, true);

        formData.append("uploaded_file",
document.getElementById('imagefile').files[0]);

        xhr.send(formData);

        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
            }

        }

        setTimeout(function () {
            window.location.reload();
        }, 2000);
    }
```

```
}
```

Admin page that displays all the users

usersa.html

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Users</title>
   <script src="/js/usersa.js"></script>
   <script src="/js/admin.js"></script>
   <link rel="stylesheet" href="/css/usersa.css">
   <link rel="stylesheet" href="/css/admin.css">
   <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
</head>

<body>
   <div id="header">
       <img id="logo"
src="/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
       <i class='bx bx-menu' onclick="openSidebar();" id="iconmenu"></i>
       <div class = "createbutton" onclick="openCreate()"> Create User</div>
   </div>



   <div class="sidebar">

       <ul id="dashboardlist" style="list-style-type:none">
           <li class="lists" onclick="mainMenu();">
               <i class='bx bxs-dashboard'></i> Dashboard
           </li>
           <li class="lists" onclick="orders();">
               <i class='bx bx-cart'></i>
               Orders
           </li>
           <li class="pageactive" >
               <i class='bx bx-user'></i> Users

           </li>
           <li class="lists" onclick="transactions();">
               <i class='bx bx-wallet-alt'></i> Transactions

           <li class="lists" onclick="catalogue();">
```

```
                <i class='bx bx-package'></i> Catalogue
            </li>
            <li class="lists" onclick="logOut();">
                <i class='bx bx-log-out'></i> Logout
            </li>
        </ul>
        <div class="profile">
            profile
        </div>
    </div>

    <div class="maincontent">
        <div class="wrap">

            <div class="search">
               <input onkeyup="filterListByNameOrEmail()" type="text"
id="searchbar"class="searchTerm" placeholder="Search user by email or name">
                <button type="submit" class="searchButton">
                 <i class='bx bx-search'></i>
               </button>
            </div>
         </div>
         <div class="filtermenu">
            <div class="allfilters"> Status <select onchange="filterListByStatus()"
id="status">
                <option value=""> </option>
                <option value="USER">USER</option>
                <option value="ADMIN">ADMIN</option>
                <option value="LOGISTICS">LOGISTICS</option>
              </select>
              Total Sales <select onchange="sortBySales()" id="amount">
                <option value=""> </option>
                <option value="Ascending">Ascending </option>
                <option value="Descending">Descending</option>
              </select>

            </div>
        </div>
         <div class="userform">

         </div>
         <div class="tablecontainer">
            <table id="maintable">
                <tr id="tableheader">
                  <th id="first">
                    User ID
                  </th>
                  <th id="column">
                      Status
                  </th>
```

```
                <th id="column">
                  Name
                </th>
                <th id="column">
                  Total Sales
                </th>
                <th id="column">
                  Email
                </th>
                <th id="last">
                </th>
              </tr>
            </table>
        </div>


            <div class="create-form">
              <div class="actual-form-create">
                  <div class="create-form-header">
                      Create User
                  </div>
                  <ul style="list-style-type:none;" class="groupofforms">
                      <input type="text" class="form" id="inputname" placeholder =
"Enter fullname" ><br>
                      <input type="text" class="form" id="inputemail" placeholder
= "Enter email"  ><br>
                      <input type="text" class="form" id="inputphone" placeholder
= "Enter phone number"  ><br>
                      <input type="text" class="form" id="inputaddress"
placeholder = "Enter address"  ><br>
                      <input type="text" class="form"
id="inputstorename"placeholder = "Enter store name" ><br>
                      <select  class="form" id="inputstatus">
                          <option value="USER">User</option>
                          <option value="ADMIN">Admin</option>
                        </select>
                      <div class="confirmbutton" onclick="confirm()">
Confirm</div>
                  </ul>
                  <div class="cancelbutton" onclick="remove()"> Cancel</div>
              </div>
          </div>

          <div class="success">
              <div class="actual-success">
                  <ul style="list-style-type:none;" class="groupofforms">
                      <i class='bx bx-check-circle' id = "cbutton"></i>
                      <h3 id="senttext-success">Success! </h3>
                      <p>You have created a new user.</p>
                  </ul>
```

```
            </div>
        </div>

    </div>

</body>

</html>
```

usersa.js

```
window.onload = function () {
    loadAllUsers();
};



function loadAllUsers() {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', 'http://localhost:5001/users');
    xhr.send();
    xhr.onload = function () {

        allUsersInformation = JSON.parse(xhr.response);
        var table = document.getElementById('maintable')
        var count = 1;
        for (user in allUsersInformation) {
            var row = table.insertRow(count);
            var userID = row.insertCell(0);
            var status = row.insertCell(1);
            var name = row.insertCell(2);
            var sales = row.insertCell(3);
            var email = row.insertCell(4);
            var button = row.insertCell(5);
            userID.innerHTML = allUsersInformation[user].userID.toString();
            status.innerHTML = allUsersInformation[user].status;
            sales.innerHTML = allUsersInformation[user].sales;
            name.innerHTML = allUsersInformation[user].name;
            email.innerHTML = allUsersInformation[user].email;
            button.innerHTML += '<div onclick ="view(this);" class="viewbutton">View
More</div>'
            count++
        }
    }
}

function view(row) {
    var table = document.getElementById('maintable')
    var rowNumber = row.parentElement.parentElement.rowIndex
    var userID = table.rows[rowNumber].cells[0].innerHTML;
```

```
    window.location.replace('http://localhost:5001/usersdash/admin/' +
userID.toString())
}


function openCreate() {
    var modal = document.querySelector('.create-form');
    modal.classList.add('create-form-active');
    let sidebar = document.querySelector(".sidebar")
    sidebar.classList.remove("active");
}

//Confirm to add a new user to the database
function confirm() {
    var newName = document.getElementById("inputname").value;
    var newEmail = document.getElementById("inputemail").value;
    var newPhone = document.getElementById("inputphone").value;
    var newAddress = document.getElementById("inputaddress").value;
    var newStoreName = document.getElementById("inputstorename").value;
    var newstatus = document.getElementById("inputstatus").value;
    //https://ricardometring.com/getting-the-value-of-a-select-in-javascript
    if (newName == "" || newEmail == "" || newPhone == "" | newAddress == "" |
newStoreName == "") {
        alert("Can't be empty");
    } else {

        var xhr = new XMLHttpRequest();
        xhr.open("POST", "http://localhost:5001/createUsers/", true);
        xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
        xhr.onload = function () {
            var users = JSON.parse(xhr.responseText);
            if (xhr.readyState == 4 && xhr.status == "200") {

            } else {
                console.error(users);
            }
        }
        var newUser = {};
        newUser.email = newEmail;
        newUser.storename = newStoreName;
        newUser.phonenum = newPhone;
        newUser.status = newstatus;
        newUser.address = newAddress;
        newUser.name = newName;

        var JSONformatdata = JSON.stringify(newUser);
        xhr.send(JSONformatdata);
        var modal = document.querySelector('.success');
        modal.classList.add('success-active');
        setTimeout(function () {
```

```
            window.location.replace('http://localhost:5001/usersdash/admin')
        }, 2000);
    }

}

//Filters the list by name or email inputted
function filterListByNameOrEmail(){
    //Front end search bar source: //searchbar: https://codepen.io/huange/pen/rbqsD
    var table = document.getElementById('maintable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length-1

    for(var x =1;x<=tableLength;x++){

if(tr[x].getElementsByTagName("td")[2].innerHTML.toLowerCase().includes(document.ge
tElementById('searchbar').value.toLowerCase()) ||
tr[x].getElementsByTagName("td")[4].innerHTML.toLowerCase().includes(document.getEl
ementById('searchbar').value.toLowerCase())){
            table.getElementsByTagName("tr")[x].style.display =""
        }
        else{
            table.getElementsByTagName("tr")[x].style.display ="none"

        }
    }

}

//Filters the user list by their status
function filterListByStatus(){
    var table = document.getElementById('maintable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length-1

    for(var x =1;x<=tableLength;x++){
        if(document.getElementById('status').value==""){
            table.getElementsByTagName("tr")[x].style.display =""
        }else{

if(tr[x].getElementsByTagName("td")[1].innerHTML.toString()==document.getElementByI
d('status').value.toString()){
                table.getElementsByTagName("tr")[x].style.display =""
            }
            else{
                table.getElementsByTagName("tr")[x].style.display ="none"

            }
        }
```

```
    }
}

//Filter the user list by sales - admin and logistics is assumed to be 0
function sortBySales(){
    var table = document.getElementById('maintable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length-1

    for(var x =1;x<=tableLength;x++){
        for(var y =1;y<=tableLength-1;y++){
            if(document.getElementById('amount').value=="Descending"){

if(parseInt(tr[y].getElementsByTagName("td")[3].innerHTML)>parseInt(tr[y+1].getElem
entsByTagName("td")[3].innerHTML)){
                    tr[y].parentNode.insertBefore(tr[y+1],tr[y])
                }
            }
            else if(document.getElementById('amount').value=="Ascending"){

if(parseInt(tr[y].getElementsByTagName("td")[3].innerHTML)<parseInt(tr[y+1].getElem
entsByTagName("td")[3].innerHTML)){
                    tr[y].parentNode.insertBefore(tr[y+1],tr[y])
                }
            }

        }
    }
}
```

Admin page to edit a specific user

edituser.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit User</title>
    <script src="/js/edituser.js"></script>
    <script src="/js/admin.js"></script>
    <link rel="stylesheet" href="/css/edituser.css">
    <link rel="stylesheet" href="/css/admin.css">
    <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
</head>
```

```
<body>
    <div id="header">
        <img id="logo"
src="/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
        <i class='bx bx-menu' onclick="openSidebar();" id="iconmenu"></i>
        <div class="cornerbutton" onclick="updateUser()">Confirm</div>
        <div class="deletebutton" onclick="deleteUser()">Delete</div>
    </div>


    <div class="sidebar">

        <ul id="dashboardlist" style="list-style-type:none">
            <li class="lists">
                <i class='bx bxs-dashboard'></i> Dashboard
            </li>
            <li class="lists" onclick="orders();">
                <i class='bx bx-cart'></i>
                Orders
            </li>
            <li class="lists" onclick="users();">
                <i class='bx bx-user'></i> Users

            </li>
            <li class="lists" onclick="transactions();">
                <i class='bx bx-wallet-alt'></i> Transactions

            <li class="lists" onclick="catalogue();">
                <i class='bx bx-package'></i> Catalogue
            </li>
            <li class="lists" onclick="logOut();">
                <i class='bx bx-log-out'></i> Logout
            </li>
        </ul>
        <div class="profile">
            profile
        </div>
    </div>

    <div class="editform">
        <div class="formheader">
            <h3>Edit User</h3>
        </div>
        <div class="formgrid">
            <div id="profilepic"></div>
            <div id="idinfo"></div>
            <div id="details">
                <div id="status">
                        <ul id ="statusul" style="list-style-type:none">
                            <li class ="spacingbox"><b><p>Status</p></b> </li>
```

```
            </ul>

        </div>
        <div id="name">
            <ul id ="nameul" style="list-style-type:none">
                <li class ="spacingbox"><b><p >Name</p></b>  </li>
            </ul>

        </div>
        <div id="address">
            <ul id ="addressul" style="list-style-type:none">
                <li class ="spacingbox"><b><p >Address</p></b>  </li>
            </ul>

        </div>
        <div id="phonenum">
            <ul id ="phonenumul" style="list-style-type:none">
                <li class ="spacingbox"><b><p >Phone Number</p></b>  </li>
            </ul>

        </div>
        <div id="storename">
            <ul id ="storenameul" style="list-style-type:none">
                <li class ="spacingbox"><b><p >Store Name</p></b>  </li>
            </ul>
        </div>
        <div id="created">
            <ul id ="createdul" style="list-style-type:none">
                <li class ="spacingbox"><b><p >Created</p></b>  </li>
            </ul>
        </div>
    </div>
    <div id="activities">
        <table id="activitiestable">

            <tr id="activities-tableheader">
              <th id="first-order">
                Transaction ID
              </th>
              <th id="column">
                Status
            </th>
              <th id="column">
                Date Ordered
              </th>
              <th id="column">
                Total Value
              </th>

              <th id="last-order">
```

```
                    </th>
                </tr>
            </table>
        </div>
    </div>

</div>
<div class="success">
    <div class="actual-success">
        <ul style="list-style-type:none;" class="groupofforms">
            <i class='bx bx-check-circle' id = "cbutton"></i>
            <h3 id="senttext-success">Success!</h3>
            <p>You have edited a user.</p>
        </ul>
    </div>
</div>
<div class="delete">
    <div class="actual-success">
        <ul style="list-style-type:none;" class="groupofforms">
            <i class='bx bx-check-circle' id = "cbutton"></i>
            <h3 id="senttext-success">Deleted</h3>
            <p>You have deleted a user.</p>
        </ul>
    </div>
</div>
</body>

</html>
```

edituser.js

```
window.onload = function () {
    loadUserInformation();
    };


function loadUserInformation(){
    var currentLocation = window.location.pathname;
    userID = currentLocation.replace("/usersdash/admin/", "");
    createXhrRequest("GET", 'http://localhost:5001/users/'+userID, function (err,
response) {
        if (err) { console.log("Error!"); }
        allUserInformation = JSON.parse(response);
    });

    //Load necessary information into either a form as a placeholder or a static
text
    document.getElementById("profilepic").innerHTML= '<img class="image"
src='+allUserInformation.specificUser[0].profilepic+'>' +
allUserInformation.specificUser[0].name
```

```
   document.getElementById("idinfo").innerHTML='<p class="idinfotext"> <b>User ID:
</b>'+ allUserInformation.specificUser[0].userID+'</p>'+ '<p  class="idinfotext">
<b> Email:</b> '+ allUserInformation.specificUser[0].email+'</p>'
   document.getElementById("statusul").innerHTML+=
'<li><p>'+allUserInformation.specificUser[0].status+'</p></li>'
   document.getElementById("nameul").innerHTML+= '<li><input type="text"
class="allform" id="form1" value="'+allUserInformation.specificUser[0].name+'"
</li>'
   document.getElementById("addressul").innerHTML+= '<li><input type="text"
class="allform" id="form2" value="'+allUserInformation.specificUser[0].address+'"
</li>'
   document.getElementById("phonenumul").innerHTML+= '<li><input type="text"
class="allform" id="form3" value="'+allUserInformation.specificUser[0].phonenum+'"
</li>'
   document.getElementById("storenameul").innerHTML+= '<li><input type="text"
class="allform" id="form4" value="'+allUserInformation.specificUser[0].storename+'"
</li>'
   document.getElementById("createdul").innerHTML+= '<li><input type="text"
class="allform" id="form5"
value="'+allUserInformation.specificUser[0].created.split('T')[0].toString()+'"
</li>'

   var count =1;

   var table = document.getElementById('activitiestable')

   //Create a table that displays all the users past transaction and current order
records
   for (transaction in allUserInformation.allTransactions)  {
       var row = table.insertRow(count);
       var transactionID = row.insertCell(0);
       var status = row.insertCell(1);
       var dateOrdered = row.insertCell(2);
       var totalValue = row.insertCell(3);
       var button = row.insertCell(4);
       transactionID.innerHTML =
allUserInformation.allTransactions[transaction].idTransaction.toString();
       status.innerHTML =
allUserInformation.allTransactions[transaction].orderStatus.toString();
       dateOrdered.innerHTML =
allUserInformation.allTransactions[transaction].dateOrdered.split('T')[0].toString(
);
       userID.innerHTML =
allUserInformation.allTransactions[transaction].userID.toString();
       totalValue.innerHTML =
parseInt(allUserInformation.allTransactions[transaction].totalValue.toString());
       button.innerHTML += '<div onclick ="viewmore(this);" class="viewbutton">View
More</div>'
       count++
```

```
    }

}

function updateUser(){
    //Function to update user information
    var newUserInformation = {};
    newUserInformation.phonenum = document.getElementById("form3").value;
    newUserInformation.name = document.getElementById("form1").value;
    newUserInformation.address = document.getElementById("form2").value;
    newUserInformation.storename = document.getElementById("form4").value;
    newUserInformation.created = parseInt(document.getElementById("form5").value);

    var currentLocation = window.location.pathname;
    userID = currentLocation.replace("/usersdash/admin/", "");
    var JSONformatdata = JSON.stringify(newUserInformation);



    var xhr = new XMLHttpRequest();
    xhr.open("PUT", "http://localhost:5001/updateUsers/"+userID, true);
    xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
    xhr.onload = function () {
        var users = JSON.parse(xhr.responseText);
        if (xhr.readyState == 4 && xhr.status == "200") {
        } else {
            console.error(users);
        }
    }
    xhr.send(JSONformatdata);
    var modal = document.querySelector('.success');
    modal.classList.add('success-active');
    setTimeout(function() {
        window.location.reload()
    }, 2000);

}



function deleteUser(){
    var currentLocation = window.location.pathname;
    userID = currentLocation.replace("/usersdash/admin/", "");
    createXhrRequest("DELETE", 'http://localhost:5001/deleteUsers/'+userID, function
(err, response) {
        if (err) { console.log("Error!"); }
    });

    var modal = document.querySelector('.delete');
            modal.classList.add('success-active');
            setTimeout(function() {
                window.location.replace('http://localhost:5001/usersdash/admin')
```

```
            }, 2000);
}
```

Admin page to view ongoing orders and orders that are waiting for approval

ordersa.html

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Orders Management</title>
   <link rel="stylesheet" href="/css/ordersa.css">
   <link rel="stylesheet" href="/css/admin.css">
   <script src="/js/ordersa.js"></script>
   <script src="/js/admin.js"></script>
   <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
</head>

<body>
   <div id="header">
       <img id="logo"
src="/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
       <i class='bx bx-menu' onclick="openSidebar();" id="iconmenu"></i>

   </div>


   <div class="sidebar">

       <ul id="dashboardlist" style="list-style-type:none">
           <li class="lists" onclick="mainMenu();">
               <i class='bx bxs-dashboard'></i> Dashboard
           </li>
           <li class="lists" onclick="orders();">
               <i class='bx bx-cart'></i>
               Orders
           </li>
           <li class="lists" onclick="users();">
               <i class='bx bx-user'></i> Users

           </li>
           <li class="lists" onclick="transactions();">
               <i class='bx bx-wallet-alt'></i> Transactions

           <li class="lists" onclick="catalogue();">
```

```
            <i class='bx bx-package'></i> Catalogue
          </li>
          <li class="lists" onclick="logOut();">
            <i class='bx bx-log-out'></i> Logout
          </li>
      </ul>
      <div class="profile">
          profile
      </div>
   </div>
   <div class="maincontent">

       <div class="pending">
        <b>Pending</b>
        <div class="filtermenu">
          <div class="allfilters">
            Total Value <select onchange="sortByAmountPending()"
id="pendingamount">
              <option value=""> </option>
              <option value="Ascending">Ascending </option>
              <option value="Descending">Descending</option>
            </select>
            Date Ordered <select onchange="sortByDatePending()"id="pendingdate">
              <option value="1"> </option>
              <option value="Recent">Recent </option>
              <option value="Latest">Latest</option>
            </select>
            Filter by User ID
            <input onkeyup="filterOrdersByStatusPending()" id="pendinguserid"
type="number">
          </div>
      </div>
        <table id="pendingtable">

            <tr id="orders-tableheader">
              <th id="first-order">
                Transaction ID
              </th>
              <th id="column">
                Status
            </th>
              <th id="column">
                Date Ordered
              </th>
              <th id="column">
                User ID
              </th>
              <th id="column">
                Total Value
              </th>
```

```
                <th id="last-order">
                </th>
              </tr>
            </table>
      </div>

        <div class="pendingpayment">

         <b>Completed/On-going orders</b>
         <div class="filtermenu">
          <div class="allfilters"> Status <select onchange="filterOrdersByStatus()"
id="status">
              <option value=""> </option>
              <option value="Payment Uploaded">Payment Uploaded</option>
              <option value="Waiting Payment">Waiting Payment</option>
              <option value="Delivering">Delivering</option>
              <option value="Processing">Processing</option>
              <option value="Completed">Completed</option>
            </select>
            Total Value <select onchange="sortByAmount()" id="amount">
              <option value=""> </option>
              <option value="Ascending">Ascending </option>
              <option value="Descending">Descending</option>
            </select>
            Date Ordered <select onchange="sortByDate()"id="date">
              <option value="1"> </option>
              <option value="Recent">Recent </option>
              <option value="Latest">Latest</option>
            </select>
            Date Completed <select onchange="sortByDate()"id="date">
              <option value="1"> </option>
              <option value="Recent">Recent </option>
              <option value="Latest">Latest</option>
            </select>
            Filter by User ID
            <input onkeyup="filterListByUserId()" id="inputuserid" type="number">
          </div>
      </div>
            <table id="orderstable">
                <tr id="orders-tableheader">
                  <th id="first-order">
                    Transaction ID
                  </th>
                  <th id="column">
                    Status
                </th>
                  <th id="column">
                    Date Ordered
                  </th>
                  <th id="column">
```

```
                Date Completed
              </th>
              <th id="column">
                User ID
              </th>
              <th id="column">
                Total Value
              </th>

              <th id="last-order">
              </th>
            </tr>
          </table>
      </div>


  </div>

  <div class="modal-container">
    <div class="actualModal">
      <h1>Transaction Details</h1>
      <div class="transcdet">
        <div id="transcID">
        </div>
        <div id="ordered">
        </div>
        <div id="detstatus">
        </div>
      </div>
      <div class="tablecontainer">
        <table id="table-list">
            <tr id="tableheader">
              <th id="first">
                Product
              </th>
              <th id="column">
                Category
              </th>
              <th id="last">
                Quantity
              </th>

            </tr>
          </table>
      </div>

      <div id="retbutton" onclick="closeTransaction();"> Return</div>
    </div>
  </div>
</body>
```

```
</html>
```

ordersa.js

```
window.onload = function () {
   loadAllOrders();
};

var allTransactions

function loadAllOrders() {
   //Get all the orders within the system
   createXhrRequest("GET", 'http://localhost:5001/getallTransactions', function
(err, response) {
       if (err) { console.log("Error!"); }
       var transactionAll = JSON.parse(response);
       allTransactions = transactionAll
   });
   var transactionCounts = 1;
   var approvalCounts = 1;
   for (transaction in allTransactions) {
       //If the status of the order is neither of these, place it into the table
       if (allTransactions[transaction].transaction.orderStatus !== "Waiting
Approval" && allTransactions[transaction].transaction.orderStatus !== "Payment
Uploaded") {
           var table = document.getElementById('orderstable')
           var row = table.insertRow(transactionCounts);
           var transactionID = row.insertCell(0);
           var status = row.insertCell(1);
           var dateOrdered = row.insertCell(2);
           var dateCompleted = row.insertCell(3);
           var userID = row.insertCell(4);
           var totalValue = row.insertCell(5);
           var button = row.insertCell(6);
           transactionID.innerHTML =
allTransactions[transaction].transaction.idTransaction.toString();
           status.innerHTML =
allTransactions[transaction].transaction.orderStatus.toString();
           if (allTransactions[transaction].transaction.dateCompleted == null) {
               dateCompleted.innerHTML = ""
           }
           else {
               dateCompleted.innerHTML =
allTransactions[transaction].transaction.dateCompleted.toString();
           }
           userID.innerHTML =
allTransactions[transaction].transaction.userID.toString();
           totalValue.innerHTML =
parseInt(allTransactions[transaction].transaction.totalValue.toString());
```

```
            dateOrdered.innerHTML =
allTransactions[transaction].transaction.dateOrdered.split('T')[0];
            button.innerHTML = '<div onclick ="viewMore(this);"
class="viewbutton">View More</div>'
            transactionCounts++
        }
        //If the status of the order is "waiting for approval", add it to this table
        else if (allTransactions[transaction].transaction.orderStatus == "Waiting
Approval") {
            var table = document.getElementById('pendingtable')
            var row = table.insertRow(approvalCounts);
            var transactionID = row.insertCell(0);
            var status = row.insertCell(1);
            var dateOrdered = row.insertCell(2);
            var userID = row.insertCell(3);
            var totalValue = row.insertCell(4);
            var button = row.insertCell(5);

            transactionID.innerHTML =
allTransactions[transaction].transaction.idTransaction.toString();
            status.innerHTML =
allTransactions[transaction].transaction.orderStatus.toString();
            userID.innerHTML =
allTransactions[transaction].transaction.userID.toString();
            totalValue.innerHTML =
parseInt(allTransactions[transaction].transaction.totalValue.toString());
            dateOrdered.innerHTML =
allTransactions[transaction].transaction.dateOrdered.split('T')[0];
            button.innerHTML += '<div onclick ="viewMorePending(this);"
class="viewbutton">View More</div>'
            button.innerHTML += '<div onclick ="acceptOrder(this);"
class="acceptbutton">Accept</div>'

            approvalCounts++
        }
    }
}

function acceptOrder(row) {
    //View an order and accept ones that are waiting for approval
    var table = document.getElementById('pendingtable')
    var rowNumber = row.parentElement.parentElement.rowIndex
    var transactionID = table.rows[rowNumber].cells[0].innerHTML;
    for (transaction in allTransactions){
        //Cyles through the transactions to find the one that is being accepted
        if(allTransactions[transaction].transaction.idTransaction==transactionID){
            var allNewVariantInformation = []
            for (details in allTransactions[transaction].details ){
                //Update the stock quantity
```

```
                var newStock =
allTransactions[transaction].details[details].variant.quantity
-allTransactions[transaction].details[details].details.quantity
                var newVariant = {
                    variantID
:allTransactions[transaction].details[details].details.variantID,
                    newStock :newStock
                }
                allNewVariantInformation.push(newVariant)
            }
            break;
        }
    }
    //New stock indicates the new quantity once the poruct has been ordered
    var newStatus = {
        orderStatus: "Waiting Payment",
        newStock: allNewVariantInformation
    }
    var JSONformatdata = JSON.stringify(newStatus);
    var xhr = new XMLHttpRequest();
    xhr.open("PUT", "http://localhost:5001/updateStatus/" + transactionID, true);
    xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
        }
    }
    xhr.send(JSONformatdata);
    setTimeout(function () {
        window.location.reload()
    }, 4000);
}


function filterOrdersByStatus() {
    var table = document.getElementById('orderstable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length - 1
    for (var x = 1; x <= tableLength; x++) {
        if (document.getElementById('status').value == "") {
            table.getElementsByTagName("tr")[x].style.display = ""
        } else {
            if (tr[x].getElementsByTagName("td")[1].innerHTML.toString() ==
document.getElementById('status').value.toString()) {
                table.getElementsByTagName("tr")[x].style.display = ""
            }
            else {
                table.getElementsByTagName("tr")[x].style.display = "none"

            }
        }
```

```
    }
}

function sortByAmount() {
    var table = document.getElementById('orderstable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length - 1
    for (var x = 1; x <= tableLength; x++) {
        for (var y = 1; y <= tableLength - 1; y++) {
            if (document.getElementById('amount').value == "Descending") {
                if (parseInt(tr[y].getElementsByTagName("td")[5].innerHTML) <
parseInt(tr[y + 1].getElementsByTagName("td")[5].innerHTML)) {
                    tr[y].parentNode.insertBefore(tr[y + 1], tr[y])
                }
            }
            else if (document.getElementById('amount').value == "Ascending") {
                if (parseInt(tr[y].getElementsByTagName("td")[5].innerHTML) >
parseInt(tr[y + 1].getElementsByTagName("td")[5].innerHTML)) {
                    tr[y].parentNode.insertBefore(tr[y + 1], tr[y])
                }
            }

        }
    }
}

function sortByDate() {
    var table = document.getElementById('orderstable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length - 1
    for (var x = 1; x <= tableLength; x++) {
        for (var y = 1; y <= tableLength - 1; y++) {
            if (document.getElementById('date').value == "Recent") {
                if (tr[y].getElementsByTagName("td")[2].innerHTML < tr[y +
1].getElementsByTagName("td")[2].innerHTML) {
                    tr[y].parentNode.insertBefore(tr[y + 1], tr[y])
                }
            }
            else if (document.getElementById('date').value == "Latest") {
                if (tr[y].getElementsByTagName("td")[2].innerHTML > tr[y +
1].getElementsByTagName("td")[2].innerHTML) {
                    tr[y].parentNode.insertBefore(tr[y + 1], tr[y])
                }
            }
        }
    }
}

function filterListByUserId() {
```

```
    var userID = document.getElementById('inputuserid').value
    var table = document.getElementById('orderstable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length - 1

    for (var x = 1; x <= tableLength; x++) {
        if (userID == "") {
            table.getElementsByTagName("tr")[x].style.display = ""
        } else {
            if (tr[x].getElementsByTagName("td")[4].innerHTML.toString() == userID)
{
                table.getElementsByTagName("tr")[x].style.display = ""
            }
            else {
                table.getElementsByTagName("tr")[x].style.display = "none"
            }
        }
    }
}

function sortByAmountPending() {
    var table = document.getElementById('pendingtable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length - 1
    for (var x = 1; x <= tableLength; x++) {
        for (var y = 1; y <= tableLength - 1; y++) {
            if (document.getElementById('pendingamount').value == "Descending") {
                if (parseInt(tr[y].getElementsByTagName("td")[4].innerHTML) <
parseInt(tr[y + 1].getElementsByTagName("td")[4].innerHTML)) {
                    tr[y].parentNode.insertBefore(tr[y + 1], tr[y])
                }
            }
            else if (document.getElementById('pendingamount').value == "Ascending")
{
                if (parseInt(tr[y].getElementsByTagName("td")[4].innerHTML) >
parseInt(tr[y + 1].getElementsByTagName("td")[4].innerHTML)) {
                    tr[y].parentNode.insertBefore(tr[y + 1], tr[y])
                }
            }
        }
    }
}

function sortByDatePending() {
    var table = document.getElementById('pendingtable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length - 1

    for (var x = 1; x <= tableLength; x++) {
        for (var y = 1; y <= tableLength - 1; y++) {
```

```
            if (document.getElementById('pendingdate').value == "Recent") {
                if (tr[y].getElementsByTagName("td")[2].innerHTML < tr[y +
1].getElementsByTagName("td")[2].innerHTML) {
                    tr[y].parentNode.insertBefore(tr[y + 1], tr[y])
                }
            }
            else if (document.getElementById('pendingdate').value == "Latest") {
                if (tr[y].getElementsByTagName("td")[2].innerHTML > tr[y +
1].getElementsByTagName("td")[2].innerHTML) {
                    tr[y].parentNode.insertBefore(tr[y + 1], tr[y])
                }
            }

        }
    }
}

function filterOrdersByStatusPending() {
    var userid = document.getElementById('pendinguserid').value
    var table = document.getElementById('pendingtable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length - 1

    for (var x = 1; x <= tableLength; x++) {
        if (userid == "") {
            table.getElementsByTagName("tr")[x].style.display = ""
        } else {
            if (tr[x].getElementsByTagName("td")[3].innerHTML.toString() == userid)
{
                table.getElementsByTagName("tr")[x].style.display = ""
            }
            else {
                table.getElementsByTagName("tr")[x].style.display = "none"
            }
        }
    }
}

function viewMore(row){
    var table = document.getElementById('orderstable')
    var rowNumber = row.parentElement.parentElement.rowIndex
    var modal = document.querySelector('.modal-container');
    var transactionID = table.rows[rowNumber].cells[0].innerHTML;

    modal.classList.add('create-form-active');
    for (transaction in allTransactions){
        if(allTransactions[transaction].transaction.idTransaction==transactionID){
            document.getElementById('transcID').innerHTML= "Transaction ID:
"+allTransactions[transaction].transaction.idTransaction
```

```
            document.getElementById('ordered').innerHTML="Date Ordered:
"+allTransactions[transaction].transaction.dateOrdered.split('T')[0]
            document.getElementById('detstatus').innerHTML="Order Status:
"+allTransactions[transaction].transaction.orderStatus

            var allDetails = allTransactions[transaction].details
            var detailsCount =1;
            for (details in allDetails){
                var detailsTable = document.getElementById('table-list')
                var detailsTableRow = detailsTable.insertRow(detailsCount);
                var productName = detailsTableRow.insertCell(0);
                var category = detailsTableRow.insertCell(1);
                var quantity = detailsTableRow.insertCell(2);
                productName.innerHTML= allDetails[details].products.name +" - "
+allDetails[details].variant.variantName
                category.innerHTML =allDetails[details].products.category
                quantity.innerHTML =allDetails[details].details.quantity

                detailsCount++
            }
        }
    }

}

function closeTransaction(){
    var detailsTable = document.getElementById('table-list')
    var numberOfRows = detailsTable.rows.length;
    for (let x = numberOfRows - 1; x > 0; x--) {
        detailsTable.deleteRow(x)
    }
    var modal = document.querySelector('.modal-container');
    modal.classList.remove('create-form-active');

}

function viewMorePending(row){
    var table = document.getElementById('pendingtable')
    var rowNumber = row.parentElement.parentElement.rowIndex
    var modal = document.querySelector('.modal-container');
    var transactionID = table.rows[rowNumber].cells[0].innerHTML;

    modal.classList.add('create-form-active');
    for (transaction in allTransactions){
        if(allTransactions[transaction].transaction.idTransaction==transactionID){
            document.getElementById('transcID').innerHTML= "Transaction ID:
"+allTransactions[transaction].transaction.idTransaction
            document.getElementById('ordered').innerHTML="Date Ordered:
"+allTransactions[transaction].transaction.dateOrdered.split('T')[0]
```

```
            document.getElementById('detstatus').innerHTML="Order Status:
"+allTransactions[transaction].transaction.orderStatus

            var allDetails = allTransactions[transaction].details
            var detailsCount =1;
            for (details in allDetails){
                var detailsTable = document.getElementById('table-list')
                var detailsTableRow = detailsTable.insertRow(detailsCount);
                var productName = detailsTableRow.insertCell(0);
                var category = detailsTableRow.insertCell(1);
                var quantity = detailsTableRow.insertCell(2);
                productName.innerHTML= allDetails[details].products.name +" - "
+allDetails[details].variant.variantName
                category.innerHTML =allDetails[details].products.category
                quantity.innerHTML =allDetails[details].details.quantity
                detailsCount++
            }
        }
    }
}
```

Admin page to view transactions and payment proofs

transactionsa.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Transaction</title>
    <script src="/js/admin.js"></script>
    <script src="/js/transactionsa.js"></script>
    <link rel="stylesheet" href="/css/transactionsa.css">
    <link rel="stylesheet" href="/css/admin.css">
    <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>

</head>

<body>
    <div id="header">
    <img id="logo"
src="/images/WhatsApp_Image_2021-12-21_at_21.15.14-removebg-preview.png">
    <i class='bx bx-menu' onclick="openSidebar();" id="iconmenu"></i>


</div>
```

```
<div class="sidebar">
   <ul id="dashboardlist" style="list-style-type:none">
       <li class="lists" onclick="orders();">
           <i class='bx bxs-dashboard' ></i> Dashboard
       </li>
       <li class="lists" onclick="orders();">
           <i class='bx bx-cart'></i>
           Orders
       </li>
       <li class="lists" onclick="users();">
           <i class='bx bx-user'></i> Users

       </li>
       <li class="lists" onclick="transactions();">
           <i class='bx bx-wallet-alt'></i> Transactions

       <li class="lists" onclick="catalogue();">
           <i class='bx bx-package'></i> Catalogue
       </li>
       <li class="lists" onclick="logOut();">
           <i class='bx bx-log-out'></i> Logout
       </li>
   </ul>

</div>

   <div class="maincontent">

       <div class="pending">
           <div class="filtermenu">
             <div class="allfilters">
               Total Value <select onchange="sortByAmount()" id="amount">
                 <option value=""> </option>
                 <option value="Ascending">Ascending </option>
                 <option value="Descending">Descending</option>
               </select>

               Filter by User ID
               <input onkeyup="filterListByUserId()" id="userid" type="number">
             </div>
         </div>
           <table id="maintable">
               <tr id="tableheader">
                   <th id="first">
                   Transaction ID
               </th>
               <th id="column">
                 Status
             </th>
               <th id="column">
```

```
            Proof of Payment
          </th>
          <th id="column">
            User ID
          </th>
          <th id="column">
            Total Value
          </th>
          <th id="last">
          </th>
        </tr>
        </table>
    </div>
    <div class="create-form">
        <div class="actual-form-create">
            <div class="create-form-header">
                Proof of Payment
            </div>
            <div id="proofofpayement">

            </div>
            <div class="cancelbutton" onclick="removeAll()"> Cancel</div>

        </div>
    </div>
    <div class="modal-container">
        <div class="actualModal">
          <h1>Transaction Details</h1>
          <div class="transcdet">
            <div id="transcID">
            </div>
            <div id="ordered">
            </div>
            <div id="status">
            </div>
          </div>
          <div class="tablecontainer">
            <table id="table-list">
                <tr id="tableheader">
                  <th id="first">
                    Product
                  </th>
                  <th id="column">
                    Category
                  </th>
                  <th id="last">
                    Quantity
                  </th>

                </tr>
```

```
                </table>
            </div>

            <div id="retbutton" onclick="closeTransaction();"> Return</div>
        </div>
    </div>
</div>

</body>

</html>
```

transactionsa.js

```
window.onload = function () {
   loadallTransactions();
};

var allTransactions

function loadallTransactions() {
   createXhrRequest("GET", 'http://localhost:5001/getalltransactions', function
(err, response) {
       if (err) { console.log("Error!"); }
       var allTransactions = JSON.parse(response);
       allTransactions = allTransactions

   });
   var count = 1;
   for (transaction in allTransactions) {
       //Get all the transactions where the payment has been uploaded and needs to
be reviewed
       if (allTransactions[transaction].transaction.orderStatus == "Payment
Uploaded") {
           var table = document.getElementById('maintable')
           var row = table.insertRow(count);
           var transactionID = row.insertCell(0);
           var status = row.insertCell(1);
           var paymentProof = row.insertCell(2);
           var userID = row.insertCell(3);
           var totalValue = row.insertCell(4);
           var button = row.insertCell(5);

           transactionID.innerHTML =
allTransactions[transaction].transaction.idTransaction.toString();
           status.innerHTML =
allTransactions[transaction].transaction.orderStatus.toString();
           paymentProof.innerHTML = '<div onclick ="viewImage(this);"
class="viewimage">View</div>'
```

```
            userID.innerHTML =
allTransactions[transaction].transaction.userID.toString();
            totalValue.innerHTML =
parseInt(allTransactions[transaction].transaction.totalValue.toString());
            button.innerHTML += '<div onclick ="viewMore(this);"
class="viewbutton">View More</div>'
            button.innerHTML += '<div onclick ="accept(this);"
class="acceptbutton">Accept</div>'
            button.innerHTML += '<div onclick ="reject(this);"
class="rejectbutton">Reject</div>'

            count++
        }
    }
}

//View the payment proof uploaded in the form of an image
function viewImage(row) {
    var modal = document.querySelector('.create-form');
    modal.classList.add('create-form-active');
    let sidebar = document.querySelector(".sidebar")
    sidebar.classList.remove("active");
    var table = document.getElementById('maintable')
    var rowNumber = row.parentElement.parentElement.rowIndex
    var transactionID = table.rows[rowNumber].cells[0].innerHTML;
    for (transaction in allTransactions) {
        if (allTransactions[transaction].transaction.idTransaction == transactionID)
{
            document.getElementById('proofofpayement').innerHTML += '<img
class="allimages" src="' + allTransactions[transaction].transaction.paymentproof +
'">'
        }
    }
}

function removeAll() {
    var modal = document.querySelector('.create-form');
    modal.classList.remove('create-form-active');
    document.getElementById('proofofpayement').innerHTML = ''
}

function accept(row) {
    var table = document.getElementById('maintable')
    var rowNumber = row.parentElement.parentElement.rowIndex
    var transactionID = table.rows[rowNumber].cells[0].innerHTML;
    for (transaction in allTransactions){
        if (allTransactions[transaction].transaction.idTransaction ==
transactionID){
            var totalSales = allTransactions[transaction].transaction.totalUnits
            var totalRevenue = allTransactions[transaction].transaction.totalValue
```

```
        }
    }
    //Update the status if accepted
    var newStatus = {
        orderStatus: "Payment Approved",
        totalSales: totalSales,
        totalRevenue: totalRevenue
    }
    var JSONformatdata = JSON.stringify(newStatus);
    var xhr = new XMLHttpRequest();
    xhr.open("PUT", "http://localhost:5001/updateStatus/" + transactionID, true);
    xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
        }
    }
    xhr.send(JSONformatdata);
    window.location.reload()
}


function reject(row) {
    var table = document.getElementById('maintable')
    var rowNumber = row.parentElement.parentElement.rowIndex
    var transactionID = table.rows[rowNumber].cells[0].innerHTML;
    var newStatus = {
        orderStatus: "Payment Rejected"
    }
    //Update the status if rejected
    var JSONformatdata = JSON.stringify(newStatus);
    var xhr = new XMLHttpRequest();
    xhr.open("PUT", "http://localhost:5001/updateStatus/" + transactionID, true);
    xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
        }
    }
    xhr.send(JSONformatdata);
    window.location.reload()
}

function sortByAmount() {
    var table = document.getElementById('maintable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length - 1
    for (var x = 1; x <= tableLength; x++) {
        for (var y = 1; y <= tableLength - 1; y++) {
            if (document.getElementById('amount').value == "Descending") {
                if (parseInt(tr[y].getElementsByTagName("td")[4].innerHTML) <
parseInt(tr[y + 1].getElementsByTagName("td")[4].innerHTML)) {
```

```
                    tr[y].parentNode.insertBefore(tr[y + 1], tr[y])
                }
            }
            else if (document.getElementById('amount').value == "Ascending") {
                if (parseInt(tr[y].getElementsByTagName("td")[4].innerHTML) >
parseInt(tr[y + 1].getElementsByTagName("td")[4].innerHTML)) {
                    tr[y].parentNode.insertBefore(tr[y + 1], tr[y])
                }
            }

        }
    }
}

function filterListByUserId() {
    var userID = document.getElementById('userid').value
    var table = document.getElementById('maintable')
    var tr = table.getElementsByTagName("tr");
    var tableLength = tr.length - 1

    for (var x = 1; x <= tableLength; x++) {
        if (userID == "") {
            table.getElementsByTagName("tr")[x].style.display = ""
        } else {
            if (tr[x].getElementsByTagName("td")[3].innerHTML.toString() == userID)
{
                table.getElementsByTagName("tr")[x].style.display = ""
            }
            else {
                table.getElementsByTagName("tr")[x].style.display = "none"
            }
        }

    }
}

function viewMore(row) {
    var table = document.getElementById('maintable')
    var rowNumber = row.parentElement.parentElement.rowIndex
    var modal = document.querySelector('.modal-container');
    var transactionID = table.rows[rowNumber].cells[0].innerHTML;

    modal.classList.add('create-form-active');
    for (transaction in allTransactions) {
        if (allTransactions[transaction].transaction.idTransaction == transactionID)
{

            document.getElementById('transcID').innerHTML = "Transaction ID: " +
allTransactions[transaction].transaction.idTransaction
```

```
            document.getElementById('ordered').innerHTML = "Date Ordered: " +
allTransactions[transaction].transaction.dateOrdered.split('T')[0]
            document.getElementById('status').innerHTML = "Order Status: " +
allTransactions[transaction].transaction.orderStatus

            var alldetails = allTransactions[transaction].details
            var detailsCount = 1;
            for (details in alldetails) {
                var detailsTable = document.getElementById('table-list')
                var detailsRow = detailsTable.insertRow(detailsCount);
                var productName = detailsRow.insertCell(0);
                var category = detailsRow.insertCell(1);
                var quantity = detailsRow.insertCell(2);
                productName.innerHTML = alldetails[details].products.name + " - " +
alldetails[details].variant.variantName
                category.innerHTML = alldetails[details].products.category
                quantity.innerHTML = alldetails[details].details.quantity

                detailsCount++
            }
        }
    }

}

function closeTransaction() {
    var detailsTable = document.getElementById('table-list')
    var numofrows = detailsTable.rows.length;
    for (let x = numofrows - 1; x > 0; x--) {
        detailsTable.deleteRow(x)
    }
    var modal = document.querySelector('.modal-container');
    modal.classList.remove('create-form-active');

}
```