Documentation of the Code:

Dependencies and technologies

- To run this code, one would need to have Node.js and MySQL installed. Since all Node.js modules are included within the code, it is not necessary to install external libraries.
- This code, however, is dependent on two main modules that had been downloaded via npm. Should the following be removed from the files, the program will cease to operate:
  - MySql2
  - Express
- The code therefore uses the Express.js framework in order to communicate with a MySQL database.

Installation and how to run

1. Make sure that you have both Node.js and MySQL installed.
2. Download and unzip the code from this repository into your chosen directory.
3. Open MySQL Workbench or any other database management system. For this installation guide, MySQL Workbench will be used.
4. Establish a new connection.
5. When prompted, enter the following information into their respective fields:

- Hostname: localhost
- Username: root
- Port: 3006

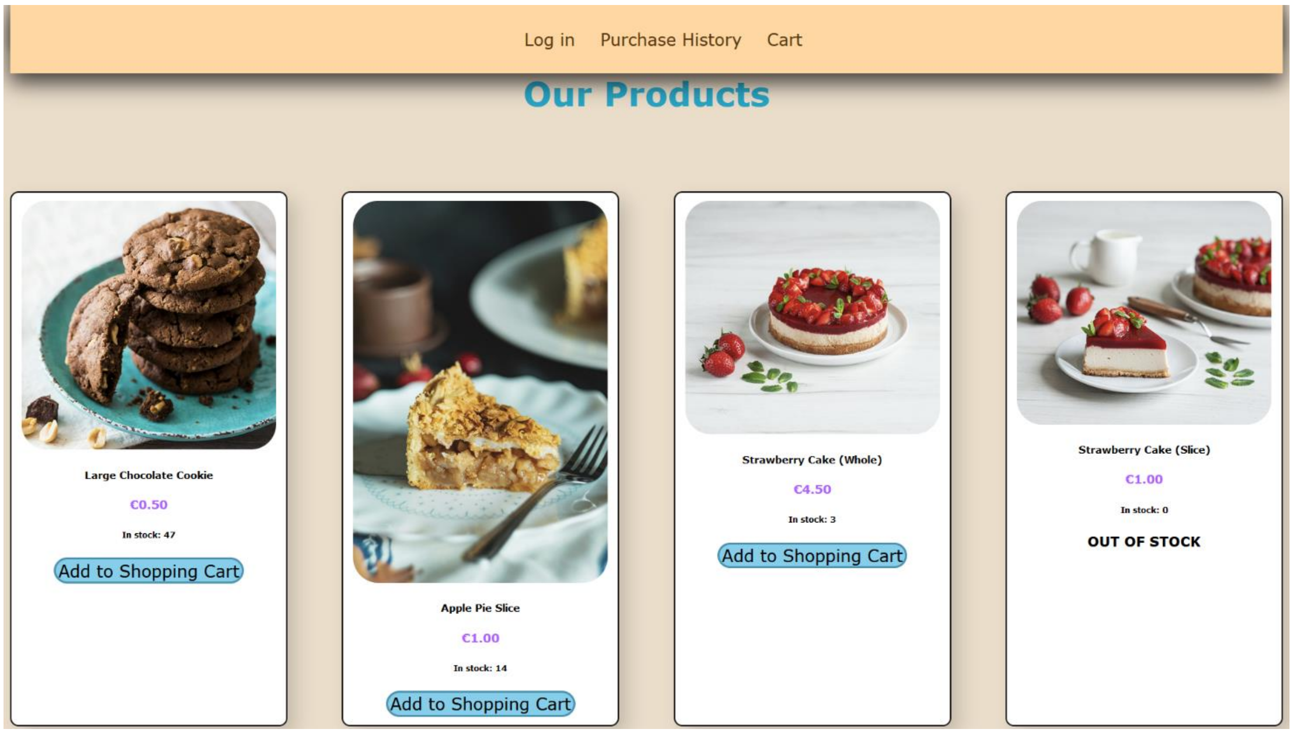
6. Open the connection and select the option to open an SQL script. Using MySQL Workbench, this option will be under `File> Open SQL Script`
7. Navigate to the directory `Phase 2` and select the file PortfolioDatabase.sql
8. Execute the query and refresh your schemas. Should the database `portfolio_phase_two` appear with the tables `accountinformation`, `purchasehistory`, and `stockinformation`, the database has been successfully installed. You can close MySQL Workbench or any other database management.
9. With a text editor of your choosing, open the `server.js` file and edit the following code:

```
const connect = mysql.createConnection({
    host: "localhost",
    user: "root",

    password: "your_password_here",

    database: "portfolio_phase_two",

    port: 3306,

});
```

*Please note that the password referred to here is the password you used for the MySQL connection during creation. Additionally, the code can be found roughly at the top of the file.

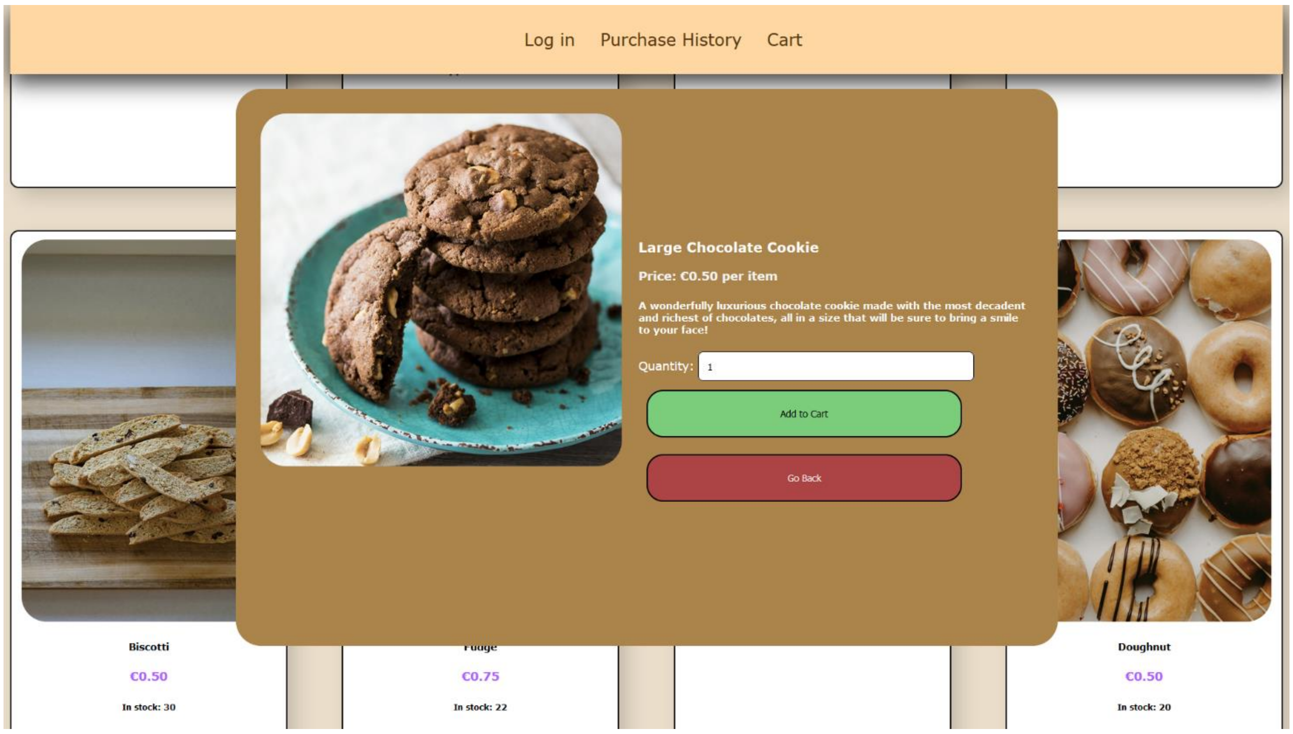
10. Open Command Prompt (Windows)/Terminal (macOS/Linux) and navigate to the directory where you extracted the files. You should be in the `Phase 2` directory. On Windows, an example directory path is as follows: `C:\Users\John\Documents\Phase 2>`
11. Run the command `node server.js`
12. Open your browser and type the following into your search bar: `localhost:3000/StorePage.html`

User Interface Documentation



- This is a screenshot of the *StorePage.html* page, which shows a title at the top of the screen, a sticky header, a main banner with some text, 12 different products, and a small box containing contact information at the bottom of the page, alongside a copyright disclaimer.

- Sticky header: this sticky header is always shown at the top of the screen. It contains three options: *Log in*, *Purchase History,* and *Cart*. Should the user be successfully logged into their account, the *Log in* option is changed to *Account*.
    - Log in → when clicked, this will redirect the user to the *LogIn.html* page, which allows the user to sign into their account
    - Purchase History → when clicked, this will redirect the user to the PurchaseHistory.html page, where the user will be able to view their history of purchases
    - Cart → when clicked, the user will be able to view their items within their shopping cart, as well as the quantity of a given item within

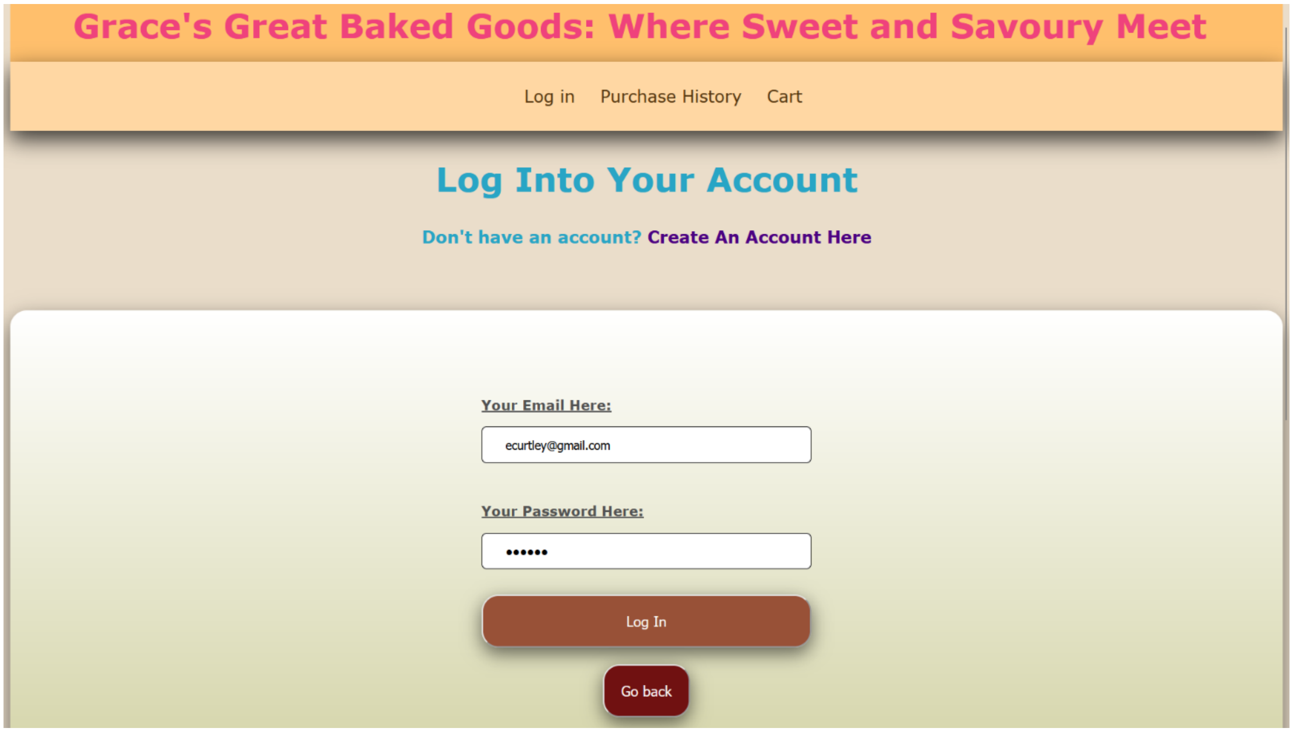


Large Chocolate Cookie

Price: €0.50 per item

A wonderfully luxurious chocolate cookie made with the most decadent and richest of chocolates, all in a size that will be sure to bring a smile to your face!

- This is a popup on the *StorePage.html* page. This appears when the *Add to Shopping Cart* button on a product listing has been clicked. Each popup has an image of the product that was listed, the name of the product, the price, a small description of the product, an input form, an *Add to Cart* button, and a *Go Back* button
    - Input form → a user inputs the amount of a certain product that they wish to buy. Should a user exceed the amount of stock available (shown on the product listing), then a browser alert is displayed
    - Add to Cart button → this button, when clicked, adds an item and the quantity to two arrays, namely the *ShoppingCartList* and *ShoppingCartAmount* arrays defined in *StorePageFunctionality.js*. An alert message is then showed to the user that the item had been added to the cart and the popup is closed
    - Go Back button → when clicked, this button simply closes the popup on the screen


Grace's Great Baked Goods: Where Sweet and Savoury Meet

Log in Purchase History Cart

Log Into Your Account

Don't have an account? Create An Account Here

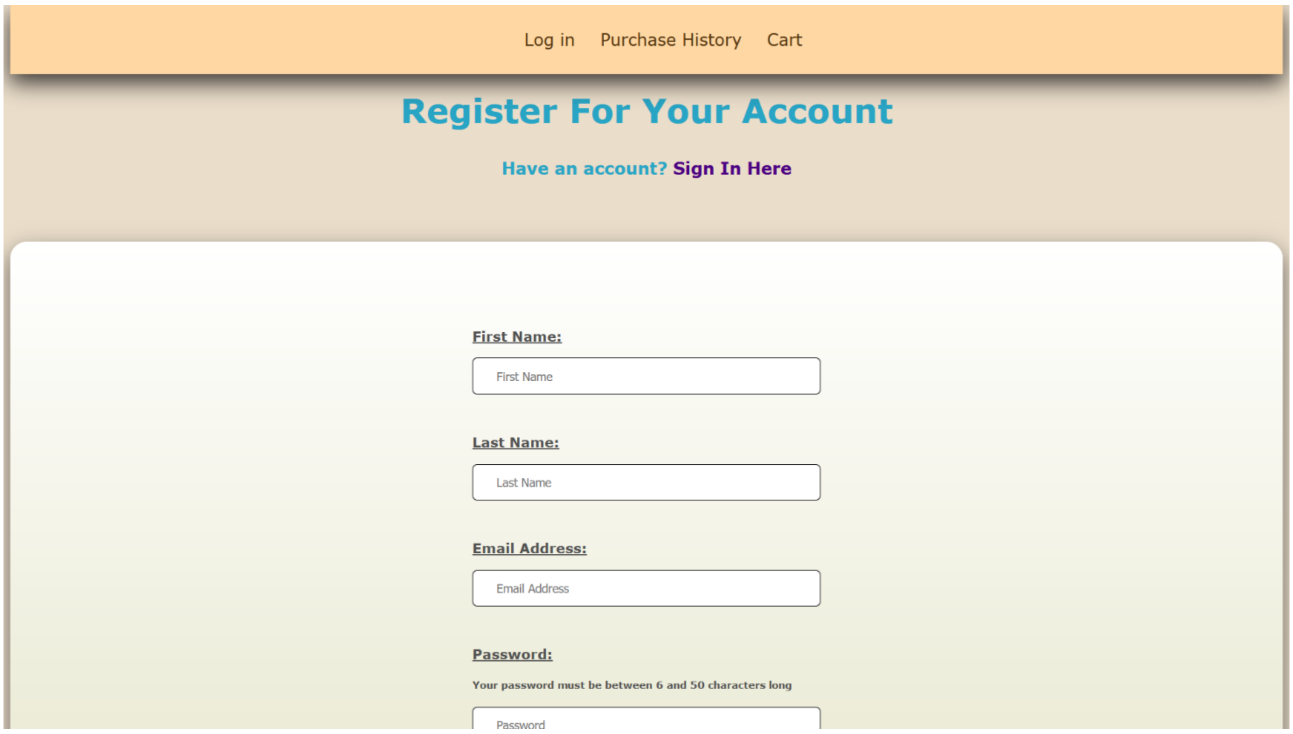
Your Email Here:

ecurtley@gmail.com

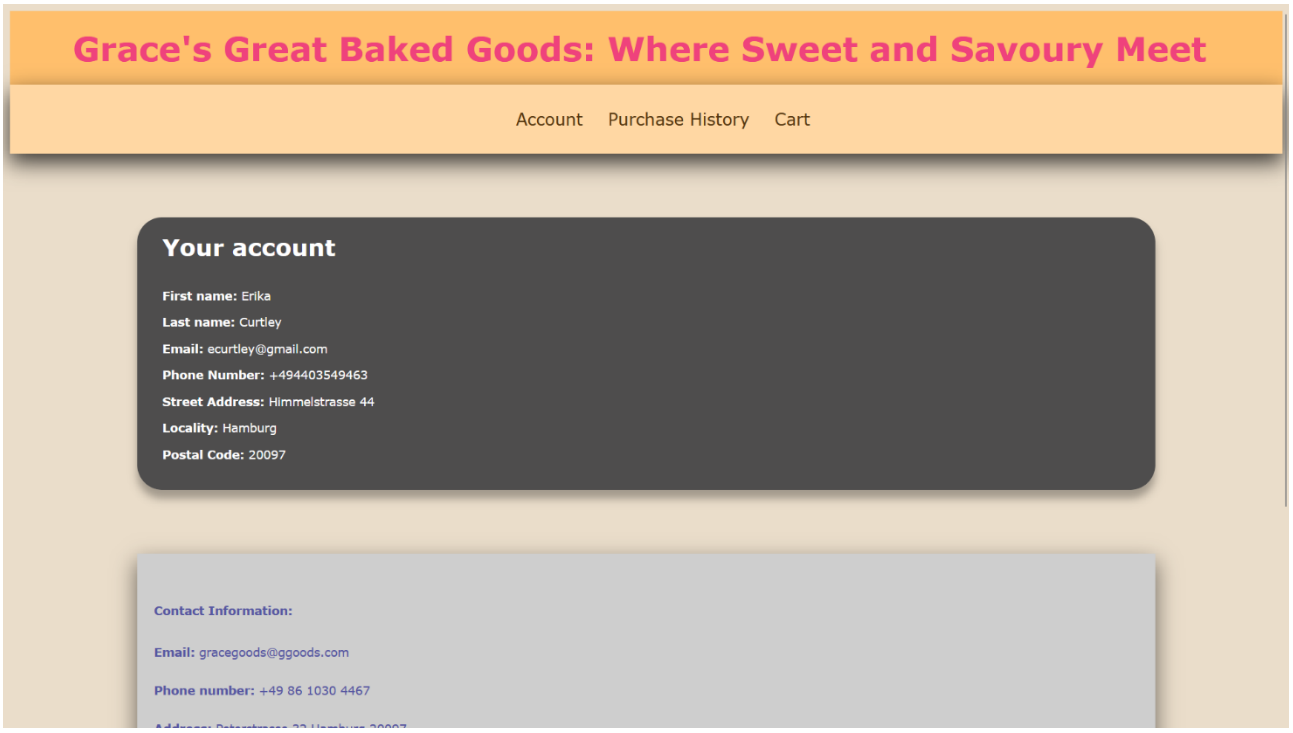
Your Password Here:

••••••

- This is a screenshot of the *LogIn.html* page, which shows a title at the top of the screen, a sticky header, a link, two input forms, a *Log In* button, a *Go Back* button, a contact information display, and some copyright information.
    - Link → when the *Create An Account Here* link is clicked, the user is redirected to the *SignUp.html* page
    - Input forms → the user is able to insert their email address and their password associated with their account in these fields.
    - Log In button → when clicked, the inputted information is compared to the information stored in the database. If the information matches, the user is shown a browser alert and gets redirected to the *StorePage.html* page with the sticky header being changed. If the inputted information is incorrect or whether the input forms are empty missing, the user is alerted of the respective issue.
    - Go Back button → when clicked, the user gets redirected to the *StorePage.html* page



- This is a screenshot of the *SignUp.html* page, which allows a user to create a new account. The page contains a title and a sticky header at the top of the screen, a link, 8 different input forms, a *Sign Up* button, and a *Go Back* button.
  - Link → when clicked, a user is redirected back to the *LogIn.html* screen
  - Input forms → here, there are several input forms that a user can insert their account information into, namely *First Name*, *Last Name*, *Email Address*, *Password*, *Phone Number*, *Street Address*, *Locality*, and *Postal Code*. It is noteworthy to mention that the input forms *Email Address*, *Password*, and *Phone Number* have additional input verifications. *Email Address* checks whether a user's input is in the format of an email address, *Password* checks whether an input is between 6 and 50 characters long, and *Phone Number* checks whether an input is formatted in a phone number. Should any of these format checks fail or if input forms are left blank, an alert is displayed describing that issue.
  - Sign Up button → this button, when clicked, retrieves the inputs and adds them as new entries to the database, thereby allowing a future user to log into their account with those credentials. A user is then shown an alert that their account has been created and redirected to the *LogIn.html* page



Your account

First name: Erika

Last name: Curtley

Email: ecurtley@gmail.com

Phone Number: +494403549463

Street Address: Himmelstrasse 44

Locality: Hamburg

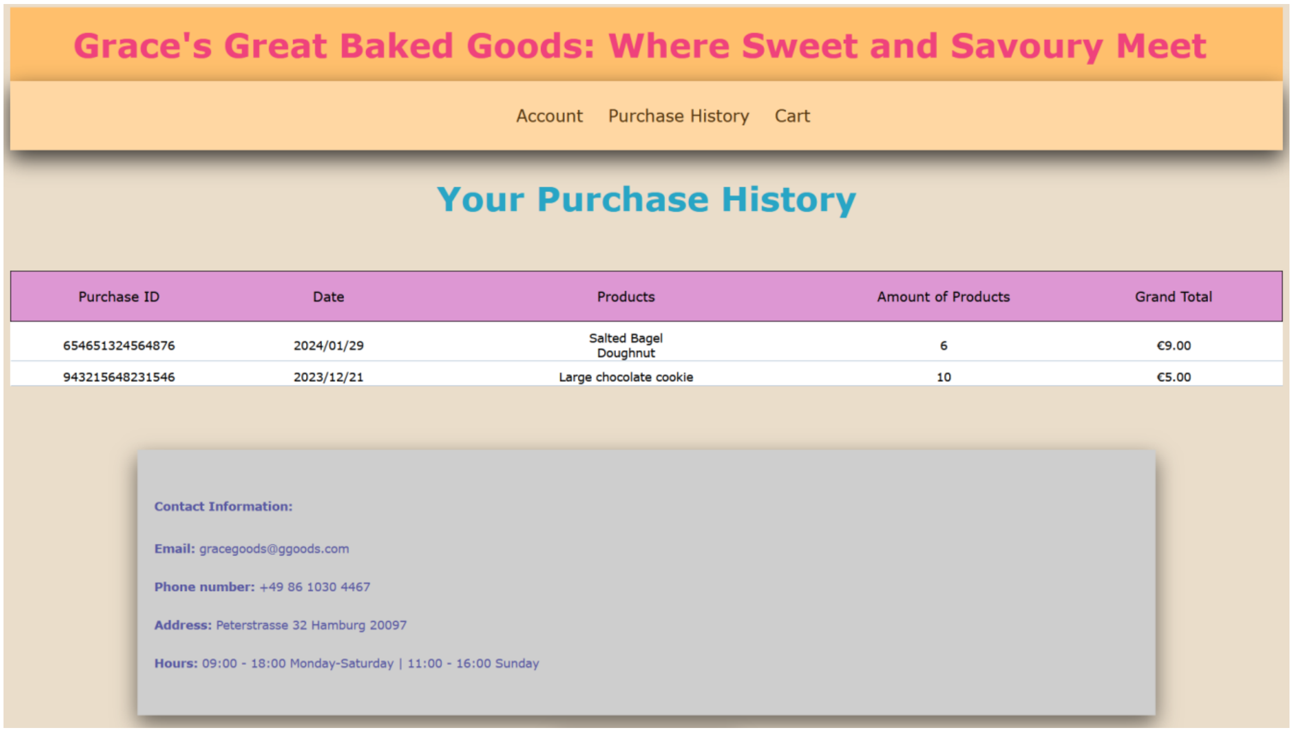
Postal Code: 20097

Contact Information:

Email: gracegoods@ggoods.com

Phone number: +49 86 1030 4467

- This is a screenshot of the *AccountInformation.html* page, which displays the information of a signed-in account. When loaded, the page retrieves information from the database by matching the signed-in user's User ID with that in the database, then displays that information under the *Your account* section. It displays the following information:
    - First name
    - Last name
    - Email
    - Phone Number
    - Street Address
    - Locality
    - Postal Code
- It is also worthy to mention that the sticky header in the screenshot does not show *Log in* as the first option, but *Account*. This indicates that a user is, indeed, signed in. This page is therefore accessed by clicking the new *Account* option.
- The page also contains a *Go Back to Main Page* button at the bottom of the screen. When clicked, it redirects the user to the *StorePage.html* screen.


Grace's Great Baked Goods: Where Sweet and Savoury Meet

Your Purchase History

Purchase ID	Date	Products	Amount of Products	Grand Total
654651324564876	2024/01/29	Salted Bagel Doughnut	6	€9.00
943215648231546	2023/12/21	Large chocolate cookie	10	€5.00

Contact Information:

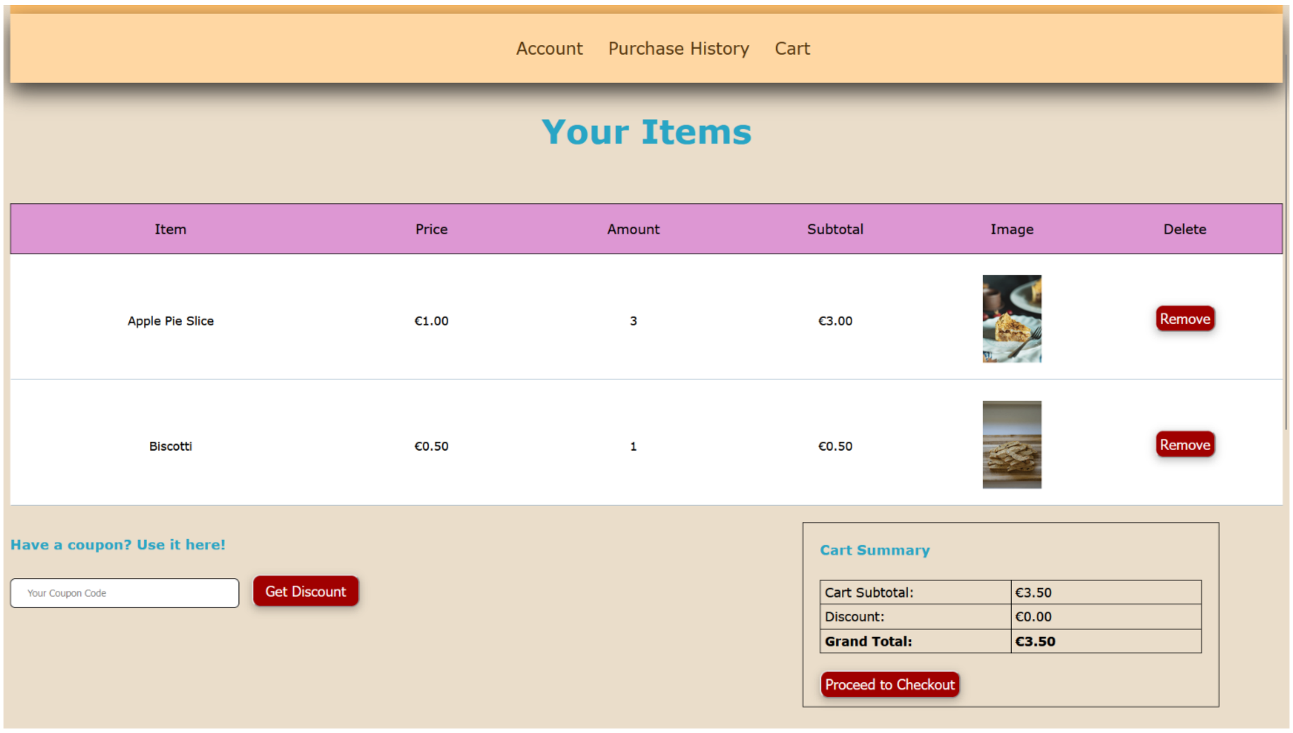
Email: gracegoods@ggoods.com

Phone number: +49 86 1030 4467

Address: Peterstrasse 32 Hamburg 20097

Hours: 09:00 - 18:00 Monday-Saturday | 11:00 - 16:00 Sunday

- This is the *PurchaseHistory.html* screen, accessible by clicking on the *Purchase History* option in the sticky header. This page contains, by default, a table header, a contact information section, and a button to go back to the main page. When a user is logged in, the page checks whether the user has any past purchases stored in the database.
    - If a user has a history, the information is dynamically displayed on the page in the correct columns under the header, as seen in the screenshot above.
    - If a user does not have a history, the page remains the same, as if a user hasn't logged in at all.
- This table is also responsive in the sense that, on smaller screens, one could horizontally scroll through the table's entries due to it not fitting within the screen.



Your Items

Item	Price	Amount	Subtotal	Image	Delete
Apple Pie Slice	€1.00	3	€3.00		Remove
Biscotti	€0.50	1	€0.50		Remove

Have a coupon? Use it here!

Cart Summary

Cart Subtotal:	€3.50
Discount:	€0.00
Grand Total:	**€3.50**

- This is the *Cart.html* page, which contains a dynamically created table, an input form for coupons with a corresponding *Get Discount* button, and a Cart Summary table which hosts the *Proceed to Checkout* button. By default, only the header of the table is displayed, but as more items are added to the cart, the more items will appear in the table.
  - Table → once the page is loaded, it retrieves all values from the arrays *ShoppingCartList* and *ShoppingCartAmount*; the former being displayed in the *Item* column and the latter being displayed in the *Amount* column. Additionally, the price of the individual item and the subtotal of each item is displayed, alongside an image of the item in question. Finally, there is the *Remove* button in each row. When clicked, the entry in the table is deleted, the respective entry in the arrays are wiped, and the updated price is shown in the *Cart Summary* table
  - Coupon input form and Get Discount button → the user has the option to input a code that allows a 5% discount to be applied to the total price. When the *Get Discount* button is clicked, the inputted code is compared to any matching entries within the database. If there is a matching entry, an alert is displayed informing the user of such and the *Cart Summary* table is updated to reflect the discount. If there had been no information inputted or if the code is incorrect, the user is informed of the respective issue via an alert
  - Cart Summary table → this table, by default, isn't shown. However, should there be any items in the cart, a table will be dynamically generated to display the subtotal of all items in the cart, the price deducted by any coupons if there had been a coupon applied, and the grand total price

    - Proceed to Checkout button → once clicked, this button stores all the items, the total amount of items bought, and the grand total price into the database. Afterwards, the user is prompted with a popup confirming that their purchase had been successful, A button within the popup is displayed which, when clicked, redirects the user to the *StorePage.html* page. The information that has been stored can then be viewed in the *PurchaseHistory.html* page

Functions

- As all functions had been documented within the code itself, this document will not go over the functionality of those functions again. However, this document will specify where each function is used within the web application.

- *gobackToMainPage()*
    - Used in *AccountInformation.html, Cart.html, LogIn.html, PurchaseHistory.html,* and *SignUp.html, as well as within the closePopUpMessageChecout()* function in *StorePageFunctionality.js*
    - This can be seen used in the *Go back* and the *Go Back to Main Page* buttons across the aforementioned HTML files
- *changeStickyHeader(signedIn)*
    - Used in the *checkIfLoggedIn()* function and the *loggingIn()* function, both found in the *StorePageFunctionality.js* file
- *showPopUpMessage(elementID)*
    - Used in *StorePage.html*
    - This can be seen used 12 separate times under the “onclick” property of the <a> tag for all the individual product listings
- *closePopUpMessage(elementID)*
    - Used in *StorePage.html* and within the *addToCart(addedItem, amountToBuy, availableStock, elementID)* function within *StorePageFunctionality.js*
    - Within *StorePage.html*, it can be seen used 12 separate times in the popup for each product listing for the *Go Back* button within said popup
- *addToCart(addedItem, amountToBuy, availableStock, elementID)*
    - Used in *StorePage.html*
    - Used 12 different times in the popup for each individual product listing and is executed when the *Add To Cart* button is clicked for that respective popup
- *storesArrays()*
    - Used in the *addToCart* function within *StorePageFunctionality.js*

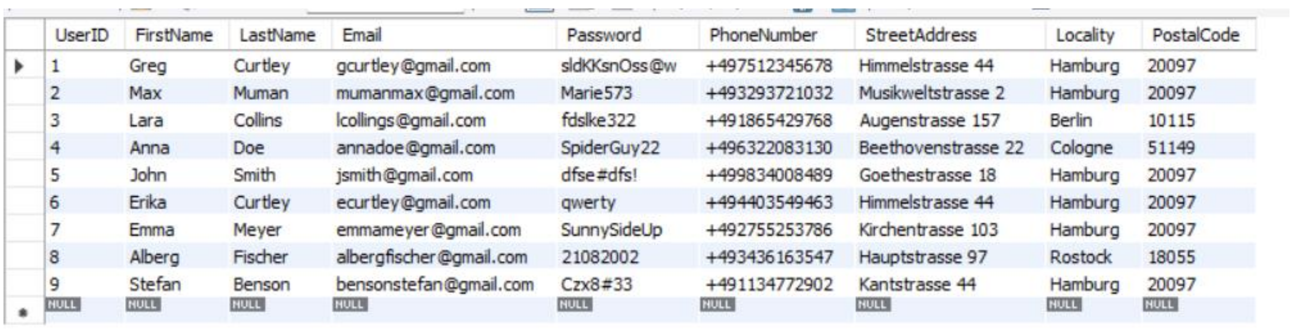
- *signUpFormatting()*
    - Found within the *signingUpForNewAccount()* function in *StorePageFunctionality.js*
    - This function is executed when the *Sign Up* button is clicked within *SignUp.html*, however an event listener is used instead of incorporating the function directly into the HTML file
- *displayOnCart()*
    - Used in *Cart.html*
    - The function is called when the page loads and can be found at the bottom of the *Cart.html* file
- *proceedToCheckout()*
    - Used in *Cart.html*
    - The function is executed when the *Proceed to Checkout* button on the page is clicked; the button having the ID *ProceedToCheckoutButton*
- *closePopUpMessageCheckout()*
    - Used in *Cart.html*
    - Can be found in the popup shown after the user has checked out and is executed when the *Go Back to Main Page* button under the *ClosePopUp13* ID has been clicked
- *displayStockForProduct(productListing, stockHeader, addToCartButton, upperLimitInPopup)*
    - Used in *StorePage.html*
    - Can be found used 12 separate times at the bottom of the page and is executed when the page is loaded. It receives the parameters of each individual product listing, hence why it is called 12 times
- *reduceStockAvailable()*
    - Used in the *proceedToCheckout()* function in *StorePageFunctionality.js*
- *displayUserAccountInfo()*
    - Used in *AccountInformation.html*
    - This function is executed when the page is loaded and can be found at the bottom of the respective HTML file
- *checkIfLoggedIn()*
    - Used in *AccountInformation.html, Cart.html, PurchaseHistory.html,* and *StorePage.html*
    - In all instances, the function is called when the page is loaded and can be found on the bottom of each respective HTML file
- *loggingIn()*
    - Used in *LogIn.html*

  - This function can be found and is executed when the *Log In* button is clicked. The button in particular has the ID *LogInButton*
- *getPurchaseHistory()*
  - Used in *PurchaseHistory.html*
  - The function is called when the page is loaded and can be found at the bottom of the HTML file
- *signingUpForNewAccount()*
  - Used in *SignUp.html*
  - Can be found attached to the *Sign Up* button with the ID *SignUpButton*. The function is executed when the button is clicked
- *checkCoupon()*
  - Used in *Cart.html*
  - Can be found assigned to the *Get Discount* button with the ID *CouponButton*. The function is executed when the button is clicked

Database data

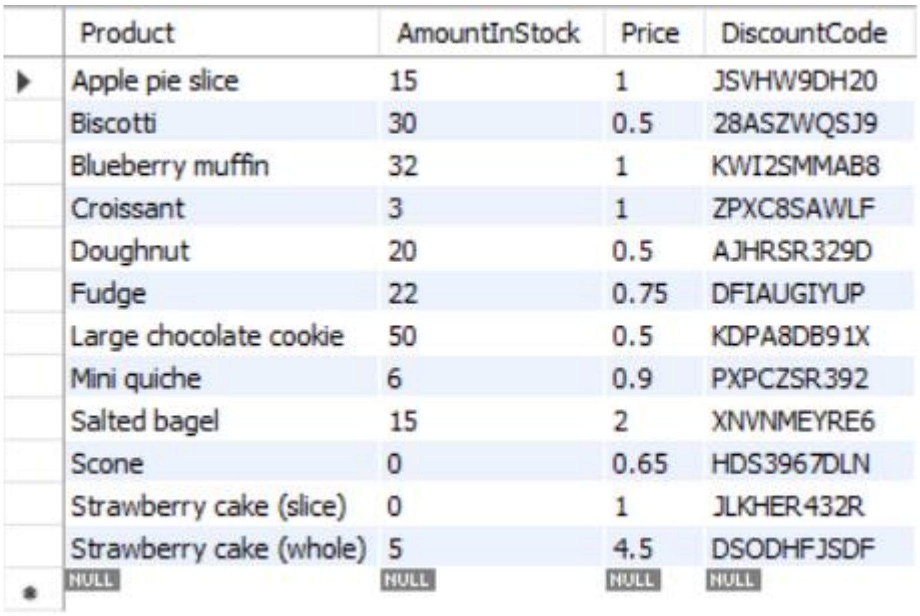
- The database currently stores three separate tables, namely *accountInformation*, *stockInformation*, and *purchaseHistory*. The following images contain the data stored in the database tables by default, i.e., when the database is initialised via the MySQL script included:

<u>accountInformation</u>



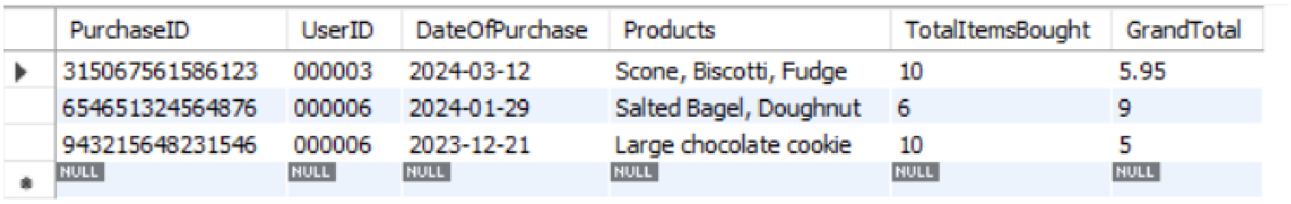
	UserID	FirstName	LastName	Email	Password	PhoneNumber	StreetAddress	Locality	PostalCode
▶	1	Greg	Curtley	gcurtley@gmail.com	sldKKsnOss@w	+497512345678	Himmelstrasse 44	Hamburg	20097
	2	Max	Muman	mumanmax@gmail.com	Marie573	+493293721032	Musikweltstrasse 2	Hamburg	20097
	3	Lara	Collins	lcollings@gmail.com	fdslke322	+491865429768	Augenstrasse 157	Berlin	10115
	4	Anna	Doe	annadoe@gmail.com	SpiderGuy22	+496322083130	Beethovenstrasse 22	Cologne	51149
	5	John	Smith	jsmith@gmail.com	dfse#dfs!	+499834008489	Goethestrasse 18	Hamburg	20097
	6	Erika	Curtley	ecurtley@gmail.com	qwerty	+494403549463	Himmelstrasse 44	Hamburg	20097
	7	Emma	Meyer	emmameyer@gmail.com	SunnySideUp	+492755253786	Kirchentrasse 103	Hamburg	20097
	8	Alberg	Fischer	albergfischer@gmail.com	21082002	+493436163547	Hauptstrasse 97	Rostock	18055
	9	Stefan	Benson	bensonstefan@gmail.com	Czx8#33	+491134772902	Kantstrasse 44	Hamburg	20097
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

stockInformation



Product	AmountInStock	Price	DiscountCode
Apple pie slice	15	1	JSVHW9DH20
Biscotti	30	0.5	28ASZWQSJ9
Blueberry muffin	32	1	KWI2SMMAB8
Croissant	3	1	ZPXC8SAWLF
Doughnut	20	0.5	AJHRSR329D
Fudge	22	0.75	DFIAUGIYUP
Large chocolate cookie	50	0.5	KDPA8DB91X
Mini quiche	6	0.9	PXPCZSR392
Salted bagel	15	2	XNVNMEYRE6
Scone	0	0.65	HDS3967DLN
Strawberry cake (slice)	0	1	JLKHER432R
Strawberry cake (whole)	5	4.5	DSODHFJSDF
NULL	NULL	NULL	NULL

purchaseHistory



PurchaseID	UserID	DateOfPurchase	Products	TotalItemsBought	GrandTotal
315067561586123	000003	2024-03-12	Scone, Biscotti, Fudge	10	5.95
654651324564876	000006	2024-01-29	Salted Bagel, Doughnut	6	9
943215648231546	000006	2023-12-21	Large chocolate cookie	10	5
NULL	NULL	NULL	NULL	NULL	NULL