Dr. Mucahid Kutlu

CMPS 350, L02

Project Phase 1

E-Commerce Platform: Phonie

Haris Khan, 202104099

AhmedFaseeh Akram, 202005685

Mohammed Al-Hasna, 202009492

# Introduction

githubLink: https://github.com/Hazey0/CMPS350-Project.git

Structure:

1. Intro
2. Project Descriptions
   1. Basic structure
   2. File and their uses
   3. Use Cases 1-6
3. Deliverables and important notes
   1. Design UI and navigation

# Project Descriptions

Our Project name is called Phonie. It is an e-commerce platform that exclusively sells phones. Once you launch live server, on the main page you will greeted a nav bar which has our logo, search bar and login button. We have 4 main types of users:

1. Guest: can view products sort products, but has to login to buy any product.
2. Customer: has to first login to buy any product and can see previous transactions
3. Seller: can add a product to sell
4. Admin: can remove any phone and can add/remove from featured phone.

Customers, sellers and admin can all view their own personal details. Customer has a name, surname, shipping address, username, password, and money balance. Seller has a company name, username, password and a bank account. Admin has a username and a password. We did not create classes, we created phones.json files and users.json which fulfil the criteria of what each user should have.

# Use Cases

**Use Case 1: Login Page**

For this use case, a customer can only login if the user credentials (username, password) are in the users.json file. Here are the screenshots showing the possible scenarios:

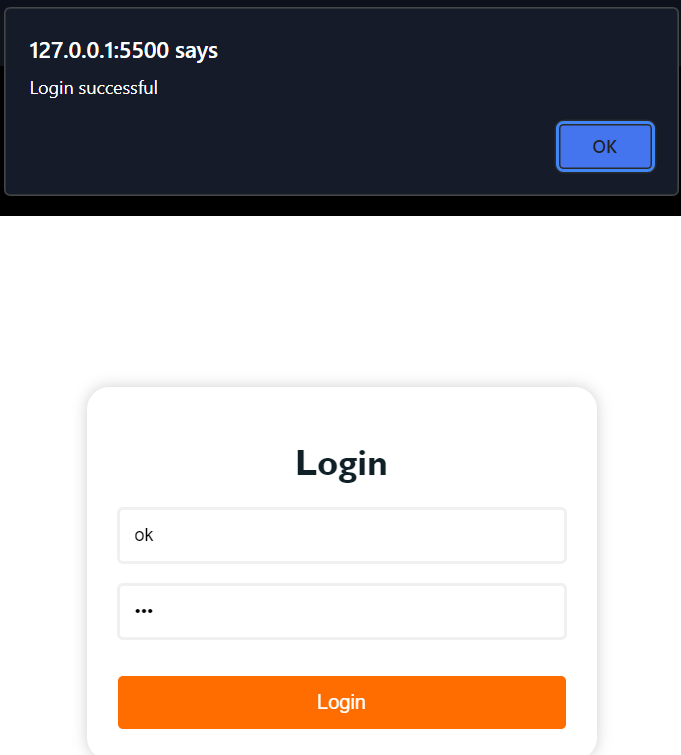


Figure 1. Correct Login

After a successful login, the logged user will be directed to the main page.

We uses localStorage to get users.json file which has all the details for each user: admin, customers, sellers. We have EventListener on login button and we have functions which verify login using the log() in loginInit.js file. In the log() we first get users.json data which will be made into an array. Then we verify if the array is not empty or if the given username and password are in the user array. If the username and password text field are empty, it will give an alert. Also, an alert will be given for the incorrect/correct login attempts.

**Use Case 2: Search Available Item**

All 4 types of user can search and view all the available products. Here are some screenshots of search functionality:

A screenshot of a phone

Description automatically generated

Figure 2. search: 512

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generated

Figure 3. search: 10 Figure 4.search: apple

The searchFunction.js file implements the search functionality. We first, start loading phone data from local storage into an object for manipulation. The ‘searchPhone’ function activates user search, clearing previous results and initializing a fresh search. At the heart of this process is the ‘searchAlgo’ function. This function analyzes user input, breaks it down into keywords, and applies multiple filters to match these keywords against phone attributes like brand, name, storage, and year. Successful searches are rendered in a specific webpage container, showing users the matching phones. Additionally, a cancel search feature, symbolized by an "X" icon, allows users to quickly clear results and view the original phone list.

**Use Case 3: Purchase an item**

In our project, we only allow customers to buy phones. And they have to be logged in as well otherwise you will get an alert and redirected to the login page:

Here are some screenshots of phone purchase (this customer has 10,000QR in his bank:A screenshot of a cell phone

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a computer

Description automatically generated A black background with white text

Description automatically generated

Figure 5. Successful Purchase

As you can see in figure 5, there are 3 quantities of this phone and after ordering successfully, you will be redirected to the transactions.html page. Which you can see what you ordered. If you go back to the same phone, you will see the quantity number will decrease by the amount of the phones bought. The customer’s money will decrease buy the phone’s price.

**Use Case 4: View the purchase history**

Once a customer is logged in, he can view his purchased Items by clicking on his name and then clicking transaction. After purchasing a phone, you will also redirected to the transaction.html page.

A screenshot of a computer

Description automatically generated

Figure 2. Transaction page

**Use Case 5: View available items on sale and sale history**

A seller can only see what phones he can sell and sold phones.

A screenshot of a cell phone

Description automatically generated

Figure 3. List of phones to sell

A screenshot of a computer

Description automatically generated

Figure 4. Previous sales

**Use Case 6: Upload an item to be sold**

When a user is logged in, he can add a phone to sell.

A screenshot of a computer

Description automatically generated

Figure 5. filling out the form correctly

**Extra Functionality**

Here is a list of extra functionality we have done

Sorting by parts

User details

Featured Devices

Admin can delete any phone

Seller can delete his own phone

Seller can increase the quantity of a specific phone

Increase the quantity of phones

# Web Design Choices

For our project, we chose to do a simple design which we chose for all pages: white, black, and orange color scheme. For all the pages and users, there will be a navigation bar which will allow each user to access functionality depending on what type of user it is. For ease of access, all users (including Guests) can view phone catalog on the main page and sort depending on 4 values: name, year, price and storage (in ascending or descending order). We also added alert messages where appropriate, for example when a user enters incorrect value or successfully purchased a phone. On the development side, we chose to split our project into 5 folders for clean and simple structure. The Script folder holds all the JavaScript files which handle all the functionalities and client-side scripts, and sub-folder is called storage which hosts the pre-made json files. The Styles folder holds all the CSS files to make our look presentable and implement our design choices we decided. In the Website folder, it holds all the html pages for our project. Media folder holds two sub-folders: Icons and Images. The final folder named documentation holds this report and demo videos.

# Code Explanation+ Testing

**Code Explanation**

In our Phonie project, we built the HTML layout to be simple and straightforward for all users, including customers, sellers, and admin. We created separate pages for each section of our e-commerce platform. The main page is where everything begins, with top phone picks and options for sorting and searching to help consumers find what they're looking for. Our login page is straightforward, allowing users to effortlessly access their accounts and welcoming new people to join us. Pages that display phone and they change depending on who is viewing, including buy buttons for customers and options for sellers to emphasize their phones. Pages built for user details and transaction records provide individualized information, improving the user experience. We have simplified the forms for selling and buying phones, with the goal of removing any bother from these transactions. We ensured a smooth experience for consumers across the Phonie platform by using a single design style and clear navigation, making purchasing and selling phones online as simple as possible. Pages built for user details and transaction records provide individualized information, improving the user experience. We have simplified the forms for selling and buying phones, with the goal of removing any bother from these transactions. We ensured a smooth experience for consumers across the Phonie platform by using a single design style and clear navigation. This gives a simple design yet functional design.

On the stylistic front, our CSS strategy captures the idea of simplicity and elegance, resulting in an easy-to-use interface. We used a consistent color palette and created the layout, which included a fixed navigation bar to ensure simple access to numerous capabilities adapted to different user roles. Our solution emphasizes responsive design and is tailored for a smooth presentation across devices, improving accessibility and usability. Our CSS organization includes distinct files for scripts, styles, and media. The interactive elements, decorated with shadows, transitions, and hover effects, guide customers around our site, combining beauty and functionality to provide an engaging user experience in the online phone marketplace.

The dynamic and functional part of our project is possible by using JavaScript and JSON, allowing for interactive and responsive features. Our JavaScript files are dedicated to specific tasks, such as rendering phone numbers and managing user logins, as well as performing search and sort operations and transaction processing. This modular strategy, combined with the usage of JSON files to store static data such as phone specifications and user information, creates a database-like environment that ensures the project's scalability and ease of maintenance (but quite insecure though). Thanks to event listeners and asynchronous fetch calls, our implementation supports capabilities like form validations, transaction handling, and dynamic content loading without the need for page reloads. The Phonie project transforms into an engaging and efficient platform by using JavaScript and Json files.

**Testing**

Here are our testing for the use cases.

Use Case 1: Loggin in

A screenshot of a login screen

Description automatically generated

Figure 6. Incorrect Login Attempts

Use Case 3: Purchase a Phone

A screenshot of a phone

Description automatically generated

Figure 7. Guest tries to buy phone

Gues user will be prompted to login and redirected to the login page

A screenshot of a phone number

Description automatically generated

Figure 8. Unsuccessful purchase attempt

Use Case 6: uploading a phone to sell:

A screenshot of a computer

Description automatically generated

Figure 9. Incorrect form fill out

For all pages we also check if the correct type of user is accessing the correct page or performing that job which only that user is allowed to do. This is done in the JavaScript files. As you can see in figure 1, a similar validation is put on every page/functionality where it is appropriate. This is to follow the guidelines in the project.