AISLE TRACKER

[1]Sonali Mondal, [1]Sai Kashyap Cheruku, [2]HemaSree Bojanapally, [2]Namratha Pilli,

[1]Cyber-Physical Systems, Northeastern University
[2]Information Systems, Northeastern University

mondal.so@northeastern.edu | cheruku.sa@northeastern.edu

bojanapally.h@northeastern.edu | pilli.n@northeastern.edu

***Abstract*—Aisle Tracker is a tool that may be effectively used to reduce the difficulties that shoppers encounter in vast retailers. The implementation of Aisle Tracker is suggested in this report utilizing Java based object-oriented programming paradigms and Scene Builder assisted JavaFX, a framework for quickly developing graphical user interfaces. The Model, View, Controller (MVC) design pattern is used to clearly divide the main parts of the application so that the front end and back-end development may be done concurrently and integrated on completion. The results of this application clearly show how to apply a dynamic aisle-based search algorithm, which can improve customers' shopping experiences by dynamically showing aisle and pricing information on their mobile devices.**

Keywords—MVC, search algorithm, retailers, user interface

I. PROBLEM DESCRIPTION

After a long work week, going shopping can be a lot of fun. Whether it's for food or just clothes and home decor, shopping can be a great weekend activity. Large shopping establishments, however, may occasionally be highly overwhelming due to the sheer volume of merchandise offered. From personal experience, we can explain that it could be challenging to locate a certain item and its price on the grocery shelves. We spend a lot of time looking for items in the several aisles before looking for scanners to look up costs. Therefore, we created an application that would make locating products when shopping easier in order to decrease the effort required. To save time and effort while shopping, the solution proposed is to create an application that will help customers who prefer going to supermarkets in person and completing in-store purchases rather than submitting orders online. The application's search box allows users to enter a product name, and the results are displayed together with the pricing of the relevant items. The consumer may then select the item they want to purchase, and the software will let them know exactly where it is present in the store. The user may hence prevent wasting time by browsing all the aisles, instead simply go there and pick it up.

II. ANALYSIS (RELATED WORK)

To prevent the distance shoppers, travel within the store and their resulting unplanned purchases at the grocery stores, we plan to develop an app that will assist customers who prefer visiting supermarkets in person and completing in-store purchases rather than placing orders online to save time and effort while shopping. The main idea of our application has come from the utility that it has in our day-to-day life and the impact it can create. We got inspired from similar applications and features being deployed in many stores like Walmart [1], Target etc. We had referred to the architecture of this utility which helped us show how a real time application might work [2].

We worked on developing this utility for use by all merchants as well as adding improvements to enhance the customer's shopping and invoicing experience. This was accomplished by including the aisle section as a component of the cart, resulting in the creation of an "aisle-based cart setup" that enables a consumer to examine the cart whilst also displaying the various goods and their pricing in each aisle.

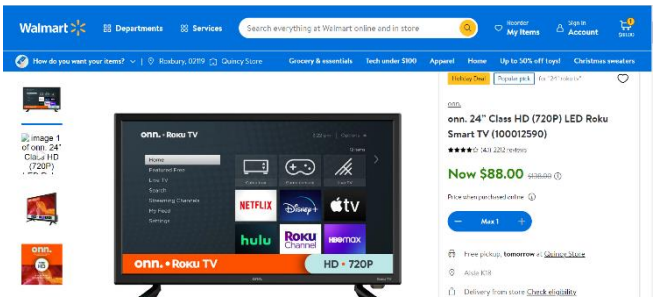

Figure 1. Walmart aisle feature

III. SYSTEM DESIGN

The application is designed to match the look and feel of an e-commerce application. The product class stores all the product details like aisle number, cost etc. The search bar is backed by a basic search algorithm to return results. The cart entry class and cart class are used to store the details and the products of the cart. The cart class stores data in the form of a linked TreeMap.

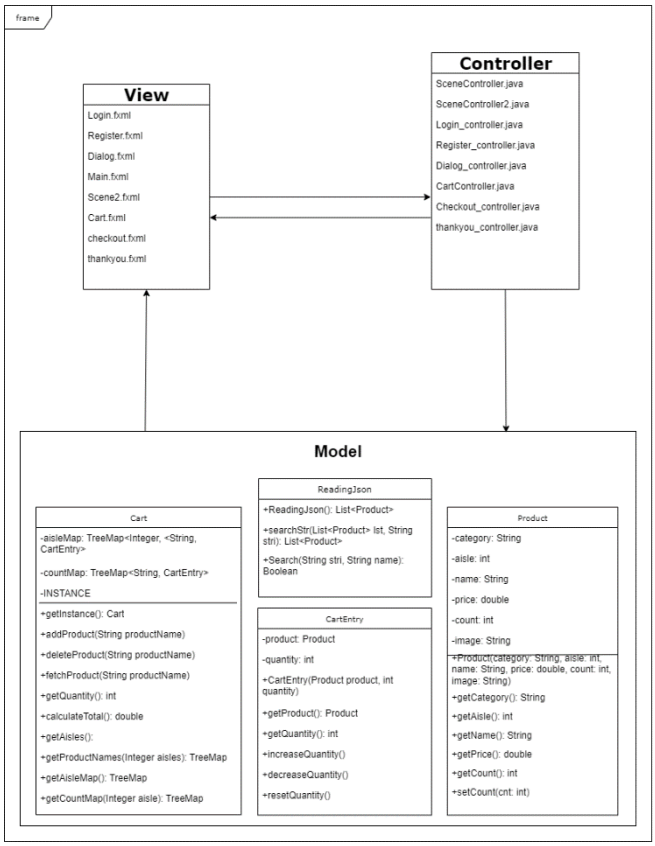

Figure 3. UML diagram

IV. IMPLEMENTATION

A. *Login/Register page*

Prior to beginning the shopping process, a user is expected to login with his/her credentials. This information is required to track user trends, give suggestions and view previous orders(can be included in future updates). For the present version of our application, the user needs to login inorder to access the cart. If there is no account present, they can go ahead and register using the "register now" option. Login authentication and user registration is facilitated by linking controller classes with the respective UI components, these controller classes perform the job of polling a json file containing registered credentials, for a login event and appending new pair of credentials to the json file, in-case of a register event. The user is notified of any incorrect authentication attempts, by means of invoking a dialog prompt(pointing at instances such as registering a used username, trying to login with an alien username etc). This dialog prompt halts the user from navigating further, unless he/she acknowledges it.

B. *Home and Search Page*

Once login is successful, the user is directed to the main page. The main page consists of a categorical view of all products available for sale. The product details are scrubbed from a Json file named “aisles.json”. This scrubbing is done in the class “ReadingJson” using the simple Json library. The “ReadingJson” class additionally created “Product” class objects for each of the products scrubbed from the Json file and stores them in a List of Product objects called “ProdArray”. The “Product” class has dataFields that store name, aisle, category, price and count data.

The UI for the categorical classification tab is done using the Tab pane object that is available on Scene Builder. Each product is displayed with its corresponding name, price, and the aisle that it can be found on. This is done to eliminate the need for firstly browsing the aisles to locate the product as well as looking for a scanner to determine the price of the same. Each product has a “+” button using which the user can add the product to their cart. On doing so, the number of products added to the cart is displayed along with a “-“ button, in case the user decides that they don’t want to in fact add that product to cart. These buttons are designed in a way that if the count of products being added to cart go below 0, the “-“ button gets disabled and if the number of products being added to cart crosses the actual count of that particular product(that is fetched using the getCount() method in the “Product” class) available in the inventory, the “+” button gets disabled. “SceneController” class is used to implement this functionality. “Main.fxml” is the corresponding xml file.

Additionally, there is a search bar using which Users can search for the specific product that they are looking for. The search is backed by a simple substring-matching algorithm that returns a list of “Product” objects with names matching the search string. The UI is designed in a dynamic manner. The resulting string search list is traversed and required data is fetched and displayed on the UI page. TextView, ImageView, TextArea are some of the Scene Builder objects that were used to implement the search page UI. “SceneController2” class is used to implement this functionality. “Scene2.fxml” is the corresponding xml file.

There are some additional buttons to navigate between pages. The “back to login” button can be used to go back to the login page. The “Home” button on the search results page can be used to navigate back to the home page. The “Cart” and “Checkout” buttons are used to jump to the respective pages.

C. *Cart Page*

On the cart page, the user finds all the items that they added to cart organized based on the aisles they are located in. The user can then simply go to those aisles and pick the items, thus saving time.

There are two classes with the help of which the cart functionality is implemented. The “CartEntry” class is created to maintain records of each “Product” class object with the corresponding quantity present in the cart. There are methods to increase and decrease the quantity corresponding to the product. The “Cart” class maintains an aisle based record of products in the cart. The cart is implemented using a nested TreeMap. The outer map is called “aisleMap” which has aisle numbers as keys and corresponding inner TreeMap as value. The inner TreeMap is called “countMap”. This map has product name as keys and the “CartEntry” object corresponding to that object as value. There are “addProduct” and “deleteProduct” methods in the cart class that call the increase and decrease quantity methods in the “CartEntry” class to perform necessary operations based on the “+” and “-“ button clicks that take place on the home page and search page. The is a calculate total method in the “Cart” class that calculates the total amount of the cart and returns the same.

The UI of the cart page is done in a dynamic manner as well. “Scroll Pane” object is being used in scene builder. There are Text fields for the aisle number to be displayed in and TextArea, TextView and ImageView objects being used for displaying product data. The aisle information, and number of products in each aisle and quantity of each product is being extracted from the nested TreeMap in the cart class. Along with this, the total price of the number of units of each product is calculated and displayed using a TextView object. The cart class has a defined “INSTANCE” method to make sure every time the “Cart” object is being called, the same instance of “Cart” is returned.

The cart page is controlled by the “CartController” class and “Cart.fxml” is the corresponding xml file.

D. *Checkout and Thankyou Page*

On clicking the checkout button, the product count is updated in the backend so that the inventory is up to date. Once the user clicks the “Checkout” button, the checkout page is loaded. The user can generate a bill using the button and the total amount to be paid will be displayed. The total amount is calculated in the “calculateTotal” method in “Cart” class. Once this is done, the user can pay at the counter and checkout from the app by clicking the button at the bottom of the page. Once this is done, the number of products in the inventory is updated by subtracting the count of checked out items from the initial count in the inventory, hence keeping the inventory updated as well. On clicking the “checkout” button, the user is redirected to a “Thank you” screen.

“Checkout_controller” and “Thankyou_controller” classes are used for this functionality and “checkout.fxml” and “thankyou.fxml” are the corresponding xml files.

V. Evaluation

- The user can either login or register with his/her credentials.

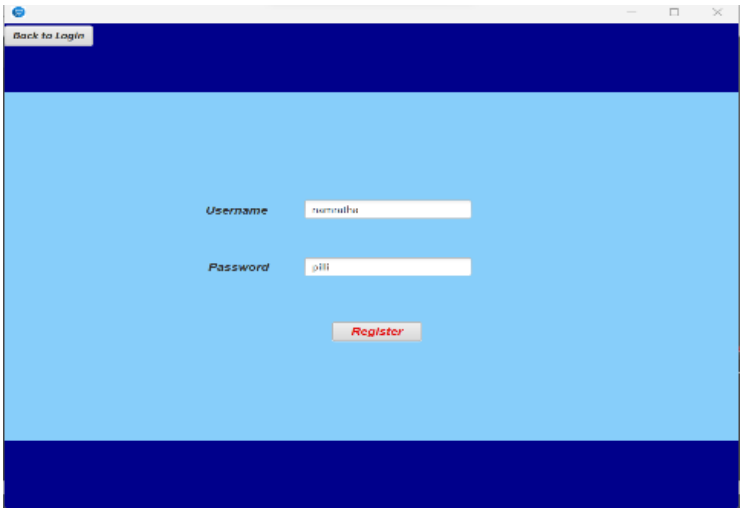

Figure 4. Register page

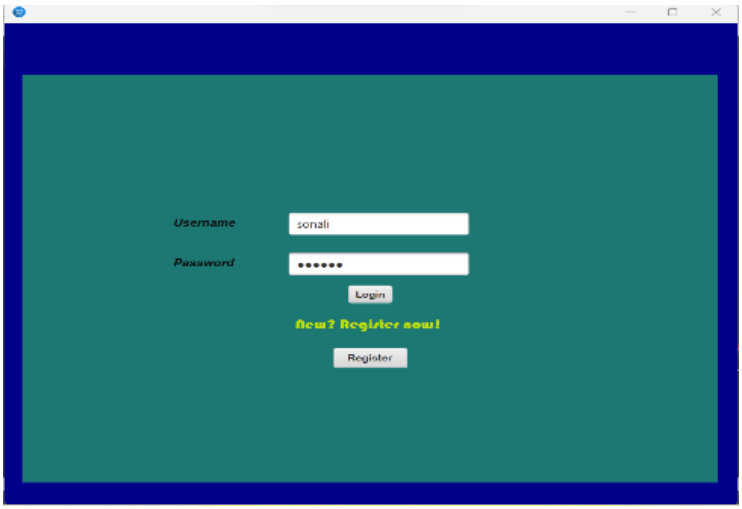

Figure 5. Login page

- When incorrect login credentials are entered a dialog box appears showing the below error message

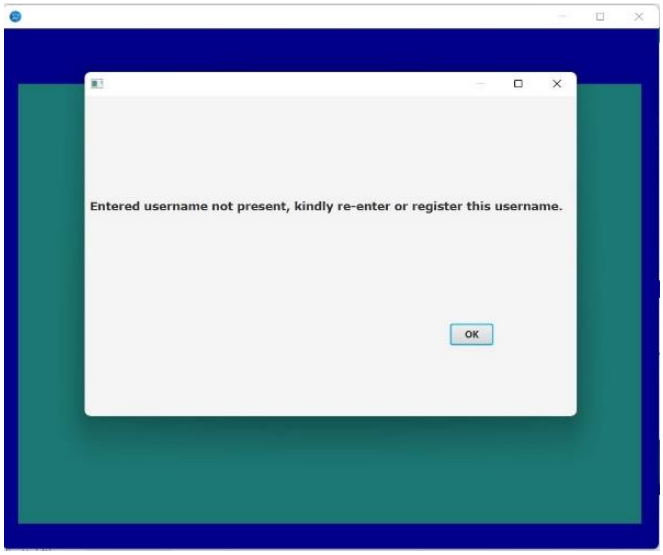

Figure 6. Error dialogue box

- Once the user successfully logins, all the product categories are displayed on the Home page.

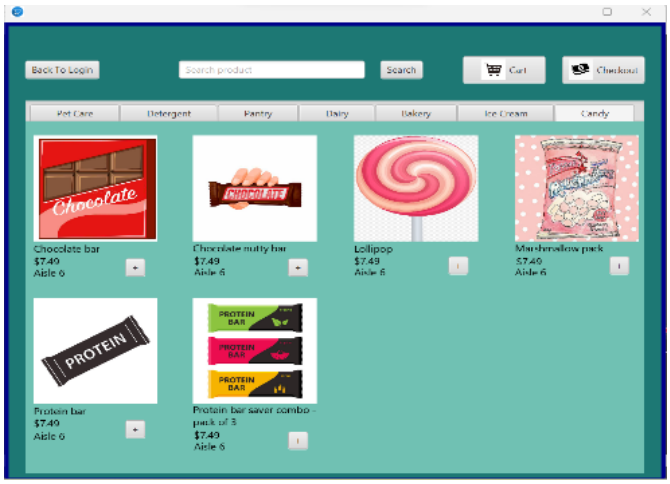

Figure 7. Categories page

- The search page displays all the products based on the search criteria entered in the search bar.

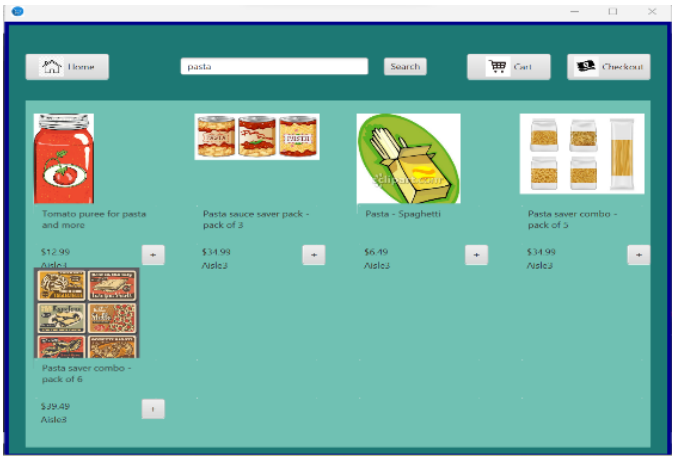

Figure 8. Search page

- The cart page has all the products consolidated based on the aisle numbers.

Figure 9. Cart page

- Once the user clicks on the checkout button, the total amount to be paid by the customer is generated by clicking on the generate bill button.

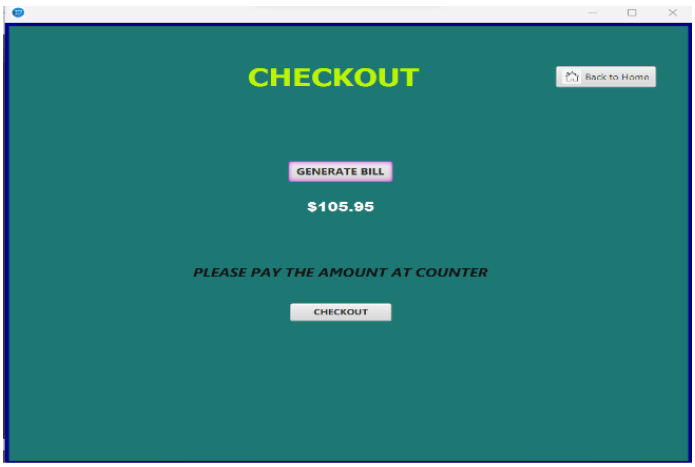

Figure 10. Checkout page

VII. Discussion (Reflection)

The aisle tracker application was successfully able to list the product with the aisle number and a search feature that helps to search required products and add to the cart. The cart section was able to fetch all the products within the consolidated aisles. One important feature of this application we achieved is it doesn't allow the user to enter a quantity of products which is no more than the quantity that are available in the store.

Login page is developed for existing users to login to the Aisle tracker application where in the Home page is displayed on login.

Register page is developed for the new users to register into the application. Additionally, messages letting the user know if they filled out all the correct details were included as validations.
Home page is developed to view all the products that are available in the store. Users can view the product details that include the aisle number, cost, and the count of the product.

Search page is developed where all the products are displayed based on the search criteria of the user that is entered in the search bar and clicks on the search button.

Cart page is developed that displays all the products added by the user with products consolidated based on the aisle number along with their price and the quantity of the product.

Checkout page is developed to view the generated bill of the products that has been added to the cart by the user and need to do the payment in the store. Validation is included to make sure the product count is updated in the backend so that the inventory is up to date.

Thank you page is developed to know the user that the order has been completed and need to be collected at the store

VIII. Conclusions and Future Work

Our key finding was that we gained increased knowledge about java concepts in depth while working on the project. We had to use TreeMap that is nested inside of other TreeMap which improved our skillset.

Benefits of using Aisle Tracker:

- Aisle tracker application save time and money of the user.
- Helps grocers reorganize the product layout to boost traffic in under visited aisles.
- Understanding where customers spend time vs where they quickly pass by helps shops decide on layout, planogram, and assortment.

The problems found but not yet explored in the project:

Our application is built only to get the Aisle details i.e., the aisle number of the product, if there is a layout of the store with aisle number then it would be much easier to get access of the products.

If your team has more time, what do you want to improve

- We can add a Layout of the store upon clicking the aisle number at the product.
- If the product is out of stock, we can get the product availability in the nearby store.
- Enable the user to optimize the total expense by recommending lower priced counterparts, if needed

IX. Job Assignment

- Sai Kashyap Cheruku - UI for Login and Registration page, Login Controller, Dialog Controller, Register Controller.
- Sonali Mondal - UI for Main page and Search page, Product class, Main class, Scene Controller classes, Cart class, Cart controller.
- Hema Sree Bojanapally- UI for Cart page, CartEntry class, Checkout controller.
- Namratha Pilli – UI for Checkout page and Thankyou page, Thank you Controller.

References

[1] [Online]. Available: https://www.walmart.com/

[2] Timothy L. Urban, 'The effect of one-way aisles on retail layout', Comput Ind Eng. 2022 Jun. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8902859/

[3] hellocodeclub(2021,November) shopping cart https://github.com/hellocodeclub/shopping-cart

[4] Hello Code(2022, May)Shopping Cart Java Project https://www.youtube.com/playlist?list=PL9LXxe1hu_BgLM9gDj0WpW1d0xbBlbWxe

[5] yaswanthrajyadiki(2015, August) Item.java https://gist.github.com/yaswanthrajyadiki/f9b2ec5fbbc1368a54de

[6] Mahmoud Hamwi(2021)JavaFX UI: Youtube Design & Dynamic GridPane https://www.youtube.com/watch?v=SITIt91SBlE

[7] Interview DOT(2015)HOW TO UPDATE A JSON ATTRIBUTE USING JSON SIMPLE JAVA https://www.youtube.com/watch?v=6A0l1nkLxcA

[8] tabnine.com https://www.tabnine.com/code/java/methods/javafx.scene.control.Button/setDisable