

SP-5 Blue - Grocery List App

Course: CS 4850 - Senior Project

Semester: Fall 2023

CCSE-Kennesaw State University

Professor Sharon Perry

Website: <https://shopshare.netlify.app/>

GitHub: <https://github.com/Senior-Project-SP5-Blue>

Team Members



Javani Wright
Team Lead



Tanvi Mistry
Technical Writer



Yasmine Samad
Software Engineer



Angel Feltus
Technical Writer

Table of Contents

| | |
|---|----|
| 1. Introduction..... | 3 |
| 1.1 Abstract | 3 |
| 1.2 Objectives | 3 |
| 1.3 Scope..... | 3 |
| 1.4 Definitions and Acronyms | 3 |
| 1.5 Assumptions..... | 3 |
| 2. System Overviews | 4 |
| 3. Design Constraints..... | 4 |
| 3.1 Product Environment | 4 |
| 4. Functional Requirements | 4 |
| 4.1 The Client | 4 |
| 4.1.1 Sign in..... | 4 |
| 4.1.2 Change/Update Personal Information..... | 5 |
| 4.1.3 Payment Processing | 5 |
| 5. Non-Functional Requirements..... | 5 |
| 5.1 Security | 5 |
| 5.2 Software Quality | 6 |
| 5.3 Usability..... | 6 |
| 5.4 Capacity | 6 |
| 6. System Architecture..... | 6 |
| 6.1 Architectural Design | 6 |
| 6.2 Decomposition | 7 |
| 6.3 Design Rationale..... | 7 |
| 7. Data Design | 7 |
| 8. Human Interface design..... | 8 |
| 8.1 Overview of User Interface | 8 |
| 9. Development Process..... | 8 |
| 9.1 Design and Development..... | 8 |
| 9.2 Complications | 9 |
| 10. Test Plan..... | 9 |
| 11. Conclusion | 10 |

1. Introduction

1.1 Abstract

ShopShare will improve the shopping experience for shoppers all over the globe. ShopShare will allow you to share a shopping list easily and effectively with friends and family with the implementation of Groups. Groups can contain between 2-10 shoppers, and each shopper within the Group will have access to a set of synchronized shopping lists. Once one shopper in the Group checks off an item on the list, the status of the item will be updated for all other shoppers in the group. ShopShare will be available on both iOS and Android devices. Both male and female customers are influenced by elements connected with ShopShare shopping, including product availability, product choice, and delivery opportunities. This kind of approach allows a simple way for customers to purchase products without physically visiting a retail location, as well as for store owners to sell products. ShopShare buying in Kennesaw, or any other location differs significantly depending on the consumer's age and the items' supply. Three recommended substitute factors—product descriptions, ShopShare delivery and substitution, and product availability—can be the main areas of focus for marketers.

1.2 Objectives

One of the key goals of ShopShare in this effort to streamline the process of setting up an online store is to make it simple for anybody to develop and design their online store without the need for technical knowledge or coding abilities. ShopShare aims to offer a user-friendly interface that is simple to use, especially for individuals who are new to online shopping. ShopShare promises to offer a full range of functions, including website hosting, payment processing, and customer assistance, that are necessary for managing an online business. ShopShare aims to assist business owners in creating a fantastic customer experience by offering a quick, dependable, secure, and user-friendly platform. Shopkeepers can encourage customer happiness and brand loyalty by participating in ShopShare, which may boost earnings.

1.3 Scope

The software product we will produce will be called ShopShare. It will be a mobile app based on a customer planner. It will let the users view their profile and upcoming category items. ShopShare is online shopping, which is the method of buying and selling goods and services through the Internet without going into a store. ShopShare is online shopping that works like in-person grocery store purchasing but within a digital environment. The objective of ShopShare is to improve this process and replace it with internet-based methods that are focused on grocery and food shopping from the comfort of your own home. Online shopping has simplified and raised the enjoyment of shopping, as there lies all the information that one can learn about a product on the internet.

1.4 Definitions and Acronyms

SRS: -Software Requirement Specification

SDS: -Software Design Specifications

Stockholder: - Person who will use the system

1.5 Assumptions

The hardware and software with the fewest requirements are assumed to be used by device users. Users of the system are therefore accustomed to the third-party software that this computer system uses.

When in use, the system expects an ongoing online connection to update it among the other members of the group.

2. System Overviews

ShopShare operates on a client-server architecture, with a mobile client application available on both Android and iOS platforms and a backend server hosted on cloud infrastructure. The mobile app serves as the user interface, allowing individuals to create, edit, and share shopping lists within designated groups. Real-time synchronization ensures that changes to shopping lists are instantly propagated among group members. The server manages user authentication, data storage, group management, and permissions, ensuring centralized and secure data management. This architecture supports scalability, security, and efficient data synchronization, making ShopShare a user-friendly and reliable collaborative shopping application.

3. Design Constraints

3.1 Product Environment

The ShopShare app will be a mobile application built using Flutter. This app builds software. This will be compatible with iOS and Android devices. ShopShare is a mobile app based on the mobile operating system of the user's device.

4. Functional Requirements

4.1 The Client

4.1.1 Sign in

4.1.1.1 Description

Using data stored in the database from when the user signed up for a membership in person, i.e., the user's email address, the user should be able to make an account which will be stored and used to sign into the application.

4.1.1.2 Stimulus/Response Sequences

For the first time after signing up at ShopShare in person, the user should set up a password for their account. The user will need to input the email address used when signing up for a member along with the newly set-up password, to sign into the app.

The user inputs their sign-in information; the system takes in that information, validates the credentials, and, if successful, gives the user access to the application. The application will not require the user to sign in on subsequent launches after initial sign-in unless the user signs out of the application or changes any of their sign-in information.

4.1.1.3 Functional Requirements

- ☐ The system must allow users to input their information.
- ☐ The system must allow users to sign in if the information is correct.
- ☐ The system must allow users to reset their password if the user decides to update/reset the password.

4.1.2 Change/Update Personal Information

4.1.2.1 Description

The user has their personal information stored in the database and that information is likely to change depending on the circumstances of the user. Therefore, the user should be able to update that information if the need arises.

4.1.2.2 Stimulus/Response Sequences

After signing up, the user can go into the app and change their personal information, and the database should be updated with the new information for the current user.

4.1.2.3 Functional Requirements

- ☐ The system must allow the user to change or update their personal information in the database.
- ☐ The system must confirm that it is the user who is responsible for the change by requesting their password.
- ☐ The system must ask the user if the changes are as intended before finalizing the change.

4.1.3 Payment Processing

4.1.3.1 Description

This feature allows customers of Shop Share to pay their membership fees through the ShopShare application.

4.1.3.2 Stimulus/Response Sequences

- ☐ Stimulus: Customer chooses shopping to purchase.
- ☐ Response: The application takes the customer to a payment screen.
- ☐ Stimulus: Customer requests to pay due balance or monthly payment due.
- ☐ Response: The application processes the payment(s) and gives the customer a receipt of the transaction.

4.1.3.3 Functional Requirements

- ☐ The application will support various payment methods, such as credit cards or online payment.
- ☐ The application will securely manage payment information.
- ☐ The application will create a receipt upon transaction completion.

5. Non-Functional Requirements

5.1 Security

The primary concern for ShopShare users is securing their personal information and login details. Users will be required to authenticate their login by entering the correct username and password; unauthorized users will not have access to any content beyond the title screen. Additionally, ShopShare will protect users with data encryption, thorough penetration, and regression testing to achieve as close to error-free code as possible, and detailed documentation on vulnerabilities to develop reinforced updates. There will also be measures put in place to ensure that only authorized users can share shopping lists; this includes a required request acceptance sent through ShopShare from one user to another. There will be

ShopShare follows a client-server architecture with a mobile client application and a dynamic backend server.

6.2 Decomposition

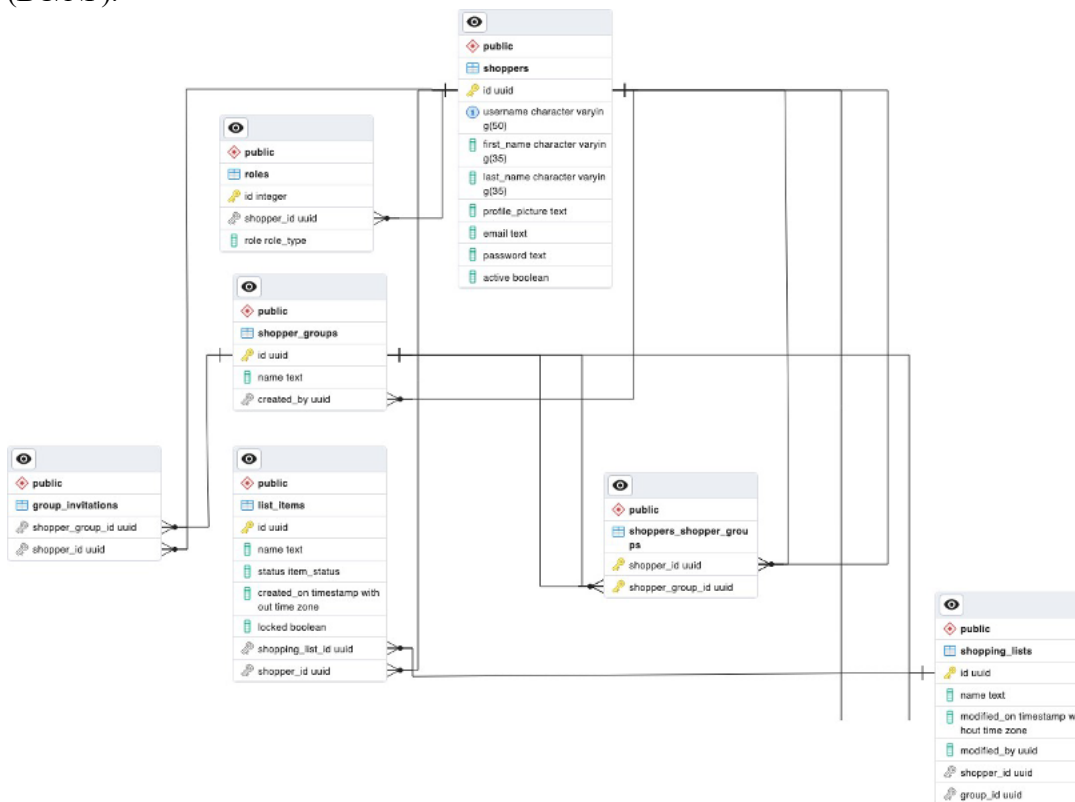
The client and server will have two distinct jobs. The mobile client will provide a user-friendly interface for creating and managing shopping lists, joining, or creating shopping groups, and accessing account information. It will utilize APIs to communicate with the backend server for data synchronization. The server is responsible for handling user authentication, data storage, real-time synchronization, and group management. It will utilize server-side scripting (Java Spring Boot) and a relational database (PostgreSQL).

6.3 Design Rationale

The client-server architecture perfectly complements the functions of ShopShare. The clients must only worry about displaying the data in a user-friendly manner, while the servers handle all the intense work behind the scenes. This separation of concerns favors users, as their devices will not be used for heavy computations. Also, all data will be stored on the server, so users' sensitive data is more secure.

7. Data Design

The data will be stored using a relational database that satisfies the Boyce-Codd Normal Form (BC/NF).



shoppers: Holds user information. This includes username, first name, last name, email, password, and other relevant information. This table is used for authentication purposes as well.

roles: Contains data regarding what roles a user has. Roles are used when determining what features a user is authorized to access.

list_items: Represents an item on a shopping list.

shopping_lists: Holds all information regarding a shopping list.

group_invitations: Used for inviting users to a shopper group. Each record represents a user's invitation to a group.

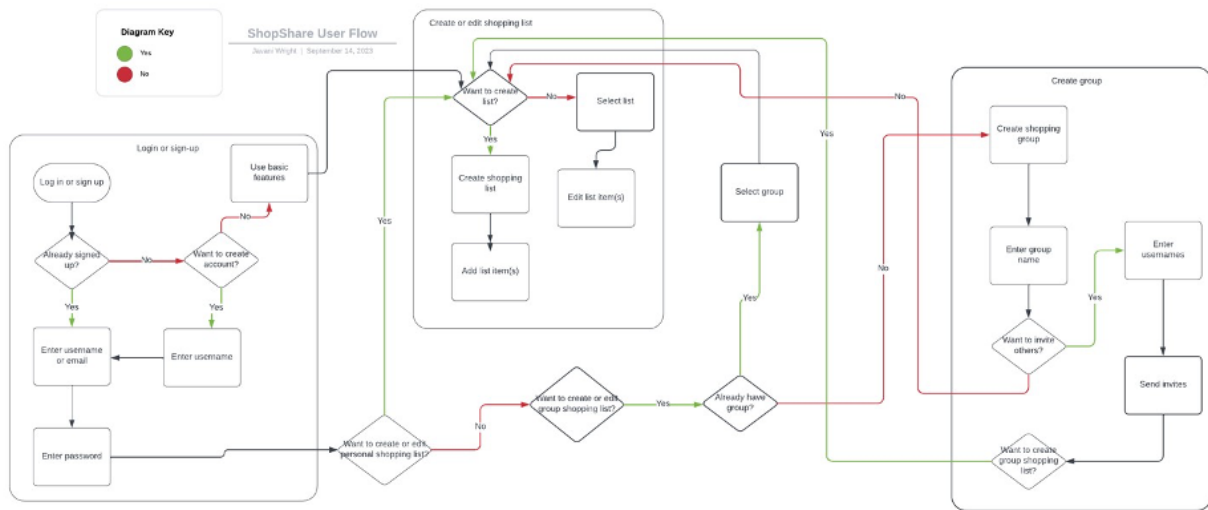
shopper_groups: Represents a group of shoppers who will all have access to the same synchronized lists.

shoppers_shopper_groups: Intersection table for the M:M relationship between shoppers and shopper_groups.

8. Human Interface design

8.1 Overview of User Interface

ShopShare boasts an intuitive and user-friendly interface that simplifies the collaborative shopping experience. Users can effortlessly create and manage multiple shopping lists, customize item details, and lock items to maintain individual preferences. The app facilitates easy group formation and invites members to collaborate seamlessly on shared lists. Real-time updates ensure that everyone in the group stays on the same page, and the platform's sleek design enhances the overall user experience, making ShopShare a convenient and efficient tool for coordinating shopping tasks with friends, family, or housemates.



9. Development Process

The development process of ShopShare included designing and implementing the back end of the system, developing the front end of the program, merging the two, and testing/addressing complications throughout the process.

9.1 Design and Development

Developers first determined what technologies, frameworks, and languages would be the most effective for the nature of ShopShare with the resources available. After much consideration, it was

decided to utilize Java Spring Boot as a backend framework, PostgreSQL as the database, and React Native for the app's functionalities and front end.

Java Spring Boot offers a straightforward and user-friendly development process so the engineers can create the product in a timely manner, with minimal training needed. There are also many built-in features that the developers took advantage of to maximize the productivity of the framework. PostgreSQL caught the eye of the team because it is an open-source asset that is also user friendly and requires minimal training. The database tool is compliant with SQL standards, which is a language and skill that a majority of the team is well-versed in. Lastly, React Native is a cost-effective resource that offers cross-platform development, which is a priority for the application to be accessible on the most widely used mobile operating system platforms. This was an effective resource to develop both the front end and some backend components.

To develop using Java Spring Boot and PostgreSQL, engineers were tasked with translating the database model to Java entities, and then create Java classes to represent the tables and their fields/entities. They were then able to utilize knowledge of the Java programming language to develop the functionalities of ShopShare and the communications between the backend, frontend, and the API. The API was set up using the REST architecture, and Postman, an API testing environment, was used to test calls made to the backend and keep track of data manipulation. The front-end development included using React Native and NPM packages. The NPM packages offered a base and foundation, and developers were not required to start from ground zero.

It was then time to merge the front-end code and the back-end code to create a functional app. The API lives in the backend of the code, so developers had to imitate these calls in the front end to call and initiate the API. Each function and button on the front end are linked to a function and/or API call. UI was developed using JavaScript and Typescript.

9.2 Complications

Some complications included the lack of resources available to the team. The engineers strived to keep the production of ShopShare as low as possible, but they were faced with issues concerning security. Also, there were issues with keeping the app on the same page although it has the capability to run on different operating systems; some functions would work on iOS but not on Android, and vice versa.

10. Test Plan

Rigorous testing and targeted updates are crucial to the overall quality and sustainability of the program. Each phase in the development process of ShopShare includes some form of testing, ensuring that all inaccuracies are addressed as soon as possible. The mobile application testing process consists of two consecutive phases that can effectively overlap with each other: developer testing and volunteer testing.

Developer testing describes targeted and comprehensive testing performed by the engineers throughout the design and implementation phase. The system was under intense scrutiny as developers validated all possible user inputs, test cases, edge cases, and the overall quality of the program. Undergoing these tests ensures all paths are accounted for, and the expected UI for each screen is populated correctly, in the intended order. Security was another aspect that was prioritized when testing the system, developers used methods to try to bypass security measures as well. Each test and any errors were documented for developers to reference and rectify.

Volunteer testing involves inviting outside volunteers to navigate through the system as if they were using it for themselves. However, they were instructed to “confuse” and ideally “break” the system to the best of their ability after conventional navigation. For example, a volunteer was instructed to input a sentence or a string in a field that calls for an integer or a long. Scenarios like this enhance error handling and allow the developers to be cautious about the endless unexpected possibilities users may produce. Each test and any errors were documented for developers to reference and rectify.


The criterion for terminating testing was based on a 95% success rate. We repeated the process of conducting tests, accounting for errors, discussing evaluations, and correcting accordingly until a minimum of 95% test case pass rate is attained. As we conducted the testing phase, however, we were also cognizant of and prioritized the efficiency in hours spent by each developer and volunteer.

Our testing and preventative process is intended to deliver a reliable and secure program that will function properly, no matter the situation. It is readily adjustable and adaptable to apply changes promptly.

11. Conclusion

ShopShare is an easy-to-use program whose simple layout makes food shopping more efficient. With a secure login function, the app guarantees security while making it simple for users to build and control personalized grocery lists. Working together, which allows registered users to collaborate on their purchasing requirements, is one of its unique characteristics. The app's objectives are rather clear: it wants to build the groundwork for a complete solution by smoothly integrating front-end and back-end features. ShopShare is integrating with PostgreSQL to accomplish an effective data archive, displaying a dedication to reliable and effective data management. All things considered, ShopShare is positioned to improve the experience of grocery shopping by fusing cutting-edge technology with intuitive functionality.


Appendix A: Mobile User Interface Design



Sign in to your account

Email


Password

 [Forgot Password?](#)


Login



Or

Don't have an account? [Register](#)

Your Lists: 


- Sunday Dinner

 Create New List ...



 

Your Lists:

- Sunday Dinner
- Sunday Brunch

 Create New List ...

Save List & Continue

Publix

←
Sunday Brunch
🔍

Veggies

Fruits

Meat

Asparagus

Zucchini

Celery

Red Onion

Yellow Onion

Cauliflower

Eggplant

Carrot

Corn

Broccoli

Mushroom

Green Onion

←
Select A Location
🔍

Within: 20 mi
change

123 Mockingbird Lane Springfield, USA
Postal Code: 12345

456 Elm Street Anytown, Imaginaria
Postal Code: 98765

789 Ocean View Drive Sunnyville, Dreamland
Postal Code: 54321

123 Mockingbird Lane Springfield, USA
Postal Code: 12345

456 Elm Street Anytown, Imaginaria
Postal Code: 98765

789 Ocean View Drive Sunnyville, Dreamland
Postal Code: 54321

123 Mockingbird Lane Springfield, USA
Postal Code: 12345

456 Elm Street Anytown, Imaginaria
Postal Code: 98765

789 Ocean View Drive Sunnyville, Dreamland
Postal Code: 54321

123 Mockingbird Lane Springfield, USA
Postal Code: 12345

456 Elm Street Anytown, Imaginaria
Postal Code: 98765

789 Ocean View Drive Sunnyville, Dreamland
Postal Code: 54321