Software Engineering
Project Report

Authors
Group 5
Front End Pair:
Storm Menges (1102107)
Sabeehah Ismail (797621)
Back End Pair:
Sergio Oliveira (1101482)
Levi Goldfein (1257360)

October 2, 2017

Shopping Route Recommender (SRRec)
Project 3

Contents

1 Project Overview

The Shopping Route Recommender (SRRec) enables a shopper to map out a route of the nearest shops in the neighbourhood to visit in order to meet their shopping needs. The system will retrieve and display a list of products that are available from the Shops' databases. The shopper/user will select their required products from the list and these will be temporarily stored in a shopping list. When the user has completed their shopping list, the system will search through the database for the relevant stores that sell the selected products. Preference will go to the shops that are close-by and sell all the required products. The system will then use the addresses of these relevant shops to calculate the shortest path, as well as the external tool, Google Maps API, to map out the route and display it to the user.

The SRRec has been implemented as a web-application, so that users may access the system on their pc or mobile device (through the web browser). Since we have implemented a prototype of the system, we had to make the following assumptions:

- We are working on a small-world example for now, where there are only a few stores available within the user's vicinity.
- The starting location is the user's home.
- Each shop follows a similar database structure and the same naming conventions.
- We will not take brands into account, but will use simple product names instead, e.g. bread, milk, eggs, etc.
- We do not take stop-overs into account. Stop-overs include coffee, drinks, lunch, etc.
- It is assumed that the user requires the route immediately after logging the products in the shopping list. The shopping list is cleared when the user closes the web-application, and will therefore be required to start a new shopping list the next time they request a recommended shopping route.
- Our system retrieves Product and Price data from Shops that have agreed to give us access to their databases.

2 Problem Statement

Our application is setup to address the problem faced by busy mothers, businessmen and the general population everyday, a lack of time. With the pace of the world increasing all the time,as well as struggling financially, people struggle to go to one shop to purchase their necessities, let alone shop around to get the best bargains and value for money.

3 Project Objectives

Our program solves the issues mentioned in the problem statement by taking in a shopping list from the user as input and then searches through our databases for the items requested and then finds the best shops whether by cheapest items or shortest route (requested by the user) and outputs a map with the route of shops the user should take to achieve their goal. Saving them much needed time and money.

4 Stakeholders and Stakeholder Descriptions

4.1 Shopper

4.1.1 Goals:

To receive a recommended route that they can follow in order to find the shops near them, that sell the required items/products that they need to shop for.

4.1.2 Summary:

The Shopper is the stakeholder who will make use of our system, the Shopping Route Recommender (SRRec/"the system"). The Shopper (user) will create their shopping list by selecting the products that they need to go shopping for, from a list of products. When the user has completed their shopping list, the system will search for the relevant shops that sell the required products. The system will make use of these shops' addresses to map out the shortest path on the Google Maps API and this route will be displayed to the Shopper to follow

4.1.3 Environment:

The SRRec is a web-application and so the Shopper can use the system wherever they have internet access.

4.2 Developer

4.2.1 Goals:

To implement a system that recommends a route for the user to follow in order to find shops near them that sell the required items that the shopper requires, in an easy, user-friendly and efficient manner.

4.2.2 Summary:

The Developer should analyse, design and implement a web-application that retrieves a list of the available products from the Shops' databases. The user must then be able to select their required products from this list, which will be logged into a shopping list that must be temporarily stored. The Developer must then implement a function that will search through

the Shops in the database for relevant stores, giving preference to Shops that are close-by and sell all the required products. The addresses of the relevant Shops must then be parsed to the external tool, Google Maps API, where a shortest path will be generated, mapped and displayed to the user. The Developer must perform appropriate testing on the system and functions, as well as perform required system maintenance and security.

4.2.3 Environment:

The Developers workplace.

4.3 Shop Outlet

4.3.1 Goals:

To gain more customers through this web-application as it recommends the store to Shoppers that are close-by, granted that the store sells the Shopper's required items.

4.3.2 Summary:

In order for the Shop Outlet to possibly appear on the web-application, the Shop Outlet must give the Developers access to their database, such as products, prices, and address. This allows the system to retrieve the list of products available to the user, and use the Shop's address to generate the shortest path for the shopping route. It is the responsibility of the Shop Outlet to keep its database updated and accessible to the Developers.

4.3.3 Environment:

The Shop Outlet's workplace/location.

5 System Requirements Specification

5.1 Product Scope

Our system is a shopping route recommender (SRRec) that maps out a route of the nearest shops in the user's area. The user will identify the items that they need to go shopping for, which could be groceries, clothing, appliances, etc. The items will be logged into a shopping list. On the user's required shopping day, they will run the SRRec and obtain a recommended route of shops to follow in a particular order to purchase the items they listed.

The SRRec will be a web-application and therefore should run on any supported web browser that has online access. The database stores each shop in the area with the items that they sell, and their respective prices.
Not only will the SRRec recommend a route of shops to purchase the user's required items, but it may also provide a route based on the shortest path for the minimal total expenses or the shortest travel time.

The benefits of the SRRec are the following:
-The user is given a recommended route of the nearest shops in the area to purchase their required items.
-If the user wants a route to purchase their items at the cheapest prices, the SRRec will provide a route based on the shortest path for the minimal total expenses.
-If the user wants a route that will allow them to purchase their items in the least amount of time, the SRRec will provide a route based on the shortest travel time.

5.1.1 Purpose

The purpose of the project is to create quality software that allows the user to enter items they require and the software will generate the shortest path to go and buy the items they require. Our main goal is to produce quality software that is relevant and provides an accurate solution to all users at any given time

5.1.2 Product Overview

Our product is a software product in which we will allow the user to input items that they require and we will generate a route which will take the shortest amount of time, Our product is most likely to target working class people who don't have much time to go to the shops. Our product will also include various help buttons and prompts to guide users on how to use the software

5.1.3 Product Function

Our product will have one main function which is allowing the user to input items that they require and generating a route which will take the shortest amount of time, we would also like to add a lowest cost route which will take into account the petrol spend as well as the prices of the items bought

5.1.4 User Classes and Characteristics

Our user class will be a general user with focus on people who have heavy time constraints with regards to work

5.1.5 Operating Environment

The operating environment will be majority of browsers as we will integrate support for most big browsers, the coding environment will be Visual Studio and the Language will be C-Sharp

5.1.6 Assumptions and Dependencies

Our assumptions and dependencies include the fact that we will receive databases of items and their various prices from various supermarkets around Gauteng. We also assume that all price changes will be updated regularly by the supermarkets and we will receive those

changes as they happen. To simulate this we will create our own scaled down databases which will contain a few products and their respective prices

5.2 Intended User and User Goals

Our intended user is anyone who needs to go to a store or various stores to do their shopping. The system is aimed to recommend a shopping route for the user to follow, based on the items they require. This route could be the shortest path for the minimal total expenses or the shortest travel time to purchase all the items.

5.2.1 Key High Level Goals and Problems

The user may not know where to go in order to purchase all the items they need. The user may also require a route that allows them to buy the items at the cheapest prices, or a route that takes the least amount of time. With this system, not only will the user be given a recommended route to shop for their items but it may also provide a route for minimal expenses or shortest travel time.

5.2.2 User Level Goals

To enter the items that they need to go shopping for and thereafter obtain a recommended shopping route to purchase these items for the minimal total expenses or the shortest travel time.

5.2.3 Environment

Either at home, work, or anywhere with online access.

5.3 Constraints

5.3.1 Legal Issues

We may come across issues where shops may not want to share their databases (such as items and prices) with us. Therefore, if we populate our system's database with such information, we may incur legal ramifications due to breach of confidentiality and incorrect listings of data.

5.3.2 Implementation and Software

If we do come across issues whereby we cannot use a shop's database, our system may not make use of it in the Path Finding Algorithm and therefore the shop won't be listed on the route. This could lead to a situation where the shop may be in the shortest path but because we cannot make use of it, the user may not obtain an accurate route based on their surroundings or shopping requirements.

We are limited by our implementation time as we are students with our own time constraints.

Since we plan to make use of Google Maps API for mapping the shopping routes, we are limited to the stores that exist on Google Maps.

Our system's database is limited to the regular pricing of items as we do not take into account item sales that can occur at shops. This also leads to a situation where our system generates a path that may not be accurate according to real-time changes.

The user is constrained to using a supported browser in order to make use of our web-application.

5.3.3 Hardware

Since our system is an online web-application, it is constrained by the response time and performance of the server and internet connection.

5.4 Pricing and Implementation Costs

The possible costs that we may incur are:

-A domain for the web-application.
R197 once-off (.co.za)
https://www.afrihost.com/site/product/domain_registration

-A server to host the web-application.
R4850($358) upfront
(Standard 1 year term: t2.micro for Windows with SQL Web in EU-frankfurt)
https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/

-Development time and maintenance.
Subject to working hours and consulting rates of the development team.

5.5 Quality

5.5.1 Correctness

The user will create an account if they are not registered on the system. If the user has an account then they will log in. The user then enters the items they require and will run the SRRec when they want to go shopping.
When the user runs the software, the SRRec must take the user's list of items, compare it to the system's database, run the required Path Finding Algorithm (shortest path for the minimal total expenses or the shortest travel time) and return the mapped-out route to the user.

5.5.2 Reliability and Efficiency

Because the SRRec is a web-application, the reliability of the system is reliant on the performance and response time of the server and internet connection.

5.5.3 Integrity

Since the SRRec generates the required shortest paths by making use of the data from various shops' databases, we need to ensure that these databases are secure to protect each store's confidentiality.
We should make use of SSL, TSL, and HTTPS to ensure security within our system's online connections and data communications.

5.5.4 Usability and Portability

Since we do not plan on including a help function (for novice users) within our web-application, we want to implement the user interface to be as simple and user friendly as we can make it, to promote ease-of-use over ease-of-learning.
We will make use of HCI principles and keep consistency throughout the web-application's interfaces. This will allow users to become comfortable and familiar with using the system.
Since the SRRec is a online web-application, it can be used on all supporting web browsers (PC and mobile devices) and this allows the user to run the SRRec wherever they may be, granted they have good internet connection.

5.5.5 Maintainability, Flexibility and Re-usability

Throughout implementation, we will develop the back-end and front-end software in a logical and consistent way so that we may easily make software changes if needed and if a problem occurs, we may locate and fix the issue easily. This also allows us to adequately reuse some parts of the software.

5.5.6 Testability

To ensure that our system is implemented correctly, we will make use of interface, validation and database testing.

Interface testing will be used to check that all UI components (such as buttons) work and trigger their respective processes such as passing data from the forms to the database, changing pages, etc.

Validation testing will be used to check and ensure that all data entered and passed from the interface forms to the database is of correct data type and format.

Database testing will be used to check and ensure that data is captured and stored correctly within their respective data stores and attributes. This also ensures that our Path

Finding algorithm retrieves and makes use of correct data to provide an accurate route to the user.

Algorithm testing will be used to check and ensure that our Path Finding algorithm works and produces a correct route that is also mapped-out for the user.

5.6 External Interface Requirements

5.6.1 User Interfaces

The design of our user interface plans to include both elegance and functionality, the GUI design is planned to suit all users and allow for users of all ages to easily engage with the application. We plan to ensure that our application has ease of navigation and a user interface that is both stylish and current.

5.6.2 Hardware Interfaces

Due to the sheet scale of the project the need for dedicated devices and support hardware is not necessary at this point in time, but as the project start to gain popularity amongst users it could call for the use of large scale severs to sore all user information, that would need to be based around the world to ensure that all users from all corners of the globe get the same quality software.

5.6.3 Software Interfaces

Our software plans to make use of MVC(Model-View-Controller) architectural pattern which separates an application into three main components:
- Models : Models objects are the part of the application that implement the logic for the applications data domain, they often retrieve and store model state in a database.
- Views: Views are the components that display the applications user interface
- Controllers: Controllers are the components that handle user interaction, work with the model , and ultimately select a view to render that display. The controller handles and responds to user input and interaction.

Our database will be developed on a SQL server as it works well with the MVC framework as well as C-Sharp, has a huge amount of functionality and allows for scalability and growth of the software if needed.

5.7 System Features

5.7.1 Minimum Shopping Expenses 1

Description and Priority

Priority: Very High
The user inputs their shopping list of, say n, items the program then outputs a list of n or less (if 2 or more items are available at the same shop) shops that supply the items on the

shopping list for the cheapest amount possible. The output will also show which shopping list items can be purchased at which shop. This feature gives the user no indication the order in which listed shops should be visited.

Stimulus and Response Sequence

-User types in shopping list
-User presses button labelled "Shopping Route Recommender?"
-System outputs table of shops and products

Functional Requirements

The requirements for this feature are a database containing shops (for example: PNP,Checkers,game, DionWired, Mr Price, Woolworth) that stock items an average shopper may need, for example: clothing, groceries, appliances etc, and specific items stocked in said shops with their respective prices. The product should also respond to invalid inputs and errors with an error message. Invalid inputs include items entered that are not found in the database.

5.7.2 Minimum Shopping Expenses 2

Description and Priority

Priority: High
This feature is basically an extension of Minimum Shopping Expenses 1 (feature 2.1), but incorporates the vital goal of the product: a route to take. The user inputs their shopping list of, n, items and their current location. The program then outputs an **ordered list** of n or less shops to visit and which items are available to purchase at each shop. The shops listed are to provide the user the cheapest possible shopping expenses and are ordered to provide the shortest route while minimum price is prioritised. The route length is irrelevant.

Map: A further extension of this feature will be to show the shops on a map with directions of the route.

Stimulus and Response Sequence

-User types in shopping list
-User presses button labelled "?Shopping Route Recommender?"
-System outputs table of shops and products
-System outputs map with directions

Functional Requirements

The basic requirements are the same as Minimum Shopping Expenses 1 with the added information of each shops GPS co-ordinates. As well as the added software capabilities of resolving precise locations and mapping.

5.7.3 Minimum Travel Time

Description and Priority

Priority: High
This feature uses the same input information as Minimum Shopping Expenses 2 (feature 2.2) (shopping list and current location) and outputs an ordered list of shops with preference to a short route length and travel time over actual shopping expenses.

Map: A further extension of this feature will be to show the shops on a map with directions of the route.

Stimulus and Response Sequence

-User types in shopping list
-User presses button labelled "?Shopping Route Recommender?"
-System outputs table of shops and products
-System outputs map with directions

Functional Requirements

Will require software to calculate shortest route and perhaps traffic awareness and avoidance protocols.

5.7.4 Minimum Total Expenses

Description and Priority

Priority: High
This feature requires the same input as features 2.2 and 2.3 and outputs an ordered list of shops that takes into account both shopping expenses and travelling time and expenses and minimises both.

Map: A further extension of this feature will be to show the shops on a map with directions of the route.

Stimulus and Response Sequence

-User types in shopping list
-User presses button labelled "?Shopping Route Recommender?"
-System outputs table of shops and products
-System outputs map with directions

Functional Requirements

This feature requires up-to-date information on travelling expenses (from the AAA or the likes). As well as the functional requirements listed in Features 2.2 and 2.3.

5.7.5 Account/Login

Description and Priority

Priority: Medium
This feature will enable users to sign-up and create an account which enables them to store pertinent information, such as:

- favourite shopping list
- favourite route or route type e.g.: Minimum Shopping Expenses
- home/work/starting location

for future use.
The inputs for this feature would be a valid user-name and password and some form of verification e.g.: email address or cellphone number.

Stimulus and Response Sequence

-User enters Sign-Up page to create an account
-System creates the necessary databases for the user
-User enters preferred information e.g.: favourite shopping list
-System saves information in correct tables in database

Functional Requirements

The requirements for this feature would extend to added databases to store user-name password keys, saved shopping lists and routes and saved locations. We would also have to provide security to ensure none of the users sensitive data is accessible.

5.7.6 Item Success Tracking

Description and Priority

Priority:Low
This feature would be an add-on to all the above route features and would remind you which items to purchase at the current shop and then track the success of your stop at the specific shop. This would enable you to recalculate your current route based on whether you found the item at this shop or not. As well as notify other users of the availability of an item at a specific shop.

Stimulus and Response Sequence

-When the user arrives at a shop the system outputs which items off the users shopping list are intended to purchased at this shop.
-User inputs whether or not they were successful.

Functional Requirements

This feature would require constant access to the users GPS and perhaps some added input from the user.

5.8 Security Requirements

This product has several security and privacy issues that must be addressed. Concerning user authentication the product will require a valid email address to create an account, then they will require a unique user-name and password pair to ensure no cross contamination of private user information and to prevent access to the users sensitive and personal stored information, such as home or work address, email address, password and frequent or favourite routes. The developers will also have to provide guarantees to participating shops that provide classified and valuable databases of their stock prices that no malicious user or third party will be able to obtain this data beyond the intended scope of the product.

6 Use Case Set and Descriptions

Use Case	Description	Actors
Create Shop	This use case creates a new Shop in the shop datastore,in the system's database	Developer
Read Shop	This use case allows the user to search/view a Shop from the Shop datastore, in the system's database	Developer
Update Shop	This use case updates details/attribute values of a Shop from the Shop datastore, in the system's database.	Developer , Shop outlet (external)
Delete Shop	This use case deletes a Shop from the Shop datastore, in the system's database.	Developer
Continued on next page		

Create Product	This use case creates a new Product in the Product datastore, in the system's database.	Shop outlet (external)
Read Product	This use case allows the user to search/view a Product from the Product datastore, in the system's database.	Developer, Shop outlet(external), Shopper
Update Product	This use case updates details/attribute values of a Product from the Product datastore, in the system's database.	Shop outlet (external)
Delete Product	This use case deletes a Product from the Product datastore, in the system's database.	Shop outlet (external)
Create Price	This use case creates a new Price in the Price datastore, in the system's database.	Shop outlet (external)
Read Price	This use case allows the user to search/view a Price from the Price datastore, in the system's database.	Developer, Shop outlet(external)
Update Price	This use case updates details/attribute values of a Price from the Price datastore, in the system's database.	Shop outlet (external)
Delete Price	This use case deletes a Price from the Price datastore, in the system's database.	Shop outlet (external)
Create Shopping List	This use case creates a Shopping List that temporarily stores the selected items/products that the user wants as per their shopping needs.	Shopper
Generate Shortest Path	This use case runs the shortest path function which calculates, maps and displays a recommended shopping route to the user, based on the items/products they require.	Developer, Shopper, Google Maps API(external)

7 Design and Domain Models (UML)

For our system, Shopping Route Recommender (SRRec), we constructed the following models as we found these appropriate for the implementation of our web-application.

- Entity-Relationship Diagram
- Data Flow Diagram
- CRC Cards
- Class Diagram
- Sequence Diagram
- Use Case Diagram

We did not model any State Machine diagrams, because our database does not store the status/state of any entities. Our system retrieves Product and Price data from shops that have agreed to give us access to their databases.
Only for the sake of producing a prototype of our system at this stage, we have decided to store the database internally for now. Our web-application thus stores the data internally in Visual Studio.

7.1 Entity-Relationship and Class Diagram

The Entity-Relationship Diagram (ERD) and Class Diagram were constructed according to the following business rules:

- One Shop has zero-to-many Prices.
  (Beacuse a Shop can sell zero-to-many Products)
- One Price belongs to one Shop
- One Product has one-to-many Prices.
  (Because Shops can sell Products at different Prices)
- One Price belongs to one Product.

Class:Shop
Primary Key: Shop_Name

Class: Price
Primary Key: Price
Foreign Key: Shop_Name and Product_Name

Class:Product
Primary Key: Product_Name

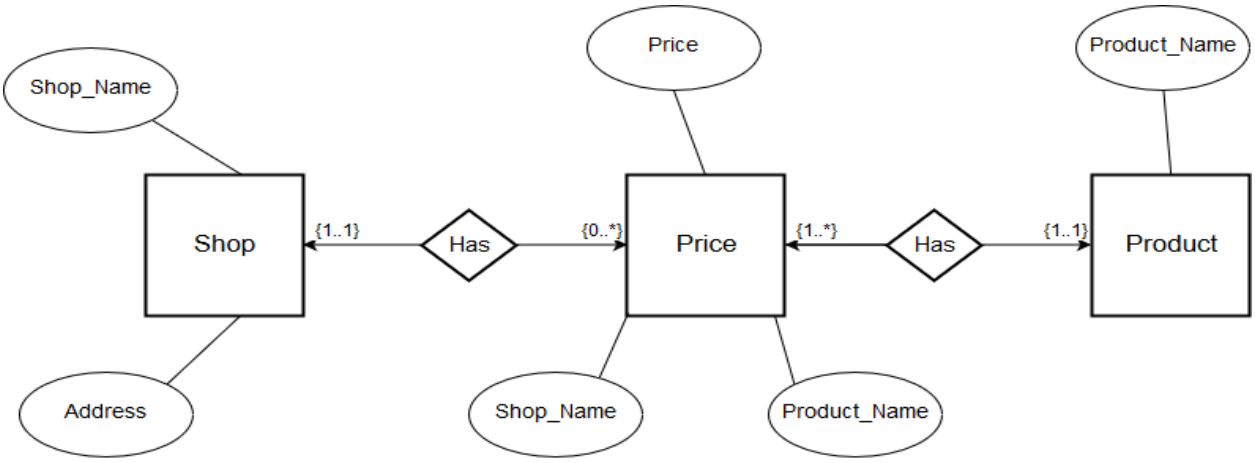

Figure 1: SRRec Entity-Relationship Diagram

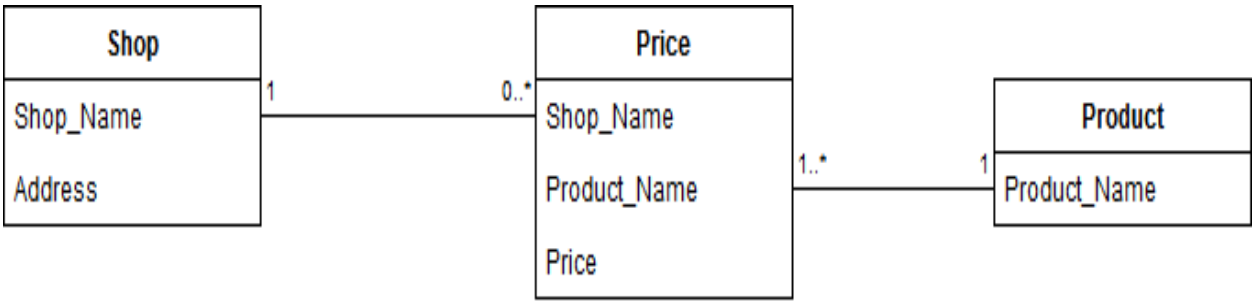

Figure 2: SRRec UML Class Diagram:association class as a full class(Price)

7.2 Data Flow Diagram(DFD)

The process begins when the Shopper (user) requests for a recommended shopping route. The system will retrieve a list of product names from the Product datastore and display them for the user to select their required products. These selected products will be logged into the shopping list, which is stored temporarily on the web-application. When the user has completed their shopping list, the system will search for shops from the Shop datastore that sell the required products. These relevant shop addresses will then be parsed to the "Generate Shortest Path" function where the web-application will make use of Google Maps API (external tool) to map the recommended route and thus display it to the user.

Note: It is assumed that the user requires the route immediately after logging the products in the shopping list.

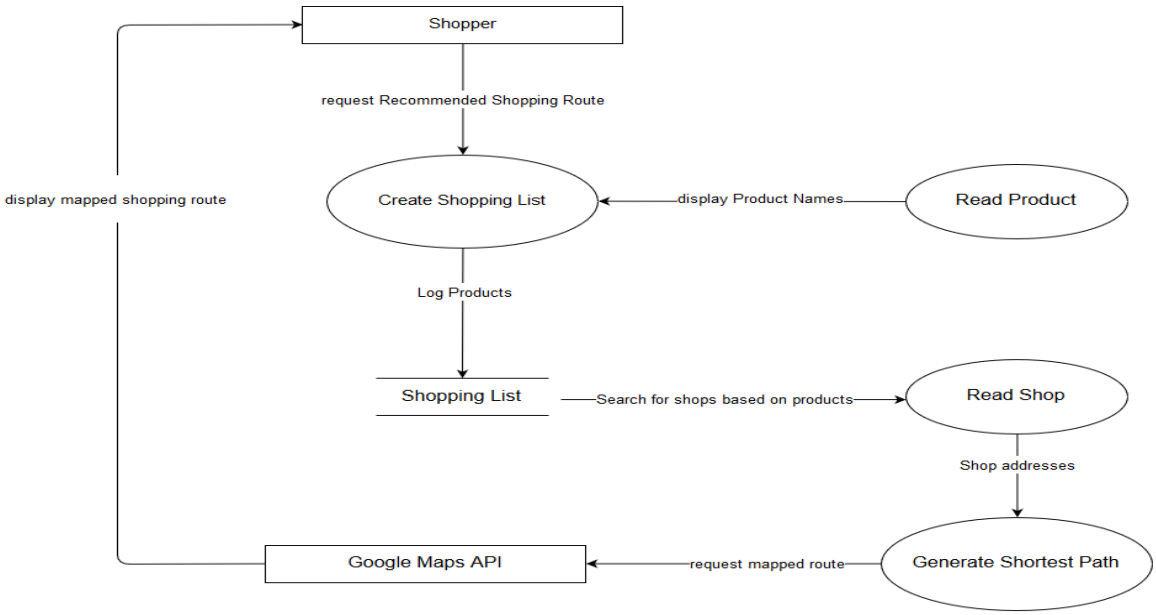

Figure 3: SRRec Data Flow Diagram

7.3 Class-Responsibility-Collaborators (CRC) Cards

We decided to make use of CRC cards, because they are helpful for early analysis in software development. These cards help identify/specify system and design components, quickly and informally.

Class Shop	
Responsibility	**Collaborators**
• Contains the Shop's name and address that will be used to generate a shortest path (with Google Maps API) for the shopper (user) to follow a recommended shopping route.	• Price

Figure 4: SRRec Shop CRC

Class	
Price	
Responsibility	**Collaborators**
• Contains the price of a product and which shop it belongs to.	• Shop • Product

Figure 5: SRRec Price CRC

Class	
Product	
Responsibility	**Collaborators**
• Contains the product's name (e.g. bread/milk/eggs) for the shopper to select as per their shopping needs.	• Price

Figure 6: SRRec Product CRC

7.4 Sequence Diagram

The SRRec sequence diagram follows the same logic process/flow as the data flow diagram (DFD).
The Shopper (user) requests for a recommended shopping route. The process will then start and the system will request a list of product names from the Product datastore. These product names will be displayed for the user to select their required products. These selected products will be logged into the shopping list, which is stored temporarily on the web-application. When the user has completed their shopping list, the system will search for shops from the Shop datastore that sell the required products. These relevant shop addresses will then be parsed to the Google Maps API (external tool) in order to map the

recommended route and thus display it to the user.

Note: It is assumed that the user requires the route immediately after logging the products in the shopping list. The shopping list is cleared when the user closes the web-application, and will therefore be required to start a new shopping list the next time they request a recommended shopping route.

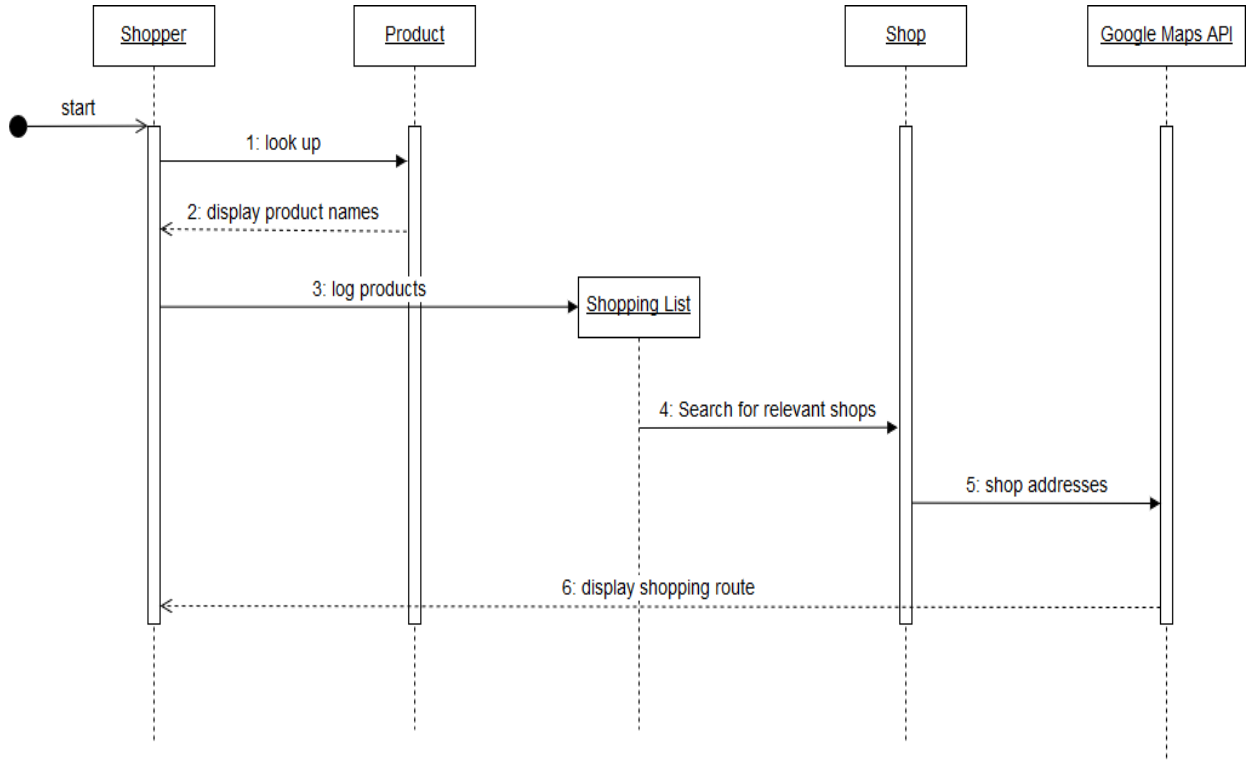

Figure 7: SRRec Sequence Diagram

7.5 Use Case Diagram

The use case diagram provides an overview of the set of use cases and the actors that interact with them.
Our system has four actors: The Shopper (user), Developer and two external actors, the Shop Outlet and Google Maps API.
Because our system makes use of Product and Price data that is retrieved from their respective Shop databases, the Shop Outlet is responsible for providing us with updated and correct data.
The Developer is responsible for the maintenance of the system's functional processes as well as which Shops are added or removed from the system.
The Shopper (user) will view the products available when creating and logging their shopping list,and thus request a recommended shopping route/shortest path.
The external actor, Google Maps API, will be the external tool used to generate the shortest

path by mapping the relevant shops and displaying the route to the user.

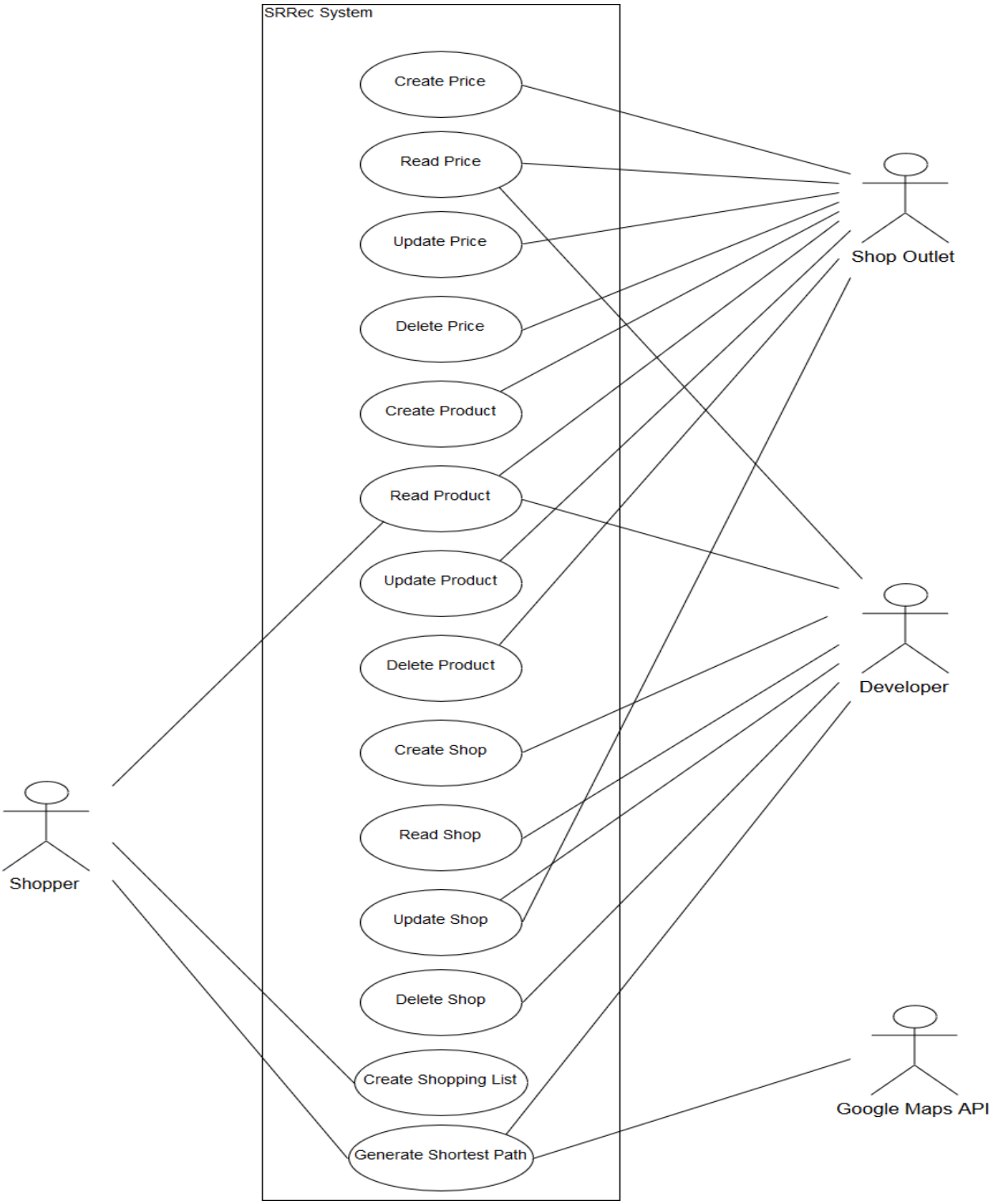

Figure 8: SRRec Use Case Diagram

8 Sprint Planning

8.1 Sprint 1: 24 Aug-31 Aug

Story Title	**Team**
Complete Lab1	Everyone
Design the database	Sergio and Levi
Design the front-end user interface	Storm and Sabeehah

8.2 Sprint 2: 31 Aug-7 Sep

Story Title	**Team**
Keep up to date with documentation	Everyone
Complete system requirements documentation	Everyone
Build database	Sergio and Levi
Implement user creates shopping list	Sabeehah and Storm
Developer adds new shop	Sergio and Levi
System reads info from shops datastore	Sergio and Levi
System fetches product from database	Sergio and Levi
User sees products on user-interface (screen)	Storm and Sabeehah
User clicks "About Us" button redirected to "Description" and "Contact Us"	Storm and Sabeehah
User clicks "Help" button redirected to "How to Use" and FAQ"	Sabeehah and Storm

8.3 Sprint 3: 7 Sep-21 Sep

Story Title	**Team**
User clicks "Description" button is shown description of company	Levi
User clicks "Contact Us" button is shown the company's contact information	Sabeehah
User clicks "How to Use" button is shown instruction on how to use application	Storm
User clicks "FAQ" button is shown a list of frequently asked questions with answers	Sergio
System reads price from database	Storm and Sabeehah
Generate shortest path	Storm and Sergio
Complete documentation	Everyone
Edit (spelling, grammar and layout of) document	Sabeehah and Levi

9 Sprint Retrospective

9.1 Sprint 1: 24 Aug-31 Aug

What did we do well?

- Team was quick to identify the pros and cons of using different architectures
- Each member researched their relevant architectures in great depth to ensure that the correct one was chosen for the problem

What could we have done better?

- Communication between team members needs to improve
- Spent less time of this sprint as too much time was allocated for research

What should we start doing?

- Make greater use of our KanBan scrum board to show the scrum master we are on schedule

9.2 Sprint 2: 31 Aug-7 Sep

underlineWhat did we do well?

- Design was agreed upon quickly so as to avoid excess back and forth
- Team members took upon the tasks they have strengths in
- Use of the Scrum board was better

What could we have done better?

- Drawing up designs took too much time
- Daily stand-ups all started late
- Communications in daily stand-ups needs to improve

What should we start doing?

- Introduce a design standard through out the project
- Make use of team Whatsapp group for Q&A

What should we keep doing?

- Assigning tasks on the scrum board to ourselves so members know who's working on what and how we're progressing

9.3 Sprint 3: 7 Sep-21 Sep

What did we do well?

- Tasked were assigned to members who wanted to work on those tasks and that played to their strengths
- Communication on team Whatsapp group was greatly improved
- Daily stand-ups took place on time
- Tasks were completed in a timeous manner

What could we have done better?

- Backups of all code and data
- Could have created our database in 3rd Normal Form rather than having to normalise it
- Instituted a naming standard
- Delegate tasks between the team better

What should we start doing?

- Pushing code to GITHUB
- Implementing a naming standard
- Do code reviews
- Testing

What should we keep doing?

- Keep assigning tasks on KanBan
- Q&A

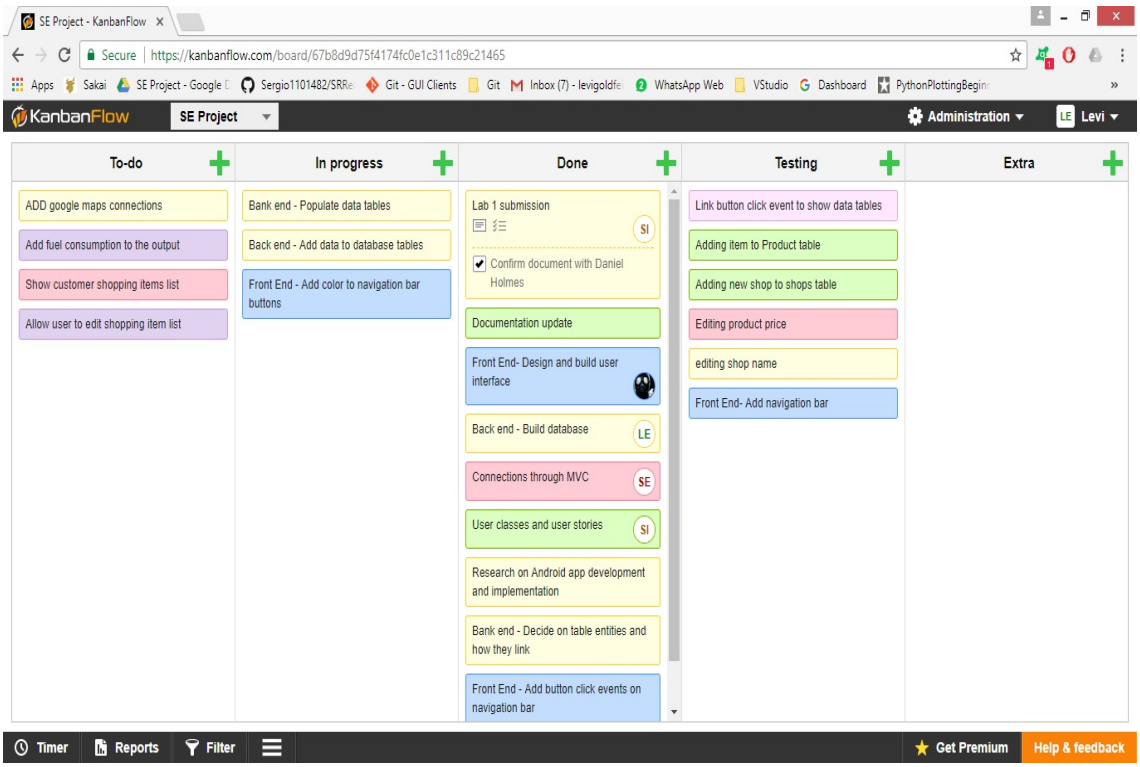

Figure 9: An example of our Sprint Board.

10 How to compile code and start modules

N.B The following software packages are required to compile and view the code

- Local DB
- SQL Server
- Visual Studio

After cloning the program from https://github.com/Sergio1101482/SRRec.git
Step 1: Open Visual Studio
Step 2: Click **File**, then **Open**, then**Project/Solution**
Step 3: Navigate to **InterfaceSE** file (which will be located in the cloned folder) and open it, then double click on **Interface.sln**
Step 4: Once loaded, click **Build** in the navigation bar, then click **Build Solution** and wait for the build to complete
Step 5: Run the program by clicking the **Run** button (the **GREEN** play button in navigation bar)

11 Testing

11.1 Back-End Testing

Back-end Tests	
TEST	**PASS/FAIL**
Retrieving entries from database	PASS
Shortest path calculation	PASS
Adding new product	PASS
Adding new shop	PASS
Delete product	PASS
Delete shop	PASS
Edit shop name	PASS
Edit price	PASS
Edit shop address	PASS
Edit product name	PASS

11.2 Front-end Testing

Front-end Tests	
TEST	**PASS/FAIL**
Click button "Shopping Route Recommender"shows "Shortest Path" button and "Lowest Cost" button	PASS
Click "Shortest Path" button shows table of products on screen	PASS
Click "Lowest Cost" shows "Functionality to come with update" message	PASS
Click on products on tables makes them grey out	PASS
Click on products on tables adds them to selected list	PASS
Click "Done" button displays shortest path on screen	PASS
Click "About Us" button shows "Description" and "Contact Us" buttons	PASS
Click "Description" button shows company description on screen	PASS
Click "Contact Us" button shows company contact information	PASS
Click "Help" button shows "FAQ" and "How to Use" buttons	PASS
Click "FAQ" button shows all frequently asked questions	PASS
Click "How to Use" button show How to make use of the application screen	PASS
Click "Settings" button shows "Customise" and "Account Settings" buttons	PASS
Click "Customise" shows "Functionality to come with update" message	PASS
Click "Account Settings" shows "Functionality to come with update" message	PASS