UNIVERSITY OF WITWATERSRAND, DEPARTMENT OF ELECTRICAL ENGINEERING

# ELEN4009 Software Engineering
# Shopping Route Recommender

April 10, 2016


**Front-End Pair:** Ronen Freeman (386910) and Luka Cakic (671913)

**Back-End Pair:** Devin Taylor (603956) and Matthew Marsden (609293)

**Abstract:** The design, implementation and testing of a prototype Shopping Route Recommender web-based application is presented. The document covers all aspects of design that the development team encountered, these aspects are detailed in the provided software requirement specification and the design document. The design aspect followed the principles of the SCRUM agile method. The final prototype is separated according to the model-view-controller design pattern, where the model consists of the relational database management system, PostgreSQL, and the view conists of a web interface. The prototype presented consists of the following functionality: the ability to add user accounts, the ability to login under the specific user accounts, the ability to add and remove items from a shopping list and the ability to generate a map, depicting the cheapest route, in conjunction with the corresponding directions. The application utilises the capabilities of the Google Maps API in order to display the route and corresponding directions. Despite the functionality present in the prototype, there are still features that are required to be added to the application. Such an example would be to allow the user to select the preferred means of route optimisation as opposed to defaulting to the cheapest route.

**Key words:** CSS, HTML, Google Maps, PHP5, PostgreSQL, Tables, User interface, Web application

# 1 INTRODUCTION

## *1.1 Problem Statement*

Shopping can often prove to be tedious and frustrating when a customer arrives at a shopping destination to find that the store is out of stock or does not stock the particular items of interest. As a result of this the customer is now forced to seek out alternative stores, which adds complication and frustration to an already stressful lifestyle [1]. Even then, the alternative stores may pose the same problem. These issues are further enhanced when the customer has made a shopping list with multiple products. There is clearly a need for a more efficient way of purchasing a list of shopping desirables.

The solution is to provide a user with a web application that structures a customers shopping experience based on user desired optimisations. The application maps out an optimised route based on reducing the travel time between stores, creating a shortest route between destinations and suggesting the stores with the cheapest product prices. Either optimisation will greatly assist a shopper with planning their weekly activities.

The purpose of this project is to therefore provide a web application that allows a user to enter their shopping list and returns to the user a map with the ideal route to the nearest stores that will fulfil their needs.

## *1.2 Project Objectives*

The project objectives are listed in detail below:

- Design a web application that allows a user to base their daily shopping on an optimised route, the optimisations being:
  - shortest travel time to meet all purchases of shopping list items
  - shortest route for minimum total expenses
  - shortest route to satisfy all purchases on the shopping list
- Generate a Google maps route that reveals the user's optimised shopping route [2]
- Implement a structurally well designed back-end data base that contains data related to various shops/stores, their locations, various products and their prices.
- Ensure the web application is user friendly and has an exceptional user experience

## *1.3 Stakeholders*

*1.3.1 Users* The proposed application is targeted at the general public. This is primarily due to the fact that nearly all member of the general public are required to shop in some capacity, be it for either items of need or items of desire. By targeting the general public it also expands the potential client base ensuring the greatest possible success of the product.

*1.3.2 Developers* The developers consists of a well rounded group of engineers. Each member of the development group displays a strength in a particular field required for the successful design of the application. These skills are based primarily on experience and include the following: web development, user experience design, database configuration and database interfacing.

*1.3.3 Project Manager* The project manager for the project has no formal training in the field of project management. Despite this, this member is often indirectly placed in positions similar to that of a project manager. As a result, the team was confident in the chosen project manager's abilities to coordinate the development team efficiently.

# 2 SOFTWARE REQUIREMENT SPECIFICATION

## *2.1 Introduction*

*2.1.1 Purpose* This document details the Software Requirements Specification (SRS) for the Shopping Route Recommender (SRRec) web application. The document also follows the IEEE standard for SRS documents [3].

Shopping can often prove to be tedious and frustrating when you find out that the shop you are at does not stock or is out of stock of certain products that are on your shopping list. One then needs to go to other shops in the hopes that those shops will have the items not yet ticked off ones shopping list. Even then, the next shop may not have the items and you have to keep going from shop to shop until you finally get all the items needed. There is clearly a need for a more efficient way of finding all the items on your shopping list in the shortest travel time, via shortest route or via the most cost effective route.

The purpose of this project is to provide a customer with a user interactive web application that allows a customer to create a shopping list of items and in return generates the ideal route the customer should travel to the stores of interest.

*2.1.2 Document Conventions* This document is intended to accompany the SRRec software and should be updated with each update to the web application. This will ensure that the document remains relevant and useful.

*2.1.3 Intended Audience and Reading Suggestions* This SRS document is intended for:

- Programmers that want to understand the outlines of how the SRRec web application works and the way in which the software has been implemented.
- Project testers to use in order to better their testing strategies since some bugs are easier to find by using an SRS document.
- End users of this web application who would like to read about how the SRRec can help them or what it is capable of doing.

*2.1.4 Project Scope* SRRec is a web application that is capable of providing a route to all the nearest shops so as to fulfil a users entire shopping list while either following the shortest route, the route with the shortest travel time or the route that will result in minimal total shopping expenses. The shopper can continuously add items to their list and these items are logged to their respective list. When the shopper finally runs the SRRec he/she will obtain their user-determined recommended route of shops to visit in a particular order according to items on their list. The application provides a Graphical User Interface (GUI) with embedded Google Maps to illustrate this generated route along with directions.

We assume the following:

- There is already a map of all shopping malls and shops that sell various items that a shopper could add to their list.
- Each shop has a database of the items that they stock and their respective prices.
- The price of each item is a simple Rand value or indicated by Rand/kilogram, etc.
- The SRRec runs on a website with access to the databases of all the shops.
- Google maps is available on the website and shows the route from one shopping mall or shop to the next.

The shopper is able to select one of three types of recommendation:

- shortest route.
- shortest route for minimal total shopping expenses.
- route with shortest travel time.

The shopper will also be able to change their selected recommendation after visiting some shops and eliminating some items from their list. The SRRec can also incorporate stop-overs for coffee, drinks, lunch, etc. At a later stage, the shopper may even be able to provide a price range that they are willing to pay for each item.

*2.1.5 Definitions and Acronyms*

- Apache: A web server
- CSS - Cascading Style Sheets: A means of defining the appearance of a web-page
- Git: A program for software version control
- HTML - Hyper Text Markup Language: A markup language for creating web-pages
- HTTP - Hyper Text Transfer Protocol: The protocol responsible for handling the behaviour of HTML over a network
- MVC - Model-View-Controller: A software development design pattern
- PHP5: A scripting language commonly associated with web development
- RDMS - Relational Database Management System: Specific type of database which models the interaction between data objects
- PostgreSQL: A type of RDMS
- UI - User Interface: The part of an application that a user interacts with
- UX - User Experience: Describes the attitude of the user with regards to interacting with the UI

*2.2 Overall Description*

*2.2.1 Product Perspective* The Shopping Route Recommender is an application used by consumers to maximise their shopping experience in terms of three preferred optimisations: minimum cost, travel distance and travel time. The consumer is able to log onto a Website and create a shopping list with a desired route being generated. Enabling a user to optimise their shopping experience is a potential success from the start, as their daily routines can become more efficiently and effectively undertaken. The application's use is not only restricted to the general public, but can also be used by businesses and companies involved in the stock collection courier service industries. The application is aimed at being user friendly, simple, and interactive with maximum customisation being a priority aspect in order to maximise an individuals needs.

*2.2.2 Product Features* The list of product features below aim to provide an easy-to-use, customizable application interface for all users.

- Interactive shopping list menu.
  - add or remove item
- Interactive optimisation selection options.
  - minimise cost
  - minimise travel time
  - minimise travle distance
- Interactive shopping area selection options.
  - select from a number of suburbs or regions
- Interactive route map displaying alternate routes for selection.
  - rotate map
  - slide map
  - zoom in/out
  - satellite view

*2.2.3 User Classes and Characteristics* The application is aimed for the general public's use as well as certain business industries.

- General Public
  - General population wanting to buy their routine shopping list
  - General population looking for more specific products and their preferred optimised route
  - Foreign individuals looking for their ideal shopping locations or travel routes
- Business Industries
  - Courier companies collecting stock or products from various distributors/stores

*2.2.4 Operating Environment* Shopping Route Recommender is an application designed to run on the most Web Browsers as well on Google Android and Mac OS X Smartphones.

- Software Requirements
  - Internet connectivity
  - Entry level Smartphone
  - Mozilla Firefox, Microsoft Edge, Google Chrome, Microsoft Explorer, Safari
- Hardware Requirements
  - Entry level Smartphone with interactive touch screen

*2.2.5 Design and Implementation Constraints* Shopping Route Recommender is platform independent and is written using PostgreSQL, PHP, JavaScript, HTML and CSS. In addition, Google Maps API is implemented for generating the desired optimised shopping route. The accuracy of the generated route and optimisation algorithms is therefore dependent on the accuracy of the Google Maps API utilised.

*2.2.6 User Documentation* An application menu will be provided within the application. The menu will house information about the application, the application settings, history tab that is used to store user recent shopping routes and a help function that will function as the "user manual" of the application.

*2.2.7 Assumptions and Dependencies* The application is functionally dependant on Google Maps API. This API is integral to displaying the shopping route and directions to the user. One assumption is therefore that Google Maps is available on the website and shows the route from one shopping mall or shop to the next. Another functional dependency is that the application depends on the product database of each shop stored in the application's database. It is therefore assumed that each shop has a database of items that they stock as well as their respective prices. These product databases are continuously updated by each of the stores. The application must therefore contain the latest product database updates from each of the stores. The SRRec database must therefore be automatically updated once a store has registered an update in its product database. This is an important functional dependency if the application is to be as accurate as possible in terms of product pricing.

*2.3 System Features*

Shopping Route Recommender was designed with user experience as its primary concern. As a result of this the product features are simplistic in nature in order to provide the customer with only the essentials. This section provides a detailed description of each system feature in order to make future system extensions as simple as possible.

*2.3.1 System Feature 1 - Adding items to shopping cart*

*Description:* The Shopping Route Recommender's primary feature is the user's ability to add shopping items to their cart. The user will be able to log into the application and add items to the shopping cart on an ad hoc basis. These items will remain in the basket until such time that the user wants to go shopping.

*Stimulus/Response Sequences:* Once the user is logged in, his/her current shopping list will be displayed (in the order in which the items were added). The user can then add items to the list and remove items from the list by typing the items names into the corresponding "Add Items" or "Remove Items" blocks. The list can be modified with the following actions:

- Add new items - The user will be able to click in the "Add Items" block and enter new items to be added.
- Remove existing items - The user will be able to enter items from the current list that are to be removed into the "Remove Items" block. These items will be removed from the shopping list.
- Update List - Once the user has entered the items to be added and/or removed, the user must click on the "Update List" button in order to enforce the desired changes to the list.

*Functional Requirements:*

- The user can add multiple items at a time by comma-separating each item.
- The user can remove multiple items at a time by comma-separating each item.
- The user can add and remove items at the same time.

#### *2.3.2 System Feature 2 - Add Location*

*Description:* In order for the route to be provided it is required that the user's location is available. The generated shopping route is entirely based on the users current location, and therefore his/her location relative the shops of interest.

*Stimulus/Response Sequences:* The user is presented with a "Add Location" option on the home screen. Upon selecting this option the user has the ability to make their location known in two different ways:

- Entering their location manually.
- Finding their location through the use of the Google Maps API.

The primary motivation behind allowing the user to manually enter their location is that not all users will have access to GPS and thus will not be able to determine their location.

The location entered at this point is the same location that will be used as the starting position for the route that is planned when the "Generate Route" option is selected.

*Functional Requirements:*

- In order to use the "find my location" option the user is required to have GPS access. Most modern day smartphones and computers contain a location services setting that allows the location of the device to be accurately mapped.

#### *2.3.3 System Feature 3 - Preferred Optimisation*

*Description:* A decision was made in order to allow the user to have control of the nature of the route that they will follow. This was due to the fact that multiple customers will view different aspects as their primary concern. In other words, some users might value the cost of things over the distance required to travel, while for others the cost of shopping products may not be of any concern.

*Stimulus/Response Sequences:* On the home page there is a "Preffered Optimisation" drop down menu. Once the user selects the preferred Optimisation" drop-down menu they will be provided with the following options:

- Fastest Route - The user will be allowed to select that they would like to take the fastest possible route in order to obtain all the items on their shopping list. The primary contributor to delays will be traffic.
- Shortest Route - The user will be allowed to select that they would like to travel the shortest possible distance in order to obtain all the items on their shopping list. This selection will not incorporate traffic information.
- Cheapest Total Cost - The optimisation will primarily consider the cost of items at the expense of the distance required to be travelled in order to obtain these items.

The user will also be provided with all alternative routes and the time expense that would be incurred if they select an alternate route. The route distance optimisation will primarily be achieved through the use of the Google Maps API.

*Functional Requirements:*

- The application is required to link to the Google Maps API.
- The user is required to have GPS activated on their mobile device if they wish to use navigation.

#### *2.3.4 System Feature 5 - Generate Route*

*Description:* The generate route option incorporates all the above mentioned features and provides the user with their preferred route. The generated route is displayed on a Google Maps view with a highlighted shopping route.

*Stimulus/Response Sequences:* Upon selection of the "Generate Route" option the user will be directed to a page that contains the Google Map with the route drawn onto it as well as a set of directions in order to allow for off-line use. If the user would prefer to navigate using a mobile device they will be able to do so through the use of Google Maps.

*Functional Requirements:*

- Access to Google Maps API is required.

#### *2.3.5 System Feature 6 - Menu*

*Description:* The SRRec web application has a menu that contains basic user desirable tabs. These tabs incorporate settings for the application, help features, information about the application and a history of generated shopping routes.

*Stimulus/Response Sequences:* The menu tabs are user interactive and can be accessed by clicking on each tab once in the menu. Each tab contains specific web application information, as mentioned.

*Functional Requirements:* The menu tab requires the GUI to be user interactive, whereby a user is able to click and select specific tabs. It is therefore strictly dependant on a functional GUI and interface.

### *2.4 External Interface Requirements*

*2.4.1 User Interfaces - GUI* The SRRec GUI is simple to use and designed such that the user is able to access all the main features easily. The interface communicates with the data layer which then incorporates the Google Maps API to deliver the most accurate and optimised shopping route. Furthermore, the application is easily portable between devices of different screen sizes, in that the content on the pages automatically adjust to fit the appropriate screen.

The most common features of SRRec's GUI are:

- The sidebar menu accessible from all the subsequent pages can be seen in Figure 1:


Figure 1: Menu bar of application

- The landing page and main application window where a user inputs their shopping list, location and preferred optimisation from which the application generates an optimised route can be seen in Figure 2 and in Figure 3 with the accessible menu expanded:

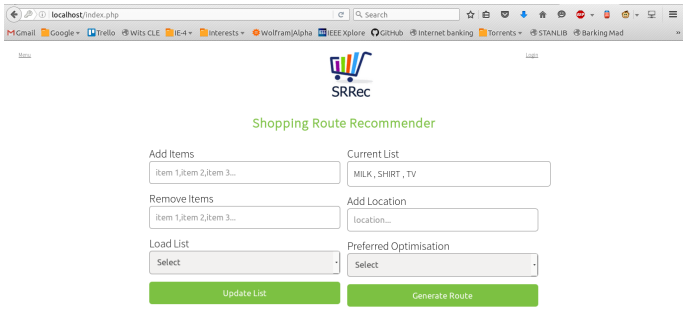


Figure 2: Home page of application

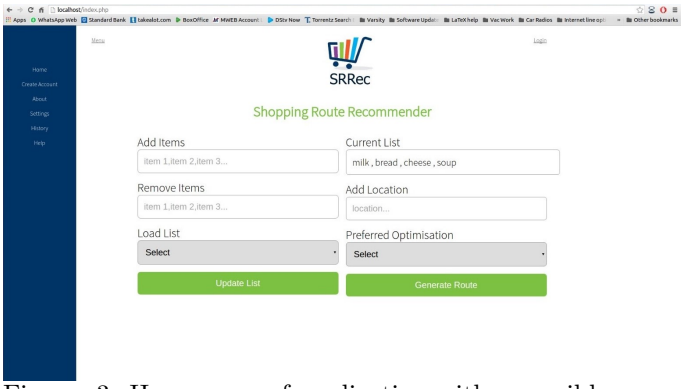


Figure 3: Home page of application with accessible menu open

- Route and Directions page generated after a user has input their shopping list, location and preferred optimisation can be seen in Figure 4:

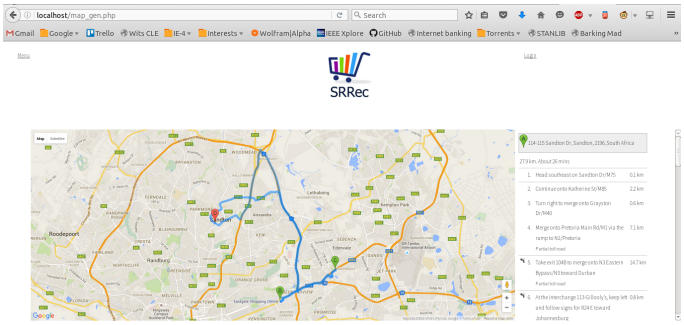


Figure 4: Page displaying route and directions

- The page where a user can either login or create a new account can be seen in Figure 5:

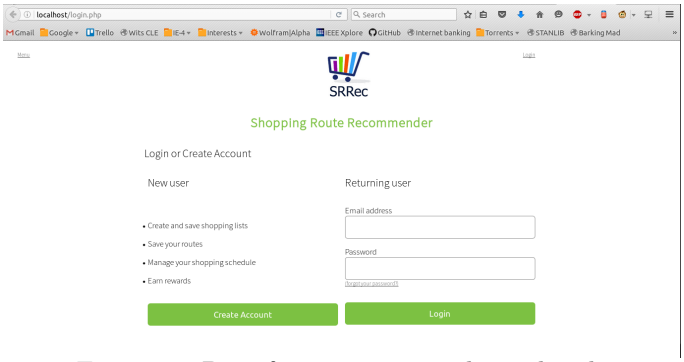


Figure 5: Page for entering user login details

- The page where a new user can create an account can be seen in Figure 6:

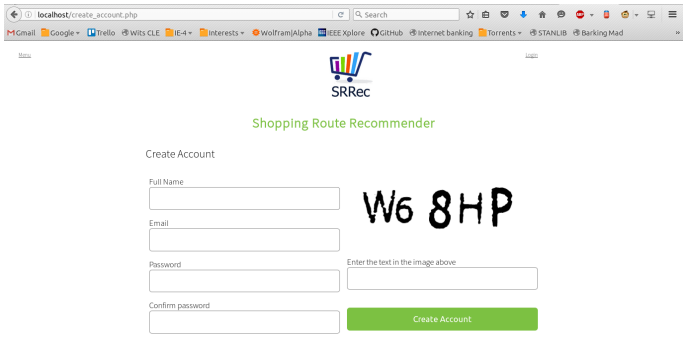


Figure 6: Page for creating a user account

*2.4.2 Hardware Interfaces* The SRRec application does not require nor support any hardware interfaces.

*2.4.3 Software Interfaces* SRRec is a web application. Therefore it will be compatible with all operating systems including smartphone and tablet operating systems. It only needs to be compatible on web browsers which support HTML, CSS, JavaScript and PHP. It has been tested on Mozilla Firefox, Google Chrome, Microsoft Edge and Safari web browsers.

Furthermore, the application makes use of the Google Maps API in order to generate and display the map with the optimised shopping route.

*2.4.4 Communications Interfaces* Since SRRec is a web application, network communications are necessary. It runs off a cloud hosted server such that it is readily available with an internet connection so as to communicate with both the user and the Google Maps API. Another communication interface is through the users GPS (if available), which the application uses to locate the nearest shops.

### *2.5 Other Non-functional Requirements*

*2.5.1 Performance Requirements* The SRRec is dependent on retrieving only relevant information from multiple large databases. This is to be done in an optimised fashion in order to allow fast data retrieval without unnecessary delays. The application should respond in real-time as the user inputs information.

Since the SRRec is a website application, any changes or updates made will be automatically available to the user the next time the website is loaded. Thus SRRec will always be up-to-date.

The application should have a maximum range for the shops selected so as not to recommend an impractical route. It should also be able to still provide a route even if certain items are not available nearby.

*2.5.2 Safety Requirements* The application should avoid taking users to the wrong locations, especially if those locations are in dangerous areas. The SRRec should only provide routes along known roads.

*2.5.3 Security Requirements* The databases used by SRRec should be protected from random access. User credentials are to be taken in by the application for users to sign into their profiles. Thus there needs to be a secure authentication system. The credentials also need to be safely stored on a secure server to protect them.

*2.5.4 Software Quality Attributes* This application incorporates the use of user credentials to allow users to store and access their lists on a server without losing their list or having to keep the website active. The application has a simplistic graphical interface that is easy-to-use. The website is easily interpreted and a new user should be able to use the website without requiring any sort of tutorial or explanations.

## 3 DESIGN DOCUMENT

The purpose of this documentation is to provide a detailed understanding of the structure of the software application. The documentation is primarily aimed at the software development team. The document aims to provide the software development team with a sense of guidance, thus ensuring the deliverables are in line with what is expected.

### *3.1 Scope*

The scope of the project, from a technical aspect, is that the design is composed of a front-end (client interface) and a back-end (server interface). The front end will consist of an interactive web application while the back-end will be a relational database. The two aspects will be required to interact in order to bring about the necessary response that the client desires.

### *3.2 Target Market*

The product is targeted at the general public. The reason being that all members of society are assumed to shop in some capacity. As a result of this the User Interface (UI) is required to be as simplistic as possible as this will avoid excluding those members of society that are not necessary technologically advanced. The primary purpose of the application being introduced is to better the overall well-being and state of mind of the general member of society.

### *3.3 Development Methods*

The system is to be developed according to the SCRUM principles [4]. This is ideally because an agile method of development is often preferred when designing an application. This is due to the frequent changes that the application is subject to. It is recommended that the reader familiarises themselves with the composed SCRUM selection document in order to understand the motivation behind the choices

made.

### 3.4 System Architectural Strategies

The overall system was designed according to the Model-View-Controller (MVC) design pattern [5]. The primary motivation for the implementation of this design pattern was due to the fact that the SCRUM design approach was implemented. Due to SCRUM, and the incremental development associated with it, the UI is subject to changes as a result of client feedback. The MVC pattern implemented is depicted below in Figure 7. The different aspect of this design will be detailed in sections to follow.

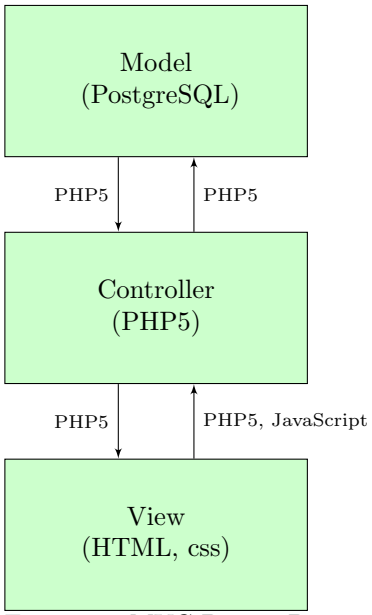

Figure 7: MVC Design Pattern

I: FRONT-END

### 3.5 Overview

The front-end's primary objective is to provide the user with a means of interacting with a complex database in a simplistic manner. The front-end is organised into four main pages. Each page is interlinked by either a direct button to that page or through the slide-out side-bar menu available on each page. The pages, with their corresponding functionality offered to the user, are listed below:

- Create account page:
  - Creating a user account
- Login page:
  - Logging in to the application with the above created credentials
  - A direct link to the create account page if the user does not yet have login details
- Index page:
  - Creating a shopping list
  - Adding to and removing from an existing shopping list
  - Creating multiple shopping lists with the above mentioned functionality
  - Selecting their preferred means of route optimisation: fastest, shortest or cheapest
  - Generating a route based on the shopping list and desired means of optimisation
- Map generation page:
  - View the generated map
  - View the generated directions

A visual overview of the front end is provided below in Figure 8. This visual aims to provide a broader view into the interaction between the application objects as well as a brief description of the utilisation of different languages.

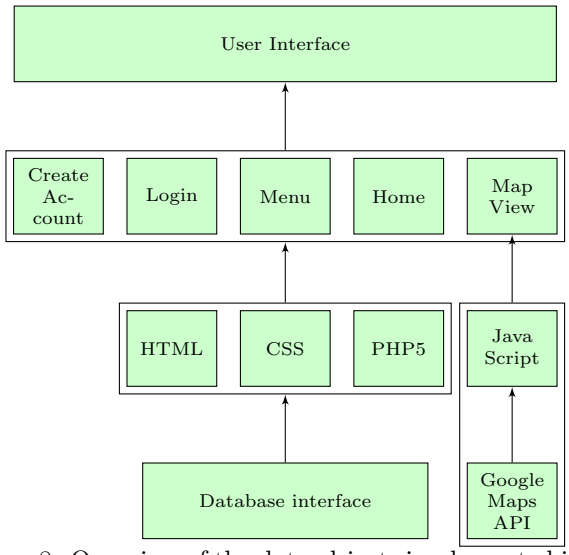

Figure 8: Overview of the data objects implemented in the front-end

### 3.6 Design Considerations

*3.6.1 Assumptions and Dependencies* The primary assumption is that the maps API that the general public is most familiar with is Google Maps. Hence, the Google Maps API was selected as the mapping utility to be used. Another assumption is that the most commonly spoken language amongst all users is English, as a result the application is designed primarily for English.

In order for a user to interact with the application the user is required to have access to an electronic device as well as an internet connection.

*3.6.2 Success Criteria* In order for the front-end of the application to be deemed a success a clean and simplistic User Interface (UI) is required.

### 3.7 Architectural Strategies

Certain architectural strategies were implemented in order to obtain a prototype in the shortest possible time. The primary motivation for implementing these strategies is that developing an effective user interface from the ground-up can be unnecessarily time consuming. The strategies implemented are mentioned below:

- JavaScript: The use of JavaScript to interface with the Google Maps API was primarily due to the fact that this code had previously been written and thus could simply be reused. Developing wrapper functions in `PHP5` to interface with the API would have consumed time and was deemed unnecessary to implement for a prototype. This was acknowledged as a future aspect to be ported to `PHP5`.
- CSS: Majority of the `css` files were implemented due to previous experience with the open source packages. Due to the familiarity of how to interact with the packages it reduced the time required to obtain the desired appearance and behaviour of the UI.

### *3.8 Use Cases*

An overview of the different use cases observed is presented below. These use cases are presented in a full detailed description, in order to avoid any misunderstanding of concepts.

Table 1: Use case for creating an account

| **Use Case** | **Description** |
|---|---|
| Name | Creating an account |
| Description | A new user wishing to utilise the application is required to first create an account |
| Primary Actor | User of application |
| Use Case Type | Client |
| Interested Stakeholders | Shops that supply data to the application |
| Goal | The successful creation of a new user account |
| Precondition | The user has entered credible details |
| Trigger | The user has entered all the required information and clicks the 'Create Account' button |
| Typical Flow of Events | • The user accesses the create account page<br>• The user enters the following information correctly: full name, email address, password and password confirmation<br>• The user clicks the 'Create Account' button<br>• The user is redirected to the home-/index page |
| Assumptions | The user has an email address to use as a username |

Table 2: Use case for logging into the user account

| **Use Case** | **Description** |
|---|---|
| Name | Login |
| Description | The user wishes to login to their existing user account |
| Primary Actor | User of application |
| Use Case Type | Client |
| Interested Stakeholders | Designers of the application |
| Goal | The user is able to successfully log into their account and view their corresponding information |
| Precondition | The user has entered their details correctly |
| Trigger | The user clicks the 'Login' button |
| Typical Flow of Events | • The user navigates to the login page<br>• The user inputs their details<br>• The user is redirected to the home-/index page |
| Assumptions | The user has previously created an account |

Table 3: Use case for adding items to the shopping list

| **Use Case** | **Description** |
|---|---|
| Name | Add items to shopping list |
| Description | The user wishes to add items to their shopping list |
| Primary Actor | User of application |
| Use Case Type | Client |
| Interested Stakeholders | Client |
| Goal | To successfully add the required item/s to the shopping list |
| Precondition | The user has created an account and successfully logged in |
| Trigger | The user selects the 'Update List' button |
| Typical Flow of Events | • The user logs in<br>• The user is redirected to the home-/index page<br>• The user enters the items that they would like to add to the shopping list<br>• The user selects the 'Update List' button<br>• The updated list is displayed on the screen for the user to view |
| Assumptions | • The user has correctly spelt the item/s they wish to add<br>• The item the user is requesting exists on the database<br>• The user has not already added the item to the shopping list |

Table 4: Use case for removing items from the shopping list

| **Use Case** | **Description** |
|---|---|
| Name | Remove item |
| Description | The user wished to remove an item from their shopping list |
| Primary Actor | User of application |
| Use Case Type | Client |
| Interested Stakeholders | Shops that supply data to the application |
| Goal | The user is able to successfully remove an item from their shopping list |
| Precondition | • The user has created a shopping list<br>• The user has added item/s to their shopping list<br>• The user has entered the list of item/s they would like to remove |
| Trigger | The user clicks the 'Update List' button |
| Typical Flow of Events | • The user logs in<br>• The user is redirected to the home-/index page<br>• The user enters the item/s they wish to remove<br>• The user clicks the 'Update List' button<br>• The updated list is displayed on the screen for the user to view |
| Assumptions | • The user has correctly spelt the item/s they wish to remove<br>• The items that the user wishes to remove are currently in their list |

Table 5: Use case for generating the map with directions

| **Use Case** | **Description** |
|---|---|
| Name | Generate Map |
| Description | The user wishes to view the recommended shopping route |
| Primary Actor | User of application |
| Use Case Type | Client |
| Interested Stakeholders | Shops that supply data to the application |
| Goal | The user is able to successfully generate a map with the associated directions |
| Precondition | The user has create and is satisfied with their shopping list |
| Trigger | The user clicks the 'Generate Map' button |
| Typical Flow of Events | • The user logs in<br>• The user is redirected to the home-/index page<br>• The user adds and/or removes items from their shopping list<br>• The user selects a preferred means of optimisation<br>• The user clicks the 'Generate Route' button<br>• The user is redirected to the page which displays the map and corresponding directions |
| Assumptions | The user has items on their shopping list |

### 3.9 Relevant Modules

#### 3.9.1 Style Modules (CSS)

*title1140.css:* 1140px is an open source css style. It is used to handle the grid and layout of the overall view of the web application. The css classes control the columns and rows of the html web page based on pixel length. Furthermore this style is responsible for page adjustment based on the screen size of the target device. The specific font format 'Source Sans Pro' and font weight are also included in the module.

*maps.css:* This style is used to specify the positions of the generated map and directions windows. Further class aspects are added in order to change the font size and positioning within the directions panel.

*idenav.css:* This css module is used to customise the side navigation menu panel. It styles the background colour, and text font and colour. It also positions the menu correctly and adjusts its extension into the page when the menu button is pressed.

*style.css:* The style css file is used to create the general style of the headers and buttons on all the web pages. This was encompassed in a single css as it would ensure consistency of all visual aspects of the web application across all web pages.

*3.9.2 Side Navigation Module* The menu is displayed in a side navigation panel. This panel slides onto the web page from the left when the 'Menu' button is selected. This functionality was achieved by using a JavaScript function called at the end of each web page so that it is always accessible. Furthermore, this module uses jQuery, a JavaScript library, designed to simplify client-side scripting of HTML.

*3.9.3 Create Account Module* The create account script is simply involved in allowing different users to create an account. The module primarily interfaces with the database in order to store the users details allowing them to access the web application at some point in the future. The script performs simple checks such as password confirmation and whether or not the information entered is of the correct form, this is all achieved through `PHP5` code embedded within the HTML code. Once the information entered satisfies all the required criteria, the page then interfaces with the database and adds the user, following this the user is redirected to the applications home page. The redirection is achieved through the use of the `PHP5` command `header(Location: 'page.php')`.

*3.9.4 Login Module* The login script is similar in nature to that of the create account module in the sense that it consists of a basic HTML form that interfaces with the database. Despite this, the login module is required to read and write from the database.

Once the user enters their credentials, the database is invoked via a POST request to obtain the password corresponding to the entered email. The embedded `PHP5` code is then responsible for comparing the password from the database to that entered by the user and bringing about the correct response. If the credentials are correct the user is redirected to the home page, otherwise the user will be required to re-enter their credentials.

*3.9.5 Home Page Module* This is the page the user interacts with to input their preferences. It is the homepage of the web application and will therefore be the first page that the user interacts with once they have logged in.

The page's main feature is the dynamically updating shopping list. The user is provided with the ability to alter the shopping list by either adding items to, or removing items from, the list. All changes made are relayed to the database, thus allowing the user to add and remove items from the shopping list on an ad hoc basis. Thus any changes made by the user will be permanently stored until such time that the user request that the route be generated. A future implementation is to allow the user to store multiple commonly used lists.

In an ideal situation the users geo-location will be used as the origin of the shopping route however, as this is not always available, the user is required to enter their desired starting location.

The last requirement from the user is for them to select their desired means of optimisation. The user is presented with the option to optimise the route by one of the following means: cheapest total cost, shortest possible route or quickest possible route. The default implementation for the prototype is the cheapest possible route as this encompasses all the basic functionality of the system.

Once the user has added all of their desired items to the shopping list they are required to select the 'Generate Route' button. This will generate a route, with directions, based on the items and chosen optimisation. All optimisation is performed by the back-end and relayed to the front-end in the form of a set of co-ordinates.

*3.9.6 Route Generation Module* This module requires information sent from the back-end. This information will consist of the users current location as well as all of the stopovers on the route. This information is sent to the Google Maps API which in turn will return the response. This response is then displayed on a Google Map in a separate window with the written directions alongside.

### 3.9.7 Google Maps API Module

- The back-end of the program returns a 2D matrix, with each matrix slot containing an X and Y co-ordinate, to the front-end. The co-ordinates correspond to the location of each shopping route waypoint.
- The front-end uses the co-ordinates of all the waypoints to insert them into a Google Maps API request message.
- The request message serves to connect to Google directions or directions matrix endpoints through 'https://maps.googleapis.com/maps/api'.
- Upon each request, the Google Maps API and front-end require an API key for each project as well as specific endpoint activations.
- The JavaScript function *initMap()* creates the *DirectionsRenderer* and *DirectionsServices* objects. It also maps the *directionsDisplay* map and panel objects to the windows in the HTML document.
- The *DirectionsServices* object initialises a call to the API containing the route information.
- The response and status are returned and displayed on the directions panel and map panel using the *directionsDisplay* object.
- The directions matrix endpoint is used to return distance and time information about the trips in order to select the optimal waypoints to send to the directions API.

## II: BACK-END

### 3.10 Overview

The back-end is composed of the Relational Database Management System (RDMS) PostgreSQL. The database provides a means of storing the relevant data obtained from different shops, storing information pertaining to the different users and manipulating the data during the optimisation calculations. The following sections will provide a description of how the back-end aspects of the application are composed.

### 3.11 Design Considerations

*3.11.1 Assumptions and Dependencies* The application is designed to run through an Apache2 server, this is primarily due to the server being easy to configure [6]. The Apache2 server is also known to integrate well with database management systems, which is essential for the functionality of the application. As previously mentioned the database management system that was chosen is PostgreSQL and the database interfacing language is `PHP5` [7].

*3.11.2 General Constraints* The system is largely constrained by the quality of data provided by the shops as well as the frequency with which this data is supplied. The application is heavily reliant on the continuous update of data as well as the data being presented to the company in a clean format. If the data is messy, i.e. contains spelling errors, incorrect formatting, etc., it will reduce the rate at which the system can be updated.

*3.11.3 Success Criteria* In order for the back-end system to be deemed a success it is required that the database provide an efficient means of handling the data and reducing the time required to display the information to the user. It would also be necessary for the database to provide a secure solution to handling the data as certain information will be deemed confidential to clients.

### 3.12 Architectural Strategies

Error handling is a fundamental aspect of the architectural design. The error handling was made the responsibility of the back-end. This was primarily to achieve and maintain the level of abstraction present in the design. The back-end is responsible for determining the required response to the user request (in error) and inform the front-end of the appropriate information to display. Certain errors that were accounted for are listed below:

- Incorrect credentials entered when creating an account
- Incorrect credentials entered when logging in to an existing account
- Item not found on the database when adding to the shopping list
- Item not found in the existing list when attempting to remove an item
- Unable to connect to the Google Maps API

### 3.13 System Architecture and Data Design

The software behind the Shopping Route Recommender is designed in a responsibility-driven structure [8]. The responsibility-driven structure is primarily centred around the client/server model and the roles that each entity plays in the communication of data objects. The main aspect of the structure is to abstract the details of how the server handles the clients request, from the client. Therefore, the design is structured in such a way that the client can only specify the intent of the requests and the back-end is configured in such a way to handle the specified request by encapsulating the means of how it responds to the clients request.

The system was divided up into multiple structures, of which objects could be created, in order to achieve this level of abstraction. The objects that could be created are listen below:

- User/Client objects
- Shopping list objects

The system is relatively simple in the sense that only two types of objects can be created. Different users can be created, all of them being encapsulated into the category of employee, and each user will be able to create multiple shopping list objects.

The user is able to request that they be allowed to create one of the above mentioned objects, the server is then responsible for checking the credibility of the request and bringing about the appropriate response. If the request is

successful the server will create and store the corresponding object on the database and notify the client of the success. if the request is unsuccessful then the server will not create the object and the client will be notified of such.

The client objects are restricted from interacting with one another to ensure the confidentiality of information, these are classified as having private responsibilities [8]. The individual client objects have ownership over their respective shopping list objects, therefore forming an object neighbourhood [8]. The client objects have the ability to interface with the database to edit their shopping list objects. The abstraction is obtained by the client entering a list of additions/deletions while the server determines the integrity of the request and carries out the required request if it is possible. The client is notified of these changes when the shopping list displayed to them is altered. Therefore, in the described situation the client plays the role of a controller while the shopping list plays the role of an information holder [8]. The shopping list could be seen to display controller roles in the sense that the map objects are reliant on them, but it is still the responsibility of the client to control the generation of the route.

A further factor that plays a role in the system is that of the map objects. The map objects are temporary in the sense that they have a limited life-time from the generation of the route to the completion of the route. The map objects are owned by the individual client objects and can also be viewed as having no responsibilities in the system but rather forming an information holder role.

### *3.14 Relevant Modules*

*3.14.1 Home Page Module* This module receives items to be added and/or removed from the users shopping list. All the new items are added to the users shopping list on the database. The items to be removed are compared with the users current shopping list. If these items are in the database they will be permanently removed.

*Load list:* In future iterations of this project a load list will be implemented. This will allow the user to have multiple lists linked to their profile on the database. The back-end will receive which list to select. The correct list will then be found on the database and returned to the front-end for the user to interact with.

This module also receives the users starting location. This location is stored in the database as the origin for the Route Generation Module.

*Preferred Optimisation:* In future iterations of this project the ability to select a preferred route optimisation will be allowed. The back-end will receive which type of optimisation to perform, namely shortest total distance, lowest total cost or shortest total time. The back-end will then use this optimisation when selecting which stores to visit and/or which route to travel.

*3.14.2 Database Permutations Module* The origin location and shopping list are received by this module. The module is then responsible for generating all the possible routes for the user. The module accesses the database to determine which shops sell the products on the users shopping list. The code then dynamically creates tables for each item on the list which contains a column corresponding to all the shops which sell the product as well as another column for all the prices of that item at the corresponding shops.

Once all the tables have been created the code then performs a `CROSS JOIN` on all the tables which provides all the possible permutations of shops that contain the items on the shopping list. In the same line of code a sum of all the prices is taken. The resulting table consists of a new route on each row with each column corresponding to a waypoint on the route. The final column of the table contains the total cost of the items for that route. The above table is then ordered in descending order according to price. This is to make determining the cheapest route easier in further calculations.

*3.14.3 Route Generation Module* This module is responsible for combining the route information into the correct format for the front-end to interact with. This module takes the above mentioned table and replaces the shops with their corresponding coordinates obtained from the database. This is achieved by reading in the above mentioned table from the database and indexing over each location in the table of routes vs waypoint as mentioned above. The code is triggered, in conjunction with the permutations module, when the user selects the 'Generate Route' button on the home page.

While iterating over each entry of shop name the module dynamically adds the corresponding shops coordinates to a 2D array. This 2D array is what is presented to the front-end, through the use of `PHP5`, in a format that the front-end is able interpret. The front-end then interacts with the Google Maps API in order to present the desired route on a map.

Since the cheapest route optimisation is the only option available in the prototype, the other means of optimisation will be achieved in the following manner. The information for the shortest route and fastest route can only be obtained from the Google Maps API. In order to differentiate between the routes, the database will present the front-end with an individual route at a time. The front-end will then interface with the API to obtain the distance and time and forward the information to the database. This concept will be repeated for all routes, at which point the back-end will have the required information to be able to determine the cheapest, fastest and shortest routes. Therefore, irrespective of the optimisation chosen the database will be able to provide the front-end with the corresponding route.

*3.14.4 Create Account Module* A new users full name, email address and password is received by this module. These credentials are then stored in a table within the database and used later for user verification by the login module.

*3.14.5 Login Module* This module receives an email address and password. These credentials are then compared with those stored in the database. If a match is made then that users specific shopping list is retrieved from the database and passed to the front-end to be displayed.

## 4 SPRINT PLANNING

One of the ways in which the group plans on both implementing and monitoring the SCRUM method is through an application called Trello [9]. Trello forms an important backbone to the structure of the SCRUM method planning, by providing a carded system. Each card forms part of a specific sprint, the cards can be taken by individual group members and upon completion of the tasks on the card (in the allocated time frame), the member updates the progress of the task. In doing so, the other group members are informed of all the individual member's project progress. A screen shot of the board currently being used can be seen below.

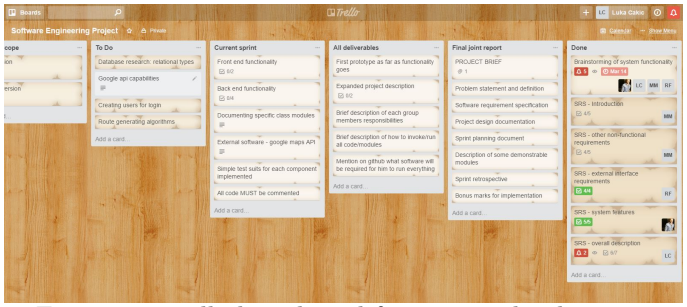

Figure 9: Trello board used for project development

### 4.1 Project Predecessor Tasks: 18 February

- Create the project backlog for the SCRUM procedure. This involves identifying the customer requirements, the developer team abilities and the list of project priorities that the team need to address during the project sprints.
- The backlog must thereafter be prioritised.
- Set up a sprint planning meeting
  - negotiate the duration of the sprints
  - select the target backlog for the sprint
  - clarify sprint requirements
  - break the sprint requirements into project tasks

### 4.2 Sprint 1: 18 February - 26 February

- The first portion of this sprint is to familiarise the group with Git and the GitHub platform. This involves configuring Git with each group members personal computers
- The next task is to create a project on GitHub
- The group is then expected to choose a project topic that will be focused on for the duration of this project
- An expanded description of the selected project, distinctly describing the front- and back-ends is to to be provided
- The description must also detail the expected inputs and outputs of the system
- The responsibilities of each paired group of students must also be detailed in this submission
- The documentation of this sprint must be uploaded onto GitHub such that it can be expanded upon at later stages in the development of the project

### 4.3 Sprint 2: 3 March - 10 March

- Set up a sprint planning meeting
  - identify the duration of this sprint
  - select the item targets from the backlog for this sprint
  - break the sprint requirements into project tasks
- The initial stage of this sprint is to firstly select a software development life-cycle (SDLC). The choice must be based on Agile SDLC
- The second stage is for the group members to choose a system architecture
- The development team must also focus on choosing an appropriate front-end interface method
- In order for the system to be fully functional an appropriate back-end must be designed. This involves selecting an HTTP server and an appropriate Database Management System (DBMS)
- Supporting API's are also brought forward, discussed and the choices are made clear with regard to the requirements of the system
- With all the aforementioned decisions being made, the main purpose of this sprint is to produce the first draft of a detailed Software Requirement Specification

### 4.4 Sprint 3: 24 March - 4 April

- Set up a sprint planning meeting
  - identify the duration of this sprint
  - select the item targets from the backlog for this sprint
  - break the sprint requirements into project tasks, identify the components that revolve primarily around the system prototype
- Finalise the structure of the back-end and front-end of the prototype. This includes decisions revolving around the frameworks, languages as well as overall appearance of the project
- Configure the HTTP server
- Configure the database to conform with the structure of the HTTP server
- Decide what `css` styles are applicable to the project
- Configure the database to interact with the front-end through the use of an interfacing language
- Integrate front-end with back-end
- Finalise system prototype
- Write test-code to ensure the prototype functions as expected

### 4.5 Sprint 4: 5 April - 11 April

- Set up a sprint planning meeting
  - identify the duration of this sprint
  - select the item targets from the backlog for this sprint

  - break the sprint requirements into project tasks, identify the required documentation for the system
- Finalise the project's software requirement specification documentation
- Construct the project's software design documentation
- A document describing how the final system is invoked must be created
- The front-end detailing must be updated and described fully, including describing the respective views accessible by all the web application users
- The back-end must also be detailed fully and the techniques describing its configuration and implementation must be finalised
- The writing and documentation of some important class modules must be produced, these must identify and illustrate the key aspects of the implemented web application solutions
- Reflect on the performance of the SCRUM method for the project

## 5 SPRINT RETROSPECTIVE

This section details the groups retrospective views on each of the project sprints. Each sprint is analysed and future improvements regarding sprint planning and execution of project tasks are presented. In addition, the positive outcomes of some of the sprint planning is discussed. Each sprint involved setting up a sprint planning meeting. These meetings would allow the group to structure each sprint and their respective tasks. This allowed the tasks to be designated to group members and ensured the group maintained the project scope and objectives.

### *5.1 Predecessor Tasks*

A detailed and prioritised project backlog was produced prior to the start of the sprints. This was essential as it assisted the project team with identifying key project tasks and allowed them to be structured and ordered, such that each sprint was well planned and therefore well executed. The group realised that the key to producing a logical and well structured sprint planning document lied with a well thought out project backlog. The team also clarified the project requirements and made certain that each member understood each requirement.

### *5.2 Sprint 1*

This sprint involved the set up of the GitHub project repository. Fortunately, GitHub had been previously implemented in other university courses and thus each group member knew Git and the GitHub platform prior to this stage. The project selection was another important stage in this project, the group had to select a project each member would be comfortable with. Following a discussion, the group chose the Shopping Route Recommender project based on the group's abilities and opinions.

### *5.3 Sprint 2*

This sprint mainly focused on producing a Software Requirement Specification document. This document was to be produced following in depth research regarding the front-end and back-end implementations. The document was well designed as each group member was assigned a portion of the document. This allowed the members to focus their attention on specific topics, thus providing more in depth detail. The document was produced using LaTex thus each member could simultaneously edit the PDF source code using Git. This maximised group efficiency as members were able to see changes as they occurred.

### *5.4 Sprint 3*

The first step for this sprint was setting up the sprint planning meeting. The meeting allowed the members to designate the workload for producing the front and back end prototypes. In addition, the team was able to discuss the requirements of the web application prototype in detail, with presented options for the application's components being discussed. The prototype that was created illustrated the basic functionality of the web application. A route would be generated based upon an input shopping list and preferred optimisation. The prototype used the Google Maps API to accurately display a shopping route. Based on the project objectives, the objectives regarding the prototype were met. In hindsight, with a more structured time management plan, the back-end database could have been expanded to include more shops and products. In addition, a price range option could have been implemented. Nevertheless, the prototype created performs its core function; it provides a user with a preferred optimised shopping route. A future improvement might be to create a more thorough test code. This code could test all aspects of the application and would thus highlight any bugs or flaws in the system.

### *5.5 Sprint 4*

This sprint involved finalising the projects documentation. All the documentation was done using LaTex and could thus be simultaneously edited by each group member. The documentation tasks were split amongst the group members. An improvement for future projects might be to perform documentation simultaneously with prototype development. In this manner, the reasoning regarding implemented methods would be well documented at a time when the decisions are fresh in the developer's mind. It is often challenging to document some development decisions at a later stage in the project, when reasoning or explanations have been forgotten.

## REFERENCES


[1] Psychologist World. *10 Causes of Stress*. Psychologist World. Available: https://www.psychologistworld.com/stress/ten-causes-of-stress-how-to-avoid-them.php#references. Last Accessed: 1 April 2016.

[2] Google. *Google Maps APIs*. Google. Available: https://developers.google.com/maps/. Last accessed: 6 April 2016.

[3] IEEE. *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Std 830-1998 (Revision of IEEE Std 830-1993).

[4] Mountain Goat Software. *SCRUM*. Mountain Goat Software. Available: https://www.mountaingoatsoftware.com/agile/scrum. Last accessed: 2 April 2016.

[5] Microsoft. *ASP.NET MVC Overview*. Microsoft. Available: https://msdn.microsoft.com/en-us/library/dd381412%28v=vs.108%29.aspx. Last accessed: 1 April 2016.

[6] Apache. *The Apache Software Foundation*. Apache. Available: http://www.apache.org/. Last accessed: 3 April 2016.

[7] PHP. *PostgreSQL*. PHP. Available: http://php.net/manual/en/book.pgsql.php. Last accessed: 8 April 2016.

[8] Wirfs-Brock, R. *A Brief Tour of Responsibility-Driven Design*. Wirfs-Brock Associates. Available: http://www.wirfs-brock.com/PDFs/A_Brief-Tour-of-RDD.pdf. Last Accessed: 9 April 2016.

[9] Trello. *About Trello*. Trello. Available: https://trello.com/about. Last accessed: 10 April 2016.