CO3016 Computing Project

“Smart Shopping Trolley”

Dissertation

Project by Kostadin Valentinov Kolev

Department of Computer Science, University of Leicester

05th of May 2016

Abstract

The objective of this project is to develop an Android application allowing its users to buy grocery items from shops near their current location. Unlike other similar apps “SmartShopping trolley” is able to give predictions to its users based on their past purchases. Also, users are able to create an account in the system and top up credit in order to buy items.

The application implements the client-server architecture style. The communication between the different parts of the system is achieved by sending HTTP requests from the client to the PHP server. The server processes the requests and browses the MySQL database in order to send the data needed to the client in a form of a JSON array. When received by the client the data is extracted from the array and presented to the user by a special class called an adapter transforming the received data into clear and easily understandable information.

The main challenges of this project were due to the fact that Android development was a concept which was not familiar to me before I started the task. Apart from this, a list of requirements had to be created in a way allowing different requirements to be prioritised in terms of their importance and amount of effort which had to be spent for their implementation.

The benefits of this project are entirely connected to the way that people shop. In other words, if carried on, this project can make shopping easier since people do not need to be present inside the particular shop in order to buy items. Apart from this, the project gave me a better understanding of the challenges which are faced during the development of an application of this size.

DECLARATION

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people’s work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people’s work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this, amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Kostadin Valentinov Kolev

Signed: K.Kolev

Date: 05.05.2016

Contents

[Introduction 1](#_Toc449492336)

[Survey of Information Sources 1](#_Toc449492337)

[Android development 1](#_Toc449492338)

[Requests and data parsing 2](#_Toc449492339)

[Why JSON? 2](#_Toc449492340)

[Server 2](#_Toc449492341)

[Why PHP? 3](#_Toc449492342)

[Requirements 3](#_Toc449492343)

[Why requirements are important 3](#_Toc449492344)

[Strategy for collecting requirements 3](#_Toc449492345)

[Volere approach 4](#_Toc449492346)

[Functional requirements 4](#_Toc449492347)

[Non-functional requirements 7](#_Toc449492348)

[Technical Requirements 9](#_Toc449492349)

[Software Architecture 11](#_Toc449492350)

[Layers 12](#_Toc449492351)

[Presentation layer 12](#_Toc449492352)

[Business logic layer 13](#_Toc449492353)

[Database layer 14](#_Toc449492354)

[Benefits 16](#_Toc449492355)

[Disadvantages 16](#_Toc449492356)

[Solutions 16](#_Toc449492357)

[Algorithms 17](#_Toc449492358)

[“Smart” algorithm 17](#_Toc449492359)

[Problems 17](#_Toc449492360)

[Solutions 17](#_Toc449492361)

[The algorithm 17](#_Toc449492362)

[Location algorithm 18](#_Toc449492363)

[Problems 18](#_Toc449492364)

[Solutions 18](#_Toc449492365)

[The algorithm 18](#_Toc449492366)

[Testing 18](#_Toc449492367)

[Critical parts 18](#_Toc449492368)

[Connection and data fetching methods 19](#_Toc449492369)

[Data parsing methods and presentation classes 19](#_Toc449492370)

[Testing strategy 19](#_Toc449492371)

[Procedure 20](#_Toc449492372)

[Test cases 20](#_Toc449492373)

[Critical Appraisal 22](#_Toc449492374)

[Bibliography 24](#_Toc449492375)

[Appendix 25](#_Toc449492376)

[Appendix 1 shopList.java 25](#_Toc449492377)

[Appendix 2 shopAdapter.java 26](#_Toc449492378)

[Appendix 3 grocery.php 26](#_Toc449492379)

[Appendix 4 shopList.php 26](#_Toc449492380)

# Introduction

“SmartShopping trolley” is an application designed to run on Android devices. Users will be able to buy different products from the registered shops. Also, users will be able to use a “smart” functionality which will help them choose the desired items. This is achieved by implementing various algorithms to store the data and distribute it to the Server. The objectives of the project were to create an application which implements the following functionalities:

* Connection to the server and exchange of requests with it;
* Access to system’s database where the data will be stored;
* Creation of locally stored data which will store some of user’s data;
* Creation of session variables used to implement payment functionalities;
* Navigation allowing users to choose the shops in their desired area.

Apart from the stated “functional” objectives this project will concentrate on several different areas in Computer Science, such as:

* Research on different stages of software engineering;
* Creating an Agile method to produce requirements for the system;
* Breaking down the approved requirements and prioritizing them.

# Information Sources Survey

During priject’s initial phases a list of topic’s main areas to be researched was created. These areas had to be approached with extra attention because they were unknown for me.

## Android development

Probably the biggest area of research which was crucial for software successful development was Android Development. Some layers in the program are involved in the usage of Android devices. The development of applications for Android was a very new area to me, so I had to pay extra attention to the research of different methods able to implement the necessary functionalities. The most difficult part in creating the android application was to create an UI able to present the data in a clear and easily understandable way. Some of the most crucial areas for the successful Android UI functionalities development were:

* XML document: records the information about different shapes and buttons used in the main pages.
* Adapters: helper classes grouping similar set of data in one place and applies some rules to look though them. These classes were used to create the lists of shops and the grocery lists for each shop.
* Listeners: methods recording user activity. Since Android devices are using touch screens, it was assumed that Android offers some listeners designed for touch screens. Such listeners were used to give extra information to the users about the desired product.

Apart from stated areas of research, the overall behavior of android programs had to be researched. In this area I did not have any previous understanding. Since Java is used for some of Android programs’ main parts it was easy to understand the syntax. However, Android programming has its differences like:

* Intents recording every window in the program;
* OnCreate() methods executed when creating the page to create the buttons holding different functionalities;
* DoInBackground() methods – these methods are closely connected to Threads in Java. They allow the execution of assyncTasks executed on the background and used to establish the connection with the server and to exchange data with it. Also, through these methods the parsing of JSON arrays is achieved.

## Requests and data parsing

In order to implement necessary functionalities a method to make calls between the different layers of the program has to be created. The easiest and the most straightforward approach for this task is to use HTTP requests.

Http requests are used to exchange messages and information between the client and the server. This is achieved by creating a request (Post or Get) which might include some attributes which will be used by the server later. These requests are created by the user and sent to the server where they are accepted and executed. Most of the requests which a user is allowed to create are connected to fetching data from the database and retrieving it to the user in a particular way. This means that most of the request will be involved in the delivery of user information or information which is important for the overall performance of the program. This deduction was the field of research for one of system’s crucial parts: parsing the data in a datatype. During my studies in the 3rd year I was introduced to the concept of XML and parsing of XML documents with DOM and SAX. However, for the sake of increasing my knowledge in the field of Computer Science, I have decided to research a better way for parsing. JSON was the best alternative and it was chosen for a way to parse the database queries.

### Why JSON?

One of the main benefits of using JSON is that it is more understandable for humans. Instead of using various tags and taking care of their proper nesting in order to produce a well formed XML document which can be parsed by an algorithm, JSON uses an ore simple way to encode data. Different entries are separated in { } and different elements are separated by commas (,). This way of visualizing different entries increased my overall understanding of data types.

Apart from the stated reason, JSON is gaining popularity in modern applicationsdevelopment. It is chosen as an alternative technique to XML, and I think that by using this approach system’s overall performance might be increased, since major companies have decided to exchange data using this format.

## Server

The server accepts http requests made by the user. During the communication (which will be explained in detail further) the request is matched to a URL holding a php file. This file holds necessary functionalities which the user requests. After contacting the database the php file returns an encoded JSON array holding particular query’s values. The creation of a php server was a challenge for me, since my programming education is more Java orientated. The research here was concentrated on finding a way to make the communication between MySQL database server and given php page.

### Why PHP?

Java was considered to be used for storing the server and managing the requests of the client. However, it was suggested to use php for doing this, because I would be able to learn a different programming language. This benefits the overall impact of the project. Apart from the stated reason, there are a couple of technical reasons to choose php over Java:

* PHP is a very supportable language for servers. Almost every server supports the execution of php pages. This provides the ability to upload the program to different servers, thus increasing greatly project’s size and usage.
* PHP is easier to learn. PHP scripts are often very short and do only a particular function. This gave me the opportunity to spend less time on developing the php server and concentrate more on system’s other parts development. Using Java more time would be spent on developing the server creating different servlets and implementing the functionalities. Instead, using PHP a script had to be created for the given functionality without worrying about server’s actual structure.

# Requirements

In order to ensure the quality of the delivered documentation and the accompanying code a requirements document had to be created. The purpose of this artefact is to show software’s functionalities and point out how the software would behave. These two distinct categories of requirements will be listed in the functional and non-functional parts of this section. The following paragraphs will outline the need of producing such a document and the approach used in their collection.

## Why requirements are important

Outlining the main set of requirements will make the scope of the whole project more accurate. In other words, if the activities which are required to be performed by the system (functional requirements) are collected and then carefully analysed, it is guaranteed that the view for program’s overall functionality will be clearer. Furthermore, by collecting the requirements different tasks will be produced, prioritized, and arranged to guarantee that the functionality of the produced software will be on the necessary level. Another important fact is that a well formed and organized requirements document can link each of the different points of this artefact to a bit of the code. This will ensure that the code will perform exactly what it is required to perform and will outline any possible differences between the desired and the actual functionality. These differences will be used later in the Testing and Debugging section of the dissertation.

## Strategy for collecting requirements

In order to create a documentation implementing the stated important points, a valid strategy for their analysis had to be used. Also, this is a way to guarantee that stated problems can be handled

* Implementign the functionalities listed in the following parts of the dissertation, the priority of some requirements changed due to my work on other modules of my degree. The development of some functionalities was postponed because it was assumed that the results of their successful implementation will not correspond to the time spent to implement them.
* Some requirements’ details changed during development process. This was due to the fact that the view about software’s overall performance changed couple of times, so a change of the requirements was needed too.
* Some initial requirements dropped from the presented list because it was considered that the benefits from their potential implementation were not enough.

## Volere approach

After a careful research about different templates which might be used to guarantee that the stated problems are solved the Volere approach (find the exact name) was used to create the requirements list. This approach is used because it provides a successful way to write “cards” which correspond to each requirement. In these “cards” different important points are recorded and in the end a calculation for the overall importance of the requirement can be made. The main points of interest in these cards are:

* Requierement ID . This field records an unique identifier for each requirement assuring this way that the requirements can be traced or grouped in different groups regarding their functionalities.
* Description – a short abstract outlining the requirement.
* Rationale-A short paragraph outlining why this requirement is important.
* Dependency. This field records requirements’ ID that needs to be implemented before implementing the given requirement. Playing these different developing scenarios helped deciding which sequence of requirements developing will be made.
* Satisfaction/dissatisfaction level – These fields record the importance of implementing the requirement. Satisfaction level shows how much the implementation of the requirement will be needed for the overall functionality of the product. The dissatisfaction level shows whether the requirement will influence if it is not implemented. Both values have crucial importance on the prioritization of requirements. For example, requirements with higher satisfaction/dissatisfaction levels had to be implemented until the next supervised meeting. However, requirements with lower scores were left for later.
* Fit-criterion – this section of every entity in the requirement document will record how it will be traced. This value shows the way to measure whether the requirement has been implemented.
* Priority – indicates the priority for implementing the requirement. Requirements with higher priority will be implemented first.

### Functional requirements

* Login System

Requirement ID: 1

Description: Users must be able to log in the systems. Users` credentials will be checked and compared to the database during login.

Rationale: By implementing login functionalities it is assured that only registered users can use the software. Also it provides a certain level of security against unauthorized access.

Fit-criterion – unit test cases which will be launched during testing. Various attempts will be made to access the system. The users who do not have credentials in the system will not be able to gain access.

Satisfaction level: 5

Dissatisfaction level: 5

Priority: High

* Payment

Requirement ID:2

Description: Users must be able to make payments after choosing the desired goods. The price of the goods must be substracted from the balance of the user giving him/her the value of the new balance.

Rationale: Payment is one of the core functionalities which the system has to implement.

Fit-criterion – unit test cases will be created. They will simulate that a user with the given username makes a payment. It will be checked whether payment calculations are correct and whether the data is stored in the database.

Satisfaction level: 5

Dissatisfaction level: 5

Priority: High

* Shop list have to be generated

Requirement ID: 3

Description: Users should be able to see the list of all shops which are present in the database of the system.

Rationale: This function will help the users see how many different shops are there in the app. Also, it will allow useful upgrades in the future, such as searching for a shop.

Fit criterion – unit test cases will be launched, it will check whether the number of shown shops is the same as the number of the shops in the database.

Satisfaction level: 4

Dissatisfaction level: 5

Priority: High

* Users must be able to view the grocery lists for each shop

Requirement ID: 4

Description: The software must be able to produce a list of the items which can be bought in the shops.

Rationale: This functionality plays a crucial role in the overall performance of the program. Its early implementation will be used to build similar functionalities.

Fit criterion – unit test cases will create a test list of groceries which will be compared to the actual grocery lists. The requirement will be considered as implemented if the lists are the same. Any difference in the list will be an indication that the there is a bug in the system

Satisfaction level: 5

Dissatisfaction level:

Priority: High

* Showing the closest shops to the user

Requirement ID: 5

Description: The closest shops to the user need to be shown based on the current coordinates of user’s device

Rationale: This function will make the process of shopping more intuitive because users will be shown shops which are closest to their location. Also, this will increase the efficiency of the app since only shops close to the user are shown.

Fit criterion – unit test cases will be launched to check whether generated shop/shops correspond to actual shops which are closest to the user

Satisfaction level: 3

Dissatisfaction level: 3

Priority: Medium

* Top up user accounts

Requirement ID: 6

Description: Users should be able to deposit money in their accounts. In this way they will be able to buy items in the shops. Once the user has paid for his account, the new balance will be shown.

Rationale: This function will allow users to put cash in their accounts. The implementation of payment functionality will be strongly dependent on this one.

Fit criterion – Unit test cases will be launched to add money to the test account. The changes will be recorded to the database. If the obtained values are the same as the predicted ones this requirement will be counted as implemented.

Satisfaction level: 5

Dissatisfaction level: 5

Priority: High

* Personal data recorded on the device

Requirement ID: 7

Description: Information about bought groceries will be stored in user’s device.

Rationale: This functionality will have a crucial impact on the creation of items suggested to the user.

Fit criterion; Unit test cases will record a set of bought items in the internal database. The test will be considered successful if the data is stored properly.

User satisfaction: 3

User dissatisfaction: 5

Priority: Medium

* Indicate measuring units for each grocery item

Requirement ID: 8

Description: Units in which the grocery items will be shown. For example, if the user wants to buy an item measured in kg the software will indicate that the item is sold in kilograms.

Rationale: Some of the items sold in the shops have different units of measurement. The system should be able to make a difference between them and to tell the user about this difference.

Fit criterion: Unit test cases will show a set of items. The units in which they are measured will be compared to the units shown to the user.

User satisfaction: 3

User dissatisfaction: 4

Priority: Medium

* Receipt for purchased items

Requirement ID: 9

Description: Before the actual payment takes place a receipt should be issued to the user. It will include all purchased items, their individual price and the total price for the purchase. In the end, the new balance should be presented.

Rationale: By issuing a receipt the program will ensure that the user knows what he/she is paying for and can see if the chosen list of items is correct. Also, the user will be able to see if he/she has the necessary balance for the transaction.

Fit criterion: Unit test cases will create receipts. After that, the values in the receipts will be checked. The requirement will be considered implemented if the generated receipts are correct.

User satisfaction: 4

User dissatisfaction: 4

Priority: High

* The app must provide sessions

Requirement ID: 10

Description: The username and the password of the user will be recorded in a session variable once the user has logged in the system.

Rationale: By implementing sessions the implementation of other functionalities such as payment is made easier since the needed values will be stored in the session. Furthermore, using sessions the application will be able to keep track of the user who has logged, and show the correct information (if more than one user share the same phone).

Fit criterion: Unit test cases will be created to record the test values in the session. Different activities, such as payment, will be performed. At the end of each activity the session values will be updated. In the end, session variables will be compared to the expected values. The requirement will be considered implemented if the values are the same.

User satisfaction: 3

User dissatisfaction: 4

Priority: Medium

* Recommended items needed to be shown tothe user based on his/her taste

Requirement ID: 11

Description: Users should be able to see a list of items recommended for them. This list will be created based on the items that the user has bought previously.

Rationale: This functionality will increase system’s. If it is implemented properly the system will be able to provide a list of items which can be bought in the specific shop. This will make working with the system faster, since most of the users tend to buy the same products.

Fit criterion: Unit tests will be created to produce different lists of items based on the items bought before. The items will be compared to a list which includes the most frequently bought items bythe user. The requirement will be considered implemented once the generated list of items is the same as the list of most frequently bought items by the user.

### Non-functional requirements

The non-functional requirements are another category of requirements implemented in the software. Non-functional requirements show how the system should behave and most importantly, what the system should be. However, unlike the previous part of the dissertation the requirements in the non-functional section cannot be traced directly to some part of the code. The way to measure them will be to see which parts of the code have the required feature. In other words, these requirements cannot be tested. However, it is possible to inspect whether the tested functionalities have the listed traits. This following section will outline the main non-functional requirements which have to be implemented in the system.

* User experience

The software should provide an easy for using user interface achieved by creating as little elements in each window as possible, and thus ensuring that only the most important information will be shown to the user. Templates will be used to present the data in pages different elements. This will guarantee that the way of presenting the data will be the same in every page.

Also, user’s experience will be increased by implementing various gestures which will trigger different events. For example, if the user wants to see more information about some product he/she will need to long press on the element and a message will appear. implementing various event listeners will ensur that the information will be presented in an interactive way which will undoubtedly enhance user’s experience.

Another way to increase the user’s experience is to decrease the time needed for the user to feel comfortable in using the application. This will be achieved by following already defined conventions in the creation of User Interface. For example, the positioning of the buttons will be the same as the way the buttons are shown in similar applications. This will help the user get familiar with the program faster which will definitely make the user happy to use the software.

* Performance

The created software needs to be able to execute the tasks as fast as possible, without any unnecessary delays. Throughout the whole development cycle the functionalities of the code have been tested and modified to ensure that the tasks are executed with the desired speed. During the initial stages of the project it was estimated that the time needed for the execution of each task will be calculated combining the time needed for different parts of the function to finish their execution. In other words, it was assumed that the execution time of most of the main functions can be decreased by making sure that the code does not make any unnecessary calls to other methods or using unnecessary data.

* Security

The designed software needs to implement functionalities which will guarantee that user’s data is stored in a secured way. During development cycle it was decided that the passwords will be stored in an external server. This initial step will provide the necessary level of security for the data since the project is only in its first stable version. Please, check the following sections of the dissertation for more information about the activities planned to increase the security even more.

Apart from stated activities, the security will be further increased by login functionalities implemented in the system. The program will be available only for registered users. This approach will limit the chance that an unauthorized entity will be able to use the application.

* Testability

Testing is an essential part in the development of any project with the size of the one that had to be developed. If proper testing strategies are used, some major and minor bugs can be found easily, without spending unnecessary time on finding the defects.

Testability refers to the degree in which the software can be tested. The way the software is structured suggests that the most critical points of system’s development will be during its connection to the server and the parsing of the JSON arrays received in response. These points have been developed to make possible the usa of the testing strategies listed in the next sections.

* Upgradeability

This non-functional requirement shows in what extent the program will be able to be upgraded. Software modules are created to be upgradable in the future. For example, by making predicted list of groceries accessible in the same the original grocery list is used. This means that it will probably be possible to create different lists of items in the future versions of the software.

By increasing the upgradeability the chances for a future upgrade will be increased too. This is a very important requirement for the project because it will allow the creation of app’s future version with a bigger number of functionalities, if this project’s results are satisfactory.

* Accuracy

This non-functional requirement can be divided into two distinct parts. When accuracy is mentioned, the term can refer to the GPS accuracy closely related to the functionalities showing different shops in the desired location. Also, accuracy can mean the accuracy of the data presented to the client which is closely related to the Grocery functionalities in the code.

For different types of accuracy different strategies will be used in order to guarantee that the data presented is accurate. These approaches will be further discussed in the next sections of the dissertation. Probably, the best way to ensure this is by implementing constant checks of the information delivered to the user. For example, when the internal database is used, its entries will be checked against the external database and any potential differences will be fixed. To guarantee that delivered GPS coordinates are accurate already created Android APIs were used to fetch the coordinates of the user.

## Technical Requirements

This part of the dissertation will explain the tools which were used in project’s development. For the sake of topic’s better understanding a clear definition of the term “tool” has to be made. In the context of this chapter of the dissertation this term will refer to different libraries used during the development. This is an important aspect of this deliverable because it will show how various fixed methods work. It was necessary to use such libraries because some of project’s methods are involved in making complex calls which could lead to huge loses of time if it were decided to implement them from scratch. Another reason why libraries were needed is that some of the send data (balance, bought items, credentials) is sensitive and its safe transportation is a priority. Using already created methods to achieve this guarantees that the data will be delivered safely since the libraries are constantly updated and any potential bugs are eliminated.

#### Android Location

Android GPS is used to fetch user’s geolocation. This is needed in order to present a list of shops in close proximity to the user. As seen in the screenshot of function’s part which involves in the creation of such list asking the user to select a radius of the search. When radius is selected, its integer representation is sent to the next page using Intent (link to intents). When the JSON representation of the shops is sent to the client, each of the entities inside is tested for proximity. This means that their geolocation is checked whether it falls within the radius selected by the user.

In order to achieve this, an already created GPS tracker was used. Its purpose is to find user’s geolocation. This is done by using a class called LocationManager. According to the API this class “provides access to the system location services. These services allow applications to obtain periodic updates of the device's geographical location”. When the GPS tracker is performed, the LocationManager fetches the current position of the user and returns it as a Location object. The longitude and latitude of this object are checked later using the getLongitude(); and getLatitude(); methods which return the representation of the type double of the values. However, if the user has not turn the GPS functionalities on his/her device the last known location of the user will be used for his current location. This fact may cause differences in the returned shops forthe user could have changed his location since the last record of his location.

Before implementing this functionality the usage of Google Play Api was considered. This Api gives the opportunity to fetch the geolocation. However, its implementation involves lots of additional steps, such as registering the application to Google Play and implementing various classes and methods to achieve the desired functionality. This could eliminate the stated problem but it is not considered as a good option for an application of this size. If the project is continued in the future the usage of Google Play Services will be strongly advised because it will greatly increase the number of functionalities provided to the user since this Api allows the usage of already defined functionalities from Google and some of their recorded data in order to increase program’s performance.

#### SQLite

SQLite is used to store user’s data needed for smart functionalities. The API of this database shows it as ”self-contained” and “zero-configurational”. This means that it is very appropriate to store user data in user’s device without any need to connect to a database or to send any data to a server. Furthermore, if we take into consideration the main disadvantage of creating software with client-server architecture, it can be seen that the client is very dependent on the server and that a potential crash in the server might affect the overall performance of the system. However, SQLite gives the opportunity to store data locally which will greatly lower the impact of a potential problem with the server because users will still be able to access the “smart” functionalities in the system but will not be able to issue payments.

In the course of system’s development different tests have been created to check whether the usage of an internal database is better in terms of performance than uploading the information into an internal database. It has been calculated that the size of the stored information is unknown since the fact that the users will buy new items periodically so they will be recorded in the internal database. This means that if the data is stored in an external database, its size will increase proportionally to application users’ number. This will be a very serious problem for future releases. However, saving the grocery information locally guarantees that this will not happen.

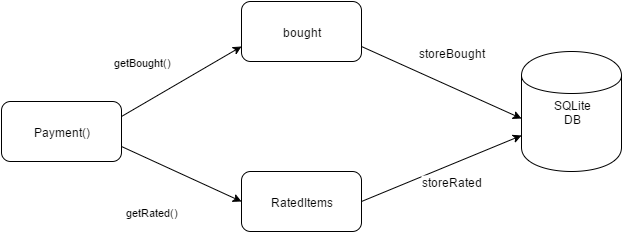
Furthermore, as mentioned in other chapters of this artefact, security is a priority for the system. This means that the way of storing information must be secured. SQLite provides extra layer of security since the data is arranged in a database and kept in user`s device. Even if the server`s security is breached, the data will not be compromised. Also, by doing this, it is guaranteed that anyone apart from the user will not be able to access the data from the local database. However, this might be a problem for future versions since some users might use the app on several devices, so it will not be possible to save the groceries on all of the used devices.

##### Usage

The storage of bought items in user`s device starts once the payment has finished successfully. When this happens, the groceries are extracted from the list. Based on these items different queries for storing the data in the database are created. Once the mentioned are executed the user is notified that the data has been saved in their device and redirected to the home page.

In order to create the suggested list, the already bought items will be fetched from the local database. This function will be executed during the creation of the grocery list for the chosen shop. While the predicted list is generated, the values will be checked against the database so that it can return only the items which are available in the shop. Another reason why the check is necessary is that it will show if some of the values in the local database are different from the ones in the external database. If such a difference is found it will be assumed that the values in the external database are correct. This means that the values in the internal database will be fixed.

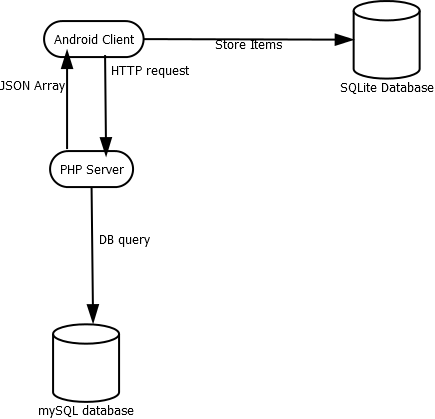
The following diagram shows how the “smart” classes are able to store the selected items in the user database:



# Software Architecture

The software architecture approach implemented in the project is Client-server. For the time being, the PHP server and the database servers share the same host. However, in the future of this project,it might be necessary to move the database server in a different host. This can be easily achieved since the connection between the different layers is done by calls which involve specifying the URL of the targeted file. This means that both layers can be separated and still the code will work only by modifying the called URL.

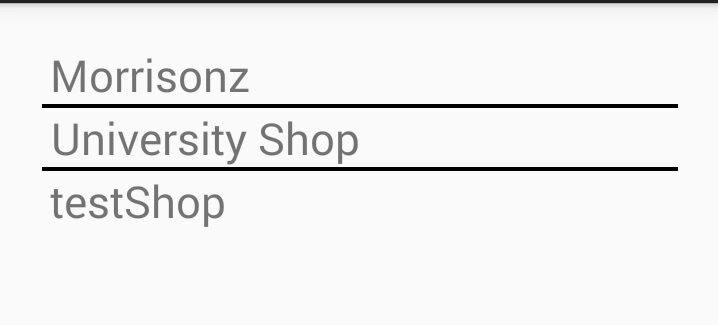
The following diagram will illustrate how the Client-server architecture is implemented in the project. Please, note that the SQLite database is part of project’s database layer even if it is located in user`s device.



## Layers

### Presentation layer

The presentation layer is responsible to deliver the information to the user. In the project, the different pages will be considered a part of this layer. In order to find out more about different operations involved in this layer one of the pages will be inspected, or more precisely, the class called shopList.java. This class is responsible to show the user the list of shops present in the database. Please, note that most of the functionalities listed here are present in other pages with slight differences in terms of returned information. The following screenshot is taken when the stated class is run:



#### AsyncTask classes

The shopList.java is the class which will fetch the shops from the database and will create the list which will be presented to the user. It can be seen from the appendix that the main functionalities such as preparing the JSON array and the population of the JSON objects in the adapter are done in the doInBackground() method. This means that additional thread will be run. The purpose of this thread is to execute a certain task while the screen is active. This method is part of GetShopAsyncTask which is responsible for the creation of the thread to be run on application’s background. The AsyncTask is executed by calling the executeAsyncTask() method which creates a new object of GetShopAsyncTask class and starts its execution.

#### onCreate() method

This is one of the required methods which needs to be implemented in every Android page. This method holds the data which will be initialised during page’s startup. In the presented case this method reads the data inside the shop\_list.xml. After this, the element with id: “Button01” is created on the page. A button listener is attached to this button to call the executeAsyncTask() method, as explained in the previous paragraph. If the button is clicked upon, it will disappear since the .setVisibility method is set to View.GONE

#### getJSONResponceFromURL

This method belongs to the helperHTTP class. This class consists of methods which will make different calls to the different pages in the server. The methods in these classes usually return a string which holds the values of the JSON arrays fetched from the server. In the given example the method returns a String value which has the ID and the name of the shops.

In order to receive the array of shops the method makes a http request to the given link. Once the connection is made, a String is connected to record the response from the server. Once the response is completed the string is returned as an output of the method.

#### parseJSONString() method

This method is used to populate an array of objects with the entries of the JSON string. As seen from appendix 1, this method begins execution with the creation of a JSONArray datastructure which takes the values from the JSON String obtained from the server. JSONArray works in a similar way as the Java Array. A for loop reads its values and creates a new object for every value. In the given example a new shop is created for each value in the array. Once created, the ID and the name of the shop are taken by optString(String,””) methods. These methods take the value of the attributes of the entries in the JSON Array. For example, shop.name=j.optString(“name”,””); will set the name of the shop to the value of the “name” element recorded in the array.

#### Adapter classes

These classes hold the rules which will be applied on the JSON arrays. Appendix 2 shows the source code for the shop adapter. As seen from the code, the adapter needs to extend the adapter which correspond to the data structure which will be populated in it. The main logic in this class is done in the getView() method. This method creates two text bars which will hold the name and the ID of the shop. When this class is created (appendix 1) the getView() method is called for each of the entries in the array, or in other words will be called when a shop is found.

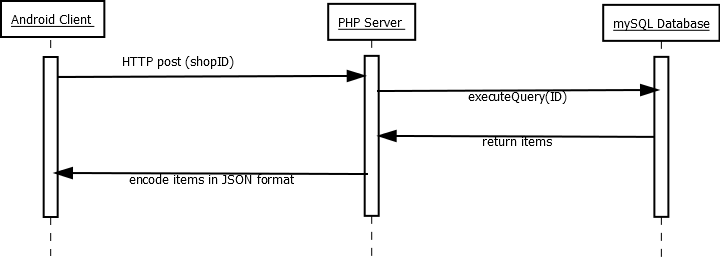
#### onPostExecute() method

This method is also obligatory for every Android page. It will be called after the onCreate() method. In the presented case this method will make sure that the adapter is populated with the values from the JSON array, or, in other words, will guarantee that the entries fetched from the database are organized according to the rules set up in the adapter. This is done by creating a new object from the shopAdapter class and populating it with the entries from the fetched array. Then a new list view is created and the adapter is assigned to it. List view is another Android object which is used to show data to the user in the form of a list.

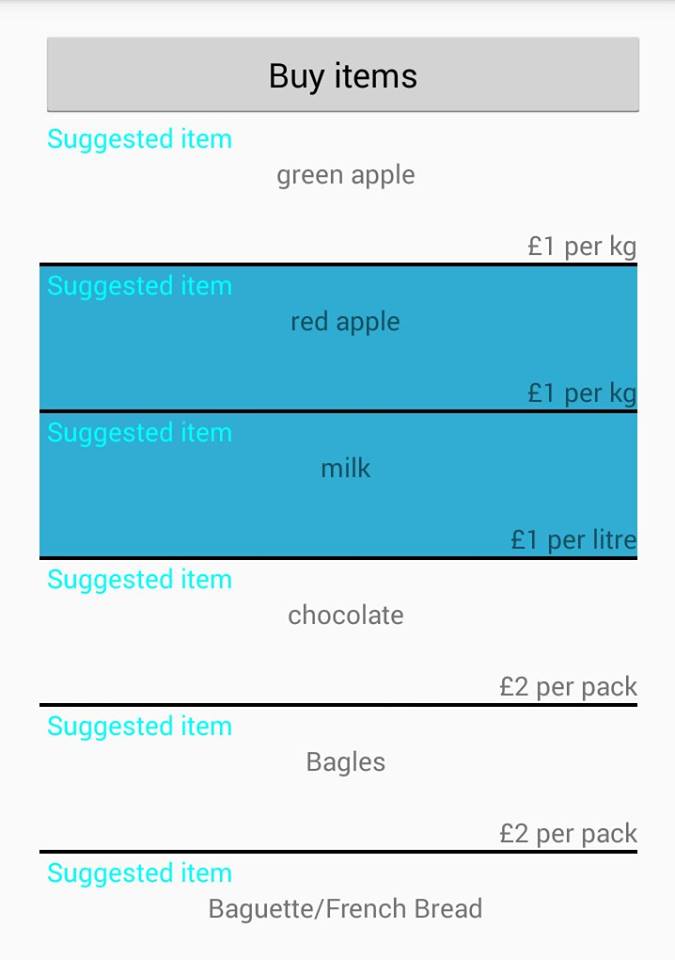
### Business logic layer

This layer will deliver the data needed from the presentation layer. It is physically separated from the presentation layer and the communication between the layers is achieved by making http requests. When the request is created it is sent to the business logic layer where the desired operations are executed. When the data is fetched it is encoded to the JSON format and returned to the client.

The following sequence diagram will show the steps which have to be made in order to ensure that the users will receive the grocery items in the shop they have chosen to buy items from. Appendix 3 contains the script which implements the steps in the sequence diagram

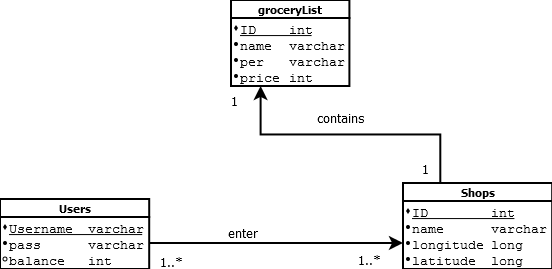


The following screenshot was taken when the activity associated with the functionalities of the php script are executed.



### Database layer

This layer will store the data which will be needed for program’s successful execution. Apart from containing information about the stores and the grocery items, the database contains a record of the users with their passwords and the balance of their accounts. The layer will be reachable only from the business logic layer because this will guarantee that the information will be divided from its presentation. The following diagram will illustrate the design of the MySQL database used in the system.



Apart from the database server which fetches the stated information, this layer theoretically includes the internal database. In order to record and later use the recorded information, the Android Application will manage the needed data by using the methods defined in dbHelper.java class. The following paragraphs will give additional information about the functionalities presented by this class.

#### Create database

In order to create the database used to store the items bought by the user, the onCreate(SQLiteDatabase db) method will be used. It will create the database using String queries similar to the ones used to create the mysql database. The following command executes the stated function:

"CREATE TABLE items ( " +

"name TEXT PRIMARY KEY, " +

"amount INT)"

As seen, this string creates the given table which includes the stated values. In other words, whenever an item is stored in the database, it will have its name recorded as a unique identifier and the amount of bought items will be recorded too. It will be used later to create a list of most frequently bought items present in the selected shop.

#### Create values

This method is initialised once the user buys a new item. It will create a query to the database which will create a new entry for the given item. The entry holds the name of the item as a primary key and also the amount of items bought. For a new item, the initial stored value will be 1. In future releases where the value could be more than one, the specified value of the item will be stored in the value row of the table. This function is achieved by executing the following statement:

"ContentValues values = new ContentValues();

values.put(KEY\_NAME, item.getName());

values.put(KEY\_AMOUNT, 1);

db.insert(TABLE\_ITEMS,

null,

values);"

As seen in this example, the values given to the query are stored in an object called ContentValues. This object’s .put method adds the given data to the ContentValue object. Once this is done, this object is stored to the given table by the db.insert() method.

#### Check for existing items

This method is used to update the value of the amount field for a certain item. This function will be initialised once the given item has records in the database. The following string will be executed by the exists(groceryItem item) method. When launched, it will check whether the given item has a record in the database by executing the following query:

"SELECT \* FROM " + ***TABLE\_ITEMS***+ " WHERE " + ***KEY\_NAME*** + " =?"

This string takes the value stored in the variable table\_items as the name of the table and generates a list of items meeting the requirement stated in the where clause. If the reply returned after the query datastructure is empty, this means that the item has not been added before. As a result, the stated method will return false. However, if the function returns true this means that the item is present in the database.

#### Update values

This method is used to update the value of an item present in the database. It is executed if the method listed above returns true. When this happens public void update(groceryItem item) will be executed. This method takes the given item as an argument and creates the following query which will update the value of the variable recorded in the “value” field:

"UPDATE " + TABLE\_ITEMS + " SET " + KEY\_AMOUNT + " = " + KEY\_AMOUNT +"+1"+ " WHERE " + KEY\_NAME + " =?" , new String[]{name}

This query uses the SET command to update the amount stored in the database and add 1 to its value. In further releases this number can be changed if shopping of multiple items is implemented in project’s functionality.

#### Future plans

In future stages of the project, the later part will be separated in a different database. This will increase the security of the data since it is separated from the rest of the information. Another reason why a separation might be needed is the fact that each database has a fixed number of possible connections which can be made at the same time. This means that if the project is expanded, more and more users will be using the system at the same time. It will be a reasonable decision to divide the databases because thus certain number of users could use some of the functionalities while the rest of them will be logging of. This will highly decrease the traffic and will increase the number of possible connections which could be made at the same time.

## Benefits

This approach of building the software provides the opportunity to create distinct layers for clients and the server. This increases the security of the product because the client will be able to obtain the needed information only by requests. This means that the top layer will not be able to view or manipulate important database queries, which highly decreases the chances for data leaks or modification of important user information. Furthermore, implementing the Client-Server logic could greatly increase project’s scalability, since the data will be kept in servers where new data could be constantly added. Also, the HTTP requests will create a new thread which will be executed in the server side. This provides the opportunity to reuse the already defined server and to provide different functionalities for different users. This means that in future, the server side of the software will be able to hold data for the functionalities provided for different users.

## Disadvantages

Probably, the only disadvantage of using this approach is the fact that software functionalities are greatly dependent on server’s “health”. This means that users will be able to access provided functionalities only if the server is online. This fact hides a lot of future challenges, such as maximizing server’s uptime or securing the server from potential attacks such as DDOS (distributed denial of services). This danger will be noticeable in a future scale increase of the product when lots of users could try to use the server at the same because in a potential DDOS attack the server might get overloaded by hacker requests which will result in a denial of services to clients.

## Solutions

As seen from the previous paragraph, the rapid expansion of the software might cause overload of requests which will affect the overall performance of the program. However, this danger can be reduced by implementing one of the following approaches:

* **Multiple servers**

Different servers can be created to support different user requests. For example, the login functionalities can be physically separated from shop functionalities on a different server. This will greatly increase the amount of requests which can be handled by the particular server. Furthermore, this will limit the impact of particular server downtime.

* **Backup servers**

Separate servers can hold sensitive user information such as credentials or payment details. In this way, even if the server is offline users will be able to access the main functionalities provided by the product. Furthermore, if some information is lost or deleted by malware it can be easily be recovered by these servers.

# Algorithms

This part of the dissertation will concentrate on different algorithms used in the system. Steps of different processes were taken in order to guarantee that produced functions were implemented properly. However, chosen algorithms have problems which had to be solved to guarantee that produced results would be realistic and accurate.

## “Smart” algorithm

This algorithm concentrates on producing a list of groceries sorted by certain criteria. It will be launched after the user buys a certain item. Chosen goods will be stored in the internal database. As mentioned in the “problems” section, accuracy of data is crucial for system’s overall performance. This algorithm will check if the entries in the internal database are the same as the ones shown by the server. By doing this, data accuracy will be obtained because any differences will be fixed. Also, the “Smart” algorithm is appropriate because it will decrease the data saved in the database servers which will immediately result in increasing the amount of connections to the server at a particular time. Moreover, implementing this algorithm will guarantee that user information will be stored on user’s device. This fact will increase the security of the data and will also limit the impact of a future server downtime because the user will still be able to access the lists of popular items even if the server is offline.

### Problems

The main problem which occurred during the execution is the difference between the entries of the internal and the external database. The following scenario happened during functionalities test.

1. The user buys a set of products.
2. User is asked to rate the products and eventually the entries are stored in the internal database.
3. Some grocery items have been deleted or changed.
4. The user enters again and the algorithm returns values different from the ones stored in the database.

The following paragraphs will give more information about the approaches used to solve this problem.

### Solutions

The initial plan of algorithm execution was reshaped in order to achieve a synchronisation between different databases. This was made possible including another step in the algorithm. Before showing the predicted groceries to the user, the algorithm will compare items values in the predicted list and will fix any differences between them. By doing this, future tests results will prove that the values returned to the user will be the same as the ones recorded in the database.

### The algorithm

1. Algorithm is initialised when the payment function has finished.
2. Items from the grocery list are recorded to the internal database.
3. If any items are found the count will be increased by the number of bought items.
4. Whenever the user buys any items again the set of items that he/she had bought will be shown. Before the actual presentation of the list the entries in the internal database will be checked against the entries in the external database. Please, note that the algorithm will show only items available in the shop.

## Location algorithm

This algorithm will concentrate on showing the shops in the given radius. In order to launch this sequence of steps navigation functionalities of the user`s device will be used. The location of the user will be recorded and compared against stored shops’ coordinates. Another purpose of this algorithm is to return any shops in the location that the user has chosen. This will be achieved by returning to the user only the shops which are n miles (where n is the number of miles that the user has chosen) away from the current location.

### Problems

After recording device’s coordinates the user might move. This problem was discovered during one of unit tests when user’s position changed after the recording. It resulted in showing a different array of shops. In other words, the produced array consisted of shops which were in a position different to current user’s position.

### Solutions

This problem was solved by adding a variation to the coordinates sent to the server. In other words, instead of sending n miles the program will send n+m where m is a constant indicating the user’s displacement. In this way, the accuracy of the returned shops will be increased.

### The algorithm

1. The program records user’s coordinates.
2. The user chooses the radius of algorithm’s action.
3. The current position and the radius are sent to the server.
4. The server checks the database and returns shops which are in a radius less than the chosen radius.

# Testing

Testing is a very important part of this size project’s development. Successfully carried tests can eliminate bugs and fix some more serious problems. This is important because if not done properly, it can lead to the release of low quality software. Probably, the most crucial part proper testing strategy design is to identify system’s critical parts. In this way, extra attention can be paid on the behaviour of certain system’s components which are more likely to cause bugs.

## Critical parts

As mentioned in previous paragraphs, the most critical parts of the system tending to create errors are located at the “point of contact” between the different layers. This means that the parts of the code that have to be tested are related to connecting to a different layer.

This assumption shows clearly that such functions fall into the following categories.

### Connection and data fetching methods

These methods will create different connections through the program. For example, the method connects the php page with the database. A good example of similar function usage can be seen in Appendix 4, or more precisely shopList.php this method is extremely important for system’s overall functionality because it provides access to the server to read data recorded in the database. If this function fails, the connection will not be made. This scenario will result in a crash of the php page which may lead to some serious problems such as serious failures in the corresponding android page. Another method which is of great importance in the class shown in Appendix 2 is … which creates the database query. The results of this query need to be carefully tested since their accuracy is of crucial importance for the Android application. If the returned values are incorrect, the user will see incorrect representation of the shops in the database. If this happens some serious issues might appear such as inconsistency of data and poor overall performance.

Another example of such method is ….. . It is located in the shopList.java class. As seen in Appendix 1, the function will return a String containing a JSON array with the shops in the database. The method getJsonResponseFromURL() is part of the HelperHttp class which creates the http requests and returns the response in the form of a String containing the results. The successful execution of this method is of crucial importance for the overall performance of the Android app. If it fails to execute, the execution of the page will terminate which will lead to a crash. If the results are incorrect, different from the original data will be shown to the client which will lead to some serious problems such as decreasing of program’s performance and if this happens in a future release it will result in great financial losses.

### Data parsing methods and presentation classes

Methods which fall in the Data parsing category are located in the Android application. They take the given JSON Array and create an array of objects corresponding to the input. The method parseJsonString() can be seen in Appendix 1. Its successful implementation guarantees that created array of Shops will be the same as the data recorded in the JSON String. However, its unsuccessful execution will cause some severe failures in the system.

Presentation classes are responsible for the successful presentation of the data returned from the server. If they are implemented correctly the data will be shown to the user in a clear and easily understandable way. However, any failures in the execution of these methods will result in the generation of difficult for understanding or even unreadable data. The class shopAdapter.java seen in Appendix 2 generates a list view with the already defined rules. This list view will take the data from the already generated list of Shops and will output the templated shops to the user (Appendix 1). If the method is implemented successfully the User will be able to view the information in a clear and understandable way. However, if this does not happen the information returned to the user will not be accurate.

## Testing strategy

The chosen testing strategy needs to guarantee that listed methods and classes are properly tested. In order to do this black box testing will be used. Testing strategies of this type take into consideration tested parts as a whole subsystem without considering the code. Choosing black box testing was a hard decision because white box testing was a very good alternative. However, it was estimated that black box will be the better option since it will not consume that much time. This will be achieved by neglecting the code and by judging whether produced outputs will correspond to the desired outputs. The decision whether the outputs are correct will be made by an Oracle. The Oracle will check whether tested functionality behaves the way it should. This will be achieved by comparing the desired result to the result generated by calling the particular class. If the results are the same it will be accepted that the class behaves the way it should. Unfortunately, the method mentioned above will not work for the presentation classes because their results cannot be compared to anything because the “outputs” will be only graphical. However, another strategy is used to check whether generated results are correct. When this bit of code is run a screenshot will be made. This picture will be compared to a visual representation of the expected output. If the pictures are similar to some degree (more than 80% similarity) it will be assumed that the test has been passed.

### Procedure

In order to test the above mentioned system’s critical components in the above mentioned way the following points will be taken into consideration while generating a test case:

1. The selected function will be launched with the test values.
2. The output of the test will be recorded in a variable.
3. The expected output will be generated and recorded to a different variable.
4. The Oracle will be created.
5. The output of the selected function will be checked against the desired output.
6. If the values are the same the test will be considered successful.
7. If the code breaks during some of the tests it will be assumed that the method should be fixed.

As mentioned above, presented procedure will not allow the execution of classes falling in “Presentation Classes” category. In order to guarantee that these parts of the code will be properly tested the following procedure will be implemented:

1. An instance of the selected class will be created.
2. A test JSON array will be assigned to that instance.
3. The code will be launched and a screenshot will be made for its output.
4. The JSON array will be modified in the desired way.
5. The output of 4) will be visualized.
6. Both screenshots will be compared.
7. If they are more than 80% similar it will be assumed that the test is valid.
8. If the similarity is less than 80% the different parts will be fixed.

### Test cases

In this section of the assignment some of the generated test cases will be shown and their results will be discussed. Tested classes can be seen in Appendices 1, 2, 4.

#### grocery.php

As an output, this script returns a JSON array representing grocery items associated with the given shop. In order to test this, a new grocery list was added to the database with items called “testOne” , “testTwo”, and “testThree”. This function’s expected behaviour is to return mentioned items as well as the attributes connected to them. After creating the grocery list manually the script returned the following result:

[{"id":"1","name":"testOne","price":"3","per":"kg","comment":"this is test one"},{"id":"2","name":"testTwo","price":"4","per":"litre","comment":"this is testTwo"},{"id":"3","name":"testThree","price":"5","per":"pack","comment":"this is testThree"}]

The given result shows that script’s output is the same as the desired output. Therefore, it can be assumed that the script works properly.

#### shopList.php

As an output, this class returns a JSON array representing the shops found in the database. In order to test this, a new shop called “abcShop” was added to the database. The expected output of calling this class will be a list of shops including “abcShop”. After manually adding the new shop in the database, the script returned the following result:

[{"id":"1","name":"myShop","longitude":"-1.1328150033950806","latitude":"52.61977767944336"},{"id":"2","name":"otherShop","longitude":"-1.12639","latitude":"52.619736"},{"id":"3","name":"testShop","longitude":"46.869107","latitude":"-18.766947"},{"id":"4","name":"abcShop","longitude":"0","latitude":"0"}]

The highlighted field shows that the shop was indeed created. This means that the script behaves the way it should.

#### shopList.java

This class is part of presentation classes listed in the paragraphs above. Its purpose is to return the list of shops found in the database. This is achieved by sending an HTTP request to the server or more precisely to the script called shoplist.php. After receiving the JSON array from the server this class extracts the data from the array and populates an instance of the shopAdapter class with the extracted information. In order to test the behaviour of this class the application is launched. The data returned by this class is compared to the data found in the shops table in the database or in other words, the output of shopList.php given in the previous paragraph. The following screenshot is taken after running the Activity associated with the class:



Obviously, the result of running the activity is the same as the contents of the database or, more precisely, the data returned from shopList.php. Therefore, it can be assumed that the behaviour of the class is the same as the expected behaviour.

# Critical Appraisal

Project’s original aim was to create an Android application allowing its users to choose a shop in their preferred radius. Also, the application had to produce a list of items which could be found in the selected shop. Furthermore, based on past purchases the program would make predictions and give the user a suggestion which item could be interesting and worth buying. Stated requirements can be seen in the developed application. However, developed application is able to give predictions only based on past purchases without taking into consideration health values of the items in the particular shop. This functionality was dropped from the requirements stated in previous chapters because it was estimated that its functionalities will not be appreciated that much by the users. Another reason for this decision was the fact that I was not familiar with Android development which resulted in extending the period for research. Also, the produced application will not allow its users to rate bought products. This functionality was dropped from the requirements list because it was estimated that it will not be widely used by the users since most of the customers do not rate the products they buy. These drawbacks to project’s initial aims can be fixed during further stages of this application’s development application.

My initial research on shopping applications for Android showed that this field is not entirely developed. Most of the existing applications are either mobile versions of web applications or do not include the functionalities developed in the project. An example of such functionality is the implementation of a local database storing user’s past purchases. Another example is that “SmartShopping trolley” is prone to updates. This means that in future versions it will be easy to implement the functionalities listed in the previous paragraph and also to develop new ones based on the already created functions. This cannot be seen in the existing Android applications which do not fall in the category of web app mobile version.

There are numerous benefits of continuing this application’s development. Probably, the biggest one is that users will be able to shop from outside the shop and collect their groceries. This could drastically reduce the queuing in shops. Also, some future developments can allow a group of users to shop sharing the price of the bought items. This functionality can change the way people shop since the presented idea is absent in all of already created apps.

Technologies used in this application’s development (PHP for server side and JSON for encoding and data parsing) are widely used by large companies. The way of both technologies’ usage by large companies is similar to the way I have used them to develop the desired functionalities. This means the created application can deliver high quality services to its users. Also, in the future, some companies could be interested in created application and integrate it into their own apps. Another benefit of the similarity of technologies implemented in my application and larger apps is that some of the already created technologies by companies can be used to improve the services of “SmartShopping trolley”. An example of such improvement can be integrating using a Facebook account to log to the system. This will allow the users of my application to use an already created account and to share some of the offers with their friends.

The project helped me to gain knowledge in areas such as Android development and a creation of a server using technology like PHP and encoding information using JSON. As stated above these technologies are trending in the IT world and will greatly help me in my future career. Apart from the technical knowledge this project helped me to understand in a greater detail how an application of this scope should be developed, as well as increase my knowledge in different steps of software development cycle. Furthermore, working on this project I gained confidence which will help me a lot in my future career. Last but not least, this project helped me to develop useful skills in order to tackle difficult problems in areas I did not have a lot of a previous experience in, such as Android development.

In conclusion, I can clearly say that this project taught me a very important lesson which will help me to advance in my future career. Also, I can say that I am pleased with the developed application regardless stated drawbacks.

# Bibliography

1. "Client Server Network: Advantages and Disadvantages." ~ I Answer 4 U. Web. 3 Dec. 2015.
2. "Concepts of Three-Tier Architecture." Ali Tarhini. 22 Jan. 2011. Web. 3 Dec. 2015. <https://alitarhini.wordpress.com/2011/01/22/concepts-of-three-tier-architecture/>
3. "JSON Tutorial." JSON Tutorial. Web. 4 Dec. 2015. <http://www.w3schools.com/json/>
4. "SQLite Home Page." SQLite Home Page. Web. 5 Dec. 2015. <https://www.sqlite.org/>
5. “Android GPS, Location Manager Tutorial”. Ravi Tamada.1 July 2012. Web. 5 Dec.2015 <http://www.androidhive.info/2012/07/android-gps-location-manager-tutorial>
6. “How to connect to MySQL database using PHP”. Web. 3.Oct.2015 <http://webcheatsheet.com/php/connect_mysql_database.php>
7. “JSON Tutorial”. W3Schools. Web 10 Dec.2015 <http://www.w3schools.com/json/>
8. “The beginners guide to Location in Android” Wed. 4 Dec.2015” <http://blog.teamtreehouse.com/beginners-guide-location-android>
9. “Using lists in Android”. Lars Vogel.29 Sept 2015. Web. 29 Nov.2015 <http://www.vogella.com/tutorials/AndroidListView/article.html>
10. *Simple SQLite Database Tutorial*. Accessed 15 Jan. 2016.<http://hmkcode.com/android-simple-sqlite-database-tutorial/>
11. Timothy Lethbridge (2004) Object-oriented Software Engineering: Practical Software Development Using Uml And Java, : Mcgraw-Hill College.

# Appendix

## Appendix 1 shopList.java

**private** **class** GetShopAsyncTask **extends** AsyncTask<Hashtable<String,String>,Void,String>{

**protected** String doInBackground(Hashtable<String,String>... params) {

Hashtable ht=params[0];

Stringjson=HelperHttp.*getJSONResponseFromURL*("http://192.168.0.3:1234/uniproject/shopList.php", ht);

**if**(json!=**null**) parseJsonString(shopList,json);

**else**{

**return** "No shops found";

}

**return** "SUCCESS";

}

**protected** **void** parseJsonString(ArrayList<Shops> shopList,String json){

**try** {

JSONArray array=**new** JSONArray(json);

**for**(**int** i=0;i<array.length();i++){

JSONObject j=array.getJSONObject(i);

Shops shop=**new** Shops();

shop.Shop\_id=j.optString("id","");

shop.name=j.optString("name","");

shopList.add(shop);

}

} **catch** (JSONException e) {

e.printStackTrace();

}

}

**protected** **void** onPostExecute(String result){

shopAdapter adapter=**new** shopAdapter(shopList.**this**,R.id.***text1***,shopList);

**final** ListView listv=(ListView)findViewById(R.id.***lv***);

listv.setAdapter(adapter);

}

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***shop\_list***);

((Button)findViewById(R.id.***Button01***)).setOnClickListener(**new** View.OnClickListener(){

**public** **void** onClick(View v) {

executeAsyncTask(); ((Button)findViewById(R.id.***Button01***)).setVisibility(View.***GONE***);

}

});

}

**private** **void** executeAsyncTask(){

Hashtable<String,String> ht=**new** Hashtable<String,String>();

GetShopAsyncTask async=**new** GetShopAsyncTask();

Hashtable[] ht\_array={ht};

async.execute(ht\_array);

}

}

## Appendix 2 shopAdapter.java

**public** **class** shopAdapter **extends** ArrayAdapter<Shops>{

**private** Context context;

ArrayList<Shops> dataObject;

**public** shopAdapter(Context context, **int** textViewResourceId,

ArrayList<Shops> dataObject) {

**super**(context, textViewResourceId, dataObject);

**this**.context=context;

}

**public** View getView(**int** position, View convertView, ViewGroup parent) {

View rowView=convertView;

**if**(rowView==**null**){

LayoutInflater inflater = (LayoutInflater) context

.getSystemService(Context.***LAYOUT\_INFLATER\_SERVICE***);

rowView = inflater.inflate(R.layout.***row\_layout***, parent, **false**);

}

TextView textView = (TextView) rowView.findViewById(R.id.***text1***);

TextView textView1 = (TextView) rowView.findViewById(R.id.***text2***);

textView.setText(""+getItem(position).name);

textView1.setText(""+getItem(position).Shop\_id);

**return** rowView

}

}

## Appendix 3 grocery.php

<?php

$hostname\_localhost ="localhost";

$database\_localhost ="uniproject";

$username\_localhost ="root";

$password\_localhost ="";

$localhost=mysql\_connect($hostname\_localhost,$username\_localhost,$password\_localhost)

or

trigger\_error(mysql\_error(),E\_USER\_ERROR);

mysql\_select\_db($database\_localhost, $localhost);

$ID = $\_POST['ID'];

$SQLCommand = "SELECT \* FROM grocerylist".mysql\_real\_escape\_string($ID);

$result=mysql\_query($SQLCommand);

$resArray = array();

$index = 0;

while ($row = mysql\_fetch\_assoc($result))

{

$resArray[$index] = $row;

$index++;

}

echo json\_encode($resArray);

?>

## Appendix 4 shopList.php

<?php

$hostname\_localhost ="localhost";

$database\_localhost ="uniproject";

$username\_localhost ="root";

$password\_localhost ="";

$localhost = mysql\_connect($hostname\_localhost,$username\_localhost,$password\_localhost)

or

trigger\_error(mysql\_error(),E\_USER\_ERROR);

mysql\_select\_db($database\_localhost, $localhost);

$SQLCommand = "SELECT \* FROM shops ";

$result=mysql\_query($SQLCommand);

$resArray = array();

$index = 0;

while ($row = mysql\_fetch\_assoc($result))

{

$resArray[$index] = $row;

$index++;

}

echo json\_encode($resArray);

?>