# 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 for storing of data and for its distribution 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 the database of the system where data will be stored
* Creation of locally stored data which will store some of the user data
* Creation of session variables which will be used to implement the payment functionalities
* Navigation which will allow 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 create requirements for the system
* Breaking down the approved requirements and prioritizing them
* Production of different documents which were crucial for the successful implementation of the project

# Survey of Information Sources

During the initial phases of the project a list of the main areas which had to be researched about the topic was created. These topics had to be involved with extra attention because they were unknown for me.

## Android development

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

* XML document – which records the information about the different shapes and buttons used in the main pages
* Adapters – helper classes which group similar set of data in one place and applies some rules for their look. These classes were used for the creation of the lists of shops and the grocery lists for every shop.
* Listeners – methods which record user activity. Since Android devices are with touch screens it was assumed that Android offers some listeners which are designed for touch screens. Such listeners were used for giving extra information to the users about a desired product

Apart from the 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 the main parts of the Android programs it was easy to understand the syntax. However, Android programming has its differences like

* Intents which record every window in the program
* OnCreate() methods. These methods are executed when the page is created. They are used to create buttons which hold different functionalities
* DoInBackground() methods – these methods are closely connected to Threads in Java. They allow the execution of assyncTasks which are executed on the background. They are used in the establishing the connection with the server and to exchange data with it. Also in this methods the parsing of JSON arrays is achieved.

## Requests and data parsing

In order to implement the 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 was 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 the crucial parts of the system – parsing the data in a datatype. During my studies in 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 a ore simple way to encode data. Different entries are separated in { } and the different elements are separated by commas (,). This way of visualizing the different entries increased my overall understanding on the topic of data types.

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

## Server

The server accepts the http requests made from the user. During the communication (which will be explained in detail in the later paragraphs) the request is matched to a URL which holds a php file. This file holds the necessary functionalities which the user requests. After contacting the database the php file returns an encoded JSON array which holds the values of the particular query. 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 of the MySQL database server with the given php page.

### Why PHP?

Java was a 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 in this way I will 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 for choosing 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 in different servers which can greatly increase the size of the project and also its usage.
* PHP is easier for learning. 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 the other parts of the development of the system. If Java was used more time would be spend on developing the server creating the different servlets and implementing the functionalities. Instead by using PHP a script had to be created for the given functionality without worrying about the actual structure of the server.

# Requirements

In order to ensure that the quality of the delivered documentation and the code that comes with it a requirements document had to be created. The purpose of this artefact is to show what functionalities does the software have and will also point out how the software would behave. These two distinct categories of requirements will be listed in the functional and nonfunctional 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 then carefully analysed it is guaranteed that the view for the overall functionality of the program will be clearer. Furthermore, by collecting the requirements , different tasks will be produced which will be prioritized and arranged in a way that will guarantee that the functionality of the produced software is 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 performs exactly what is required from it to perform and will outline any possible differences between the desired and the actual functionality. These difference will be used later in the Testing and debugging section of the dissertation.

## Strategy for collecting requirements

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

* During the implementation of the functionalities listed in the following subsections parts of the dissertation the priority of some of the requirements changed because of doing work on the other modules from my degree. The development of some of the functionalities was postponed because it was assumed that the results from their successful implementation will not correspond to the time spend on implementing them.
* The details about some of the requirements changed during the development process. This was due to the fact that the view about the overall performance of the software changed couple of times so a change of the requirements was needed too.
* Some of the 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:

* ID of the requirement – This field records an unique identifier for each of the requirements. In this way it is assured that the requirements can be traced or grouped in different groups regarding their functionalities.
* Description – a short abstract which will outline the requirement.
* Rationale- A short paragraph which outlines why this requirement is important.
* Dependency- This field records the ID of the requirements that need to be implemented before implementing the given requirement. By doing this different developing scenarios were considered which decided which sequence of developing of requirements 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 of the values had a 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 with a level of security against unauthorized access

Fit-criterion – unit test cases which will be launched during testing. Various of attempts will be made to access the system. The suggested behavior is that 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 have 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 the calculations in the payment 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 do be made in the future such as searching for a shop

Fit criterion – unit test cases will be launched which will check whether the number of the shown shops is the same as the number of 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 the 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 which check whether the generated shop/shops correspond to the 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 the payment functionality is strongly dependent to this one.

Fit criterion – Unit test cases will be launched which will add money to the test account. The changes will be recorded to the database. If the obtained values are the same as the predicted values 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 a suggested items for 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

* Output the 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 input 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 of the 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 as 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, by 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 which will record the test values in the session. Different activities such as payment will be done. At the end of each activity the values in the session will be updated. In the end the session variables will be compared to the expected values. The requirement will be considered as implemented if the values are the same

User satisfaction: 3

User dissatisfaction: 4

Priority: Medium

* Recommended items need to be shown on the user based on his/her taste

Requirement ID: 11

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

Rationale: This functionality will increase the performance of the system. If its implemented properly the system will be able to provide a list of items which can be bought for the specific shop. This will make the work with the system faster since most of the users tend to buy the same products

Fit criterion: Unit tests will be created which will 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 from the user. The requirement will be considered as 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 is another category of requirements which is 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 for measuring 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. This will be achieved by creating as few elements in each window as possible. By doing this it will be ensured that only the most important information will be shown to the user. The presentation of the data in the different elements in the pages will be done by using templates. This will guarantee that the way of presenting of the data will be the same in every page.

Also, user 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 needs to long press on the element and a message will appear. By implementing various event listeners it will be ensured that the information will be presented in an interactive way which will undoubtedly will enhance the user experience. Another way to increase the user 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 the User Interface. For example, the positioning of the buttons will be the same as the way buttons are shown in similar applications. This will help the user to get familiar with the program faster which will definitely make the user happy to use the software.

* Performance

The software created 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 is calculated by combining the time needed for the 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 the user data is stored in a secured way. During the 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 planned activities which will increase the security even more.

Apart from the stated activities security is furtherly increased by the login functionalities implemented in the system. The program will be available for use only from 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 of structure of the software suggests that the most critical parts of the system will be during the connection to the server and the parsing of the JSON arrays received as a response. These parts of the system have been developed in such a way that the usage of the testing strategies listed in the next sections are possible.

* Upgradeability

This non-functional requirement shows in what extend the program will be able to be upgraded. The modules created in the software are created in such a way that a future upgrade will be possible. For example by developing the predicted list of groceries the way to access the original grocery list was used. This means that probably, different lists of items will be able to be created 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 a future version of the app with more functionalities if the results from this project are satisfactory.

* Accuracy

This nonfunctional requirement can be divided into two distinct parts. When accuracy is mentioned the term can refer to the GPS accuracy which is closely related to the functionalities which show 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 the different types of accuracy different strategies will be used in order to guarantee that the data presented is accurate. These approaches will be furtherly discussed in the next sections of the dissertation. Probably the best way to ensure that this is true is by implementing constant checks of the information which is delivered for 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 the delivered GPS coordinates are accurate already created Android APIs were used to fetch the coordinates of the user.

# Software Architecture

The software architecture approach implemented in the project is Client-server.This means that the user of the Smart Shopping Cart will connect to the Server and will send HTTP requests for the necessary information. After that the server will fetch the information from the database using queries. The data collected from the queries will be encoded as an array of JSON objects and returned to the Client side. From there the JSON arrays will be decoded and parsed to the appropriate classes as array of objects.

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 in the specifying of the URL of the targeted file. This means that both of the layers can be separated and still the code will work only by modifying the called URL.

## 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 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, by implementing the Client-Server logic the scalability of the project can be greatly increased. This is true because the data will be kept in servers where new data can 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 can hold data for the functionalities provided for different users.

## Disadvantages

Probably the only disadvantage of using this approach is the fact that the functionalities of the software are greatly dependent to the “health” of the server. This means that users will be able to access the provided functionalities only if the server is online. This fact hides a lot of future challenges such as maximizing the uptime of the server or securing the server from potential attacks such as DDOS (distributed denial of services). This danger will be noticeable in a future increase of the scale of the product when lots of users can be 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 for supporting different user requests. For example, the login functionalities can be physically separated on a different server from the shop functionalities. This will greatly increase the amount of requests which can be handled by the particular server. Furthermore this will limit the impact of a particular server downtime

* **Backup servers**

Separate servers can hold sensitive user information such as credentials or payment details. In this way even if a 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 the different algorithms used in the system. Steps of different processes were taken in order to guarantee that the produced functions were implemented properly. However, the chosen algorithms have problems which had to be solved to guarantee that the produced results are realistic and accurate.

## “Smart” algorithm

This algorithm concentrates on producing a list of groceries which is sorted by certain criteria. It will be launched after the user buys a certain item. The chosen goods will be stored in the internal database. As mentioned in the “problems” section accuracy of data is crucial for the overall performance of the system. 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, by implementing this algorithm it will be guaranteed that user information will be stored on the device of the user. 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 execution is a difference between the entries of the internal and the external database. The following scenario happened during one of the test of the functionalities.

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 of the grocery items have been deleted or changed
4. The user enters again and the algorithm returns different values 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 execution of the algorithm was reshaped in such a way that a synchronisation between the different databases was achieved. This was possible by including another step in the algorithm. Before showing the predicted groceries to the user, the algorithm will compare the values of the items in the predicted list and fix any differences between them. By doing this the results from the future tests approved that the values returned to the user are the same as the values 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 items are found the count is increased by the number of bought items
4. Whenever the user buys 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 which are 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 the navigation functionalities of the user`s device will be used.The location of the user will be recorded and compared against the coordinates of the stored shops. Another aspect of this algorithm is to return any shops in the location that he user has chosen. This will be achieved by returning to the user only 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 the coordinates of the device the user might move. This problem was seen during one of the unit tests when the position of the user changed after the recording. It resulted in showing a different array of shops. In other words, the produced array consisted of shops which were from a different position to the current position of the user.

### Solutions

The way that this problem was solved was to add 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 which indicates the displacement of the user. In this way the accuracy of the returned shops will be increased.

### The algorithm

1. The program records the coordinates of the user
2. The user chooses the radius of action of the algorithm
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

# Tools

This part of the dissertation will explain the tools which were used in the creation of the project For the sake of better understanding of the topic 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 the different libraries which were used during the development. This is an important aspect of this deliverable because it will show how various of the fixed methods work. There was a need for the usage of such libraries because some of the methods in the project involved in making complex calls which could lead into huge loses of time if it was 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. By using already methods to achieve this it is guaranteed that the data will be delivered safely since the libraries are constantly updated and any potential bugs are eliminated.

## SQLite

SQLite will be used to store the user data needed for the smart functionalities. According to 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 the device of the user without the need to connect to a database or send 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 to 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.

During the course of development of the system different tests have been created to check whether the usage of an internal database is better in terms of performance than uploading the information in 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 the number of users who are using the application. This will be a very serious problem for future releases. However, by saving the grocery information locally, it is guaranteed that this will not happen.

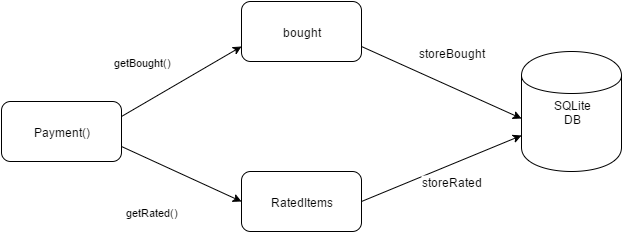
Furthermore, as mentioned in the other chapters of this artefact, security is a priority for the system. This means that the way of storing information must be secured. SQLite provides extra layer of security since the data is arranged in a database and kept in the user`s device. Even if the server`s 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 the 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 will be able to store the selected items in the user database:



## Google location API

This is a set of libraries created by Google to present the necessary functionalities in Navigation and most importantly for the project- location services. The methods from the API obtain the coordinates of the device which will be used later by the “Location” algorithm explained in the previous sections. The reason for choosing this API was its simplicity.

Before the actual implementation of the code the following steps had to be executed:

1. **The functionalities are called in a new activity which implements the Google maps activity. By implementing the already created activity most of the coding part is done by the Eclipse IDE. This activity consists of a map fragment which shows the position of the user and the nearby shops on a mini map.**
2. **In order to use the functionalities provided from the API “Smart Shopping Trolley” .was registerd in Google Play Services.This means that an unique API key was obtained.**
3. **The provided key was added to google-maps-api.xml which collects all the important options for this functionality**
4. **Permissions for getting the current location of the users were added to the manifest.xml file.**

As seen from the listed points the application had to be registered to the Google Play Services. This allowed the application to implement the functionalities. However, this was not enough to guarantee that the functionality works properly. The manifest.xml file contains all of the valuable information about the Android project. Before any functionality is used in the program, permissions have to be gained. This is achieved by modifying the values in the .xml file. More information about this file is provided in the next section of the dissertation.

### Usage

Here outline how the API is used

# Implementation

As mentioned in the previous sections “Smart Shopping Trolley” implements the client-server architecture. Furthermore, the program is divided into three layers with different functionalities. This section of the dissertation will outline the distinct layers. The following paragraphs will explain more about the structure of the software outlining the three distinct layers and explain their functionalities by outlining the main methods in each of the layers

## Presentation layer

The presentation layer is responsible for delivering the information to the user. In the project the different pages will be considered as a part of this layer. In order to understand more about the different operations which are involved in this layer one of the pages will be inspected. Please, note that most of the functionalities listed here are present in the other pages with slight differences in terms of returned information.

### AsyncTask classes

The shopList.java is the class which will fetch the shops from the database and create a 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 do execute a certain task while the screen is active. This method is part of the GetShopAsyncTask which is responsible for the creation of the thread which will be run on the background of the application. The AsyncTask is executed by calling the executeAsyncTask() method which creates a new object of the GetShopAsyncTask class and starts its execution.

### onCreate() method

This is one of the required methods which need to be implemented in every Android page. This method hold the data which will be initialised during startup of the page. 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 which will call the executeAsyncTask() method explained in the previous paragraph. If the button is clicked 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 this 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 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 under 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.

Appendix 3 shows the code for the grocery.php file which returns the grocery list of the selected shop. The beginning of the php script begins with the creation of variables which hold the information needed in order to connect to the database. Then a connection to the database is made using these variables. The connection is established by *mysql\_select\_db($database\_localhost,* *$localhost);* method, where the inputs correspond to the already mentioned variables

The id number of the selected shop is obtained by getting the value from the http request. This extraction is achieved by the following line of code: *$ID = $\_POST['ID'];*

When the id is obtained a query to the database is made. The following line of code implements the database search: *$SQLCommand = "SELECT \* FROM grocerylist".mysql\_real\_escape\_string($ID);* Probably the most important part of this query is  *.mysql\_real\_escape\_string().* This command appends the ID to the search. In other words, the query will select the items from a table named *grocerylistID.*

Once the query has been executed a while loop is created which will carry on recording the received values to an array called *$resArray*. Once the recording of the data has finished the array will be encoded to the JSON syntax using the echo *json\_encode($resArray);* function.

## Database layer

This layer will store the data which will be needed for the successful execution of the program. 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 in this way it will be guaranteed that the information will be divided from its presentation.

The data is divided in different tables which correspond to the different entities in the database. Each member of the tables have a unique identification number which will be used in some of the functionalities in the system. The items which are available in each shop are separated in a different table. The names of this tables consist of the ID of the shop. In this way it can be seen which list of items correspond to a given shop. Furthermore, by doing this it was possible to implement the query listed in the previous step. Its successful implementation greatly increased the performance of the system since obtaining of the appropriate grocery list is possible by knowing the unique ID of the shop.

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 in this way users can use some of the functionalities while the rest of them are logging of. This will highly decrease the traffic and will increase the number of possible connections which can be made at the same time.

# Testing

Testing is a very important part of the development of a project with this size. The 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 a low quality software. Probably, the most crucial part of the design of a proper testing strategy is identifying the critical parts of the system. In this way extra attention can be paid on the behavior of certain components of the system which are more likely to cause bugs.

## Critical parts

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

From the assumption made it can be clearly seen 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 the usage of a function like this can be seen in Appendix 2, or more precisely ……… This method is extremely important for the overall functionality of the system 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 from 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 happen some serious issues might appear such as inconsistency of data and poor overall performance.

Another example for such a method is ….. . It is located in the shopList.java class. As seen from 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 the performance of the program and if this happens in a future release it will result in a 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 which correspond to the input.The method parseJsonString() can be seen in Appendix 1. Its successful implementation guarantees that the 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 right the data will be shown in a clear and easy for understanding way to the user. 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 the listed methods and classes are properly tested. In order to do this black box testing will be used. Testing strategies from this type will take into consideration the tested parts as a whole subsystem without caring for the code. Choosing black box testing was a hard decision since the fact that white box testing was a very good alternative. However, it was estimated that black box will be the better option because it will not consume that much time. This will be achieved by neglecting the code and judging whether the produced outputs will correspond to the desired outputs. The decision whether the outputs are correct will be made by a specially designed class which have the functionalities of an Oracle. The source code of a class like this can be seen in Appendix 4. An instance of this class is created every time a test runs. By calling the compare() method the oracle will show if the returned values are correct by generating a Boolean variable. If the returned value is true it will be assumed that the test has been successful.

Unfortunately, the method mentioned above will not work for the presentation classes because their results will not be able to be compared to anything. This is so because the “outputs” will be only graphical. However, another strategy is used to test whether the generated results are correct. When the 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 already mentioned critical components of the system by the way mentioned above 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 is created
5. .Compare(output1,output2); function is called
6. If the values are the same the test will be considered as unsuccessful
7. If the code breaks during some of the tests it will be assumed that the method should be fixed

As mentioned above the presented procedure will not be possible for execution on the classes which fall in the “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 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 of the 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. The methods which are tested can be seen in Appendices 1,2,3. The name of these methods are as follows:

**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);

?>