BuySafe
Summary report

Version 1.0

Table of Contents

Executive Summary

BuySafe is an application that informs customers about the food related products that they want to buy. It is be able to inform customers about harmful contents that might cause health problems, taking into consideration conditions or allergies that a person might have. The application will warn users not to buy products that might harm them according to product content and preconfigured ingredients watch list. Furthermore, customers will be informed if there is recall history or safety incident reports linked to the manufacturer that produces the products they want to buy.

The project team consisted of a group of students enrolled in Distributed Software Development course (DSD) which is executed in cooperation between MDH-IDT[6] and UZG-FER[7]. Three team members who were working on the project were located in Croatia and three were located in Sweden. The course has a successful history in leading such projects, despite the geographical separation between the team members.

Considering the geographical distribution of team members, special attention was given to communication organization. The project team had regular meetings with all the team members attending.

The outcome of this project is the BuySafe system that consists of a server application and an Android mobile application. The system functions as a typical client-server where the Android mobile application is the client and server application is the Java server that returns the data about products gathered by the parsers. Parsers gather data from multiple free data sources like Amazon[3], FoodFacts[1] and RAPEX[2].

1. Introduction

This paper starts with the problem statement stated in *Section 2* and the project scope in *Section 3,* followed with *Section 4* in the management plan has been presented. The *Section 5* describes the project plan which we have been following during the development. The development process has been described in the *Section 6.* The requirements specifications have been described in the *Section 7.* Section 8 discusses architectural design and *Section 9* the implementation methods. Validation and verification of the project has been described in *Section 10* and the outcomes of the project are described in the *Section 1. Section 11* and *12* state what we have learned and summarize the whole paper.

2. Problem statement

News and reports of companies recalling their products from the market as they pose a certain health risk to the customers are becoming ever frequent. The customers became more cautious when buying products trying to avoid possible health problems that the products could cause or simply trying to buy quality products, despite the fact that there are many organizations and associations concerned with the protection of consumer rights and monitor the quality and safety of the products, for the customers it is mainly difficult to get the desired information, furthermore they aren't even introduced to the fact that this information exists.

Currently there are no adequate applications for mobile devices that would offer customers the option of checking products according to their content and warning them of possible negative effects on their health. Current applications help shoppers compare product by price and inform them on the contents of the products. Such applications are RedLaser[8] and BuyVia[9]. The goal of the BuySafe project is to develop such an application which would allow customers to quickly and easily get the information about products, so that they could buy better and healthier products.

3. Project Scope

BuySafe project is developed as a part of DSD course taught in collaboration by MDH-IDT, UZG-FER and UL-PDM. Since this project has also been conceived as a SCORE project it includes all the guidelines defined by the course supervisors as well as SCORE project proponent.

These included timely submission of all necessary documents like weekly reports, summary reports, requirement definitions, project plan, design description documentation, acceptance test plan and project releases namely alpha, beta and release candidate.

The project started in early October and ran until middle of January. The workload during this period was well distributed among the team members. This is a short duration for developing a full-fledged project since there are other courses that the team members had to attend.

There were plenty of obstacles which needed to be overcome during the whole project. First we had to find suitable open data sources, which wasn't quite easy for our needs considering there aren't many product databases that offer the required data. We also had to learn new technologies to work on like Android where we had little experience. There was also some trouble integrating third-party software into the application, as we didn't have any experience using it prior to this project. Despite the difficulties that we had, we managed to overcome everything and finish the project.

4. Management Plan

4.1 Project group

Project team consists of six team members. Three team members, Juraj Murgić, Saša Marjančić, and Želimir Kompes are working in Zagreb. The rest of the three, Trevor Jagerfield, Fouad Yaseen, and Xiaowei Ma are working in Vasteras. The project manager is Juraj Murgić. He coordinated and organized the team members and resources. Trevor Jagerfield was the team leader responsible for organizing team members in Vasteras and coordinating with Juraj Murgić.

Project includes outside consultants on application functionality and application domain. Consultants for application functionality are: Marko Paripović (Association for Consumer Protection „Potrošački centar"), Nela Kovačević (Federation of Associations for Consumer Protection in Croatia) and Carol Pollack-Nelson, PhD (International Consumer Product Health & Safety Commission). Consultant on application domain is: Siniša Bošković (internal medicine, MD).

4.2 Asynchronous communication

The most of the asynchronous communication was done via mail and on Google Groups. Google Groups were chosen because of the simplicity of the information sharing. Želimir Kompes was responsible for the maintenance of the group.

4.3 Synchronous communication

Because of the geographical distance between team members we had regular meetings on weekly basis. The meetings were held on Wednesday and Sunday at 20:00 hours. One of our team members moved back to China so to accommodate for the time difference we moved the meetings to 15:00 hours. Meetings were held at least once a week, on Wednesdays and/or Sundays, except during holidays (1-7 of January). All meetings were held as a group audio conference via Skype. Meetings were organized by project leader, Juraj Murgić.

4.4 Event organization and time management

Time and place of every meeting was published on shared Google Calendar. Availability and obligations of the team members during this project were also posted by each member individually on the same Google Calendar.

4.5 Documentation management

Project documentation was managed in two ways. For the documents we used SkyDrive. SkyDrive was used as a repository for official documentation and contained all project documentation. When the SkyDrive documentation was updated it was uploaded to our SVN repository and to the official project site. SkyDrive maintenance was performed by Fouad Yaseen.

All source code and finished documentation was uploaded to SVN repository. Repository contained five main directories, *Code*, *Documentation*, *Final product*, *Other* and *Reports*. For better organization, folder *Documentation* contains separated folders for every category of the documentation. Directory *Other* was used for internal documents that were being used during development, like data sources or sample codes.

Repository URL: svn://lapis.rasip.fer.hr/svn/dsd12/SafeShopper
All team members were able to access repository with their unique username and private password[1]

[1] DSD team will provide SVN access if requested

5. Project Plan

5.1 Basic work delegation

In order to match the three tier architecture and have clearer responsibilities, the whole group is divided into three subgroups based on team member's preference and programming experience. Juraj Murgić and Saša Marjančić are responsible for service layer development, Želimir Kompes and Fouad Yaseen are responsible for presentation layer development, Trevor Jagerfield and Xiaowei Ma responsible for data layer management.

5.2 Activity plan

Activity plan is summarized in the Gantt chart on figure 7-1. Under resource names are the initials of team members responsible for the activity. The activity plan had three major revisions. The first after the project specification was done, the second after the first sprint was done and the third after the second sprint was done.

ID	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	Basic work delegation	3 days	15	17		JM
2	**Project specification**	**10 days**	**22**	**2**	**1**	**All**
3	Gathering and defining project requirements	5 days	22	27	1	All
4	Application functionalities defined	5 days	29	2	3	All
5	Application architecture defined	2 days	1	2	3	ZK,SM
6	Application design defined	3 days	29	31	3	
7	Data sources defined	27 days	29	4	3	XM,TJ
8	Set up VM with all components	1 day	5	5	2	XM
9	Set up Kunagi	1 day	5	5	2	XM,JM,SM
10	Create MySQL database on VM	1 day	5	5	2	JM,XM
11	Define backlog	1 day	6	6	10	JM
12	**Sprint implement basic functionality**	**13 days**	**7**	**23**	**11**	**All**
13	**Implementing backlog tasks**	**12 days**	**7**	**22**	**11**	**JM,SM,ZK,FY**
14	Implementing server side functionality	12 days	7	22	11	JM,SM
15	Implementing client side functionality	12 days	7	22	11	FY,ZK
16	Basic functionality test	1 day	23	23	13	JM,TY
17	GUI test	1 day	23	23	13	ZK,SM
18	**Implement data parsers**	**12 days**	**7**	**22**	**10**	**TJ,XM**
19	Create parser for Wall-Mart web page	12 days	7	22	2	TJ
20	Create parser for the RAPEX databse	12 days	7	22	2	XM
21	**Backlog grooming**	**1 day**	**23**	**23**	**13**	**All**
22	Documentation updated	1 day	28	28	12	All
23	Define backlog	1 day	26	26	12	JM
24	**Sprint - optional functionality implementation**	**24 days**	**27**	**28**	**23**	**All**
25	Data sources parsing	18 days	5	28	7	TJ,XM
26	**Implementing backlog tasks**	**15 days**	**27**	**17**	**13**	**JM,SM,ZK,FY**
27	Implementing server side functionality	15 days	27	17	23	JM,SM
28	Implementing client side functionality	15 days	27	17	23	FY,ZK
29	GUI test	3 days	18	20	28	JM,ZK
30	Optional functionality test	3 days	18	20	26	SM,TY
31	Documentation updated	3 days	18	20	26	All
32	Define backlog	1 day	18	18	26	JM
33	Holliday	5 days	31	4	24	All
34	**Sprint - Application testing and final implementation**	**5 days**	**7**	**11**	**33**	**All**
35	**Implementing backlog tasks**	**5 days**	**7**	**11**	**32**	**JM,SM,ZK,FY**
36	Implementing server side functionality	5 days	7	11	32	JM,SM
37	Implementing client side functionality	5 days	7	11	32	FY,ZK
38	GUI and performance test	5 days	7	11	32	JM,FY
39	Application acceptance test	5 days	7	11	32	All
40	Application finished	1 day	14	14	34	All
41	Documentation finished	3 days	15	17	40	All
42	Project finished	1 day	18	18	41	All

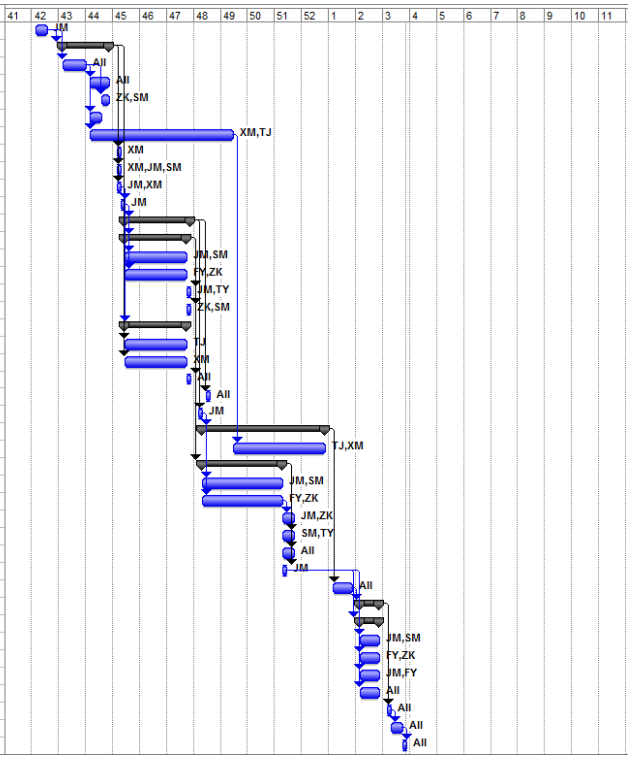

Figure 5-1. The Gantt chart of the project from week 42/2012 to 3/2013

5.3 Milestones

The project milestones are defined on the table 5-2. Only three milestones were not met in the defined time because of the problems with the integration of third party software (the barcode scanner). The rest were met on time.

Id	Milestone Description	Responsible Dept./Initials	Finished week				Metr.	Rem.	
			Plan	Forecast		Actual			
				Week	+/-				
M001	Basic work delegation (Responsibilities by date document)	JM	42	42	0	42	0		
M002	Gathering and defining project requirements (Project plan document)	All	43	44	0	43	0		
M003	Application functionalities defined (Requirements document)	All	44	45	0	44	0		
M004	Application architecture and object model defined (Requirements document and Design description document)	ZK,SM	44	45	0	45	0		
M005	Data sources defined (Project vision document updated)	XM,TJ	49	50	0	48	-1		
M006	Basic functionality implemented (Defined in project requirements document) (Application source code)	JM,SM,ZK,FY	49	50	0	49	0		
M007	Basic functionality test (Application test document)	JM,TY	50	50	0	50	0		
M008	GUI test (Application test document)	ZK,SM	50	50	0	50	0		
M009	Project documentation updated (Project documentation)	All	50	50	0	52	2		
M010	Optional functionality implemented (Defined in project requirements document) (Application source code)	JM,SM,ZK,FY	52	1	0	3	2		
M011	GUI test (Application test document)	JM,ZK	1	2	1	2	0		
M012	Optional functionality test (Application test document)	SM,TY	1	2	1	2	0		
M013	Documentation updated (Project documentation)	All	1	2	1	3	1		
M014	GUI and performance test (Application test document)	JM,FY	1	2	0	2	0		
M015	Application acceptance test (Application test document)	All	1	2	0	2	0		
M016	Application finished (Application source code + data base file)	JM,SM,ZK,FY	2	3	0	3	0		
M017	Documentation finished (Project documentation)	All	2	3	0	3	0		
M018	Project finished (Final project report document)	All	3	3	0	3	0		

Table 5-2. Project milestones

5.4 Project risks

At the beginning of the project we defined the possible risks that could occur during the project and defined

ways to lessen the potential impact that those risks had on the project. The defined risks are defined on the table 7-3.

Possibility	Risk	Preventive action
low	No barcode scanner functionality.	Remove barcode reading functionality and implement "search by title" only.
high	No adequate data sources.	Create a database to test application functionalities and find data sources at a later date.
medium	The project gets prolonged.	Avoid implementing complicated functionalities and using too much data sources.
medium	Developers lack knowledge of the technology.	All developers work together to find the easiest possible solution.
medium	Members not available due to some reason.	Responsible members try to finish the tasks before their unavailability. If that is not possible, other members may be assigned more tasks.
medium	Work coordination over distance	Use project management tools, weekly meeting to discuss progress and clear work delegation.
low	Setting up the development environment takes more time.	Try to seek help from the supervisor in this regard. In the worst case, change the development environment.

Table 5-3. Project risks

6. Development Process

We used a modified version of the SCRUM development process. The SCRUM components we used were the *SCRUM Team* which consisted of Juraj Murgić, Saša Marjančić, Želimir Kompes, Fouad Yaseen, Xiaowei Ma and Trevor Jagerfield. Juraj was the *Product Owner* and he was responsible for product management and its quality. Fouad was *Scrum Master* ensuring the smooth working of the SCRUM team and enforcing SCRUM practices. We defined three *SCRUM Sprints*. At the beginning of every team meeting there was a *Sprint Review Meeting*. The development tool that we used to supplement SCRUM is Kunagi [5]. We defined the project risks (Figure 5-6.) and project quality requirements (Figure 5-5.) using it. Also Kunagi was used to delegate work between team members (Figure 5-4.) based on the stories grabbed form the product backlog.

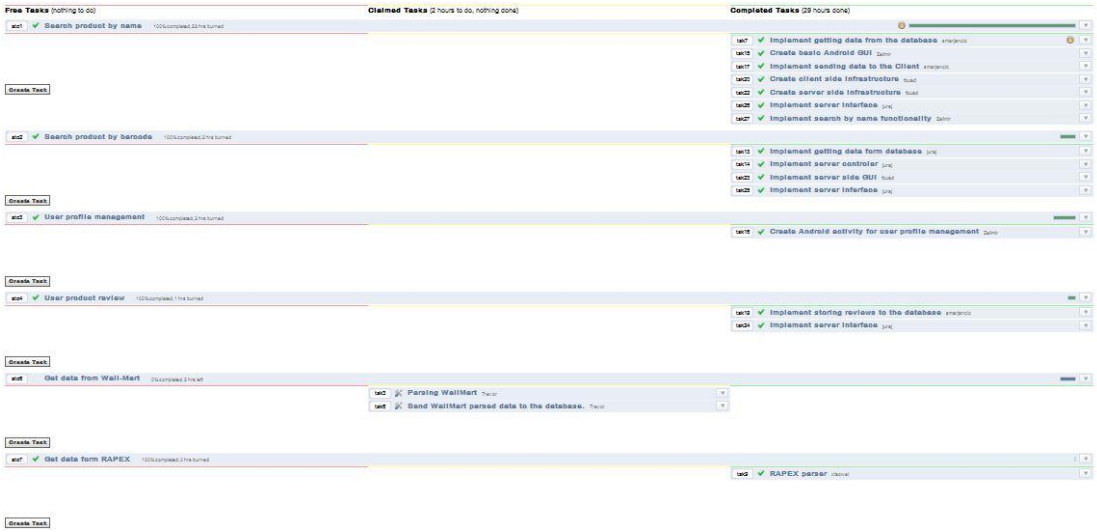

Figure 5-4. White board of the SCRUM project

On the Sprint Review Meeting each of the team members described what work he had done after the last meeting and problems that he had faced. That way all of the team members were informed about

the progress of the rest of the team. The Wednesday meetings were meetings to review the progress we were making that week. On Sunday meetings we discussed the work we were going to do the following week. The work was then defined in the Work delegation document on SkyDrive. An example on how the work was defined in the work delegation document is shown in the figure 7-4.

First sprint started on 7-11-2012 and ended on 27-11-2012. The goal of the sprint was to implement all of the basic functionality of the application. The goal was reached for all aspects except for the barcode scanner which was left for the second sprint because there was no way to test the barcode scanner as the application was not jet in a state ready to be publish to a device.

Second sprint started on 27-11-2012 and ended on 18-12-2012. The goal of the sprint was to update the basic functionality and implement the optional functionality. The goal was reached except for the barcode scanner because integration of the barcode scanner application to our application encountered some problems that were hard to fix because the barcode scanner application was a black box of code. We anticipated such problems (see project risks) and added the search by name functionality to basic functionalities. As planned by the end of this sprint all our data sources were defined and parsers were up and running.

Third sprint stared on 7-1-2013 and ended on 14-1-2013. The goal of the sprint was to implement all of the basic and optional functionality we didn't implement in the previous sprints. This sprint was conceived as a time reserve in case we didn't have time to implement all of the defined functionalities. In case all of the functionalities were implemented then the goal was to improve the usability of the application.

Figure 5-5. Kunagi project quality requirements

Picutre 5-6. Kunagi project risks

Activity Owner(s)	Activity	Work load	Result / Comment
ALL	Team meetings and related activities	24 (6x4h)	MoM-s
ALL	Comment source code	6 (6x1h)	Source code on SVN updated
TJ, XM, SM, JM	Interview people to get feedback on the application functionality and design	20 (4x5h)	Pool and results of the pool put on the SVN under Other\Customer research and feedback\
ZK, FY, JM	Finish designing the new GUI for the client	9 (3x3h)	Picture of the application logo on SVN under Other\Design Defined colors for the application under Other\Design\Design specification.doc Mock-ups of the GUI on SVU under Other\Design
ZK, FY	Implement GUI design	30 (2x15h)	Client GUI changed
XM	Update parsers to run from the server store data to the database	2	Parsers on the server with the ability to be run from the server
ZK, FY	Write the beta presentation	10 (2x5h)	Beta presentation of the project on SVN under BuySafe\Documentation\Beta presentation\
SM	Write MoM-s	2	MoM-s added to Documentation/Minutes of Meeting/ on SVN
JM,SM	Add error handling to the server	6 (2x3h)	Code of server updated to include error handling
XM	Try to deploy server on VM and to run parsers from server code	6	Server running on VM
FY, TJ	Write JUnit test for client side functionality (implemented and the things we will implement)	10 (2x5h)	Client code on SVN updated
ZK	Implement shopping list functionality on client	8	Client code on SVN updated
TJ	Test implemented functionality	2	Test report of implemented functionality

Table 5-7. Work delegation document example

7. Requirements Specification

As our application was intended for a wide group of users we defined our application requirements in two steps. In the first step we brainstormed ideas what the application should do and tried to find people who were dealing with the problem of consumer product health and safety. Based on our ideas and the input of our consultants we defined the basic requirements for the application (defined in the section 9.1.).

In the second step we gathered feedback from users. We interviewed 94 people on the way they define quality product and the way they buy products. Also we interviewed 64 people on the application functionality and design. The feedback gathered suggested detailed manufacturer information, prices, recommendation of similar products, etc. Based on the gathered feedback we made some minor changes to the application as we didn't have time to implement big changes with only one week left.

7.1 Basic requirements

7.1.1 Basic functionality

Basic goal of the application is to inform people about the product that they buy so that they could buy better quality products. It will achieve this by informing the user about the product manufacturer, contents and quality of the product. Product quality will be estimated by the product contents and user reviews of the product. Product contents will be flagged if they are estimated to be dangerous or harmful to people. It will user profiles that contain user allergies and/or illnesses, data sources that contain products that are estimated to be dangerous by state authorities and data sources that contain product contents that are estimated to be harmful. The information it will display will come from reliable data sources or even official sources like state authorities. It will also have a tutorial that will give users helpful tips on how to buy quality products by informing them what things to look out for when buying.

7.1.2 Application interface

To make the application practical it should be run on Android smart phones with the ability to scan the barcode of the product that the customer is intending to buy. Also to make the application easier to use it will include the ability to search the products by name because not all products will have defined barcodes. User interface should be simple, intuitive and easy to use.

7.1.3 Optional functionality

Optional functionality is intended to help people while they shop for food. Users will be able to scan multiple products and the application will compare them and show the user which of the scanned products has the best quality. Also users will be able to define a list of products at home that they intend to buy a shopping list. Users will have the ability to save products that they are not certain about buying so that they can research them later. To help improve the functionality of the application users should be able to flag products that they think are harmful or even dangerous. Products that we missed by the data sources.

7.1.3.1 Database

The database has to be optimized to enable fast searches by product barcode and name.

7.1.3.2 Data integrity

The application will use different data sources that are regularly updated. Our application will also update its database once per week according to the data sources (Amazon, FoodFacts, BuySafe), to maintain data integrity.

7.1.3.3 Susceptibility to testing

It will be possible to test the application client, data and server layers separately.

7.1.3.4 Interoperability

The application will have an interface that will enable it to interact with the Android client and potentially other clients as well.

7.1.3.5 Performance

The application will have a maximum response time of 10 sec.

7.1.3.6 Usability

The interface will ably by the three click rule.

7.1.3.7 Security

Because of the sensitive nature of the user profile data that consists of user allergies and illnesses no recognizable user parameters like name and surname will be stored. All the profile information will be stored locally on the user's smart phone.

7.1.3.8 Interface

The application will have an intuitive and user friendly interface that will allow users to easily understand how the application is supposed to be used.

7.2 Requirements

List of all functional and non functional requirements is shown on table 7-1.

Identity	Status	Priority	Reference	Description	Source
				Server Side	
SS-1	I	1	3.7	Handle exceptions and errors efficiently and act accordingly in the case of exceptions.	Sys
SS-2	I	3		Schedule database updates from external sources.	Ds
				Product Handling	
PH-1	I	1		UC2 - User searches for product	Ctm
PH-2	I	1		UC3 - User searches product by barcode	Ctm
PH-3	I	1		UC4 - User searches for product by name	Ctm
PH-4	I	1		UC5 - User chooses wanted product from list	Ctm
PH-5	I	1		UC7 - User writes the review of the product	Ctm
PH-6	I	1		UC8 - User flags the product	Ctm
PH-7	I	2		UC14 - User searches all safe products	Ctm
PH-8	A	2		UC15 - Compare products	Ctm
PH-9	M	1		UCX - User scans the product	Ctm
PH-10	M	1		UCX - User rates the product	Ctm
PH-11	D	1		UCX - Recall and incident history	Ctm
				Shopping List Handling	
SLH-1	I	2		UC6 - User adds product to shopping list	Ctm
SLH-2	I	2		UC9 - User views shopping list	Ctm
SLH-3	I	2		UC10 - User starts the process of adding the product to the shopping list	Ctm
SLH-4	I	2		UC11 - User deletes the shopping list	Ctm
SLH-5	I	2		UC12 - User deletes a product from shopping list	Ctm
SLH-6	M	2		UC10 - User enters product name	Ctm
				User Profile Handling	
UPH-1	I	1		UC1 - User creates/edits profile	Ctm
UPH-2	M	1		UCX - User enters personal data	Ctm
UPH-3	M	1		UCX - User enters allergies or illnesses	Ctm
				Android Client	
AC-1	I	1		Use android SDK 2.3 to ensure compatibility with maximum number of devices.	Sys
AC-2	I	1	3.11	XML-based layout to speed up the UI-design process.	Sys
AC-3	I	1		UC13 - User exits the application	
AC-3.1	I	1		Save data locally in device before exiting.	Ds
AC-3.2	I	1		Close all connections.	Sys
				Database and Parser Handling	
DBPH-1	I	3	3.3	Optimize database to enable faster searches.	Sys
DBPH-2				Handle different data sources accordingly.	
DBPH-2.1	I	1		Connect to database for locally stored data.	Ds
				Non-functional Requirements	
NFR	I	2	3.8	Response time should be less than 10 seconds.	Ds

Table 7-1. Project requirements

7.3 Use cases

The diagram describing use case relations is shown on the figure 7-2. As the diagram is self-explanatory we do not include a detailed description of it here.

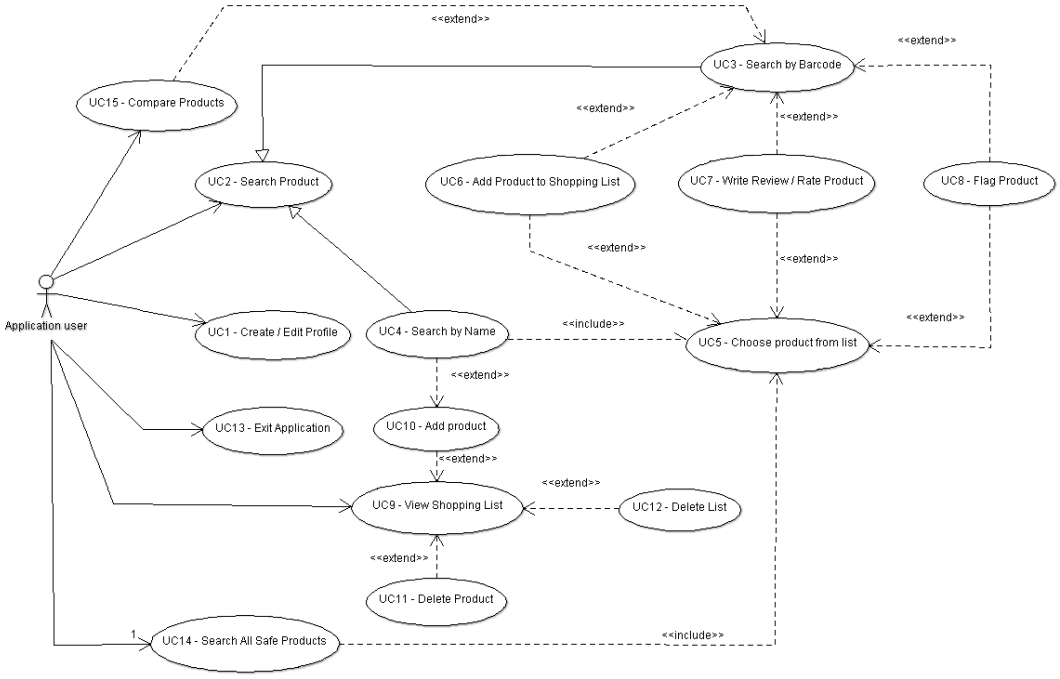

Figure 7-2. Use case diagram

8. Architectural Design

8.1 Conceptual design

The architecture follows a client-server model. Client has module necessary for interaction with the user, logic module, persistence module and a module responsible for exchanging data with the server. Server has logic module, scheduler module and a module used for interaction with the client. The following components have been implemented:

Android Activity - This component manages the GUI and is responsible for interaction with the user.

Client-side Logic - It contains all the manager and entity classes that are required for client-side business logic.

Persistence Module - Persistence module saves and loads data from the android memory.

Request Handler (client) - This component connects to the server. It sends the data to server as name-value pairs and receives data as XML strings.

JSON-parser - It has classes that parse the JSON strings and instantiates the entity classes based on data in the strings.

Request Handler (server) - This module receives data from client and sends it to the required manager classes. It also sends data back to client as XML strings.

Server-side controller - It contains all the manager and entity classes that are required for server-side business logic.

Database Connector - This module is responsible for managing connections with the database. It also contains the stored SQL queries.

Scheduled Parser - This module handles the updating of database based on the new information in external data sources. Classes in this module check for updates after fixed intervals of time.

Model classes - This module consists of entity classes used throughout the project.

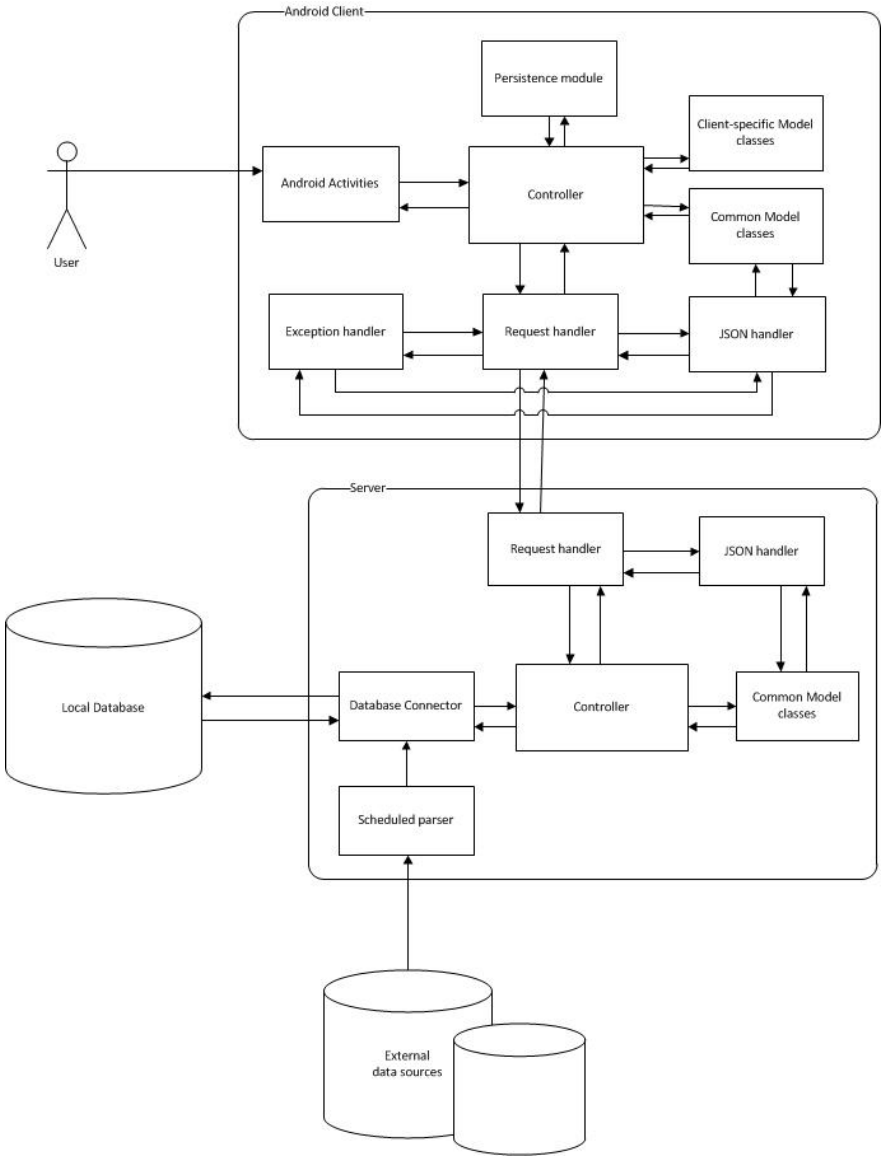

Figure 8-1.: Software architecture

8.2 Behavioral design

The following sequence diagram describes how user manages his profile and then checks if the product he/she wants to buy is safe or not. First the user clicks on the "search product" button, and event handler will pass the message to ClientManager and try to get his profile from memory. If there is no history profile returned, then the user will be required to create a new profile which contains the illness or allergic information. After the profile is done, the user will be provided with two options, which are search by barcode and search by name. Both options will make the ServerMangager check the product that the user input against the database. The database returns the contents of the product to ServerManager which will check if the user profile conflicts with the contents of the product. Besides, ServerManger also checks if the product the user wants to buy is forbidden on the market. Finally the results will be passed to the user.

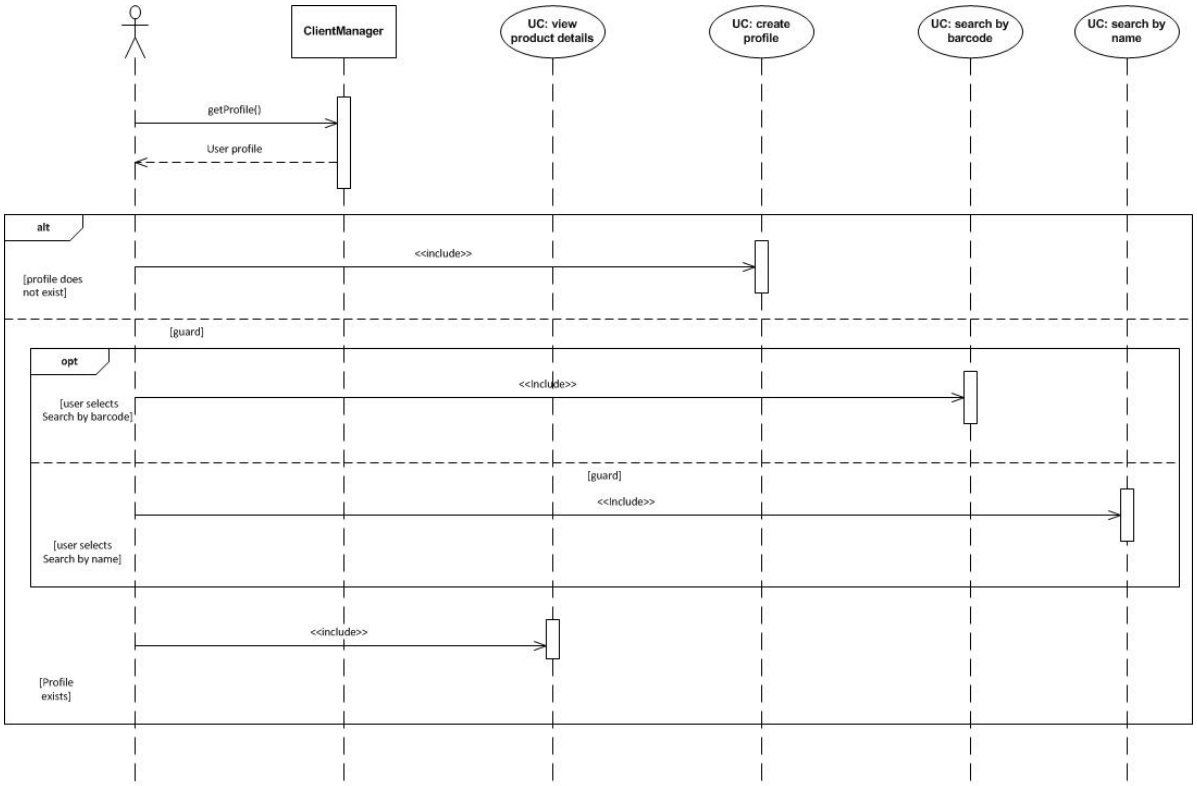

Figure 8-2. Sequence diagram

9. Implementation

Client side application has been developed for Android mobile devices and requires continuous connection to the server for full functionality. On the server side Apache Struts2 web framework is used for handling client requests for resources.

For the database, MySQL is used because because all team members are familiar with usage of the database. The goal of Struts is to separate the model (application logic that interacts with a database) from the view and the controller (instance that passes information between view and model). Struts2 provides the controller (a Servlet known as ActionServlet).

The application uses three data sources. The RAPEX [2] data source is used to make sure that the products aren't banned from counties or seriously dangerous to the user. The second is the FoodFacts[1] data source which with Amazon[3] data source is used to gather product details like contents and nutrition facts. Also they are used to gather user reviews and feedback on product quality like weight watchers points[4].

After the user sends a request to the server for some resource, Servlet filter (Filter Dispatcher) looks at the request and then as per the mapping of URL, request is forwarded to appropriate Action Class. Selected action is executed to perform the requested operation. Depending on a request from application, selected action performs an SQL query to database and extracts required data. After data processing, JSON response will be generated and sent to the client. Thereafter, client receives response and processes the received data.

10. Verification and Validation

BuySafe project was tested to meet all the requirements outlined in the requirement definition document. Functionalities were tested with test cases that concern only smaller scope of each requirement, as many requirements have impact on several other components so it would be hard to test through one test case. Unit test cases were designed separately for the Client and the Server Application.

10.1 Approach

Application is tested continuously during the whole project development. On the server side, unit tests are written by developers and on the client side tests are executed manually during development and automated at the end of development. In the first iteration, basic application functionality, without client-server data exchange, was tested. Application was tested for different data input and that all implemented functions are working properly. In the second iteration, client-server connection is established. Application is tested for correct data exchange and correct client-server communication. Multiple testing scenarios are made to ensure the communication functionality. At the end of development, behavioral and automated black-box Android tests are executed. Most common Android-related situations which are considered crucial for Android development are tested.

Application is tested in the following way:

- Change in screen orientation: For devices that support multiple orientations, Android detects a change in orientation when the user turns the device. When Android detects a change in orientation, its default behavior is to destroy and then re-start the foreground Activity. The following is tested:
    - Is the screen re-drawn correctly?
    - Does the application maintain its state? The Activity should not lose anything that the user has already entered into the UI.
- Dependence on external resources: Application depends on network access. Test scenarios are executed to see what happens when the resource is not available. When application intends to use the network, it can notify the user if access is unavailable.
- Robotium: Used for automated black-box testing. Robotium is a test framework created for writing powerful and robust automatic test cases for Android applications. Function, system and acceptance test scenarios can be easily written.
- The Monkey: Program that runs on emulator or device and generates pseudo-random streams of user events such as clicks, touches, or gestures, as well as a number of system-level events. Monkey is used to stress-test application in a random manner.
- Manually by developers

10.1.1 Approach to configuration and installation

Application can be installed on the Android device or locally in Android emulator on the machine used by tester. For realistic usability testing a hardware device is required. Network access is required. Data needed by the application will be stored on a remote server and fetched via network. Developers and project leader will provide full manual, required steps and guidance for setting up the environment

10.2 Tested features

Identity	Status	Priority	Reference	Description	Source
PH-1	I	1		UC2 - User searches for product	Ctm
PH-2	I	1		UC3 - User searches product by barcode	Ctm
PH-3	I	1		UC4 - User searches for product by name	Ctm
PH-4	I	1		UC5 - User chooses wanted product from list	Ctm
PH-5	I	1		UC7 - User writes the review of the product	Ctm
PH-6	I	1		UC8 - User flags the product	Ctm
PH-7	I	2		UC14 - User searches all safe products	Ctm
PH-8	A	2		UC15 - Compare products	Ctm
SLH-1	I	2		UC6 - User adds product to shopping list	Ctm
SLH-2	I	2		UC9 - User views shopping list	Ctm
SLH-3	I	2		UC10 - User starts the process of adding the product to the shopping list	Ctm
SLH-4	I	2		UC11 - User deletes the shopping list	Ctm
SLH-5	I	2		UC12 - User deletes a product from shopping list	Ctm
SLH-6	M	2		UC10 - User enters product name	Ctm
UPH-1	I	1		UC1 - User creates/edits profile	Ctm

Figure 10-1. Tested application features

11. Outcomes

11.1 Application

The main part of our project has been the client application for Android mobile platform. Although, the whole system couldn't work without the server application, it only serves to support the client application. As this application has been designed to be used by customers of different groups and ages we have tried to make it user-friendly and attractive to the eye. The following pictures show the main screen of the Android client application.

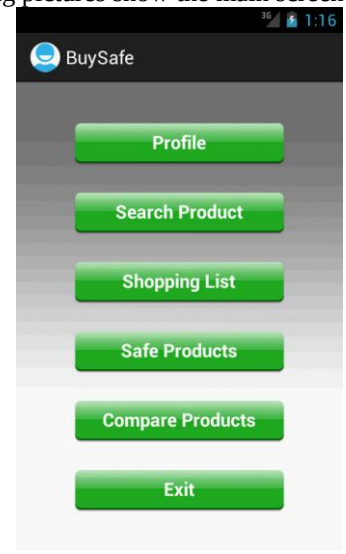

Figure 11-1. Main screen application

From this screen users can choose various actions that they want to perform. By pressing on the profile

option users can create their own profile which contains information that as search conditions. The main feature of the application is the search product option which is show on the following figure.

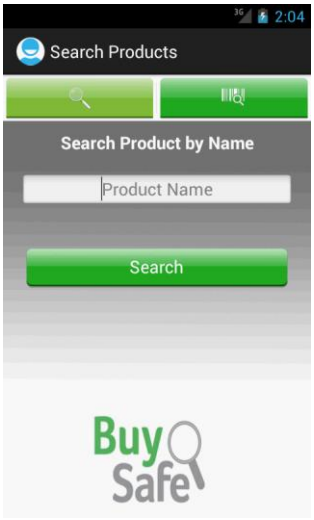

Figure 11-2. Search products

Search products is possible by using name of the product or barcode which is scanned using the mobile phone's integrated camera. After searching for a product user can look at the details of that product which is shown on the following figure. User can look at different product information including the nutrition facts of the products, overall rating of the product given by other users, etc. User can also create his own review about the product or warn other users about the product if he thinks it is necessary.

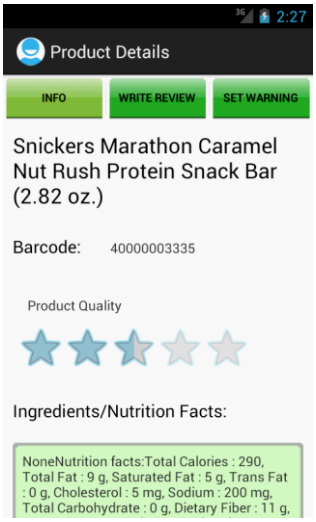

Figure 11-3. Product details

From the main screen user can also choose the shopping list which allows him to create a list of items before going to the store. This action is shown on the following figure.

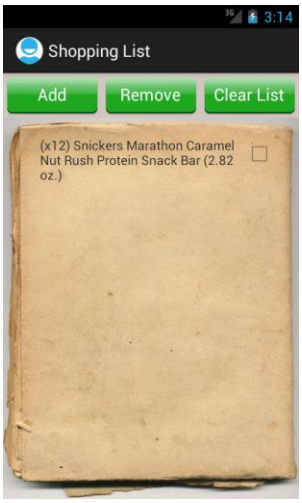

Figure 11-4. Shopping list

11.2 Other outcomes

Other than the final product we also produced many documents along the way for better project tracking. We have created document regarding project and design description, about requirements and documents containing final project report and acceptance test plan. Every week during the project each of us has filled a week report based on which we wrote summary week reports. As we had regular weekly meetings, we also wrote minutes of meeting documents which were used as reminders. All the documentation can be found at [10].

12. Lessons learned

This project was a great educational experience for all of us. It thought us a lot of valuable things and has made us better people and developers. From all of the problems we faced we gained experience and become friends.

One of the first obstacles we faced was the communication barrier. Not realizing it at first but our ideas were interpreted by other team members in different ways. This resulted in things being done differently from what was agreed. Over time we learned how to better communicate with each other by repeating ourselves in different ways and using examples as much as possible. Also team members asked for an elaboration when something was unclear to them. In general in team as diverse as our own team members have to be very specific while communicating so the rest of the team members understand them. This applies to written agreements like mail and post. Members need to be as specific as possible.

Another problem was work delegation over distance and in different time zones. Team members didn't work at the same time or at the same place so work had to be defined in a way that didn't include a lot of communication between team members that had to bi instant for the work to be done efficiently. The work was delegated to smaller sub teams which then divided the work between team members. In that way all team members worked on individual tasks most of the time.

One other problem that we had was integration with third party software. The problem had presented itself while we were on the second sprint. As we had an alternative way of searching for products we dropped the priority of the barcode scanner so other functionalities would be implemented. We have learned that a lot of problems can be solved before they occur by good risk management.

13. Summary

The BuySafe project was developed by a group of students enrolled in the Distributed Software Development (DSD). The DSD course is a collaborative course in which participated students and professors form three different universities. During the course length (which was only 13 weeks) we had to complete the project. We faced problems along the way but with good communication and hard work we managed to solve all of them. Good planning and fixed team meeting ensured all of the team members participated in the work and gave their input. Good work delegation and regular updates on team meetings kept all team members informed about the team progress. The project was developed in three sprints. The first sprint to implement basic functionality, second sprint to implement optional functionality and the third sprint to implement what was missed in the first two sprints. The initial project idea was changed from warnings about products in general to just food products because they are the ones that can do the most hart to people if they aren't careful.

We are a very diverse team with members from four countries and an age difference of 25 years between team members. Those diversities brought different perspectives and ideas to the project which we merged to define the basic functionalities for the BuySafe application.

All in all the BuySafe project was a great educational experience for all team members. We learned new technologies and work environments while working with new people from different countries and different cultural backgrounds. We learned how to communicate with each other, collaborate and resolve problems. Also we learned how to delegate tasks and how to write project documentation. Project BuySafe is a project that all of the team members are really proud to have participated in.

Red and signed by:

Project manager:
Juraj Murgić (juraj.murgic@gmail.com), Date: _____ Signature: _____

Team leader

Trevor Jagerfield (jagerfield@yahoo.com), Date: _____ Signature: _____

Team members:

Xiaowei Ma (xma12001@student.mdh.se), Date: _____ Signature: _____

Fouad Yaseen (fouad2000@gmail.com), Date: _____ Signature: _____

Zelimir Kompes (zelimir_kompes@hotmail.com), Date: _____ Signature: _____

Saša Marjančić (s.marjancic@gmail.com), Date: _____ Signature: _____

Project supervisor:

Marin Orlić (marin.orlic@fer.hr), Date: _____ Signature: _____

14. References

[1] FoodFacts - http://www.foodfacts.com/ (30.1.2013.)
[2] RAPEX - http://ec.europa.eu/consumers/dyna/rapex/rapex_archives_en.cfm (30.1.2013.)
[3] Amazon - http://www.amazon.com/ (30.1.2013.)
[4] Weight watchers points plus - http://www.weightwatchers.com/plan/eat/plan.aspx (30.1.2013.)
[5] Kunagi - http://kunagi.org/ (30.1.2013.)
[6] School of Innovation, Design and Engineering - https://www.mdh.se/idt (30.1.2013.)
[7] University of Zagreb, Faculty of Electrical Engineering and Computing - http://www.fer.unizg.hr/en (30.1.2013.)
[8] RedLaser - https://itunes.apple.com/us/app/redlaser-barcode-scanner-shopping/id474902001?mt=8 (30.1.2013.)
[9] BuyVia - https://itunes.apple.com/us/app/buyvia-price-comparison-for/id572807843?mt=8 (30.1.2013.)
[10] BuySafe project main page- http://www.fer.unizg.hr/rasip/dsd/projects/safeshopper (30.1.2013.)