**Software Requirements Specification**

<< Version 1.0>>

April, 2016

Project:

**100 beers E-Store**

Asan Mollov

Filip Filipov

Lilyana Babacheva

Nikolay Dimitrov

QA Fundamentals - April 2016 TEAM "Agatoton",

Software University – Sofia

# Contents

[Contents 2](#_Toc449124240)

[1. Introduction 3](#_Toc449124241)

[1.1. Purpose 3](#_Toc449124242)

[1.2. Scope of Project 3](#_Toc449124243)

[1.3. Definitions, Acronyms and Abbreviations 4](#_Toc449124244)

[2. Overall description 4](#_Toc449124245)

[2.1. System Environment 5](#_Toc449124246)

[2.2. Basic flow 6](#_Toc449124247)

[2.3. Constraints 6](#_Toc449124248)

[3. Functional Requirements Specification 6](#_Toc449124249)

[3.1. Description 6](#_Toc449124250)

[3.2. Technical Issues 8](#_Toc449124251)

[3.3. Use cases 8](#_Toc449124252)

[4. Interface Requirements 18](#_Toc449124253)

[4.1. GUI - Various interfaces for the product 18](#_Toc449124254)

[4.2. Hardware interface 20](#_Toc449124255)

[4.3. Software interface 20](#_Toc449124256)

[5. Design constrain 21](#_Toc449124257)

[6. Non - Functional Requirements 21](#_Toc449124258)

[6.1. Security 21](#_Toc449124259)

[6.2. Reliability 21](#_Toc449124260)

[6.3. Availability 21](#_Toc449124261)

[6.4. Maintainability 22](#_Toc449124262)

[6.5. Portability 22](#_Toc449124263)

# Introduction

## Purpose

The purpose of this document is to present a detailed description of the **100 beers E-store System**. It will explain the purpose and features of the system, theinterfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. It will serve as a guide to everyone, who is responsible to the project.

**100 beers E-store System** is intended to help the user to buy beer products forminternet. It should give all of the staff the necessary information to develop and test the software and what can and cannot the system do.

## Scope of Project

This software system is an E-store system for online shopping. This document describes what features will be in the scope of the software and what are not in the scope of software. The project is divided in two parts:

1. Public part where people can visit the system and buy beers
2. Private part where administrator can maintaining the public part

More specifically, this system is designed to allow a user to view and buy beers from different categories and to allow an administrator to manage the system (to add and remove products, to upload and remove pictures, etc.)

## Definitions, Acronyms and Abbreviations

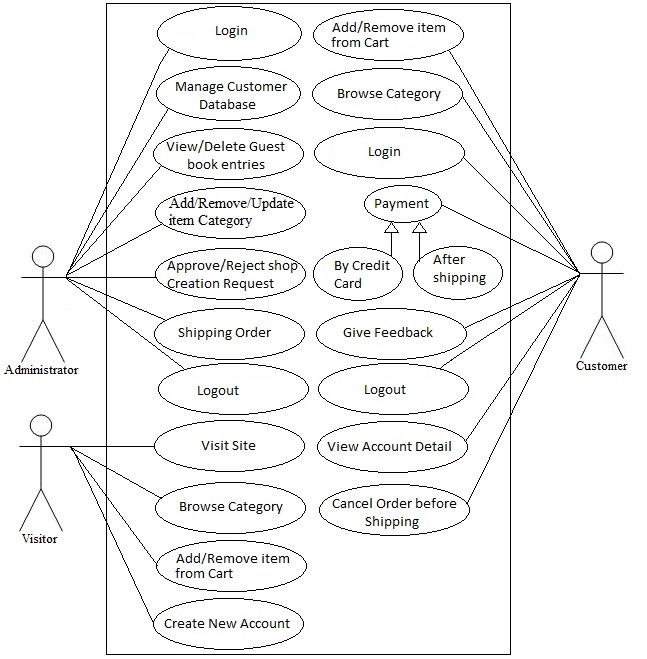
  GUI- Graphical User Interface

          SRS- Software Requirement Specification

# Overall description

The E-store system enables customers to browse through products of shops, make orders and a system administrator to approve and reject the orders and maintain lists of shop categories and beers.

* 1. System Environment



The E-store system has three actors – Visitor, Customer and Administrator. All actors access system thought internet. System is divided in two parts. First part is for visitors and customers who can view products and buy products. Second parts - the administrator is allowed to manage the system (to add and remove products, to upload and remove pictures, etc).

## Basic flow

The customer wants to buy item. The system shows all product categories to customer. If customer select item then they are listed in shopping cart for buying. The payment will be made with credit card or postpaid by shipping. If customer wants to cancel the order before shipping then he or she can cancel it. Customer can see the buying report on account detail.

## Constraints

The following are the constraints:

* The project must be completed within the budget
* The project must be completed within a specified period of time.
* The system should be up 24/7.
* The system should enforce user authentication security and guarantee reliability.
* The system should support browsers such as Mozilla Firefox v38 and higher IE11, Google Chrome v37 and higher.

# Functional Requirements Specification

## Description

This section provides requirement overview of the system. Various functional modules that can be implemented by the system will be:

3.1.1 Registration

If customer wants to buy the product then he/she must be registered, unregistered user can’t buy products.

3.1.2 Login

Customer logins to the system by entering valid user id and password for the site. The user should not be allowed to make sql injection and enter special words.

3.1.3 Changes to Cart

User have access to Cart with or without registration. If he/she add products in Cart and press button ‘Checkout’ he/she have two options to enter his registration or continuing like guest where he/she enter personal detail and get registration.

The user should be allowed to enter positive quantity between 1 and 100. By string, floating number, larger or fewer quantity the system should display message.

3.1.4 Payment

For customer there are many type of secure billing will be prepaid as debit or credit card, postpaid after shipping. The security will provide by the third part like Pay-Pal etc.

3.1.5 Logout

After the payment or surf the product the customer will logged out.

3.1.6 Report Generation

After all transaction the system can generate email and then sent one copy to the customer’s Email-address and another one for the system data base to calculate the monthly transaction.

3.1.7 Contact form

User can use contact form for make inquiry if he/she have questions. The user should enter valid email. Message should be displayed by wrong email or security code.

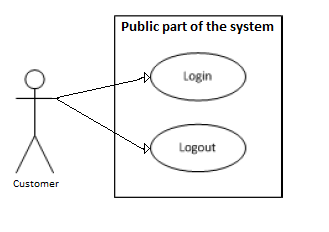
## Technical Issues

This system will work on client-server architecture. It will require an internet server and which will be able to run application. The system should support browsers such as Mozilla Firefox v38 and higher IE11, Google Chrome v37 and higher.

## Use cases

3.3.1 Use case Login functionality

**Use case: Login in public part of the system**

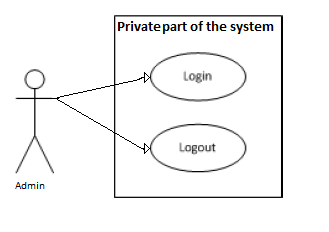


| Name | Login in the public part of the system |
| --- | --- |
| Summary | This use case describes how an actor can login in public part in the system |
| Goal | 1. The user has an account  2. The user wants to login  3. The user is not already logged in |
| Actor | User |
| Preconditions | The actor is on login form |
| Basic Flow | 1. The user open login page  2. The system display login page  3. The user enter email  4. The user enter password  5. The user click login button  6. The system check for valid credentials  7. The system redirect user to the profile page. |
| Alternative flows | 1. The user open login page  2. The system display login page  3. The user enter email  4. The user enter password  5. The user click login button  6. The system check for valid credentials and check fails  7. The system display error message indicating invalid log and allows the user to log again. |
| Post conditions | 1. The user is logged in to the system  2. The user has access to the functions of public part of the system |

**Use case: Logout of public part of the system**

| Name | Logout of public part of the system |
| --- | --- |
| Summary | This use case describes how an actor can logout of public part of the system |
| Goal | The user wants to logout of his/her profile on system |
| Actor | User |
| Preconditions | 1. The actor is logged in  2. The actor no longer wants to be logged in |
| Basic Flow | 1. The user is done with using the system  2. The click on logout link  3. The system display successful logout message |
| Alternative flows | N/A |
| Post conditions | The user successfully logout of the system |

**Use case: Login in private part of the system**



| Name | Login in the private part of system |
| --- | --- |
| Summary | This use case describes how an admin can login in private part of the system |
| Goal | 1. The admin has an account  2. The admin wants to login  3. The admin is not already logged in |
| Actor | Admin |
| Preconditions | The actor is on admin login form |
| Basic Flow | 1. The admin open login page of administration page  2. The system display login page  3. The admin enter email  4. The admin enter password  5. The admin click login button  6. The system check for valid credentials  7. The system redirect user to the appropriate home page. |
| Alternative flows | 1. The user open login page  2. The system display login page  3. The user enter email  4. The user enter password  5. The user click login button  6. The system check for valid credentials and check fails  7. The system display error message indicating invalid log and allows the user to log again. |
| Post conditions | 1. The admin is logged in to the private part of the system  2. The user has access to the functions of private part of the system |

**Use case: Logout of private part of the system**

| Name | Logout of private part of the system |
| --- | --- |
| Summary | This use case describes how an actor can logout of private part of the system |
| Goal | The admin wants to logout of his/her profile on system |
| Actor | Admin |
| Preconditions | 1. The actor is logged in  2. The actor no longer wants to be logged in |
| Basic Flow | 1. The admin is done with using the system  2. The click on logout link  3. The system display successful logout message |
| Alternative flows | N/A |
| Post conditions | The admin successfully logout of the system |

3.3.2 Use case Cart functionality

**Use case: Adding and modifying products in the cart as visitor**

| Name | Adding beer to “Cart” section |
| --- | --- |
| Summary | This use case describes how an actor can add items in the shopping cart. |
| Goal | The user wants to add beer to his list in “Cart”. |
| Actor | Visitor |
| Preconditions | The visitor is entered in the site. |
| Basic Flow | 1. The user clicks on one of the category in the above frame of the screen – By country, by style or by color. Than he chooses from the drop down menu one of the categories and navigates to the item he wishes to add to the cart and clicks on beer.  2. The system displays screen with the details about the beer and the user choose the count of the beers he wants to add in the cart. He clicks + button to increase the number of the beers and the – button to decrease. The user can also write down the count of the wished beers.  The user should write quantity between 1 and 100. By string, floating number, larger or fewer quantity the system should display message.  3. The user clicks on the button for adding in the cart. A field displays the total cost of the shopping cart and the count of the products.  4. The user repeats the steps for all the items he wants to add to the cart.  5. The system updates the new quantity and displays the modified info. |
| Alternative flows | 1. If user closes the window without proceeding for payment, the cart is not stored in the system for a pre-decided number of days. |
| Extensions | 1. If the user enters a string, containing integers and letters, the system displays an appropriate error message. |
| Post conditions | The user successfully removes existing items in the cart or adds new items to the basket. |

**Use case: Adding and modifying products in the cart**

| Name | Adding beer to “Cart” section |
| --- | --- |
| Summary | This use case describes how an actor can add items in the shopping cart. |
| Goal | The user wants to add beer to his list in “Cart”. |
| Actor | Registered user |
| Preconditions | The actor is on the Cart screen and has already logged in. |
| Basic Flow | 1. The registered user clicks on one of the category in the above frame of the screen – By country, by style or by color. Than he chooses from the drop down menu one of the categories and navigates to the item he wishes to add to the cart and clicks on beer.  2. The system displays screen with the details about the beer and the user choose the count of the beers he wants to add in the cart. He clicks + button to increase the number of the beers and the – button to decrease. The registered user can also write down the count of the wished beers.  The user should write quantity between 1 and 100. By string, floating number, larger or fewer quantity the system should display message.  3. The user clicks on the button for adding in the cart. A field displays the total cost of the shopping cart and the count of the products.  4. The user repeats the steps for all the items he wants to add to the cart.  5. The system updates the new quantity and displays the modified info. |
| Alternative flows | 1. If user closes the window without proceeding for payment, the cart is stored in the system for a pre-decided number of days, before getting flushed, so that the user can return to the cart in the future. |
| Extensions | 1. If the user enters a string, containing integers and letters, the system displays an appropriate error message. |
| Post conditions | The user successfully removes existing items in the cart or adds new items to the basket. |

**Use case: Only adding products to the cart**

| Name | Adding beer to “Cart” section |
| --- | --- |
| Summary | This use case describes how an actor can add items in the shopping cart. |
| Goal | The user wants to add beer to his list in “Cart”. |
| Actor | Visitor/Customer |
| Preconditions | The actor is on the Cart screen (and has already logged in). |
| Basic Flow | 1. The user clicks on one of the category in the above frame of the screen – By country, by style or by color. Than he chooses from the drop down menu one of the categories and navigates to the item he wishes to add to the cart and clicks on beer.  2. The user clicks on the green button for adding to the cart. The user can’t modify the quantity from this screen  3. A field displays the total cost of the shopping cart and the count of the products.  3. The user repeats the steps for all the items he wants to add to the cart.  4. The system updates the new quantity and displays the modified info. |
| Alternative flows | 1. If user closes the window without proceeding for payment, the cart is not stored in the system for a pre-decided number of days. |
| Post conditions | The user successfully adds new items to the basket. |

**Use case: Delete products**

| Name | Adding beer to “Cart” section |
| --- | --- |
| Summary | This use case describes how an actor can delete items from the shopping cart. |
| Goal | The user wants to delete products from his list in “Cart”. |
| Actor | User |
| Preconditions | The actor is on the Cart screen |
| Basic Flow | 1. The user clicks on the cart field under the section My profile  2. The system displays screen with the details about the wished products.  3. The user clicks on the X button next to the product that he wants to delete.  4. The user repeats step 3 for all the items he wants to delete from the cart. |
| Alternative flows | 1. If user closes the window without proceeding for payment, the cart is not stored in the system for a pre-decided number of days. |
| Post conditions | The user successfully removes existing items in the cart |

3.3.3. Use case Contact Form

**Use Case: User trying to send an enquiry with a valid details**

| Name | Sending message with valid email address |
| --- | --- |
| Summary | This use case describes how the user has to send messages via the contact form. |
| Goal | The user should be able to send a message with a valid email address. |
| Actor | Registered user |
| Preconditions | The actor is on the Cart screen and has already logged in. |
| Basic Flow | 1. The user is filling in all the fields- valid email address, title and text message.  2. He is trying to send the inquiry existing email address.  3. The user is clicking on the “Send button”.  4. The system is sending the message. |
| Alternative flows | N/A |
| Extensions | N/A |
| Post conditions | The user has to be able to send the enquiry only with a valid email address. |
|  |  |

**Use case : User trying to send an enquiry with an invalid security code.**

| Name | Sending message with invalid security code. |
| --- | --- |
| Summary | This use case describes how the user has to send messages via the contact form. |
| Goal | The user should be unable to send a message with invalid security code. |
| Actor | Registered user |
| Preconditions | The actor is on the Cart screen and has already logged in. |
| Basic Flow | 1. The user is filling in all the fields, but the security code is wrong.  2. He is trying to send the enquiry with the wrong security code.  3. The user is clicking on the “Send button”.  4. The system is responding with an error message. |
| Alternative flows | N/A |
| Extensions | N/A |
| Post conditions | The user successfully removes the wrongly tapped security code and can add a new one until the other details are saved. |

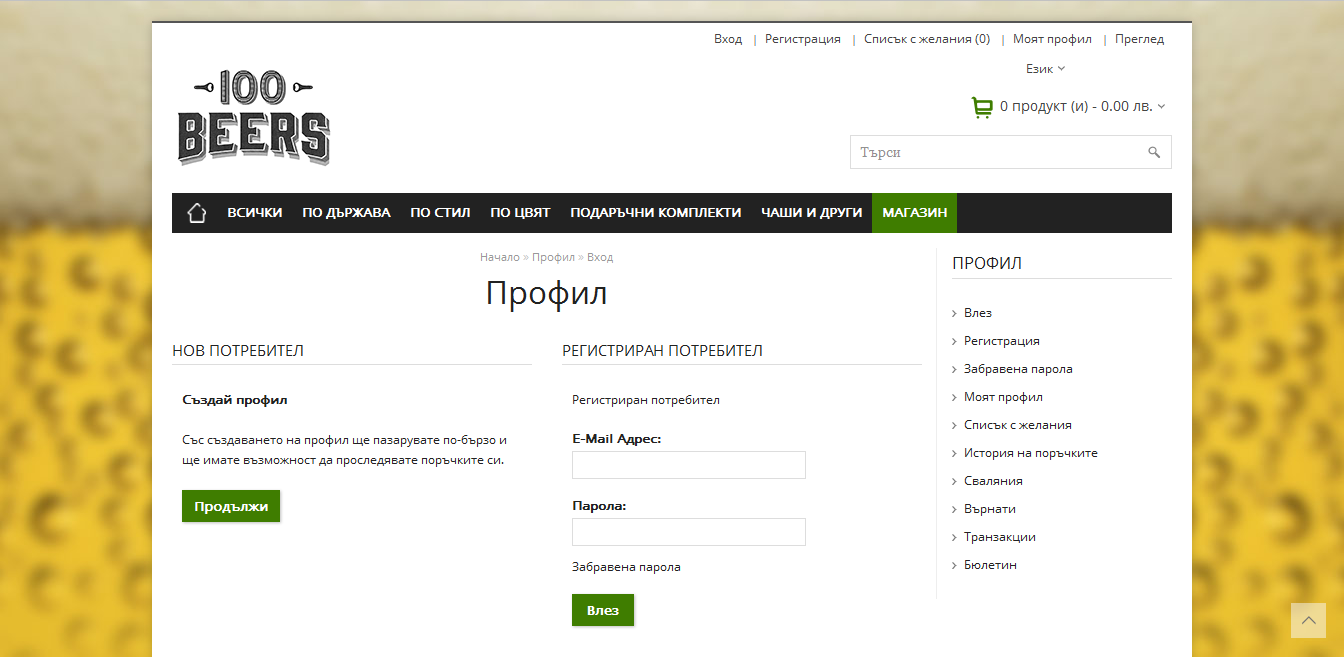
**Use case: User trying to send an enquiry with invalid email address.**

| Name | Sending message with invalid email address |
| --- | --- |
| Summary | This use case describes how the user has to send messages via the contact form. |
| Goal | The user should be unable to send a message with invalid email address. |
| Actor | Registered user |
| Preconditions | The actor is on the Cart screen and has already logged in. |
| Basic Flow | 1. The user is filling in all the fields, but the email address is wrong.  2. He is trying to send the enquiry with the wrong email address.  3. The user is clicking on the “Send button”.  4. The system is responding with an error message. |
| Alternative flows | N/A |
| Extensions | N/A |
| Post conditions | The user successfully removes the wrongly tapped email and can add a new one until the other details are saved. |

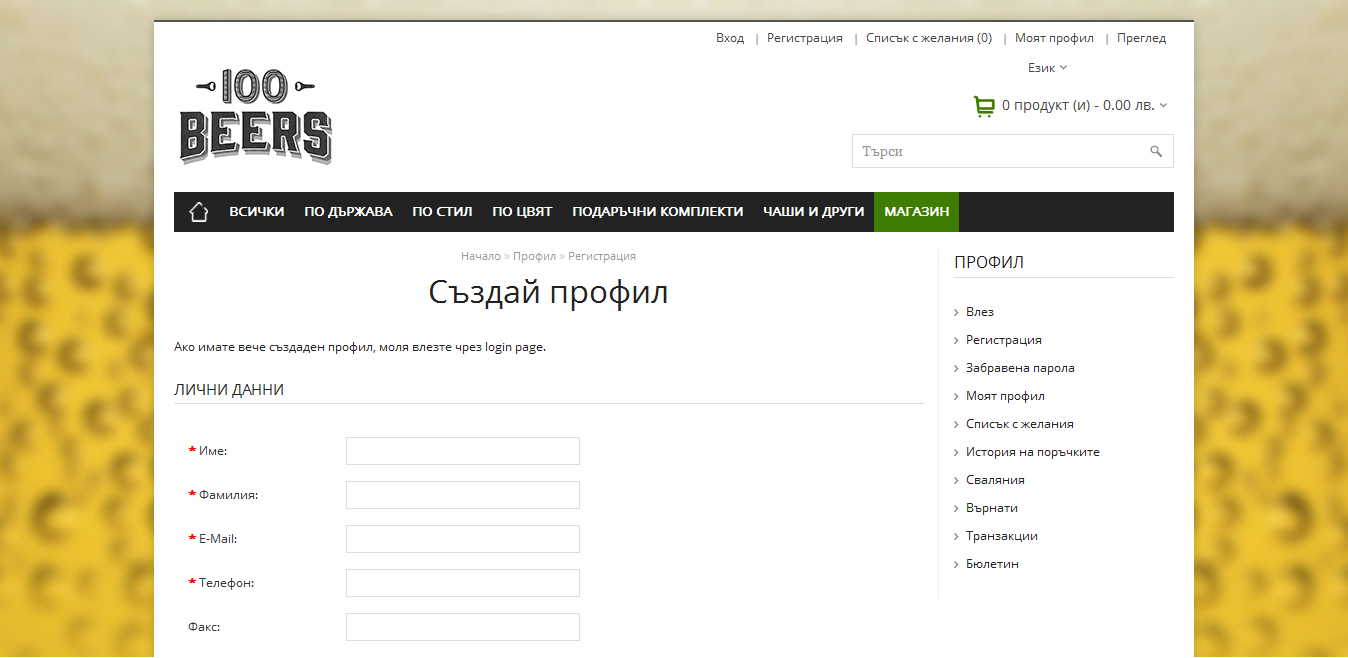
# Interface Requirements

## 4.1. GUI - Various interfaces for the product

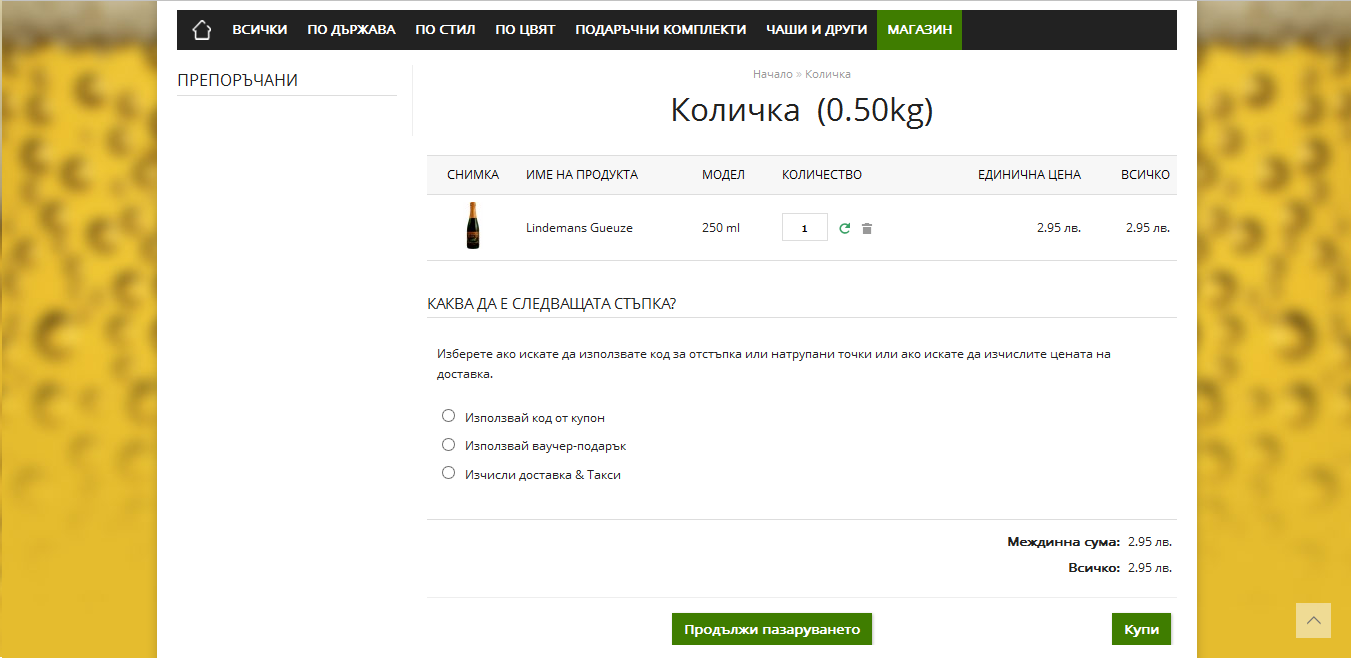
4.1.1 Login Page



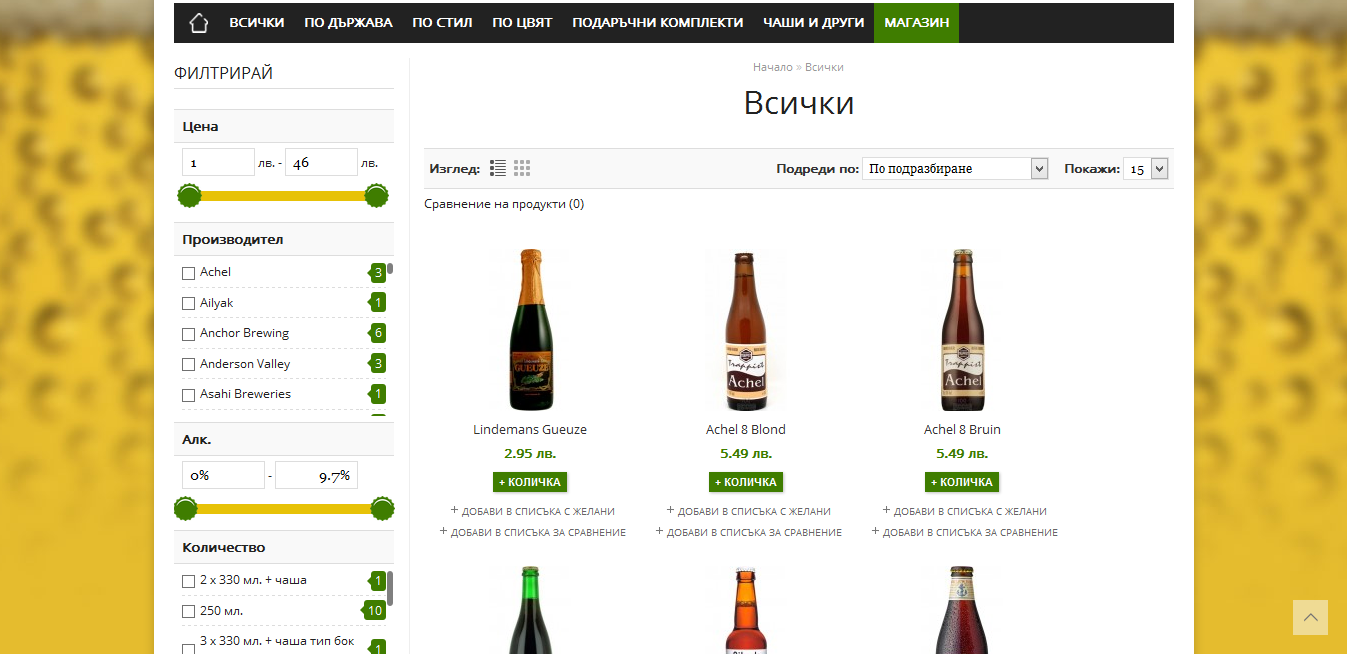
4.1.2 Registration Page



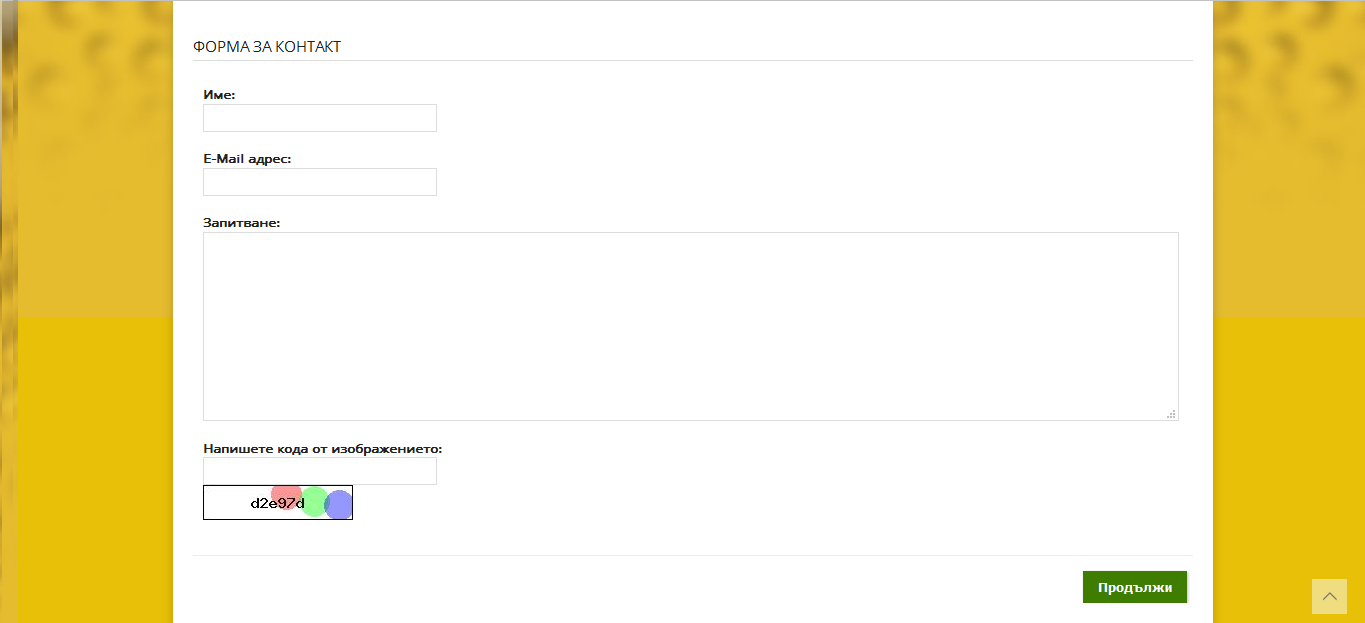
4.1.3 Cart

****

4.1.4 Product Page



4.1.4 Contact Form



## 4.2. Hardware interface

The System must run over the internet, all the hardware shall require to connect internet will be hardware interface for the system. As for e.g. WAN – LAN, Ethernet Cross-Cable.

## 4.3. Software interface

The system is on server so it requires the any scripting language like PHP, VBScript etc. The system require Data Base also for the store the any transaction of the system like MYSQL etc. system also require DNS(domain name space) for the naming on the internet. At the last user need web browser for interact with the system.

# Design constrain

The system shall be built using a standard web app development tool that conforms to standards like HTML, XML etc.

# Non - Functional Requirements

## Security

The system use SSL (secured socket layer) in all transactions that include any confidential customer information. The system must automatically log out all customers after a period of inactivity. The system should not leave any cookies on the customer’s computer containing the user’s password. The system’s back-end servers shall only be accessible to authenticated administrators. Sensitive data will be encrypted before being sent over insecure connections like the internet.

## Reliability

The system provides storage of all databases on redundant computers with automatic switchover. The reliability of the overall program depends on the reliability of the separate components. The main pillar of reliability of the system is the backup of the database which is continuously maintained and updated to reflect the most recent changes. Thus the overall stability of the system depends on the stability of container and its underlying operating system.

## Availability

The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs. In case of a of a hardware failure or database corruption, a replacement page will be shown. Also in case of a hardware failure or database corruption, backups of the database should be retrieved from the server and saved by the administrator. Then the service will be restarted. It means 24 X 7 availability.

## Maintainability

A commercial database is used for maintaining the database and the application server takes care of the site. In case of a failure, a re-initialization of the program will be done. Also the software design is being done with modularity in mind so that maintainability can be done efficiently.

## Portability

The application is HTML and scripting language based. So The end-user part is fully portable and any system using any web browser should be able to use the features of the system, including any hardware platform that is available or will be available in the future.

An end-user is use this system on any OS; either it is Windows or Linux.

The system shall run on PC, Laptops.