Toymania

Software Design Document

Name (s) and student numbers:

Ali Al Fatly	0944292
Tristan Kirwan	0944035
Hisham Agzanay	0944464
Jamie Surowiec	0942897
Liam de Waal	0943700

Date: (03/11/2018)

TABLE OF CONTENTS

1. Introduction

1.1 Purpose

The purpose of this document is to give a comprehensive overview of the design of the webshop that is to be created within the context of Project C.

1.2 Scope

The system is a toys webshop where there is an administrator and a customer login, the administrator have the ability to add/delete/update a product/customer account. The customer has the ability to update his/her balance, to purchase a product, to add a product to his/her wish list or cart and to view his/her view history. The webshop have a search engine which gives the customers and the administrators the ability to filter the products by category and price.

We will only be creating the software needed for the system and not the hardware.

1.3 Overview

This document contains the design of our system. In the course of this file we will show the design of the system and the design of our database. The end-product is a webshop. Our customer, henceforth named 'client', is the fictitious company Toymania, that specializes in children's toys. The webshop aims to sell toys to the parents of children from age 3 to age 12.

1.4 Reference Material

During the project, the following documents are referenced:

- General Data Protection Regulation (GDPR), known as the Algemene Verordening Gegevensbescherming (AVG) from April 27th 2016: https://autoriteitpersoonsgegevens.nl/sites/default/files/atoms/files/gdpr.pdf
- ISO 10008:2013, edition 1: https://www.iso.org/standard/54081.html

2. System Overview

2.1 Main functional goal

In this document we will make a webshop for toys. The main goal is to encourage our customers to buy at least one of the toys.

2.2 Conditions and regulations

The project end-product is an ecommerce website, selling toys, with admin functionalities ranging from managing the product database to managing orders and user-accounts. The user is supposed to be able to browse products, register, place an order and select favourite products. Actual payment functionalities are not part of the project scope.

The end-product has to comply with several standards and regulations:

- Keurmerk Webwinkels[1];
- Webwinkel Keur[2];
- Webshop Topper[3];
- Algemene Verordening Persoonsgegevens (AVG)[4];
- ISO 10008:2013, Quality Management and Quality Assurance[5].

2.3 SDLC

During the project, scrum will be used as SDLC. This implies that the end-product will be created incrementally during one-week periods (named sprints). At each meeting with the client (named Product Owner in SCRUM), the priorities for the upcoming sprint will be formulated in a sprint backlog and the past sprint will be reviewed. The team will choose 1 negative point to work on and to improve, this is called a retrospective. It is the explicit intention that after every sprint a potentially shippable product will be delivered. Backbone of the product is the so-called Product Backlog, a document describing all the components of the product. The scrum team and the product owner will also define the definition of done. The definition of done is an agreed-upon description of the end-product.

The SCRUM team is a multidisciplinary group of informatics students that govern themselves. The end-responsibility of the product lies with the product owner, who also has a last say in the design decisions. The scrum-master is a team-member, who assists in the case a team-member encounters an impediment during the process. The process is monitored each day during small stand-up meeting called 'Daily Scrum' where every team member will explain what he did yesterday, what he will do today and what he is expecting to be doing tomorrow. The team member is also able to ask for help if he is stuck. Each completed task is monitored using a burndown chart, which gives an overview of the completed and uncompleted tasks for the whole product.

2.4 Background

[1] https://www.webwinkelkeur.nl.

[2] https://www.webwinkelkeur.nl.

[3] https://www.webshoptoppers.nl.

[4] https://autoriteitpersoonsgegevens.nl/nl/onderwerpen/avg-europese-privacywetgeving/algemene-informatie-avg.

[5] https://www.iso.org/standard/54081.html.

The customer is an owner of a toys shop and is in need of a webshop to sell his products.

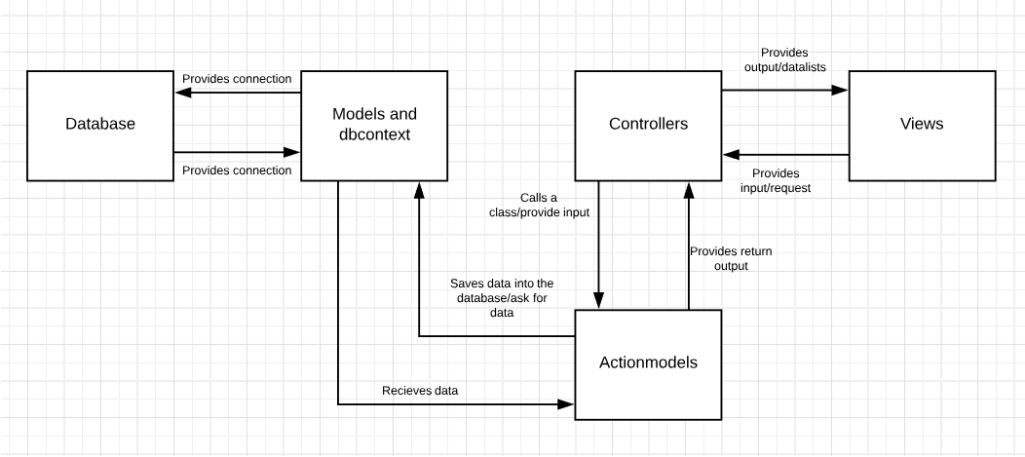
3. System Architecture

3.1 Architectural Design

The Software architecture used is the ASP.NET Core Web Application MVC 5 Model by Microsoft.

The modules are as follows:

1. Database: data storage module using MSSQLSERVER database;
2. TSE-models and TSE-dbcontext: representation of data tables as objects in the program;
3. Shoppingstoremodel: class used to define the functionalities of the shopping cart;
4. Shoppingstorecontroller,: used to handle requests from the Shoppingstoreview;
5. Shoppingstoreview: webpage for the shoppingstore;
6. Identitymodel: class used to define the functionalities of identity;
7. Identity Controller: used to handle Identity-related requests;
8. Main(home/shared) controller: handles requests from the Main view;
9. Main(home/shared) view: main webpage;
10. Statistics Modelview: class used to define the functionalities of the statistics page of the Admin area;
11. Statistics Controller: handles requests from the Statistics Modelview.



The modules start backend with the database. The local database is a separate entity connected to the project via ADO.Net Entity Framework. The connectionstring is the TSE-models and TSE-dbcontext. The database is also accessible by the controllers via the connectionstring.

Moving on to the action performing classes we have Shoppingstoremodel, identitymodel and statistics modelview. Those 3 classes contain a library of functions which gets implemented by their counterpart controller. Shoppingstorecontroller implements shoppingstoremodel, identity controller implements identitymodel and statistics controller implements statistics modelview. The return results of the implemented

function is taken by a view and then printed on the screen. The user can also interact with the webshop via the views. The views are capable of calling a controller based on an input or a click of a button. The controllers are also able to save an input into the database.

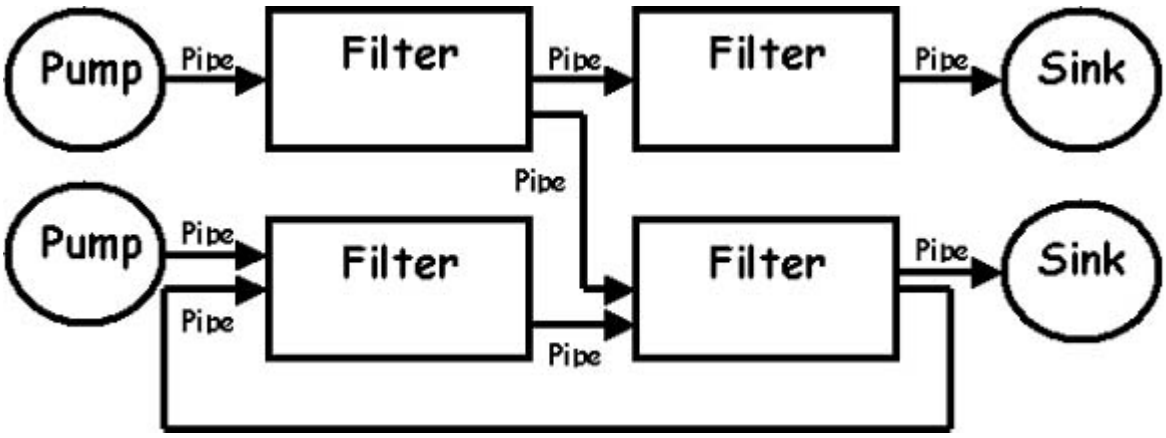
Viewmodels are used to provide a model for a view directly. This is most useful when the view has to employ some dynamism, as in the case of a shopping cart.

Detailed description of request handling & description of MVC architecture

The MVC model is best described following an actual user request to search the database. When browsing the available products on a web page, the user is actually searching through a selection of the products database. When wanting to view a specific product, the user clicks 'view details' next to the icon of the desired product. A request is sent to the controller, containing the specific product id of the desired product, which in turn searches the database for a record in the table of products containing that very id as primary key. When a match is found, the controller returns the corresponding view (in this case product detail), which renders a page with product details using the data from the record that has just been fetched from the database. The user can view the details of the selected product.

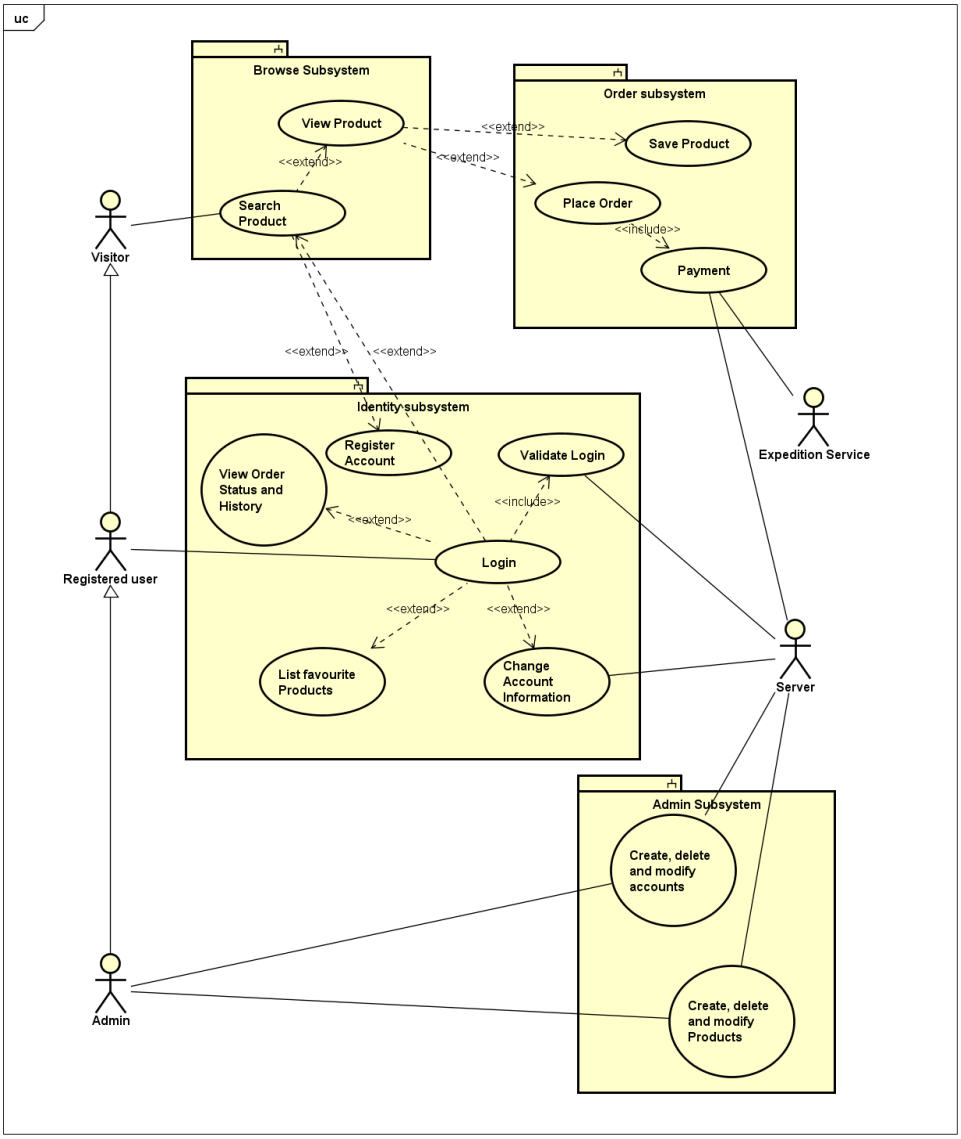
Pipe-filter architecture

The pipe-filter architecture is a data stream pattern. It is most suited for handling requests and searches. It works as following:
The pump is the data source, the pipe's transfer the data, the filter removes unrequested data and formats them and passes them on to the pipe, the sink is the end-user or software.



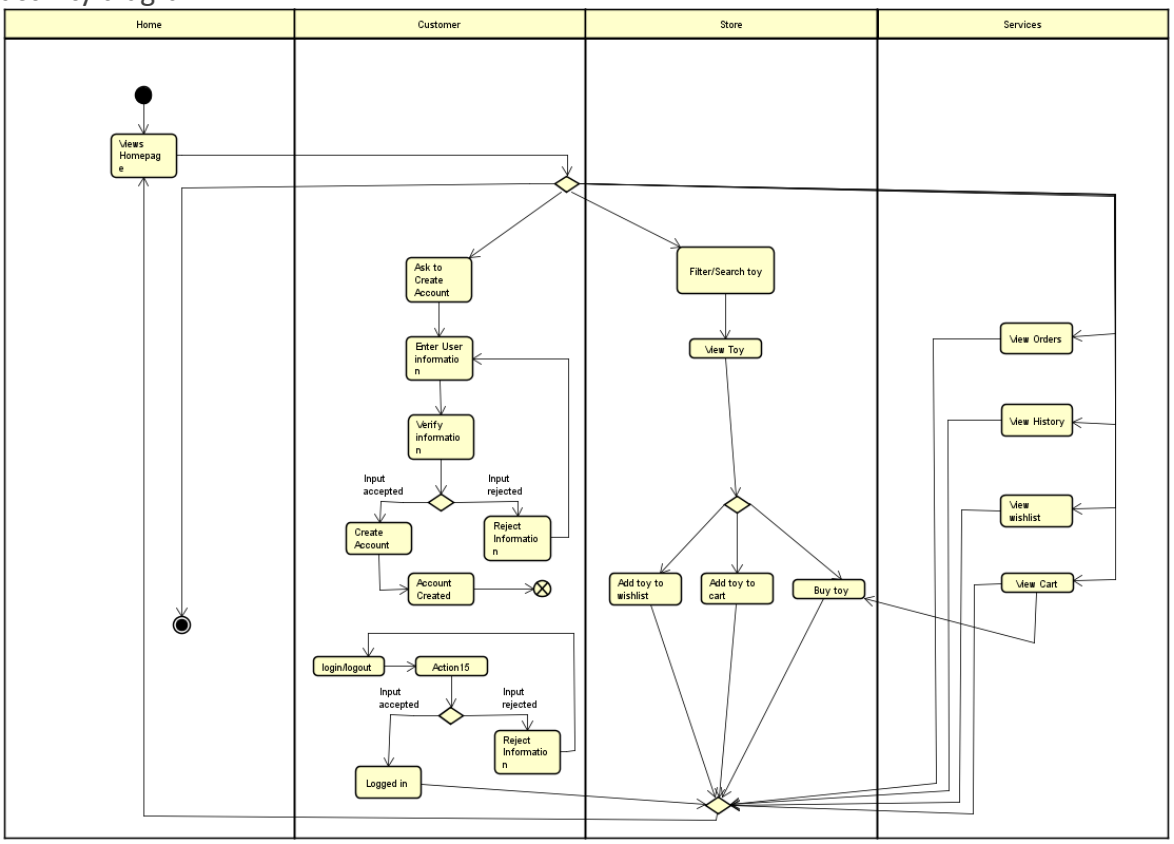
3.2 Decomposition Description

Use Case Diagram



Class diagram:
The Class diagram is added as a pdf reference.

activity diagram:



3.3 Design Rationale

We used asp.web MVC, because we are developing a complex webshop where the product owner wants multiple components working together. MVC is the most suitable architecture for this task, because MVC reduces the complexity of an application by dividing the logic into multiple components. This way we are capable of saving time instead of calculating how we are going to program a big task. We will also save time, because we will be able to work in parallel. That means that each individual will be working on a different component.

An alternative architecture would be the MVP architecture (Model-View-Presenter). The reason asp.web MVC has been chosen over MVP is that it almost works the same. the difference between MVC and MVP is that inside MVC the view can ask information from the model while in MVP the view must send a request to the presenter and the presenter will ask the information and send it back to the view. We see that this extra steps are necessary which slows down the performance of a web application while in a MVC this is not the case. A view can directly ask information from the model or viewmodel. We can also provide the security advantage of MVP inside a MVC project by wright a “[authorize]” class to define data classification.

We could have also chosen asp.web form, however MVC is ahead of asp.form security wise since it uses SOLID principles in its structure, as the actions are divided between multiple components. MVC is also easier to maintain since every component can work separately from the other.

Pipe-filter is less suited for a webshop because we need to show a lot of pages to the customers. This is the weakness of pipe-filter architecture. Showing a lot of different information can lead to complexity and a lot of workload while MVC does the job with less complexity.

Furthermore in 3.2 we made functional UML over OO UML, because the project is a small project and a general view which is given by functional UML is enough.

We didn't make OO UML, because it is unnecessary since the project is small and functional UML is enough.

4. Data Design

4.1 Data Description

The data within the application is stored using an SQL server database. In the MVC design pattern, the data structures are defined within the Model as classes and processed through the Controller. The Controller is able to request data from the database, using commands formulated in C#, which are automatically converted into SQL queries.

The database used is called TSE.

4.2 Data Description

See appendix I & II.

5. Component Design

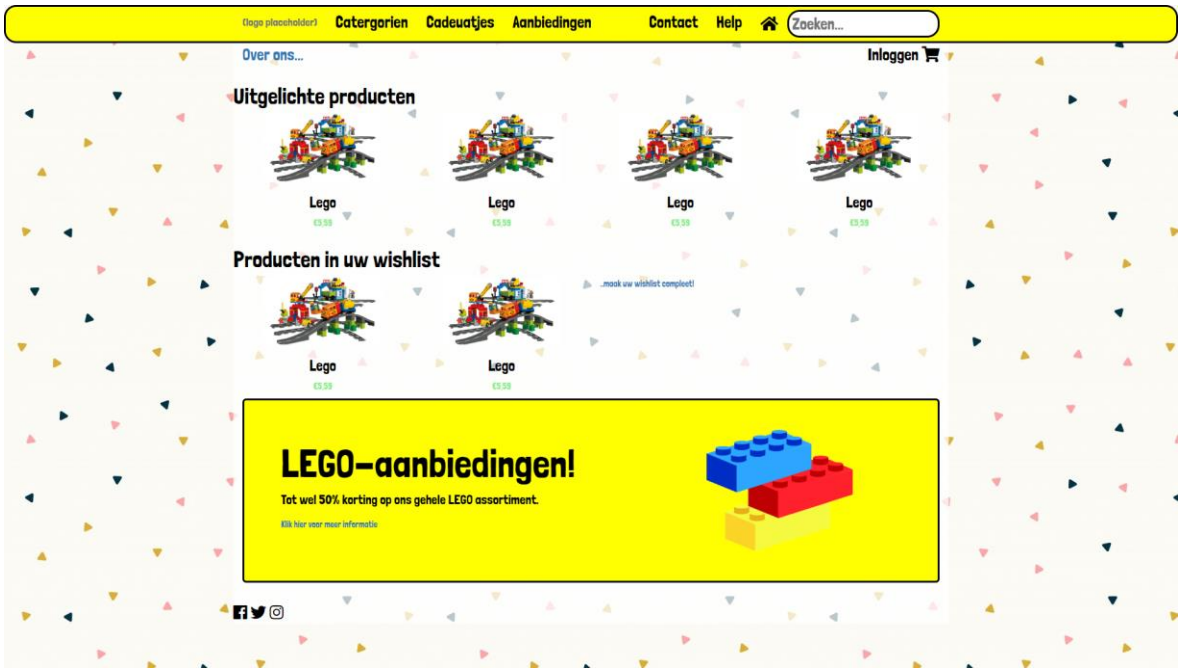
See appendix I , II.

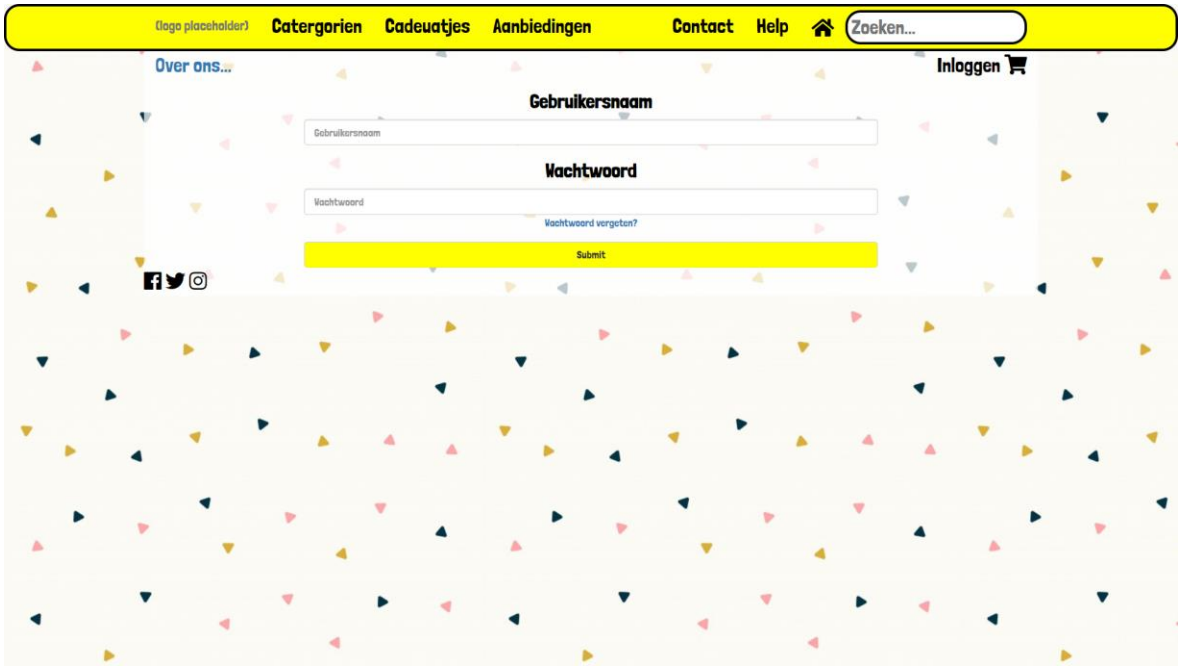
6. HUMAN INTERFACE DESIGN (high level)

6.1 Overview of User Interface

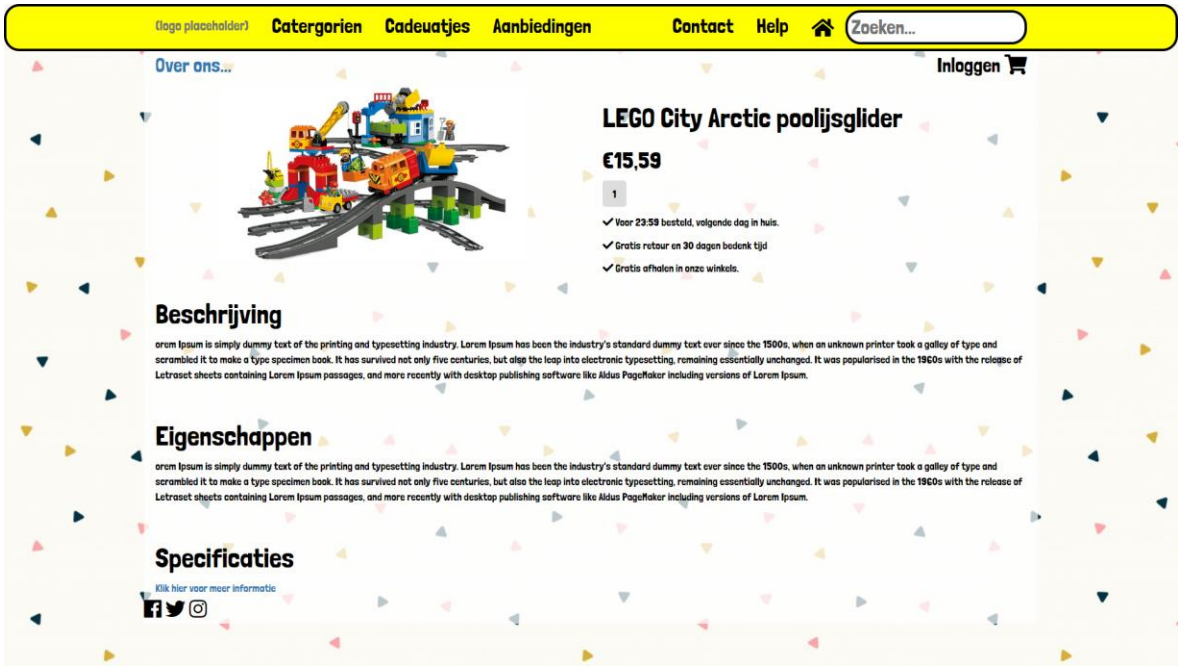
As a user, I must be able to look through all the different toys that are available in the webshop, search through items and buy items. If I'm thinking about buying an item, I should be able to save it in a wishlist. When I'm sure about buying an Item, I must be able to buy it through a shopping cart. If I'm not logged in yet, I must be able to login and after that I don't need to give my personal details again. If I'm not logged in yet, I must be able to login or register for a new account. When I'm logged in, I must be able to see or edit my personal details, see my order history **(and see my credit?)**.

6.2 Screen Images

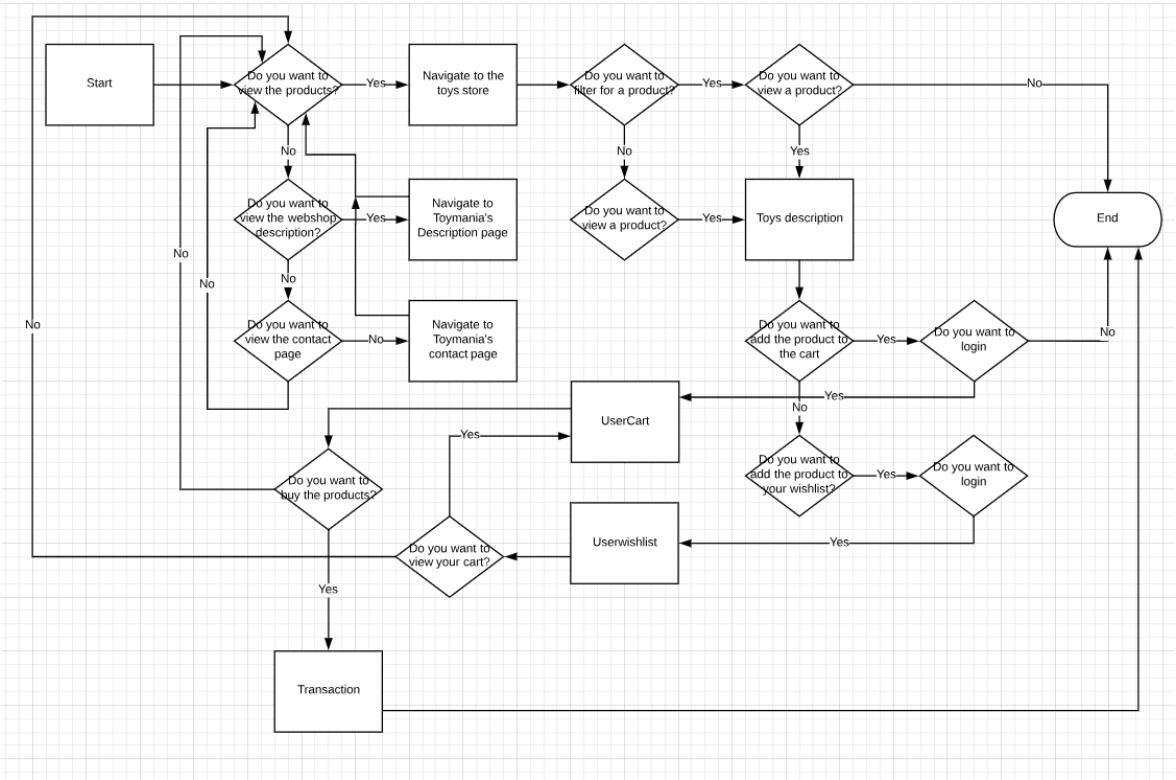



(logo placeholder)
Catergorien
Cadeuatjes
Aanbiedingen
Contact
Help
Zoeken...
Over ons...
Inloggen
Gebruikersnaam
Gebruikersnaam
Wachtwoord
Wachtwoord
Wachtwoord vergeten?
Submit

`

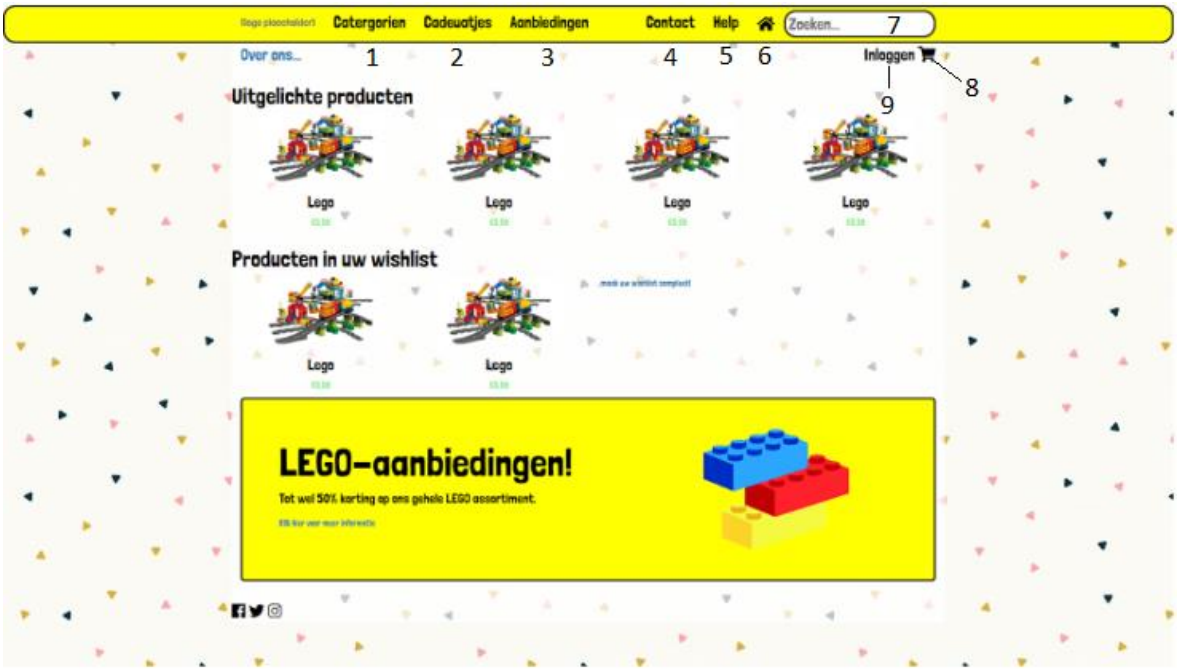

(logo placeholder)
Catergorien
Cadeuatjes
Aanbiedingen
Contact
Help
Zoeken...
Over ons...
Inloggen
LEGO City Arctic poolijsglider
€15,59
1
✓ Voor 23:59 besteld, volgende dag in huis.
✓ Gratis retour en 30 dagen bedenk tijd
✓ Gratis afhalen in onze winkels.
Beschrijving
orem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
Eigenschappen
orem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
Specificaties
Klik hier voor meer informatie

Navigation Flowchart



6.3 Screen Objects and Actions

There are the following screen objects and actions:



Navigation buttons:

1. Categorieën : this button will show all the category
2. cadeautjes : this button will show all the presents
3. aanbiedingen : this button will redirect to a page with the sales of the week
4. contact : this button will redirect the user to the contact page
5. help : this button will redirect to the page where the user can ask for help
6. home : this button will redirect the user to the home page
7. zoeken : the user can search any products through the search engine
8. cart : the user can add products to the shopping cart
9. Inloggen : the user can create a account on the webshop or log in

Action buttons:

1. Add to cart
2. Buyout
3. Remove from cart

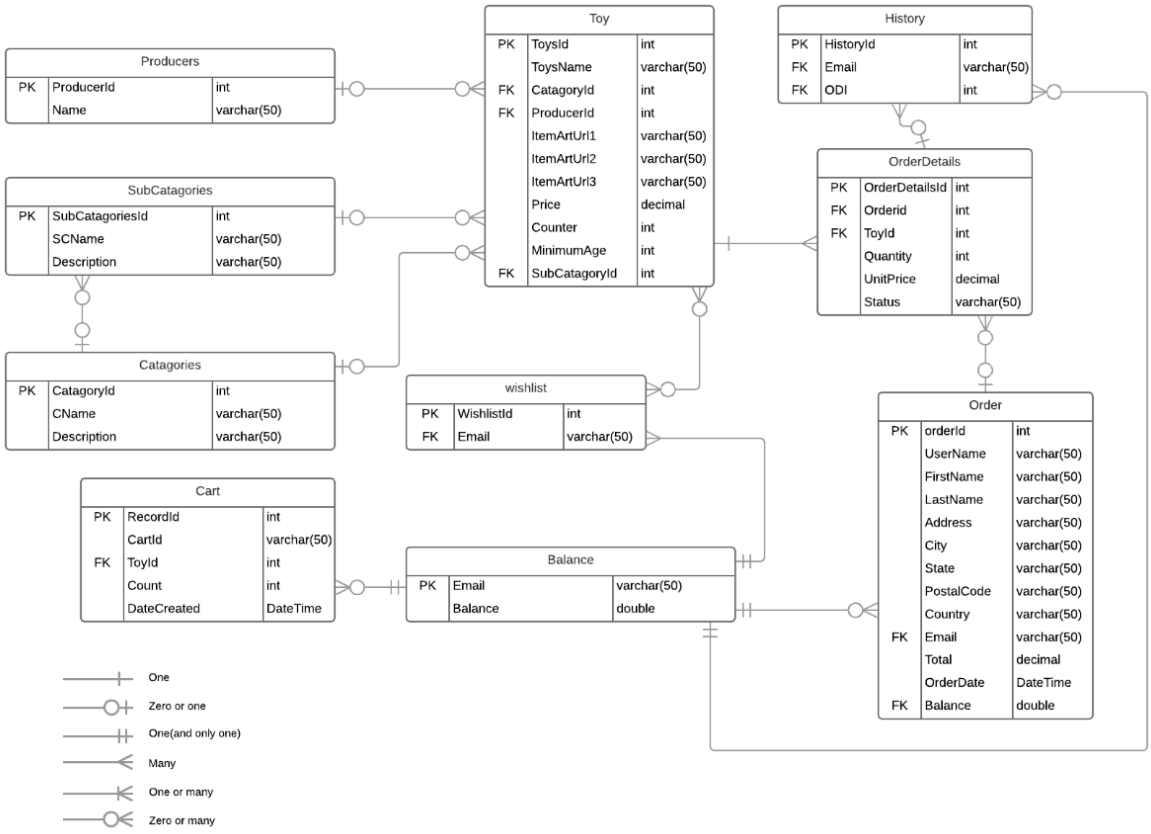
7. Requirements Matrix

The User stories and requirements were originally written down in Dutch, they were translated to English in order to maintain consistency within the SSD.

System component	Functional requirement code	Subsystem
Search bar function	M0001	Browse
Registration and login menu function	M0002	Identity
Wishlist function	M0003	Identity
Shopping cart function	M0004	Order
Order history functionality	M0005	Identity
Administrator customer account management	M0006	Admin
System database connection	M0007	All
Login data validation	M0008	Identity
Administrator visualization	M0009	Admin
Homepage button	M0010	Browse
Most sold article display	S0001	Browse
Price filter options	S0002	Browse
Functional navigation bar	S0005	Browse
Coupon code payment service	S0006	Order
Track and trace service	W0001	Order
Recommended products function	W0002	Browse

8. Appendices

Appendix I(ERD)



Functions	
LoginConfirmation	A function that redirects the user to a confirmation page
VerifyCode	Verifies if a given password or Email-Address is according to the set rules
Login	Validates login information
PasswordForgot	Redirect to password reset page
Register	Allows the customer to register an account
SetPassword	Sets password for an account
ConfigureTwoFactor	Configures two-factor authentication for an account
LastRecord	Gets the last added record from a database
GetCount	Gets the amount of records
GetCart	Gets the content of a shopping cart
AddToCart(Toy Toy)	Adds data to a shopping cart
RemoveFromCart(int id)	Removes a record from a shopping cart
EmptyCart(HttpContextBased Context)	Empties the content of a shopping cart
GetCartProduct	Returns the products in the cart
CreateOrder(Order Order)	Takes an order as input and registers it into the database
GetCartId(HttpContextBase Context)	Returns the cart id/id's depending on the current page
MigrateCart(string email)	Makes sure that the cart is set for the correct Email-Address
Produce statistics (IEnumerable<list<T>> Tables)	Takes a table or multiple tables in as input and produce statistics as output

Appendix II

Component diagram of browsing and ordering functionalities.