

# **Strawberry - webeC Project**

Livio Bieri (livio.bieri@students.fhnw.ch)

07.03.2015

# Contents

<b>1. Introduction</b>	<b>3</b>
1.1. What's the point? . . . . .	3
<b>2. Requirements</b>	<b>5</b>
2.1. Usecase Overview . . . . .	6
2.1.1. Usecase: Login with username and password . . .	7
2.1.2. Usecase: Logout . . . . .	7
2.1.3. Usecase: Display recent changes to shopping lists	8
2.1.4. Usecase: Add user to existing shopping list . . .	8
2.1.5. Usecase: Remove item to shopping lists . . . . .	9
2.1.6. Usecase: Create new shopping lists . . . . .	10
<b>3. Conceptual Data Model</b>	<b>11</b>
3.1. Overview . . . . .	11
3.1.1. Entity: User . . . . .	11
3.1.2. Entity: Shoppinglist . . . . .	11
3.1.3. Entity: Item . . . . .	12
3.1.4. Entity: Notification . . . . .	12
3.2. Operations . . . . .	12
<b>A. Appendix</b>	<b>13</b>
A.1. Development Workflow . . . . .	13
<b>References</b>	<b>14</b>

# 1. Introduction

## 1.1. What's the point?

**Strawberry** is a collaborative shopping list. Strawberry aims to simplify doing groceries in households with more than one person. No more problems because you forgot that piece of paper called grocery list at home or you did not see the message from your girlfriend telling you to bring that one important item.

Strawberry  
the collaborative shopping list



## 1. Introduction

---

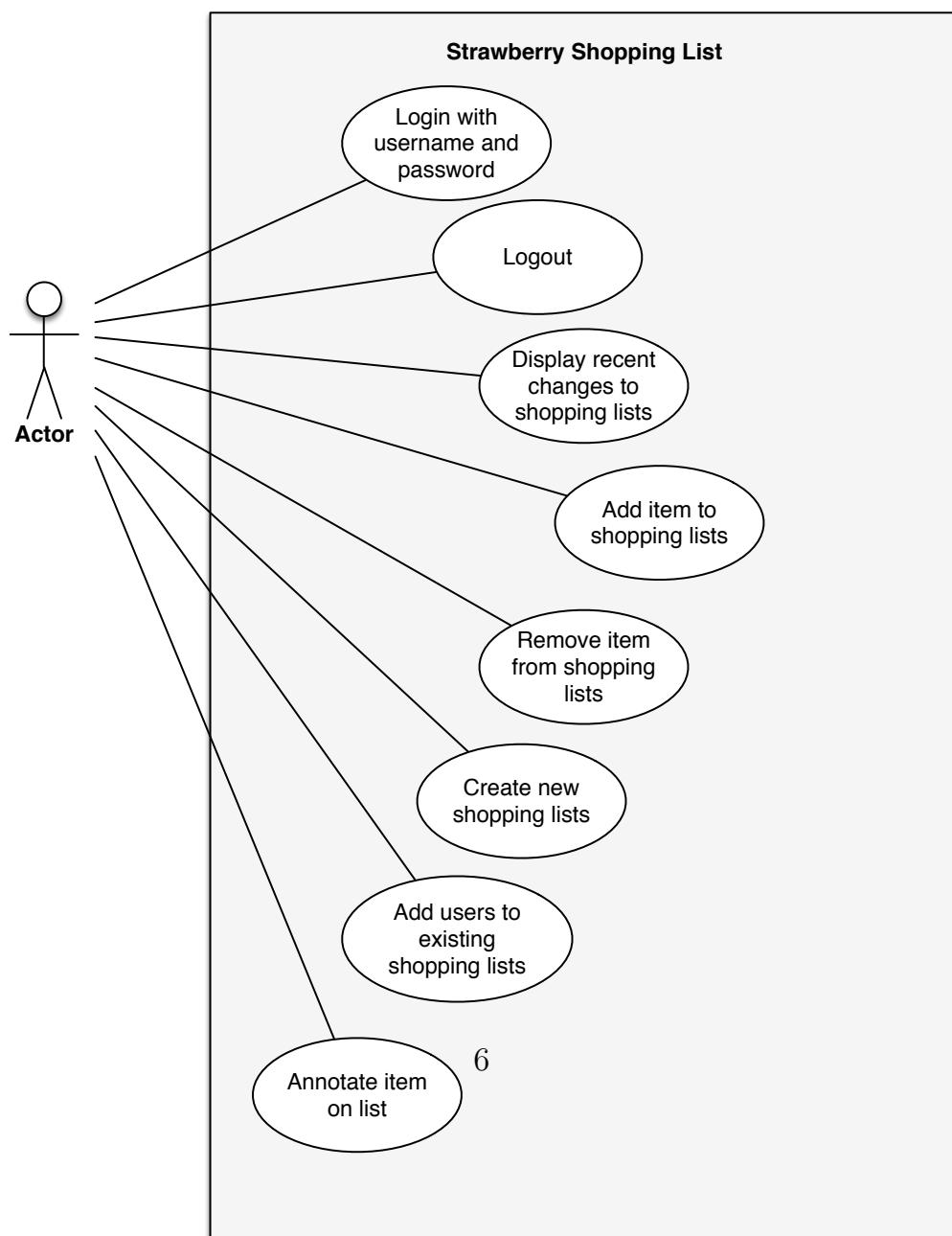
**Strawberry** in this early version will support the following features:

- **Shared lists** with as many users as you wish. You can simply invite everyone with a Strawberry account to take part in your list. Each user can many have as many seperate lists with different people as he wants. For example: *“me and girlfriend”*, *“me and family”* or *“me and my flatmates”*. Of course it is also possible to have a private list just for yourself.
- **Simple item-based lists** simply add items and remove them by checking them off the list. You can add a short note to each item seperately. For example if you want to have a certain brand of chocolate.
- **Twitter Style Typeahead** with common shopping items. Such as “Apples”, “Bananas” or “Strawberries” in item input field.



## 2. Requirements

### 2.1. Usecase Overview



## 2. Requirements

---

### 2.1.1. Usecase: Login with username and password

---

Use Case	Login with username and password
<b>Description</b>	Allows user to login with his username + password
<b>Actors</b>	User
<b>Preconditions</b>	Not logged in (see /loggedin response)
<b>Basic Flow</b>	User fills in username + password, clicks login
<b>Alt. Flow</b>	None
<b>Postconditions</b>	<i>User is logged in (has session token)</i>
<b>Notes</b>	-

---

### 2.1.2. Usecase: Logout

---

Use Case	Logout
<b>Description</b>	Logout user from running logged in session.
<b>Actors</b>	User
<b>Preconditions</b>	Logged in (see /loggedin response)
<b>Basic Flow</b>	User clicks logout button.
<b>Alt. Flow</b>	User enters URL /logout
<b>Postconditions</b>	<i>User is logged out (has session token revoked)</i>
<b>Notes</b>	-

---

## 2. Requirements

---

### 2.1.3. Usecase: Display recent changes to shopping lists

---

Use Case	Display recent changes to shopping lists
Description	Shows a history of recent changes.
Actors	User
Preconditions	Logged in (see /loggedin response)
Basic Flow	User pressed the bell-icon in the top right.
Alt. Flow	None.
Postconditions	List is shown.
Notes	-

---

### 2.1.4. Usecase: Add user to existing shopping list

---

Use Case	Add user to existing shopping list
Description	User can invite others to existing shopping lists.
Actors	User
Preconditions	Logged in (see /loggedin response), list exists.
Basic Flow	User searches / enters usernames and presses add.
Alt. Flow	None.
Postconditions	User has been added to the list as collaborator.
Notes	-

---



## 2. Requirements

---

### **2.1.5. Usecase: Remove item to shopping lists**

see *Usecase: Add item to shopping lists*. Same as this but instead of adding item will be removed.

## 2. Requirements

---

### 2.1.6. Usecase: Create new shopping lists

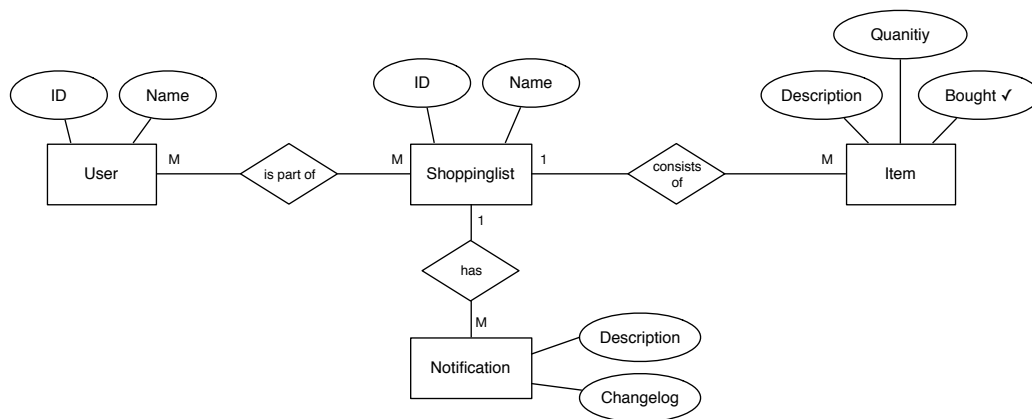
---

Use Case	Create new shopping lists
<b>Description</b>	Creates a new empty shopping list.
<b>Actors</b>	User
<b>Preconditions</b>	Logged in (see <code>/loggedin</code> response)
<b>Basic Flow</b>	User enters the name of the new list and clicks add.
<b>Alt. Flow</b>	None.
<b>Postconditions</b>	A new empty shopping list has been created.
<b>Notes</b>	-

---

## 3. Conceptual Data Model

### 3.1. Overview



#### 3.1.1. Entity: User

A user of the system (*as in member of (many / one) shopping lists*).

#### 3.1.2. Entity: Shoppinglist

Set of items for one or several users. The list has **Notifications** about the latest changes so all members can see the latests changes to the list.

### 3. Conceptual Data Model

---

#### 3.1.3. Entity: Item

An item is defined as following: A **description** such as *Strawberry*, a **quantity** for example *some*, *one* or *many* and a **bought** checkmark which indicates an item that has been ticked of the the list (*thus no longer is required to be bought*).

#### 3.1.4. Entity: Notification

*see Shoppinglist.*

### 3.2. Operations

---

Actor	Operation	Arguments	Result
User	Add item to list	<b>item</b>	<b>success</b> or <b>fail</b>
User	Tick item on list	<b>item</b>	Changes <b>bought</b> to <b>true</b>
User	Add user to list	<b>user</b>	<b>success</b> or <b>fail</b>
User	Remove user from list	<b>user</b>	<b>success</b> or <b>fail</b>
User	Login / Logout	<b>un/pw</b>	<b>Session ID</b>
Server	Notify change	<b>changes</b>	-

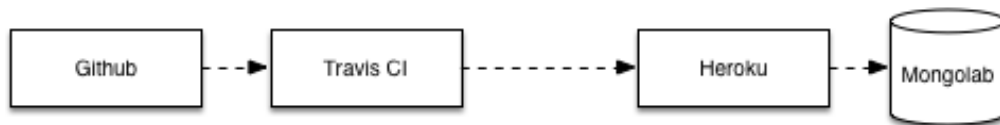
---

**Notes:** **fail** could be returned when you try to access a list via id you are not a member of.

# A. Appendix

## A.1. Development Workflow

The development environment and workflow is straightforward:



By default there is a [git pre-commit hook](#) setup which will run the unit tests locally and ensure the code is according the guidelines (jscs, jshint). Once committed locally changes will be pushed to Github and Travis (Continues Integration) will compile the project and run the unit tests once more. As soon this is done and there are no errors the code will be deployed automatically on Heroku.

# References