**Airell Rasendriya Bachtiar**

***‘t Sloepke B.V.***

**User Requirements Specification**

Boat Renting

# Table of Contents

[Table of Contents 1](#_Toc90989378)

[**1.** **Introduction** 2](#_Toc90989379)

[1.1. Purpose 2](#_Toc90989380)

[1.2. Project Scope 2](#_Toc90989381)

[1.3. Planning 2](#_Toc90989382)

[**2.** **Overall Description** 3](#_Toc90989383)

[2.1. Product Perspective 3](#_Toc90989384)

[2.2. Product Features 3](#_Toc90989385)

[2.3. User Classes and Characteristics 3](#_Toc90989386)

[2.4. Operating Environment 4](#_Toc90989387)

[**3.** **System Features** 4](#_Toc90989388)

[3.1. Core Requirements 4](#_Toc90989389)

[3.2. Major Requirements 4](#_Toc90989390)

[3.3. Minor Requirements 4](#_Toc90989391)

[3.4. Functional Requirements 4](#_Toc90989392)

[3.4.1. Desktop Application 4](#_Toc90989393)

[3.4.2. Web Application 5](#_Toc90989394)

[**4.** **Functional Requirements** 5](#_Toc90989395)

[**5.** **Use Cases** 5](#_Toc90989396)

[5.1. Desktop Application Use Case 5](#_Toc90989397)

[5.2. Web Application Use Case 9](#_Toc90989398)

[**6.** **User Interface** 12](#_Toc90989399)

[**7.** **External Interface Requirements** 18](#_Toc90989400)

[7.1. User Interfaces 18](#_Toc90989401)

[7.2. Hardware Interfaces 19](#_Toc90989402)

[7.3. Software Interfaces 19](#_Toc90989403)

[8. **Non-functional Requirements** 19](#_Toc90989404)

[8.1. Safety Requirements 19](#_Toc90989405)

[8.3. Software Quality Attributes 19](#_Toc90989406)

# **Introduction**

## Purpose

The purpose of this document is to build a desktop application and web application for the company, *‘t Sloepke B.V.,* that will support their reservation process of boats and additional items/equipment.

## Project Scope

The purpose of this project is to support and to make ease the process of renting boat for customer and employee. This project will have a web application for customer that wants to rent a boat in this company. It will make customer easier to check in and check out the offer *‘t Sloepke B.V.* have. Additionally, for employee, this project will have desktop application for easier management of the rented items in this company. This project will have a database server that holds items such as boats and records for easier overview.

## Planning

This project starts on week 12, Monday, 21 November 2021, and will be finished on week 16 before Wednesday, 22 December 2021. For the convenient of this chapter, we will refer week 12 to week 1 and week 16 to week 5.

Week 1 is the start of this project, the goal for this week will be to make User Requirements Specification, Test Plan, Unified Modelling Language or UML Diagram, and initialize Visual Studio 2019. We expect User Requirements Specification and Test Plan will be finished and can be changed based on the feedback we got from the client. UML Diagram is in motion and it will have rough sketch of how it will work.

Week 2 will be finishing off the User Requirements Specification and Test Plan based on feedback from the client. UML Diagram will be finished and ready for feedback with the client. This week we can start on implementing the class based on UML Diagram.

Week 3 will be finishing off the UML Diagram and can be changed based on need from the applications and feedback from the client. We can continue to implement the application this week.

In week 4 we will start making the design for both Desktop Application and Website Application and continue to implement the code based on the UML.

Week 5 will be testing week to find all sort of bugs inside the application. In this week we can start working on the Test Report from the Test Plan we made in week 1.

For better visualization, below is the table for the timeline.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Objectives | 1 | 2 | 3 | 4 | 5 |
| User Requirements Specification |  |  |  |  |  |
| Test Plan |  |  |  |  |  |
| UML Diagram |  |  |  |  |  |
| Desktop Application |  |  |  |  |  |
| Web Application |  |  |  |  |  |
| Test Report |  |  |  |  |  |

# **Overall Description**

## Product Perspective

Boat renting service system will store the following information.

* Boat details:

Type of boats, kayak, canoe, sailboat laser and sailboat valkf. Quantity of each type of boats.

* Item details

Name of items, such as, life jackets, waterproof containers, etc. and quantity of each item.

* Customer details:

Name of customer, address, phone, and email are going to be recorded.

* Reservation details:

Date of reservation, location, boats and items, reservation number, rent duration, and price.

## Product Features

The major feature of this product is customer will be able to rent boats and items through a web application. Employee of this company will be able to manage all the items that can be rented.

Additional features of this system are coupons and maintenance of boats. Coupons can be added by employees from the desktop application and can be applied when customer is on the process of reservation for a rent. Employees will be able to manage the boats

## User Classes and Characteristics

User of the system can see the information of prices listed for renting boats and items. The system will support 2 types of user privileges, Customer and Employee. Customers will have access to customer function and Employees will have access to both customer function and employee function.

The customer should be able to do the following function:

1. Customer should be able to make a reservation of boats with selected date, type of boats, duration of renting, location, and additional items
2. Customer should be able to edit or cancel the reservation 7 days before the agreed date of reservation
3. Customer should be able to apply coupons for extra discount on checkout

The employee should have the following function:

1. Employee should be able to manage the boats and items of the company’s inventory
2. Employee should be able to view, edit, and remove reservation that has been made
3. Employee should be able to make coupons for discount on the renting process
4. Employee should be able to choose a boat for maintenance and make it unavailable for a period of time

## Operating Environment

* Operating System: Windows 10 or higher
* Desktop Application: Windows Forms App, C#
* Web Application: ASP.NET Core Razor Pages
* Database: MySQL

# **System Features**

## Core Requirements

* FR-01: Manage Catalogue with Boats and Items

Employees at the main office must be able to manage the boats and items of the company’s inventory.

* FR-02: Support Rental Process

Customer must be able to reserve and have access to manage their own reservation in web application.

Employees must be able to manage a reservation of a customer to allow them to do their job. They cannot create a reservation as an employee.

## Major Requirements

* + FR-06: Coupons

Customer will be able to use 1 coupon per reservation in return of rewards such as, discount. Coupons can be created by the employee in the desktop application.

3 types of coupons are, unlimited coupons, limited with certain number of times the coupon can be used, and usable by specific user for 1 time.

## Minor Requirements

* + FR-08: Maintenance of Boats

Employee will be able to mark a boat as “under maintenance” for a certain period of time. When the boat is under maintenance it is unavailable for rental. System will show the last time since the boat is last maintained so employee can choose to service the boat.

## Functional Requirements

### Desktop Application

* + FR-DA-01: Login

Employee should be able to login and operate the desktop application.

* + FR-DA-02: Manage Catalogue of Boats and Items

Employee should be able to create, edit, view, and remove the boats and items of the company’s inventory.

* + FR-DA-03: Manage Customer Reservation

Employee should be able to edit, view, and remove reservation from customer.

* + FR-DA-04: Registration

Employee should be able to register as an employee

* + FR-DA-05: Coupons Creation

Employee should be able to make coupons, choose coupon types, and discount amount

* + FR-DA-06: Maintenance of Boats

Employee should be able to choose boat to be unavailable due to maintenance

### Web Application

* + FR-WA-01: Login

Customer should be able to login to the web application

* + FR-WA-02: Registration

Customer should be able to create new account in the web application

* + FR-WA-03: Manage Reservation

Customer should be able to create, edit, view, and remove reservation in the web application.

* + FR-WA-04: View Items

Customer should be able to view all of the items that available and can be rented

* + FR-WA-05: Apply Coupon

Customer should be able to apply coupon on checkout in the renting process

# **Requirements**

|  |  |  |
| --- | --- | --- |
| **ID** | **Requirements** | **MoSCoW** |
| FR-DA-01 | Employee should be able to login to the system | **Must** |
| FR-DA-02 | Employee should be able to manage catalogue of boats and items | **Must** |
| FR-DA-03 | Employee should be able to manage reservation | **Must** |
| FR-DA-04 | User should be able to register to the system as an employee | **Must** |
| FR-DA-05 | Employee should be able to create coupons | **Must** |
| FR-DA-06 | Employee should be able to assign a boat to maintenance | **Must** |
| FR-WA-01 | Customer should be able to login to the website | **Must** |
| FR-WA-02 | User should be able to register to the website as a customer | **Must** |
| FR-WA-03 | Customer should be able to create and manage reservation | **Must** |
| FR-WA-04 | Customer should be able to view all items | **Must** |
| FR-WA-05 | Customer should be able to apply coupon to the reservation | **Must** |
| FR-AD-01 | Web application should have a good user interface for the customer | **Should** |
| FR-AD-02 | Desktop application can manage customer | **Could** |
| FR-AD-03 | Customer should be able to pay online | **Won’t** |

# **Use Cases**

There are 2 different actors, Employee and Customer. Employee can get access to the system through desktop application and Customer can access the system through web application.

## Desktop Application Use Case

**ID:** UC-DA-01

**Use Case:** Employee Registration

**Functional Requirement:** FR-DA-04

**Actor:** Employee

**Prerequisite:** Desktop application is opened

**Main Success Scenario:**

1. User chooses login option
2. User chooses register option
3. System asks for username, email, and password
4. User fills in the credentials
5. User confirms the registration
6. System registers the new user and shows the login page

**Extensions:**

4a. User fills in wrong credentials or doesn’t fill credentials

1. System shows error message
2. System resets the fields
3. Return to step 3

**ID:** UC-DA-02

**Use Case:** Employee Login

**Functional Requirement:** FR-DA-01

**Actor:** Employee

**Prerequisite:** Desktop Application is opened

**Main Success Scenario:**

1. User chooses login option
2. System asks for username and password
3. User fills the credentials
4. User confirm login
5. System shows the list of all items that can be rented

**Extensions:**

3a. User fills in wrong credentials or doesn’t fill anything

1. System shows error message
2. System resets the fields
3. Return to step 2

**ID:** UC-DA-03

**Use Case:** View Rent Item

**Functional Requirement:** FR-DA-02

**Actor:** Employee

**Prerequisite:** UC-DA-02

**Main Success Scenario:**

1. User chooses view all item page
2. System shows the list of all items that can be rented

**ID:** UC-DA-04

**Use Case:** Add Item

**Functional Requirement:** FR-DA-02

**Actor:** Employee

**Prerequisite:** UC-DA-02

**Main Success Scenario:**

1. User chooses add item option
2. System asks for item’s information to be filled in
3. User fills in the information
4. User confirms to add item
5. System adds the item to

**Extensions:**

3a. User fills in wrong information and doesn’t fill a required information

1. System shows an error message
2. System resets the fields
3. Return to step 2

**ID:** UC-DA-05

**Use Case:** Edit Item

**Functional Requirement:** FR-DA-02

**Actor:** Employee

**Prerequisite:** UC-DA-02 and UC-DA-04

**Main Success Scenario:**

1. User chooses an existing item
2. User chooses to edit item
3. System shows all information of the chosen item
4. User changes the information of the chosen item
5. User saves the changes
6. System shows the list of all items

**Extensions:**

1a. User doesn’t choose any item

1. System shows error message
2. To step 6

4a. User puts wrong information or puts empty fields

1. System shows error message
2. Return to step 3

5a. User doesn’t save the changes

1. Item doesn’t change
2. To step 6

**ID:** UC-DA-06

**Use Case:** Remove Item

**Functional Requirement:** FR-DA-02

**Actor:** Employee

**Prerequisite:** UC-DA-02 and UC-DA-04

**Main Success Scenario:**

1. User chooses an existing item
2. User chooses to edit item
3. System shows all information of the chosen item
4. User chooses remove option
5. System shows the list of items

**Extensions:**

1a. User doesn’t choose any item

1. System shows error message
2. To step 6

**ID:** UC-DA-07

**Use Case:** View Reservation

**Functional Requirement:** FR-DA-03

**Actor:** Employee

**Prerequisite:** UC-DA-02 and Reservation has been made

**Main Success Scenario:**

1. User chooses show all reservation option
2. System shows the list of all reservations that has been made

**ID:** UC-DA-08

**Use Case:** Edit Reservation

**Functional Requirement:** FR-DA-03

**Actor:** Employee

**Prerequisite:** UC-DA-02 and Reservation has been made

**Main Success Scenario:**

1. User chooses an existing reservation
2. User chooses to edit reservation
3. System shows all the information of the chosen reservation
4. User changes the information of the chosen reservation
5. User saves the changes
6. System shows the list of all reservation

**Extensions:**

1a. User doesn’t choose a reservation

1. System shows error message
2. To step 6

4a. User puts wrong information or leaves it empty

1. System shows error message
2. Return to step 3

5a. User doesn’t save the change

1. Reservation doesn’t change
2. To step 6

**ID:** UC-DA-09

**Use Case:** Remove Reservation

**Functional Requirement:** FR-DA-03

**Actor:** Employee

**Prerequisite:** UC-DA-02 and Reservation has been made

**Main Success Scenario:**

1. User chooses an existing reservation
2. User chooses to edit reservation
3. System shows all the information of the chosen reservation
4. User chooses remove reservation option
5. System removes the selected reservation

**Extensions:**

1a. User doesn’t choose a reservation

1. System shows error message
2. To step 6

**ID:** UC-DA-10

**Use Case:** Create Coupon

**Functional Requirement:** FR-DA-05

**Actor:** Employee

**Prerequisite:** UC-DA-02

**Main Success Scenario:**

1. User chooses add coupon option
2. System shows all information to be filled in
3. User fills all information
4. User confirm add a coupon
5. System saves the new coupon

**Extensions:**

3a. User fills wrong information or doesn’t fill anything

1. System shows error message
2. Return to step 2

**ID:** UC-DA-11

**Use Case:** Put Boat on Maintenance

**Functional Requirement:** FR-DA-06

**Actor:** Employee

**Prerequisite:** UC-DA-02 and UC-DA-04

**Main Success Scenario:**

1. User chooses an existing boat
2. User chooses to edit boat
3. System shows all the information of the chosen boat
4. User chooses put boat on maintenance option
5. System saves the change and boat will be on maintenance

**Extensions:**

1a. User doesn’t choose a boat

1. System shows error message
2. To step 6

## Web Application Use Case

**ID:** UC-WA-01

**Use Case:** CustomerRegistration

**Functional Requirement:** FR-WA-02

**Actor:** Customer

**Prerequisite:** Web application is opened

**Main Success Scenario:**

1. User chooses login option
2. User chooses register option
3. System asks for username, email, and password
4. User fills in the credentials
5. User confirms the registration
6. System registers the new user and shows the login page

**Extensions:**

4a. User fills in wrong credentials or doesn’t fill credentials

1. System shows error message
2. System resets the fields
3. Return to step 3

**ID:** UC-WA-02

**Use Case:** Customer Login

**Functional Requirement:** FR-WA-01

**Actor:** Customer

**Prerequisite:** Web application is opened

**Main Success Scenario:**

1. User chooses login option
2. System asks for username and password
3. User fills the credentials
4. User confirm login
5. System shows the list of all items that can be rented

**Extensions:**

3a. User fills in wrong credentials or doesn’t fill anything

1. System shows error message
2. System resets the fields
3. Return to step 2

**ID:** UC-WA-03

**Use Case:** View Item

**Functional Requirement:** FR-WA-04

**Actor:** Customer

**Prerequisite:** UC-WA-02

**Main Success Scenario:**

1. User chooses view all item page
2. System shows the list of all items that can be rented

**ID:** UC-WA-04

**Use Case:** Add Reservation

**Functional Requirement:** FR-WA-03

**Actor:** Customer

**Prerequisite:** UC-WA-02

**Main Success Scenario:**

1. User chooses items to be added
2. User goes to cart page
3. System shows all items that user has chosen
4. User chooses to proceed
5. System shows all information to make a reservation
6. User fills up all information
7. User chooses to save
8. System adds new reservation
9. System shows all reservation page

**Extensions:**

3a. System doesn’t show any item

1. User chooses to proceed
2. System shows error message
3. System shows all items

6a. User puts wrong information or leaves it empty

1. System shows error message
2. Return to step 6

**ID:** UC-WA-05

**Use Case:** View Reservation

**Functional Requirement:** FR-WA-03

**Actor:** a

**Prerequisite:** UC-WA-02 and UC-WA-04

**Main Success Scenario:**

1. User chooses show all reservation option
2. System shows the list of all reservations that has been made

**ID:** UC-WA-06

**Use Case:** Edit Reservation

**Functional Requirement:** FR-WA-03

**Actor:** Customer

**Prerequisite:** UC-WA-02 and UC-WA-04

**Main Success Scenario:**

1. User chooses an existing reservation
2. User chooses to edit reservation
3. System shows all the information of the chosen reservation
4. User changes the information of the chosen reservation
5. User saves the changes
6. System shows the list of all reservation

**Extensions:**

1a. User doesn’t choose a reservation

1. System shows error message
2. To step 6

4a. User puts wrong information or leaves it empty

1. System shows error message
2. Return to step 3

5a. User doesn’t save the change

1. Reservation doesn’t change
2. To step 6

**ID:** UC-WA-07

**Use Case:** Remove/Cancel Reservation

**Functional Requirement:** FR-WA-03

**Actor:** a

**Prerequisite:** UC-WA-02 and UC-WA-04

**Main Success Scenario:**

1. User chooses an existing reservation
2. User chooses to edit reservation
3. System shows all the information of the chosen reservation
4. User chooses remove reservation option
5. System removes the selected reservation

**Extensions:**

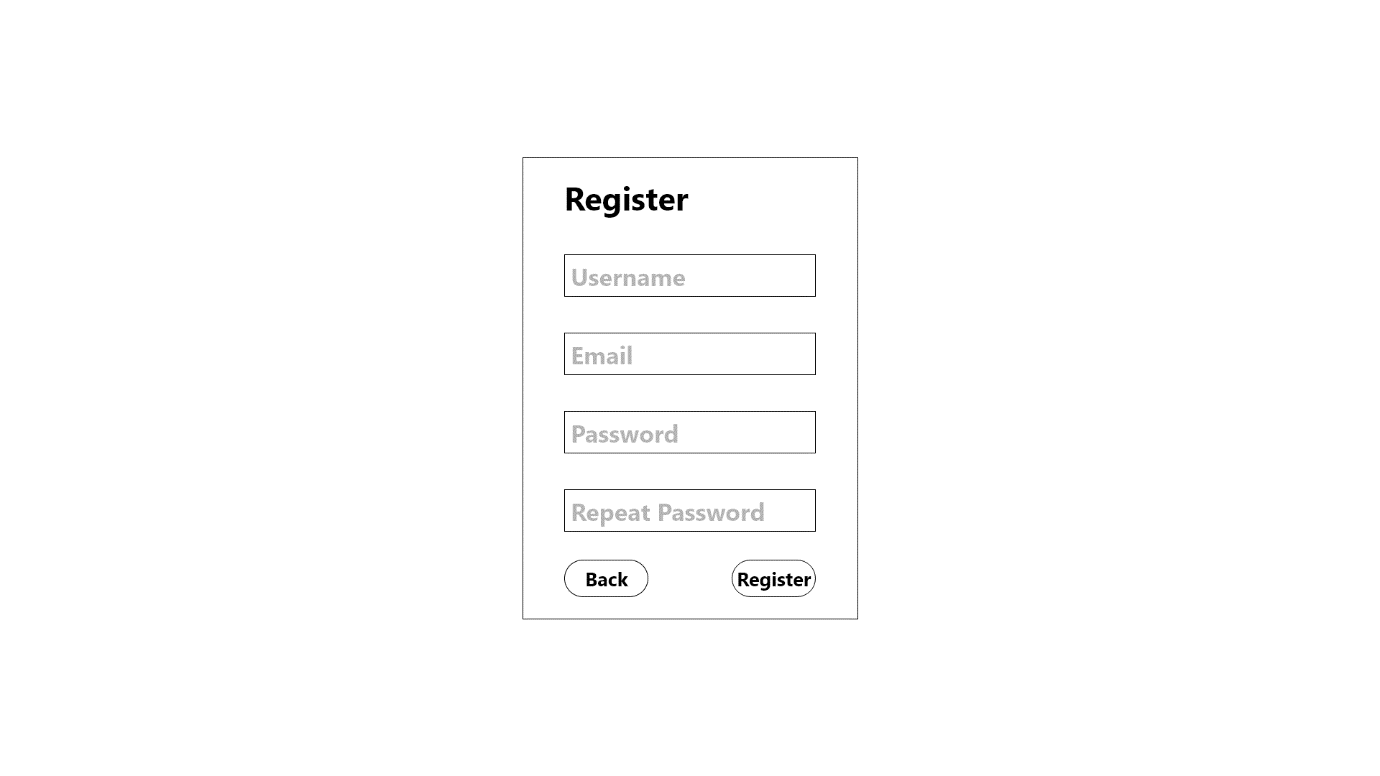
1a. User doesn’t choose a reservation

1. System shows error message
2. To step 6

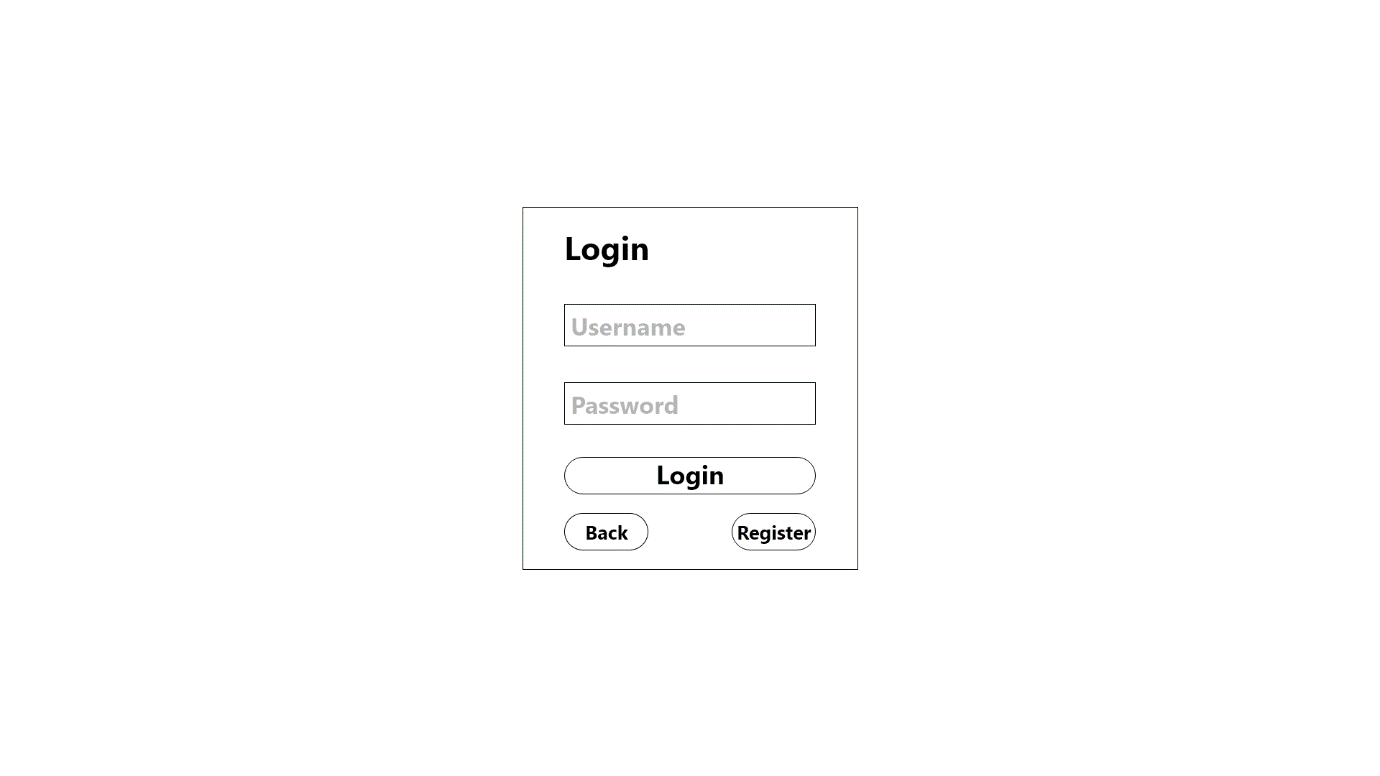
# **User Interface**

These is the wireframe of the design we will be implementing on both desktop application and web application.

For both desktop application and web application the login page and register will be the almost the same design.

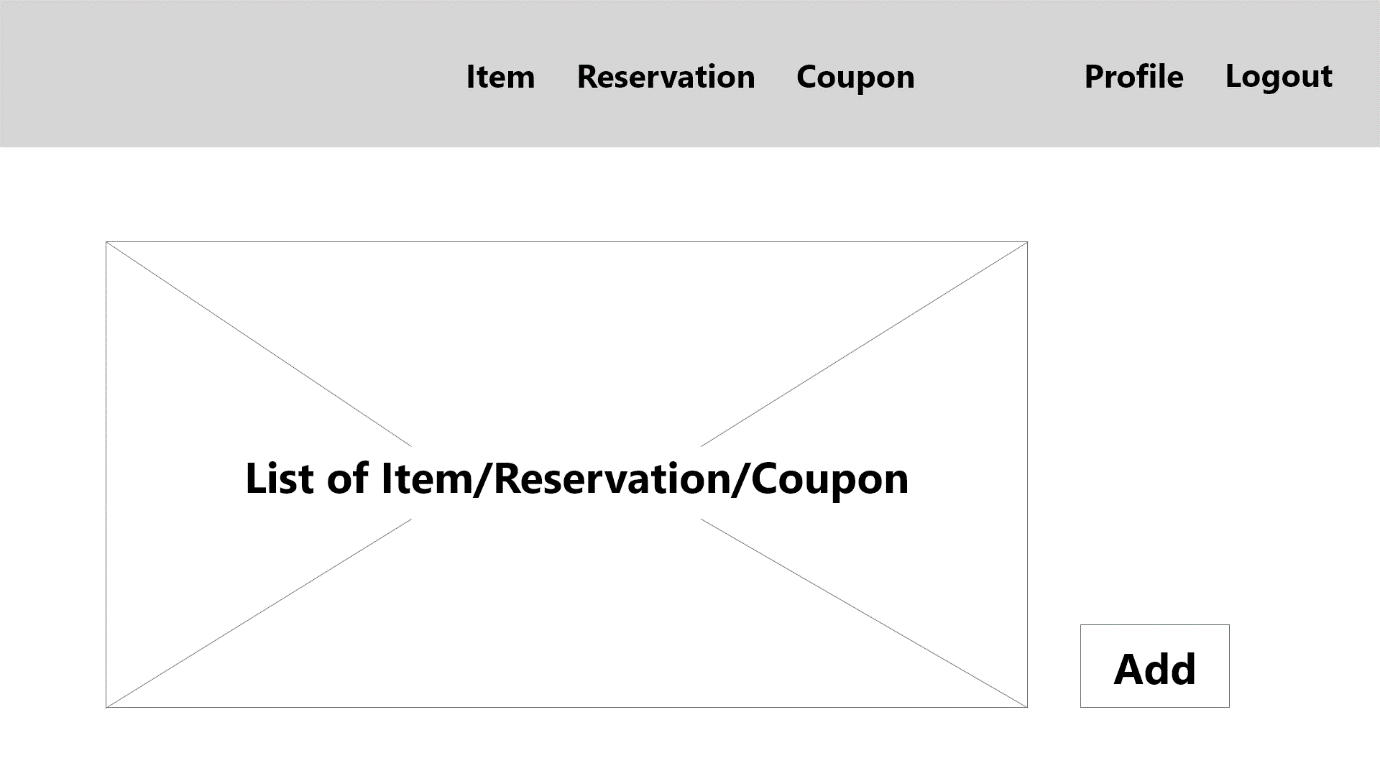


1 Login page for desktop application and web application.

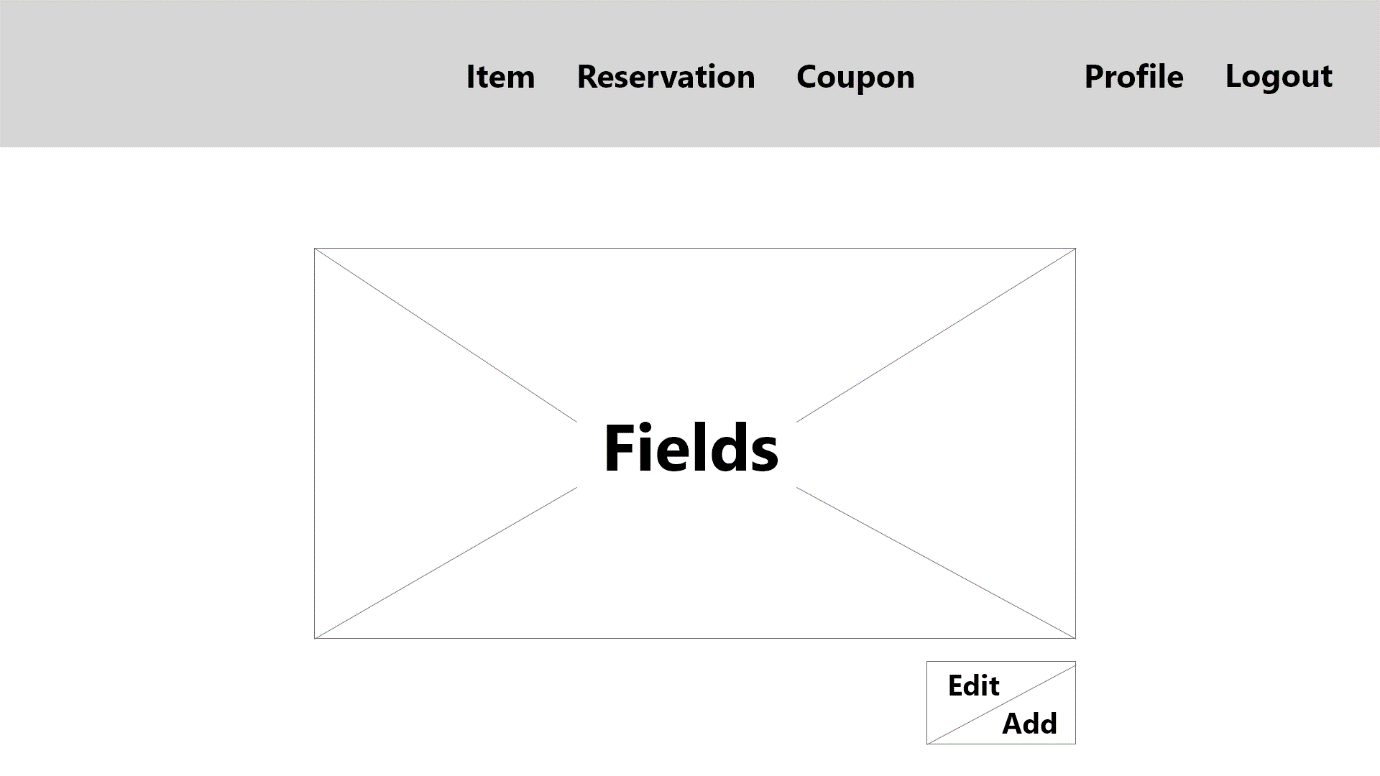


2 Register page for both desktop application and web application

Wireframes below, number 3 to 6, are for the desktop application.

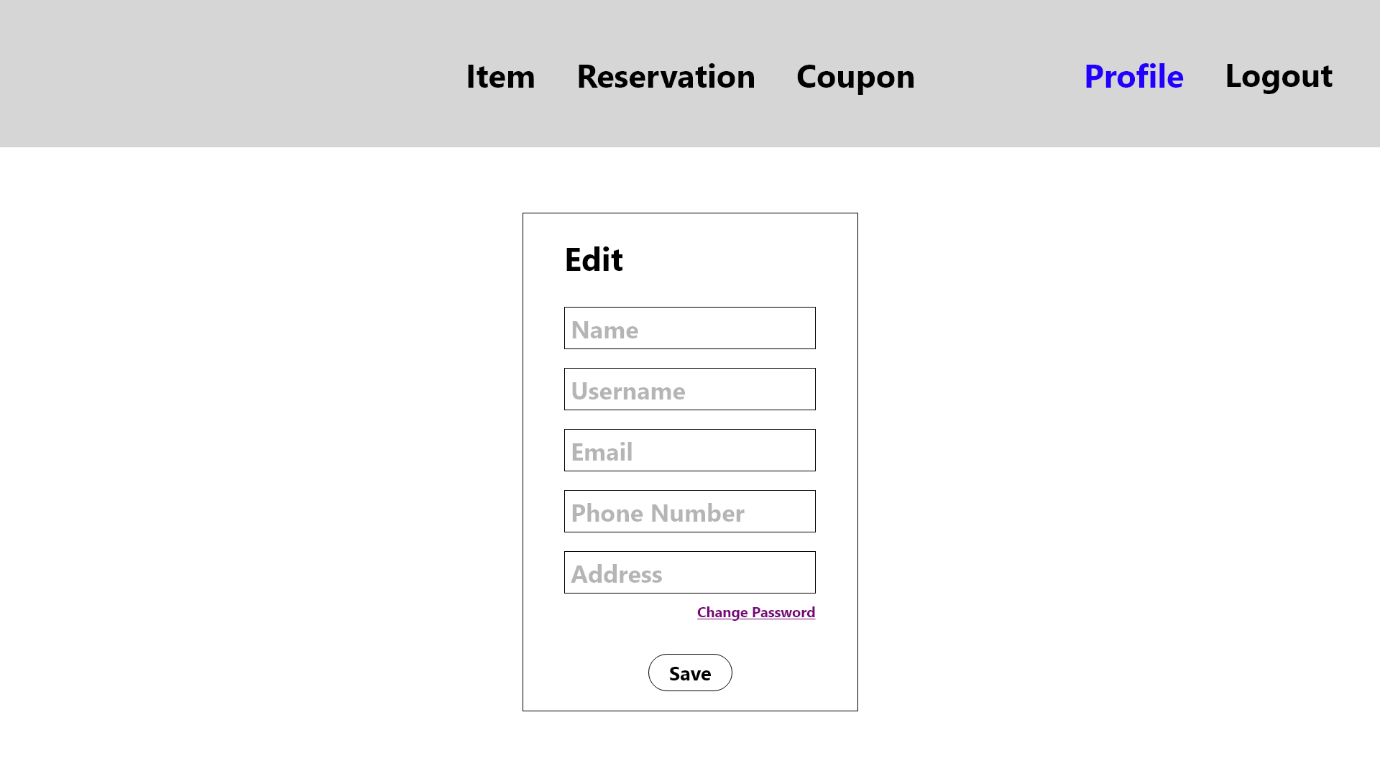


3 Home page of the desktop application

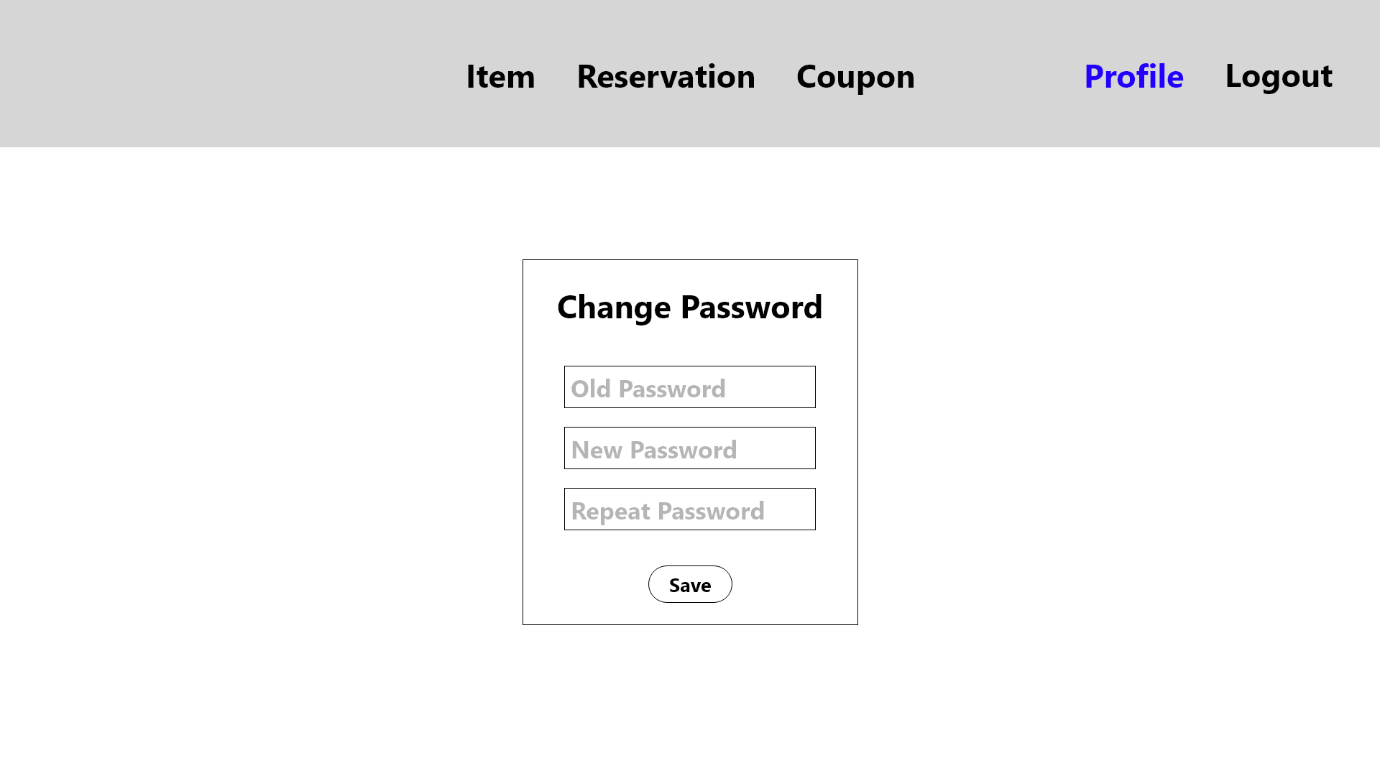


4 Edit/add page of the desktop application

Note that item, reservation, and coupon will have the same format as for displaying and editing it but it has different fields.

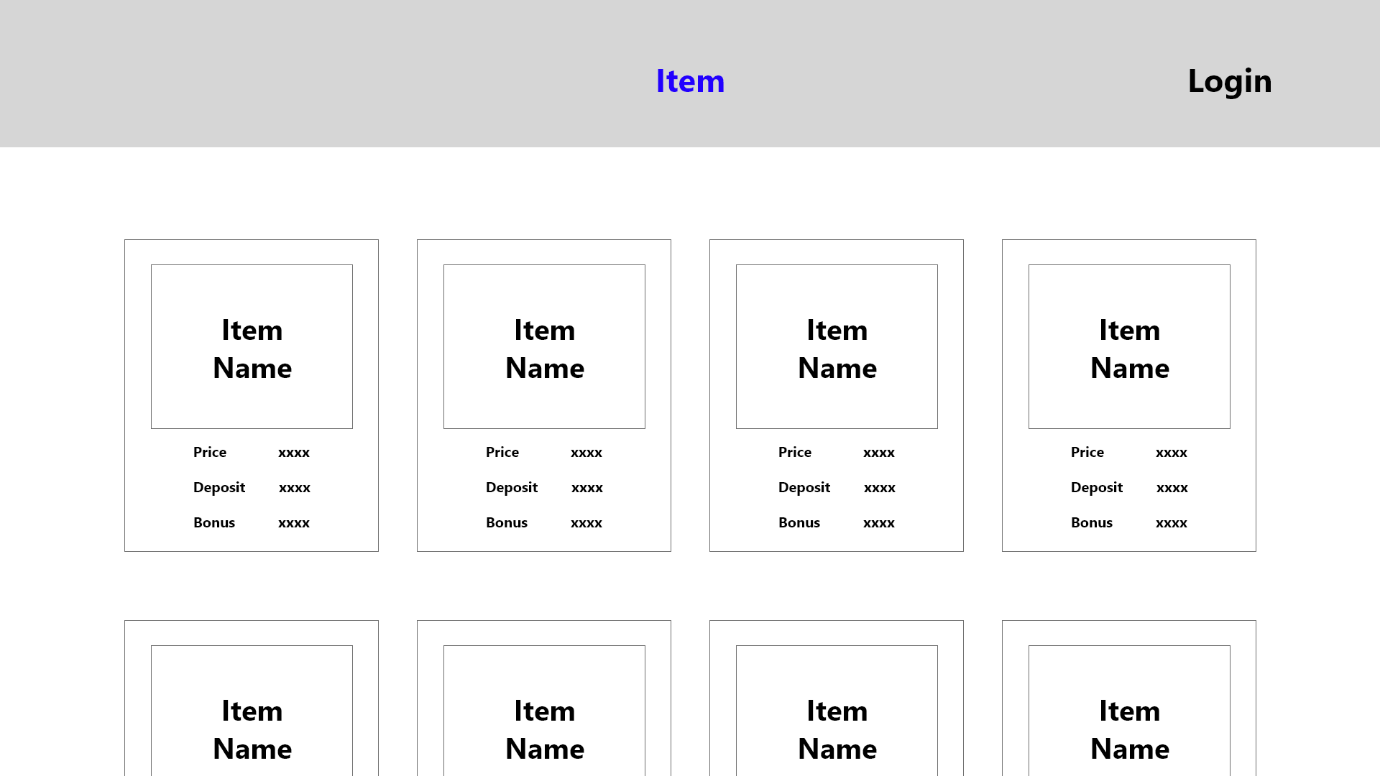


5 Edit profile page of desktop application

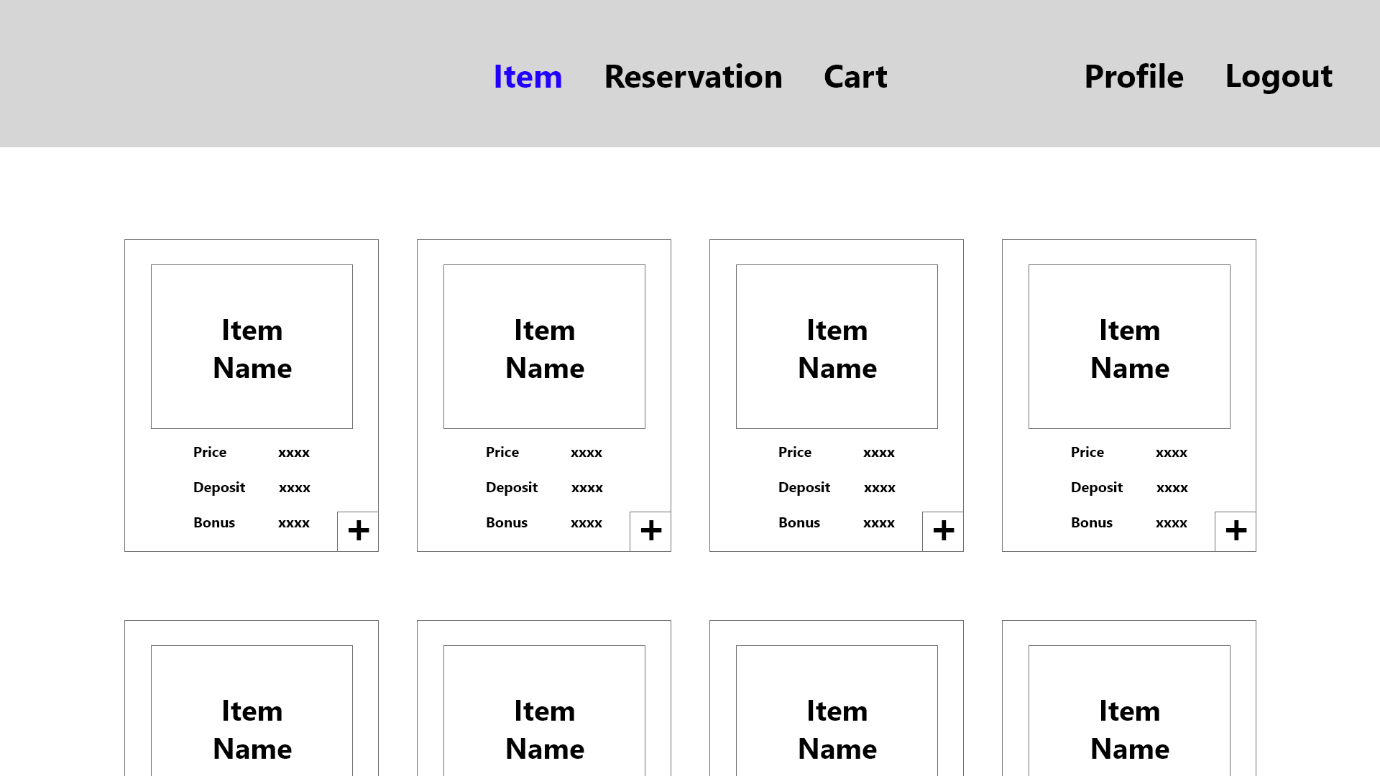


6 Change password page of desktop application

Below are the wireframes for the web application.

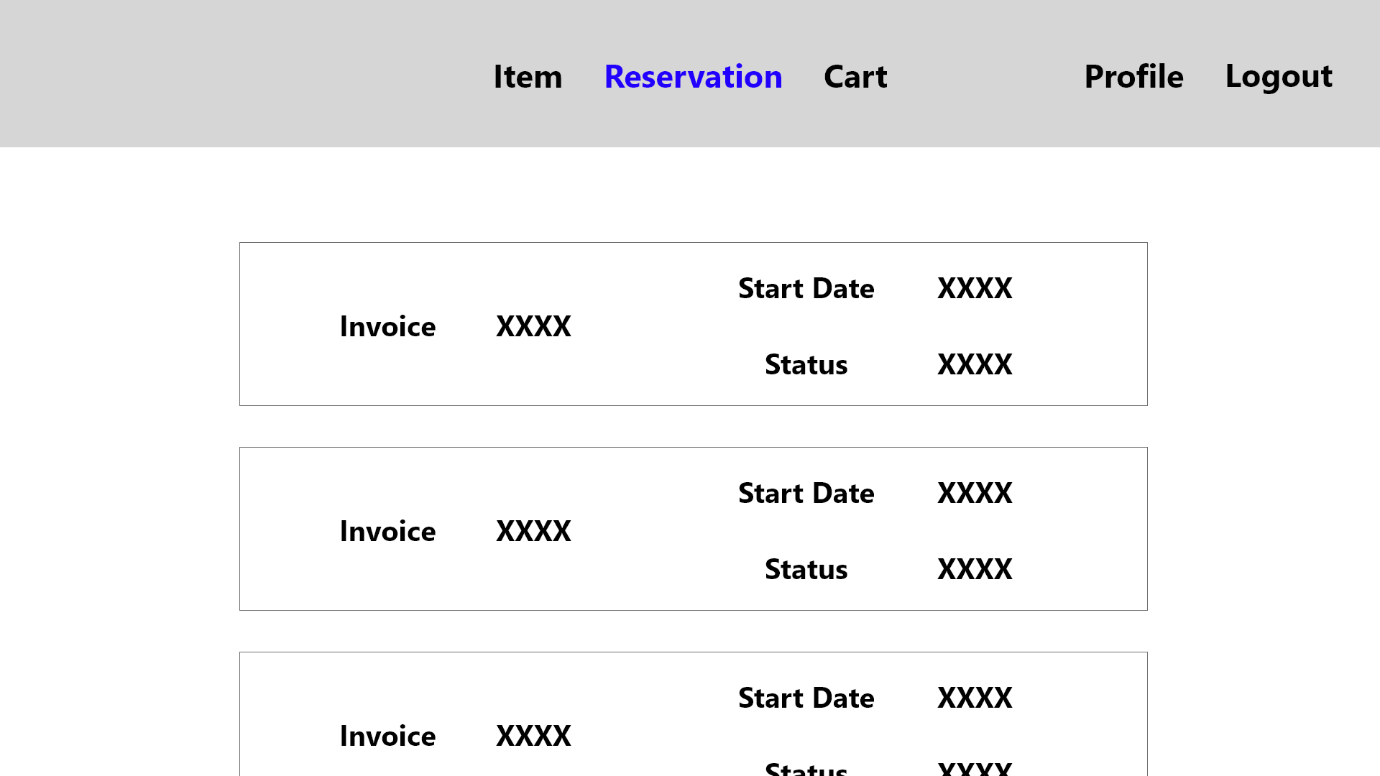


7 Home page before login of the web application

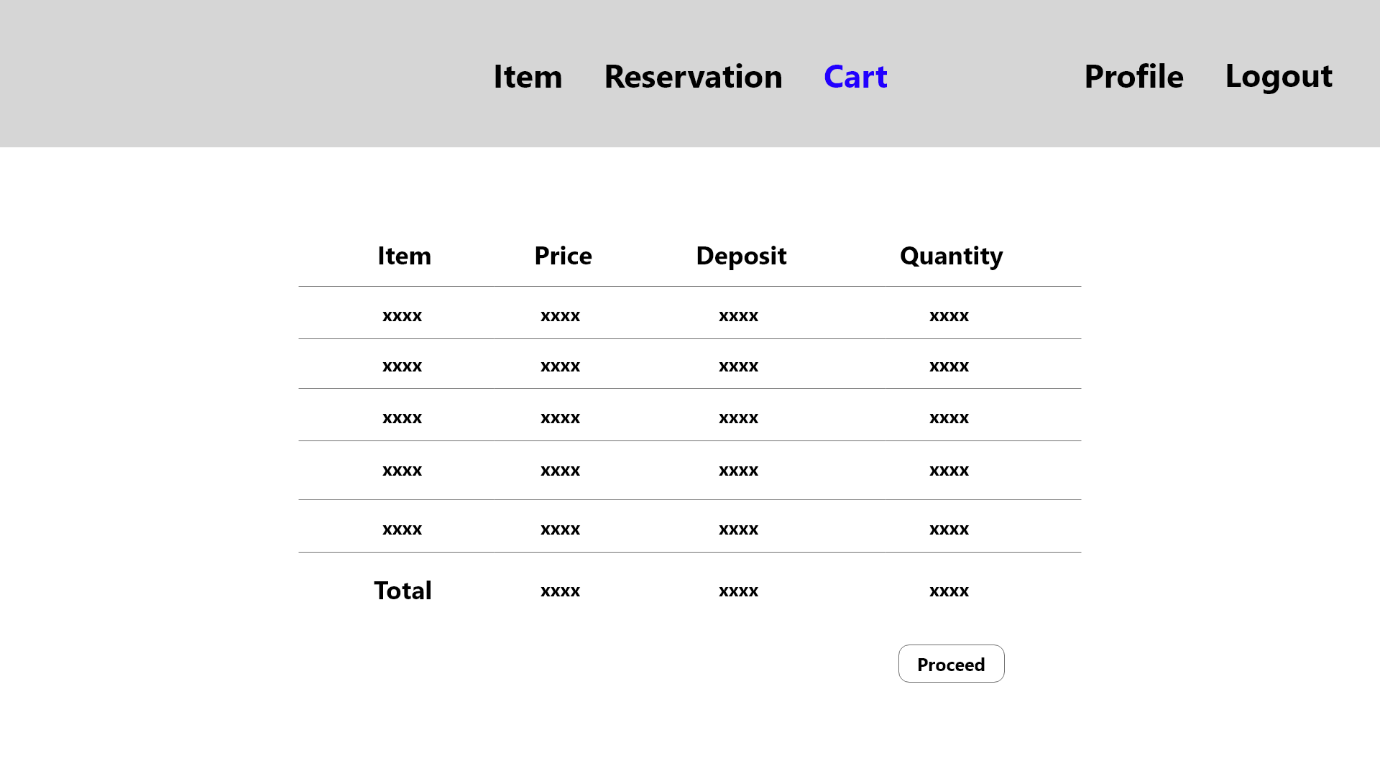


8 Home page after login of the web application

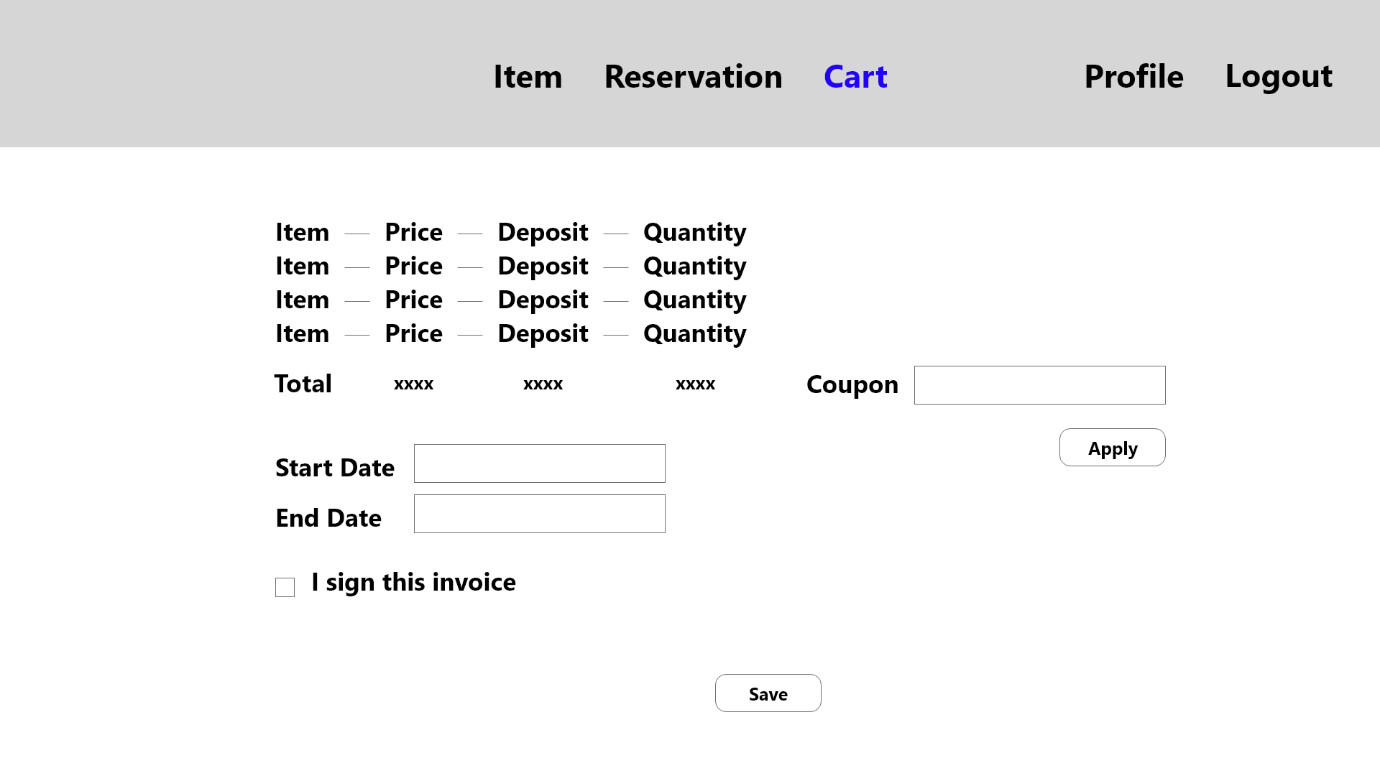
The different between before after login page beside the work login, logout, and profile is the addition of plus sign for the user (Customer) to add that item to their cart.



9 Reservation page of the web application

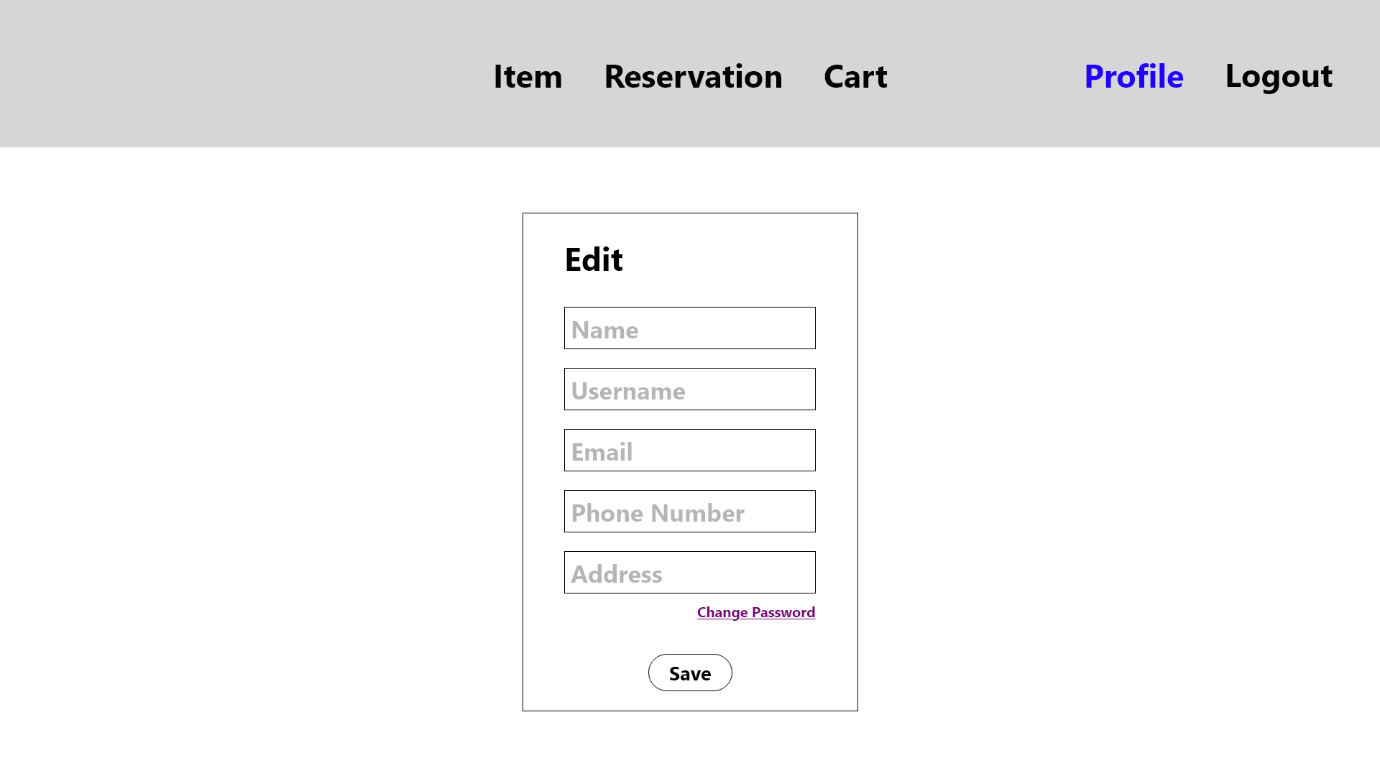


10 Cart page of the desktop application

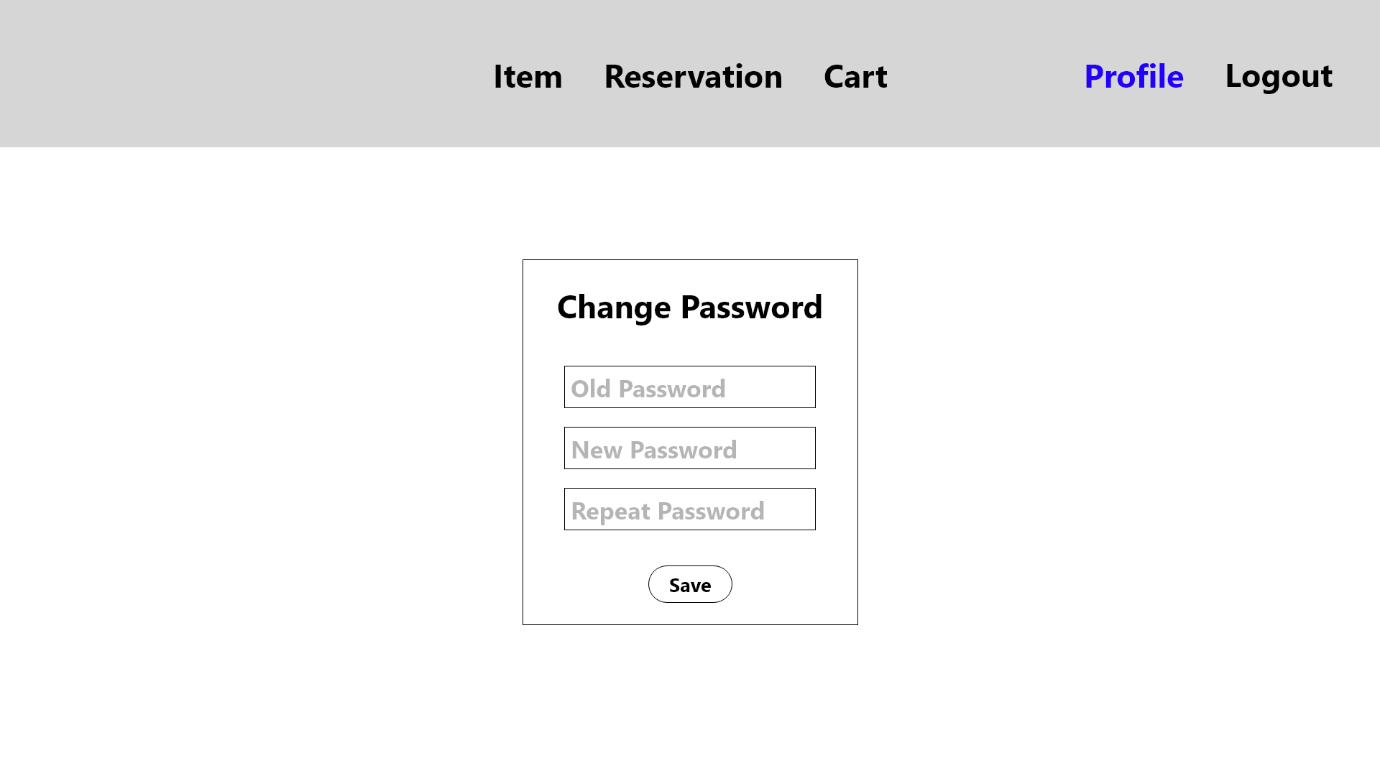


11 Cart after proceed page of the web application

This page, it’s the continuation from the cart page. User (Customer) can save the reservation for those items they have chosen. They have to put the start date and end date of the item they are going to use. They have to check the box to declare that they have sign the reservation and will pay for both deposit and the actual price. Customer can also either include the coupon to get a discount or leave it empty.



12 Profile page of the web application



13 Change password page of the web application

# **External Interface Requirements**

## User Interfaces

* Front-end Software: ASP.NET Core Razor Page
* Back-end Software: MySQL and C#
* Source Control: FHICT GitLab

## Hardware Interfaces

* Windows
* Browser that supports ASP.NET Core Razor Page
* Computer that supports C# language
* VPN connection to vdi.fhict.nl

## Software Interfaces

|  |  |
| --- | --- |
| **Software** | **Description** |
| Operating System | We used windows because for it’s the most common operating system for computer |
| Database | We use MySQL for it is one of the requirements of using it for this project |
| Desktop Application | We use C# for it is one of the requirements of using it for this project |
| Web Application | We use ASP.NET Core Razor Page for it is one of the requirements of using it for this project and its combability with C# |
| VPN | We must connect the VPN to vdo.fhict.nl to access their database to store information for this project |
| FHICT GitLab | For easier overview and progress of this project we use FHICT GitLab |

# **Non-functional Requirements**

## Safety Requirements

If there is a damage on either database, desktop application or web application, a copy of previous version is needed. Regular backup for database is required to maintain its information. Frequent Git push to a server is required for a minimum damage.

* 1. Security Requirements

Basic security such as password should be applied. Employee feature cannot be access by a customer. Database storage should be chosen carefully for maximum security to avoid a leak of information.

## Software Quality Attributes

* Extensibility:

Applications should be able to easily add new features.

* Maintainability:

Applications should be able to easily maintained and fix any problem.

* Securability:

Security measure on password in database.

* Bug Free:

Applications should not have any crash throughout the process.