My Little Shop Website

Version 1.0

May 10, 2018

Team H

Pham Ho Linh Thu (Team leader)

Pham The Vinh

Tra Ngoc Nguyen

Trinh Trung Dung

Lam Tuan Hung

Table of Contents

1.0. Introduction

1.1. Purpose

The purpose of this document is to present a detailed description of the My Little Shop Website. It will describe the requirements of the project, how we designed the project and what the website will do.

1.2. Scope of Project

This web application is designed for small businesses to manage their products, their reports and their branches. More specifically, this web allows a director to see sales reports and control inventories of each branch, and allows a staff in a shop to scan barcode and see information of a product.

2.0. Specific Requirements

2.1 External interfaces

- This program is distributed over the internet as a web application.
- Using mouse and keyboard for navigation and input method.
- Using the on-board webcam for capturing barcode.
- Using MySQL database software host by {insert host}.

2.1 Functional Requirements Specification

This section outlines the precondition, main scenario and exception scenario for each use cases of our project:

- Use case 1: Login

Use Case Name	Login
Trigger	The user enters username and password and press login
Precondition	The user has a validated account.
Main scenario	System look up the username, hashed of the password then direct the user to pages correct to their role.
Exception scenario	If the look up fail, the program will alert the user.

- Use case 2: Logout

Use Case Name	Logout
Trigger	The user clicks the logout button
Precondition	The user has a validated account and already logged in
Main scenario	System remove the user from the session and return to login page
Exception scenario	none

- Use case 3: Change password

Use Case Name	Change password
Trigger	The user presses the change password button
Precondition	The user has a validated account and already logged in
Main scenario	- System load the change password page - The user is required to enter old password and the new password

	- The system hashes the new password and save it (the hash) to database
Exception scenario	none

- Use case 4: Add shops/items

Use Case Name	Add shops/items
Trigger	The user presses the add item button
Precondition	The user has a validated account, already logged in and have the role of manager
Main scenario	- The system loads the add shop/item form - The user is required to fill the form - The system adds the information of the new shop/item to database
Exception scenario	- The system detects the barcode of the product already exist in the database - The system alerts the user that the barcode already exists

- Use case 5: Delete shops/items

Use Case Name	Add shops/ items
Trigger	The user presses the delete shops/items button
Precondition	The user has a validated account, already logged in and have the role of manager
Main scenario	- The system loads the delete shop/item form - The user is required to fill the form with the barcode of the product - The system removes the information of the shop/item from database
Exception scenario	- The system detects the barcode of the product not exist in the database - The system alerts the user that the barcode doesn't exists

- Use case 6: Add an employee

Use Case Name	Add an employee
Trigger	The user presses the add employee button
Precondition	The user has a validated account, already logged in and have the role of manager

Main scenario	- The system loads the add employee form - The user is required to fill the form - The system adds the information of the new employee to database
Exception scenario	- The system detects the ID number already exist in the database - The system alerts the user that the ID number already exist

- Use case 7: Remove an employee

Use Case Name	Remove an employee
Trigger	The user presses the remove employee button
Precondition	The user has a validated account, already logged in and have the role of manager
Main scenario	- The system loads the remove employee form - The user is required to fill the form - The system disables the account in the database
Exception scenario	- The system detects the ID number doesn't exist in the database - The system alerts the user that the ID number doesn’t exist

- Use case 8: Assign an employee to a shop

Use Case Name	Assign employees to a shop
Trigger	The user presses the edit employee button
Precondition	The user has a validated account, already logged in and have the role of manager
Main scenario	- The system loads the employee information - The user changes the shop ID - The system modifies the information to database
Exception scenario	none

- Use case 9: Scan barcode

Use Case Name	Scan barcode
Trigger	The user shows the barcode to the active webcam
Precondition	The user has a validated account, already logged in and have the role of seller.

Main scenario	- The system deciphers the barcode to a string - The system lookup the barcode in the database - If the barcode is valid, load the item information to the client
Exception scenario	none

- Use case 10: See items/reports list

Use Case Name	See items/reports list
Trigger	The user clicks the show item/report information button
Precondition	The user has a validated account, already logged in
Main scenario	- If the user is a manager. System lookup all item in the database and loads it to the client. - If the user is a seller. System lookup all item available to the respective shop and loads it to the client.
Exception scenario	none

- Use case 11: Save transactions

Use Case Name	Save transactions
Trigger	The user clicks the save transaction button
Precondition	The user has a validated account, already logged in and have the role of seller. There are items in the shopping cart
Main scenario	- The system check for the available quantities of the items - If there are enough items in the storage, the system make changes in the database.
Exception scenario	If there are not enough items in the storage, the system does not make changes in database and alert user.

2.3 Performance Requirements

The requirements in this section provide a detailed specification of the user interaction with the software:

- Listing feature: all required information should be prominent and easy to find for the user.
- Usage of the result in the list view: The results displayed in the list view should be user friendly and easy to understand. Selecting an element in the result list should only take one click.

- Barcode scanning efficiency: the result of scanning barcode should happen continuously with 100% success rate.
- Transaction commitment: the commitment of transaction should only take one click in the case of success.
- System dependability: The system should alert the user if any invalid information are entered or something wrong happen with the functionality of the service.

2.4 Design constraints

This section includes the design constraints on the software caused by the hardware:

- Internet connection: this software is a web-app that required internet access to run.
- The software is only tested on the Google Chrome browser.
- The software required a webcam to work.

3.0. Design and Architecture

3.1 Technology specification

Because of the limited time requirement of the project (18 days) and the inexperience of all the team members in building web application, most of the choices we made on technology focus on the fast and simple implementation aspect.

These are the main technologies and libraries that we have used for the project:

- Spring Boot: Back-end framework, written in java. Since all of the team members have been familiar with Java for a time, we decided to use Java for our project. Among all of the Java options for web development, Spring Boot stands out to be the most suitable one for us because it offers faster configuration and helps us build the application faster.

- Spring Security: supports security and authentication for the website. We chose this because Spring security is a robust security frameworks that works well with Spring Boot and offers many features like password encoding, role base authentication, token base authentication.

- Spring Data JPA (Hibernate): access and store data on database. We chose this, again, because of its simplicity and it works well with Spring Boot

- MySQL: database language

- QuaggaJS: fast and simple barcode scanner

- Datatables.js: sort and filter tables, with clean interface

- Bootstrap: frontend library. Since we don’t want to spend too much time on the user interface, bootstrap provides us the clean interface with little effort

- GitHub: version control. We chose GitHub because of its popularity but we didn’t manage to use it right.

3.2 Use case diagram

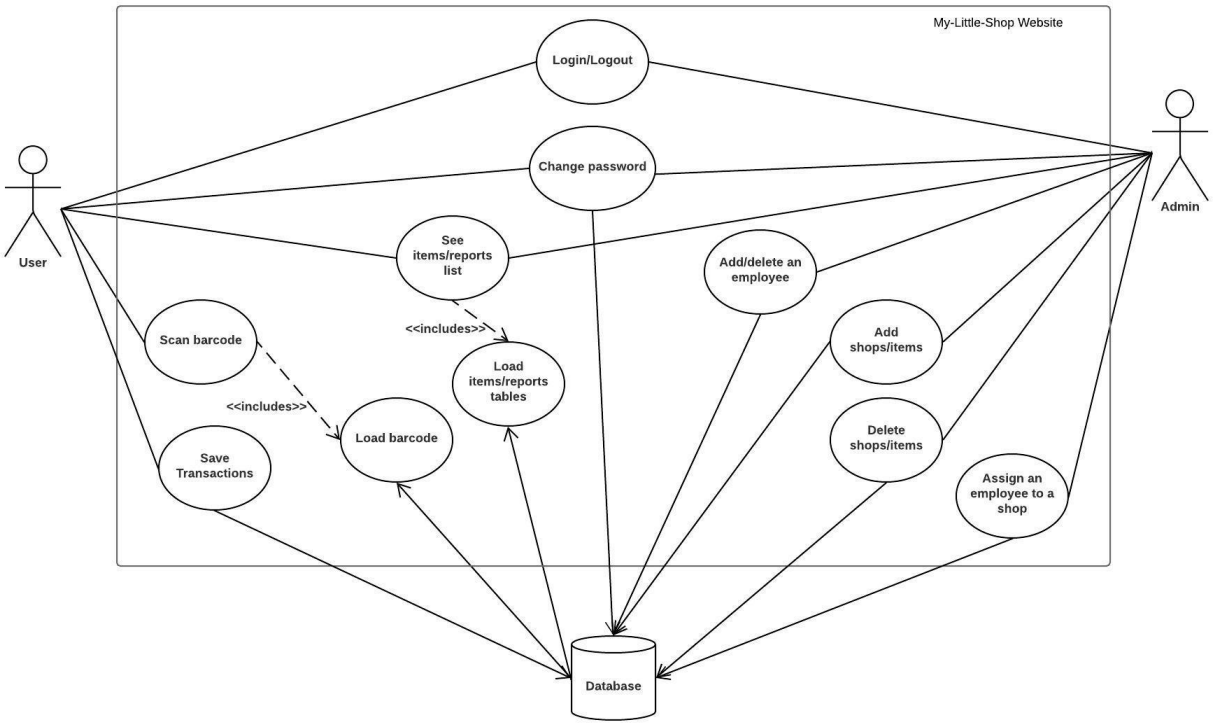

Figure 1 – Use case diagram

3.3 Class diagram

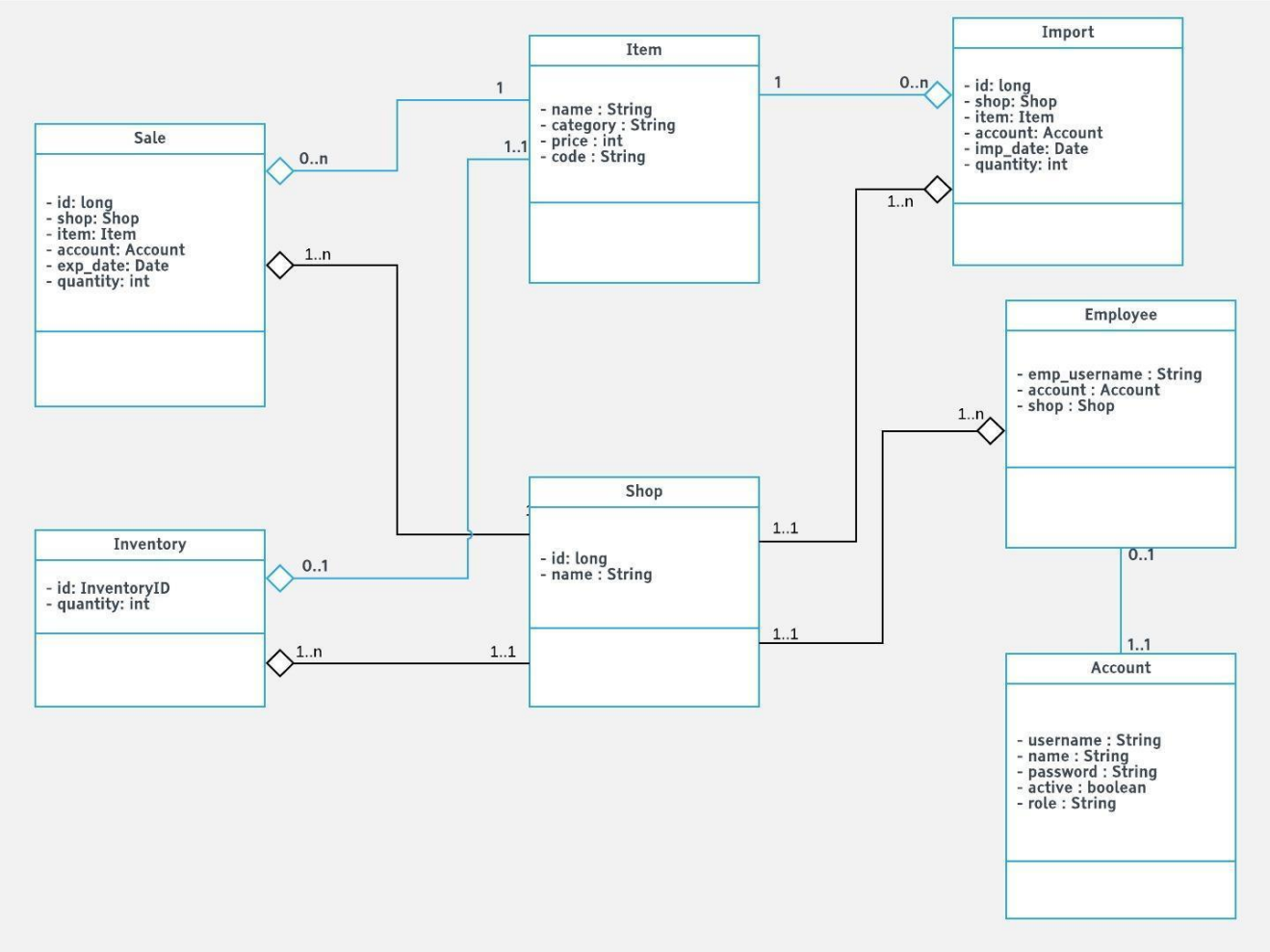

Figure 2 – Class diagram

3.4 Activity diagram

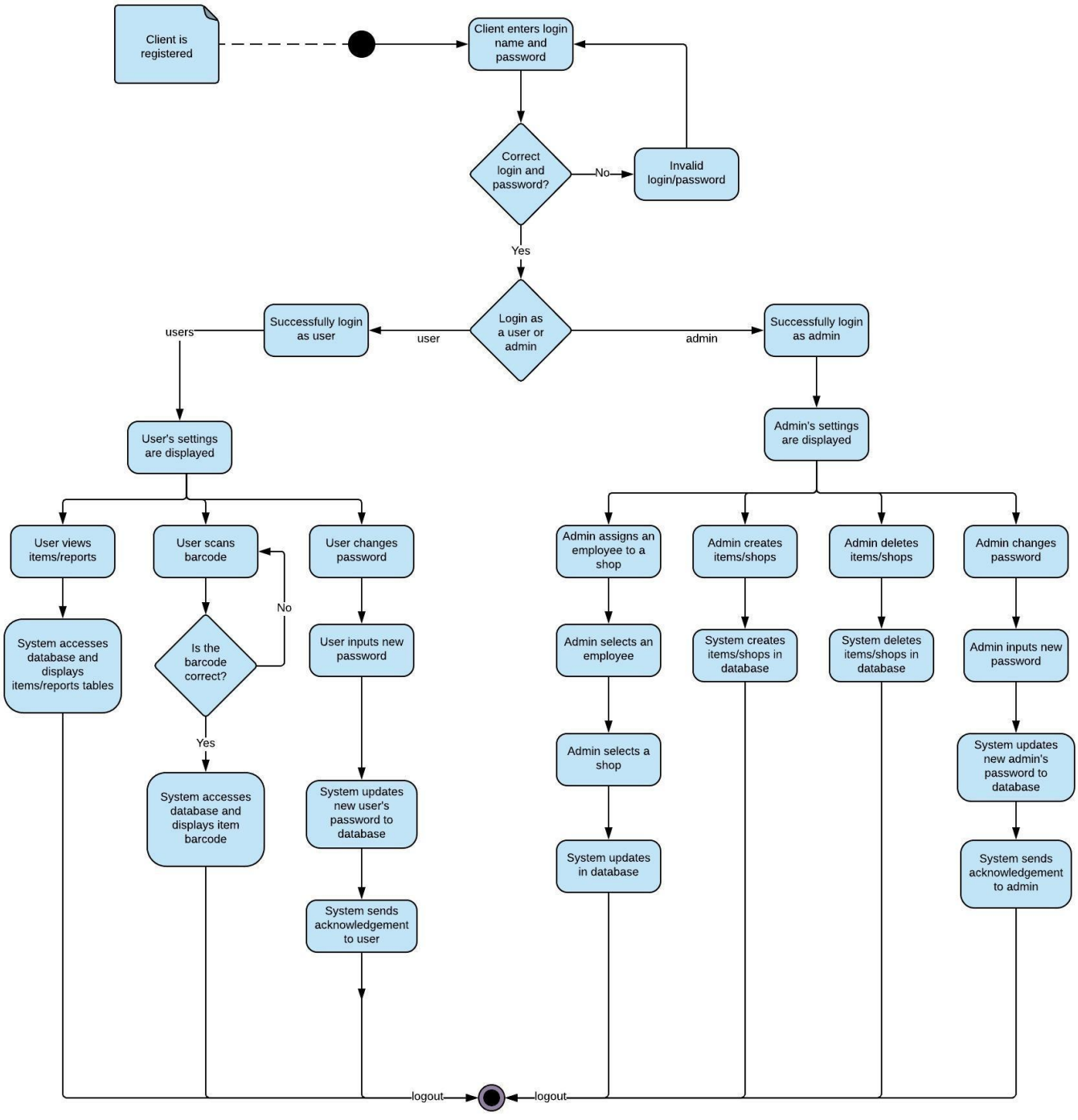

Figure 3 – Activity diagram

3.5 Deployment diagram

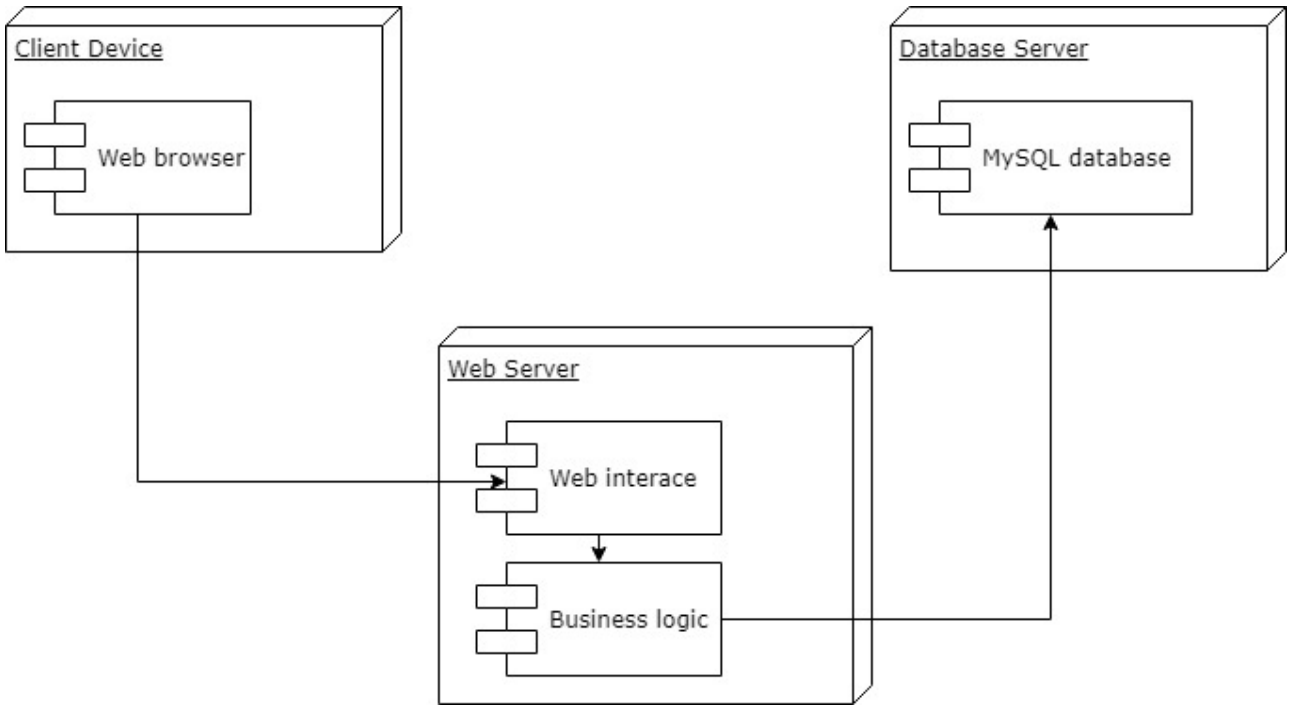

Figure 4 – Deployment diagram

4.0. Project management

4.1 Tasks management

We assigned tasks for each member to deliver the final product:

- Trinh Trung Dung: Designs the database, code the Entity classes and connect database to server. Write the Risk Management part of this document.

- Pham Ho Linh Thu: codes the html and JavaScript for the admin view, draw class diagram and use case diagram for this document, integrate documents of other members.

- Lam Quoc Hung: codes the html and JavaScript for the user view, draw activity diagram and deployment diagram for this document.

- Tra Ngoc Nguyen: Implements barcode scanner, security and authentication of users. Write the requirement specification of this document.

- Pham The Vinh: codes the Repository and Controller classes, integrate codes of other members into final product. Write the tasks management, technology specification and installation guide for this document.

4.2 Version control management

For the first half of the project, we used GitHub for version control and storing our codes. However, because of none of the team members have experience in using GitHub, it caused a lot of problems for us to push codes to the server. Since then, we had to have one member (Vinh) to integrate codes from all of the members into the final product instead of using GitHub.

4.3 Risk management

Risk	Probability of occurrence	Impact	Mitigation plan
Failure to meet performance requirements	High	High	- Use suitable tools - Train team member in performance engineering
Data exploitation	Medium	high	Keep software up-to-date

Distributed Denial of Service	Medium	High	Setting up network which is distributed, hardened and secure.
Run out of budget	Medium	High	- Keep the team informed of the budget status - Seek additional funding - Reduce the project's scope
Deadline is not met	High	High	Have a backup plan for project and extend the deadline to update the software.
Admin may disable himself	Low	High	Grant a new role for the owner whose account cannot be deleted or disabled.

4.4 Future development

It will be more complete if we can add this feature in the future:

- Price of a product can be changed in database or discounted in transactions: in real life, prices is not always fixed. For now, we couldn't add this feature because it will affect the database and we have not come up with an effective solution.