**Software Project Documentation**

MyLittleShop

Team:

1. Nguyen Hai Duc
2. Trần Minh Hoàng - 11112
3. Le Hoang Quan - 9307
4. Nguyen The Viet (Team leader) – 9990

**Prepared by:**

Nguyen The Viet

**Table of contents**

1. **Introduction**
   1. **Overview**

MyLittleShop is a desktop application available for PC users. In our project, we assume that each shop uses a laptop and has internet connection to use the application and there are 3 available shops. The application provides access to the available web services (server side) via **Simple Object Access Protocol (SOAP).** This app will allow employee (client) to perform barcode scanning, get the data of the products in the database for the check-out process, and manager (client) to get the view of the database of all shops (including the view of each shop) in order to perform re-stock process and manage the financial lists.

This document provides information of the requirements of the project, the architecture of the application, the schedule (tasks that each member has performed). Additionally, project constraints (risks and changes) are also stated as the last part of the documentation.

**1.2. Scope**

The application provides a role-based system which allows user to log in and access to different functions:

1. As Client: User is able to use the scanner to scan the barcode and retrieve the data of the products
2. As Manager: User is able to send request to retrieve sale report and data of all the shops (view of database). He/she can also send request to re-stock items of a particular shop if possible.
3. **Requirements**
   1. **Functional requirements**
      1. **Use case 1:**

Description: **Login System**

Actor: client (employee & manager)

Basic Path:

1. Client opens the application on the laptop desktop.
2. Client inputs data (username and password) and chooses a role (Employee/Manager).
3. System sends request to the server side to validate authentication.
4. System prompts success, client can access to the functions based on his/her role (Employee and Manager).
5. Client clicks “Logout”
6. Returns to login page.

Alternate Path:

4. System prompts failure, client re-enters the username and password again  
 5. Back to step 3 of the basic path.

**2.2.2. Use case 2:**

Description: **Barcode Scanner**

Actor: Client (Employee)

Basic Path: (assume user has successfully logged in)

1. Employee clicks “Scan” to start the process.
2. Camera stream appears, the barcode is scanned through the stream (128-Barcode).
3. Once a barcode is scanned, the data will be stored and appear on the GUI, the camera stream will stop.
4. Employee can go back to step 1 and perform scan again, otherwise go to step 5.
5. Employee clicks “Submit”.
6. The server receives the request, returns the check-out list and updates database.
7. Employee clicks “Logout”.
8. Returns to login page.

**2.2.3. Use case 3:**

Description: **Barcode Generator**

Actor: Client (Manager)

Basic Path: (assume user has successfully logged in)

1. Manager inputs code (string) to be generated
2. Manager clicks “Generate Barcode”
3. An image of barcode (.jpg) is generated
4. Manager clicks “Logout”
5. Returns to login page

**2.2.4. Use case 4:**

Description: **Manager View**

Actor: Client (Manager)

Basic Path:

1. User successfully logs in as Manager.
2. Server returns the view of all shop
3. Returns to login page.

**2.2.5. Use case 5:**

Description: **Restock Item(s)**

Actor: Client (Manager)

Basic Path:

1. User successfully logs in as Manager
2. Server returns the view of the product database of all shops.
3. Manager changes (restock) the quantity of a product in a particular shop by taking items from other shops if available.
4. Manager clicks “Submit”.
5. Server receives request, update the database and return message.
6. Manager clicks “Logout”
7. Returns to login page.

**2.2.6. Use case 6:**

Description: **Sale Report Generating**

Actor: Client (Manager)

Basic Path:

1. User successfully logs in as Manager
2. Server returns the view of the product database of all shops
3. Manager clicks “Sale Report”
4. Server returns list of products that have been checked out (name, price and timestamp)
5. Manager clicks “Logout”
6. Returns to login page.

**2.2 Non-functional requirements**

**2.2.1 Security**

* All data of the shops is stored in the database, which can only be accessed via back-end of the server side. Users (client side) only know about the available services on the server side and are able to send request to access the services.
* Authentication: application provides login system (role-based) for users to access to the services. Additionally, data of users (username and password) is stored in the database as **hash value** (SHA256)
* Messages sent between client and server should be encrypted for security purpose.

**2.2.2 Cost of server**

Demo of application will run on local server for cost-saving purpose. Additionally, hiring server and buying assisting services in the future should not cost no more than $5 each month.

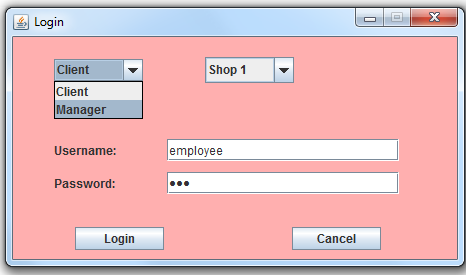
**3. Architecture**

MyLittleShop is a client – server application which is built on JavaEE with Dynamic Web Project and Axis2 Web services. Client is a desktop application and server is a web application that provides Axis2 web services. The back-end of the server connects to the database remotely by using Java and JBDC as connector to MySQL.

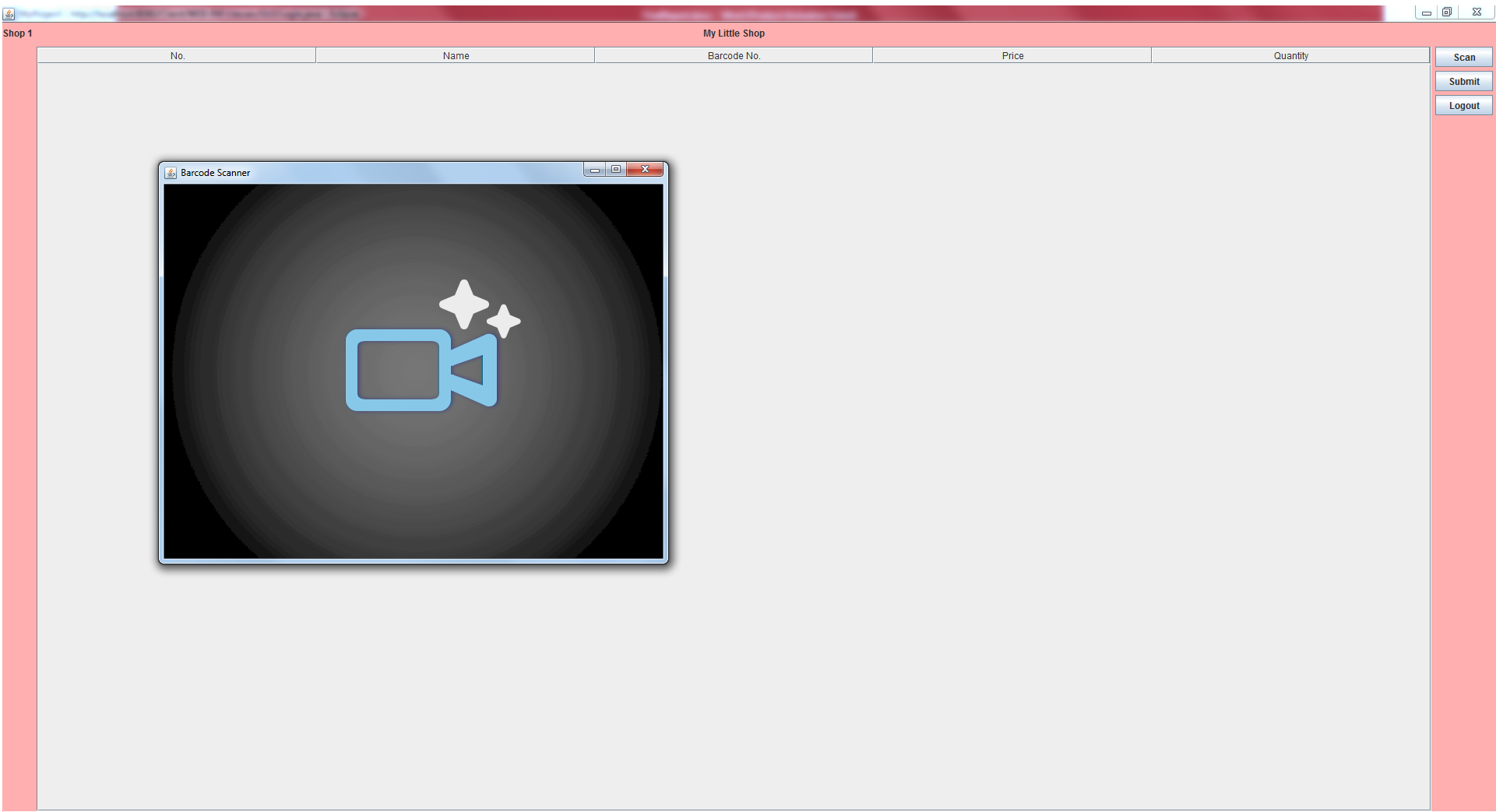
* Client is a desktop application which is mainly coded in Java
* Server side provides Axis2 web services for client side which are run on Tomcat server 7.0+.
* Back-end of server is coded in Java, which handles client requests as parameters and connects to Database (MySQL) remotely by JDBC.
* Application makes use of the portable web application phpMyAdmin that provides administrations tool for MySQL (port 3306) that runs on the local server (Apache server port 80).

**Client:** A desktop application:

1. Graphical User Interface is created mainly with Java Swing Library:
   1. Initial GUI for starting the application: Login system
      1. Role: Client/Manager
      2. Username
      3. Password
      4. Shop: chosen shop

**Figure 3.1 Login interface (Java Swing)**

* 1. GUI of Employee:
     1. Scan: opens camera stream to scan the barcode
     2. Submit: sends request to server side (retrieving product name, storing transactions)
     3. Logout: return the login page

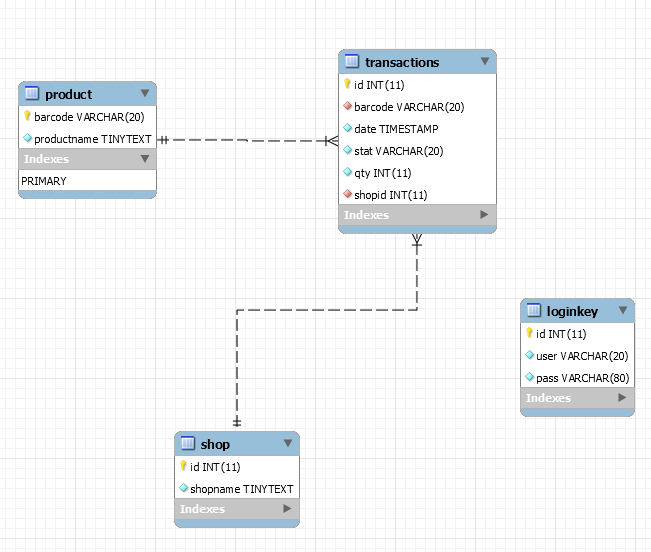


**Figure 3.2 Employee interface (Java Swing)**

* 1. GUI of Manager:
     1. W
     2. R

**Server:** Web application provides Axis2 web services (Back-end is written in Java):

1. Run on Tomcat server 7.0+
2. Communicates with client side via HTTP
3. Connects with Database (MySQL) via JDBC (queries)
4. Database: uses phpMyAdmin (run on Apache local web server) tool to run SQL queries. The database is also hosted in a local server which can be accessed from the server side.



**Figure 1.4 Basic database structure**

**5. Management Control**

**5.1 Version Control**

The project source code is uploaded weekly on Github (with support tool Git Bash to clone the local repository, pull, push, commit and merge branch, etc.). Since there were some unexpected errors that prevented some laptops from creating webs services axis2 that cost too much time to fix, we decided to upload different .rar files with different versions to github in order to keep track of the server errors.

Link to github: <https://github.com/KuroViet97/MyLittleShop/>

KuroViet97: Viet (master)

Hoang19101997: Hoang

nguyenhaiduc1994: Duc

quan: Quan  
  
The version control is a bit messy, and it is my fault not to control it well for this project. Specific tasks are transparently listed in 5.2 part.

**5.2 Tasks**

**5.2.1. Le Hoang Quan – 9307**

* + Design graphical user interface for client side (Java Swing)
  + Integrate client functions into GUI
  + Implement codes that handle all client requests to server, display data on GUI (client-side)
  + Handle queries for adding new shops and products on back-end server.

**5.2.2. Nguyen Hai Duc –**

* + Implement Barcode reader & generator in client side
  + Draw diagrams of the design architecture of the projects
  + Assist writing the report about **how axis2 web services in server communicate with client**

**5.2.3. Hoang**

* + Design the connection between client and server using Axis2 web services
  + Implement security parts:
    - Messages (XML) between client and server are exchanged via **HTTPS port 8443** instead of HTTP port 8080
    - Token-based login system (WS-security of Axis2)

**5.2.4 Nguyen The Viet – 9990**

* + Design project architecture
  + Create relational database (MySQL) and connect it to server (using JDBC)
  + Implement codes for hash function (sha256) in back-end server. (Login passwords are stored as hash value inside the database)
  + Implement codes for back-end server: getting queries for creating transaction in/out, viewing transaction (by different dates), comparing transaction (for adding stocks), getting data based on barcode
  + Write documentation of the project
  + Write installation instruction for the project

**5.3 Risk Management**

**5.3.1. Random people are chosen to form a group**:

* + Risk: People have different strength, experience and weakness.
  + Solution: face-to-face meeting discussion about the common design of the architecture, analyzing potential of each member and assigning appropriate task to each member. Regular communication to each other is highly concerned.

**5.3.2. Changes in requirements:**

* + Risk: Old functions may need to be changed to adapt new requirements, or possible additional functions can be added into the applcation