Benjamin Osemene, Shukrullah Zahiri, Andrew Ohwoka.

Keyin college  St. John’s, NL

Java document

Table of Contents

[1.0 User Documentation 2](#_Toc174392344)

[1.1 What is the application about? 2](#_Toc174392345)

[1.2 Explanation of the classes and their work 2](#_Toc174392346)

[1.3 Accessing the Application 2](#_Toc174392347)

[1.4 Class diagram relationship. 3](#_Toc174392348)

[2.0 Development Documentation 4](#_Toc174392349)

[2.1 Source code Directory Structure 4](#_Toc174392350)

[2.2 Build Process 4](#_Toc174392351)

[2.4 Development Standards 4](#_Toc174392352)

[3.0 Deployment Documentation 5](#_Toc174392353)

[3.1 Installation process 5](#_Toc174392354)

[3.2 Post-Deployment checks 5](#_Toc174392355)

# User Documentation

This part of the document describes the user document of the console-based E-Commerce platform using Java and PostgreSQL.

## 1.1 What is the application about?

The Java E-Commerce application is a console-based system designed to manage user roles (buyers, sellers, and admins) and product data efficiently. The application allows users to register, log in, and perform actions based on their role. The main features include user registration and authentication, role-based functionality (such as browsing, adding, and managing products), and product management. The intended users of the application are buyers who browse products, sellers who manage their product listings, and admins who oversee all users and products within the system.

## 1.2 Explanation of the classes and their work

The function of the console-based E-Commerce platform using Java and PostgreSQL.

* **User:** The User class serves as the base class for all users in the system. It includes common attributes such as; username ( unique identifier for the user), password (user's password, securely hashed using Bcrypt), email (user's email address).The role of the user (buyer, seller, or admin).This class also contains methods to get and set these attributes.
* **UserDAO:** The UserDAO class is responsible for managing all CRUD (Create, Read, Update, Delete) operations related to users in the PostgreSQL database. It interacts with the database to add new users, retrieve existing users, update user information, and delete users. This class ensures that all database interactions are performed securely and efficiently.
* **UserService:** UserService handles the business logic related to users. It provides methods for user registration, where passwords are hashed before being stored, and for user authentication, where credentials are verified during login. Additionally, it includes validation logic to ensure that user data (such as email format) is correct before processing.
* **Product, ProductDAO, ProductService:** Product class represents a product in the system, including attributes such as name, price, quantity, and sellerId. It provides methods to get and set these values. ProductDAO handles CRUD operations related to products in the database. It allows for adding new products, retrieving product details, updating product information, and deleting products from the system. ProductService manages the business logic related to products, including validating product details before they are stored or updated in the database.

## Accessing the Application

The application is accessed by:

* **Registration:** When the application starts, the user is prompted to either register or log in. To register, the user must provide a username, password, email, and role (buyer, seller, or admin). After registration, the user can log in to access the system.
* **Login:** Users log in by entering their username and password. Upon successful login, the system identifies the user's role and displays a specific menu with options based on their role.
* **Navigating the Menu:** Buyers can browse and search for products. Sellers can add, update, delete, and view their products. Admins can view and manage all users and products in the system.

## 1.4 Class diagram relationship.

* **Inheritance**: Buyer, Seller, and Admin classes inherit from the User class.
* **Associations**: There are associations between User and Product classes, showing how users (sellers) manage products, and how users (buyers) interact with products.

# 2.0 Development Documentation

The Javadoc are essentials for providing clear and structured documentation for the java classes and methods. All classes, methods and attributes documented using Javadoc. This can be seen in the java codes.

## 2.1 Source code Directory Structure

The source code directory structure for the java project is shown below.

* src/: Contains all the Java source files (.java).
  + main/java/com/ecommerce
    - AuthenticationService.java
    - Main.java
    - Menu.java
    - Product.java
    - ProductDAO.java
    - User.java
    - UserDAO.java
    - UserRole.java
    - UserService.java
  + build/
  + lib/
  + docs/
  + REAME.md

## 2.2 Build Process

The project is built using **Maven**, a popular build automation tool for Java projects.

**2.3 Compiler Time Dependencies**

The project has the following dependencies:

* **JDK**: Ensure you have JDK 8 or later installed.
* **PostgreSQL JDBC Driver**: Required for database connectivity.
* **Bcrypt**: Used for password hashing and validation.

## 2.4 Development Standards

The project follows the following coding standards:

* **Coding Standards**: Follow Java naming conventions, use meaningful variable names, and ensure code readability.
* **Version Control**: Use Git for version control and maintain a clean commit history.
* **Testing**: Write unit tests for all classes and methods using a testing framework like JUnit.
* Use Javadoc comments for documentation.
* **Exception Handling**: Custom exceptions are used to handle specific scenarios, with meaningful error messages for easy debugging.

# 3.0 Deployment Documentation

This document serves as the deployment of the develop a console-based E-Commerce platform using Java and PostgreSQL system application.

## Installation process

To deploy the application in a production environment:

* Set up the environment by installing Java, PostgreSQL, and any other required software.
* Configure the database by running the SQL scripts to create the necessary tables.
* Build the project using the build tool (Maven/Gradle).
* Deploy the packaged application (JAR file) to the production server.
* Update any configuration files (e.g., database connection details) as needed.

## Post-Deployment checks

After deploying the application, perform the following checks:

* **Database Connections**: Verify that the application can connect to the database and perform CRUD operations.
* **User Interface**: Ensure that the console-based UI is accessible and displays the correct menus based on the user's role.
* **Functionality**: Test all role-based functionalities (registration, login, product management) to ensure they work as expected.