**CS673 Software Engineering**

**Team1 - Terriers Mall**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Haron Dai | Team & Security Leader | *Haoran Dai* | Oct.19,2023 |
| Bhargav Gundapaneni | Implementation Leader | *Bhargav Gundapaneni* | Oct.19,2023 |
| Xiaocheng Kang | Requirement & configuration leader | *Xiaocheng Kang* | Oct.19,2023 |
| Huanzhou wang | Design Leader | *Huanzhou Wang* | Oct.19,2023 |
| Qinchen Gu | Q & A Leader | *Qinchen Gu* | Oct.19,2023 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **1.00** | **Bhargav Gundapaneni** | **2023-10-19** | **Intial release: User registration and login functionality** |
|  |  |  |  |

[Introduction](#_heading=h.gjdgxs)

[Software Architecture](#_heading=h.30j0zll)

[Class Diagram](#_heading=h.1fob9te)

[UI Design (if applicable)](#_heading=h.3znysh7)

[Database Design (if applicable)](#_heading=h.2et92p0)

[Security Design](#_heading=h.tyjcwt)

[Business Logic and/or Key Algorithms](#_heading=h.3dy6vkm)

[Design Patterns](#_heading=h.1t3h5sf)

[Any Additional Topics you would like to include.](#_heading=h.4d34og8)

[References](#_heading=h.2s8eyo1)

[Glossary](#_heading=h.17dp8vu)

# Introduction

In this document, we describe the design and architecture of our web-based user registration and login system developed using Spring Boot. This system provides users with the ability to securely register and log in to access protected resources.

A Software Design Document (SDD) serves as the foundational blueprint for software development. It offers a comprehensive overview that encapsulates the entire architecture, design paradigms, and foundational structures upon which the software is built and expanded. The intention behind such a document is to provide clarity, direction, and a roadmap that any stakeholder — be it developer, designer, tester, or manager.

For our project, "terriersMall

# Software Architecture

* Presentation Layer: Developed using Thymeleaf templates to create user-friendly web interfaces for registration and login.
* Business Logic Layer: Utilizes Spring Boot services and repositories to manage business logic, database operations, and validations.
* Data Layer: Leverages the JPA (Java Persistence API) and a relational database for persistent storage of user information.

**System Decomposition:**

The terriersMall software system is decomposed into the following main components:

1. **User Management Component**: This component handles user registration, login, and profile management.
2. **Product Management Component**: Allows for product additions, modifications, and deletions.
3. **Order Management Component**: Manages customer orders, including order placement, tracking, and history.

Each component interacts with the main database to fetch and store information.

**System Architecture Diagram**:

**Interface & Dependencies**:

* User Management is dependent on the authentication framework for login and signup processes.
* Product Management fetches product details from the database.
* Order Management requires access to both User and Product Management for user details and product information respectively.

**Framework Used**: For our system, we're using: Spring Boot

* **Backend**: Java
* **Database**: MySQL
* **Frontend**: React

# Class Diagram

* Class: **User**
  + Attributes: ID, FirstName, LastName, Username, Email, Password
  + Methods: register(), login(), updateProfile(), logout()

**Product Management**:

* Class: **Product**
  + Attributes: ProductID, Product Name, Price, Type, Stock, Pictures, Details
  + Methods: addProduct(), modifyProduct(), deleteProduct()

**Order Management**:

* Class: **Order**
  + Attributes: userID, orderID, totalAmount, receiver, shippingAddress, phone, paymentStatus, creationTime
  + Methods: placeOrder(), trackOrder(), getOrderHistory()

# UI Design (if applicable)

1. Login/Signup Page:

Fields: Username/Email, Password, Confirm Password (for registration)

Buttons: 'Login', 'Register'

Links: 'Forgot Password', 'Register here' (on login page), 'Already have an account? Login here' (on registration page)

Product Listing Page:

Display: List/Grid view of products with Product Name, Price, and Image

Filters: By Type, Price Range

Buttons: 'Add to Cart', 'View Details'

1. Order Page:

Display: List of products in cart, with totalAmount at the bottom

Fields: receiver, shippingAddress, phone

Buttons: 'Place Order', 'Continue Shopping'

# Database Design (if applicable)

# Use Cases and Fields

User Form “terriersMall”

Here are the use cases and fields for adding customer

* **ID:** This field stores a unique identifier for each customer. It is necessary for identifying the customer in the system.
* **FirstName:** This field stores the first name of the customer. It is necessary for identifying customers in the system.
* **LastName:** This field stores the last name of the customer. It is necessary for identifying customers in the system.
* **Username:** This field required for the User login
* **Email:** This field stores the email address of the customer. It is necessary for email login.
* **Password:** This field stores the password. It is necessary for login details.
* **Confirm Password:** This field stores the password for confirmation during registration.

**1.** **User Form Details :**

| **Field** | **What it Stores** | **Why it’s Needed** |
| --- | --- | --- |
| ID | This field stores a unique identifier for each customer. | This is necessary for tracking customer order history and identifying the customer in the system |
| FirstName | This field stores the first name of the customer. | This is necessary for identifying customers in the system. |
| LastName | This field stores the last name of the customer. | This is necessary for identifying customers in the system. |
| UserName | This is required for login Purpose | This is necessary for login into website |
| Email | This field stores the email address of the customer. | This is necessary for Email Login |
| Password | This Field Stores Password | This is necessary for login Details |
| CofirmPassword | This Field Stores Password and final one | This is necessary for login Details |

**Order Cases and Fields**

| **Field** | **What it Stores** | **Why it’s Needed** |
| --- | --- | --- |
| userID | Store the unique identifier for each user/customer | Important to find what did this customer ordered |
| orderID | Unique identifier for each item | Important to check the item numbers to see if we need to stop this item order or not (Available or not) |
| totalAmount | The total amount of items which stored in the cart | Important to let user/customer know how many items they added and the total price of the items. For admin, they can know how many items left |
| receiver | Receiver which the user/customer provided for the admin | Admin needs to know who will receive item(s) |
| shippingAddress | The address which provided by user/customer | Admin can know where to send item(s) |
| phone | Phone number which provided by user/customer | Admin can let drivers/senders know how to contact with user/customer |
| paymentStatus | The payment which user/customer provided | Admin will know if the transaction is approved or not |
| creationTime | What time did the user/customer checkout the items | Admin will know when this order places, and shipping them at provided time |

**Product Form Details :**

Product Form “terriersMall”

Here are the use cases and fields for adding customer

1. ProductID

* Description: The ProductID is a unique alphanumeric code or number assigned to each product in the inventory.
* Importance: It serves as the primary key for product identification, ensuring that every product has a distinct identifier. This prevents any confusion or overlap between products, especially in large databases.

2. Product Name

* Description: The Product Name is the title or label given to a product, often describing its main features or characteristics.
* Importance: It allows both customers and administrators to quickly identify and refer to a product. A clear and descriptive product name can also influence purchase decisions.

3. Price

* Description: The Price attribute indicates the cost at which a product is sold to customers, usually expressed in the local currency.
* Importance: Price is a critical factor in purchasing decisions. Displaying the price transparently ensures customers know the value of what they are buying and aids in budgeting.

4. Type

* Description: The Type attribute classifies products into various categories or groups based on shared characteristics or uses.
* Importance: By categorizing products, customers can easily navigate and filter through products, enhancing their shopping experience and ensuring they find what they're looking for more efficiently.

5. Stock

* Description: Stock represents the quantity of a particular product available for sale in the inventory.
* Importance: Keeping track of stock levels is crucial for managing inventory, preventing overselling, and informing customers about product availability.

6. Pictures

* Description: The Pictures attribute contains visual images showcasing different angles and features of the product.
* Importance: Visual representations give customers a clearer idea of the product's appearance, design, and features, leading to informed purchasing decisions.

7. Details

* Description: The Details attribute offers an in-depth description of the product, highlighting its features, specifications, benefits, and any other relevant information.
* Importance: Detailed product descriptions provide customers with comprehensive knowledge about the product, ensuring they are well-informed before making a purchase.

| **Field** | **Description** | **Purpose** |
| --- | --- | --- |
| ProductID | Stores a unique identifier for each product. | Essential for product management and ensuring each product can be distinctly identified within the system. |
| Product Name | Contains the name of the product. | Allows users and administrators to recognize and refer to the product by its name in the system. |
| Price | Holds the retail price of the product. | Provides customers with the cost of the product and is crucial for transaction processes. |
| Type | Indicates the category or type of the product (e.g., electronics, apparel, etc.). | Helps in categorizing products, making it easier for customers to search and filter products based on their preferences. |
| Stock | Reflects the current inventory level of the product. | Vital for sales operations. If stock levels reach zero, customers are informed that the product is out of stock, preventing overselling. |
| Pictures | Contains visual representations or images of the product. | Gives customers a visual idea of what they are purchasing, enhancing the shopping experience. |
| Details | Provides a detailed description of the product, including its features, specifications, and other relevant information. | Offers customers comprehensive information about the product, aiding in their purchase decision. |

# Security Design

* + Spring Security: Implements authentication and authorization.
  + BCrypt Password Encryption: Encrypts user passwords before storing them in the database to enhance security.
  + SQL Injection Prevention: Uses JPA repository methods, which are inherently protected against SQL injection.

# Business Logic and/or Key Algorithms

* + Password Encryption: BCrypt algorithm is used for hashing and verifying passwords.
  + Email Validation: Checks if an email is in the correct format and if it already exists in the database.

# Design Patterns

* **Singleton Pattern**:
  + **Usage**: Ensure that a class has only one instance, and provide a global point of access to this instance.
  + **Implementation**: Our database connection utilizes the Singleton pattern to ensure there's only one active database connection throughout the application's lifecycle.
* **Observer Pattern**:
  + **Usage**: Allow an object to publish changes to its state so other objects can react in response.
  + **Implementation**: When a product's stock level changes, other components (like the recommendation system or inventory management) get notified and act accordingly
* **MVC (Model-View-Controller):** Separates the application logic into three interconnected components.
* **Repository Pattern:** Abstracts database operations and gives a cleaner and more readable business laye

# Any Additional Topics you would like to include.

* Responsive Design: The UI adapts to different screen sizes to ensure a consistent user experience across devices.
* Error Handling: Custom error messages are provided to guide the user during registration and login.

# References

Spring Boot Documentation: [<https://spring.io/projects/s>pring-boot] (<https://spring.io/projects/spring-boot>)

Spring Security: <https://spring.io/projects/spring-security>

Thymeleaf Documentation: <https://www.thymeleaf.org/doc/tutorials/3.0/thymeleafspring.html>

# Glossary

* Spring Boot: An open-source Java-based framework used to create stand-alone, production-grade Spring-based applications.
* Thymeleaf: A modern server-side Java template engine for both web and standalone environments.
* JPA: Java Persistence API, a specification for accessing, persisting, and managing data between Java objects and a relational database.
* BCrypt: A password hashing algorithm designed to build a cryptographically secure hash of a user's password.
* Repository Pattern: A design pattern that mediates data from the data storing layer and the business logic/application layer.
* MVC: Model-View-Controller, a design pattern commonly used for developing user interfaces that divides the application into three interconnected components