Doompause Enterprise

Ryan Coon

CST-451 Capstone Project Final Architecture & Design

Grand Canyon University

Instructor: Professor Amr Elchouemi

Revision:

Date: March 17, 2024

**ABSTRACT**

The company, with its rich history in the trade of medieval weapons, is seeking to expand its reach to customers globally through an online platform. The company has been in business for hundreds of years, offering medieval weapons for both defensive and offensive combat. Over the years, it has built a reputation for quality and authenticity. However, with changing times and the rise of digital commerce, the company has found it increasingly difficult to reach customers.

The proposed web application will allow users to view items, read weapon descriptions, and see the prices of the items. It will also offer an online shopping cart where customers can add, remove, and edit the quantity of products. This digital transformation aims to make shopping more convenient for customers and expand the company’s customer base. By addressing these needs, the company hopes to continue its tradition of excellence while adapting to the modern marketplace.

|  |
| --- |
| History and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 3/3/2024 | Ryan Coon | Initial draft for review/discussion |
| 3/10/2024 | Ryan Coon | Project Requirements Document |
| 3/17/2024 | Ryan Coon | Final Architecture & Design |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

**Integrated Instructor Feedback into Project Documentation**

Yes  No

**TABLE OF CONTENTS**

Design Overview 4

Detailed High-Level Solution Design 5

Detailed Technical Design 6

Appendix A – Technical Issue and Risk Log 7

Appendix B – References 8

Appendix C – External Resources 9

Design Introduction

1. The purpose of this web application is to expand the company’s reach to customers globally, adapting to the changing shopping habits of customers in the digital age. The benefits include increased reach, improved customer experience, and efficient operations. The main objectives are to develop a user-friendly web application with an intuitive interface, ensure secure online transactions with Spring Boot Security, efficiently manage data with MySQL, and launch the web application within the specified timeline and budget.
2. Use the template to list the project deliverables that are to be included external to this Design Specification (Data Dictionary, API Design, etc.).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Deliverable Acceptance Log | | | | | |
| ID | Deliverable Description | Comments | Evaluator (internal or external as applicable) | Status | Date of Decision |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |

***NOTE: If necessary, you may add subsections to those listed in order to match the requirements in the assignment description. Do not remove any top level sections (those that are listed in the Table of Contents).***

Detailed High-Level Solution Design

1. Provide a detailed overview of how the proposed design fits into the overall solution.
   1. Create diagrams to logically and physically depict the system. This can be illustrated using UML Component, UML Deployment, and UML Activity diagrams or simply block diagrams done in a drawing tool such as Visio.
   2. This section should also include any solution configuration changes that will be required to develop and implement the proposed solution.
   3. Describe the approach and resources required to assure non-functional requirements (such as security and performance) will be met with this solution.
   4. The purpose of the detail solution architecture is to provide sufficient information for a developer to produce the system.
2. Provide a detailed inventory of hardware and software technologies that will be used in the solution:
   1. List any Frameworks or third party libraries that will be used.
   2. List any Proof of Concepts to be completed (POC) to ensure that the technologies and frameworks selected are the best fit for purpose, cost effective, and proper to solve the problem. This section should also be updated with the purpose/rationale for the POC and the results of the POC.

Use the templates below to list the Proof of Concepts, hardware, and technologies.

|  |  |  |
| --- | --- | --- |
| Proof of Concepts | |  |
| **Description** | **Rationale** | **Results** |
| 1.Create a simple API backend and a simple front end | To ensure a Springboot backend and a React front end will work together | A springboot backend will in fact work with a React front end, |
| 2 - |  |  |
| 3 - |  |  |
| 4 - |  |  |
| 5 - |  |  |

|  |
| --- |
| Hardware and Software Technologies |
| 1 - Integrated Development Environment (IDE): An IDE like Visual Studio Code, Sublime Text, or Atom for writing and managing code. |
| 2 - Version Control System: Git for version control to track changes. |
| 3 - Frontend Development: JavaScript, HTML, and CSS for creating the user interface. You might also use libraries or frameworks like React to simplify the development process. |
| 4 - Backend Development: Spring Boot and Spring Boot Security for server-side operations. |
| 5 - Database Management System: MySQL Workbench for managing application data. |
| 6 - |

**Logical Solution Design:**

A screenshot of a computer

Description automatically generated

**Physical Solution Design:**

A screenshot of a computer

Description automatically generated

Detailed Technical Design

**General Technical Approach:**

The project scope is to develop a web application for the company to sell its medieval weapons to online customers. This application will address the business need of reaching a wider customer base and adapting to the changing shopping habits of customers.

**Solution**

The web application will have the following features:

Product Display: The application will display a variety of medieval weapons. Each item will have a detailed description and price.

Shopping Cart: Customers will be able to add, remove, and edit the quantity of products in their shopping cart. This feature will provide customers with a flexible and convenient shopping experience.

User-Friendly Interface: The application will have an intuitive and easy-to-navigate interface, making it easy for customers to browse products and make purchases.

Secure Payment Gateway: The application will integrate a secure payment gateway for safe and secure transactions.

This solution aims to bring the company’s hundreds of years of business into the digital age, making its unique products easily accessible to customers around the globe. By doing so, it hopes to continue its tradition of excellence while adapting to modern market demands.

**Key Technical Design Decisions:**

Frontend (Client Tier): The frontend of the application will be built using JavaScript, HTML, and CSS. These technologies are standard for web development and will ensure a user-friendly interface and seamless user experience. The front end will be responsible for presenting data to users and handling user interactions.

Backend (Server Tier): The backend will be powered by Spring Boot and Spring Boot Security. Spring Boot simplifies the setup of standalone Spring applications, while Spring Boot Security will handle secure transactions. MySQL will be used for database management, ensuring efficient data storage and retrieval. The backend will handle business logic, data storage, security, and communication with the frontend.

**Database ER Diagram:**

A diagram of a company

Description automatically generated with medium confidence

**Database DDL Scripts:**

DROP TABLE IF EXISTS `user\_details`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `user\_details` (

`id` int NOT NULL AUTO\_INCREMENT,

`created\_on` datetime(6) DEFAULT NULL,

`email` varchar(255) DEFAULT NULL,

`enabled` bit(1) DEFAULT NULL,

`first\_name` varchar(255) DEFAULT NULL,

`last\_name` varchar(255) DEFAULT NULL,

`user\_key` varchar(255) DEFAULT NULL,

`phone\_number` varchar(255) DEFAULT NULL,

`updated\_on` datetime(6) DEFAULT NULL,

`user\_name` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `UK\_7fx13tfge5grrevwfx5qqwdxn` (`user\_name`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

DROP TABLE IF EXISTS `product`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `product` (

`id` int NOT NULL AUTO\_INCREMENT,

`brand` varchar(255) DEFAULT NULL,

`category` varchar(255) DEFAULT NULL,

`description` longtext,

`image` varchar(255) DEFAULT NULL,

`item\_title` varchar(255) DEFAULT NULL,

`price` double NOT NULL,

`seller\_id` int DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `FKesd6fy52tk7esoo2gcls4lfe3` (`seller\_id`),

CONSTRAINT `FKesd6fy52tk7esoo2gcls4lfe3` FOREIGN KEY (`seller\_id`) REFERENCES `seller` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

DROP TABLE IF EXISTS `cart\_items`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `cart\_items` (

`cartid` int NOT NULL AUTO\_INCREMENT,

`quantity` int NOT NULL,

`buyer\_id` int DEFAULT NULL,

`product\_id` int DEFAULT NULL,

PRIMARY KEY (`cartid`),

KEY `FKedryofpdu4slsh36ilpb5y5a4` (`buyer\_id`),

KEY `FKl7je3auqyq1raj52qmwrgih8x` (`product\_id`),

CONSTRAINT `FKedryofpdu4slsh36ilpb5y5a4` FOREIGN KEY (`buyer\_id`) REFERENCES `buyer` (`id`),

CONSTRAINT `FKl7je3auqyq1raj52qmwrgih8x` FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

**Flow Charts/Process Flows:**

Context Diagram

A diagram of a flowchart

Description automatically generated

Member Profile DataFlow

A diagram of a company

Description automatically generated

Category DataFlow

A diagram of a company

Description automatically generated

Product DataFlow

A diagram of a product

Description automatically generated

Order DataFlow

A diagram of a product

Description automatically generated

Process Flow Diagram

A diagram of a company

Description automatically generated

**Sitemap Diagram:**

A diagram of a diagram

Description automatically generated

**User Interface Diagrams:**

Homepage

A screenshot of a web page

Description automatically generated

Products Page

A screenshot of a web page

Description automatically generated

Login Page

A screenshot of a computer

Description automatically generated

Cart Page

A screenshot of a web page

Description automatically generated

Member Profile Page

A screenshot of a computer

Description automatically generated

About Us Page

A screenshot of a web page

Description automatically generated

**UML Diagrams:**

Member Screenflow

A screenshot of a computer

Description automatically generated

Spring Security Framework

A diagram of security

Description automatically generated

**Service API Design:**

**RESTful API Design**: Representational State Transfer (REST) APIs provide a way for the client-side of an application to communicate with the server-side. This architectural style allows different components of the application to communicate over HTTP in a way that mimics the behavior of web browsers and servers.

Localhost:8080/api/users

Localhost:8080/api/allproducts

Localhost:8080/api/addproduct

Localhost:8080/api/deleteproduct/{productid}

Localhost:8080/api/product/{productid}

Localhost:8080/api/updateProduct/{productid}

The Rest controllers are integrated within the user and admin controllers. These calls will be used by the front end to make additions and edits to products as well as new users.

**NFR’s (Security Design, etc.):**

Appealing design and an easy to use interface will be maintained to keep users coming back. Surveys can be used to gain feedback on what the user would like to see or how it could make the website work/flow better. Product descriptions will be obtained through the products manufacturer, and they will be contacted to provide a more descriptive product description. Product images can be obtained this way as well.

**Operational Support Design:**

At this point and time logging will not be supported. All tracking can be done from reports generated through the database.

**Other Documentation:**

You should insert any additional drawings, storyboards, white board pictures, project schedules, tasks lists, etc. that support your approach, design, and project. If you have no supporting documentation, explain the rationale why you are able to leave this section as N/A.

Proposed Schedule

|  |  |
| --- | --- |
| **Week 1-2** | **Create backend using Springboot and spring security** |
| **Week 3-4** | **Create frontend using React and JavaScript** |
| **Week 5-6** | **Add customers product data to database** |
| **Week 6-8** | **Run webapp on localhost and test all features (register new user, user login, admin login, add update and remove products.)** |
| **Week 8-10** | **Test 3rd party programs (payment methods)** |
| **Week 10-12** | **Deploy to test with small market** |
| **Week 12-14** | **Bug and issue fixes** |
| **Week 14** | **Deploy and test again with a larger market** |
| **Week 15** | **Fully deploy for customer business** |

Proposed Web Logo

A video game logo with an owl

Description automatically generated

Appendix A – Technical Issue and Risk Log

1. Use the template to identify and monitor project issues and risks.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Issues and Risk Log | | | | | | | | |
| **Issue or Risk** | **Description** | **Project Impact** | **Action Plan/Resolution** | **Owner** | **Importance** | **Date Entered** | **Date to Review** | **Date Resolved** |
| I/R | What is the issue or risk? | How will this impact scope, schedule, and cost? | How do you intend to deal with this issue? | Who manages this issue? |  |  |  |  |
| I | No asset images for products | Without images, users will not be able to see what they are purchasing | Get images from the customer | Ryan | High | 03/03/2024 | 03/17/2024 | n/a |
| R | 3rd party integration | Some functions of the website may not work | Check with compatibility of 3rd party components before deployment | Ryan | Medium | 03/03/2024 | 03/17/2024 | n/a |
| R | Security | Webapp may be at risk of hacking | Integrate salting and hashing of sensitive users information in database | Ryan | High | 03/03/2024 | 03/17/2024 | n/a |
| R | Performance | Under high traffic, application not perform optimally. | Ensure that it can handle a large amount of traffic. | Ryan | High | 03/03/2024 | 03/17/2024 | n/a |
| R | Compliance | Users data and privacy may be put at risk. | Ensure that the website adheres to all laws and regulations including data protection and privacy. | Ryan | High | 03/03/2024 | 03/17/2024 | n/a |
| R | Users will not find UI appealing | This will cause users to lose interest of using the customers website. | Deploy multiple versions for testing to ensure that there is at least one that is visually appealing. | Ryan | High | 03/03/2024 | 03/17/2024 | n/a |
|  |  |  |  |  |  |  |  |  |

Appendix B – References

*Functional Specification Documents: your complete guide*. (n.d.). Copyright © Justinmind 2023. All Rights Reserved. https://www.justinmind.com/blog/functional-specification-documentation-quick-guide-to-making-your-own/

GeeksforGeeks. (2022). Unified Modeling Language UML activity diagrams. *GeeksforGeeks*. https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/

Appendix C – External Resources

|  |  |
| --- | --- |
| **GIT URL:** | *https://github.com/rcoon1/CST451Project* |
| **Hosting URL:** | *The Hosting URL (if applicable).* |