TheCouchPotato.com: An E-commerce Platform for Comfort Items

Christopher Markel

CST-451 Capstone Project Requirements Document

Grand Canyon University

Instructor: Professor Amr Elchouemi

Revision: 1

Date: 8/27/2023

**ABSTRACT**

TheCouchPotato.com is an e-commerce platform designed to cater to a niche market of comfort items, providing a one-stop solution for consumers seeking products that enhance their comfort and relaxation. This project aims to develop a user-friendly platform that ensures secure transactions, offers a diverse range of comfort products, and promotes sustainable and ethically sourced products. This Requirements Document outlines the functional and non-functional requirements, system design, technical requirements, system logical model, screen definitions and layouts, and security considerations necessary for the successful development and deployment of TheCouchPotato.com. The document serves as a comprehensive guide for stakeholders and development teams to understand and implement the necessary components and features of the platform.

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

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 8/27/2023 | Christopher Markel | Initial draft for review/discussion |
|  |  |  |
|  |  |  |

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

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

**Integrated Instructor Feedback into Project Documentation**

Yes  No

**TABLE OF CONTENTS**

Functional Requirements 4

Non-Functional Requirements 5

Technical Requirements 6

Logical System Design 7

User Interface Design 8

Reports Design 9

Functional Requirements

**Use Cases**

Describe the sequence of functional actions a project performs with either textual Use Cases, UML Use Case diagrams, or if using Scrum, provide a link to the User Stories (see template included in course materials).

NOTE: Once the functional requirements have been completed, there may be situations where Use Cases or User Stories may need to be taken out of scope, possibly due to technical challenges or timeline challenges. Any Use Cases or User Stories that are taken out of scope once the project development has started must be approved by the mentor and instructor with justification as to why the functionality is being removed from the project. The following must be updated if any Use Cases or User Stories are taken out of scope:

|  |  |  |
| --- | --- | --- |
| **Use Case or User Story** | **Approval Date** | **Justification** |
| User Registration: Allows a new user to register by providing their name, email address, and password. |  |  |
| User Login: Allows a returning user to log in using their email address and password. |  |  |
| Product Search: Allows a user to search for products by entering keywords. |  |  |
| Product Filter: Allows a user to filter products by price range, category, and brand. |  |  |
| Product Sort: Allows a user to sort products by price, popularity, and rating. |  |  |
| Add to Cart: Allows a user to add products to their cart for later purchase. |  |  |
| View and Update Cart: Allows a user to view their cart and update the quantity of items in it. |  |  |
| Remove from Cart: Allows a user to remove items from their cart. |  |  |
| Checkout: Allows a user to enter their shipping and billing information to complete their purchase. |  |  |
| View Order History: Allows a user to view their past purchases and order details. |  |  |
| Manage Products (Admin): Allows an admin to add, update, and delete products from the catalog. |  |  |
| Manage Categories (Admin): Allows an admin to organize products into different categories. |  |  |
| Manage Orders (Admin): Allows an admin to view order details, update order status, and generate reports on orders and sales. |  |  |
| Manage User Accounts (Admin): Allows an admin to view user details and update user information. |  |  |
| Access from Different Devices: Allows a user to access the application from various devices such as desktop, laptop, tablet, or mobile phone. |  |  |

**Architecture**

*The e-commerce web application will follow a standard Model-View-Controller (MVC) architecture. The application will be built using the Spring Boot framework and Java programming language, with a MySQL database for storing user and product information. Figure 1 below shows the high-level architecture of the application.*

*A diagram of a data flow

Description automatically generated*

*Figure 1: In this diagram, the web client communicates with the MVC controller layer through HTTP. The controller layer handles business logic and communicates with the data access layer to retrieve or update data from the database layer.*

**User Registration**

*This requirement involves the ability for users to create an account on the e-commerce web application. Users should be able to provide their personal information such as name, email, and password to register an account. Upon successful registration, the user should receive a confirmation email or notification. This requirement may also include the ability for users to edit their profile information.*

| ID | Requirement |
| --- | --- |
| FR-1.1 | The application shall allow users to register for an account by providing their name, email address, and a password. |
| FR-1.2 | The application shall validate that the email address provided by the user is not already associated with an existing account. |
| FR-1.3 | The application shall store user account information in a MySQL database. |
| FR-1.4 | The application shall display a confirmation message to the user upon successful registration. |

**User Login**

*This requirement involves the ability for registered users to log in to the e-commerce web application. Users should be able to enter their username and password to authenticate and access their account. The system should verify the user's credentials and redirect them to the appropriate landing page. In case of invalid login credentials, appropriate error messages should be displayed.*

| ID | Requirement |
| --- | --- |
| FR-2.1 | The application shall allow registered users to log in by providing their email address and password. |
| FR-2.2 | The application shall verify the user's credentials before granting access to protected areas of the site. |
| FR-2.3 | The application shall display an error message if the user's credentials are invalid. |

**Product Browsing**

*This requirement involves the ability for users to browse the products available on the e-commerce web application. Users should be able to search for products using keywords or categories, view product details such as description, price, and availability, and add products to their cart. The system should also provide filtering and sorting options for product listings.*

| ID | Requirement |
| --- | --- |
| FR-3.1 | The application shall display a list of available products on the home page. |
| FR-3.2 | The application shall allow users to filter products by category, price, and other relevant attributes. |
| FR-3.3 | The application shall display detailed information about each product, including price, description, and images. |
| FR-3.4 | The application shall provide users with the ability to add products to a shopping cart. |

**Product Purchasing**

*This requirement involves the ability for users to purchase products on the e-commerce web application. Users should be able to add products to their cart, proceed to checkout, provide shipping and billing information, and complete the payment process. The system should also provide confirmation of the order, estimated delivery dates, and order tracking information.*

| ID | Requirement |
| --- | --- |
| FR-4.1 | The application shall allow users to view the contents of their shopping cart and update quantities as necessary. |
| FR-4.2 | The application shall allow users to enter their shipping and billing information during the checkout process. |
| FR-4.3 | The application shall display a summary of the order before processing payment. |
| FR-4.4 | The application shall process payments securely using a third-party payment gateway. |
| FR-4.5 | The application shall send a confirmation email to the user upon successful order completion. |

**Database**

*The MySQL database will be the backbone of the e-commerce web application, storing all user, product, and order information. The database will need to be designed to handle a large number of concurrent users and requests and should be scalable for future growth. The database will be secure, protecting user information from unauthorized access, and will be reliable to minimize the risk of data loss or corruption. Additionally, the database will need to be easy to maintain and administer, allowing for efficient updates and backups.*

| ID | Requirement |
| --- | --- |
| FR-5.1 | The database should allow users to register an account and store their personal information securely. |
| FR-5.2 | The database should allow users to login to their account securely. |
| FR-5.3 | The database should store product information including name, description, price, and availability. |
| FR-5.4 | The database should store order information including products ordered, quantity ordered, and total price. |
| FR-5.5 | The database should allow users to view their order history. |

Non-Functional Requirements

**Use Cases**

Describe the sequence of non-functional actions a project performs with either textual Use Cases, UML Use Case diagrams, or if using Scrum, provide a link to the User Stories (see template included in course materials).

NOTE: Once the non-functional requirements have been completed, there may be situations where Use Cases or User Stories may need to be taken out of scope, possibly due to technical challenges or timeline challenges. Any Use Cases or User Stories that are taken out of scope once the project development has started must be approved by the mentor and instructor with justification as to why the functionality is being removed from the project. The following must be updated if any Use Cases or User Stories are taken out of scope:

|  |  |  |
| --- | --- | --- |
| **Use Case or User Story** | **Approval Date** | **Justification** |
| Usability: Ensures the application is user-friendly and easy to navigate for all users. |  |  |
| Performance: Ensures the application loads quickly and responds to user interactions without delay. |  |  |
| Security: Ensures the application is secure and protects user data from unauthorized access. |  |  |
| Scalability: Ensures the application can handle a large number of users and data without performance degradation. |  |  |
| Compatibility: Ensures the application is compatible with various devices and browsers. |  |  |
| Accessibility: Ensures the application is accessible to users with disabilities. |  |  |

| ID | Requirement |
| --- | --- |
| NFR-1.1 | Security: All user data should be protected from unauthorized access, and the system should be protected against attacks such as SQL injection, cross-site scripting, and cross-site request forgery. |
| NFR-1.2 | Performance: The system should be able to handle a large number of concurrent users, and response times for user registration, login, product browsing, and purchasing should be fast and consistent. |
| NFR-1.3 | Availability: The system should have a high level of uptime, with minimal downtime for maintenance and upgrades. |
| NFR-1.4 | Scalability: The system should be able to handle growth in user traffic and data storage requirements. |
| NFR-1.5 | Compatibility: The system should be compatible with a wide range of web browsers and operating systems. |
| NFR-1.6 | Usability: The user interface should be intuitive and easy to use, with clear instructions and error messages. |
| NFR-1.7 | Reliability: The system should be reliable, with minimal errors or crashes. |
| NFR-1.8 | Maintainability: The system should be easy to maintain, with clear documentation and well-organized code. |
| NFR-1.9 | Accessibility: The system should be accessible to users with disabilities, complying with web accessibility guidelines. |
| NFR-1.10 | Data integrity: The database should maintain the integrity of data, with proper data validation and error handling. |
| NFR-1.11 | Backup and recovery: The database should be regularly backed up and have a plan for disaster recovery. |
| NFR-1.12 | Data privacy: The system should comply with relevant data privacy laws and regulations, with user consent and proper handling of personal data. |

Technical Requirements

**Use Cases**

Describe the tools and technologies used in the project.

NOTE: Once the technical requirements have been completed, there may be situations where technologies or tools may need to be taken out of scope or changed, possibly due to technical challenges or timeline challenges. Any technologies or tools that are taken out of scope or changed once the project development has started must be approved by the mentor and instructor with justification as to why the functionality is being removed from the project. The following must be updated if any technologies or tools are taken out of scope or changed:

|  |  |  |
| --- | --- | --- |
| **Technology or Tool** | **Approval Date** | **Justification** |
| React.js: A JavaScript library for building user interfaces. It will be used to develop the frontend of the application. |  |  |
| Java: A high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It will be used to develop the backend of the application. |  |  |
| MySQL: An open-source relational database management system. It will be used to store the application data. |  |  |
| Github: A distributed version control system that will be used to track changes, revert to previous stages, and work on different branches simultaneously. |  |  |
| Basic HTML Authentication: A simple authentication scheme built into the HTTP protocol. The client sends HTTP requests with an Authorization header field containing the word 'Basic' followed by a space and a base64-encoded string 'username:password'. It will be used for user authentication. |  |  |
| AWS (Amazon Web Services): FUTURE implementation- A comprehensive cloud computing platform that will be used to host the application. |  |  |

**User dialogs and control flow:**

The application will be designed to provide an intuitive user interface with clear navigation paths. The user interface will be designed using HTML, CSS, and JavaScript and will provide a simple and user-friendly experience for the users. The application will use a model-view-controller (MVC) design pattern to manage user interactions and control flow. The presentation layer will handle user inputs, the business layer will handle the logic and processing of the inputs, and the data layer will handle data storage and retrieval.

For more details, see *User-related Tasks/Features* below.

**User-related tasks/features:**

The e-commerce application will provide a range of user-related tasks and features such as user registration, user login, product browsing, product purchasing, and user account management. User registration will allow new users to create an account on the platform. User login will allow registered users to access the application and perform tasks such as browsing products and making purchases. Product browsing will enable users to view the available products and their details, and product purchasing will allow users to make purchases and track their orders. User account management will allow users to edit their personal information, track their orders, and manage their account settings.

**Background tasks:**

The application will perform several background tasks such as data validation, data processing, and database management. Data validation will ensure that the data entered by the user is in the correct format and meets the application's requirements. Data processing will involve executing business logic and performing necessary calculations and operations on data. Database management tasks will involve managing the database connection and ensuring that the data is stored and retrieved efficiently.

**Server and client side:**

The e-commerce application will be designed with both server-side and client-side functionality. The server-side functionality will include the business logic and data storage and retrieval, while the client-side functionality will involve the presentation layer and user interactions. The server-side functionality will be implemented using Spring Boot MVC and Java, while the client-side functionality will be implemented using HTML, CSS, and JavaScript.

**Data Security:**

To ensure the security of user data, basic authentication will be implemented in the system. This means that users will be required to provide a username and password to access the system. The password will be hashed and salted before being stored in the MySQL database.

Additionally, the system will implement data validation to prevent any malicious code or unauthorized access. Data validation will be applied to all inputs received from users, including form submissions and API requests. This will help to prevent any SQL injection attacks or other forms of malicious code injection.

The system will also use HTTPS to encrypt all data sent between the user's browser and the server, ensuring that data is transmitted securely over the internet.

In addition to these measures, regular backups of the database will be taken to prevent data loss in the event of a system failure or security breach. These backups will be stored securely and encrypted to prevent unauthorized access.

Overall, these measures will ensure that the system is secure, and that user data is protected from unauthorized access or malicious attacks.

*Authentication:*

* *The system should prompt the user for their username and password upon login.*
* *Usernames and passwords should be stored securely in the database, using encryption or hashing.*
* *Passwords should be validated against password complexity rules, such as minimum length, complexity requirements, and expiration policies.*
* *If a user enters an incorrect username or password, the system should display an error message and allow the user to try again.*
* *After a certain number of failed attempts, the system should lock the user account for a period of time to prevent brute-force attacks.*

*Data validation:*

* *The system should validate all user input, including form submissions and API requests, to prevent malicious input and injection attacks.*
* *Input validation rules should be defined for each input field, such as length, data type, format, and allowed characters.*
* *The system should display meaningful error messages when input validation fails, indicating which field is invalid and why.*
* *The system should also prevent SQL injection attacks by sanitizing all database queries and using prepared statements or parameterized queries.*

**Interfaces with other systems:**

The e-commerce application will be designed to interface with other systems such as payment gateways, shipping carriers, and inventory management systems. Payment gateways will enable users to make secure online payments, while shipping carriers will allow users to track their orders and receive updates on the status of their deliveries. Inventory management systems will enable the application to manage and update the availability of products in real-time.

Logical System Design

Provide a diagram of the logical architecture of the system.

**A diagram of a product

Description automatically generated**

*Figure 2: This logical system diagram shows a high-level overview of the system, how the components are interconnected, and how they contribute to the system.*

**Shopping Cart Logical User Diagram**

**A diagram of a flowchart

Description automatically generated**

*Figure 3: The shopping cart logical user diagram shows a high-level overview of the shopping cart system, how the components are interconnected, and how they contribute to the system.*

User Interface Design

Provide a sitemap and user interface design diagram for each user interface screen in the application, if not applicable, define the components of the project as described in the handbook.

**Site Map**

*The site map for the e-commerce website includes the following pages:*

*Home page: Displays website banners/logo, registration/login prompts.*

*Login page: Allows users to log in to their existing account.*

*Register page: Allows users to create a new account by providing personal information.*

*User admin page: Displays user profile information, allows users to change their password and update their account information.*

*Product admin page: Allows administrators to add, edit, and delete products.*

*Search page: Allows users to search for products by keywords, categories, or other criteria.*

*Shopping cart: Displays the list of items selected for purchase and allows users to modify the quantity of items.*

*Checkout page: Allows users to review their order details, select a shipping method, and enter payment information.*

**A diagram of a product

Description automatically generated**

*Figure 4: This site map provides a clear and logical structure for the website, allowing users to easily navigate and find the information they need.*

**Index**

A screenshot of a web page

Description automatically generated

*Figure 5: A wireframe of the Index, or Home page*

**Products**

A screenshot of a web page

Description automatically generated

*Figure 6: A wireframe of the Products page*

**Login**

A screenshot of a login page

Description automatically generated

*Figure 7: A wireframe of the Login page*

**Registration**

A screenshot of a login form

Description automatically generated

*Figure 8: A wireframe of the Registration page*

**Add Product**

A screenshot of a web page

Description automatically generated

*Figure 9: A wireframe of the Add Product page*

**Update Product**

A screenshot of a web page

Description automatically generated

*Figure 10: A wireframe of the Update Product page*

**Delete Product**

A screenshot of a web page

Description automatically generated

*Figure 11: A wireframe of the Delete Product page*

**Reports Design**

Provide a listing of the reports that the system will provide, if applicable. If not, state that the system does not produce any reports and provide additional documentation as described in the handbook.

The system does not produce any reports.

In lieu of a reports design, here is a visualization of what one of the UI pages might look like. Here is an example of a Products page, which lists all products in the database.

A screenshot of a computer

Description automatically generated

**Security Issues**

*Data Breach:* Unauthorized access and retrieval of sensitive data by malicious actors.

*SQL Injection:* A type of injection attack in which an attacker can execute malicious SQL code in the backend database.

*Cross-Site Scripting (XSS):* A type of attack in which malicious scripts are injected into trusted websites.

*Payment Fraud:* Unauthorized transactions made using stolen payment information.

*Account Hijacking:* Unauthorized access and control over a user's account.

To address these security issues, the following measures will be implemented:

*Data Encryption:* Encrypting data in transit and at rest.

*Input Validation:* Validating all user inputs to prevent SQL injection and XSS attacks.

*Authentication and Authorization:* Implementing strong authentication and authorization mechanisms to ensure that only authorized users can access certain parts of the application.

*Secure Payment Processing:* Using a secure and trusted third-party payment processor to handle online transactions.

*Regular Security Audits:* Conducting regular security audits to identify and fix any security vulnerabilities.

**Security Matrix**

| **Action** | **Guest User** | **Registered User** | **Admin** |
| --- | --- | --- | --- |
| View Products | Yes | Yes | Yes |
| Search Products | Yes | Yes | Yes |
| Add Products to Cart | Yes | Yes | Yes |
| View Cart | Yes | Yes | Yes |
| Checkout | No | Yes | Yes |
| View Order History | No | Yes | Yes |
| Leave Reviews and Ratings | No | Yes | Yes |
| View Product Recommendations | No | Yes | Yes |
| Receive Notifications | No | Yes | Yes |
| Save Products to Wishlist | No | Yes | Yes |
| Track Order Status | No | Yes | Yes |
| Access Application on Devices | Yes | Yes | Yes |
| Manage Products | No | No | Yes |
| Manage Categories | No | No | Yes |
| Manage Orders | No | No | Yes |
| Manage User Accounts | No | No | Yes |
| Configure Application Settings | No | No | Yes |

**Risk Mitigation Plan**

| **Risk** | **Probability** | **Impact** | **Mitigation Strategy** | **Contingency Plan** |
| --- | --- | --- | --- | --- |
| Data Breach | High | High | Data Encryption, Regular Security Audits | Notify users, rectify the breach |
| SQL Injection | Medium | High | Input Validation | Rectify the vulnerability |
| XSS Attack | Medium | High | Input Validation | Rectify the vulnerability |
| Payment Fraud | Medium | High | Secure Payment Processing | Refund affected users, rectify issue |
| Account Hijacking | Medium | High | Strong Authentication and Authorization | Reset affected user accounts |
| Project Delays | Medium | Medium | Project Management | Allocate additional resources |
| Budget Overruns | Low | Medium | Budget Management | Reallocate funds, seek additional funding |

**References**

Bigelow, S. (2020). Functional vs. nonfunctional requirements in software engineering. Retrieved from <https://www.techtarget.com/searchsoftwarequality/answer/Functional-and-nonfunctional-requirements>

chitrasingla2001. (2022). Functional vs Non Functional Requirements. Retrieved from <https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/>

GALLI, B. J. (2017). Risk Management in Project Environments: REFLECTION OF THE STANDARD PROCESS. Journal of Modern Project Management, 39–49.

Maiden, N. (2013). Monitoring Our Requirements. IEEE Software, 30(1), 16–17. <https://doi-org.lopes.idm.oclc.org/10.1109/MS.2013.10>