RevShop

(Application Development)

Center of Excellence
10-May-2023

Version 1.0

Table of Contents

Table of Contents.....2
Application Overview.....3
Core Functional Scope.....3
Standard Functional Scope.....3
Definition of Done.....4
Competency wise scoping.....4
Non-Functional Expectations.....9

Application Overview

The RevShop project aims to develop a secure, user-friendly, and versatile e-commerce application for both buyers and sellers. The core functionalities for buyers include browsing products, adding products to a cart, checkout, and payment processing. Sellers can add products, manage inventory, and fulfill orders. The project's completion will be demonstrated through a cloud-hosted working version, technical presentation, and associated diagrams.

Core Functional Scope

Buyer user account:

As a buyer, I should be able to:

1. Register on the platform.
2. Login into the application using email and password.
3. View product details including image, price, description, and user review.
4. Browse products by category or keywords.
5. Add or remove products from the cart and provide quantity.
6. Checkout and enter shipping and billing information.
7. Get email notifications when an order is placed.
8. View order history.
9. Review products.
10. Save the product as a favorite.
11. Make payment using the payment gateway.

Seller account:

As a seller, I should be able to:

1. Register as a seller with email, password, and business details.
2. Login into the application using email and password.
3. Manage inventory of products.
4. Add new products with price and description.
5. See placed orders.
6. Receive email notifications when a user places an order.
7. Provide discounted price along with the maximum retail price.
8. View product review.
9. Get web notifications when the product's quantity is less than the threshold. (Seller sets the threshold value).

Standard Functional Scope

Registered users should be able to log in, change the password and request for a forgotten password (which will be sent to their registered email).

Definition of Done

- Working application demonstration.
- Sharing the associates' code repo for technical evaluation with:
    - ERD Diagram
    - Architecture Diagram

Competency wise scoping

Competency	Application Type	Expectations
Java SQL REST / Java SQL REST Unix / Java SQL REST Gradle / C# SQL REST / Python SQL / JavaScript SQL / JavaScript NoSQL	Console Based Application	**User Inputs:** 1. Ability to accept the user inputs from console 2. Providing recommended format in which the user to key in the inputs 3. Validate the user given inputs for format and convert to appropriate type for application usage. **System outputs:** 1. Use formatted outputs for better readability and understanding. E.g., currency and date values should be formatted well. 2. Display the reports in the appropriate format such as tables etc. **User Navigation:** 1. Provide a number-based menu items for the user to navigate for different use cases 2. Handling user selections and providing appropriate screen / feature to the user **Validation and Error Handling:** 1. Validate the user inputs for its types and format. 2. Display functional related user messages (either for input/error/output) - no system error codes. 3. Handle the exceptions and errors gracefully.

		Logging: 1. Ensure the application is using proper logging framework and methods. 2. Ensure the application's log level is configured using configuration files so that it can be changed without changing the code. 3. Also ensure that the application logging is configured to output to the mentioned log file. **Testing**: 1. Ensure sufficient test cases are written using appropriate testing frameworks. 2. Ensure the code coverage closed to be 80% **Security**: 1. Ensure the SQL/NoSQL injection threat is taken care. **Coding Standard:** 1. Use the industry coding standards and conventions. 2. Modular based code development for better reusability. 3. Ensure proper usage of resource objects such as database connectivity objects to avoid resource leakages. 4. Ensure proper usage of design patterns and application layering (such as Business Service, DAO Layer etc.) wherever applicable.
Web Fundamentals (HTML, CSS and JS)	Web Navigational Prototype	**User Experience:** 1. Have an intuitive design for the user to work with the application without any training or guidance 2. Have clean & consistent UI, color theme and easy to use navigations 3. Use bootstrap framework for responsive pages 4. Have proper tab indexing for users to navigate between the fields without usage of mouse. **User Inputs & outputs:** 1. Have appropriate HTML fields for the user inputs

		2. Wherever possible use the client-side validations for the user input 3. Display the appropriate user info/error message with appropriate colors and icons **Performance:** 1. Use compressed images / assets to increase the page performance 2. Use the application validated using the Chrome's Lighthouse tool and improved based on the report **Dataset:** 1. For any prepopulated data such as to render table rows use JSON file as the DataSource and use standard open-source library to read and render in the web pages. **General standards:** 1. Ensure the w3 standards are implemented for better accessibility. E.g., Using alt attribute for image tag. 2. Ensure the SEO recommended meta tags are added.
Web Development with Angular / Web Development with Angular JS/ Web Development with React / Web Development with React Redux	Enhanced Web Navigational Prototype using the selected framework (either Angular/React/Vue.JS)	**Same as Web Fundamental Competency above, with the following additions:** **Framework Specific** 1. Ensure the appropriate APIs are used for any of the API calls 2. Ensure the routing is centrally configured 3. Best practices & design patterns are to be followed 4. Implement the end-to-end testing framework and get to know the headless execution of end-to-end framework. **Deployment artifacts:** 1. The deployment artifacts should be minified and obfuscated if required. **Security**: 1. Ensure the CORS restriction is applied, if applicable.

		2. Ensure Route Guarding/Authenticated Routing is implemented. 3. Ensure that the secrets are stored as environment variables using secure credential storage.
Spring Boot / ASP.NET API REST / Java SQL REST / Java SQL REST Unix / Java SQL REST Gradle / SpringBoot with Authentication & Messaging	Identifying and Developing All APIs for any front-end application to consume	**REST Standards:** 1. Ensure the REST standards are followed for API naming, HTTP Operation and Response (output definition) 2. Secure the protected APIs 3. Define a common URL pattern for public and secure APIs 4. Proper documentation of APIs with Input and Output Samples to be documented 5. Ensure that each micro service is defined to do one job. 6. Provide / enable to API gateway to route the request through a single channel 7. Ensure the micro services are enabled for traceability and monitoring with appropriate dashboard to render the micro services health and logging. 8. Use the recommended design patterns to aggregate the micro services (wherever is applicable) **Modularity:** 1. If the application is enhanced from an existing console-based project, try to reuse the existing modules. 2. Refactor the existing code based on the current tech stacks **Logging:** 1. Ensure the application is using proper logging framework and methods. 2. Ensure the application's log level is configured using configuration files so that it can be changed without changing the code. 3. Also ensure that the application logging is configured to output to the mentioned log file. 4. Ensure the centralized logging implementation

		Testing: 1. Ensure sufficient test cases are written using appropriate testing frameworks. 2. Ensure the code coverage closed to be 80% **Security**: 1. Ensure the CORS restriction is applied. 2. SQL Injection thread is taken care 3. Throttling is to be taken care of. 4. Cross site Scripting to be avoided. 5. Ensure that the secrets are stored as environment variables using configuration files or secure credential storage.
Devops .NET Azure GHA/Devops(CI/CD) Azure Jenkins/Devops(CI/CD) AWS Jenkins	Implementing DevOps Practices and CI/CD Pipelines for Cloud-Based Applications	1. Ensure to implement DevOps best practices, emphasizing collaboration, automation, and continuous improvement throughout the software development and deployment lifecycle. 2. Leverage cloud services promptly to meet specific application hosting and deployment needs. 3. Ensure consistent and repeatable resource deployment by applying Infrastructure as Code (IaC) principles to define and provision cloud resources programmatically. 4. Implement cloud-specific security measures to protect applications and data. 5. Create and manage CI/CD pipelines using Jenkins or GitHub Actions to automate building, testing, and deploying applications. 6. Implement pipelines in either scripted or declarative form to streamline the software delivery process.

Non-Functional Expectations

- Application development supposed to follow the Scrum process
- Application password should be encrypted using appropriate hashing algorithms
- Applications should use the recommended authentication token such as JWT or other equivalent