e-Bi - Software Requirements Document (SRD)

Version Control

Version	Date	Author	Change
0.1	01/10	Gabriel Pinheiro	Document creation.
0.2	03/10	Gabriel Pinheiro	Add introduction and initial functional requirements.
0.3	07/10	Gabriel Pinheiro	Add catalog and chekout functional requirements, assumptions and constraints.

1. Introduction and Purpose

1.1 Introduction

e-Bi is a full back-end environment project for an e-commerce platform. It aims to include all the basic functions (and some extra ones) of a real e-commerce system, from product selection to payment integration and product shipping. All external integrations will use staged resources whenever possible; if not feasible, mocks will be employed, as this project does not seek to be a real production application.

1.2 Purpose

The purpose of this project is to build a tangible and realistic e-commerce back-end from scratch. All the effort aims to facilitate learning in areas such as requirements evaluation, architectural decision-making, best coding practices, and system integration. Throughout the development process, challenges will be encountered, problems solved, and lessons learned. Additionally, the project will focus on identifying failures and exploring resolutions.

2. Requirements

2.1 Functional Requirements

1. **Product Catalog Management**
    - The system **must** provide featured products to display on the front page.
    - The system **must** allow users to search for products by name, SKU, or keyword.
    - The system **must** display available product categories for browsing.
    - The system **must** allow users to search for products within specific categories.
    - The system **must** provide both a simple and a detailed description of products.
    - The system **must** support filtering of products based on name, price, discount, or relevance.

- The system **must** allow users to mark products as favorites for easy reference.
- The system **must** display products currently on sale or under special promotions.

2. **Shopping Cart and Payment Processing**
    - The system **must** securely process user payments.
    - The system **must** display all order details, including product names, quantities, prices, and shipping costs, before final confirmation.
    - The system **must** offer at least two payment options.
    - The system **must** list all products in the cart, showing product images, names, quantities, and total prices.
    - The system **must** allow users to select, add, or change the shipping address during the checkout process.
    - The system **must** display available payment options at checkout and allow users to select their preferred method.
    - The system **must** allow users to apply discount codes or coupons before finalizing the purchase.
    - The system **must** notify users of successful payment confirmation via on-screen notification and/or email.
3. **User Account Management**
    - The system shall allow users to manage their accounts (create, update, delete).
    - The system shall implement two-factor authentication (2FA) for account security.
    - The system shall send emails to users about promotions and offers.
4. **Logistics and Reporting**
    - The system shall generate invoices for purchases.
    - The system shall integrate with a shipping provider for item collection or generate reports of items to be shipped.

5. **Inventory Management**
    - The system shall manage product inventory levels and update stock accordingly.
6. **Product Reviews and Ratings**
    - The system shall allow users to rate products and leave comments.
7. **Promotional Features**
    - The system shall enable users to view promotional offers on selected products.
8. **Order History**
    - The system shall allow users to view their order history and status.
9. **Wishlist Functionality**
    - The system shall allow users to create and manage wishlists of products they are interested in.
10. **User Notifications**
    - The system shall send notifications to users regarding order updates, promotions, and other relevant information.

2.2 Non-functional Requirements

1. **Performance**:
    - The system should handle a minimum of 10 requests per second.
2. **Scalability**:
    - The system architecture should support the addition of more services or features with minimal disruption.
3. **Security**:
    - The system must comply with the Brazilian General Data Protection Law (LGPD) by ensuring secure collection, processing, and storage of personal data, obtaining user consent for data processing, and allowing users to access and delete their data

- The system must enforce secure communication using HTTPS for all external connections.
- Sensitive endpoints should require two-factor authentication (2FA).

4. **Availability**:
    - The system should ensure minimal downtime during updates or maintenance.
5. **Maintainability**:
    - Code should follow established clean code principles, making it easy to maintain and extend.
    - Documentation for each service should be updated with every new feature or bug fix.
6. **Usability**:
    - APIs must have comprehensive error handling and clear response structures.
7. **Logging and Monitoring**:
    - The system should log all API requests, including metadata, errors, and warnings, to a centralized logging service.
    - Real-time monitoring should be in place to track system health and raise alerts for anomalies.

3. External Requirements

4. Assumptions and Constraints

4.1 Assumptions

1. **User Behavior**:
    - Users will have a stable internet connection.

2. **Technology**:
    - The system will integrate with third-party services that will have APIs available.
3. **Infrastructure**:
    - The system will be deployed in a non-production environment.

4.2 Constraints

1. **Budget**:
    - The project will utilize free resources, including Google Cloud’s free tier, Cloud Functions, GitHub Actions for CI/CD.
    - Certain features, such as full integration with external payment systems and shipping services, may be staged or mocked due to the project's limited budget.
2. **Technology Choices**:
    - The project is limited to technologies that the development team is familiar with or that can be learned during the project’s duration.
3. **Performance**:
    - The system will not be optimized for high traffic or complex data processing initially, focusing instead on showcasing functionality and architectural best practices.
    - The services will have limited resources during development (CPU, memory) as they will primarily be running on free or budgeted cloud tiers.