Software Requirements Specification

for

Ecommerce-Experiment

Prepared by Andrew Harley

05/12/2023

Table of Contents

Table of Contents ii
Revision History ii
1. Introduction 1
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 Product Scope 1
1.5 References 1
2. Overall Description 2
2.1 Product Perspective 2
2.2 Product Functions 2
2.3 User Classes and Characteristics 2
2.4 Operating Environment 2
2.5 Design and Implementation Constraints 2
2.6 User Documentation 2
2.7 Assumptions and Dependencies 2
3. External Interface Requirements 3
3.1 User Interfaces 3
3.2 Hardware Interfaces 3
3.3 Software Interfaces 3
3.4 Communications Interfaces 3
4. System Requirements 3
5. Other Nonfunctional Requirements 5
5.1 Performance Requirements 5
5.2 Safety Requirements 5
5.3 Security Requirements 5

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

An ecommerce website is to be designed based on this document. The product is completely included and described by the requirements in this document. See Product Scope for more details about the purpose of making this software.

1.2 Document Conventions

The Lilex Nerd Font was used in the descriptions of each subsection in this document. Any obvious highlighting is to demonstrate emphasis. Some requirements may be revised in the future of the creation of this document. Requirements may be parental, or inherited from a parent. A parent requirement may have multiple inherited requirements. Throughout the document, it will refer to EF which is an acronym for Entity Framework of C#. ORM in this document stands for Object Relational Mapper.

1.3 Intended Audience and Reading Suggestions

The intended audience of this document is prospective or interested employers. Suggested *technical* reading is about ORMs (or differences between Micro-ORMs and ORMs as EF is an ORM and Dapper is a Micro-ORM), pnpm, PostgreSQL, and Playwright (which is used for testing).

1.4 Product Scope

The vision of this website is that it will be able to demonstrate the software engineer's skills associated with SQL and TypeScript/React. The goal is to showcase these skills to the intended audience. Sub-goals of this project include, but are not limited to, C# (Entity Framework), HTML, CSS, architectural development, PostgreSQL (And SQL itself), and Test-Driven Development (TDD).

1.5 References

This SRS may refer to a GitHub repository, such as the main one utilized for this project: https://github.com/HAL703/ecommerce-experiment

2. Overall Description

2.1 Product Perspective

This software is designed to showcase skills to prospective employers, and for learning more about SQL (PostgreSQL)/React/TypeScript. It also allows me to learn more about using pnpm exclusively. A major goal of the software is to extend my SQL query making skills. It is not supposed to have any realistic functions as an e-commerce platform. Note that this major section will develop throughout the software's life-cycle.

2.2 Product Functions

The software will be designed to serve the user products that they can buy. The user will be able to add products to a cart, and then buy with a credit card. PayPal or any other third-party service equivalent will not be a focus for this project. The user will have an account and be able to access settings for that account. Do note that the products will either be made manually or auto-generated as fake products. TBD

2.3 User Classes and Characteristics

TBD

2.4 Operating Environment

This project will be designed to work with a SQL database service, such as PostgreSQL. The backend of the project will be C# with Entity Framework/Dapper. I am choosing EF and Dapper both as I want to develop my SQL exclusive skills with Dapper and show that I can do the same with EF (since EF is an ORM, it doesn't really help develop SQL proficiency). This website will be configured to work on mobile devices, such as Android or iOS, Windows 10/11, and Linux (with Firefox as the main browser).

2.5 Design and Implementation Constraints

TBA

2.6 User Documentation

User documentation will be developed over-time and displayed as a markdown (.md) document in the central GitHub repository.
See the linked section.

2.7 Assumptions and Dependencies

This project relies on React/Typescript libraries and the functionality of Entity Framework, Dapper, and Npgsql in C#. Mobile development will likely be towards the end of the project. Playwright/NUnit will be used for E2E and unit testing in this project. PostgreSQL will be used as the database of choice, using Datagrip/Rider and pgAdmin.

3. External Interface Requirements

3.1 User Interfaces

See *https://github.com/HAL703/ecommerce-experiment/tree/main/diagrams* for a list of diagrams used to design the website. These diagrams depict what the user will be interacting with as they explore the website. Note that a CheckoutPage is missing from the diagrams, as it is believed that it is easier for the page to be developed with the general intention of being a giant form for the address, credit card, etc. Possibly need an additional page for shipping costs. Can utilize an overlay for the shipping costs/final checkout.

3.2 Hardware Interfaces

A user will simply access the website from their mobile device or via a computer connected to the Internet.

3.3 Software Interfaces

NUnit/Vite (for Vitest) for testing REST APIs and a SQL service will be used, such as PostgreSQL. See the actual requirements for a more detailed explanation.

3.4 Communications Interfaces

The user must have encrypted data in the form of passwords, credit card information, and possibly address information.

4. System Requirements

This section contains all the formal requirements needed to create the software described in this SRS. The format will be a bulleted list. Each requirement has either **shall**, **will**, or **shall/will** at the beginning of it to indicate that it's **necessary**, **wanted**, or **will be done if time allows** in that order. **This list may be expanded as the project progresses.**

(1) **Shall** have a registration/login system, including credit card and address entries.
(2) **Shall** have an encryption system for more private user data.
(3) **Shall** have a top bar with a logo, search function, cart, and account entries.
(4) **Shall** have a sub-top bar with categories for items, only to be used on the store page.
(5) **Shall** have 20-30 items per page on the store page. Each item shall have a picture and a few functions like favoriting stored in a gear icon.
(6) **Shall** have some way to formally deal with picture storage.
(7) **Shall** have an item showcase page with images, a description, and the ability for the user to add an item to a cart.
(8) **Shall** have an intro page that restricts the user to either registration or logging in before doing anything else.
(9) **Shall** have an account page where the user can change all various fields such as credit card information, address, email, password, etc.
(10) **Shall** have functions to reset various fields such as email in the account page.
(11) **Shall** have an about page that gives more information about the website, or where to contact for more information.
(12) **Shall** have a checkout page with all the required fields for making a purchase of an item.
(13) **Shall** use both C# Entity Framework and Dapper to interact with PostgreSQL.
(14) **Shall** use React/TypeScript/Vite/Playwright for the frontend.
(15) **Shall** have Windows, Linux (Firefox), and mobile (Android and iOS) compatibility.
(16) **Will** have a collapsible sidebar on the store page for filters or other sub-options for the user.
(17) **Shall/Will** have a recommended section on the item page for the user to look at more products.
(18) **Shall/Will** an extensive testing environment for E2E and unit tests. 80% code coverage.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

No serious performance requirements are necessary for this project, unless it proves to be a serious problem. This section will be updated as necessary.

5.2 Safety Requirements

The only serious safety requirements would be associated with usage of a database, and the prevention of dropping a production database or anything equivalent in damage.

5.3 Security Requirements

The user’s password and credit card information **must** be encrypted to ensure protection if a breach were to occur in the database. The same may be true for the user’s address.