Instant Order eCommerce System Requirements

Langelihle Lucky Mhlongo

2024

Table of Contents

[Overview 2](#_Toc163257345)

[User stories 3](#_Toc163257346)

[Use Case Diagram 4](#_Toc163257347)

[Entity Relationship Diagram 5](#_Toc163257348)

[Class Diagram 6](#_Toc163257349)

[Product Backlog 7](#_Toc163257350)

[Technical Architecture 9](#_Toc163257351)

[Source Control 9](#_Toc163257352)

[Directory and naming services 9](#_Toc163257353)

[Assumptions and Dependencies 9](#_Toc163257354)

[Lessons Learned 10](#_Toc163257355)

# Overview

Instant Order is an e-commerce system that offers customers quality clothes at low cost. Customers can navigate through various categories in the system to view different products catered to by Instant Order. Admins can manage products and customer orders.

The system will have a clear navigation menu for easy browsing through product categories. The system will have a fast and accurate search functionality, allowing customers to find products quickly. Moreover, the system will have filters and sorting options to refine search results based on price, brand, or category criteria.

The system will have well-organised product listings displaying key information such as product name, price, and availability. The system will include product ratings and reviews to help users make informed purchase decisions. Customers will be able to add items to their cart without navigating away from the product page, and they can add products to their Wishlist.

The system will integrate user accounts to store payment and shipping details for seamless checkout. Moreover, it will consist of a streamlined checkout process with minimal steps to complete an order. Customers and Admins will be able to view orders and manage shipping addresses from their account dashboard.

# User stories

This section represents the user stories involved in the project.

As a customer

I want to be able to search for a product using category, brand, and price

So I can find the product I am looking for more quickly

As a customer

I want to be able to view products

So I can add them to my cart

As an admin

I want to be able to manage products

So I can allow customers to purchase them

As a customer

I want to be able to add products to my Wishlist

So I can view them later

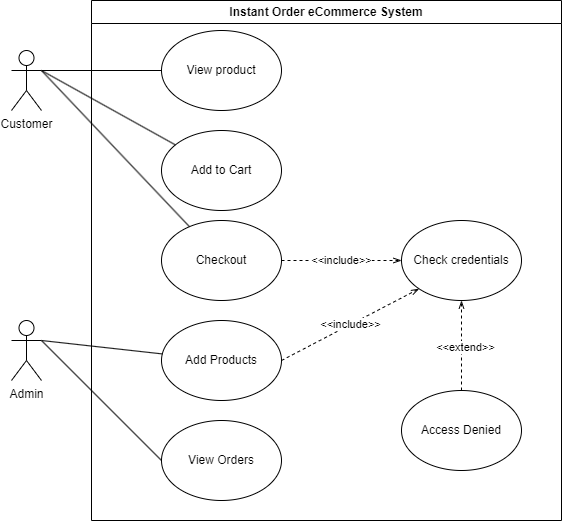
As a customer

I want to be able to checkout

So I can purchase the products in my cart

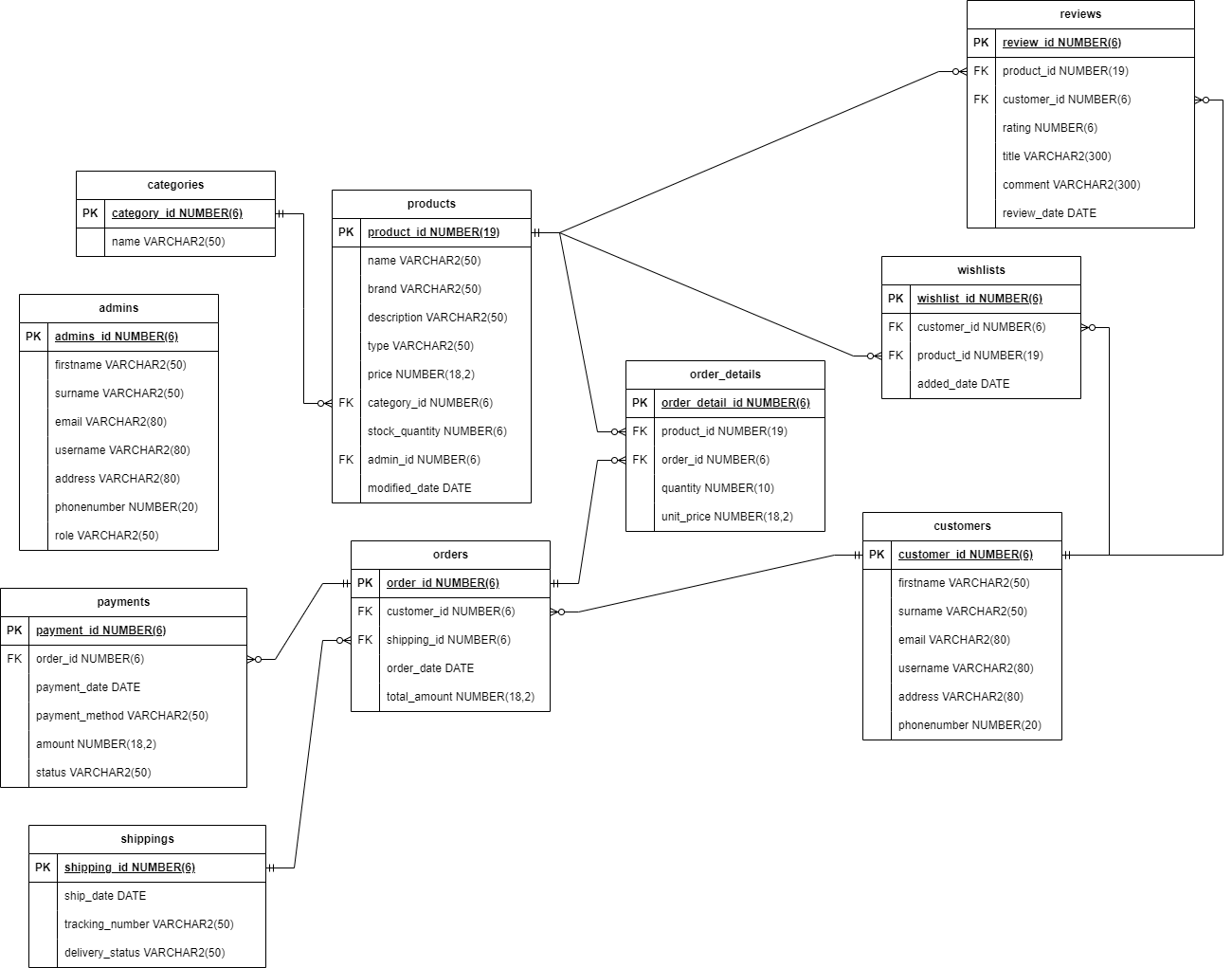
# Use Case Diagram

This section represents the use case diagram that describes the project’s use cases.



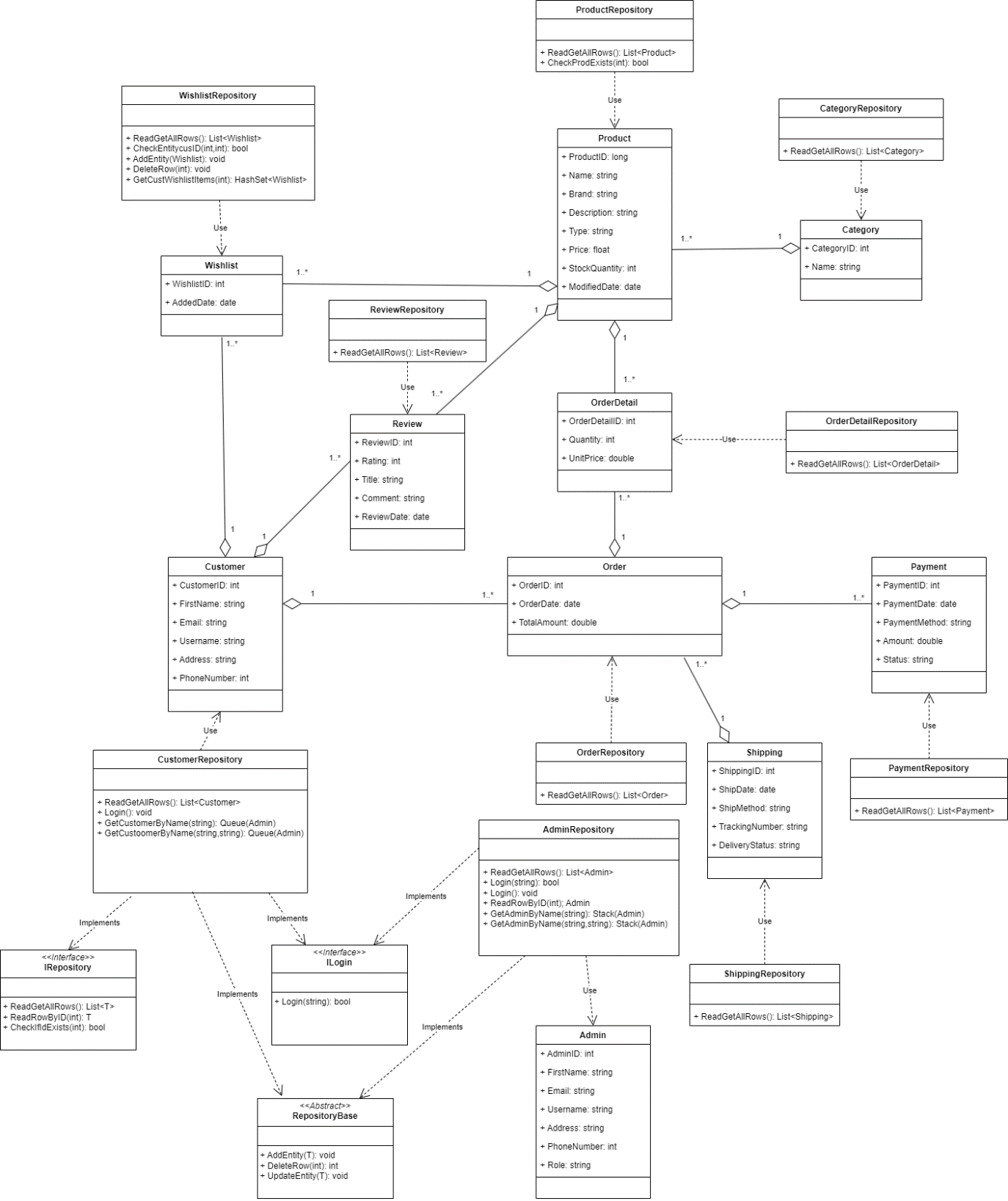
# Entity Relationship Diagram

This section represents the entity relationship diagram comprising all the entities involved in the project.



# Class Diagram

This section represents the class diagram comprising all the classes involved in the project.



# Product Backlog

This section represents the product backlog, which consists of the users, requirements, priority, and completion status.

The **Product Goal/objective** is to create full stack application using C# .NET as shown below:



**Sprint Backlog 1**



**Sprint Backlog 2**



**Sprint Backlog 3**



# Technical Architecture

The development environment for the system encompasses the tools, frameworks, and methodologies utilised throughout the software development lifecycle. This section outlines the primary components and configurations employed during development.

**Integrated Development Environment (IDE):**

Visual Studio 2022 - The primary IDE for developing, debugging, and deploying the system. Visual Studio provides a rich set of features, including code editing, version control integration, and project management tools.

**Programming Language and Framework:**

C# - The system is primarily developed using the C# programming language, leveraging its robust features, strong typing, and extensive .NET libraries.

.NET Framework 6.0 - The latest version of the .NET framework is utilised for building and running the system, offering enhanced performance, security, and language features.

**Development Methodology**

The development methodology adopted for the project is Agile, with a focus on iterative development, frequent releases, and continuous feedback loops. Specifically, the Scrum framework is utilised, employing sprint-based development cycles characterised by regular sprint planning, daily stand-ups, and sprint reviews.

In terms of development environment setup, the developer is provided with a high-performance workstation running Windows 11 or later. The workstation is equipped with ample RAM, CPU, and storage resources to support development tasks efficiently.

Visual Studio 2022 Community edition is the primary IDE utilised for development. It is configured with necessary components including the .NET Framework, ASP.NET, and Git integration. Additionally, the developer may integrate additional development tools and plugins into Visual Studio as per project requirements. These tools and plugins may include code analysers, debugging extensions, and productivity tools, enhancing the development process and productivity.

# Source Control

Git Repo: **eCommerce-Console-App**

# Directory and naming services

LLMeCommerce

# Assumptions and Dependencies

The assumptions and dependencies of this project are outlined in this section.

**Assumption:** Developers have access to high-performance workstations running Windows 11 or later. This assumption is based on the expectation that developers will have sufficient computing power, memory, and storage capacity to compile, test, and debug code efficiently. This assumption also implies that developers are familiar with the Windows operating system and can leverage its features and functionalities during development.

**Dependency:** Dependency on Visual Studio 2022 Community edition. The project relies on Visual Studio 2022 Community edition as the primary IDE for software development activities. This dependency is critical as Visual Studio provides a comprehensive suite of tools, libraries, and features necessary for building, debugging, and deploying applications using various programming languages, including C#. Furthermore, the dependency on Visual Studio 2022 Community edition offers a compelling combination of features, compatibility, accessibility, and community support, making it an ideal choice for facilitating efficient and cost-effective software development within the project.

# Lessons Learned

This section details the lessons learned throughout the project lifecycle.

Lesson Learned: Treat the requirements specification as a living document that evolves over time. Regularly review and update requirements based on feedback, lessons learned from previous projects, and changes in business or technical environments. Continuous improvement ensures that the requirements remain relevant and up-to-date throughout the project lifecycle.

Lesson Learned: Implement revision control and versioning mechanisms to manage changes to requirements effectively. Versioning ensures that the evolution of requirements is documented, and changes are tracked over time. It also facilitates collaboration among team members and allows for the rollback of changes if needed.

Lesson Learned: Break down requirements into smaller, manageable units to facilitate understanding and implementation. Instead of a single requirement for “customer cart”, the requirement should be split into separate requirements for “Add to cart”, “view cart items,” and “remove cart items.” This helps with prioritising tasks to complete and manage your time effectively.

Lesson Learned: Every class should focus on a single responsibility to ensure the code is easy to understand and manageable. For example, a customer class should concentrate on activities specific to a customer, such as getting customer details, updating details, etc. This also helps other developers on the team to understand your code easily.

Lesson Learned: Always test your code using debugging tools to see what is happening with the variables; this helps identify flaws in the code. Also, it helps with being familiar with code so that when you encounter an error, you know what to do to fix it.

Lesson Learned: XML comments help to make code clear and understandable by providing structured documentation that describes the purpose, behaviour, and usage of code elements. By documenting requirements using XML comments, developers create a formal reference for the behaviour and functionality of the software, aiding in communication and understanding among team members.

Lesson Learned: Test-Driven Development ensures that code meets requirements and functions correctly by writing tests before implementation. This approach validates that the code behaves as expected and helps catch bugs early in the development process.