

**Software Engineering and Testing. BSC Year 2, 2024/2025**

**(Assignment 3 - 20%)**

**Assessment 3: Design and Draft Implementation**

**Submitted by: RUBEN SOB B00166481**

**EMI HERDMAN B00165844**

**FRANKLIN OTWAMOAHENE ADDO B00159369**

**Submission date**

**23/04/2025**

**Declaration**

I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ordinary Degree in Computing in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated.

Author: Ruben Sob Dated: 21/03/2025

Author: Emi Herdman Dated: 21/03/2025

Author: Franklin Otwamoahene Addo Dated: 21/03/2025

**Table of Contents**

# Title: REFSolutions

# *Abstract / Executive Summary (200 Words max)*

This document outlines the methodologies, requirements, and progression of the project to develop an online electronics retail website. The website will cater to technophiles, offering a user-friendly interface with functionalities such as user registration and login, transaction history, product search, and a form for customers to sell their products. The website will be developed using PHP, HTML, and CSS, with a database to store product and user data. The project employs Object-Oriented Analysis and Design , UML diagrams, and Entity Relationship Diagrams (ERD) to ensure a robust and scalable design. The document also includes a detailed revision history, use case specifications, and class diagrams to illustrate the design decisions. The project is currently in progress, with regular updates and version control to ensure continuous improvement.

# Project Definitions

The purpose of the document is to outline the all the methodologies, requirements as well as the progression of the project completed so far. The project is about developing an online website that sells electronics that is computers, phones etc. The functional specification of the website includes, register and login pages, view history of transactions, search bar for searching products, as well as a form for customers to use to sell their products through the website. The website will have a very friendly user interface and also have a product page with almost all the items on sale. The website will have a database consisting of tables that will store data. The website will develop in the PHP programming language as well as HTML and CSS. Technophile (people interested in electronics) is the target market for the website.

# Document Revision

Rev. 1.0 initial version

Rev. 2.0 – 21/02/2025

Rev. 3.0 – 24/02/2025

Rev. 4.0 – 03/03/2025

Rev. 5.0 – 10/03/2025

Rev. 6.0 – 17/03/2025

Rev. 7.0 – 21/03/2025

This is the current status of the project. The project is done by the use of version control system that is every version is an upgrade of the previous version.

# Methodology

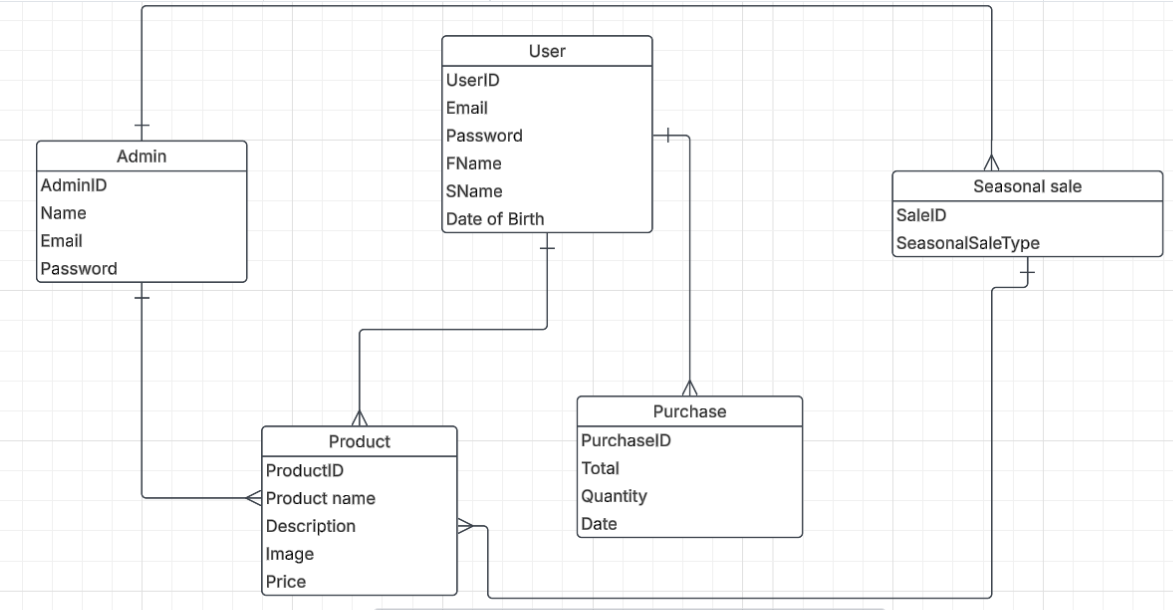
The project is designed with the use of ERD, class diagrams, use case specifications, the type of user interface designed for the website to incorporate. We created conceptual ERD and logical ERD as well as class diagram. The user interface for project was also designed with wireframe. The user-friendly interface template was chosen to ensure ease of navigation and a smooth user experience, aligning with the functional specifications of the project. The ERD will be used to design the database structure, showing entities, attributes, and relationships between them. The purposes of the classes  are used to encapsulate data and behaviour. A class diagram represents the structure of the system by showing classes, their attributes, methods, and relationships. Conceptual and logical ERD shown below.

**Conceptual ERD**

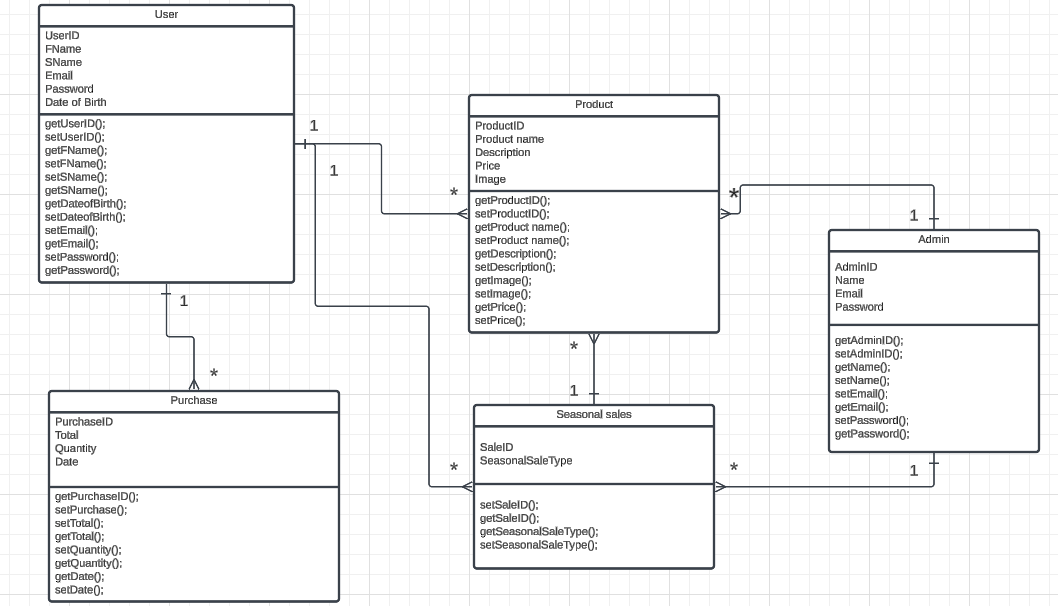
A diagram of a product

AI-generated content may be incorrect.

**Logical ERD**



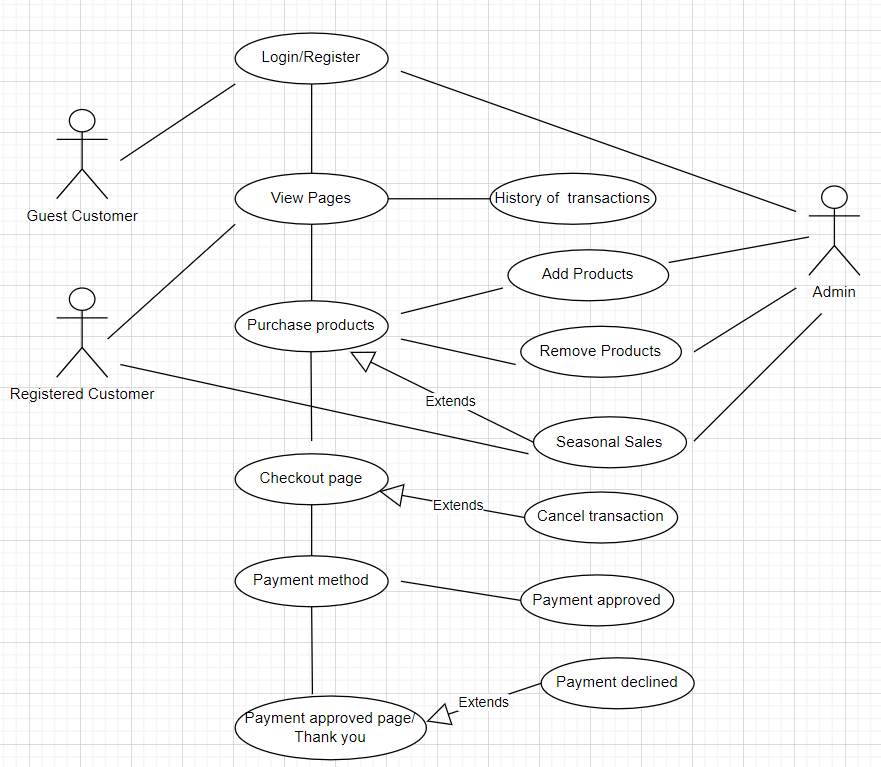
**Class Diagram**



1. **Requirements**

4.1 **Use Cases**

Use case was derived from the user requirements. Use cases describe the interactions between customers and the system such as registering, logging in, searching for products, and viewing transaction history. Seasonal sales was added to the use cases so that customers can products on sales during certain sales.

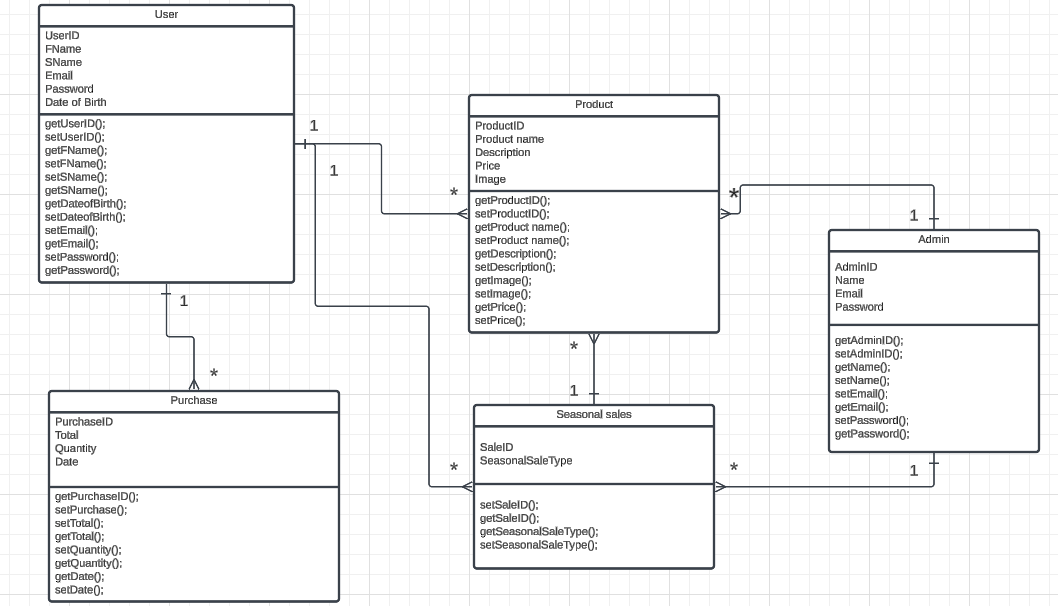
Below is the updated use cases. 

4.2 **Use Case Specifications**

Use Case Specifications provide detail descriptions of user requirements. These specifications are used to develop classes, attributes, methods, and database tables. The words in the bold format were used to develop the databases tables, attributes as well as entities.

|  |  |
| --- | --- |
| View Pages | * View content and **history** of purchases. * Displays **description** and **price** of product. * Display **images** of products. |
| Login/Register | * **Customers** will provide **email** and **password**. * **Customers** will provide **First name** and **Surname**. * **Authentication** of customer. * Administrator will use **username** and **password**. * **Authentication** of the **admin**. |
| Add products | * **Admin** can **add** products to the products page using the database connected. |
| Remove products | * **Admin** can **remove** products from the products pages. |
| History of transaction | * **History** of **purchases** of customers will be provided using the database. |
| Cancellation of transaction | * **Customer** can enter **transaction details** * System makes request to the database to search the exact transaction. * Database return the transaction details to the system.   • Customers must confirm the **transaction** • Database delete the transaction.   * Database send the refund to the system. * System sends **refund** information to the **customer**. |
| Seasonal sales | * **Products on sale** are added to the products page. * **Customers** can **view products** that are on sale. * **Admin** can **stock products** that are on sales on the website. |
| Checkout | * Display the **list of products** alongside the price. * Breaks the **entire cost** of the purchase to the **customer**. |

1. **Class Diagram**



The class diagram represents the system structure, defining key entities, their attributes, and relationships. Below are the design decisions for each class and their connections:

**User Class**

**Attributes**: Stores essential user details such as UserID, Email, Password, FName, SName and Date of Birth.

**Methods**: Includes getter and setter methods for FName, SName and DateofBirth and other atrributes.

**Relationship**:

One-to-Many relationship with **Purchase** – A user can make multiple purchases.

One-to-Many relationship with **Product** – A user can purchase multiple products.

**Admin Class**

**Attributes**: Stores AdminID, Name, Email, Password.

**Methods**: Includes getter and setter methods for password management and other atrributes.

**Relationship**:

One-to-Many relationship with **Product** – Admin manages multiple products.

One-to-Many relationship with **Seasonal Sales** – Admin oversees seasonal promotions.

**Product Class**

**Attributes**: Includes ProductID, Product Name, Description, Price and Image.

**Methods**: Provides getter and setter methods for Image and Price and other atrributes.

**Relationship**:

Many-to-One relationship with **Admin** – Products are managed by an admin.

Many-to-Many relationship with **Seasonal Sales** – Products can be part of multiple seasonal sales.

Many-to-One relationship with **Purchase** – A purchase contains multiple products.

**Purchase Class**

**Attributes**: Includes PurchaseID, Total, Quantity, and Date.

**Methods**: Provides getter and setter methods for date and other atrributes.

**Relationship**:

Many-to-One relationship with **User** – A purchase belongs to a single user.

Many-to-One relationship with **Product** – A purchase consists of multiple products.

**Seasonal Sales Class**

**Attributes**: Includes SaleID, SeasonalSaleType.

**Methods**: Provides getter and setter methods for SeasonalSaleType and SalesID.

**Relationship**:

One-to-Many relationship with **Admin** – Admins manage multiple seasonal sales.

Many-to-Many relationship with **Product** – Seasonal sales include multiple products.

**Entity Relationship Diagram Explanation**

The **ERD** follows similar relationships as the class diagram but focuses on database structuring:

**User & Purchase**: One user can make multiple purchases.

**Purchase & Product**: A purchase consists of multiple products, forming a many-to-many relationship.

**Admin & Product**: Each product is managed by a single admin, forming a one-to-many relationship.

**Admin & Seasonal Sales**: An admin manages multiple seasonal promotions.

**Product & Seasonal Sales**: A many-to-many relationship exists, as products can belong to multiple seasonal sales.

**Class Diagram**

The class diagram includes the following **object-oriented principles**:

**1. Generalization (Inheritance)**

* **Base Class:** A BaseUser class can be introduced for shared attributes like Email and Password.
* **Sub Classes:**

User (inherits from BaseUser) – Represents customers.

Admin (inherits from BaseUser) – Manages the system.

**2. Partial Composition (Aggregation)**

* **Relationship Between Product and Seasonal Sales**

A product can exist independently of seasonal sales but can be part of one.

**3. Full Composition**

* **Relationship Between User and Purchase**

A purchase cannot exist without a user (if a user is deleted, all related purchases must also be deleted).

# Conclusions

The project has progressed significantly, with the design phase nearing completion. The use of UML diagrams, OOAD, and ERD has provided a clear roadmap for development. Some changes to the original proposal were necessitated by design considerations, such as the addition of new use case and refinement of class relationships.

We recommend we continue to refine the design based on the feedback and ensure all the functional requirements are met before completing the implementation of the project.

Checklist: Is your document complete and correct?

*Content:*

* Does the design include all requirements from the customers’ needs
* Are you satisfied with all parts of the document?
* Do you believe all parts have been implemented?
* Have you explained your methodology and design choices?
* Have you clearly articulated your understanding of the purpose of all diagrams created ?
* What are these diagrams? Why you need them? How were they developed?
* Is each part of the document in agreement with all other parts?
* Does the design create a solution for the initial proposal?

*Completeness*:

* Are all the necessary components specified?
* Are the design specifications precise enough?
* Are all sections from the document template included – if changed, why?

*Clarity*:

* Is the design reasonable?
* Is the level of details for each design section appropriate?
* Is the design written in a language appropriate to the intended audience of software engineering teams?
* Are all items clear and unambiguous?