Software Engineering and Testing. BSC Year 2, 2024/2025
(Assignment 3 - 20%)

Assessment 3: Design and Draft Implementation

Submitted by: Cian Donnelly (B00160418), Sean Harmon Breen (B00164655), & Aaron Dignam (B00164709)

21/03/2025

Declaration

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

Author: Cian Donnelly, Sean Harmon, Aaron Dignam

Dated: 21/03/2025

Table of Contents

Title: ASC Motors

Executive Summary

This document provides an in-depth analysis of our project. We cover different areas of our website, such as its purpose, the non-functional requirements, the functional requirements, and the system's structure. Using UML diagrams and object-oriented programming principles, we improved our software's core components and the definition of our database's structure. Our project includes use case specifications, class diagrams, and entity relationship diagrams to allow all parties to understand the ideas, changes, and developments to the software. This document thoroughly describes the reasoning behind design choices and mentions how the website meets any of the user's requirements. We also share the adjustments that needed to be implemented as well as the necessary design to further enhance our project.

1. Project Definitions

- Purpose of document
  - This document is used to serve as a design and implementation draft for our software engineering project. Throughout the document, we include the website's functional requirements as well as specify any components.

- What is the project?
  - Our project is an e-commerce website used to buy modifications for cars. The webpage includes purchasing products, login, and a cart which allows us to remove or add items upon purchasing. We also included a recommended car list upon visiting the home page.

- Functional Specifications

  - Log In: The user should be able to log in using either email or username and a password. If the input is wrong and/or nothing is inputted into the field, then an error message should pop up.
  - Payment: The payment should allow the user to receive a purchase using currency to buy the product by enabling the user to enter their card details, name, address, and number.

- Cart: This area should allow the user to add or subtract items they wish or want to purchase into or from the cart, if they wish to continue purchasing the product a button will be available within the cart to proceed to the checkout where payment details are taken.
- Confirmation: This area should notify the user that their order was successfully purchased and will be delivered within a certain time frame.
- Product Information: This area should provide detailed information about each product as well as its price and an image of the product.
- Product Tracking: This area will allow the user to view the location of their purchase at the current time, the time of next departure (if abroad), and the estimated time of arrival at the user's location.
- Recommendations: This aspect should suggest products based on the activity of that specific user.

- Main components of the software system
  - Frontend: User interface (HTML, CSS, JavaScript).
  - Backend: Security and Layout (PHP).
  - Database: Storage for user accounts, products, and orders (MySQL).

2. Document Revision

version 1 - this is the first version of our website, it consists of only the basic PHP, mostly consisting of the home product and search pages. This was used and set up for laragon.

version 2 - in this version, we drafted our initial used case secs and outlined the functionality that we needed to include.

version 3 - here we made some updates to the website changing the search page into a bar fleshing out the products and home page adding a cart as well as making the use case specifications.

version 4 - in v4 we created the initial database, and implemented it using MySQL

version 5 - here both of our physical and Logical ERDs were created. completed the database adding any finishing touches. and used some PHP on the website for some minor testing through the database.

version 6 - as of most recently we have implemented CRUD operations through PHP using our ERD structure, connected through our database.

3. Methodology

System models – UML

- To demonstrate our project, we must use UML diagrams, this enables us to view the system structure, the allocated interactions, and the relationships between various objects.

Use of, and necessity of OOAD

- The use of OOAD techniques allows developers to go into the backend and see the allocated code and what it's used for as well as allowing developers to call in certain pieces that they want on another page. (E.G. Nav Bar).

Purpose of using classes / What is a class diagram?

- A class diagram represents the structure of a system, the relationships between various objects as well as the interactions they have with one another.
-

Static Versus Dynamic Case Diagrams?

- Static Diagrams: This diagram allows for the visibility of the structure of a system.
- Dynamic Diagrams: This diagram displays the interactions of a system over some time.

What is an ERD?

- This is an Entity Relationship Diagram, which represents the structure of a database, the attributes, as well as the relationships.

Purpose of using classes?

- The purpose of using classes is to aid in multiple aspects of your project, code reusability, encapsulation, and scalability. Instead of writing code for each user or page, you can create a class with code you intend to continuously use throughout your project. Encapsulation allows you to protect your data and prevent accidental modifications to your site, it also hides implementation details and only shows necessary functionality on your pages. Scalability allows you to easily expand your website, as you have each area of your code sectioned off into different classes, such as Products, Payment, and Cart. If you need to make edits to any of these sections or expand them to other areas of your site you can do this a lot more consistently with classes.

Volatile versus Persistent storage – Object Instances / Database?

- Volatile Storage: This is when data is stored as memory which gets lost after the system shuts down.
- Persistent Storage: This is when data is stored in a database and remains available after the system shuts down.

4. Requirements

4.1 Use Cases

- When creating a use case diagram, it should represent user interactions with the system. For example:

- Confirmation

- Log In

- Product Info

- Track Product

- Recommendations

4.2 Use Case Specifications

- Confirmation (Payment: Once the payment has gone through a message is prompted on the user's device acknowledging that the payment was successfully completed, User: The user receives an email or message notifying the user that payment was completed, Admin: Admin then confirms if the details upon purchase are authentic if so admin then delivers a message).

- Log In (User: The user is prompted to log in using an email/username and password, the user has to enter these into the designated area, once the user has this area filled out the login button is pressed and if all details are correct the user will be allowed into the webpage).

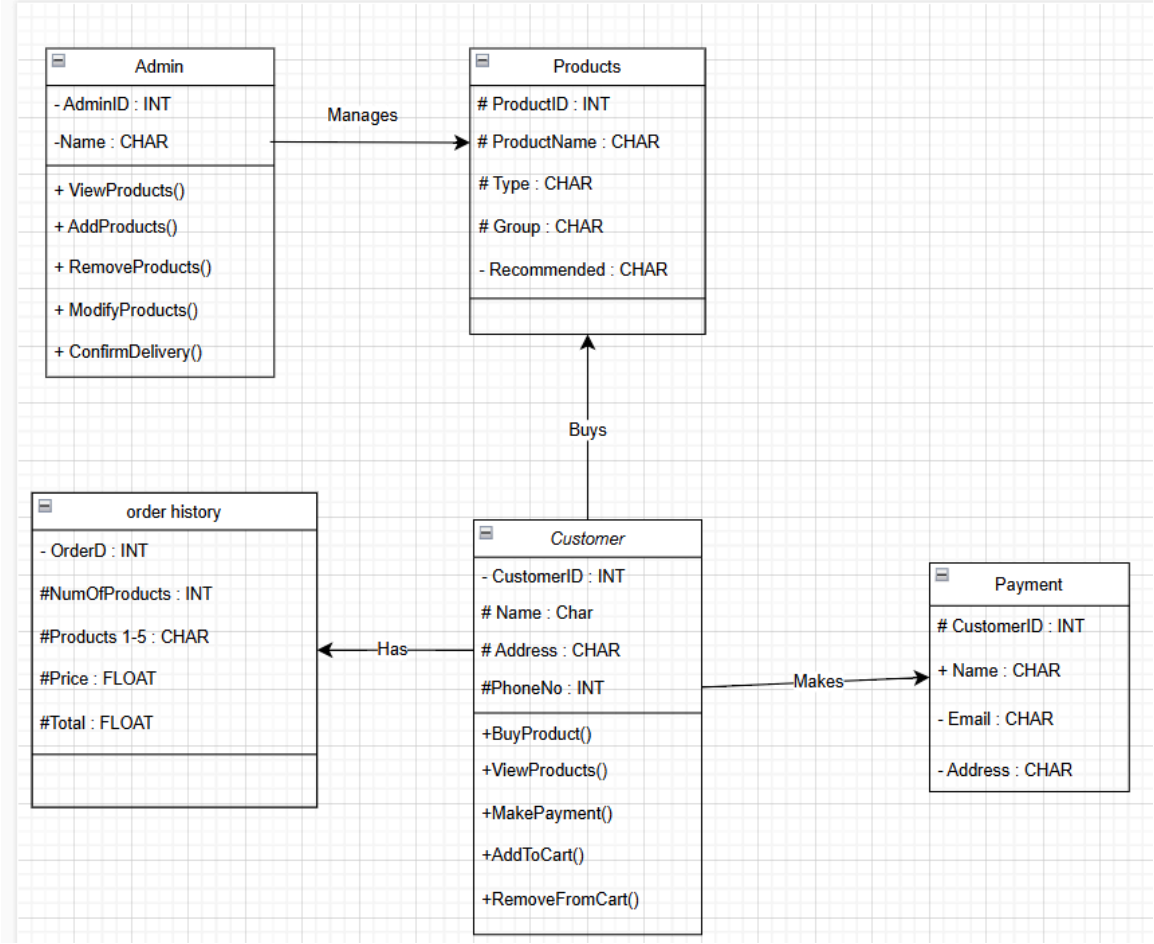
- Product Info (User: The user can view all the products at any pace as well as all of the details on a certain product, Admin: The Admin has to be updating information on the website all the time even if it's a minor detail).

- Track Product (User: The user can view where the product is as well as the estimated time upon arrival, Admin: Admin needs to be updating this area constantly on where the package is and the estimated time in which the delivery will arrive at a certain location).

- Recommendations (User: Receives product recommendations based on the user's behavior and on what the user has previously been purchasing or clicking, Admin: Can configure recommendation settings or highlight featured products).

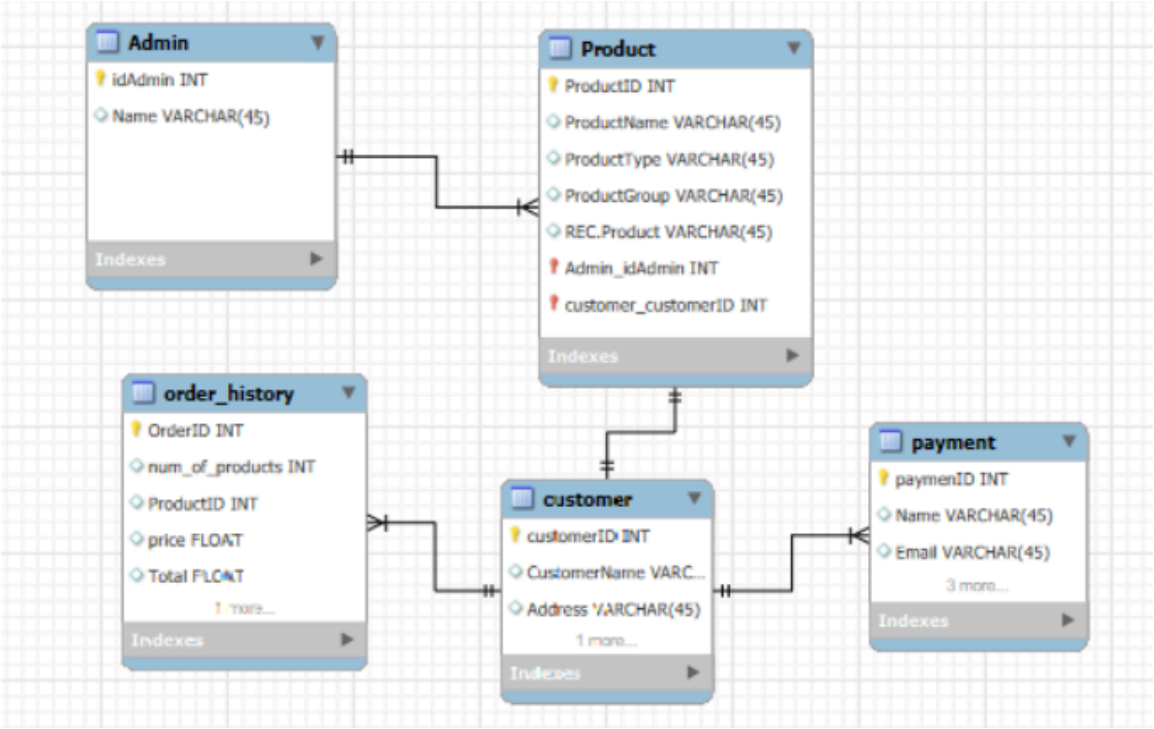
5. Case Diagrams

Class Diagram



Each class or table in this diagram corresponds to a major entity in the system of our website. The admin class includes methods to manage and update. AddProducts(), RemoveProducts(), ModifyProducts(). These set up the ability to adjust these aspects in the website. product classes can be adjusted and managed by admins as well as purchased by the customers. The customer class is handling our user actions such as BuyProduct(), AddToCart() and MakePayment(). allowing them to select products, payments, and track orders and history. the payment links to the customer class allowing the customer to purchase their selected products. the order history allows the customers to view their previous orders.

Entity Relationship Diagram



This entity relationship diagram shows our refined view of how our key data classes interact with our e-commerce car mods system. In the center of our system is our customer table, storing many key pieces of personnel information, our customer table links directly to our order history and payment tables ensuring the ability to make multiple payments through a one-to-many relationship. Similarly, a single customer can make multiple orders and multiple products. linking it into our order history, which included orderID and other pieces of information allowing us to track these key pieces from our orders. The product table is also linked to the admin and customer tables allowing admins to manage the products and customers to view and add products to the cart. overall this ERD diagram sets up a good structure for crud operations and support for our site while reinforcing the core features that we need included in our e-commerce site.
– Show all relationships, multiplicities,

6. Conclusion

At this point, we have completed the design and draft implementation, including implementing key functions that we have outlined throughout our project. We now have a functioning website including a functional homepage, a product page as well as a search bar cart, and a slideshow for recommended vehicles. These features were developed by us using php through laragon. as well as these we developed payment features and product management functional. as designed in our use case spec we created this website alongside a database and php classes.

using basic object-oriented principles and uml we designed our class and ERD diagrams to reflect the design and relationships between the components in our project. Our erd shows our relational integrity that supports core CRUD operations. Our class diagram included such things as encapsulation and composition of our project. all of the key actions of our users such as log-in product browsing and checkout.

our project compared to our first proposal made has had some changes in our attempt to improve the user experience, this was done making a more fleshed out home page than what we initially planned. We included such things as a product slideshow and integrated searching. These changes were made to incorporate recommendations or other specials. The project is now a stable website with a lot of smooth functionality. continuing from this we intend to polish down our project perfecting user experience and focus on reining handling of errors and validation.