Software Engineering and Testing. BSC Year 2, 2020/2021
(Assignment 3 -   20%)

Assessment 3: Design and Draft Implementation

Submitted by: Names, Student numbers

Matas Bagdonas - B00149694,

Patryk Kurzelewski - B00153346,

Enrico Ortenzi - B00156016

Submission date

19th March 2024

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: ____________________________ Dated: ______________

Author: ____________________________ Dated: ______________

Author: ____________________________ Dated: ______________

Table of Contents

Title: Bits und Volts Computers 3
1. Project Definitions 3
2. Document Revision 4
3. Methodology 5
4. Requirements 5
Use Case Specs Classes attributes and methods 6
    Customer Class : 6
    Item Class : 6
    Basket Class : 6
    Ticket Class : 6
    Checkout Class : 6
    Filter Class : 6
5. Conclusions 9

Title: Bits und Volts Computers

This document aims to illustrate a clear and visual understanding of how the implementation of our classes will form the final software requested by the client. More specifically, the ERD diagram and Class-Package diagrams will illustrate the different dependencies that are present among our classes, demonstrating which classes are independent or dependent on each other.

We have utilized one generalization in our system, where a user class branches out into the Customer class and Admin class. Moreover, considering our use of classes, our team has chosen a functionality-driven approach to the client's request, which means there will be greater emphasis on the functionalities and coordination between web pages, rather than a very high, appealing graphical interface. This small compromise will provide the client with a functional software and the diagrams in this document highlight a chronological order of how the customer will interact with the system.

1. Project Definitions

- Purpose of document

This document seeks to provide a clear and visual representation of how our classes will operate collaboratively to create the software requested by the client. It will also outline the method(s) used to gain named software. Specifically, the ERD diagram, Class-Package diagrams and Use Case specifications will show the various connections between our classes, indicating which ones operate independently and which ones rely on each other.

- What is the project?

The project is to provide a functional environment in the form of a website from which the user can purchase computer system components. When it comes to the technological support services, this project aims to give the opportunity to customers to get their devices fixed or buy a brand-new device.

- Functional Specifications

System models – We will make use of an Entity Relationship Model, a Class-Package Diagram, and a Use Case specification to provide a visual representation of the software's structure

Development language – HTML, PHP, CSS, JAVASCRIPT.

Persistent storage – We will make use of persistent/ non-volatile storage in the form of MySQL, as customers who wish to make orders will need an account that will need to be stored in a database. Furthermore, a session id will be provided to customers who have already logged in to ease the checkout process.
Interface & Software / Hardware API

- Main components of the software system,

The main components of the system are a product browsing system that will be in the form of a scrollable catalogue that the customer can navigate. Secondly, there will be an account system that will accommodate users who wish to save products to their temporary basket. More importantly, a logged-in user will then be able to make an order for any selected product in the basket. Lastly, the ability to make customer requests if anything were to fail with the product or during the delivery of the item/ items.

2. Document Revision

Rev. 1.0 18/03/2024 – Final Version

3. Methodology

For the outlined software, we must use an Object Orientated Analysis and Design (OOAD) approach the reason for this is because OOAD is better for the complex and evolving nature of a computer parts shop. It provides a more flexible, adaptable, and scalable framework that aligns well with the inherent characteristics of object-oriented design and development.

Purpose of using classes / What is a class diagram? - A class diagram is a type of UML, the reason for its use is to be able to Create/Read/Update and Delete classes. This makes managing the outcome of our classes modular and reusable as class diagrams are just temporary storage.

Static Vs. Dynamic Case Diagrams - Static class diagrams represent the static structure of a system, including classes and their relationships. Dynamic class diagrams illustrate the dynamic behaviour of the system, showing interactions between objects during runtime.

What is an ERD? - An ERD is an Entity Relationship Diagram; it outlines the relationships between different entities (e.g. Customer to Orders). It also visually represents these relationships in a flow like chart, with lines Connecting them representing their cardinalities.

Volatile versus Persistent storage – Object Instances / Database? - The use of classes means we can have objects which will hold the data temporarily. And the use of a database will ensure that whatever data we need to keep and store permanently will be stored and ready to be used.

4. Requirements

4.1 Use Cases

1. Browse Items

2. Add to Basket

3.Login

4.Register

5.Customer Support Ticket

6.Checkout

7.Filter Items

4.2 Use Case Specifications

Use Case Specs Classes attributes and methods

Classes:

Customer Class:

- Attributes: email address, password, name, surname, address
- Methods: register(), login()

Item Class :

- Attributes: name, price, specifications, availability
- Methods: getItemDetails(), addToBasket()

Basket Class :

- Attributes: items, totalAmount
- Methods: addItems(), calculateTotal()

Ticket Class :

- Attributes: customer concern, status
- Methods: createTicket(), viewTicket(), resolveTicket()

Checkout Class :

- Methods: processCheckout(), estimateDeliveryDate()

Filter Class :

- Methods: filterByCategory(), filterByPrice(), filterByAvailability()

Database Tables :

Customer Table :

- customer_id (Primary Key), email, password, name, surname, address

Items Table :

- item_id (Primary Key), name, price, specifications, availability

Basket Table :

- Fields: basket_id (Primary Key), customer_id (Foreign Key), item_id (Foreign Key), quantity

Tickets Table :

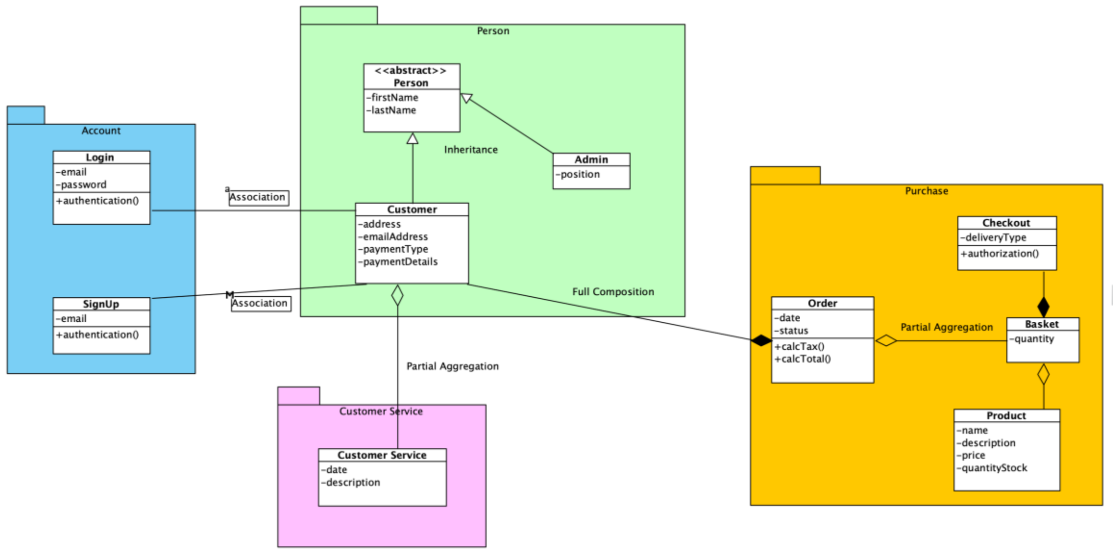
- Fields: ticket_id (Primary Key), customer_id (Foreign Key), concern, status

Orders Table :

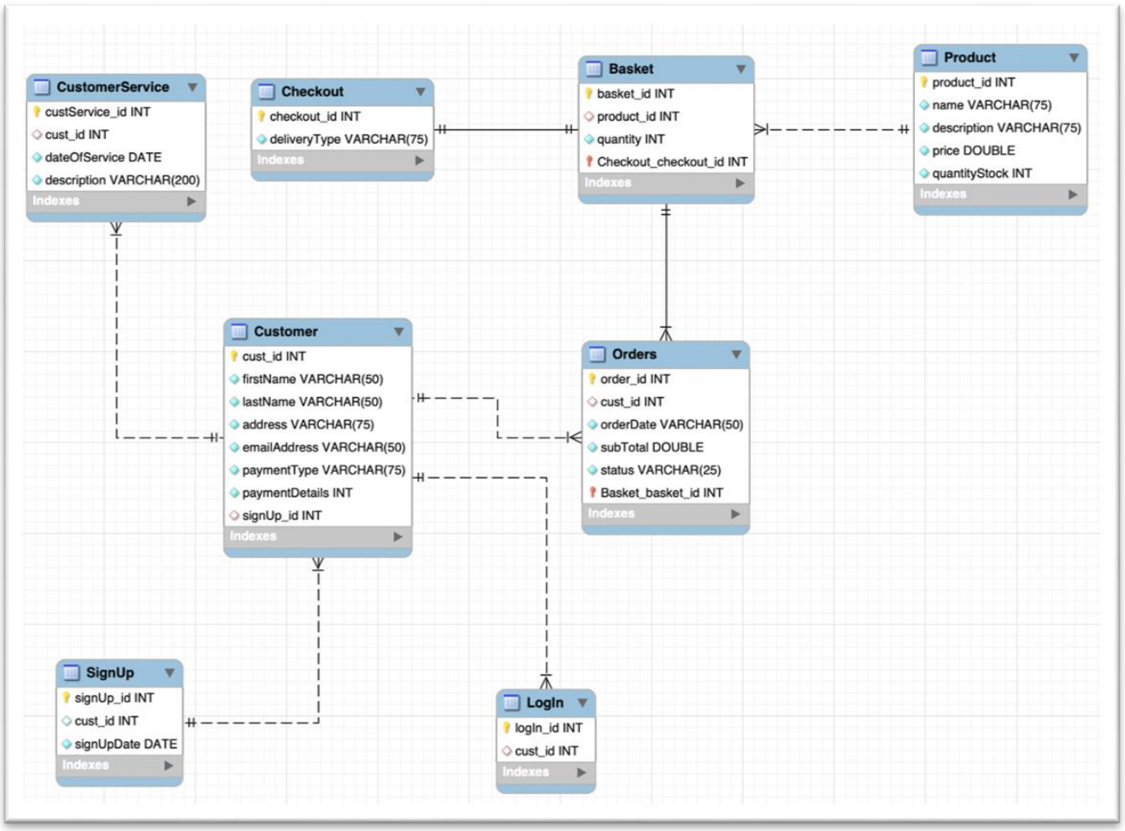
- Fields: order_id (Primary Key), customer_id (Foreign Key), total_amount, card_details, order_date, delivery_date

4. Case Diagrams

Class Diagram – relationships, multiplicities, associations.



The Class-Package diagram above includes 4 separate packages. We have several instances of partial and full aggregations and one example of inheritance. We have a Person package with an abstract superclass called Person from which the Customer and Admin class inherit the 'firstName' and 'lastName' attributes. The LogIn and SignUp entities in the Customer package are dependant entities on the Customer entity. While the Customer entity is independant, the Account package cannot live by itself.



Here is an Entity Relationship Diagram. The Customer Entity is crucial as it links to our system. Customers interact with the software by browsing and selecting products, which are promptly added to their basket. Upon proceeding to checkout and payment, an order is placed, associating the customer with the Order Entity. This allows users to view their basket contents during the checkout process . This is why the diagram is designed in the current manner.

5. Conclusions

In conclusion, the modern version of the software design is very similar to the one we proposed in the original, but due to a couple formalities we decided to change the way it will structured by adding 2 more classes to our Customer entity/class (Admin and User.) these will facilitate whether the person who is accessing the website wants to be authenticated as an administrator to be able to edit and manage data. Or the person who is accessing the website to use the software to make a purchase this helps us manage access to the software if technical issues arise.

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?