Software Engineering and Testing. BSC Year 2, 2024

(Assignment 3 - 20%)

Assessment 3: Design and Draft Implementation

Submitted by:

Leonardo Rodrigues B00155567

Nikita Ciuciu B00152278

Julia Udvari B00153580

18/03/2024

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: Leonardo Rodrigues Dated: 18/03/2024

Author: Nikita Ciuciu Dated: 18/03/2024

Author: Julia Udvari Dated: 18/03/2024

Table of Contents

Title: DUI-Livery

DUI-livery is an innovative initiative aimed at reducing driving under the influence (DUI) incidents by offering an online platform for convenient and safe alcohol purchase. In response to the alarming statistic of 184 deaths on Irish roads in 2023 due to DUI, this project provides a critical solution that promotes responsible drinking and enhances road safety. The platform facilitates the purchase and delivery of alcoholic beverages, targeting individuals who prefer not to drive to off-license stores. It includes essential components such as a login, authentication, ordering, delivery tracking, and search systems, catering to customers and drivers alike. Built on PHP, MySQL, Apache, and hosted in the Laragon environment, DUI-livery stands as a robust, secure, and user-friendly platform. This venture not only serves as a preventive measure against DUI but also offers convenience to consumers and flexible earning opportunities for drivers, embodying a comprehensive approach to tackling a significant societal challenge.

1. Project Definitions

- **Purpose of Document:** This document outlines the design and draft implementation plan for DUI-livery, a project aimed at reducing DUI incidents by providing an online platform for alcohol purchase and delivery. It details the design choices, methodologies, system models, and the initial steps towards implementation.

- **What is the Project:** DUI-livery is a web-based platform designed to prevent DUI by offering a service that allows individuals to purchase alcoholic beverages

online, thus eliminating the need to drive to off-license stores. It emphasizes safety, convenience, and responsible drinking.

- **Functional Specifications:** The project will include features such as user authentication, product browsing, order placement and delivery. These functions are aimed at ensuring a seamless user experience while promoting safety and convenience.

- **Main Components of the Software System:** Key components include a login and authentication system, ordering system, delivery tracking system, search functionality, and database management for user and order information.

2. Document Revision

Rev. 1.0 date – initial version

3. Methodology

System Models – UML:

Unified Modelling Language (UML) is a standardized modelling language consisting of an integrated set of diagrams, designed to specify, visualize, construct, and document the artifacts of a software system. UML helps in defining the structure and behaviour of systems, facilitating clear communication among developers and guiding the development process.

Use of and Necessity of OOAD:

Object-Oriented Analysis and Design (OOAD) is a structured method for analysing and designing a system by viewing it as a group of interacting objects, each with its own responsibilities. OOAD is necessary for creating robust, scalable, and maintainable software. It allows developers to break down a system into manageable parts, making it easier to understand, develop, and test.

Purpose of Using Classes / What is a Class Diagram:

Classes are the blueprint of objects in object-oriented programming; they encapsulate data for the object and methods to manipulate that data. A class diagram, a component of UML, visually represents the classes, their attributes, methods, and the relationships among classes. It serves as a foundation for developing software, ensuring structure and consistency.

Static Versus Dynamic Case Diagrams:

Static diagrams, such as class diagrams and ER diagrams, represent the structure of a system. They show how each component is related but don’t indicate how or when they interact. Dynamic diagrams, like sequence diagrams and activity diagrams, illustrate how objects interact and the flow of control among them over time, providing insight into the system's behaviour.

What is an ERD:

An Entity Relationship Diagram (ERD) is a graphical representation of entities and their relationships to each other, typically used in database design. It helps in defining the data elements and their structures, offering a clear schema for how data is stored, connected, and accessed.

Purpose of Using Classes:

The use of classes allows for abstraction, encapsulation and inheritance in software development. Classes help organize and structure the software, making the code more modular, reusable, and easier to maintain and scale.

Volatile versus Persistent Storage – Object Instances / Database:

Volatile storage refers to temporary storage that is lost when the program ends or the device is turned off, such as RAM. Persistent storage refers to data saved on a long-term basis, like databases or disk storage, which remains accessible across different sessions and restarts.

User Interface Template Chosen and How It Can Aid in Executing the Functional Specification of the Project:

The selection of a user interface (UI) template is guided by the project's functional specifications to ensure that the UI meets user needs and enhances the overall user experience. A well-chosen UI template can aid in executing the functional specifications by providing a clear, intuitive, and responsive design, making it easier for users to navigate the platform and perform desired actions, such as browsing products, placing orders, and tracking deliveries. This alignment between the UI and functional specifications is crucial for the success of DUI-livery, as it directly impacts user satisfaction and engagement.

4. Requirements

4.1 Use Cases

1. **Place Order**
   Actors: Customer
   Description: Allows customers to browse the catalogue, select alcoholic beverages, and place an order for home delivery.

2. **Order Delivery**
   Actors: Driver
   Description: Enables drivers to view pending deliveries, choose an order to deliver, and update the order status upon delivery.

4.2 Use Case Specifications

Place Order:

Preconditions: Customer must be registered and logged in.

Main Flow:

- Customer logs into their account.
- Customer browses the product catalogue.
- Customer adds selected items to their cart.
- Customer confirms order details and submits the order.
- Postconditions: Order is recorded in the system and made available to drivers for delivery.

Order Delivery:

Preconditions: Driver must be registered, logged in, and verified.

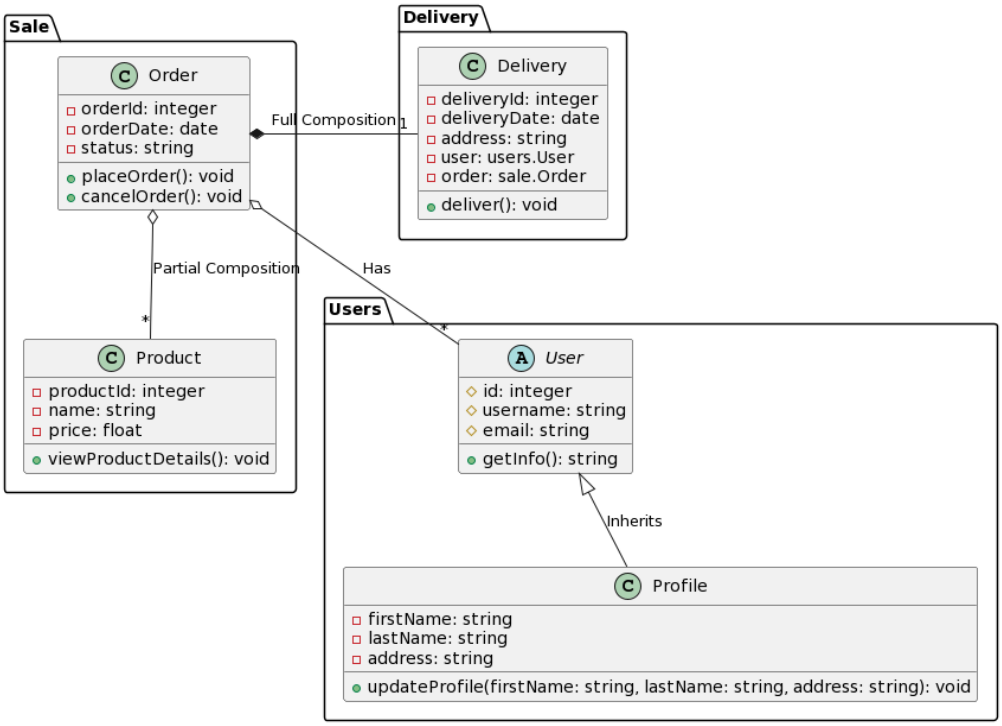
Main Flow:

- Driver logs into their account.
- Driver views a list of available deliveries.
- Driver selects an order to deliver.
- Driver updates the order status upon completion.
- Postconditions: Order status is updated to "delivered," and customer is notified.

To translate use case specifications into system design for the DUI-livery project, we first extracted key functionalities and actors from the use cases, such as customers placing orders and browsing products. This guided the identification of essential classes like Customer, Order, and Product, each with specific attributes (e.g. customerID, orderID, productPrice) and methods (e.g., placeOrder(), addToCart()). These classes were directly linked to database table designs, ensuring attributes matched table columns for data persistence. For instance, the Customer class attributes informed the structure of a customer’s database table. This methodical approach from use case analysis to class and database design ensured the software's architecture was robustly mapped out, aligning closely with functional requirements and user interactions defined in the use case specifications.

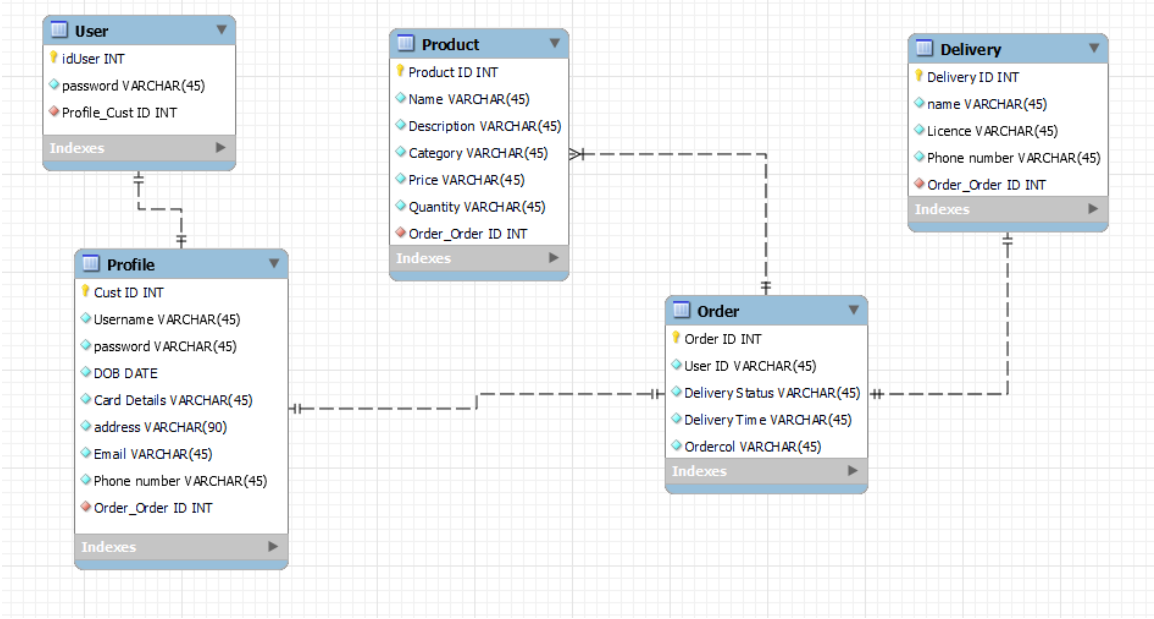
5. Case Diagrams

5.1 Class Diagram:



The class diagram for the DUI-livery system is designed for modularity and scalability. Order is central, holding Product items, implying that products are not standalone but parts of orders. The Delivery class is linked to Order, suggesting that orders can be delivered. User is a generic class, extended by Profile for additional personal details, allowing user information to be expanded as needed. These choices ensure a clear, maintainable structure where sales, user management, and delivery components are distinct yet interrelated.

5.2 Entity Relationship Diagram:



The ERD for DUI-livery organizes user information, product details, orders, and deliveries into interconnected entities for efficient data management. Users are linked to profiles for personal details, products to orders for tracking purchases, and orders to deliveries for logistics. This schema ensures streamlined operations, from user registration to product delivery, within a coherent database structure.

6. Conclusions

Conclusions and Progress:

The project has successfully implemented a critical functionality — the order creation system. This is a significant milestone as it forms the backbone of the service. However, not all parts of the project have been completed, indicating that while progress has been made, the project is still missing some components.

Recommendations:

Given the delays caused by an unforeseen snowstorm, it is recommended to reassess the project timeline and adjust deadlines accordingly to accommodate for lost time. It's also prudent to factor in buffer times for future unforeseen events to ensure more realistic time management.

Changes Necessitated by the Design:

The introduction of a new table to explain order details suggests a refinement in the database schema to better capture and manage transaction specifics. This reflects a responsive approach to design, where adjustments are made to enhance system clarity and functionality.

7. References

Software Development Glossary: 88 Essential Terms | Clutch.co. (2023, March 29). https://clutch.co/resources/software-development-glossary-88-essential-terms

What is an Entity Relationship Diagram (ERD)? (n.d.). Lucidchart. https://www.lucidchart.com/pages/er-diagrams

OOAD - UML Analysis Model. (n.d.). https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_uml_analysis_model.htm#:~:text=The%20Unified%20Modeling%20Language%20(UML,of%20an%20object%2Doriented%20system.

Class diagram. (2024, March 9). Wikipedia. https://en.wikipedia.org/wiki/Class_diagram#:~:text=In%20software%20engineering%2C%20a%20class,and%20the%20relationships%20among%20objects.

What Is Persistent Memory? (2023, November 30). Pure Storage. https://www.purestorage.com/fr/knowledge/what-is-persistent-memory.html#:~:text=Volatile%20storage%2C%20represented%20by%20RAM,where%20data%20reliability%20is%20paramount.