

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M'Hamed BOUGARA – Boumerdes



Institute of Electrical and Electronic Engineering

Department of Electronic

Project Report Presented in Partial Fulfilment of
the Requirements of the Degree of

‘Master’

In Computer Engineering

Title:

**Web-based application for cash-on-delivery
orders' management system**

Presented By:

- **Mr. Mahfoudi Yahia Chamesse Eddine**

Supervisor:

Dr. Touzout W

Registration Number:/2023

Abstract

The project's goal is to build a dynamic web application specifically designed for the order management system in a cash-on-delivery e-commerce store. The developed program aims to automate order management, delivery tracking, and customer interaction management to streamline administration and staff operations. The project focuses on evaluating the current practices and identifying prospective areas for progress in order to improve the order fulfillment process. The application and database structure were designed using the Unified Modeling Language (UML). Moreover, the application is designed using the Spring Boot framework from Java for the implementation phase since it provides a reliable and adaptable way to build standalone apps. As a result of its tried-and-true architecture and potent performance, PostgreSQL was chosen as the database management system (DBMS). The Angular framework is chosen for its capacity to produce dynamic and effective web applications; thus, it was used to build the user interface. This combination technology aims to produce a potent, effective, and user-friendly order management system.

Keywords: Web application, SPA, Spring boot, Angular, HTML, CSS, Java, JavaScript and UML.

Dedication

I dedicate this work to my beloved parents, my brothers and my sisters, to all my friends, and to all people who were there for me with help, advice and best wishes. Thank you all for your support.

Acknowledgements

First and foremost, I thank ALLAH, for helping me to finish this modest work. It is my belief in him that helped me persevere at times when it seemed impossible to go on. I would like to thank my supervisor Dr. TOUZOUT who gave me the opportunity to work on this project. My deepest thanks to my friend IMAD who helped me with his knowledge and guides. And gratitude to all who supported me through my educational journey in the institute of electrical and electronics engineering of the University M'Hamed BOUGARA – Boumerdes.

Table of content

| | |
|--|-----|
| Abstract | I |
| Dedication | II |
| Acknowledgements | III |
| Table of content..... | IV |
| List of tables | VI |
| List of figures..... | VII |
| General Introduction..... | 1 |
| Chapter 1 Project Overview and Web Applications..... | 4 |
| 1.1 Introduction..... | 4 |
| 1.2 Subject presentation | 4 |
| 1.2.1 Problem statement | 5 |
| 1.2.2 Objectives..... | 5 |
| 1.3 Web application..... | 6 |
| 1.3.1 Definition | 6 |
| 1.3.2 Single-Page Application..... | 6 |
| 1.3.3 Web client and Web server | 7 |
| 1.4 HTTP..... | 7 |
| 1.5 URL | 7 |
| 1.6 RESTful API..... | 8 |
| 1.7 Conclusion | 8 |
| Chapter 2 Tools and technologies | 10 |
| 2.1 Introduction..... | 10 |
| 2.2 Development tools | 10 |
| 2.2.1 IntelliJ IDEA | 10 |
| 2.2.2 Visual studio code..... | 10 |
| 2.2.3 Spring framework and Spring boot..... | 11 |
| 2.2.4 Angular | 11 |
| 2.2.5 Web browser..... | 11 |
| 2.3 Programming languages..... | 11 |
| 2.3.1 Java..... | 12 |
| 2.3.2 HTML | 12 |
| 2.3.3 CSS | 12 |
| 2.3.4 JavaScript | 13 |
| 2.3.5 TypeScript | 13 |
| 2.4 Security | 13 |
| 2.4.1 Bcrypt..... | 13 |
| 2.4.2 JSON Web Token..... | 14 |
| 2.5 Conclusion | 15 |

| | | |
|--------------------------------|---|-----------|
| Chapter 3 | System Design..... | 17 |
| 3.1 | Introduction..... | 17 |
| 3.2 | Design Requirements | 17 |
| 3.2.1 | Modeling language | 17 |
| 3.2.2 | Unified modeling language (UML) | 17 |
| 3.3 | Use Case Diagram | 18 |
| 3.3.1 | Definition: | 18 |
| 3.3.2 | Relationships | 18 |
| 3.4 | Application modeling..... | 19 |
| 3.4.1 | Actors | 19 |
| 3.4.2 | Use cases..... | 20 |
| 3.4.3 | Application's use case diagrams | 22 |
| 3.4.4 | Textual description of use cases | 24 |
| 3.5 | Introduction to database | 27 |
| 3.5.1 | Definition | 27 |
| 3.5.2 | Non-relational model | 27 |
| 3.5.3 | Relational model..... | 28 |
| 3.5.4 | PostgreSQL..... | 28 |
| 3.5.5 | Database Schema..... | 28 |
| 3.5.6 | Application entity relationship diagram | 32 |
| 3.6 | Conclusion | 34 |
| Chapter 4 | System Implementation | 36 |
| 4.1 | Introduction..... | 36 |
| 4.2 | Interfacing with the application..... | 36 |
| 4.2.1 | Login Interface | 36 |
| 4.2.2 | Dashboard interfaces for each user | 37 |
| 4.2.3 | List of users' interface and add new user interface | 38 |
| 4.2.4 | Add new product interface..... | 40 |
| 4.2.5 | List of Products interface | 40 |
| 4.2.6 | List of delivery agencies interface..... | 41 |
| 4.2.7 | Add new delivery agency interface..... | 41 |
| 4.2.8 | Add orders interface | 42 |
| 4.2.9 | In confirmation tab interface | 42 |
| 4.2.10 | In preparation tab interface | 44 |
| 4.2.11 | In dispatch tab interface..... | 44 |
| 4.2.12 | In delivery tab interface | 45 |
| 4.2.13 | Delivered and Returned tabs interfaces | 45 |
| 4.2.14 | Cancelled tab interface | 46 |
| 4.3 | Conclusion | 46 |
| General Conclusion..... | | 47 |
| Future Work..... | | 47 |
| Bibliography | | 48 |

List of tables

| | |
|--|----|
| Table 3.1 Summary of use case diagram's relationships and used arrows. [24] | 19 |
| Table 3.2 Use cases of each actor | 21 |
| Table 3.3 Textual description for authentication use case | 24 |
| Table 3.4 Textual description for manage users use case | 25 |
| Table 3.5 Textual description of manage orders use case | 25 |
| Table 3.6 Textual description for manage products use case | 26 |
| Table 3.7 Textual description of manage delivery agencies use case | 26 |
| Table 3.8 used arrows in ER diagram of this system | 32 |

List of figures

| | |
|---|-----------|
| Figure 1.1 Order fulfillment process | 4 |
| Figure 1.2: URL parts [6] | 8 |
| | |
| Figure 2.1 Encoded JSON web token [18] | 14 |
| | |
| Figure 3.1 Admin use case diagram..... | 22 |
| Figure 3.2 Phone agent use case diagram | 23 |
| Figure 3.3 Delivery guy use case diagram..... | 23 |
| Figure 3.4 Delivery agency system API use case diagram..... | 24 |
| Figure 3.5 ER diagram of this system's database..... | 33 |
| | |
| Figure 4.1 Authentication interface..... | 36 |
| Figure 4.2 Admin's initial dashboard interface..... | 37 |
| Figure 4.3 Phone agent's initial dashboard interface | 38 |
| Figure 4.4 Delivery guy's initial dashboard interface..... | 38 |
| Figure 4.5 Users' list interface..... | 39 |
| Figure 4.6 Add new user interface..... | 39 |
| Figure 4.7 Add new product interface..... | 40 |
| Figure 4.8 List of products interface | 40 |
| Figure 4.9 List of delivery agencies interface | 41 |
| Figure 4.10 Add new agency interface | 41 |
| Figure 4.11 Add orders interface | 42 |
| Figure 4.12 In confirmation tab interface..... | 43 |
| Figure 4.13 Expanded row from in confirmation tab interface..... | 43 |
| Figure 4.14 In preparation tab interface..... | 44 |
| Figure 4.15 In dispatch tab interface..... | 44 |
| Figure 4.16 In delivery tab interface | 45 |
| Figure 4.17 Delivered tab interface | 45 |
| Figure 4.18 Returned Tab interface | 46 |
| Figure 4.19 Cancelled tab interface | 46 |

List of abbreviations

- **SPA:** Single Page Application.
- **API:** Application Programming Interface.
- **RDBMS:** Relational Database Management System.
- **CSS:** Cascading Style Sheets.
- **HTML:** Hypertext Markup Language.
- **HTTP:** Hypertext Transfer Protocol.
- **JS:** JavaScript.
- **JSON:** JavaScript Object Notation.
- **JWT:** JSON Web Token.
- **SQL:** Structured Query Language.
- **UML:** Unified Modeling Language.
- **URL:** Uniform Resource Locator.
- **VS code:** Visual Studio Code.

General Introduction

The Internet has elevated the computer and communication industries to an extraordinary level. The invention of telephones, radios, and computers made this unique integration of capabilities possible.

A web application is an application that consumers can access over the Internet. The application is accessible from any Internet-connected computer using a standard web browser. Alternatively, it is a software system that provides an interface via a web browser. A web application plays a crucial function in all fields, including education, healthcare, and libraries. A web application can provide ease in every discipline because it saves clients' time. When everyone has access to the internet, a web application is extremely beneficial and practical for customers.

Operating a cash-on-delivery e-commerce store poses some challenges. It requires a robust logistics and payment collection system to manage the cash transactions and ensure timely and accurate delivery. It also carries a higher risk of returns or cancellations since customers have the option to refuse payment at the time of delivery.

The objective of this project is to design and implement an interactive, dependable, and user-friendly web application to facilitate and improve the order fulfillment process of cash-on-delivery e-commerce stores.

This report explains how a well-implemented web application can facilitate a reduction in the workload of employees handling online store orders.

Our report is divided into four major sections:

The first chapter describes the subject and problem statement, defines the project's objectives, and provides an overview of web applications. **The second chapter** introduces the tools and

technologies used in the project. **The third chapter** is dedicated to the design. It integrates segments of our development process using the UML modeling language. **The fourth chapter** describes the project's implementation and results by describing the various user interfaces of our application's front-end. Finally, this report concludes with a **general conclusion** as well as with recommendations for future work.

CHAPTER 1

PROJECT OVERVIEW AND WEB APPLICATIONS

1.1 Introduction

This chapter is about the project problem identification. Then, we talk about the intended outcomes and the steps to take in order to provide a better solution to the problem at hand. Given that the focus of the project is on the design and implementation of a web application, we conclude by providing an introduction to web applications and other related web concepts.

1.2 Subject presentation

Our goal is to create and implement an effective web application that will allow cash-on-delivery e-commerce companies to handle their orders, leads, and order fulfillment by automating order confirmation and monitoring the process.

The figure below shows the order fulfillment process and the statuses that an order may go through:

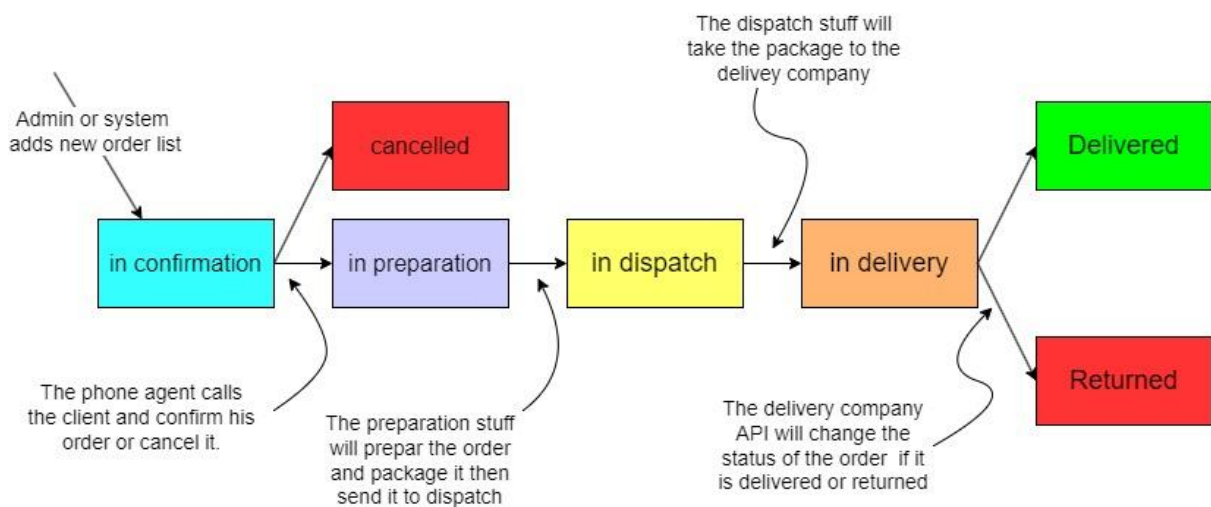


Figure 1.1 Order fulfillment process

1.2.1 Problem statement

In order to build our project, we have discussed with local online store and we came up with some of the problems they are facing among which:

- Orders and leads management by every person on the team.
- Lack of centralization of the data of customers and orders to be treated by phone agents.
- A great deal of tedious work because of going through multiple platforms such as google sheets and excel.
- Inability to store data efficiently for retargeting purpose.
- No efficient way to coordinate between delivery guy and the phone agents.

1.2.2 Objectives

Our objective is to design and implement a web application to solve the problems mentioned above by:

- Automating the orders confirmation by the phone agents.
- Task management through set of permission for each user.
- Importing of order to the database from google sheets and excel.
- Storing the leads in relational data base for easy selecting and searching.
- Tracking the workflow of the phone agents.
- Collection of data.
- Coordinate data and work with local delivery company

1.3 Web application

1.3.1 Definition

A computer program that runs on a web server is referred to as a "web application" or "web app". Web apps must be accessed through a web browser, as opposed to conventional desktop applications, which are launched by your operating system. The advantages of web apps over desktop ones are numerous. Web apps do not need to be developed for many platforms because they operate inside web browsers. For instance, a single Chrome application will run on both Windows and OS X. When a web app is updated, developers are not required to provide consumers with software upgrades. Users can access the updated version of the program by making server-side updates. [1]

1.3.2 Single-Page Application

Single-page applications are programs that run entirely within a browser and do not require page reloading to function. One uses daily apps of this kind. Examples of this include Gmail, Google Maps, Facebook, and GitHub.

With no page reloads or added wait times, SPAs are all about providing an excellent UX by imitating a "natural" environment in the browser — no page reloads, no extra wait time. It is just one web page that one visits which then loads all other content using JavaScript and its frameworks — which they rely on.

SPA renders pages directly in the browser after separately requesting the markup and data. Thanks to cutting-edge JavaScript frameworks like Angular, Ember.js, Meteor.js, and Knockout.js, we are able to do this.

Single-page websites assist in keeping the user in a single, pleasant digital environment where content is presented in a straightforward, user-friendly, and practical way. [2]

1.3.3 Web client and Web server

1.3.3.1 Web client

A Web client is typically the Web browser installed on the user's computer or mobile device. It may also refer to browser extensions and utility applications that augment the browser to support specialized site services. [3]

1.3.3.2 Web server

A web server is a computer that contains web server software and the component files of a website (such as HTML documents, images, CSS stylesheets, and JavaScript files). A web server connects to the Internet and facilitates physical data exchange between web-connected devices.

On the software side, a web server consists of a number of components that regulate how web users gain access to hosted files. This is at minimum an HTTP server. An HTTP server is software that comprehends URLs (web addresses) and HTTP (the protocol your browser uses to display web pages). An HTTP server is accessible via the domain names of the websites it hosts, and it transmits the content of these websites to the end user's device. [4]

1.4 HTTP

HTTP is a protocol used to retrieve resources like HTML documents. It is the basis for any data exchange on the Web and is a client-server protocol, meaning requests are initiated by the recipient, typically a Web browser. Reconstructing a document from its sub-documents, such as text, layout description, images, videos, and scripts, yields the final document. [5]

1.5 URL

URL is an abbreviation for Uniform Resource Locator. A URL is nothing more than the address of a particular, unique Web resource. In theory, every valid URL should refer to a distinct resource. Such resources can be an HTML page, a CSS document, an image, etc. In

practice, there are exceptions, the most common of which is a URL that points to a resource that no longer exists or has relocated. As both the resource represented by the URL and the URL itself are managed by the web server, the proprietor of the web server is responsible for managing both the resource and its URL. [6]

A URL is composed of different parts, some mandatory and others optional. The most important parts are highlighted on Figure 1.1

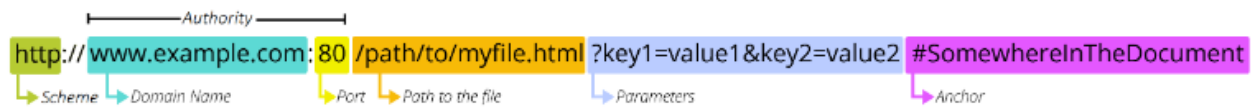


Figure 1.2: URL parts [6]

1.6 RESTful API

A RESTful API, also known as a RESTful web service or REST API, is based on representational state transfer (REST), which is an architectural design for an application program interface (API) that makes use of HTTP requests to access and manipulate data. That data can be used to GET, PUT, POST, and DELETE data types, which refer to the receiving, updating, creating, and deleting of operations concerning resources.

An API for a website is code that allows two software programs to communicate with each other. The API outlines the correct way for a developer to write a program that requests services from an operating system or other application. [7]

1.7 Conclusion

Through this chapter, we specified the problems that cash-on-delivery stuff are facing and we proposed to help solving them using a web application. Then we introduced a summary of a web application and some concepts of the web.

CHAPTER 2

TOOLS AND TECHNOLOGIES

2.1 Introduction

Programmers use a variety of tools and technologies in computer science to implement a system, which is defined as the realization of a technical specification of an algorithm as a program. This chapter describes the tools and technologies used to execute this project.

2.2 Development tools

2.2.1 IntelliJ IDEA

IntelliJ IDEA is an Integrated Development Environment (IDE) for JVM languages such as Java and Kotlin that aims to maximize developer efficiency. It performs routine and repetitive duties for you by providing intelligent code completion, static code analysis, and refactoring, allowing you to concentrate on the productive and enjoyable aspects of software development.

IntelliJ IDEA provides a set of inspections that are built-in static code analysis tools. They help developers find potential bugs, locate dead code, detect performance issues, and improve the overall code structure. [8]

2.2.2 Visual studio code

Visual Studio Code is a free open source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS. Although the editor is relatively lightweight, it includes some powerful features that have made VS Code one of the most popular development environment tools in recent times. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET). [9]

2.2.3 Spring framework and Spring boot

Java Spring Framework is a popular, open source, enterprise-level framework for creating standalone, production-grade applications that run on the Java Virtual Machine (JVM).

Java Spring Boot (Spring Boot) is a tool that makes developing web application with Spring framework faster and easier through these core features:

- Create stand-alone Spring applications
- Automatically configure Spring and 3rd party libraries whenever possible
- An opinionated ‘starter’ dependencies to simplify configuration. [10]

2.2.4 Angular

Angular is a platform and framework for developing HTML and TypeScript-based single-page client applications. TypeScript is used to write Angular. It implements essential and optional functionality as a collection of TypeScript libraries that applications can import.

Certain fundamental concepts guide the architecture of an Angular application. Angular components and modules comprise the Angular framework's fundamental building blocks. [11]

2.2.5 Web browser

A web browser is a piece of software that enables users to locate, access, and view web pages. Web browsers are mainly utilized for displaying and gaining access to websites on the Internet, as well as other content created with languages such as Hypertext Markup Language (HTML). Browsers convert HTTP-delivered web pages and websites into human-readable content. [12]

2.3 Programming languages

In this section, we will present and define the several programming languages used to develop our web application.

2.3.1 Java

Java is a class-based, object-oriented, general-purpose programming language designed to have fewer implementation dependencies. It is an infrastructure for application development. Java is thus quick, secure, and reliable. It is extensively employed in the development of Java applications for laptops, data centers, game consoles, scientific supercomputers, mobile phones, etc. [13]

2.3.2 HTML

HTML, or hypertext markup language, is a format for displaying information retrieved from the Internet. Each retrieval unit is referred to as a Web page (from the World Wide Web), and these pages frequently contain hypertext links that permit retrieval of related pages. HTML is the language used to encode web pages.

Markup tags in HTML define document elements like headings, paragraphs, and tables. They mark up a document so that a web browser can display it. The browser interprets the elements and adapts the layout of headings, paragraphs, and tables to the screen size and fonts available to it. A browser does not display HTML tags, but instead uses them to determine how to display a document. [14]

2.3.3 CSS

CSS, which stands for Cascading Style Sheets, specifies how HTML elements should appear on a screen, in print, or in other media. It is used to define web page styles, including design, structure, and display modifications for various devices and screen sizes.

CSS allows web designers to create a uniform appearance across multiple pages of a website. Commonly used styles need only be declared once in a CSS document, as opposed to describing the design of each table and text block in the HTML of a page. [15]

2.3.4 JavaScript

JavaScript is a programming language that is dynamic. It is a lightweight scripting language that is typically incorporated into web pages; its implementations permit client-side scripts to interact with the user and generate dynamic pages. It is an object-oriented programming language that is interpretable.

JS renders web pages in an interactive and dynamic way. This allows the sites, among other things, to respond to events, display special effects, accept variable text, validate data, create cookies, and recognize the user's browser.

JavaScript can be used to update and alter both HTML and CSS. It has the ability to calculate, manipulate, and validate data. [16]

2.3.5 TypeScript

TypeScript is a strongly typed programming language based on JavaScript that provides improved capabilities at any scale. Typescript adds syntax to JavaScript to facilitate tighter editor integration and early error detection by editors.

2.4 Security

2.4.1 Bcrypt

Bcrypt is a function for hashing passwords that was developed by Niels Provos and David Mazières and presented at USENIX in 1999. In addition to incorporating a salt to protect against rainbow table attacks, Bcrypt has an adaptive function: the iteration count can be increased over time to make it slower, so it remains resistant to brute-force search attacks even as computing power increases. [17]

2.4.2 JSON Web Token

JSON Web Tokens (JWTs) are an open standard (RFC 7519) that define a compact and self-contained method for securely transmitting information between parties in a JSON object format. This information can be verified and trusted because it is digitally signed. [18]

2.4.2.1 JWT Structure

JWTs consist of three parts separated by dots (.), which are:

- Header
- Payload
- Signature

Therefore, an encoded JWT typically looks like shown in Figure 2.1

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4  
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.  
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Figure 2.1 Encoded JSON web token [18]

2.4.2.2 JWT Use cases

JWTs are beneficial in numerous situations:

Authentication: Once the user has authenticated in, every subsequent request will include the JWT, granting the user access to routes, services, and resources that are permitted with that token. Single sign-on is a feature that employs JWT extensively today due to its ease of use across different domains. [19]

Information exchange: JWTs can be used for the secure transmission of information between participants. Because they can be signed, such as with public/private key pairs, you can be certain that the senders are who they claim to be. [19]

2.5 Conclusion

In this chapter, we provided simple and succinct definitions of the tools and technologies we used to develop our interactive, dynamic, and responsive web application.

CHAPTER 3

SYSTEM DESIGN

3.1 Introduction

This chapter will illustrate our application's entire design procedure. We will use modeling to gain a deeper understanding of the system we are building and its behavior in order to create a web application that meets the requirements of consumers.

UML will be used for modeling, and the use case diagram will be utilized to represent the player's responsibilities. The conclusion of the chapter provides an overview of the database and all of its migrations.

3.2 Design Requirements

3.2.1 Modeling language

A modeling language is a formal language used to represent in a structured manner complex systems, processes, and structures. They are utilized to express information, knowledge, and systems in a consistent structure defined by a set of principles. These principles are used to interpret the meaning of the structure's components. Each modeling language is designed for a particular purpose, such as representing software designs, business processes, or the physical configurations of systems. [20]

3.2.2 Unified modeling language (UML)

UML, which stands for Unified Modeling Language, is a general-purpose, developmental modeling language in the field of software engineering that aims to standardize the visualization of system design. Grady Booch, Ivar Jacobson, and James Rumbaugh at Rational Software created it between 1994 and 1995. UML consists of a collection of graphic notation techniques for creating visual representations of software-intensive systems. It provides a variety of structural and behavioral diagrams. [21]

3.2.2.1 Structural diagram

Structural UML diagrams are a form of diagram that depicts the time-independent elements of a specification. Class, composite structure, component, deployment, object, and package diagrams are also included. [22]

3.2.2.2 Behavioral diagram

Behavioral UML diagrams are a form of diagram that illustrates the behavioral characteristics of a system or business process. Included are activity, sequence, use case, state, communication, interaction, and timing diagrams. [22]

3.3 Use Case Diagram

3.3.1 Definition:



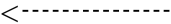
UML's use case diagram is a behavior or dynamic diagram. Actors and use cases are used to model the functionality of a system in use case diagrams. Use cases are a collection of required system actions, services, and functions. Actors are individuals or entities executing predetermined positions within a system. The actor can be a human or other external system. [23]

Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation. [24]

3.3.2 Relationships

The following table provides a summary of the relationships present within the use case diagram, offering a clear overview of the interactions and dependencies among the various components and actors.

Table 3.1 Summary of use case diagram's relationships and used arrows. [24]

| Symbols | Relationship description |
|--|--|
| Association  | Association: Use cases are associated with the actors that perform them. Aline is used to link actors to use cases |
| <<Include>>  | Include: A dependency between a base use case and an included use case is indicated by an include relationship. Every time the base use case is conducted, the included use case is also executed, or, to put it another way, the base use case is incomplete without an included use case. When a use case is included, a dashed line with an arrow pointing towards the included use case is drawn. |
| <<Extend >>  | Extend: A base use case and an extended use case are also present. When the base use case is executed, the extended use case will sometimes, but not always, occur. The extended use case will only occur if certain conditions are met. Another way to think about it is that you have the option to extend the behavior of the base use case. |

3.4 Application modeling

We are going to design and model our application using use cases diagrams.

3.4.1 Actors

There are four types of actors in our design model:

- **Admin:** admin access to the system and control its overall aspects. They are the ones who has full access to the database and can make necessary operations, such as adding

users, products and delivery agencies. Admins also can view all the parts and interfaces of the system and perform update, delete and edit orders.

- **Phone Agents:** are key players in the process. They are responsible for reaching out to clients to either confirm or cancel their orders. Additionally, they are often the first point of contact for incoming phone orders. Their duties include viewing and managing orders that are assigned to them at the confirmation stage, as well as accessing the product list and placing new orders as needed.
- **Delivery guy:** Their tasks involve preparing these orders and forwarding them to the delivery company. Therefore, they have the capability to manage orders at various stages, including preparation, dispatch, and delivery.
- **Delivery agency system API:** serves as an essential link in the order fulfillment process. Following the delivery or return of an order by the delivery company, the system updates the order's status via an API endpoint.

3.4.2 Use cases

Use cases are represented using ovals labeled with verbs that describes the system's functions. In our application, we define the following use cases:

- **Authentication:** is used to determine the authority and permissions of the user. Login before accessing the app is required to insure the user identity.
- **Manage users:** Allows the admin to create, edit and delete users' accounts.
- **Manage order:** Allows the system users to add, edit, delete or update the orders' status.
- **Manage Product:** Gives the admin the ability to create, edit or delete products to the system. While it allows the other actors viewing the products only.
- **Manage Delivery Agencies:** Allows the admin to add, edit or delete delivery agencies and their APIs.

Table 3.2 Use cases of each actor

| Actors | Use cases |
|-----------------|---|
| Admin | <ol style="list-style-type: none"> 1. Authenticate: login. 2. Manage users: add, edit, view and remove 3. Manage orders: add, delete, consult, updating their status or assign them to a phone agent 4. Manage products: add, delete, view or edit. 5. Manage delivery agencies: add, remove, view or edit |
| Phone agent | <ol style="list-style-type: none"> 1. Authenticate: login. 2. Manage orders: phone agents can only manage the orders that are assigned to them at the confirmation level. 3. View products: check a product price or description. 4. View delivery agencies: check available agencies. |
| Delivery guy | <ol style="list-style-type: none"> 1. Authenticate: login. 2. Manage orders: starting from preparation level, delivery guy can fully manage orders until they reach in delivery stage. 3. View products: check a product price and id. 4. View delivery agencies: check available agencies and direct orders to them. |
| Delivery agency | <ol style="list-style-type: none"> 1. Authenticate: delivery agencies authenticate through API token and id. 2. Manage orders: agency system API can manage orders at “in delivery stage” and mark them as delivered or returned. |

3.4.3 Application's use case diagrams

In this section we present the actors' use case diagrams as shown in figures below.

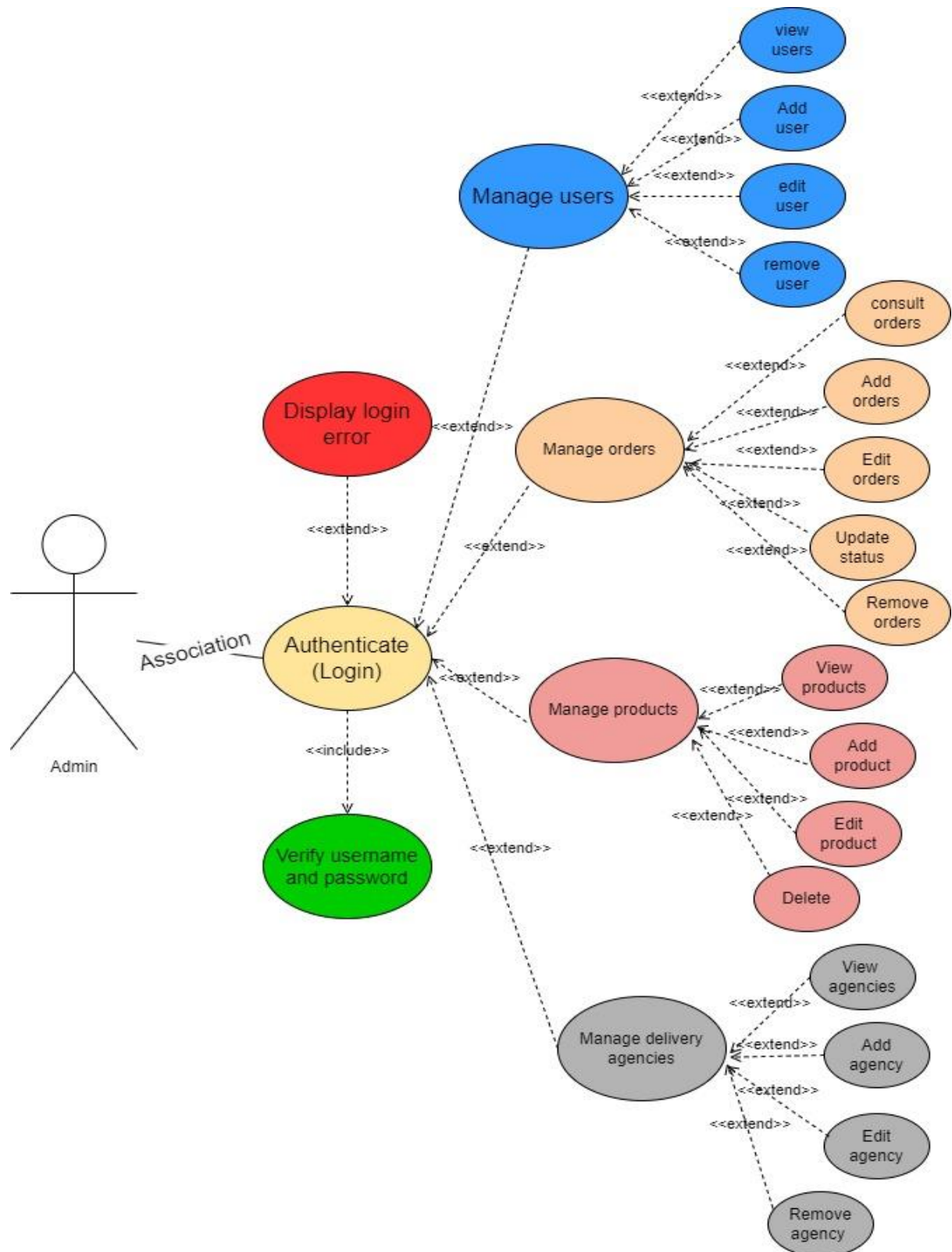


Figure 3.1 Admin use case diagram

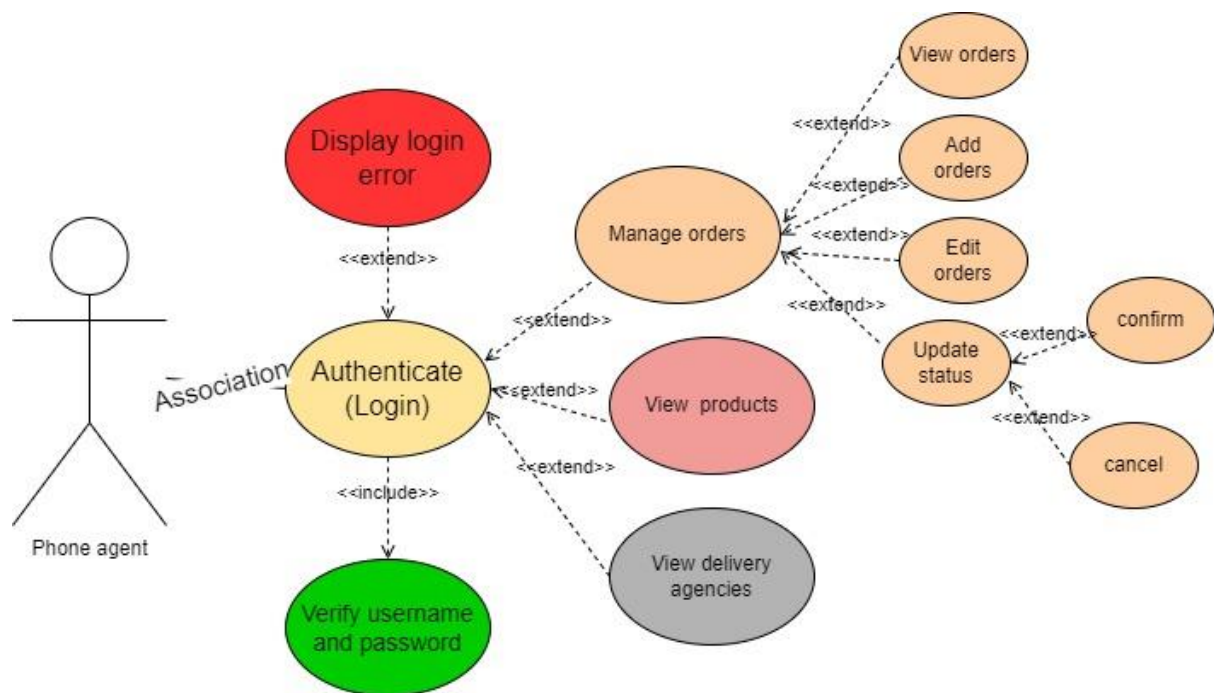


Figure 3.2 Phone agent use case diagram

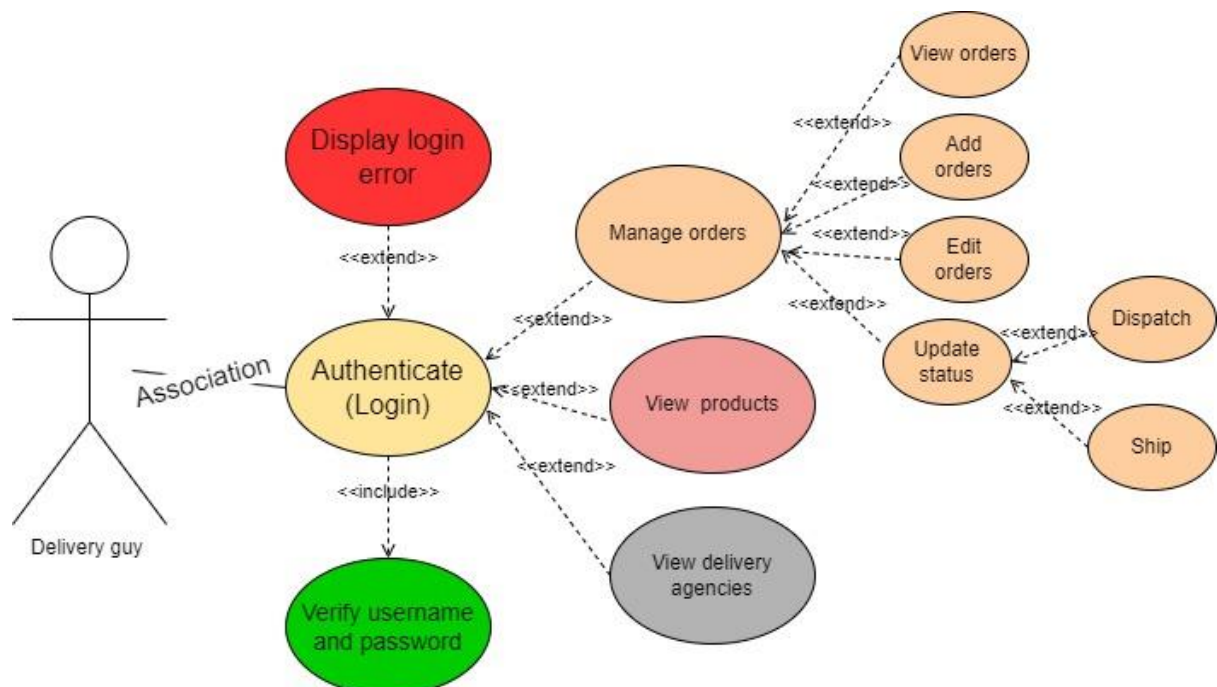


Figure 3.3 Delivery guy use case diagram

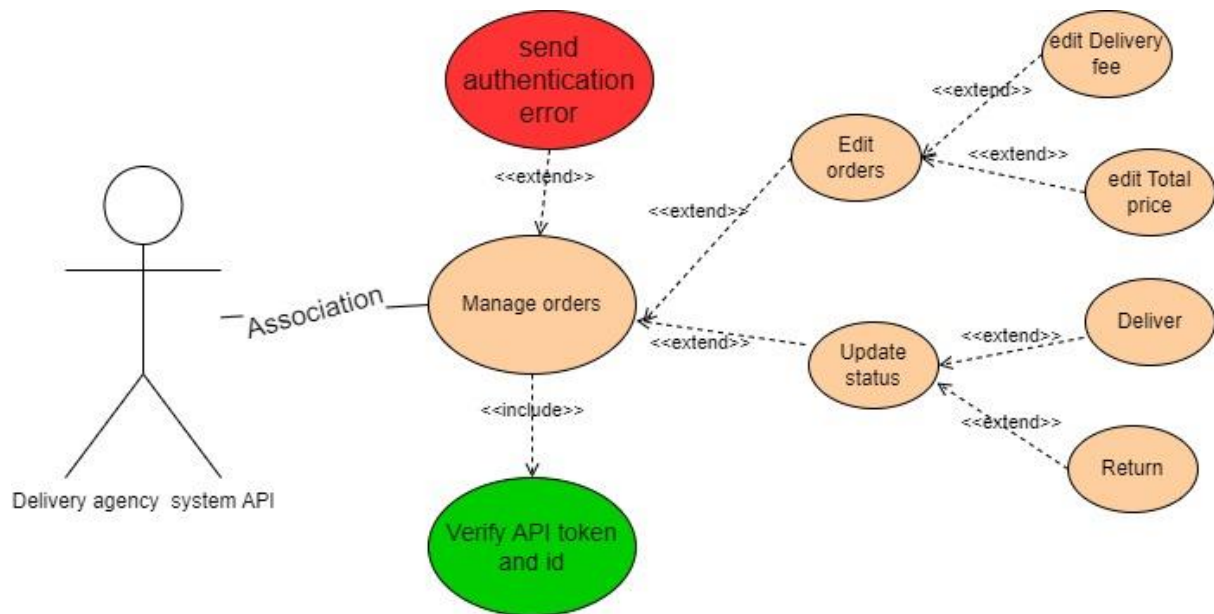


Figure 3.4 Delivery agency system API use case diagram

3.4.4 Textual description of use cases

In this section we present the textual description for different use cases as shown in tables below.

Table 3.3 Textual description for authentication use case

| Use case name | Authentication |
|---------------|---|
| Actor | Admin, Phone agent, Delivery guy |
| Objective | Authenticate to have access to the application |
| Precondition | Browser and access to internet |
| Scenario | <ol style="list-style-type: none"> 1. The user lunches the application via a browser. 2. The system asks for username and password. 3. The user enters his name and password. 4. The system checks the conformity of the information entered by sending an authentication query to the server. 5. The server verifies the query and send favorable answer. 6. The user accesses the application |
| Alternative | If the username or password is wrong or missed the system displays an error message. |

Table 3.4 Textual description for manage users use case

| | |
|----------------------|--|
| Use case name | Manage users |
| Actor | Admin |
| Objective | Manage users to add new user |
| Precondition | Authentication to admin account |
| Scenario | <ol style="list-style-type: none"> 1. The admin accesses to users tab. 2. The admin clicks on “add user” in the interface 3. The admin enters username, email, phone, password, and chooses a role for the new user, then presses add 4. The system sends query to the server for processing and checking. 5. The system creates new account. |
| Alternative | If the username or password is wrong or missed the system displays an error message. |

Table 3.5 Textual description of manage orders use case

| | |
|----------------------|---|
| Use case name | Manage orders |
| Actor | Admin, Phone agent, Delivery guy |
| Objective | Manage orders to edit or update status. |
| Precondition | Authentication to have access to the order |
| Scenario | <ol style="list-style-type: none"> 1. User opens the application and accesses to the allowed orders or level to see the orders list at that stage. 2. User clicks on the desired order to expand details in the interface, then performs the editing or updating status. 3. The system sends a query to the server for processing the changes. 4. The server sends message back that says the change is done. |
| Alternative | If the query fails, an error message will be displayed to the user (return to 2). |

Table 3.6 Textual description for manage products use case

| | |
|----------------------|---|
| Use case name | Manage Products |
| Actor | Admin |
| Objective | Manage projects to add new product. |
| Precondition | Authentication to admin account |
| Scenario | <ol style="list-style-type: none"> 1. The admin accesses to products tab. 2. The admin clicks on “add product” in the interface 3. The admin enters product’s information, then presses add button. 4. The system sends query to the server for processing and checking. 5. The system creates new products in the database. 6. The user accesses the application |
| Alternative | If the product Id is wrong or the product name is not unique, the system displays an error message. |

Table 3.7 Textual description of manage delivery agencies use case

| | |
|----------------------|--|
| Use case name | Manage users |
| Actor | Admin |
| Objective | Manage users to add new user |
| Precondition | Authentication to admin account |
| Scenario | <ol style="list-style-type: none"> 1. The admin accesses to delivery agencies tab. 2. The admin clicks on “add agency” in the interface 3. The admin enters agency name, API id, API token. 4. The system sends query to the server for processing and checking with the system of the agency. 5. The system adds new agency. |
| Alternative | If the data are not valid the system displays an error message. |

3.5 Introduction to database

3.5.1 Definition

A database is typically defined as a structured collection of data or information that is systematically organized for expeditious search and retrieval by a computer, also known as an electronic database. The fundamental purpose of databases is to streamline the processes of data storage, retrieval, modification, and deletion along with various other data-processing tasks. A Database Management System (DBMS) is utilized to extract data from the database in response to specific queries. [25]

There are four primary types of databases: hierarchical, network, relational, and non-relational models, each offering unique advantages for specific data storage and retrieval needs.

3.5.2 Non-relational model

A non-relational database model represents a database that does not use the conventional model of using rows and columns schema typically observed in traditional database systems. Non-relational databases employ a storage model tailored to the unique needs of the data type being housed. This could involve storing data as uncomplicated key/value pairs, as JSON documents, or structured as a graph featuring edges and vertices. These distinct storage methods enhance the flexibility and efficiency of managing diverse data types. The data in the fields of a document can be encoded in a variety of ways, including XML, YAML, JSON, BSON, or even stored as plain text. The fields within documents are exposed to the storage management system, enabling an application to query and filter data by using the values in these fields. [26]

3.5.3 Relational model

A relational database is a collection of data entities with pre-defined relationships between them. These data entities are systematically arranged into tables, each comprising of columns and rows. The role of these tables is to store data about the entities that the database is intended to represent.

Every column in a table is allocated to contain a specific type of data, and a field is for holding the actual attribute value. The rows within a table denote a group of related values of a single object or entity.

Unique identifiers, referred to as ‘primary keys’, can be used to label each row within a table. Relationships between rows in different tables can be established through the use of ‘foreign keys’. The structural design of this system allows for diverse ways to access the data without reorganizing the database tables themselves. [27]

3.5.4 PostgreSQL

PostgreSQL, often simply Postgres, is an open-source relational database management system (RDBMS) that emphasizes extensibility and SQL compliance. It was originally developed at the University of California, Berkeley in the 1980s and has since become an enterprise-class database system backed by an active community of developers.

Key features of PostgreSQL include complex queries, foreign keys, triggers, updatable views, transactional integrity, and multi-version concurrency control. [28]

3.5.5 Database Schema

One of the most important steps that controls and manages the information used by the user is the database. The database schema of this application is comprised of six tables, described in this section.

1) Users' table:

It consists of 5 attributes.

- **username:** is the username that should be provided by the admin when adding new user. It is considered as the primary key.
- **email:** is the email address that the admin needs to contact the users if it is needed. This attribute should be unique.
- **phone:**
- **role:** is the role of the user in the application (admin, phone agent, delivery guy)
- **password:** is the password of the user that is set by the admin when they create the user. It is encrypted and stored in the database using Bcrypt.

2) Orders' table

It consists of 12 attributes.

- **order_id:** is unique identification number that is considered as the primary key.
- **client_name:** this attributes holds the name of the client related to the order.
- **client_phone:** is the phone number that the phone agents will call to confirm or cancel the order.
- **wilaya:** is the wilaya where the order should be sent by the delivery agency later on.
- **comune:** is the comune or the address of the client, it is useful for the delivery process.
- **assigned_to:** holds the username of the user that the order is assigned to. It is a foreign key related to table 'users'.
- **delivery_agency_id:** is also a foreign key from table 'delivery_agencies'.
- **notes:** is where notes about an order are stored as a text.

- **delivery_type:** is one of two types (stop desk, home delivery). It is required in the delivery process by the delivery agencies.
- **reduction:** is a value that will be reduced from the total price, if it is present.
- **delivery_fees:** is the delivery fee that is taken by the delivery agency or set by the user processing this order.
- **total_price:** represents the amount of cash that the client is required to pay when they receive their order

3) Status' table

This table represents the history of actions and statuses that the order has been through.

It consists of 5 attributes.

- **status_id:** is identification number that is considered as the primary key
- **order_id:** is a foreign key from table orders, so the status can be related only to one order.
- **changed_by:** holds the user that performed the action.
- **change_date:** is the time and date of the actions performed.
- **status:** is the level of the order for example: in confirmation, in preparation, delivered...

4) Products' table

It consists of 7 attributes.

- **product_id:** is unique identification that is considered as the primary key.
- **product_name:** is the name of the product, and it should be unique.
- **product_price:** holds the product's unit price.
- **description:** is a text description of the product, that is useful to phone agents.

- **product_URL:** holds the URL to the landing page where the product is being ordered. Also useful for the phone agents.

5) Products_orders' table

In a relational database, when there is a many-to-many relationship between two entities, it is common to use a joint table, also known as a junction table or an associative table. The joint table serves as an intermediary between the two entities and facilitates the representation of the relationship.

The joint table typically consists of two or more columns, each representing a foreign key referencing the primary keys of the associated entities. Additional columns can be added to store any relevant attributes specific to the relationship itself.

In our application we have many-to-many relationship. So we used a joint table between products and orders. It consists of 4 attributes:

- **order_id:** is foreign key and a part of the composite primary key.
- **product_id:** is foreign key and a part of the composite primary key.
- **quantity:** is the quantity of the ordered product
- **product_order_price:** is by default quantity times the product unit price. It can be modified by the user.

6) Delivery agencies' table

It consists of 4 attributes:





- **agency_id:** is unique identification that is considered as the primary key.
- **agency_name:** holds the name of the delivery agency.
- **api_token:** holds the required token. It used for HTTP requests to the delivery agency system API end points.

3.5.6 Application entity relationship diagram

In this section, we present the ER diagram of our application. The ER diagram serves as a visual representation of the entity-relationship model that depicts the structure and relationships among various entities within our application. By examining this diagram, one can gain a comprehensive understanding of how different entities interact and connect with each other, providing valuable insights into the underlying data architecture.

Table 3.8 shows used arrows in ER diagram that is represented in Figure 3.5.

Table 3.8 used arrows in ER diagram of this system

| Symbol | Relationship |
|---|--|
|  | 0..1 : 0..N zero or one –to- zero or many |
|  | 1..N : 1 one or many –to- one |
|  | 1 : 1..N one –to- one or many |
|  | 1 : 0..N one –to- zero or many |

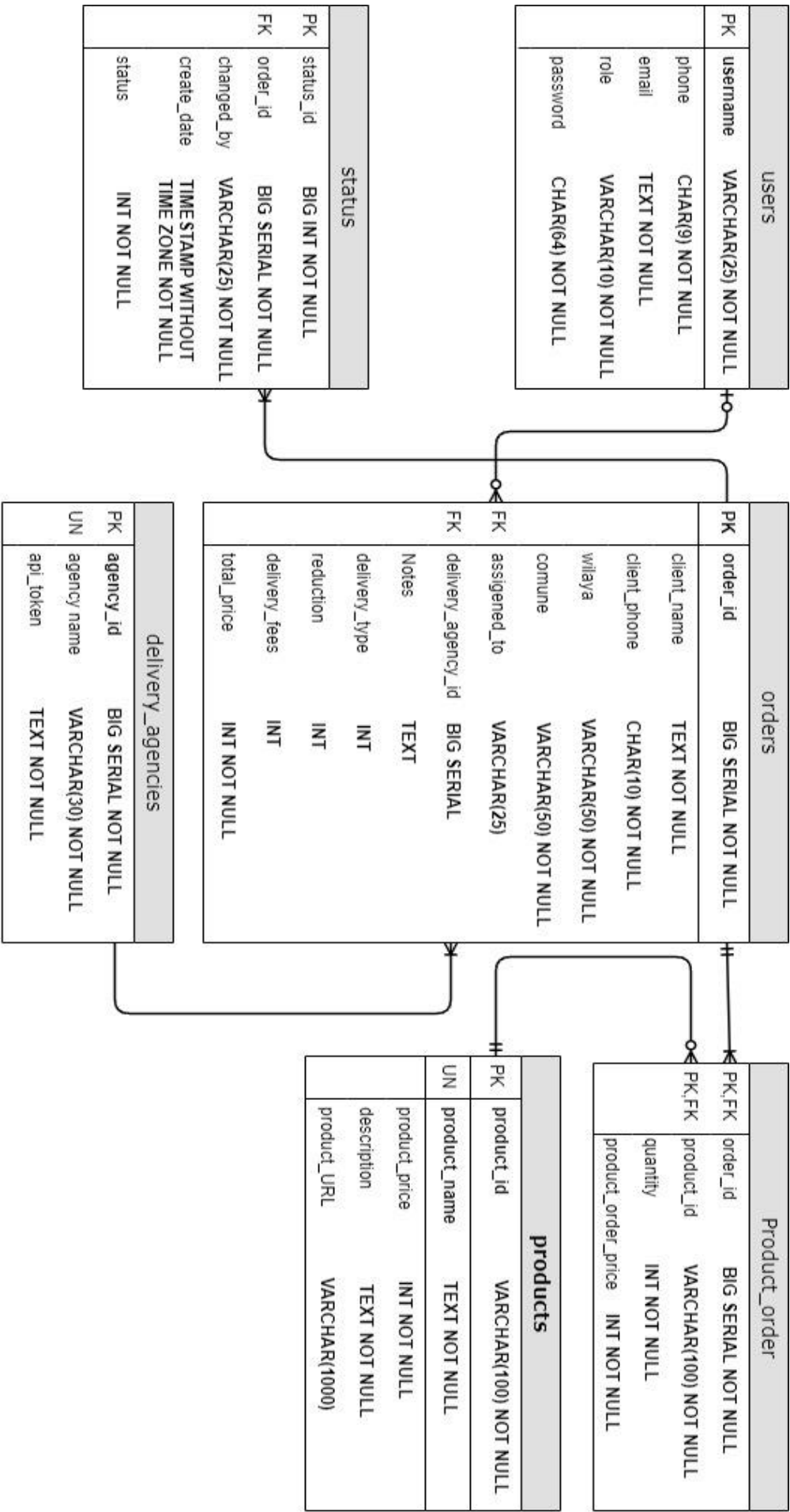


Figure 3.5 ER diagram of this system's database

3.6 Conclusion

In this chapter, we used UML diagrams to model our web application. We started by outlining the details of our program and then used a case diagram to show how actors attain their goals. Finally, we have given a brief overview of database models and demonstrated our application's entity diagram.

CHAPTER 4

IMPLEMENTATION AND TESTS

4.1 Introduction

After designing the application and specifying the roles of each user, this last chapter is concerned with the implementation. We give a presentation of the application and its functionalities accompanied with some user interfaces.

4.2 Interfacing with the application

After implementing the application with the tools specified in the previous chapters, in this section, we present the different interfaces, the user will be dealing with. It is to be noted that all the filled data that are saved in the database was generated by a faker library that give as a random string.

4.2.1 Login Interface

As all secured applications, this following interface is the first one that will be displayed when the user wants to access the application. This application came with a single admin account. This admin is responsible of adding other users to the system. When the user enters username and password, the system sends a query to the server for checking. If this information exists in the database they can access, otherwise an error message will be displayed. Figure 4.1 below shows the authentication interface.

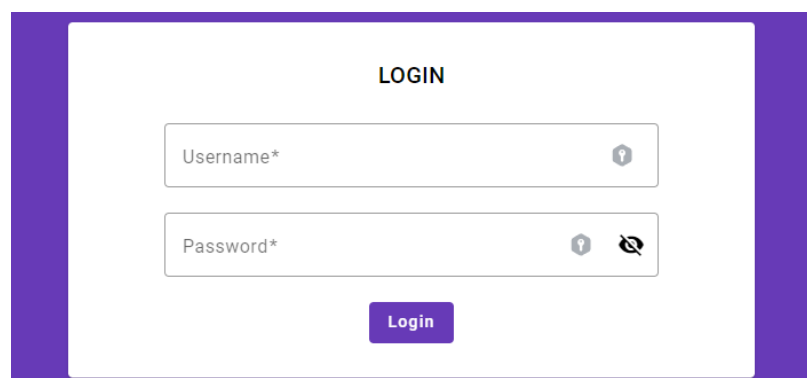


Figure 4.1 Authentication interface

4.2.2 Dashboard interfaces for each user

After login, all users have three parts in their dashboard interface, a header, a sidebar and a main section where the user can go to different tabs. The main section and sidebar differ between admins and other users:

- For admins, the dashboard contains all the tabs and features available in the application. The main section initiated with the tab “add” for adding orders as shown in Figure 4.2
- For phone agents, it has limited tabs on both sidebar and main section. The main section initiated with the tab “in confirmation” as shown in Figure 4.3
- For delivery guy, it has less tabs and features than a phone agent can access. The main section initiated with the tab “in preparation” as shown in Figure 4.4

The screenshot displays the 'COD manager' Admin dashboard. The top navigation bar is purple with a 'Log out' button. Below it, a horizontal tab bar shows various order status tabs: 'Add', 'In confirmation', 'In preparation', 'In dispatch', 'In delivery', 'Delivered', 'Returned', and 'Cancelled'. The 'Add' tab is currently selected. On the left, a dark sidebar contains a menu with 'Dashboard', 'Users', 'Products', and 'Delivery Agencies'. The main content area is titled 'Add order using Form' and is divided into two columns. The left column, 'Client information', includes fields for 'client name*', 'client phone*', 'Wilaya*', 'Comune*', and a 'Notes' text area. The right column, 'Products form', includes a 'Select an option' dropdown, 'quantity', 'Price', 'DA', and a '+' button. Below these are fields for 'Sub-price', 'Delivery fee', 'Reduction', and 'Total price'. At the bottom of the main section, there is a 'submit' button. Below the main form, there is a section for 'Add orders using excel file' with a 'Choose File' button and an 'Upload' button.

Figure 4.2 Admin's initial dashboard interface

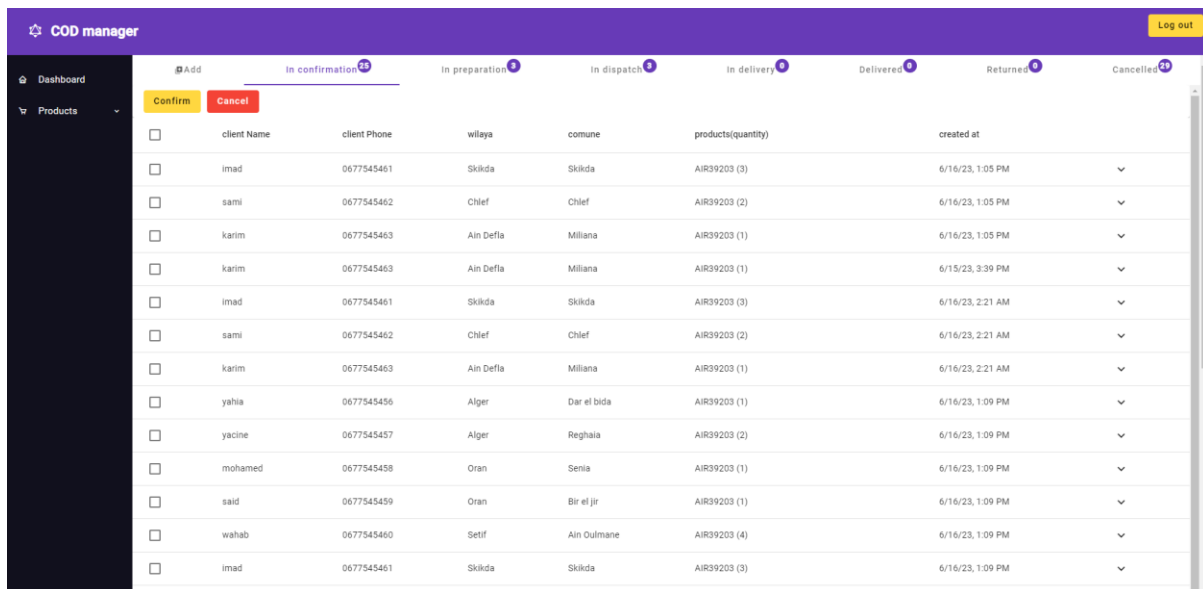


Figure 4.3 Phone agent's initial dashboard interface

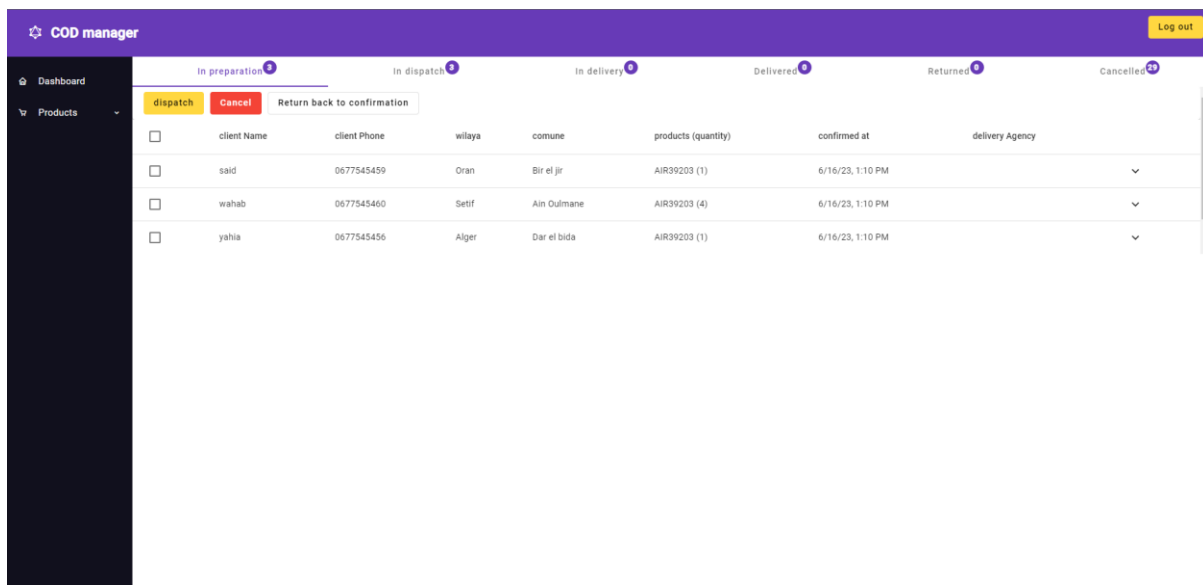
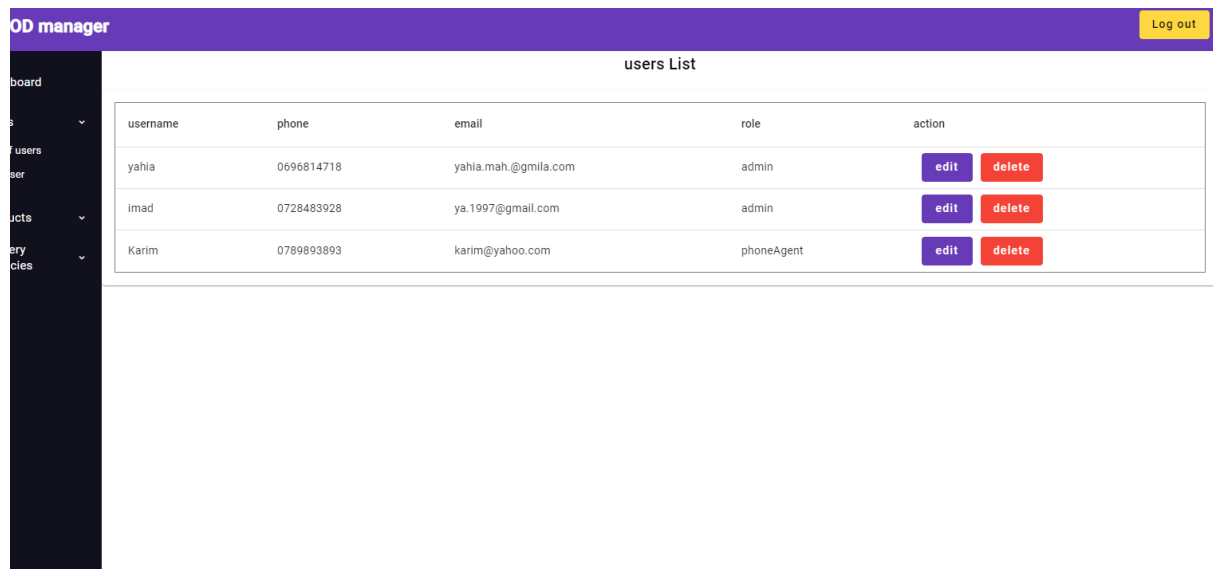


Figure 4.4 Delivery guy's initial dashboard interface

4.2.3 List of users' interface and add new user interface

Only the admins have access to users list and add user interfaces through “Users” dropdown menu in the sidebar.

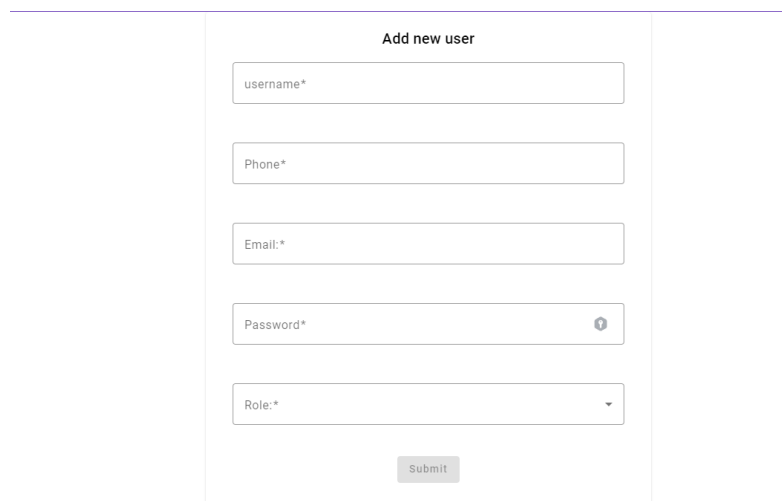
- **List of users' interface** is a table of the users added to the system with some actions buttons. It is shown in Figure 4.5



| username | phone | email | role | action |
|----------|------------|----------------------|------------|---|
| yahia | 0696814718 | yahia.mah.@gmila.com | admin | edit delete |
| imad | 0728483928 | ya.1997@gmail.com | admin | edit delete |
| Karim | 0789893893 | karim@yahoo.com | phoneAgent | edit delete |

Figure 4.5 Users' list interface

- **Add new user interface** is a form fields where the admin enters a new user information then click submit as shown in Figure 4.6. After clicking submit, the system sends a request to the backend to add new user. If the job is done successfully, a pop-up window appears and the page will be redirected to the users list interface. As a sort of security the “Submit” button is disabled (the button is in light gray) until the fields are filled correctly



Add new user

username*

Phone*

Email:*

Password* 1

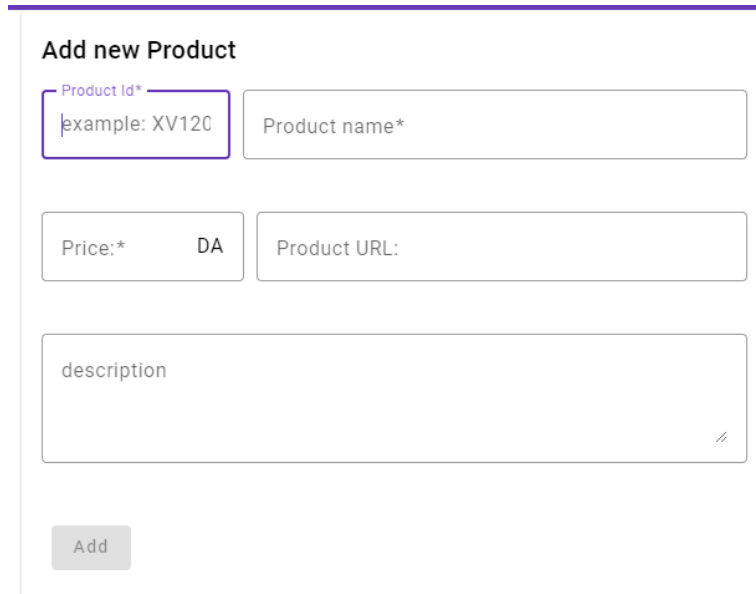
Role:* ▼

Submit

Figure 4.6 Add new user interface

4.2.4 Add new product interface

Figure 4.7 shows this interface that can be accessed only by the admin to add new product and store it in the database.



The form titled "Add new Product" contains the following fields and controls:

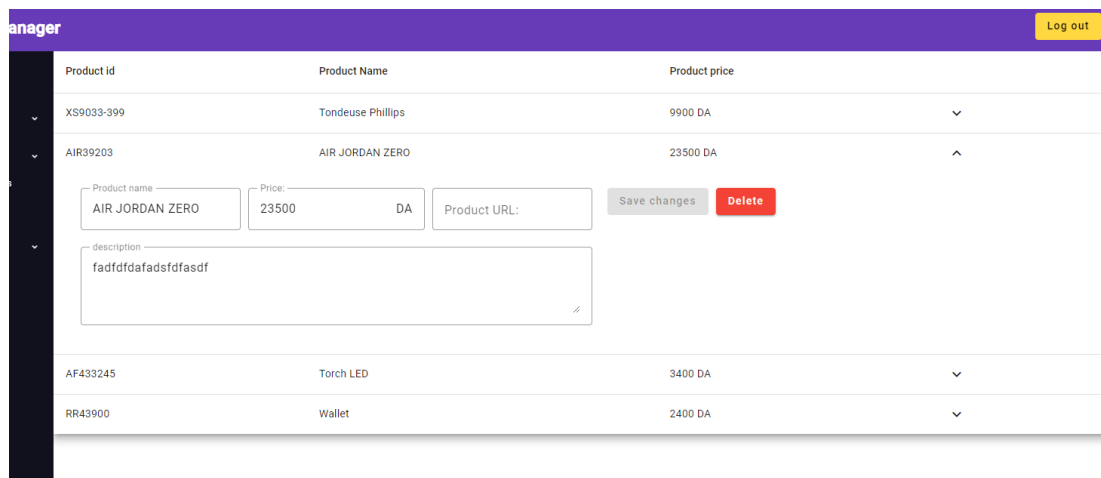
- Product Id*:** A text input field with the placeholder text "example: XV120".
- Product name*:** A text input field.
- Price:*:** A text input field with the value "DA".
- Product URL:** A text input field.
- description:** A large text area with the placeholder text "description".
- Add:** A button at the bottom left of the form.

Figure 4.7 Add new product interface

4.2.5 List of Products interface

This interface is accessed by all users through “Products” dropdown menu on the sidebar, but only admins can perform actions on this interface. It has table of the products stored in the database. This table has expandable rows that allow the users to see details or perform actions.

Figure 4.8 shows the products’ list interface.



The interface shows a table of products with columns: Product id, Product Name, and Product price. The table has four rows of product data. The second row is expanded, showing a form for editing the product details.

| Product id | Product Name | Product price | |
|---|-------------------|---------------|---|
| XS9033-399 | Tondeuse Phillips | 9900 DA | ▼ |
| AIR39203 | AIR JORDAN ZERO | 23500 DA | ▲ |
| <div> <div>Product name</div> <div>AIR JORDAN ZERO</div> </div> <div> <div>Price:</div> <div>23500</div> <div>DA</div> </div> <div> <div>Product URL:</div> <div></div> </div> <div> <div>Save changes</div> <div>Delete</div> </div> <div> <div>description</div> <div>fadfdafadsfdasfd</div> </div> | | | |
| AF433245 | Torch LED | 3400 DA | ▼ |
| RR43900 | Wallet | 2400 DA | ▼ |

Figure 4.8 List of products interface

4.2.6 List of delivery agencies interface

This interface is only accessible by the admins through “Delivery Agencies” dropdown menu in the side bar. It shows a table of the delivery agencies that the system can interact with. Figure 4.9 shows list of delivery agencies interface.

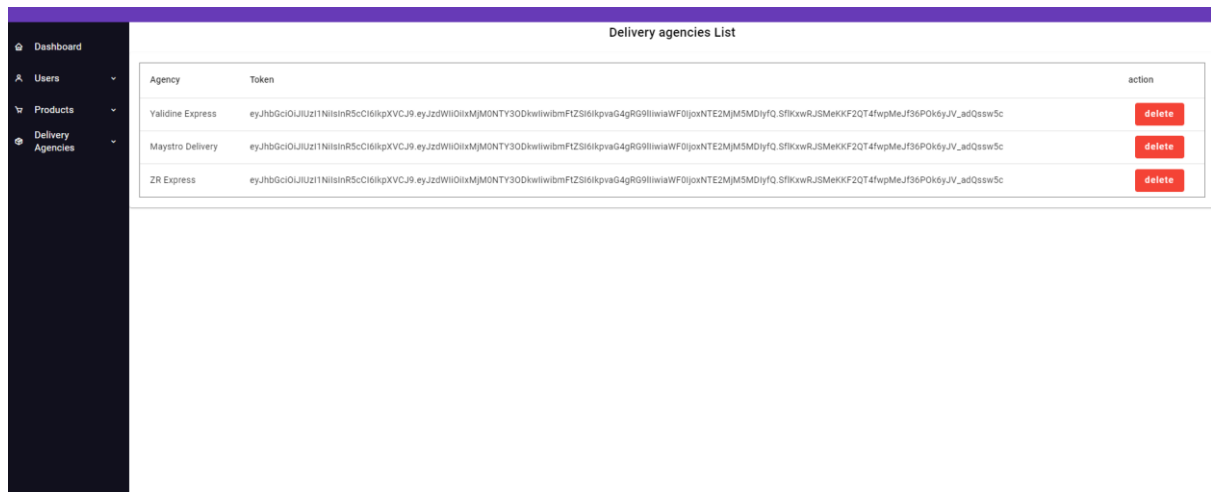


Figure 4.9 List of delivery agencies interface

4.2.7 Add new delivery agency interface

Figure 4.10 shows this interface that can be accessed only by the admin to add new delivery agency.

Add new agency

Agency Name:*

Agency Token*

Add

Figure 4.10 Add new agency interface

4.2.8 Add orders interface

This interface can be access by admins and phone agents through “Add” tab in the main section of the dashboard. Through this interface, admins or phone agents can add an order using the form fields while only admins can add orders using upload excel file feature. Figure 4.11 describes this interface.

Figure 4.11 Add orders interface

4.2.9 In confirmation tab interface

After a user adds orders, the new orders will have a status “in confirmation”. These orders will appear in the tab “in confirmation” in form of table that has expandable and selectable rows as shown in Figure 4.12. This tab is only accessible to admins and phone agents. Admins can assign orders to any phone agent while the phone agents can see only the orders that are assigned to them in this tab. The user can confirm, cancel or expand and edit the order information as Figure 4.13 shows an expanded row.



|  Add | In confirmation ²⁵ | In preparation ³ | In dispatch ³ | In delivery ⁰ | Delivered ⁰ | Returned ⁰ | Cancelled ²⁹ |
|---|---------------------------------------|--|--------------------------|--------------------------|------------------------|-----------------------|-------------------------|
| <input type="button" value="Confirm"/> | <input type="button" value="Cancel"/> | <input type="text" value="Assign user"/> <input type="button" value="assign"/> | | | | | |
|  | client Name | client Phone | wilaya | comune | products(quantity) | created at | |
| <input checked="" type="checkbox"/> | imad | 0677545461 | Skikda | Skikda | AIR39203 (3) | 6/16/23, 1:05 PM | ▼ |
| <input checked="" type="checkbox"/> | sami | 0677545462 | Chlef | Chlef | AIR39203 (2) | 6/16/23, 1:05 PM | ▼ |
| <input checked="" type="checkbox"/> | karim | 0677545463 | Ain Defla | Milliana | AIR39203 (1) | 6/16/23, 1:05 PM | ▼ |
| <input type="checkbox"/> | karim | 0677545463 | Ain Defla | Milliana | AIR39203 (1) | 6/15/23, 3:39 PM | ▼ |
| <input type="checkbox"/> | imad | 0677545461 | Skikda | Skikda | AIR39203 (3) | 6/16/23, 2:21 AM | ▼ |
| <input checked="" type="checkbox"/> | sami | 0677545462 | Chlef | Chlef | AIR39203 (2) | 6/16/23, 2:21 AM | ▼ |
| <input type="checkbox"/> | karim | 0677545463 | Ain Defla | Milliana | AIR39203 (1) | 6/16/23, 2:21 AM | ▼ |
| <input type="checkbox"/> | yahia | 0677545456 | Alger | Dar el bida | AIR39203 (1) | 6/16/23, 1:09 PM | ▼ |
| <input type="checkbox"/> | yacine | 0677545457 | Alger | Reghaia | AIR39203 (2) | 6/16/23, 1:09 PM | ▼ |

Figure 4.12 In confirmation tab interface

| | | | | | | | |
|--------------------------|-------|------------|-----------|----------|--------------|------------------|---|
| <input type="checkbox"/> | karim | 0677545463 | Ain Defla | Milliana | AIR39203 (1) | 6/16/23, 1:05 PM | ▼ |
| <input type="checkbox"/> | karim | 0677545463 | Ain Defla | Milliana | AIR39203 (1) | 6/15/23, 3:39 PM | ▲ |

client name*

karim

client phone*

0677545463

Wilaya*

Ain Defla

Comune*

Milliana

Notes

History of actions:

AIR JORDAN ZERO

AIR39203

quantity

1

PRICE

23500

DA

−

Select a product

quantity

Price

DA

+

Sub-price

23500

Delivery fee

0

Reduction

0

Total price

23500

Select a delivery Agency

Maystro Delivery

Select a delivery Agency

Stop Desk




Figure 4.13 Expanded row from in confirmation tab interface

4.2.10 In preparation tab interface

After phone agents confirm orders, these orders status will change to “in preparation”. These orders will appear in the tab “in preparation” in form of table that has expandable and selectable rows as shown in Figure 4.14. This tab is accessible to all users, but only admins and delivery guy can take actions on this tab. This tab is designed for delivery guys to prepare the orders that appear on it. Then they will send the orders to “in dispatch” tab. Admins can return orders back to confirmation on this interface.

| | client Name | client Phone | wilaya | comune | products (quantity) | confirmed at | delivery Agency |
|--------------------------|-------------|--------------|-----------|-------------|---------------------|-------------------|-----------------|
| <input type="checkbox"/> | imad | 0677545461 | Skikda | Skikda | AIR39203 (3) | 6/17/23, 11:19 PM | ▼ |
| <input type="checkbox"/> | sami | 0677545462 | Chlef | Chlef | AIR39203 (2) | 6/17/23, 11:19 PM | ▼ |
| <input type="checkbox"/> | karim | 0677545463 | Ain Defla | Milliana | AIR39203 (1) | 6/17/23, 11:19 PM | ▼ |
| <input type="checkbox"/> | yahia | 0677545456 | Alger | Dar el bida | AIR39203 (1) | 6/17/23, 11:19 PM | ▼ |
| <input type="checkbox"/> | yacine | 0677545457 | Alger | Reghala | AIR39203 (2) | 6/17/23, 11:19 PM | ▼ |
| <input type="checkbox"/> | wahab | 0677545460 | Setif | Ain Oulmane | AIR39203 (4) | 6/17/23, 11:19 PM | ▼ |
| <input type="checkbox"/> | mohamed | 0677545458 | Oran | Senia | AIR39203 (1) | 6/17/23, 11:19 PM | ▼ |
| <input type="checkbox"/> | said | 0677545459 | Oran | Blir el jir | AIR39203 (1) | 6/17/23, 11:19 PM | ▼ |
| <input type="checkbox"/> | wahab | 0677545460 | Setif | Ain Oulmane | AIR39203 (4) | 6/17/23, 11:19 PM | ▼ |
| <input type="checkbox"/> | imad | 0677545461 | Skikda | Skikda | AIR39203 (3) | 6/17/23, 11:19 PM | ▼ |

Figure 4.14 In preparation tab interface

4.2.11 In dispatch tab interface

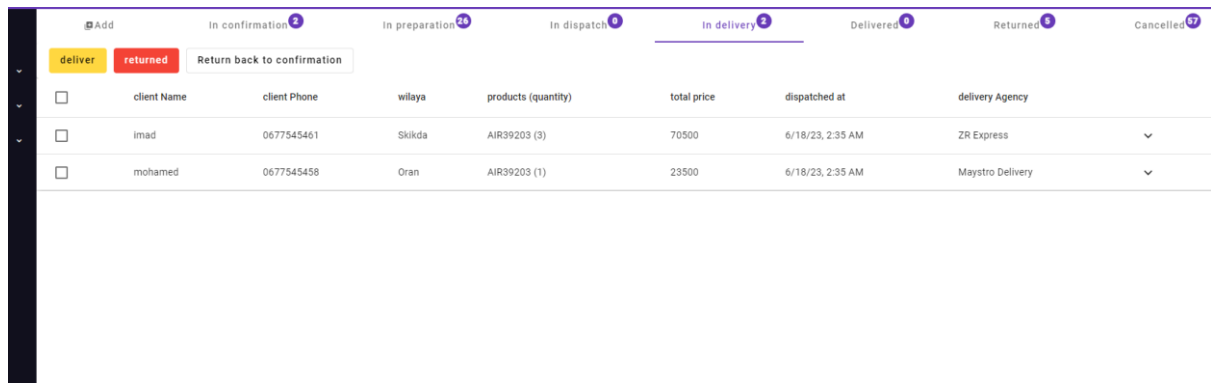
This interface is designed for delivery guy. It has a ship button to send an HTTP request to the selected delivery agency system API. It is accessible to all users with different action buttons.

| | client Name | client Phone | wilaya | products (quantity) | total price | prepared at | delivery Agency |
|--------------------------|-------------|--------------|--------|---------------------|-------------|------------------|--------------------|
| <input type="checkbox"/> | imad | 0677545461 | Skikda | AIR39203 (3) | 70500 | 6/18/23, 1:42 AM | ZR Express ▼ |
| <input type="checkbox"/> | mohamed | 0677545458 | Oran | AIR39203 (1) | 23500 | 6/18/23, 1:45 AM | Maystro Delivery ▼ |

Figure 4.15 In dispatch tab interface

4.2.12 In delivery tab interface

This interface is accessible to all users. Orders in this tab are synchronized the delivery agency deliver API. Therefore, if an order is delivered or returned by the delivery agency, the system will automatically update its status. Only admins can take actions on the orders in this status tab if the system APIs fails to synchronize.

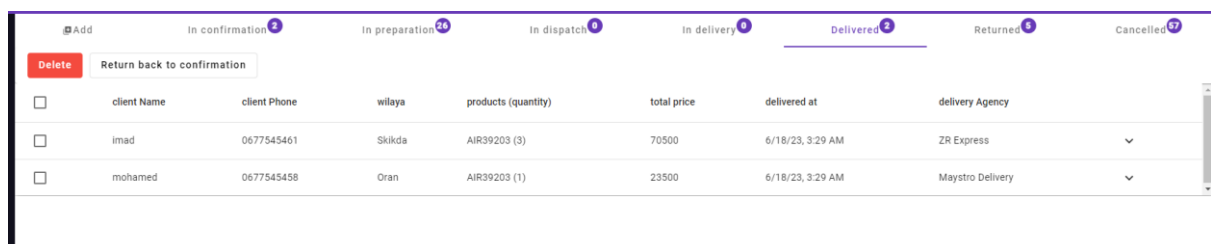


| | client Name | client Phone | wilaya | products (quantity) | total price | dispatched at | delivery Agency |
|--------------------------|-------------|--------------|--------|---------------------|-------------|------------------|------------------|
| <input type="checkbox"/> | imad | 0677545461 | Skikda | AIR39203 (3) | 70500 | 6/18/23, 2:35 AM | ZR Express |
| <input type="checkbox"/> | mohamed | 0677545458 | Oran | AIR39203 (1) | 23500 | 6/18/23, 2:35 AM | Maystro Delivery |

Figure 4.16 In delivery tab interface

4.2.13 Delivered and Returned tabs interfaces

Figure 4.17 shows the “Delivered” tab interface that exhibits orders that have successfully processed by the delivery agency and synchronized with system API. Where Figure 4.18 shows “Returned” tab that exhibits orders that are returned by the delivery agency. These two interfaces are accessible to all users, but only admins can take actions on these tabs.



| | client Name | client Phone | wilaya | products (quantity) | total price | delivered at | delivery Agency |
|--------------------------|-------------|--------------|--------|---------------------|-------------|------------------|------------------|
| <input type="checkbox"/> | imad | 0677545461 | Skikda | AIR39203 (3) | 70500 | 6/18/23, 3:29 AM | ZR Express |
| <input type="checkbox"/> | mohamed | 0677545458 | Oran | AIR39203 (1) | 23500 | 6/18/23, 3:29 AM | Maystro Delivery |

Figure 4.17 Delivered tab interface

| | Add | In confirmation (9) | In preparation (26) | In dispatch (6) | In delivery (9) | Delivered (9) | Returned (1) | Cancelled (7) |
|--------------------------|-------------|---------------------|---------------------|---------------------|-----------------|------------------|------------------|---------------|
| <input type="checkbox"/> | client Name | client Phone | wilaya | products (quantity) | total price | returned at | delivery Agency | |
| <input type="checkbox"/> | mohamed | 0677545458 | Oran | AIR39203 (1) | 23500 | 6/18/23, 3:36 AM | Maystro Delivery | ▼ |

Figure 4.18 Returned Tab interface

4.2.14 Cancelled tab interface

This interface exhibits the cancelled orders. As it is shown in Figure 4.19, this tab has “Return back to confirmation” that send selected orders to confirmation status for retargeting purpose. This interface can be accessed by all users.

| | Add | In confirmation (9) | In preparation (26) | In dispatch (6) | In delivery (9) | Delivered (9) | Returned (1) | Cancelled (97) |
|--------------------------|-------------|---------------------|---------------------|--|-----------------|------------------|-----------------|----------------|
| <input type="checkbox"/> | client Name | client Phone | wilaya | products (quantity) | total price | cancelled at | delivery Agency | |
| <input type="checkbox"/> | yahia | 0677545456 | Alger | AIR39203 (1) | | 6/15/23, 7:18 PM | | ▼ |
| <input type="checkbox"/> | yacine | 0677545457 | Alger | AIR39203 (2) | | 6/15/23, 7:18 PM | | ▼ |
| <input type="checkbox"/> | karim | 0677545463 | Ain Defla | AIR39203 (1),AF433245 (1),XS9033-399 (2) | | 6/15/23, 7:18 PM | | ▼ |
| <input type="checkbox"/> | mohamed | 0677545458 | Oran | AIR39203 (1) | | 6/15/23, 7:18 PM | | ▼ |
| <input type="checkbox"/> | sami | 0677545462 | Chief | AIR39203 (2) | | 6/15/23, 7:18 PM | | ▼ |
| <input type="checkbox"/> | karim | 0677545463 | Ain Defla | AIR39203 (1) | | 6/15/23, 7:18 PM | | ▼ |
| <input type="checkbox"/> | yahia | 0677545456 | Alger | AIR39203 (1) | | 6/15/23, 7:18 PM | | ▼ |
| <input type="checkbox"/> | yacine | 0677545457 | Alger | AIR39203 (2) | | 6/15/23, 7:18 PM | | ▼ |
| <input type="checkbox"/> | said | 0677545459 | Oran | AIR39203 (1) | | 6/16/23, 9:52 PM | | ▼ |
| <input type="checkbox"/> | wahab | 0677545460 | Setif | AIR39203 (4) | | 6/16/23, 9:52 PM | | ▼ |
| <input type="checkbox"/> | yahia | 0677545456 | Alger | AIR39203 (1) | | 6/16/23, 9:52 PM | | ▼ |
| <input type="checkbox"/> | said | 0677545459 | Oran | | | 6/15/23, 7:39 PM | | ▼ |
| <input type="checkbox"/> | sami | 0677545462 | Chief | AIR39203 (2) | | 6/17/23, 6:13 PM | | ▼ |

Figure 4.19 Cancelled tab interface

4.3 Conclusion

In this chapter, we have given a brief explanation about the various user interfaces of our application’s front-end, and how different users may interact with the system, access the necessary information, and submit the required data in a simplified and convenient way so that different tasks and scenarios will be accomplished effectively.

General Conclusion

In this report, we have presented the different steps to design and develop a web application intended for managing the order fulfilment process of cash-on-delivery system. In order to implement this project, we have gathered analyzed different problems encountered by the staff of a local online store.

The main objective of our project is to facilitate and manage the tasks of staff by centralizing the data in one application including the security of data and the rapid access through a friendly user interface to do the required actions in simple way. We have implemented a web application, so it can be accessed from anywhere at any time using internet.

This project provided me with an excellent opportunity to handle new web technologies and concepts, as well as to expand my understanding of programming in general, and web development in particular.

Future Work

For more completion of the work of this application and making it fulfill the requirement of the order management system, many things can be added:

The expansion of the application work to automate and synchronize the import of orders from many sources like an e-commerce websites or Facebook leads system. The system end points that connects with a delivery agency can be improved to match many local delivery agencies' systems. Finally, inventory and stock management system can be added to this application.

Bibliography

- [1] "techterms," [Online]. Available: https://techterms.com/definition/web_application. [Accessed 5 June 2023].
- [2] "medium," [Online]. Available: <https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58>. [Accessed 5 June 2023].
- [3] "web client," [Online]. Available: <https://www.pcmag.com/encyclopedia/term/web-client>. [Accessed 6 June 2023].
- [4] "web server," [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server. [Accessed 6 June 2023].
- [5] "An overview of HTTP," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>. [Accessed 6 June 2023].
- [6] "what is URL?," [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL. [Accessed 6 June 2023].
- [7] "What is REST API?," [Online]. Available: <https://www.techtarget.com/searcharchitecture/definition/RESTful-API>. [Accessed 6 June 2023].
- [8] "IntelliJ IDEA overview," [Online]. Available: <https://www.jetbrains.com/help/idea/discover-intellij-idea.html>. [Accessed 6 June 2023].
- [9] "Documentation for Visual Studio Code," [Online]. Available: <https://code.visualstudio.com/docs>. [Accessed 06 June 2023].
- [10] "What is spring boot," [Online]. Available: <https://www.ibm.com/topics/java-spring-boot>. [Accessed 6 June 2023].
- [11] "Angular - Introduction to Angular concepts," [Online]. Available: <https://angular.io/guide/architecture>. [Accessed 6 June 2023].
- [12] "What is a web browser?," [Online]. Available: <https://www.techopedia.com/definition/288/web-browser>. [Accessed 6 June 2023].
- [13] "What is Java?," [Online]. Available: <https://www.guru99.com/java-platform.html>. [Accessed 6 June 2023].
- [14] "HTML," [Online]. Available: <https://www.britannica.com/technology/HTML>. [Accessed 6 June 2023].
- [15] [Online]. Available: https://www.w3schools.com/css/css_intro.asp. [Accessed 6 June 2023].

- [16] "JavaScript - Overview," [Online]. Available: http://www.tutorialspoint.com/javascript/javascript_overview.htm. [Accessed 4 June 2023].
- [17] "Bcrypt," [Online]. Available: <https://en.wikipedia.org/wiki/Bcrypt>. [Accessed 3 June 2023].
- [18] "JSON web token introduction," [Online]. Available: <https://jwt.io/introduction/>. [Accessed 3 June 2023].
- [19] "Token Based Authentication made easy," [Online]. Available: <https://auth0.com/learn/token-based-authentication-made-easy>. [Accessed June 2023].
- [20] P. P.Chen, "The Entity-Relationship Model—Toward a Unified View of Data," *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9-36, 1976.
- [21] G. Booch, J. Rumbaugh and I. Jacobson, *The Unified Modeling Language User Guide 2nd edition*, 2005.
- [22] "Introduction to the Diagrams of UML," [Online]. Available: <http://agilemodeling.com/essays/umlDiagrams.htm>. [Accessed 24 May 2023].
- [23] "Use case diagrams," [Online]. Available: <https://www.smartdraw.com/use-case-diagram/>. [Accessed June 2023].
- [24] "What is use case diagram?," [Online]. Available: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>. [Accessed June 2023].
- [25] "Database," [Online]. Available: <https://www.britannica.com/technology/database>. [Accessed June 2023].
- [26] "Non-relational data and NoSQL," [Online]. Available: <https://learn.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data>. [Accessed June 2023].
- [27] "What is a relational database?," [Online]. Available: <https://aws.amazon.com/relational-database/>. [Accessed May 2023].
- [28] "What is PostgreSQL?," [Online]. Available: <https://www.postgresql.org/docs/current/intro-what-is.html>. [Accessed May 2023].