  
Parcel Delivery Tracking

Requirement Specification Document

*Table of Contents*

[Overall Description 2](#_Toc1853540203)

[Purpose 2](#_Toc1664312152)

[Intended Audience 2](#_Toc1848542560)

[Intended Use 3](#_Toc440894686)

[Scope 4](#_Toc1718608766)

[Definition and Acronyms 4](#_Toc239601368)

[User Stories 4](#_Toc563399963)

[Project Backlog Item 6](#_Toc1389606221)

[Entity Relationship Diagram 7](#_Toc1838233237)

[Table Descriptions 8](#_Toc1078387959)

[Use Case Diagram 9](#_Toc1478323390)

[System Features 10](#_Toc171714810)

[Functional Requirements 11](#_Toc1176374663)

[Non-Functional Requirements 11](#_Toc313327447)

[Security Log in 11](#_Toc220740789)

[User Registration and Login 11](#_Toc1968907980)

[Technology Stack 12](#_Toc810395737)

[API Swagger Endpoint List 14](#_Toc376125809)

[Parcel Information Endpoints 14](#_Toc694430099)

[Personnel Information Endpoints 15](#_Toc627619742)

[Delivery Information Endpoints 16](#_Toc826110153)

[Wireframe 17](#_Toc1440411487)

[Home Page 17](#_Toc2083893390)

[Admin Page 18](#_Toc1459760682)

[Driver page 18](#_Toc1071722428)

[Log in and Registration Page 19](#_Toc1677048632)

[Lesson Learned 19](#_Toc712409072)

# Overall Description

Parcel Delivery Tracking System is a comprehensive software application designed to streamline and enhance the logistics and delivery processes for a delivery company. This system is built to cater to a diverse clientele, including businesses, shops, warehouses, and individual customers, who require efficient parcel tracking and delivery services.

# Purpose

The purpose of the Parcel Delivery Tracking System is to provide a comprehensive and efficient software solution for managing parcel tracking and delivery logistics for a delivery company. This system aims to streamline the entire parcel lifecycle, from creation to delivery, while ensuring ease of use.

# Intended Audience

The intended audience for the Parcel Delivery Tracking System include:

* **Delivery Company**: The system is designed for the delivery company, enabling it to optimise its parcel management, delivery operations and client interaction.
* **Clients**: Businesses, Shops, Warehouses, and individual customers who require parcel delivery services are the primary end-users of the system. Clients can request parcel deliveries and track the status of their parcels.
* **Delivery Personnel**: Delivery drivers, managers, other personnel responsible for handling deliveries will use the system to access delivery details, update statuses.
* **Administrators**: System administrators will have access to administrative features for user managements.

# Intended Use

The intended use of the Parcel Delivery Tracking System is to:

* Facilitate clients in requesting parcel deliveries providing parcel details, and tracking parcel statuses.
* Allow delivery personnel to access delivery assignments, update delivery statuses, and manage their availability.

# Scope

The scope of the Parcel Delivery Tracking System includes the following key functionalities:

* **Parcel Management**: Creation, update, and tracking of parcels, including sender and receiver information, parcel weights, statuses, and delivery dates.
* **Client Management**: Registration, profile management.
* **Delivery Management**: Assignment of delivery personnel to parcel, real-time status update, and tracking of deliveries.

# Definition and Acronyms

* EF – Entity Framework
* ERD – Entity Relationship Diagram
* API – Application Programming Interface
* DB/Db – Database
* ORM – Object Relational Mapper
* DBMS – Database Management System
* SDK – Software Development Kit

# User Stories

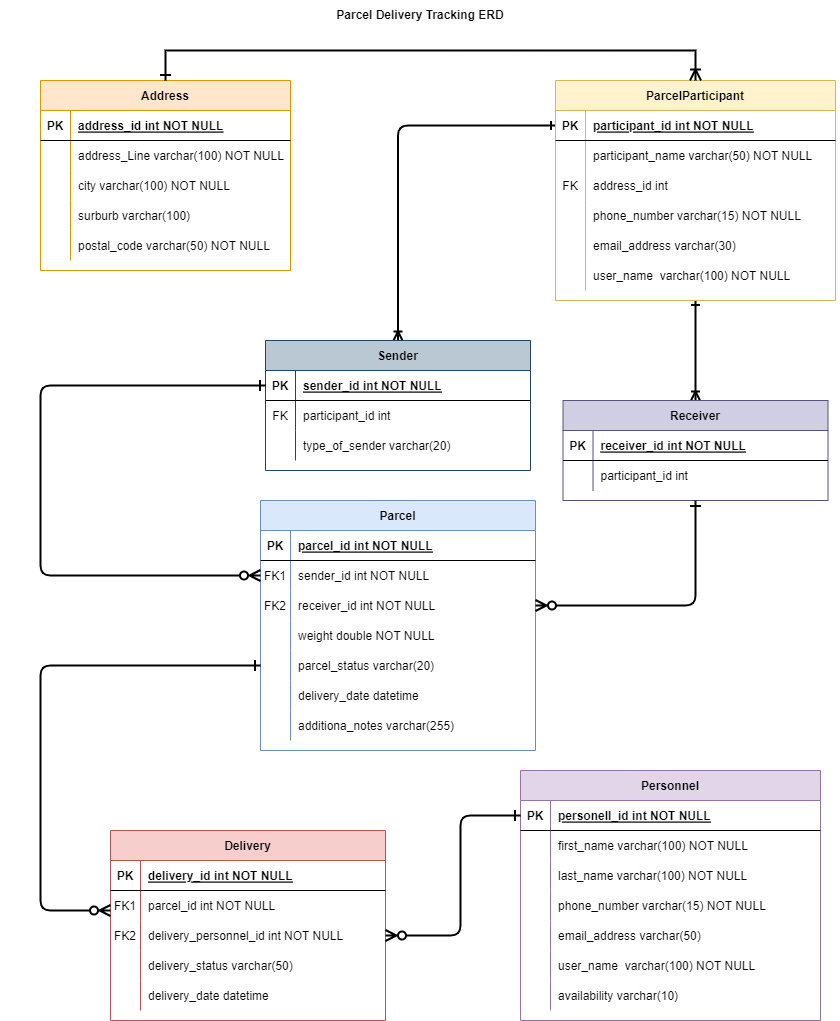
|  |  |  |
| --- | --- | --- |
| As a <User> | I want <Something> | So, I can <Benefit> |
| As an administrator | I want to view a list of all deliveries | So, I can have an overview of current delivery status. |
|  | I want to delete a delivery | So, I can remove erroneous or unnecessary delivery records from the system |
|  | I want to update delivery details | So, I can correct any inaccuracies or make necessary changes to delivery information |
|  | I want to retrieve information about a specific parcel | So, I can monitor the entire parcel inventory |
|  | I want to delete a parcel | So, I can remove parcel that are no longer relevant or in stock |
|  | I want to update parcel details | So, I can keep parcel information up to date |
|  | I want to retrieve information about specific personnel | So, I can have access to their details |
|  | I want to view a list of all personnel | So, I can manage and monitor personnel within the system |
|  | I want to delete a personnel record | So, I can remove personnel who are longer associated with the organisation |
|  | I want to update personnel details | So, I can keep personnel information accurate and current |
| As a manager |  |  |
|  | I want to view a list of all deliveries | So, I can oversee the progress and status of deliveries |
|  | I want to retrieve information about a specific delivery | So, I can quickly access details about a particular delivery when needed |
|  | I want to update delivery details | So, I can make necessary adjustments or corrections to delivery information as required. |
|  | I want to retrieve information about a specific personnel member | So, I can access their details when necessary |
|  | I want to view a list of all personnel | So, I can manage and monitor the personnel under my supervision |
|  | I want to update personnel details | So, I can ensure that personnel information remain accurate and up to date |
| As a driver | I want to view a list of all deliveries assigned to me | So, I can see which deliveries I need to complete |
|  | I want to update the status of deliveries I’m responsible for | So, I can indicate when a delivery has been completed or if there are any issues. |

# Project Backlog Item

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | User | Item | Priority | Status |
| 1 | Admin | Read a list of deliveries | 1 | Not Started |
| 2 | Admin | Read a list of parcels | 1 | Completed |
| 3 | Admin | Read a list of personnel | 1 | Not Started |
| 4 | Admin | Read a list of parcels by parcel status (Delivered/In Transit/On hold) | 1 | Completed |
| 5 | Admin | Remove a Parcel from the system | 3 | Completed |
| 6 | Admin | Update a Delivery, Parcel, or Personnel | 3 | Not Started |
| 7 | Manager | Read a list of deliveries | 1 | Not Started |
| 8 | Manager | Read a list of deliveries | 1 | Not Started |
| 9 | Manager | Create a new delivery – assign a driver | 1 | Not Started |
| 10 | Manager | Input deliveryID, display delivery details | 1 | Not Started |
| 11 | Manager | Read a list of by delivery status (Complete/In Progress/Scheduled) | 2 | Not Started |
| 12 | Driver | Read a list of deliveries, assigned to me | 1 | Not Started |
| 13 | Driver | Update delivery status | 1 | Not Started |
| 14 |  | Change the ID’s data type from int to guid for security | 2 | Not Started |
| 15 | Sender | Read user account details and deliveries associated to the account | 3 | Not Started |
| 16 | receiver | Read user account details and deliveries associated to the account | 3 | Not Started |
| 17 |  | Read User Details to their respective page | 3 | Completed |
| 18 |  | Add an endpoint to get user details | 2 | Completed |
| 19 |  | Add an endpoint to get user full name | 3 | Completed |

# Entity Relationship Diagram

The major features of the parcel delivery tracking database system as shown in below ERD.



## Table Descriptions

1. Address Table:

* Contains information about addresses including address line, city, suburb, and postal. code. The table will be used to store address details for parcel participants (sender/receiver).

1. Parcel Participations Table:

* Represents participants (either sender or receiver) involved in parcel transactions.
* References Address table to store address information.
* Include fields for participant name, phone number, and email address.

1. Sender Table:

* Represent senders of parcels.
* References the ParcelParticipant table for sender details.
* Includes a “type” field, which can specify the type of sender (e.g., individual, business).

1. Receiver Table:

* Represent receivers of parcels.
* References the ParcelParticpartion Table for receive details.

1. Parcel Table:

* Contain information about parcels, including details such as sender, receiver, parcel weight, status, delivery date, and additional notes.
* References the Sender and Receiver Tables to establish relationships.

1. Personnel Table:

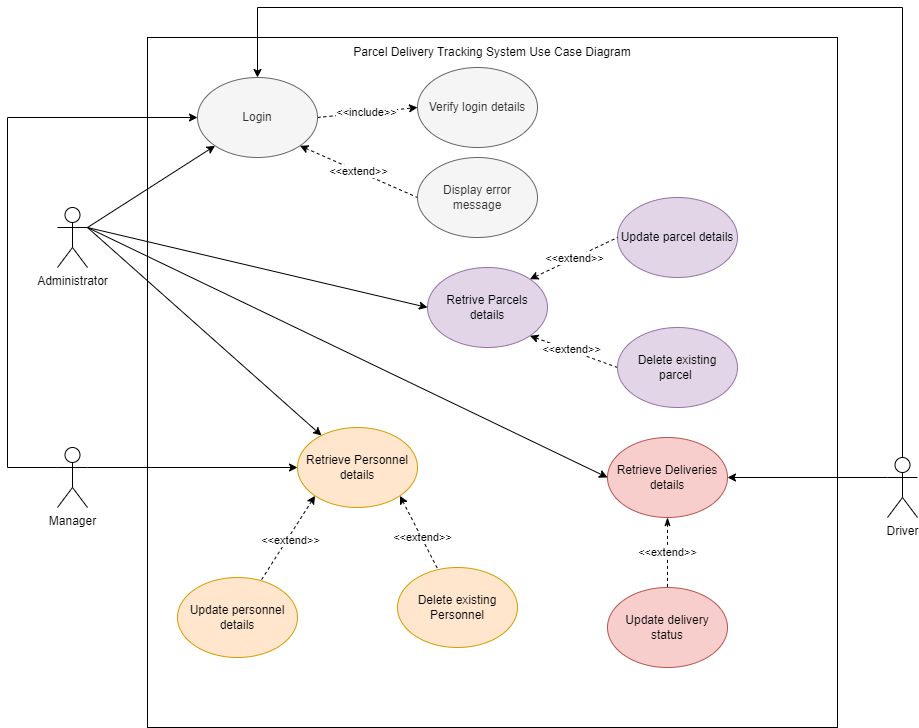
* Represents personnel involved in parcel delivery.
* Incudes fields for first name, last name, phone number, email address, role, and availability.

1. Delivery Table:

* Represents delivery transactions.
* References the Parcel Table for the parcel being delivered and the Personnel table for the delivery personnel.
* Includes fields for delivery status and deliver date.

# Use Case Diagram

The use case diagram of the parcel delivery tracking application is shown below.



# System Features

## Functional Requirements

* **Parcel Management**: The system allows for the efficient management of parcels, capturing details such as sender and receiver information, parcel weight, and status (In Transit, Delivered, On Hold, Cancelled)
* **Client Management**: Client, whether businesses or individuals, can register and access the system. They can maintain their profiles, including contact information and relevant details.
* **User Authentication**: The system incorporates user authentication and role-based access control. Clients register as users with unique usernames and passwords, allowing them to log in to access their accounts and request parcel deliveries.
* **Parcel Tracking:** Clients can request parcel deliveries by providing parcel details. They can track the status of their parcel in real-time
* **Delivery Management:** The system assigns delivery personnel based on availability. Delivery personnel update the delivery status as they complete deliveries or encounter issues.

## Non-Functional Requirements

* **Security**: The system incorporates user authentication and authorization and role-based access control.
* **Monitoring and Logging**: The system log relevant events, errors, and user activities for auditing and troubleshooting purposes.
* **Data Integrity:** Data stored in the application database should maintain its accuracy, and consistency.

### Security Log in

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Security |  |  |  |
|  |  | Username | Password | Role |
|  | Admin | AdminOne | Admin123 | Administrator |
|  | Manager | ManagerOne | Manager123 | Manager |
|  | Driver | DriverOne | Driver123 | Driver |

### User Registration and Login

|  |  |
| --- | --- |
| Registration | Login |
| {  "userName": "LorrySmith",  "email": "Smith.Lorry@gmail.com",  "password": "Lo@134Sm",  "firstName": "Lorry",  "lastName": "Smith",  "role": "Administrator"  } | Login  {  "userName": "LorrySmith",  "password": "Lo@134Sm"  } |

## Technology Stack

|  |  |  |
| --- | --- | --- |
| Backend |  |  |
|  | Technology | Package and Version |
|  | Programming language | C# |
|  | ORM Framework | Entity Framework 6.4.4 |
|  | Authentication Middleware | Microsoft.AspNetCore.Authentication.JwtBearer 6.0.1 |
|  | Identity Framework | Microsoft.AspNetCore.Identity.EntitFramework 6.0.1 |
|  | Identity UI | Microsoft.AspNetCore.Identity.UI 6.0.1 |
|  | Database ORM | Microsoft.EntityFrameworkCore 6.0.1 |
|  | Database Design Tool | Microsoft.EntityFrameworkCore.Design 6.0.1 |
|  | Database Provider | Microsoft.EntityFrameworkCore.SqlServer 6.0.1 |
|  | EF Tools | Microsoft.EntityFrameworkCore.Tools 6.0.1 |
|  | Code Generation Design | Microsoft.VisualStudio.Web.CodeGeneration.Design 6.0.1 |
|  | API Documentation | Swashbuckle.Asp.NetCore 6.2.3 |
|  | JWT Token Management | System.IdentityModel.Token.Jwt 6.15.1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Frontend |  |  |  |
|  | Technology | Package and Version | Path |
|  | Web Technologies | JavaScript, HTML, and CSS |  |
|  | HTTP Request | Axios |  |

|  |  |  |
| --- | --- | --- |
| Database |  |  |
|  | Technology | Package and Version |
|  | Microsoft SQL Server Management Studio |  |

|  |  |  |
| --- | --- | --- |
| Testing |  |  |
|  | Technology | Package and Version |
|  | In-Memory Db Testing | Microsoft.EntityFrameworkCore.InMemory 7.0.11 |
|  | Unit Testing Framework | Nunit 3.13.3 |
|  | Testing SDK | Nunit3TestAdapter 4.2.1 |
|  | Nunit Analyzer | Nunit.Analyzer 3.3.0 |

# API Swagger Endpoint List

## Parcel Information Endpoints

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | HTTP Method | Endpoint: Parcels | Description |
|  | GET | https://localhost:7200/api/Parcels | Get all the parcels |
|  | POST | https://localhost:7200/api/Parcels | Post Parcel |
|  | GET | https://localhost:7200/api/Parcels/1 | Get a Parcel by ID. Return Parcel |
|  | PUT | https://localhost:7200/api/Parcels/17 | Put parcel by id. Returns the updated parcel details |
|  | DELETE | https://localhost:7200/api/Parcels/17 | Delete a parcel by Id. Delete parcel Id |
|  | GET | https://localhost:7200/api/Parcels/Delivered | Get all the parcels with a parcel status “Delivered” |
|  | GET | https://localhost:7200/api/Parcels/InTransit | Get all the parcels with a parcel status “In Transit” |
|  | GET | https://localhost:7200/api/Parcels/OnHold | Get all the parcels with a parcel status “On Hold” |
|  | GET | https://localhost:7200/api/Parcels/3/parcels | Get all the parcels for a specific sender, by using the sender Id |

## Personnel Information Endpoints

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | HTTP Method | Endpoint: Personnel | Description |
|  | GET | https://localhost:7200/api/Personnel | Get all the Personnel |
|  | POST | https://localhost:7200/api/Personnel | Post new Personnel |
|  | GET | https://localhost:7200/api/Personnel/18 | Get a Personnel by ID. Return Personnel |
|  | PUT | https://localhost:7200/api/Personnel/18 | Put Personnel by id. Returns the updated personnel details |
|  | DELETE | https://localhost:7200/api/Personnel/18 | Delete Personnel by Id. Delete Personnel with ID 18 |
|  | GET | https://localhost:7200/api/Personnel/fullname?firstName=Mark&lastName=tomalson | Get a Personnel by first and last name. Returns Mark Tomalson |
|  | GET | https://localhost:7200/api/Personnel/offduty | Get all Personnel on duty |
|  | GET | https://localhost:7200/api/Personnel/offduty | Get all Personnel off duty |

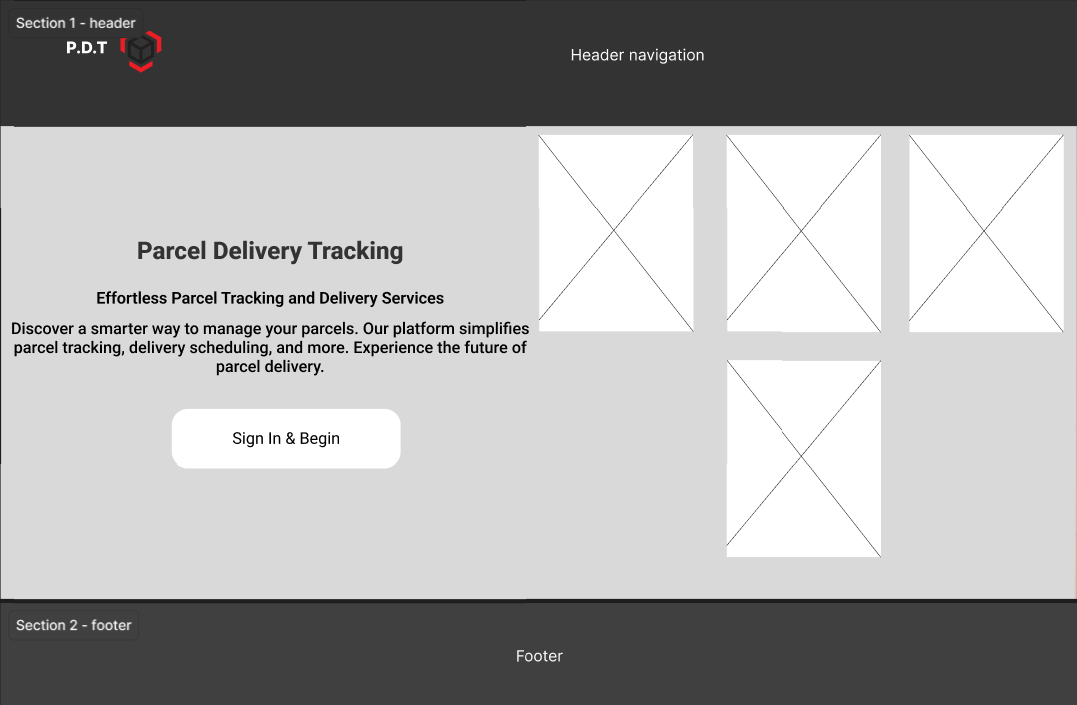
## Delivery Information Endpoints

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | HTTP Method | Endpoint: Personnel | Description |
|  | GET | https://localhost:7200/api/Deliveries | Get all the Deliveries |
|  | GET | https://localhost:7200/api/Deliveries/9 | GET a Delivery by Id |
|  | PATCH | https://localhost:7200/api/Deliveries/9 | PATCH Delivery by Id. Update the personnel and delivery status |
|  | GET | https://localhost:7200/api/Deliveries/InProgress | Get all Deliveries with a delivery status “Progress” |
|  | GET | https://localhost:7200/api/Deliveries/Completed | Get all Deliveries with a delivery status “Completed” |
|  | GET | https://localhost:7200/api/Deliveries/Scheduled | Get all Deliveries with a delivery status “Scheduled” |
|  | GET | https://localhost:7200/api/Deliveries/1/deliveries | Get all Deliveries for a specific personnel |

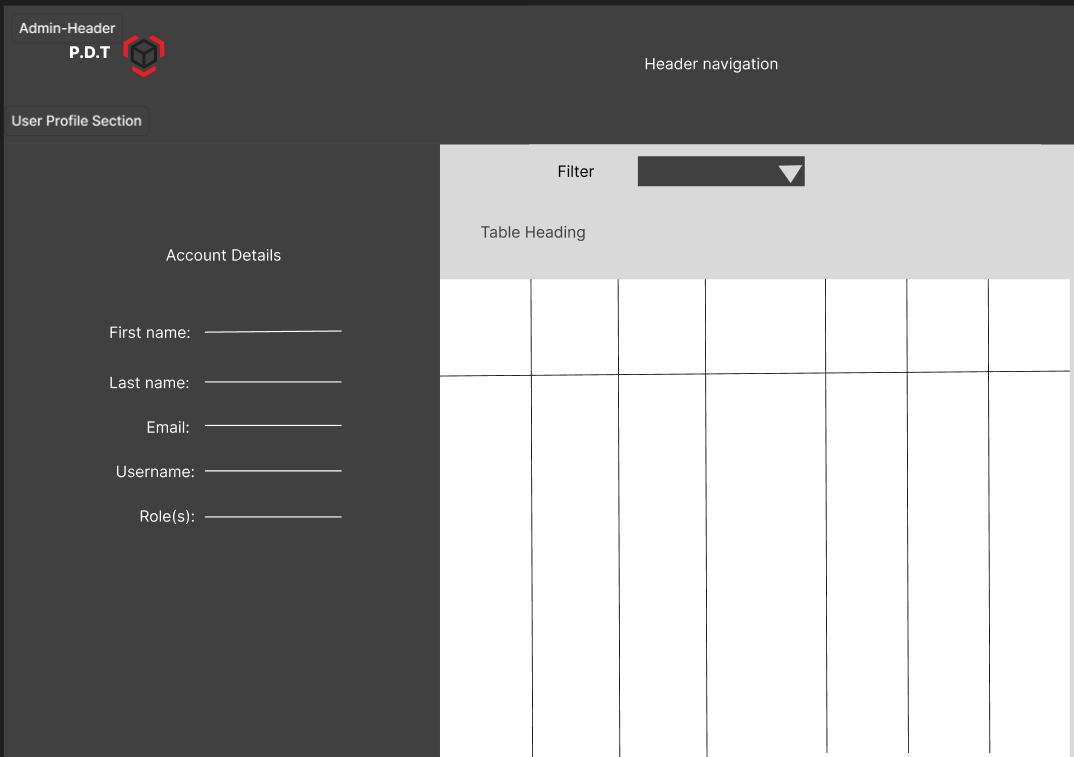
# Wireframe

## Home Page

The home page serves as the entry point for users. It should provide a welcoming and intuitive interface with easy access to essential features and information.

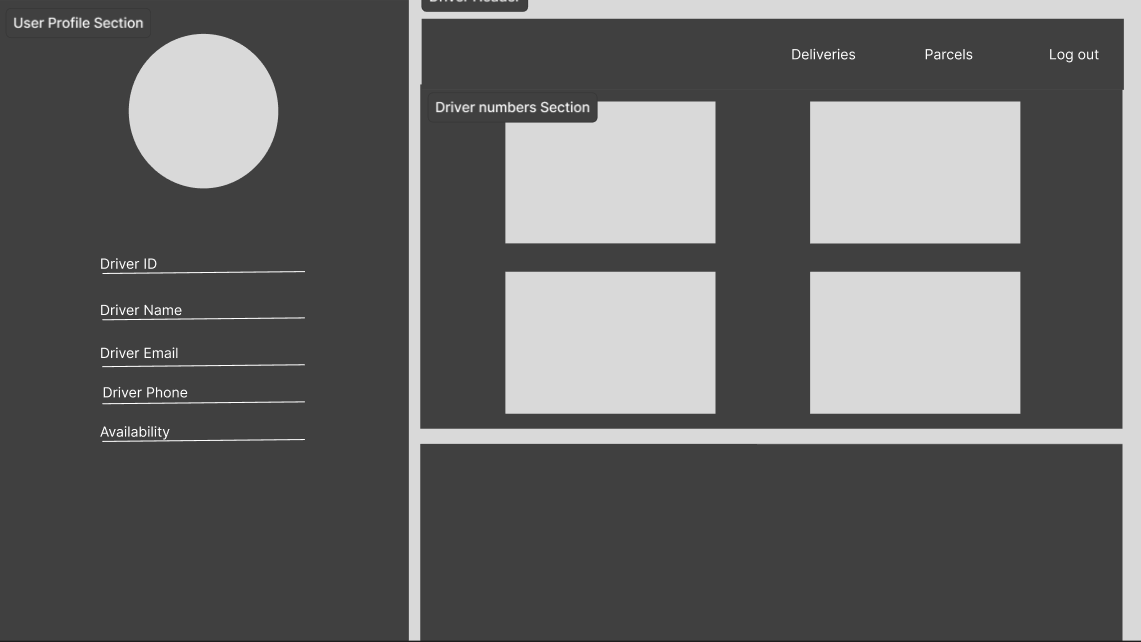


## Admin Page

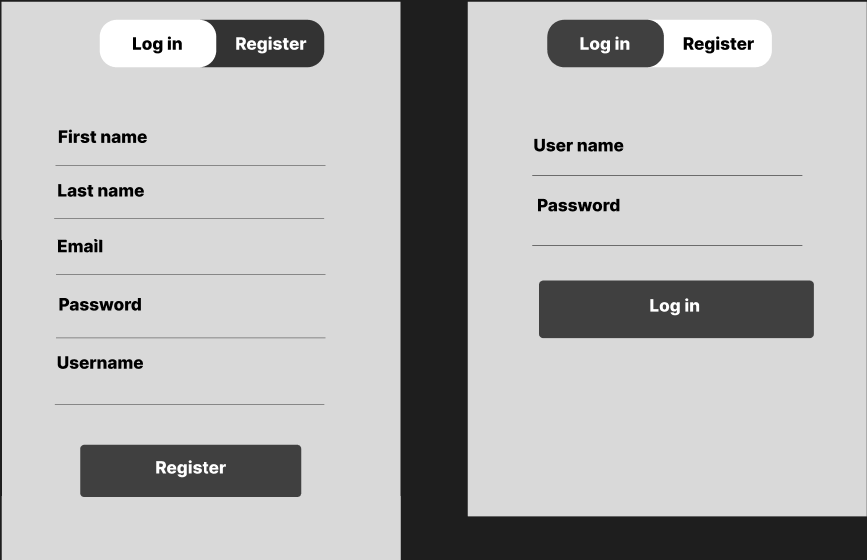
This page is accessible only to administrators and provides tools for user management and system administration.

## Driver page

This page is accessible only to drivers and provide tool for driver such as updating the status of a delivery and view all the deliveries associated to a driver.



## Log in and Registration Page



# Lesson Learned

* I learned how to build a full stack application – In this case JavaScript. HTML, and CSS I used for the frontend, while C#, ASP.NET Web API, and EF were selected for the backend
* HTML and CSS for UI – I learned how to use HTML and CSS to create user interfaces.
* JavaScript Event handling –I learned how to use JavaScript event handling. Event listeners, such as click events, that I used to trigger actions in response to user input.
* JWT Token handling – I learned how to manage JWT token securely by storing and clearing them in the local storage.
* Debugging: I learned how to use the browser developer tools and use console logs for debugging to identify and address issues in my code.