Morroco Drive - A Cab Booking Application

Software Engineering

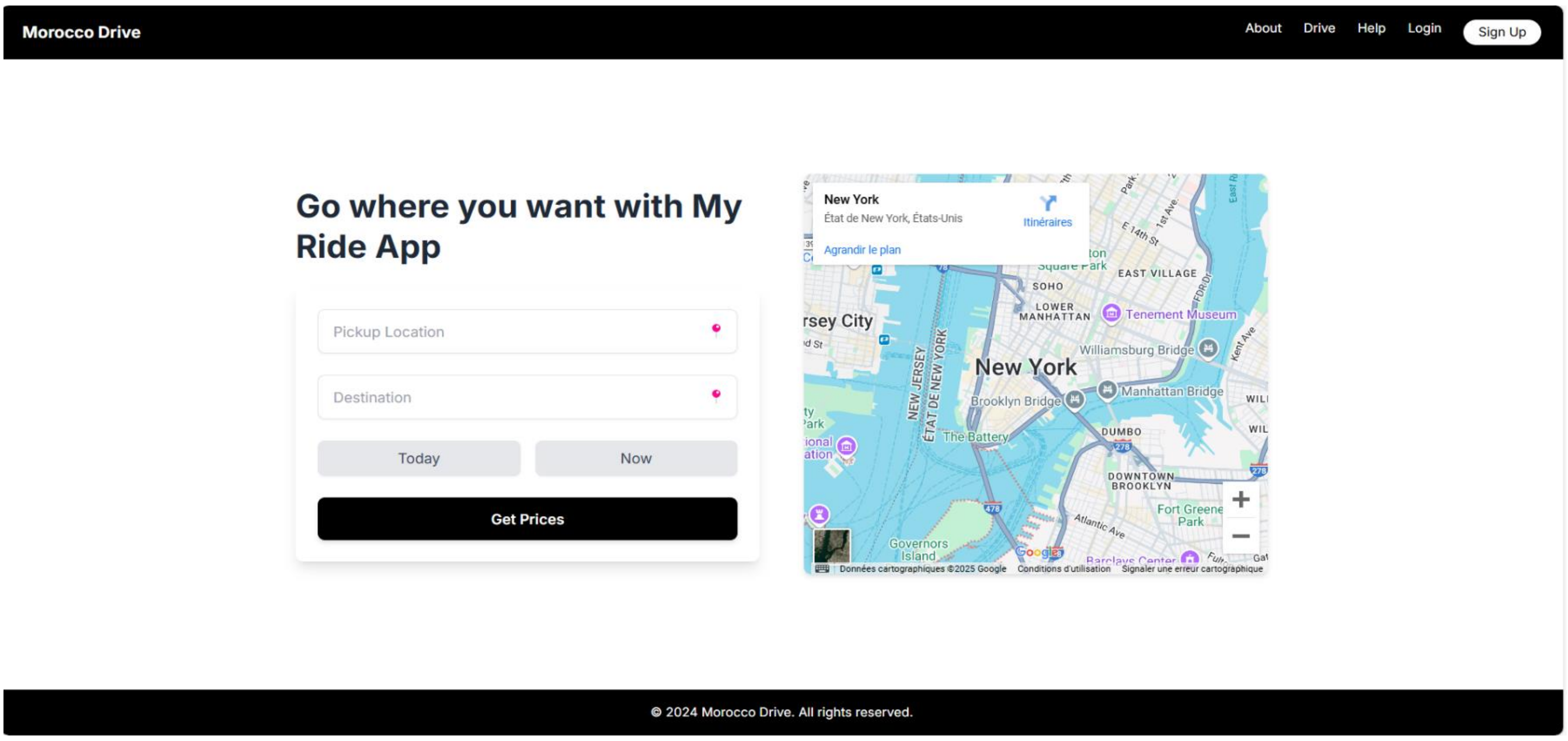
Project Report

Prepared By

- **Zyad Fri**
- **Yassir Bousseta**
- **Jaafar Yeffou**
- **Samia Lachgar**

Instructor:

- **Amine ABouaomar**



Introduction

Morocco Drive is a sophisticated cab booking application developed to provide users with an easy and efficient way to book a cab by simply specifying their pickup and destination locations. Built on a robust backend using Java and Spring Boot, and paired with an intuitive frontend developed in Next.js and TailwindCSS, the application ensures a seamless and responsive user experience. This project demonstrates the ability to develop a full-stack application with modern technologies and frameworks, addressing real-world challenges in transportation management.

Objectives

The primary objectives of the Morroco Drive application are:

- To provide users with a seamless platform for booking cabs.
- To enable drivers to manage rides efficiently.
- To utilize modern web technologies for optimal performance and scalability.

Tech Stack

The development of Ride Fast leverages the following technologies:

Backend:

- **Languages and Frameworks:** Java, Spring Boot, Spring Security, Spring Data JPA.
- **Authentication and Security:** JWT Authentication.
- **Database:** MySQL.
- **Testing Tools:** Postman, JUnit, Mockito, TestContainers, RestAssured.
- **Containerization:** Docker.

Frontend:

- **Frameworks:** ReactJS, Next.js (v14).
- **Styling:** TailwindCSS, Material UI.
- **State Management:** Redux Toolkit.
- **Languages:** TypeScript.

System Architecture

The system is designed with a modular architecture, ensuring scalability and maintainability. The application consists of two main modules:

1. **Backend Module:** Responsible for handling APIs, user authentication, database management, and business logic.
2. **Frontend Module:** Responsible for user interface, client-side logic, and communication with the backend.

Implementation Details

❖ *Design Patterns Implementation*

Controller Package

1. RideController

Pattern Used: Facade Pattern

Description:

The `RideController` serves as a single interface to manage all ride-related operations, hiding the complexities of the underlying service implementations from the client. By using the Facade Pattern, the controller provides a unified API for managing rides.

Implementation:

- The controller handles REST endpoints such as `/rides`, `/rides/{id}`, and `/rides/book`.
- Internally, it delegates the logic to specific services like `RideServiceImpl`, ensuring separation of concerns.

```java
@RestController
@RequestMapping("/rides")
public class RideController {
    private final RideService rideService;

    @PostMapping("/book")
    public ResponseEntity<?> bookRide(@RequestBody RideRequest request) {
        RideResponse response = rideService.bookRide(request);
        return ResponseEntity.ok(response);
    }
}
```

Advantages:

- Simplifies interaction for clients by encapsulating the underlying complexities.

- Promotes a clean separation between the presentation layer and business logic.

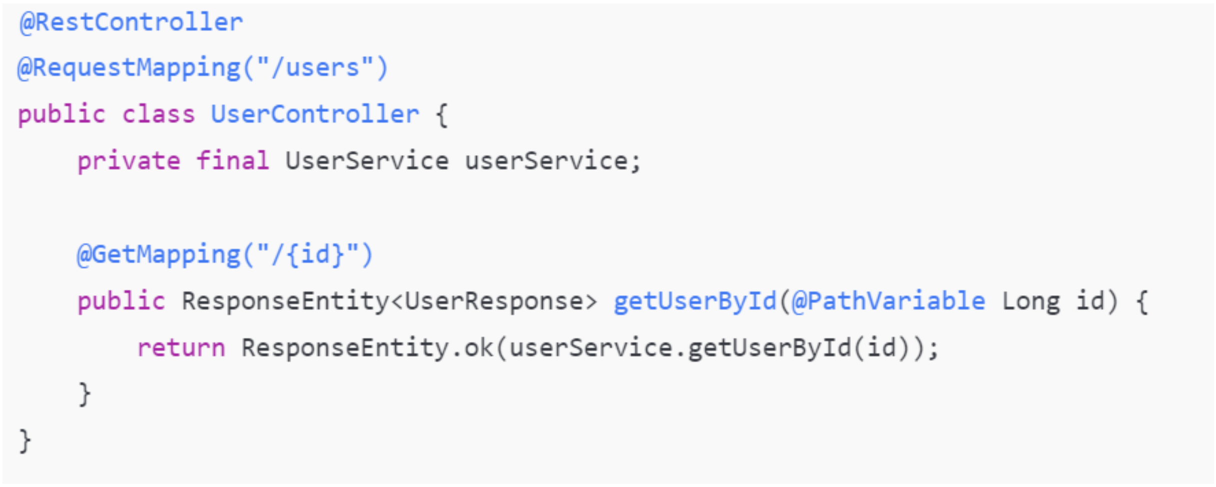
UserController

Pattern Used: **Singleton Pattern**

To ensure only a single instance of the UserController exists in the application context, the **Singleton Pattern** is applied. This guarantees thread safety and consistent behavior throughout the application lifecycle.

Implementation:

- Leveraged Spring's built-in **@RestController** and singleton-scoped beans to create a single instance of the controller.
- Methods include user-related operations like registration, login, and profile management.



```java
@RestController
@RequestMapping("/users")
public class UserController {
    private final UserService userService;

    @GetMapping("/{id}")
    public ResponseEntity<UserResponse> getUserById(@PathVariable Long id) {
        return ResponseEntity.ok(userService.getUserById(id));
    }
}
```

Advantages:

- Ensures a single instance is reused, reducing resource consumption.
- Prevents redundant instantiation of controllers.

❖ ***Service Implementation Package***

1. RideServiceImpl

Pattern Used: **Strategy Pattern**

The **Strategy Pattern** is used to dynamically switch between different ride calculation strategies (e.g., fare calculation for distance-based vs. time-based rides).

Implementation:

- A RideStrategy interface is defined with multiple implementations (**DistanceBasedRideStrategy, TimeBasedRideStrategy).**
- RideServiceImpl delegates the strategy dynamically based on ride type

```java
public class RideServiceImpl implements RideService {
    private RideStrategy rideStrategy;

    public void setRideStrategy(RideStrategy rideStrategy) {
        this.rideStrategy = rideStrategy;
    }

    @Override
    public RideResponse bookRide(RideRequest request) {
        double cost = rideStrategy.calculateCost(request);
        return new RideResponse(cost);
    }
}
```

-

2. NotificationServiceImpl

Pattern Used: **Observer Pattern**

The **Observer Pattern** is used to notify multiple subscribers (e.g., users, drivers) about the status of a ride.

Implementation:

- A NotificationService acts as a subject with methods to add, remove, and notify observers.
- Observers **(UserNotification, DriverNotification) i**mplement the Observer interface.

```java
public class NotificationServiceImpl implements NotificationService {
    private List<Observer> observers = new ArrayList<>();

    public void addObserver(Observer observer) {
        observers.add(observer);
    }

    public void notifyObservers(String message) {
        for (Observer observer : observers) {
            observer.update(message);
        }
    }
}
```

3. UserServiceImpl

Pattern Used: Factory Pattern

The **Factory Pattern** is used to create different types of users (e.g., `Driver`, `Passenger`) based on a common interface.

Implementation:

- A `UserFactory` class creates instances of users dynamically.

```java
public class UserFactory {
    public static User createUser(String userType) {
        if (userType.equals("DRIVER")) {
            return new Driver();
        } else if (userType.equals("PASSENGER")) {
            return new Passenger();
        }
        throw new IllegalArgumentException("Invalid user type");
    }
}
```

Summary of Benefits

- ✓ **Clean Architecture**:
- ✓ Each design pattern promotes separation of concerns and modularity.
- ✓ **Scalability**:
- ✓ The patterns make it easier to add new features (e.g., new user types, payment methods).
- ✓ **Readability**:
- ✓ Patterns such as Factory and Strategy simplify complex logic, making the codebase more maintainable.
- ✓ **Real-Time Updates**:
- ✓ The Observer Pattern ensures real-time notifications for users and drivers.
- ✓ **Consistency**:
- ✓ The Template Method Pattern enforces a consistent workflow for payment processing.

Software and Tools Required

To set up and run the Ride Fast application, the following software and tools are required:

- **Java Development Kit (JDK):** Version 17 or above.
- **Node.js**
- **Git**
- **MySQL Client**
- **Docker**
- **Integrated Development Environments (IDEs):** IntelliJ IDEA, Spring Tool Suite (STS), Eclipse, NetBeans, Visual Studio Code.

Installation

1. Clone the Git repository to your local machine:

   https://github.com/m-elhamlaoui/se-project-icode

Running the Backend

1. **Navigate to the Backend Directory:**
2. **Setup Database:** Update the application.yml file with your MySQL credentials and server configuration:

```yaml
server:
  port: 8080
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/ride_fast_db?createDatabaseIfNotExist=true
    username: root
    password: mysql
  jpa:
    hibernate:
      ddl-auto: update
```

3. **Run the Server:**

   ❖ ./mvnw spring-boot:run

Running the Frontend

1. **Navigate to the Frontend Directory:**

   ❖ cd ride_fast_frontend

2. **Install Dependencies:**

   ❖ npm install

3. **Update Proxy Configuration:** Update next.config.mjs to ensure API requests are routed to the backend:

```javascript
async rewrites() {
  return [
    {
      source: "/api/:path*",
      destination: "http://localhost:8080/api/:path*", // Replace with your backen
    },
  ];
},
```

4. **Run the Frontend:**

   ❖ npm run dev

❖ **API Endpoints**

The application provides a range of API endpoints for various functionalities. Below are some of the key endpoints:

User Management

- **Register User:**

```
POST /api/v1/auth/register/user
Params: { fullname, mobile, email, password }
```

- **Login User:**

```
POST /api/v1/auth/login
Params: { email, password, userType }
```

Driver Management

- **Register Driver:**

```
POST /api/v1/auth/register/driver
Params: { fullname, email, password, mobile, latitude, longitude, license details
```

- **Login Driver:**

```
POST /api/v1/auth/login
Params: { email, password, userType }
```

Ride Management

- **Book Ride:**

```
POST /api/v1/ride/request
Params: { pickupArea, destinationArea, coordinates, JWT Token }
```

- **Accept Ride:**

```
POST /api/v1/ride/accept
```

Complete Ride:

```
POST /api/v1/ride/complete
```

Responses

Success Responses

Example for User Login:

```json
{
  "statusCode": 200,
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR...",
  "refreshToken": "dfkjngfng4h5nf42sgh42s...",
  "message": "Got All Data Successfully",
  "success": true
}
```

Error Responses

Examples:

- **Missing Fields:**

```json
{
  "statusCode": 400,
  "message": "All fields are required",
  "success": false
}
```

Unauthorized Access:

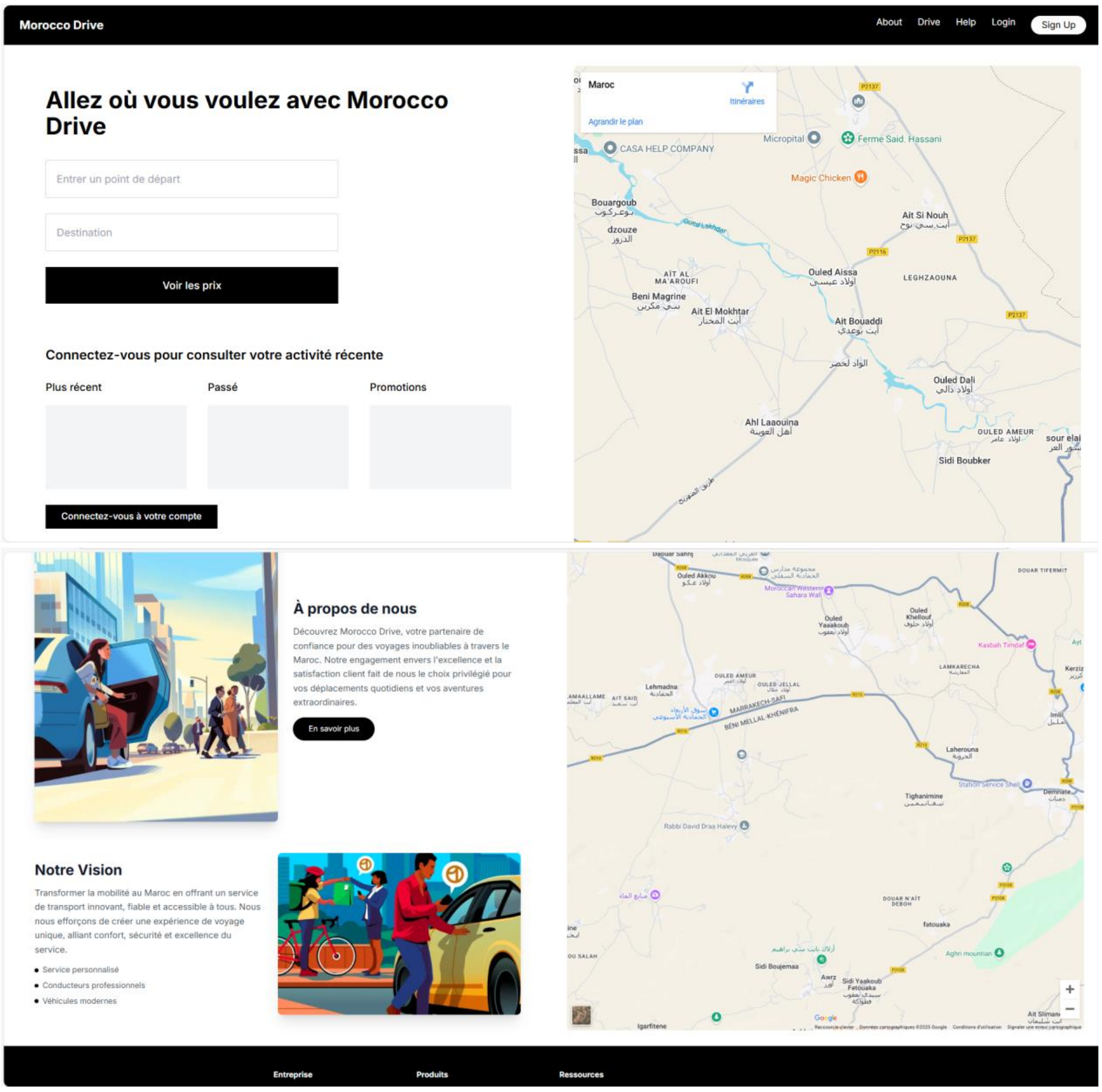
```json
{
  "statusCode": 401,
  "message": "You need to be logged in first",
  "success": false
}
```

Conclusion

Morocco Drive exemplifies the use of modern web development technologies to address a real-world problem. The project showcases best practices in API design, authentication, state management, and responsive UI development, offering a complete solution for cab booking and management.

SnapShots

❖ **The Auth Page**



Morocco Drive

About Drive Help Login Sign Up

Allez où vous voulez avec Morocco Drive

Entrer un point de départ

Destination

Voir les prix

Connectez-vous pour consulter votre activité récente

Plus récent | Passé | Promotions

Connectez-vous à votre compte

À propos de nous

Découvrez Morocco Drive, votre partenaire de confiance pour des voyages inoubliables à travers le Maroc. Notre engagement envers l'excellence et la satisfaction client fait de nous le choix privilégié pour vos déplacements quotidiens et vos aventures extraordinaires.

En savoir plus

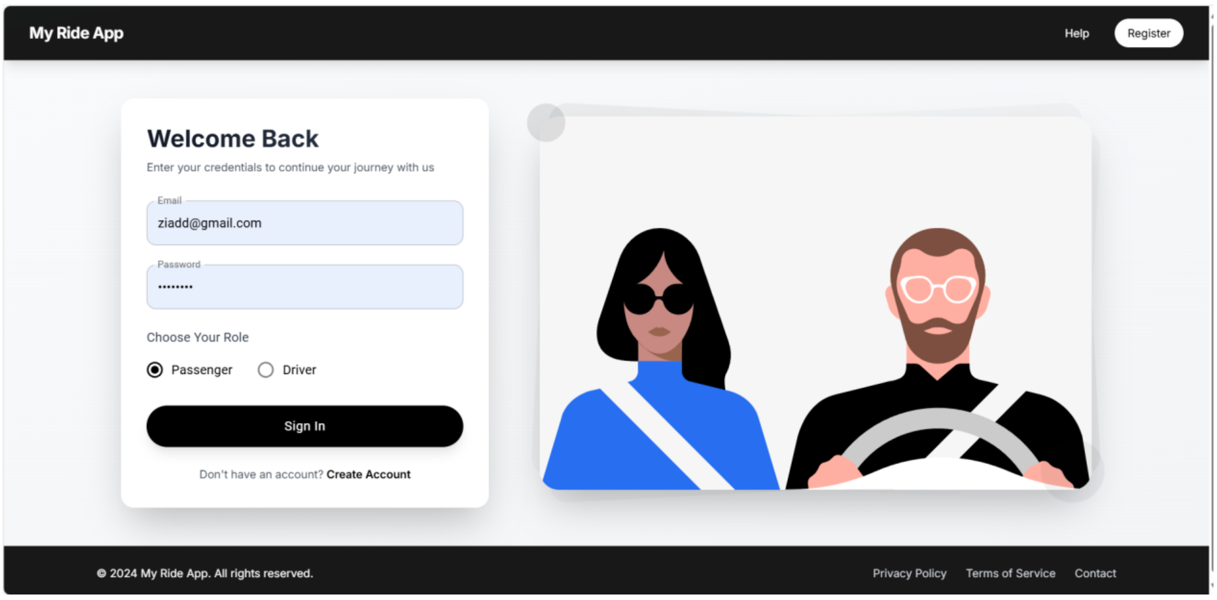
Notre Vision

Transformer la mobilité au Maroc en offrant un service de transport innovant, fiable et accessible à tous. Nous nous efforçons de créer une expérience de voyage unique, alliant confort, sécurité et excellence du service.

- Service personnalisé
- Conducteurs professionnels
- Véhicules modernes

Entreprise Produits Ressources

❖ The User Login/Register User Page



My Ride App

Help

Register

Welcome Back

Enter your credentials to continue your journey with us

Email: ziadd@gmail.com

Password: ••••••••

Choose Your Role

[✓ Passenger] Driver

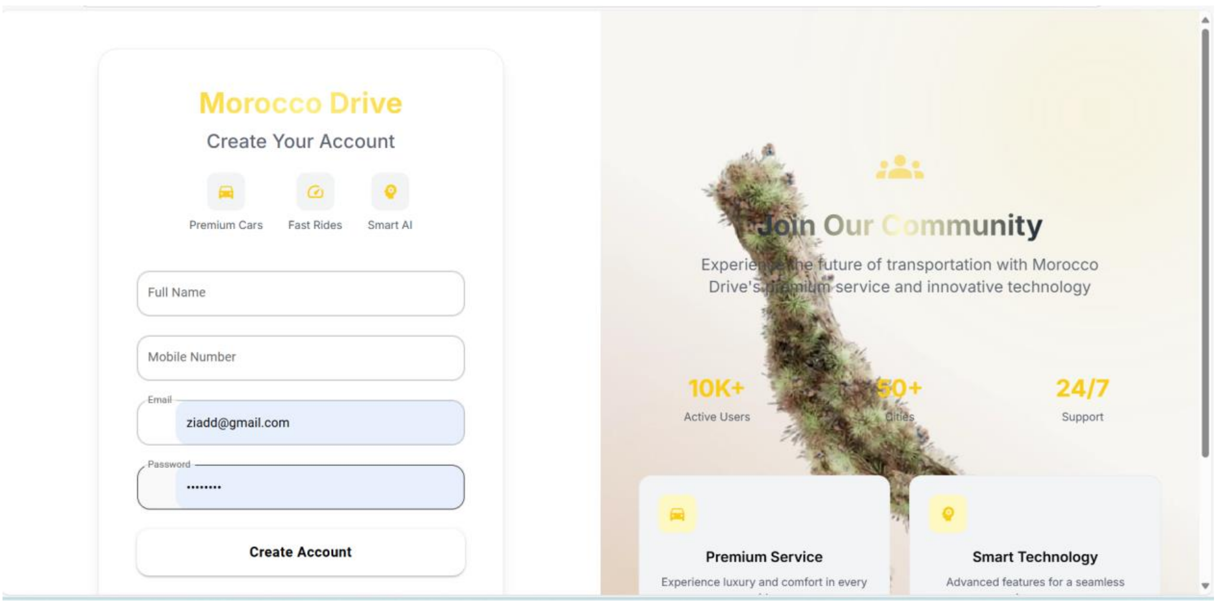
Sign In

Don't have an account? **Create Account**

© 2024 My Ride App. All rights reserved.

Privacy Policy Terms of Service Contact

❖ Driver Login/Register Form


Morocco Drive

Create Your Account

Full Name

Mobile Number

Email: ziadd@gmail.com

Password: ••••••••

Join Our Community

Experience the future of transportation with Morocco Drive's premium service and innovative technology

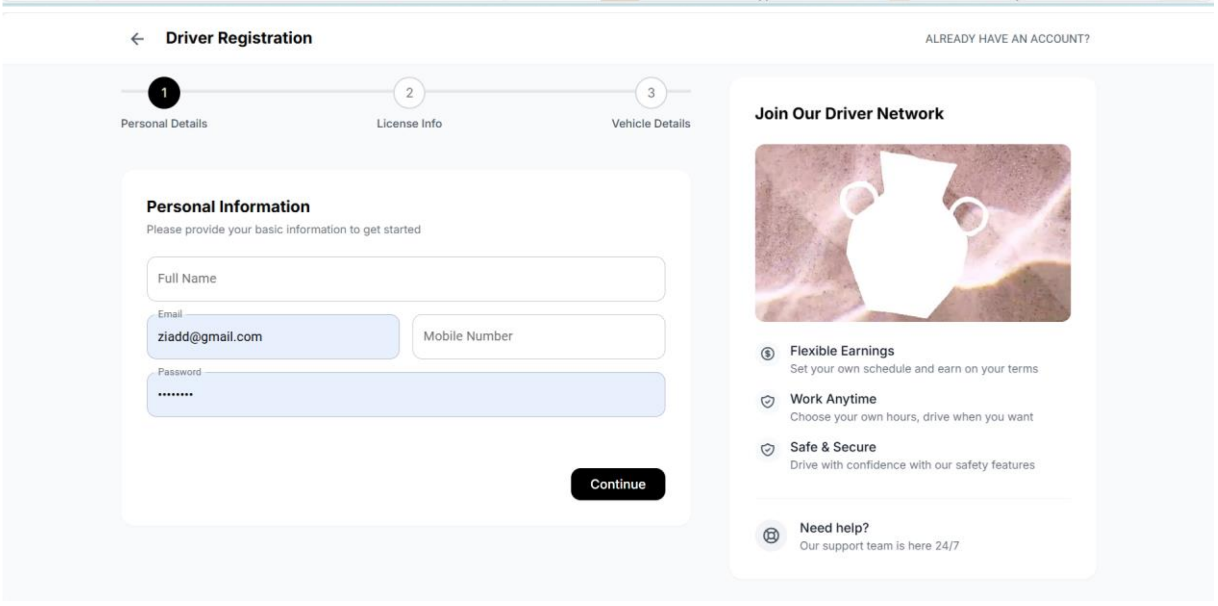
10K+	50+	24/7
Active Users	Cities	Support

Premium Service

Experience luxury and comfort in every

Smart Technology

Advanced features for a seamless



Personal Information

Please provide your basic information to get started

Full Name

Email: ziadd@gmail.com

Mobile Number

Password: ••••••••

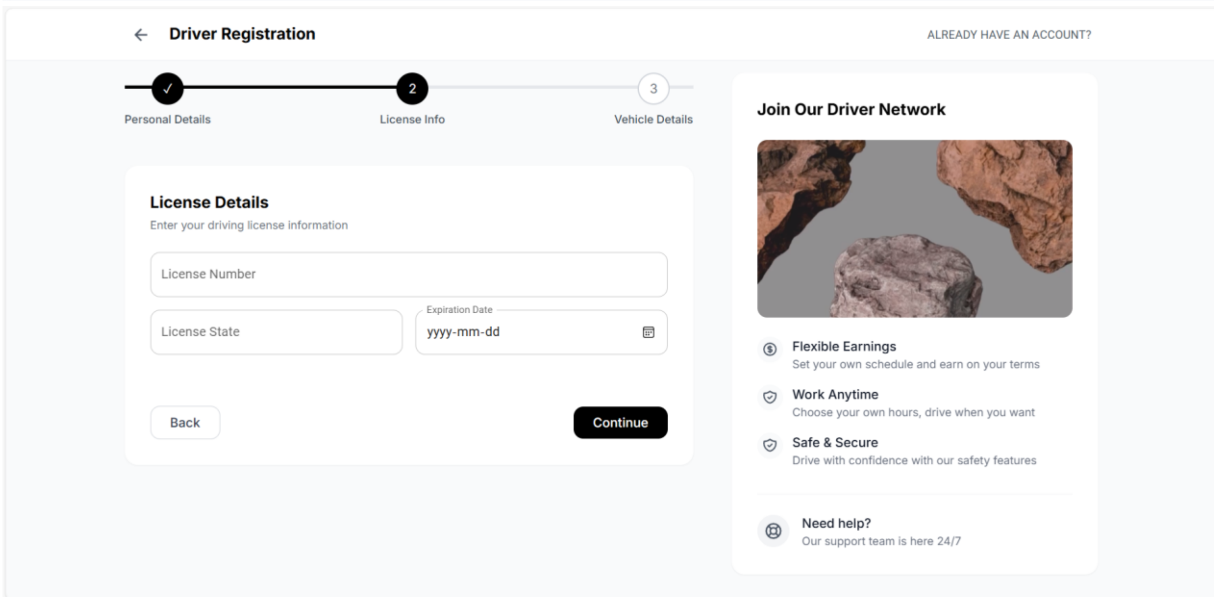
Join Our Driver Network

Flexible Earnings
Set your own schedule and earn on your terms

Work Anytime
Choose your own hours, drive when you want

Safe & Secure
Drive with confidence with our safety features

Need help?
Our support team is here 24/7



License Details

Enter your driving license information

License Number

License State

Expiration Date: yyyy-mm-dd

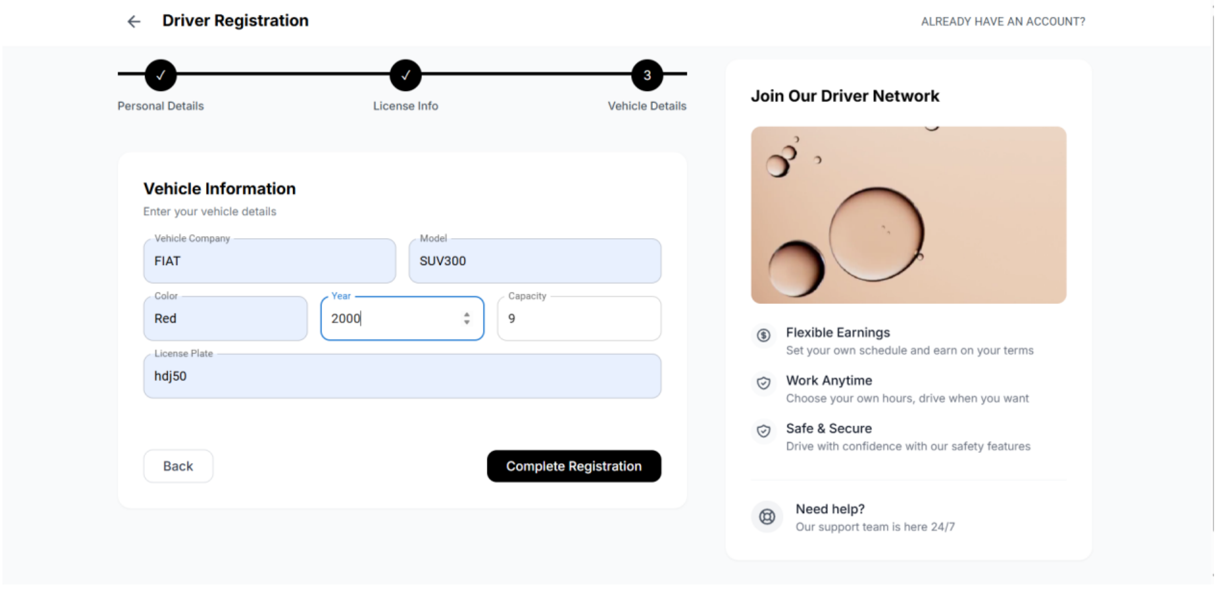
Join Our Driver Network

Flexible Earnings
Set your own schedule and earn on your terms

Work Anytime
Choose your own hours, drive when you want

Safe & Secure
Drive with confidence with our safety features

Need help?
Our support team is here 24/7



Driver Registration

ALREADY HAVE AN ACCOUNT?

Personal Details ✓ — License Info ✓ — Vehicle Details 3

Vehicle Information

Enter your vehicle details

Vehicle Company	FIAT
Model	SUV300
Color	Red
Year	2000
Capacity	9
License Plate	hdj50

Back | Complete Registration

Join Our Driver Network

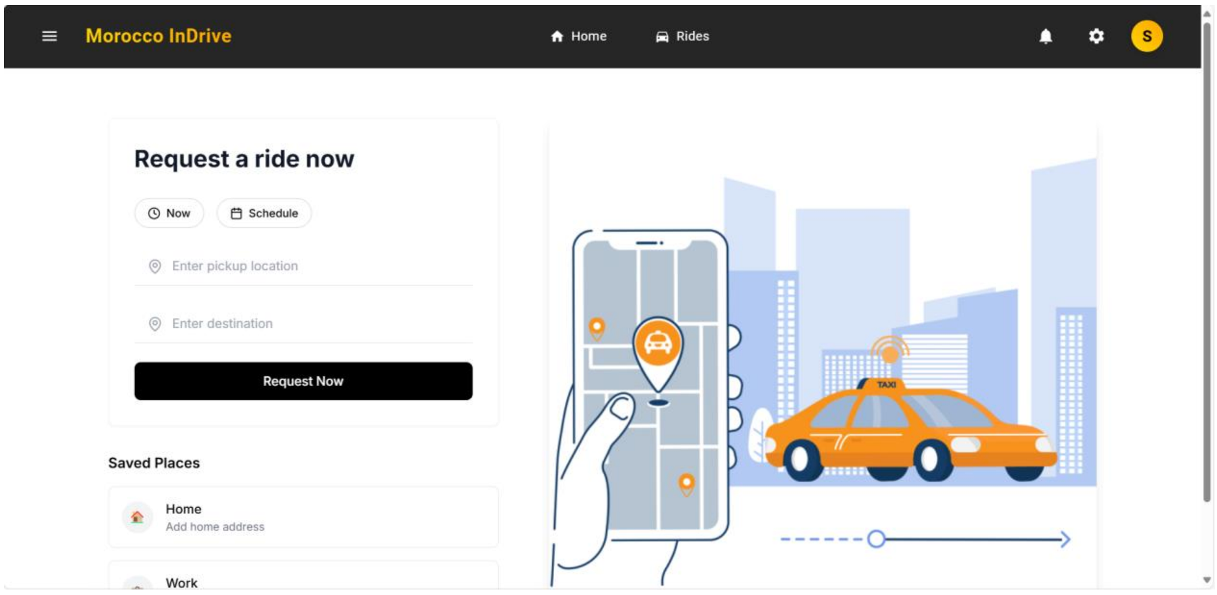
Flexible Earnings — Set your own schedule and earn on your terms

Work Anytime — Choose your own hours, drive when you want

Safe & Secure — Drive with confidence with our safety features

Need help? — Our support team is here 24/7

- ❖ The page From Where The user Enters The Pickup Location and Destination
- ❖ In this Stage when the user book his ride the opt is sent to the Available Driver
- ❖ The Dashboard of an Available driver
- ❖ In this Stage when the user book his ride the opt is sent to the Available Driver
- ❖ Now the rides appear to its dashboard



Morocco InDrive — Home — Rides — S

Request a ride now

Now | Schedule

Enter pickup location

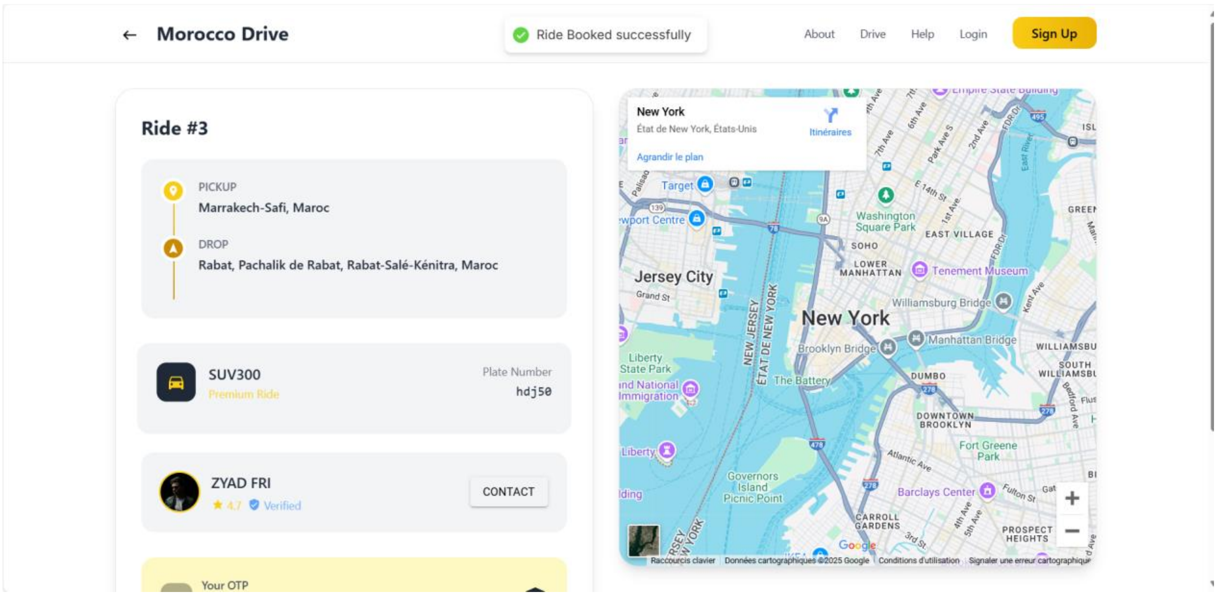
Enter destination

Request Now

Saved Places

Home — Add home address

Work


← Morocco Drive

Ride Booked successfully

About Drive Help Login Sign Up

Ride #3

PICKUP
Marrakech-Safi, Maroc

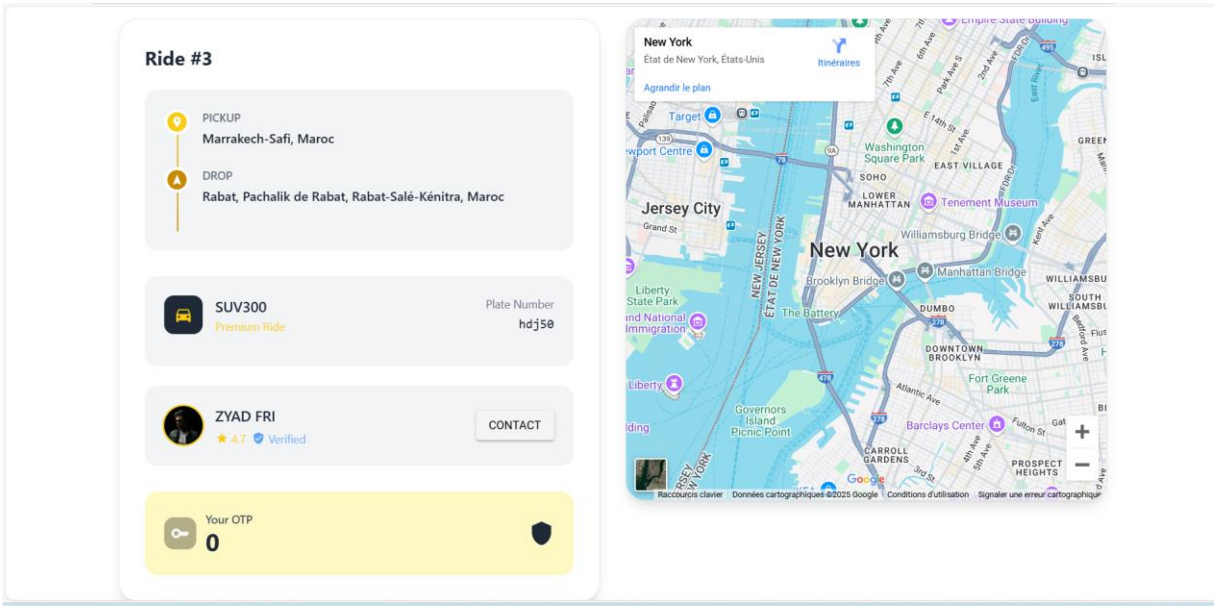
DROP
Rabat, Pachalik de Rabat, Rabat-Salé-Kénitra, Maroc

SUV300
Premium Ride
Plate Number
hdj50

ZYAD FRI
★ 4.7 Verified
CONTACT

Your OTP

New York
État de New York, États-Unis
Itinéraires
Agrandir le plan
Jersey City
New York
Brooklyn Bridge
Manhattan Bridge
Williamsburg Bridge
Washington Square Park
Tenement Museum
Barclays Center
Governors Island Picnic Point
Raccourcis clavier Données cartographiques ©2025 Google Conditions d'utilisation Signaler une erreur cartographique


Ride #3

PICKUP
Marrakech-Safi, Maroc

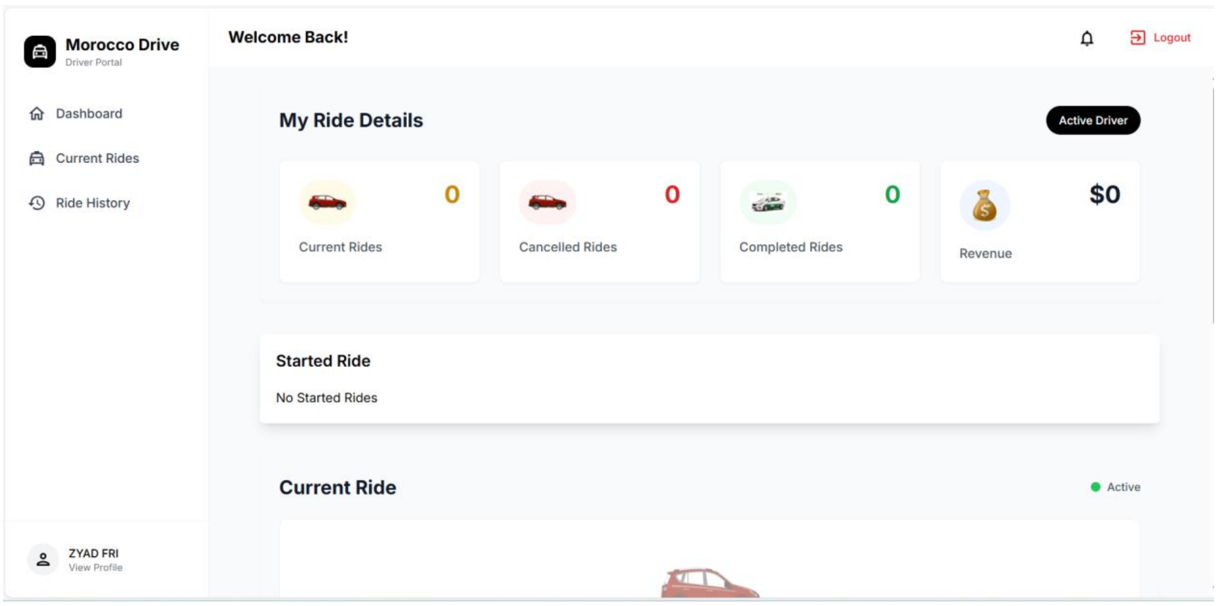
DROP
Rabat, Pachalik de Rabat, Rabat-Salé-Kénitra, Maroc

SUV300
Premium Ride
Plate Number
hdj50

ZYAD FRI
★ 4.7 Verified
CONTACT

Your OTP
0

New York
État de New York, États-Unis
Itinéraires
Agrandir le plan
Jersey City
New York
Raccourcis clavier Données cartographiques ©2025 Google Conditions d'utilisation Signaler une erreur cartographique



Morocco Drive
Driver Portal

Dashboard
Current Rides
Ride History

ZYAD FRI
View Profile

Welcome Back!

Logout

My Ride Details

Active Driver

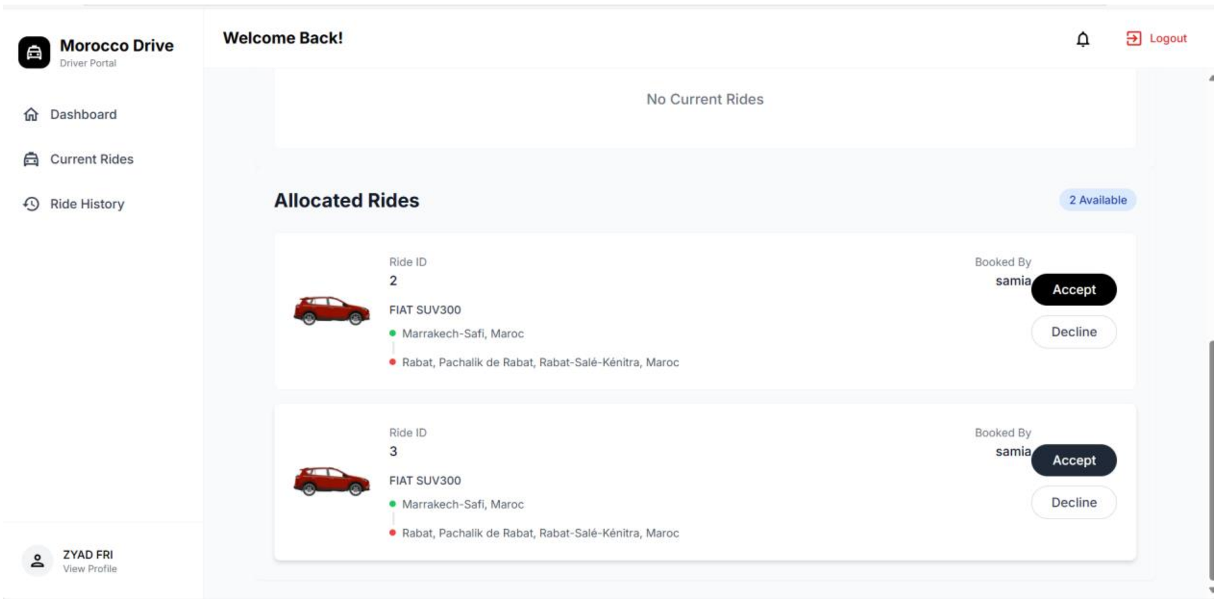
Current Rides	Cancelled Rides	Completed Rides	Revenue
0	0	0	$0

Started Ride

No Started Rides

Current Ride

● Active



Morocco Drive
Driver Portal

Dashboard
Current Rides
Ride History

ZYAD FRI
View Profile

Welcome Back!

Logout

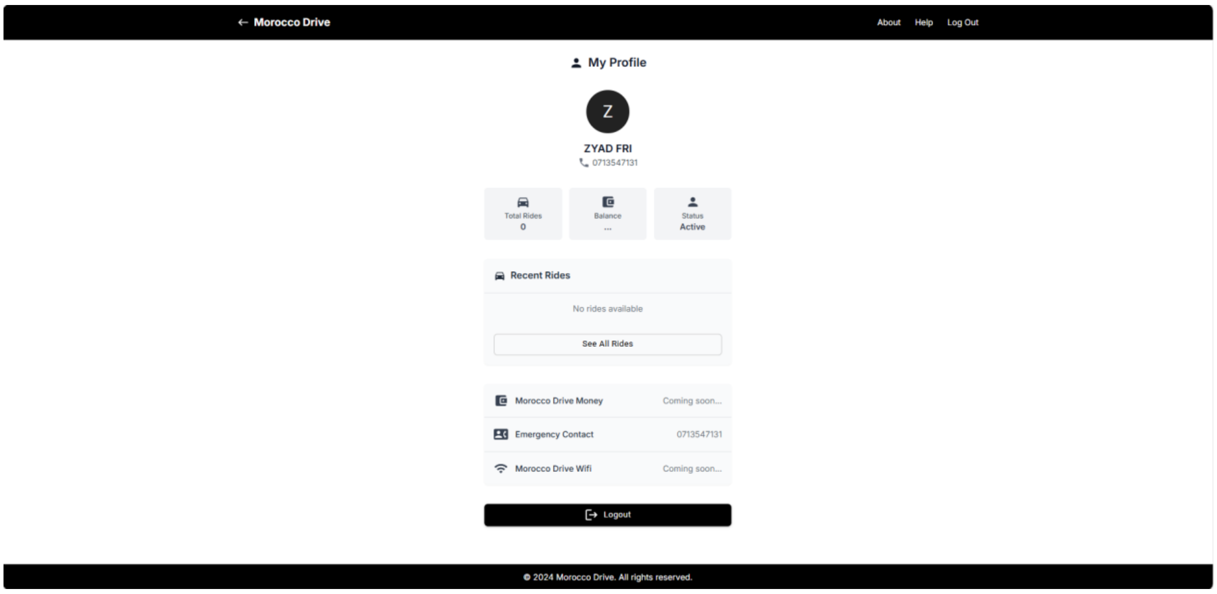
No Current Rides

Allocated Rides

2 Available

Ride ID		Booked By	
2	FIAT SUV300 ● Marrakech-Safi, Maroc ● Rabat, Pachalik de Rabat, Rabat-Salé-Kénitra, Maroc	samia	Accept / Decline
3	FIAT SUV300 ● Marrakech-Safi, Maroc ● Rabat, Pachalik de Rabat, Rabat-Salé-Kénitra, Maroc	samia	Accept / Decline

❖ Profile Page


← Morocco Drive

About Help Log Out

My Profile

Z

ZYAD FRI

0713547131

Total Rides	Balance	Status
0	...	Active

Recent Rides

No rides available

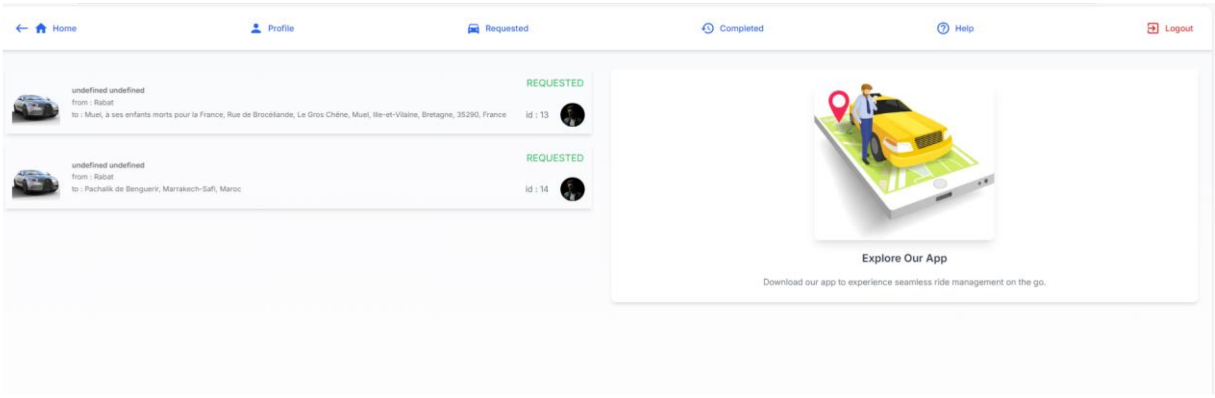
See All Rides

Morocco Drive Money	Coming soon...
Emergency Contact	0713547131
Morocco Drive Wifi	Coming soon...

Logout

© 2024 Morocco Drive. All rights reserved.

- ❖ All the rides are still in the requested Phase since no driver confirms yet any ride.

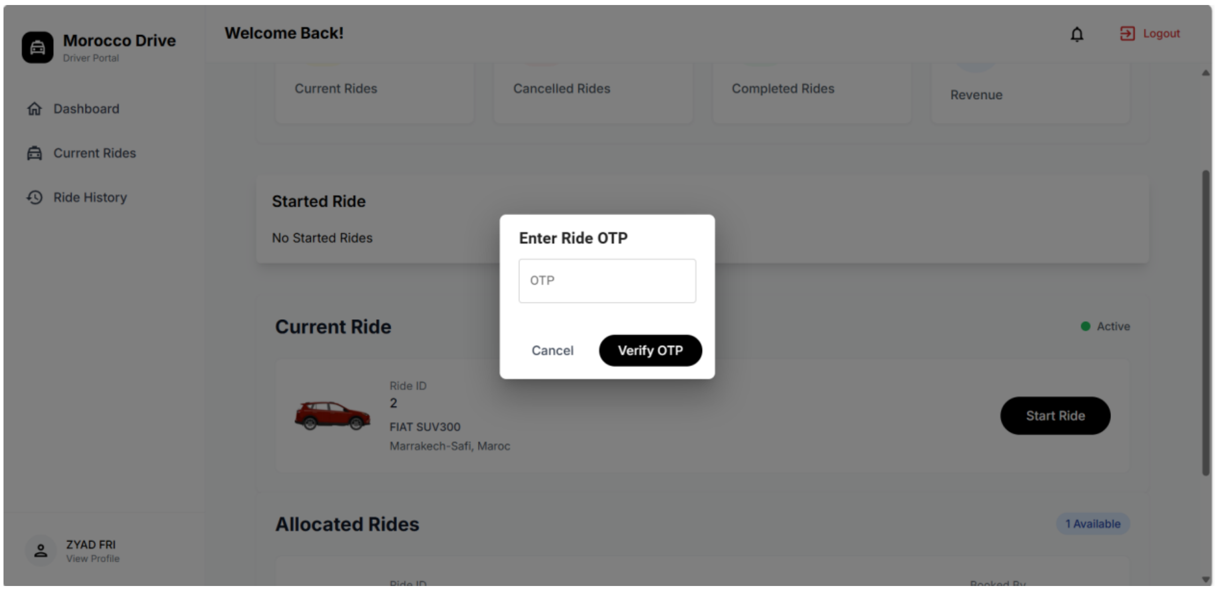

Home Profile Requested Completed Help Logout

undefined undefined
from : Rabat
to : Muel, à ses enfants morts pour la France, Rue de Brocéliande, Le Gros Chêne, Muel, Ille-et-Vilaine, Bretagne, 35290, France
REQUESTED
id : 13

undefined undefined
from : Rabat
to : Pachalik de Benguerir, Marrakech-Safi, Maroc
REQUESTED
id : 14

Explore Our App

Download our app to experience seamless ride management on the go.


Morocco Drive
Driver Portal

Dashboard
Current Rides
Ride History

ZYAD FRI
View Profile

Welcome Back!

Logout

Current Rides | Cancelled Rides | Completed Rides | Revenue

Started Ride
No Started Rides

Enter Ride OTP
OTP
Cancel | Verify OTP

Current Ride
Active

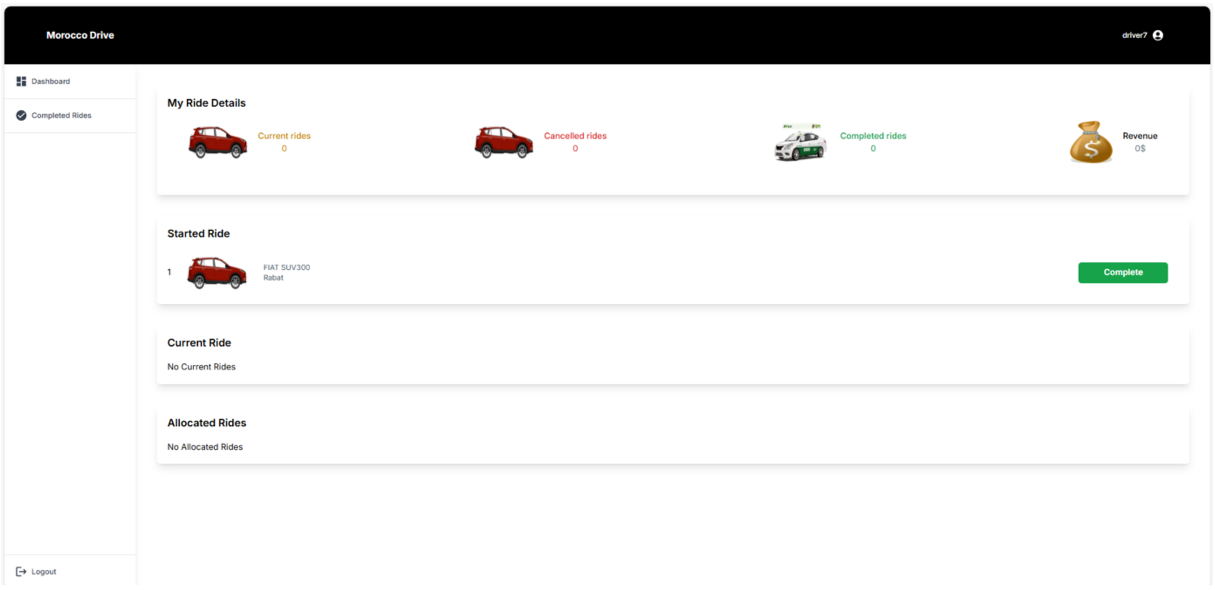
Ride ID
2
FIAT SUV300
Marrakech-Safi, Maroc

Start Ride

Allocated Rides
1 Available

Ride ID
Booked By

- ❖ When the otp is sent to the user.it is displayed as below
- ❖ Here when the Driver Enter the opt that was sent to the user the ride starts



Morocco Drive

driver7

Dashboard

Completed Rides

My Ride Details

Current rides 0

Cancelled rides 0

Completed rides 0

Revenue 0$

Started Ride

1 FIAT SUV300 Rabat

Complete

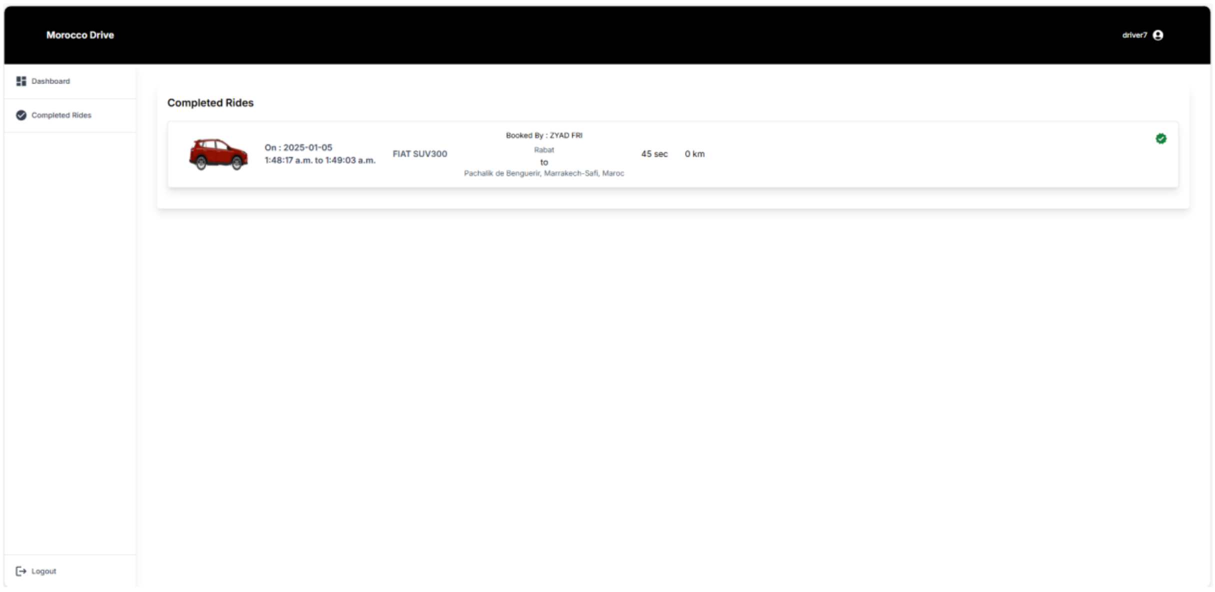
Current Ride

No Current Rides

Allocated Rides

No Allocated Rides

Logout



Morocco Drive

driver7

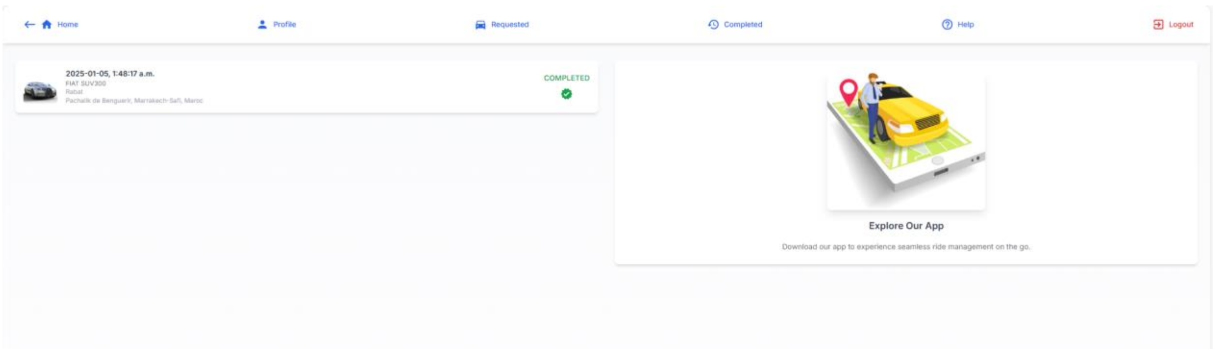
Dashboard

Completed Rides

Completed Rides

On : 2025-01-05 1:48:17 a.m. to 1:49:03 a.m. FIAT SUV300 Booked By : ZYAD FRI Rabat to Pachalik de Benguerir, Marrakech-Safi, Maroc 45 sec 0 km

Logout



Home Profile Requested Completed Help Logout

2025-01-05, 1:48:17 a.m.
FIAT SUV300
Rabat
Pachalik de Benguerir, Marrakech-Safi, Maroc

COMPLETED

Explore Our App

Download our app to experience seamless ride management on the go.

❖ Test Implementation

Drive Backend

▪ *Test Coverage Overview*

1. Authentication Service Tests (AuthServiceTest.java)

The authentication service tests cover critical user management functionality:

Authentication test coverage includes:

- User registration (signup) validation
- Login functionality verification
- Duplicate user registration handling
- Password encoding verification

Key test cases:

- Successful user registration with proper validation
- Handling of duplicate email registrations
- Successful user login verification
- Password encryption verification during registration

2. Ride Service Tests (RideServiceTest.java)

The ride service tests encompass the core business logic for ride management:

Test coverage includes:

- Ride request processing
- Driver assignment logic
- Ride status transitions
- Fare calculation
- OTP validation

Specific test scenarios:

- Successful ride request with driver assignment
- Handling of no available drivers
- Ride acceptance flow verification
- OTP validation during ride start
- Fare calculation during ride completion
- Ride status transitions throughout the journey

3. Calculator Service Tests (CalculatorServiceTest.java)

These tests focus on essential calculation functionalities:

Test coverage includes:

- Distance calculation between coordinates
- Duration calculation

- Fare calculation based on distance

Notable test cases:

- Distance calculation between Marrakech and Casablanca
- Journey duration calculation
- Fare calculation based on distance covered

4. Basic Calculator Tests (CalculatorTest.java)

Simple arithmetic operation testing:

- Basic addition functionality
- Input validation
- Commented out test cases for failure scenarios and exception handling

5. Application Context Test (MoroccoDriveApplicationTests.java)

Basic application context loading test to ensure proper Spring Boot configuration.