Software Requirements Specification (SRS)
UniRide – A Ride-Sharing Application

Nour El Houda EL IAMANI
Aya BENJELLOUN

Updated: November 30, 2025

1 Introduction

This document is the **Software Requirements Specification (SRS)** of the UniRide application. UniRide is a campus-based ride-sharing Android application that allows students, staff, and visitors to request, offer, and coordinate rides inside and outside the university campus.

This SRS consolidates the finalized functional and non-functional requirements obtained after requirement elicitation, analysis, specification, validation, and actual implementation.

2 Functional Requirements

2.1 Rider Interface

2.1.1 Authentication & Profile

- (FR-1) The system shall allow riders to register using Firebase Authentication.
- (FR-2) The system shall allow riders to log in and log out securely.
- (FR-3) The system shall require riders to provide a Moroccan phone number (+212).
- (FR-4) The system shall allow riders to edit their profile information.

2.1.2 Ride Searching & Booking

- (FR-5) The system shall display available ride offers in real time using Firestore listeners.
- (FR-6) The system shall allow riders to search for rides using Google Places Autocomplete.
- (FR-8) The system shall allow riders to filter rides by ride type (inside or outside campus).
- (FR-9) The system shall allow riders to specify the number of passengers.

- (FR-10) The system shall calculate ride fare automatically using distance, price per kilometer, and passenger count.

2.1.3 Ride Management

- (FR-11) The system shall allow riders to view their ride history.
- (FR-12) The system shall allow riders to view their booked rides.
- (FR-13) The system shall allow riders to cancel his rides bookings.
- (FR-14) The system shall display ride status using color-coded badges.

2.1.4 Communication

- (FR-15) The system shall allow riders to contact drivers through WhatsApp with a pre-filled message.

2.1.5 Ride Tracking

- (FR-16) The system shall provide real-time ride status updates using Firestore snapshot listeners.

2.2 Driver Interface

2.2.1 Authentication & Verification

- (FR-16) The system shall allow drivers to upload required documents (ID, insurance, license).
- (FR-17) The system shall restrict ride creation until the admin verifies the driver's documents.

2.2.2 Ride Offer Management

- (FR-18) The system shall allow drivers to create ride offers with pickup, destination, date, time, seats, and price.
- (FR-19) The system shall automatically calculate ride distance.
- (FR-20) The system shall display lists of active, full, departed, cancelled, and completed ride offers.

2.2.3 Ride Operations

- (FR-21) The system shall allow drivers to start and finish rides.
- (FR-22) The system shall allow drivers to cancel ride Offers .
- (FR-23) The system shall update ride status automatically based on conditions.

2.2.4 Real-time Updates

- (FR-24) The system shall update the passenger list in real time using Firestore listeners.

2.2.5 Communication

- (FR-25) The system shall allow drivers to contact riders through WhatsApp using a pre-filled message.

2.3 Admin Interface

2.3.1 User Management

- (FR-26) The system shall allow admins to view all users.
- (FR-27) The system shall allow admins to verify or reject driver documents.
- (FR-28) The system shall allow admins to delete user accounts.

2.3.2 Ride Management

- (FR-29) The system shall allow admins to filter users by role and status.

2.3.3 Dashboard

- (FR-30) The system shall display statistics including total riders, drivers, pending and approved users.

3 Non-Functional Requirements

3.1 Performance

- (NFR-1) The system shall provide Firestore real-time updates with less than 2 seconds delay.
- (NFR-2) The system shall compute distances efficiently using external APIs.

3.2 Security

- (NFR-3) The system shall authenticate users securely using Firebase Authentication.
- (NFR-4) The system shall ensure all communication occurs over HTTPS.
- (NFR-5) The system shall enforce Firestore security rules to restrict unauthorized access.

3.3 Usability

- (NFR-6) The system shall provide a modern and intuitive Material Design interface.

3.4 Portability

- (NFR-7) The system shall run on Android devices with API level 21 or above.

3.5 Maintainability

- (NFR-8) The system shall be structured using software design patterns such as Singleton, Adapter, and Strategy.

4 Data Models

4.1 User Model

```
class User {
    String userId;
    String name;
    String email;
    String phoneNumber;
    String role;
    Long createdAt;
}
```

4.2 Driver Model

```
class Driver extends User {
    String licenseUrl;
    String insuranceUrl;
    String idCardUrl;
    String verificationStatus;
}
```

4.3 Rider Model

```
public class Rider extends User {

    private int cancellationCount;
    private boolean suspended;
    private List<String> rideHistory;
}
```

4.4 AdminUser Model

```
public class AdminUser extends User {
}
}
```

4.5 Ride Model

```
class Ride {
    String rideId;
    String riderId;
    String riderName;
    String riderPhone;
    String offerId;
    String driverId;
    String pickupLocation;
    String destination;
    Long departureTime;
    int totalPassengers;
    double totalFare;
    double distanceKm;
    String status;
}
```

4.6 RideOffer Model

```
class RideOffer {
    String offerId;
    String driverId;
    String driverName;
    String driverPhone;
    String pickupLocation;
    String destination;
    String rideType;
    Long departureTime;
    int totalSeats;
    int availableSeats;
    double pricePerSeat;
    double distanceKm;
    String carModel;
    String licensePlate;
    String status;
    Long createdAt;
}
```

5 Use Case Diagram (Description)

Actors:

- Rider
- Driver
- Admin

- Firebase Backend
- Google Maps API
- WhatsApp

Use Case Diagram :

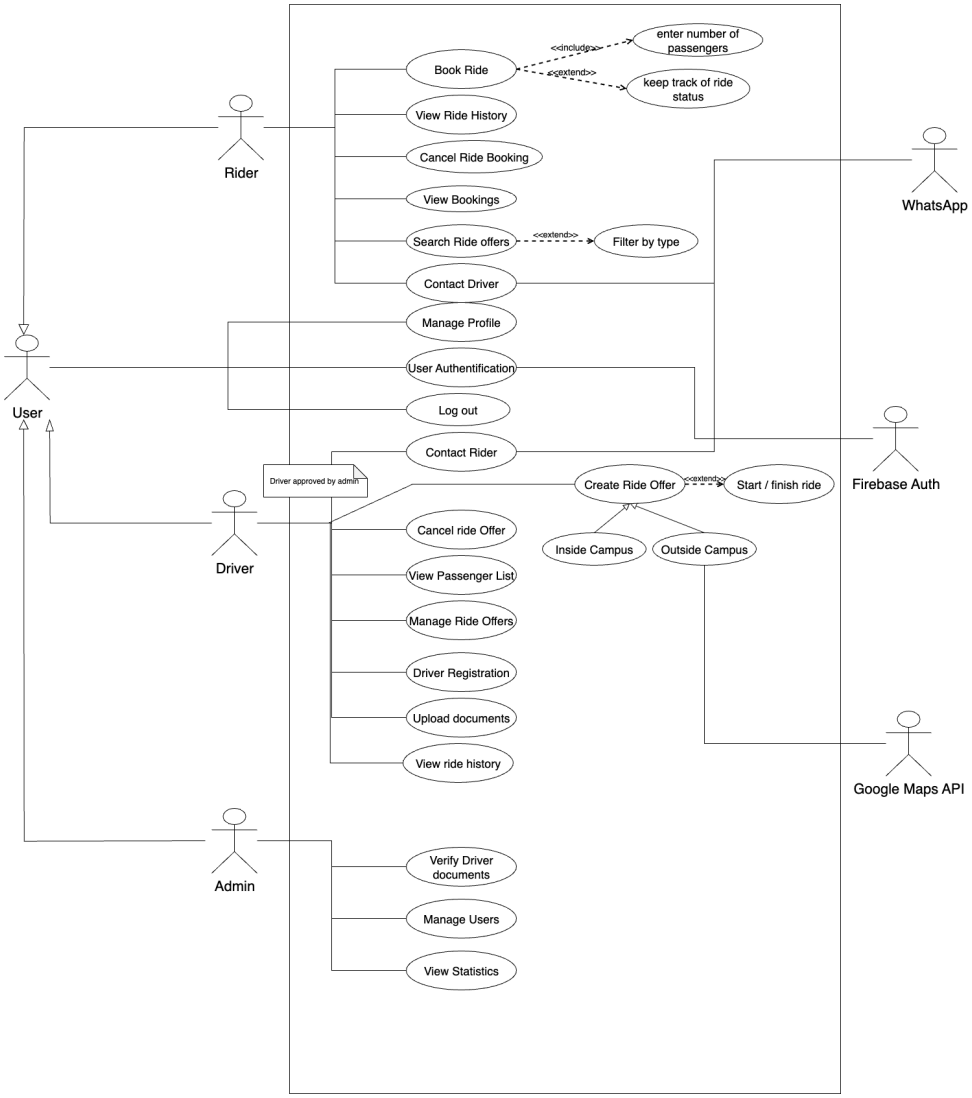

Figure 1: Use Case Diagram

6 Conclusion

The UniRide application successfully integrates core ride-sharing functionalities tailored for a university environment. This Software Requirements Specification outlined the functional and non-functional requirements necessary to implement a secure, scalable, and user-friendly system.