FahrFlex (Share-A-Ride) – Technical Documentation

Project Overview

FahrFlex is a full-stack ridesharing application that connects drivers with passengers traveling between German cities. The application uses a modern stack with **Spring Boot 3.3** (backend) and **Angular 21** (frontend).

Technology Stack

Backend

- **Framework:** Spring Boot 3.3.0 (Java 17+)
- **Database:** H2 In-Memory Database (development)
- **ORM:** Hibernate/JPA (automatic schema generation)
- **Build Tool:** Maven
- **API Style:** RESTful APIs

Frontend

- **Framework:** Angular 21 (Standalone Components)
- **Language:** TypeScript 5.9
- **Styling:** SCSS + custom design system
- **State Management:** Reactive Forms + Services

Database Schema

The database follows a relational model with **10 main entities**.

1) PERSON (Users Table)

Column	Type	Description
id	INTEGER (PK)	Auto-generated primary key

name	VARCHAR	Full name
gender	VARCHAR	male/female/other
age	INTEGER	User's age
profile_picture	VARCHAR	URL to profile image
home_city	VARCHAR	User's home city
bio	VARCHAR(1000)	User biography/description
phone_number	VARCHAR	Contact phone
email	VARCHAR	Email address
car_id	INTEGER (FK)	References CAR table (nullable)
languages	VARCHAR	Comma-separated language codes
chatiness_level	INTEGER	1–5 scale for conversation preference
overall_km_covered	INTEGER	Total kilometers traveled

---

2) CAR (Vehicles Table)

Column	Type	Description
id	INTEGER (PK)	Auto-generated primary key
make	VARCHAR	Car manufacturer (e.g., BMW)
model	VARCHAR	Car model (e.g., 320i)
available_seats	INTEGER	Max passenger seats
luggage_space	INTEGER	Luggage capacity (0–5)
smoking_allowed	BOOLEAN	Smoking policy
pets_allowed	BOOLEAN	Pet policy
plate	VARCHAR	License plate number
build_year	INTEGER	Manufacturing year

insurance_id	INTEGER (FK)	References INSURANCE table

---

3) INSURANCE (Insurance Policies)

Column	Type	Description
id	INTEGER (PK)	Auto-generated primary key
name	VARCHAR	Insurance company name
policy_name	VARCHAR	Policy tier name
policy_number	VARCHAR	Policy identifier
coverage	ENUM	Haftpflicht / Teilkasko / Vollkasko
passenger_driver_insurance	BOOLEAN	Covers passengers

---

4) RIDEOFFER (Available Rides)

Column	Type	Description
id	INTEGER (PK)	Auto-generated primary key
departure_city	VARCHAR	Starting city
destination_city	VARCHAR	End city
departure_time	TIMESTAMP	Scheduled departure
seats_available	INTEGER	Seats offered
price_per_person	DOUBLE	Cost per passenger (€)
luggage_count	INTEGER	Max luggage items
driver_person_id	INTEGER (FK)	References PERSON
car_id	INTEGER (FK)	References CAR
smoking_allowed	BOOLEAN	Smoking policy for this ride
pets_allowed	BOOLEAN	Pet policy for this ride

music_allowed	BOOLEAN	Music preference
chat_level	INTEGER	Expected conversation level
additional_notes	VARCHAR(500)	Driver's notes
flexible_time	BOOLEAN	Time flexibility flag
flexibility_minutes	INTEGER	± minutes flexibility
stops	VARCHAR(500)	Comma-separated stopovers
accepted_payment_methods	VARCHAR	CASH, CARD, PAYPAL

---

5) RIDE (Confirmed Trips)

Column	Type	Description
id	INTEGER (PK)	Auto-generated primary key
ride_offer_id	INTEGER (FK)	References RIDEOFFER
departure_city	VARCHAR	Starting city
destination_city	VARCHAR	End city
departure_time	TIMESTAMP	Actual departure time
driver_person_id	INTEGER (FK)	References PERSON (driver)
seats_remaining	INTEGER	Available seats after bookings
luggage_remaining	INTEGER	Available luggage space
status	VARCHAR	ACTIVE / IN_PROGRESS / COMPLETED / CANCELLED

---

6) RIDEREQUEST (Booking Requests)

Column	Type	Description
id	INTEGER (PK)	Auto-generated primary key
ride_offer_id	INTEGER (FK)	References RIDEOFFER
person_id	INTEGER (FK)	References PERSON (passenger)
pickup_location	VARCHAR	Passenger pickup point
dropoff_location	VARCHAR	Passenger drop-off point
timestamp	TIMESTAMP	Request creation time
luggage_count	INTEGER	Passenger’s luggage
pet	BOOLEAN	Traveling with pet
kid	BOOLEAN	Traveling with child
payment_method	ENUM	CASH / CARD / PAYPAL
status	ENUM	PENDING / ACCEPTED / REJECTED / CANCELLED
ride_id	INTEGER (FK)	References RIDE (when accepted)
passenger_id	INTEGER (FK)	References PASSENGER (when accepted)

---

7) PASSENGER (Confirmed Passengers)

Column	Type	Description
id	INTEGER (PK)	Auto-generated primary key
ride_id	INTEGER (FK)	References RIDE
person_id	INTEGER (FK)	References PERSON
luggage_count	INTEGER	Passenger’s luggage
payment_method	ENUM	CASH / CARD / PAYPAL
payment_outstanding	BOOLEAN	Payment status
pickup_location	VARCHAR	Custom pickup point

dropoff_location	VARCHAR	Custom drop-off point
pet	BOOLEAN	Has pet
kid	BOOLEAN	Has child
seats_consumed	INTEGER	Total seats used (1 + pet + kid)

Additional Entities

- **BOOKING:** Tracks booking history and status
- **REVIEW:** Driver/passenger ratings and feedback
- **CHATMESSAGE:** In-app messaging between users

Entity Relationships Diagram (Text Version)

INSURANCE (1:N) CAR
PERSON (1:1) CAR
PERSON (1:N) RIDEOFFER
RIDEOFFER (1:N) RIDE
RIDE (1:N) PASSENGER
RIDEOFFER (1:N) RIDEREQUEST → (becomes) PASSENGER when accepted

---

Implemented Features

Core Features

1) Find a Ride (`/find-ride`)

- Search by **departure city**, **destination city**, and **date**
- **City autocomplete** with fuzzy matching
- Advanced filters:
  - Seats needed
  - Budget range (min/max price)
  - Time slots (Morning / Afternoon / Evening / Night)
  - Preferences (smoking, pets, music allowed)
  - “I am” filters (student, smoker, pet owner)

- Sort options: price (low/high), departure time, duration
- Ride cards:
  - Driver info (avatar + rating)
  - Route timeline (departure/arrival)
  - Available seats indicator
  - Price per person
  - Stopovers display
- Booking flow: request to join with custom pickup/dropoff

2) Post a Ride (`/post-ride`)

- 7-step wizard:
  1. Route (departure + destination, swap supported)
  2. Stopovers (optional intermediate stops)
  3. Schedule (date/time + flexibility ±15/30/60 min)
  4. Vehicle (select registered car, set seats + luggage)
  5. Preferences (smoking/pets/music toggles, chat slider 1–5)
  6. Pricing (suggested price calculator, manual override, payment methods)
  7. Review (final summary)
- Progress tracking: visual step indicator + completion state
- Real-time form validation with error messages

3) Driver Profile (`/driver-profile/:id`)

- Driver info + bio
- Car details + insurance info
- Ratings and reviews
- Ride history

4) My Rides (`/my-rides`)

- Upcoming rides (driver or passenger)
- Past ride history
- Ride status tracking
- Cancel ride functionality

5) Booking Confirmation (`/booking-confirmation`)

- Booking detail summary
- Payment information
- Confirmation number

Communication Features

6) Chat (`/chat`)

- In-app messaging between driver and passenger
- Real-time updates
- Message history

Payment Features

7) Payment (`/payment`)

- Multiple payment methods (Cash, Card, PayPal)
- Payment status tracking
- Transaction history

Feedback Features

8) Reviews (`/reviews`)

- Rate drivers (1–5 stars)
- Write text reviews
- View received reviews

9) Feedback (`/feedback`)

- Submit app feedback
- Report issues

Static Pages

- Landing (`/`)
- About Us (`/about-us`)
- Contact Us (`/contact-us`)
- Support (`/support`)
- Privacy Policy (`/privacy-policy`)
- Imprint (`/imprint`)

REST API Endpoints

Ride Offers

Method	Endpoint	Description
GET	/api/rideoffers/search	Search ride offers with filters
GET	/api/rideoffers/{id}	Get ride offer details
POST	/api/rides	Create new ride offer
GET	/api/rides/driver/{id}	Get driver’s ride offers

Ride Requests

Method	Endpoint	Description
POST	/api/riderequest	Create booking request
GET	/api/riderequest/{id}	Get request status
PUT	/api/riderequest/{id}/accept	Driver accepts request
PUT	/api/riderequest/{id}/reject	Driver rejects request

Cities

Method	Endpoint	Description
GET	/api/cities	Get all available cities

Persons

Method	Endpoint	Description
GET	/api/persons/{id}	Get person details
GET	/api/persons/{id}/rides	Get person’s ride history

Sample Data

The database is pre-seeded with realistic test data.

- **Users:** 9 persons (mix of drivers and passengers)
  Home cities include Ulm, Neu-Ulm, Munich, Stuttgart, Augsburg, Frankfurt; various chatiness levels and preferences.
- **Vehicles:** 7 cars (VW, BMW, Mercedes, Audi, Tesla, Toyota, Ford) with different seat capacities and policies.
- **Ride Offers:** 18 offers (today, tomorrow, this week, next week)
  Routes across major German cities; price range **€8–€40 per person**; various preferences and stopovers.

Design Highlights

- **Primary Color:** #00b894 (FahrFlex Green)
- Modern, card-based UI with smooth transitions
- Responsive (mobile-first)
- Accessibility (labels, keyboard navigation)
- UI aligned with Balsamiq wireframes

Running the Project

Backend (port 8080)

1. Open terminal
2. Run:

- cd backend
- mvn spring-boot:run

Frontend (port 4200)

1. Open terminal
2. Run:

- cd frontend/share-a-ride-ui
- npm install
- npm start

Access: http://localhost:4200

Team: EAE Group 5
Project: FahrFlex – Share-A-Ride Application