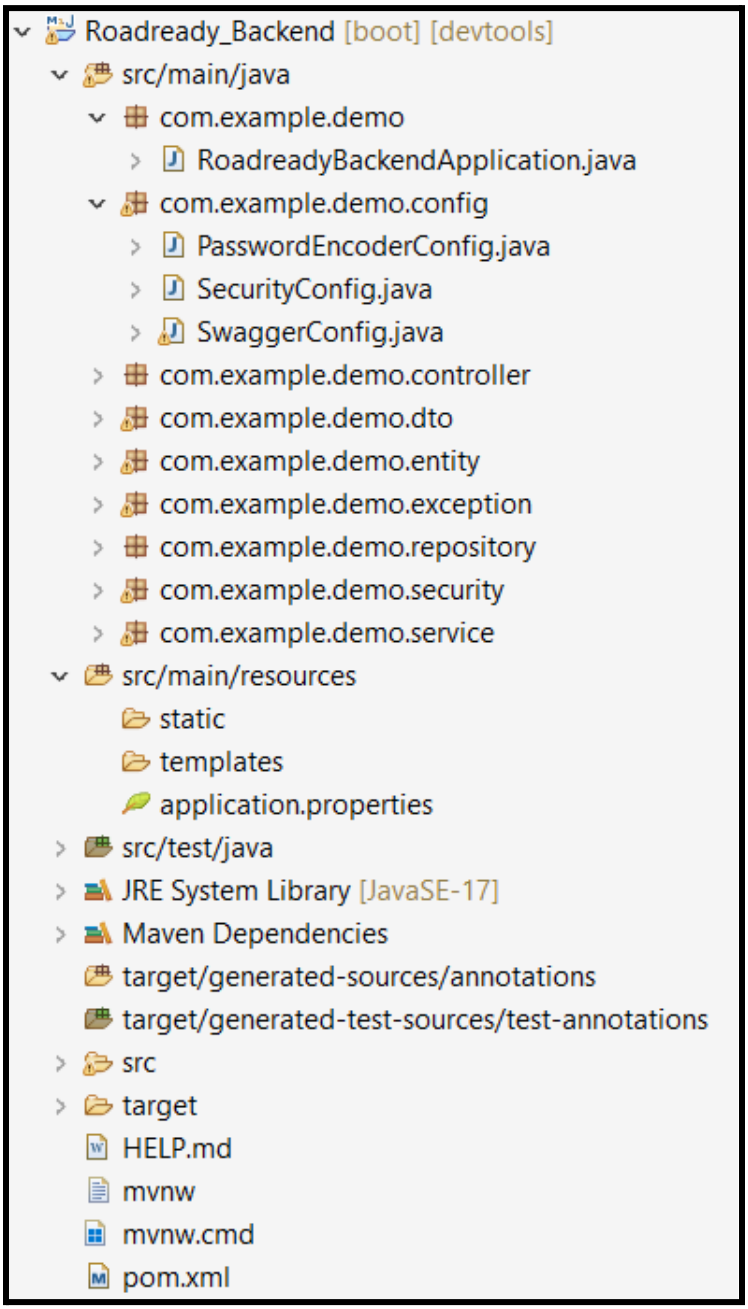
RoadReady Car Rental System – Backend Project Documentation

Project Overview:

RoadReady is a full-fledged car rental backend system built using Java Spring Boot, JWT authentication, and MySQL. It facilitates customers to browse and reserve cars, manage bookings, and make payments. Admins can manage users, cars, and reservations through RESTful APIs secured with JWT-based authentication and authorization.

Package Structure:



The project follows a clean MVC architecture and is divided into the following main packages:

1. com.example.demo.entity

   Contains all entity (model) classes mapped to database tables using JPA annotations.

   - User.java: Represents a user with fields like id, name, email, password, role, etc.
   - Car.java: Represents a car available for rental with fields like brand, model, pricePerDay, etc.
   - Reservation.java: Represents a reservation entry, connected to Car and User.
   - Payment.java: Represents payment details for a reservation.

2. com.example.demo.repository

   Contains interfaces extending JpaRepository for CRUD operations.

   - UserRepository
   - CarRepository
   - ReservationRepository
   - PaymentRepository

3. com.example.demo.service

   Contains business logic and service classes implementing operations.

   - UserService
   - CarService
   - ReservationService
   - PaymentService

4. com.example.demo.controller

   Contains REST controllers for handling HTTP requests.

- AuthController: Handles login and JWT generation.
- CarController: CRUD APIs for cars.
- ReservationController: APIs for making and managing reservations.
- PaymentController: Processes payments.
- UserController: Gets user profile information.

5. com.example.demo.dto

   DTOs used to transfer data between layers:

   - AuthRequestDTO
   - AuthResponseDTO
   - ReservationRequestDTO
   - ReservationResponseDTO
   - PaymentRequestDTO
   - PaymentResponseDTO

6. com.example.demo.security

   Handles JWT authentication logic.

   - JwtTokenProvider: Generates and validates JWT tokens.
   - JwtAuthenticationFilter: Intercepts and validates token in each request.

7. com.example.demo.config

   - SecurityConfig.java: Configures Spring Security, sets protected paths and filters.

JWT Authentication & Authorization:

JWT Token Generation:

- On successful login (/api/auth/login), a JWT is generated containing:

    - Subject: user email
    - Claim: role (e.g., CUSTOMER or ADMIN)
    - Expiry: 24 hours (as configured in jwt.expiration)
- The token is signed using a secret key with HS256 algorithm:

  {jwt.secret=RoadReadySuperSecureJwtSecretKey_2025_XYZ}

Example:
1)Email: shraddha@example.com
Password: shraddha123

Generated Tocken:

Bearer
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzaHJhZGRoYUBleGFtcGxlLmNvbSIsInJvbGUiOiJDVVNUT01FUiIsImlhdCI6MTc1MDg0Njg3NywiZXhwIjoxNzUwOTMzMjc3fQ.yT5xQnjn2Y4NgOkjlihxIE34833Wj7zthjC1v54BoaU

2) Email: avani@123.com
Password: avani123

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhdmFuaUAxMjMuY29tIiwicm9sZSI6IkNVU1RPTUVSIiwiaWF0IjoxNzUwODc0NzE0LCJleHAiOjE3NTA5NjExMTR9.U7UAex_Pj3MUlZqx0bIBKPtRo-tK4EMzPPtlEIUww9M",

"role": "CUSTOMER}

Authentication Flow:

- User logs in → token generated and returned.
- Token is passed in Authorization header for each request: Authorization: Bearer
- JwtAuthenticationFilter intercepts request:
    - Validates token
    - Extracts user and role
    - Grants access based on role using Spring Security.

Role-Based Access Control:

Roles:

- CUSTOMER: Can view available cars, make reservations, view reservations, and make payments.

- ADMIN: Can manage users, cars, reservations.

SecurityConfig.java defines access control rules:

```
http
 .authorizeHttpRequests(auth -> auth
 .requestMatchers("/api/auth/**").permitAll()**
 **.requestMatchers("/swagger-ui/**", "/v3/api-docs/**").permitAll()**
 **.requestMatchers("/api/reservations/**", "/api/payments/**", "/api/users/me").hasRole("CUSTOMER")**
 **.requestMatchers("/api/cars/**").permitAll()
 .anyRequest().authenticated()
 )
```

API Flow:

Here's how typical operations work:

1. Login (POST /api/auth/login)

- Request: email, password

- Response: token, role

2. Add Car (ADMIN)

- POST /api/cars

- Requires Authorization header (ADMIN)

3. Get All Cars

- GET /api/cars

- Public or secured

4. Create Reservation (CUSTOMER)

- POST /api/reservations

- Token required
- Request: carId, dates, pickup/drop locations

5. View Reservations

- GET /api/reservations
- Returns user-specific reservations

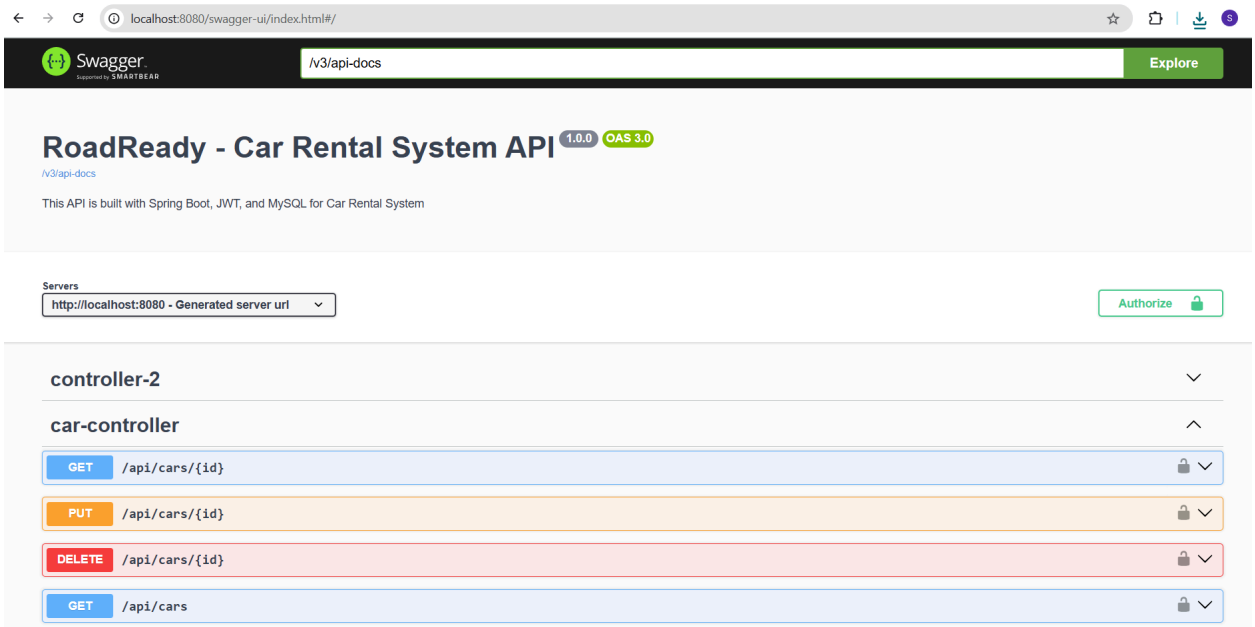
6. Make Payment

- POST /api/payments
- Requires reservationId and amount

API Testing:

All APIs can be tested via:

- Postman (with token in header)
- Swagger UI at /swagger-ui.html (Authorize → paste token)
  http://localhost:8080/swagger-ui/index.html



Output Screenshots:

RoadReady Backend Application is running:

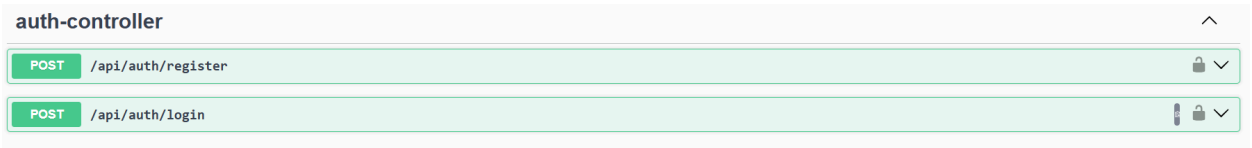
RoadreadyBackendApplication.java

```java
package com.example.demo;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class RoadreadyBackendApplication {

    public static void main(String[] args) {
        SpringApplication.run(RoadreadyBackendApplication.class, args);
        System.out.println("ROadReady application is running");
    }
```

Problems | Javadoc | Declaration | Console | Progress

Roadready_Backend - RoadreadyBackendApplication [Spring Boot App] C:\Users\LENOVO\Downloads\spring-tools-

```
2025-06-27T15:05:58.618+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] c
2025-06-27T15:05:58.832+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] .
2025-06-27T15:05:58.833+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] .
2025-06-27T15:06:03.214+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] .
2025-06-27T15:06:03.720+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] .
2025-06-27T15:06:04.515+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:05.779+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:05.817+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:05.818+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:05.991+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:05.992+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] w
2025-06-27T15:06:06.489+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:06.703+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:06.823+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:07.618+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:07.707+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] c
2025-06-27T15:06:08.072+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] c
2025-06-27T15:06:08.076+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] c
2025-06-27T15:06:08.235+05:30  WARN 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:08.238+05:30  WARN 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:10.630+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:10.927+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] j
2025-06-27T15:06:12.348+05:30  WARN 11536 --- [Roadready_Backend] [  restartedMain] J
2025-06-27T15:06:12.536+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:14.612+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:16.060+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] o
2025-06-27T15:06:16.103+05:30  INFO 11536 --- [Roadready_Backend] [  restartedMain] c
ROadReady application is running
```



- User Registration

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/api/auth/register' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Shraddha G",
  "email": "shraddhagavkare13@gmail.com",
  "password": "shraddha13",
  "phone": "8766828269",
  "address": "PUNE"
}'
```

Request URL

```
http://localhost:8080/api/auth/register
```

Server response

Code	Details
200	Response body

```
{
  "id": 8,
  "name": "Shraddha G",
  "email": "shraddhagavkare13@gmail.com",
  "password": "$2a$10$tJhE1LZ9.E0nUWNscjYB6uxLNFPXdrXrGwYJbFOywQ8NNYCGy28G6",
  "role": "CUSTOMER",
  "phone": "8766828269",
  "address": "PUNE"
}
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	address	email	name	password	phone	role
1	Pune, India	shraddha@example.com	Shraddha	$2a$10$HCcSmHcCIOqAvL/w0sFUrubK5lNPWW...	9876543210	CUSTOMER
2	Pune, India	vasu@example.com	Vasudha	$2a$10$eoA9i/KhugnM7rE7hmDmQeryMua2jd...	8776543210	CUSTOMER
3	Karad, India	ro@example.com	Rahul	$2a$10$fmLcVJKriRU4BviezjzycueITA3JHEsSnL...	8076543210	CUSTOMER
4	HQ	admin@example.com	Admin	$2a$10$tvr34H7vwzItUhuXEqTAH.3Uq9h5Odro...	9999999999	CUSTOMER
5	Mumbai	avani@123.com	Avani	$2a$10$MbtI5PQLIU5dF8oxJaOwDe8d2QYJQE...	6547895236	CUSTOMER
6	Bengluru	virat@123.com	Virat	$2a$10$L2N3KD4J.B8/.PuAM5G5uel4lw4w.VhLr...	5263895236	CUSTOMER
7	Killari	sonali@123.com	Sonali	$2a$10$3d1jyqqdH4xAXdvjvJ.H/Oi8Pnj3QwmH...	9874563214	CUSTOMER
8	PUNE	shraddhagavkare13@gmail.com	Shraddha G	$2a$10$tJhE1LZ9.E0nUWNscjYB6uxLNFPXdrXr...	8766828269	CUSTOMER
NULL	NULL	NULL	NULL	NULL	NULL	NULL

- **Login API with response token**

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/api/auth/login' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "shraddhagavkare13@gmail.com",
  "password": "shraddha13"
}'
```

Request URL

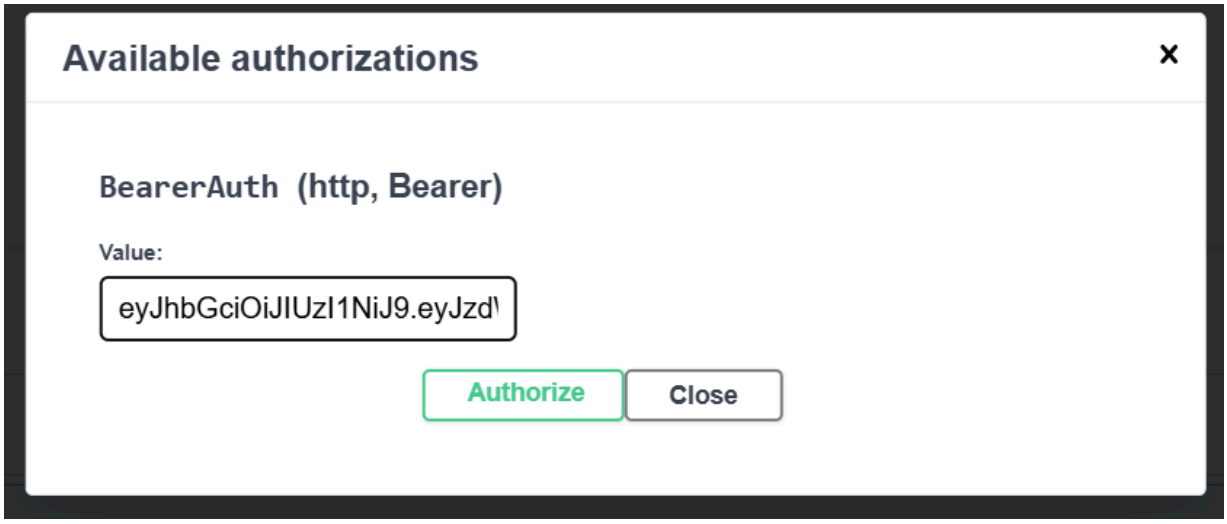
```
http://localhost:8080/api/auth/login
```

Server response

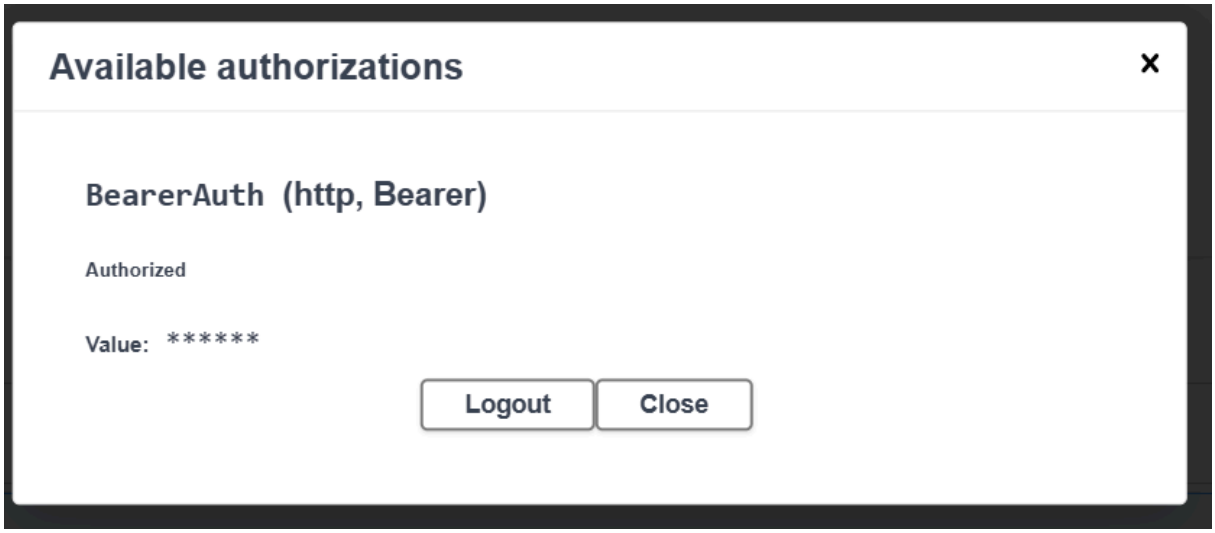
Code	Details
200	Response body

```
{
  "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzaHJhZGRoYWdhdmthcmUxM0BnbWFpbC5jb20iLCJyb2xlIjoiQ1VTVE9NRVIiLCJpYXQiOjE3NTEwMTc4MzEsImV4cCI6MTc1MTEwNDIzMX0.ugrtWFPEB0iX0RuIQ2kedu3Ql8XIEaYp9FSw7F4D4V8",
  "role": "CUSTOMER"
}
```

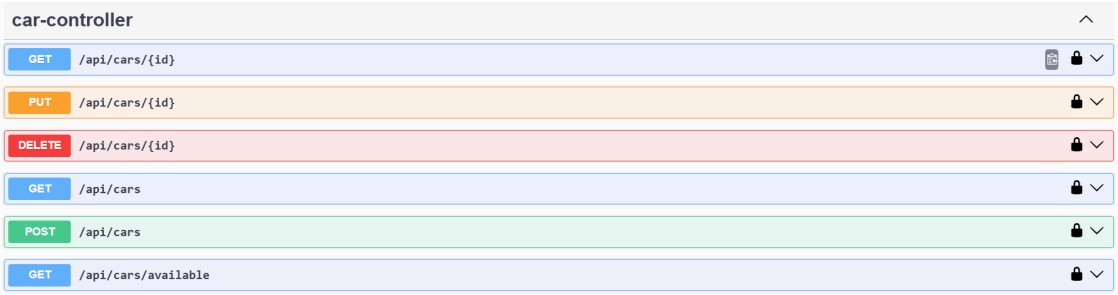
Download



Swagger Authorize dialog:



Car controller API:



- Add Car API

Curl

```
curl -X 'POST' \
  'http://localhost:8080/api/cars' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzaHJhZGRoYWdhdmthcmUxM0B
  -H 'Content-Type: application/json' \
  -d '{
"brand": "Tata",
"model": "Harrier",
"year": 2024,
"color": "Red",
"type": "SUV",
"pricePerDay": 3200,
"available": true,
"licensePlate": "MH14YZ9001"
}'
```

Request URL

```
http://localhost:8080/api/cars
```

Server response

Code	Details
200	Response body

```
{
  "id": 13,
  "brand": "Tata",
  "model": "Harrier",
  "year": 2024,
  "color": "Red",
  "type": "SUV",
  "pricePerDay": 3200,
  "available": true,
  "licensePlate": "MH14YZ9001"
}
```

- Get All Cars API (Customer)

Request URL

```
http://localhost:8080/api/cars
```

Server response

Code | Details

200

Response body


```
[
  {
    "id": 1,
    "brand": "Hyundai",
    "model": "Creta",
    "year": 2022,
    "color": "White",
    "type": "SUV",
    "pricePerDay": 2500,
    "available": false,
    "licensePlate": "MH12TR4567"
  },
  {
    "id": 2,
    "brand": "Hyundai",
    "model": "Creta",
    "year": 2028,
    "color": "Black",
    "type": "SUVE",
    "pricePerDay": 3000,
    "available": false,
    "licensePlate": "MH12TR4007"
  },
  {
    "id": 3,
    "brand": "Suzuki",
    "model": "Creta",
```


Get All Available cars:

Request URL

```
http://localhost:8080/api/cars/available
```

Server response

Code	Details
200	Response body


```
[
  {
    "id": 5,
    "brand": "Toyota",
    "model": "Innova Crysta",
    "year": 2023,
    "color": "Silver",
    "type": "SUV",
    "pricePerDay": 2800,
    "available": true,
    "licensePlate": "MH15TC1235"
  },
  {
    "id": 6,
    "brand": "Tata",
    "model": "Innova Crysta",
    "year": 2020,
    "color": "Grey",
    "type": "SUVE",
    "pricePerDay": 2800,
    "available": true,
    "licensePlate": "MH15TC1200"
  },
  {
    "id": 9,
    "brand": "Honda",
    "model": "City",
```


Get Car ith ID:

Request URL

```
http://localhost:8080/api/cars/1
```

Server response

Code	Details
200	Response body

```
{
  "id": 1,
  "brand": "Hyundai",
  "model": "Creta",
  "year": 2022,
  "color": "White",
  "type": "SUV",
  "pricePerDay": 2500,
  "available": false,
  "licensePlate": "MH12TR4567"
}
```

Update car details with ID:

Request URL

```
http://localhost:8080/api/cars/1
```

Server response

Code	Details
200	Response body

```
{
  "id": 1,
  "brand": "Tata",
  "model": "Harrier",
  "year": 2024,
  "color": "Red",
  "type": "SUV",
  "pricePerDay": 3200,
  "available": true,
  "licensePlate": "MH14YZ9001"
}
```

- Create Reservation API

reservation-controller

GET	/api/reservations
POST	/api/reservations
DELETE	/api/reservations/{id}

Request URL

```
http://localhost:8080/api/reservations
```

Server response

Code	Details
200	Response body


```
{
  "id": 6,
  "carId": 10,
  "carBrand": "Maruti Suzuki",
  "carModel": "Swift",
  "startDate": "2025-06-27",
  "endDate": "2025-06-27",
  "pickupLocation": "Pune",
  "dropoffLocation": "mumbai",
  "status": "PENDING"
}
```

- View Reservation (Customer)

Request URL

```
http://localhost:8080/api/reservations
```

Server response

Code	Details
200	Response body


```
[
  {
    "id": 6,
    "carId": 10,
    "carBrand": "Maruti Suzuki",
    "carModel": "Swift",
    "startDate": "2025-06-27",
    "endDate": "2025-06-27",
    "pickupLocation": "Pune",
    "dropoffLocation": "mumbai",
    "status": "PENDING"
  },
  {
    "id": 7,
    "carId": 11,
    "carBrand": "Skoda",
    "carModel": "Slavia",
    "startDate": "2025-06-27",
    "endDate": "2025-06-27",
    "pickupLocation": "Pune",
    "dropoffLocation": "mumbai",
    "status": "PENDING"
  }
]
```


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

id	dropoff_location	end_date	pickup_location	start_date	status	car_id	user_id
2	Mumbai	2025-07-05	Pune	2025-07-01	PENDING	2	1
3	Pune	2025-07-05	Goa	2025-04-18	PENDING	3	1
4	Pune	2025-07-05	Goa	2025-04-18	PENDING	4	1
5	Chennai	2025-06-26	Bengluru	2025-06-26	PENDING	1	1
6	mumbai	2025-06-27	Pune	2025-06-27	PENDING	10	8
7	mumbai	2025-06-27	Pune	2025-06-27	PENDING	11	8
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Make Payment API

payment-controller

POST `/api/payments`

GET `/api/payments/reservation/{reservationId}`

Request URL

```
http://localhost:8080/api/payments
```

Server response

Code	Details
200	**Response body**


```
{
  "id": 3,
  "amount": 10000,
  "paymentMethod": "ONLINE",
  "paymentStatus": "SUCCESS",
  "paymentDate": "2025-06-27T15:37:25.482241",
  "reservation": {
    "id": 6,
    "startDate": "2025-06-27",
    "endDate": "2025-06-27",
    "status": "PENDING",
    "pickupLocation": "Pune",
    "dropoffLocation": "mumbai"
  }
}
```

Request URL

```
http://localhost:8080/api/payments/reservation/2
```

Server response

Code	Details
200	Response body


```
{
  "id": 1,
  "amount": 3000,
  "paymentMethod": "CARD",
  "paymentStatus": "SUCCESS",
  "paymentDate": "2025-06-26T07:04:21.744506",
  "reservation": {
    "id": 2,
    "startDate": "2025-07-01",
    "endDate": "2025-07-05",
    "status": "PENDING",
    "pickupLocation": "Pune",
    "dropoffLocation": "Mumbai"
  }
}
```

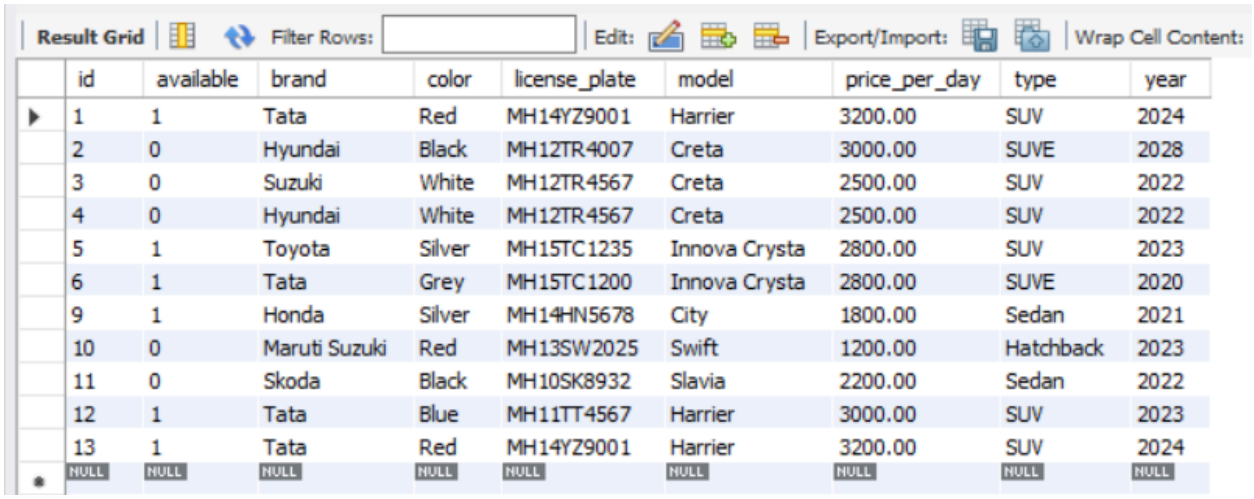

- **Database view in MySQL for users, cars, reservations**

User:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	address	email	name	password	phone	role
1	Pune, India	shraddha@example.com	Shraddha	$2a$10$HCcSmHcCIOqAvL/w0sFUrubK5lNPWW...	9876543210	CUSTOMER
2	Pune, India	vasu@example.com	Vasudha	$2a$10$eoA9i/KhugnM7rE7hmDmQeryMua2jd...	8776543210	CUSTOMER
3	Karad, India	ro@example.com	Rahul	$2a$10$fmLcVJKriRU4BviezjzycueITA3JHEsSnL...	8076543210	CUSTOMER
4	HQ	admin@example.com	Admin	$2a$10$tvr34H7vwzItUhuXEqTAH.3Uq9h5Odro...	9999999999	CUSTOMER
5	Mumbai	avani@123.com	Avani	$2a$10$MbtI5PQLIU5dF8oxJaOwDe8d2QYJQE...	6547895236	CUSTOMER
6	Bengluru	virat@123.com	Virat	$2a$10$L2N3KD4J.B8/.PuAM5G5uel4lw4w.VhLr...	5263895236	CUSTOMER
7	Killari	sonali@123.com	Sonali	$2a$10$3d1jyqqdH4xAXdvjvJ.H/Oi8Pnj3QwmH...	9874563214	CUSTOMER
8	PUNE	shraddhagavkare13@gmail.com	Shraddha G	$2a$10$tJhE1LZ9.E0nUWNscjYB6uxLNFPXdrXr...	8766828269	CUSTOMER
NULL	NULL	NULL	NULL	NULL	NULL	NULL

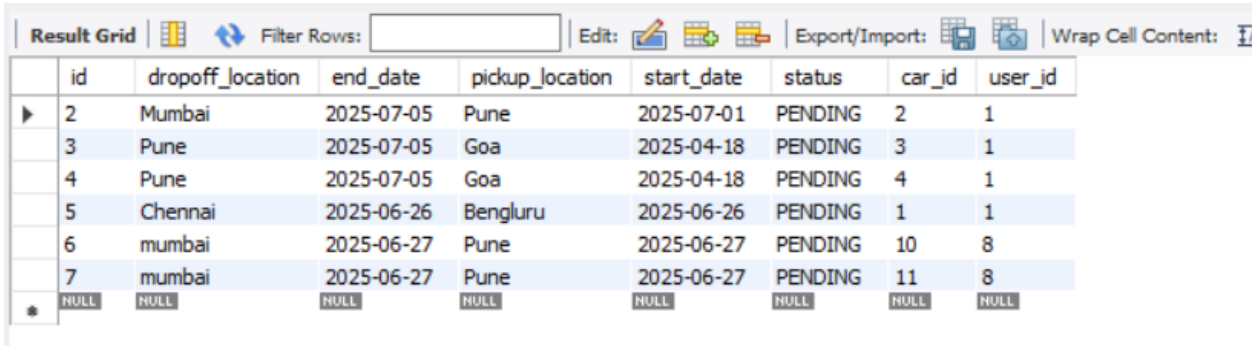
Car:



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	available	brand	color	license_plate	model	price_per_day	type	year
1	1	Tata	Red	MH14YZ9001	Harrier	3200.00	SUV	2024
2	0	Hyundai	Black	MH12TR4007	Creta	3000.00	SUVE	2028
3	0	Suzuki	White	MH12TR4567	Creta	2500.00	SUV	2022
4	0	Hyundai	White	MH12TR4567	Creta	2500.00	SUV	2022
5	1	Toyota	Silver	MH15TC1235	Innova Crysta	2800.00	SUV	2023
6	1	Tata	Grey	MH15TC1200	Innova Crysta	2800.00	SUVE	2020
9	1	Honda	Silver	MH14HN5678	City	1800.00	Sedan	2021
10	0	Maruti Suzuki	Red	MH13SW2025	Swift	1200.00	Hatchback	2023
11	0	Skoda	Black	MH10SK8932	Slavia	2200.00	Sedan	2022
12	1	Tata	Blue	MH11TT4567	Harrier	3000.00	SUV	2023
13	1	Tata	Red	MH14YZ9001	Harrier	3200.00	SUV	2024
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

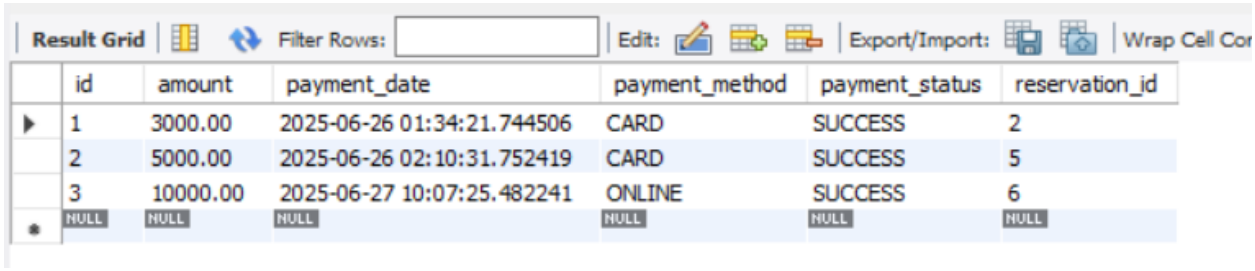
Reservation:



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	dropoff_location	end_date	pickup_location	start_date	status	car_id	user_id
2	Mumbai	2025-07-05	Pune	2025-07-01	PENDING	2	1
3	Pune	2025-07-05	Goa	2025-04-18	PENDING	3	1
4	Pune	2025-07-05	Goa	2025-04-18	PENDING	4	1
5	Chennai	2025-06-26	Bengluru	2025-06-26	PENDING	1	1
6	mumbai	2025-06-27	Pune	2025-06-27	PENDING	10	8
7	mumbai	2025-06-27	Pune	2025-06-27	PENDING	11	8
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Payment:



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Cor

id	amount	payment_date	payment_method	payment_status	reservation_id
1	3000.00	2025-06-26 01:34:21.744506	CARD	SUCCESS	2
2	5000.00	2025-06-26 02:10:31.752419	CARD	SUCCESS	5
3	10000.00	2025-06-27 10:07:25.482241	ONLINE	SUCCESS	6
NULL	NULL	NULL	NULL	NULL	NULL

User Controller:

user-controller

GET /api/users

GET /api/users/me

Request URL

```
http://localhost:8080/api/users
```

Server response

Code	Details
200	Response body


```
[
  {
    "id": 1,
    "name": "Shraddha",
    "email": "shraddha@example.com",
    "password": "$2a$10$HCcSmHcCIOqAvL/w0sFUrubK5lNPWWNPzH8.SNNuHlXx3HH0po4lS",
    "role": "CUSTOMER",
    "phone": "9876543210",
    "address": "Pune, India"
  },
  {
    "id": 2,
    "name": "Vasudha",
    "email": "vasu@example.com",
    "password": "$2a$10$eoA9i/KhugnM7rE7hmDmQeryMua2jdWg6r99EZ612OeHbrL0vWnK6",
    "role": "CUSTOMER",
    "phone": "8776543210",
    "address": "Pune, India"
  },
  {
    "id": 3,
    "name": "Rahul",
    "email": "ro@example.com",
    "password": "$2a$10$fmLcVJKriRU4BviezjzycueITA3JHEsSnLqgSEq7cfNv4syppXVJS",
    "role": "CUSTOMER",
    "phone": "8076543210",
    "address": "Karad, India"
```

Project Flow Summary:

1. User logs in and gets JWT token.
2. Customer can view cars, reserve a car, and pay for it.
3. Admin can add/update/delete cars and view all reservations.
4. Token ensures that only allowed users access APIs.
5. Backend handles validation, business logic, and database interaction.

Hash Algorithm:

- Algorithm used: HMAC-SHA256 (HS256)
- Token is signed with a 256-bit secret key using jjwt (Java JWT) library.
- Signature ensures token is not tampered with.

Additional Notes:

- Token expiration is set to 24 hours (86400000 ms)

- Custom exception handling returns user-friendly error messages.
- Swagger and Springdoc are used for API documentation and testing.
- JPA handles database operations using repositories.