TrimbleCarsApplication.java

package com.trimble.cars;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class TrimbleCarsApplication {

public static void main(String[] args) {

SpringApplication.run(TrimbleCarsApplication.class, args);

}

}

Car.java

package com.trimble.cars.model;

import javax.persistence.\*;

@Entity

public class Car {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String owner; // Car owner or registrant

private String model;

private String status; // "Ideal", "On Lease", "On Service"

public Car() { }

public Car(String owner, String model) {

this.owner = owner;

this.model = model;

this.status = "Ideal"; // Default status when a car is registered

}

// Getters and setters

public Long getId() { return id; }

public String getOwner() { return owner; }

public void setOwner(String owner) { this.owner = owner; }

public String getModel() { return model; }

public void setModel(String model) { this.model = model; }

public String getStatus() { return status; }

public void setStatus(String status) { this.status = status; }

}

Lease.java

package com.trimble.cars.model;

import javax.persistence.\*;

import java.time.LocalDateTime;

@Entity

public class Lease {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "car\_id")

private Car car;

private String leasedBy; // End customer identifier

private LocalDateTime leaseStart;

private LocalDateTime leaseEnd; // Null if lease is active

public Lease() { }

public Lease(Car car, String leasedBy) {

this.car = car;

this.leasedBy = leasedBy;

this.leaseStart = LocalDateTime.now();

}

// Getters and setters

public Long getId() { return id; }

public Car getCar() { return car; }

public void setCar(Car car) { this.car = car; }

public String getLeasedBy() { return leasedBy; }

public void setLeasedBy(String leasedBy) { this.leasedBy = leasedBy; }

public LocalDateTime getLeaseStart() { return leaseStart; }

public void setLeaseStart(LocalDateTime leaseStart) { this.leaseStart = leaseStart; }

public LocalDateTime getLeaseEnd() { return leaseEnd; }

public void setLeaseEnd(LocalDateTime leaseEnd) { this.leaseEnd = leaseEnd; }

}

CarRepository.java

package com.trimble.cars.repository;

import com.trimble.cars.model.Car;

import org.springframework.data.jpa.repository.JpaRepository;

public interface CarRepository extends JpaRepository<Car, Long> { }

LeaseRepository.java

package com.trimble.cars.repository;

import com.trimble.cars.model.Lease;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface LeaseRepository extends JpaRepository<Lease, Long> {

List<Lease> findByLeasedBy(String leasedBy);

List<Lease> findByCarId(Long carId);

}

CarService.java

package com.trimble.cars.service;

import com.trimble.cars.model.Car;

import com.trimble.cars.repository.CarRepository;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CarService {

private final CarRepository carRepo;

public CarService(CarRepository carRepo) {

this.carRepo = carRepo;

}

// Register a new car

public Car registerCar(Car car) {

car.setStatus("Ideal");

return carRepo.save(car);

}

public Car getCar(Long id) {

return carRepo.findById(id).orElseThrow(() -> new RuntimeException("Car not found"));

}

public List<Car> getAllCars() {

return carRepo.findAll();

}

// Update the status of the car

public Car updateCarStatus(Long carId, String status) {

Car car = getCar(carId);

car.setStatus(status);

return carRepo.save(car);

}

}

LeaseService.java

package com.trimble.cars.service;

import com.trimble.cars.model.Car;

import com.trimble.cars.model.Lease;

import com.trimble.cars.repository.LeaseRepository;

import org.springframework.stereotype.Service;

import java.time.LocalDateTime;

import java.util.List;

@Service

public class LeaseService {

private final LeaseRepository leaseRepo;

private final CarService carService;

private static final int MAX\_ACTIVE\_LEASES = 2; // Maximum active leases per customer

public LeaseService(LeaseRepository leaseRepo, CarService carService) {

this.leaseRepo = leaseRepo;

this.carService = carService;

}

// Start a lease on a car

public Lease startLease(Long carId, String leasedBy) {

long activeLeases = leaseRepo.findByLeasedBy(leasedBy)

.stream().filter(l -> l.getLeaseEnd() == null).count();

if (activeLeases >= MAX\_ACTIVE\_LEASES) {

throw new RuntimeException("Maximum active leases reached for user: " + leasedBy);

}

Car car = carService.getCar(carId);

if (!"Ideal".equalsIgnoreCase(car.getStatus())) {

throw new RuntimeException("Car is not available for lease.");

}

Lease lease = new Lease(car, leasedBy);

leaseRepo.save(lease);

carService.updateCarStatus(carId, "On Lease");

return lease;

}

// End a lease

public Lease endLease(Long leaseId) {

Lease lease = leaseRepo.findById(leaseId)

.orElseThrow(() -> new RuntimeException("Lease not found"));

if (lease.getLeaseEnd() != null) {

throw new RuntimeException("Lease already ended.");

}

lease.setLeaseEnd(LocalDateTime.now());

leaseRepo.save(lease);

carService.updateCarStatus(lease.getCar().getId(), "Ideal");

return lease;

}

public List<Lease> getLeasesForCar(Long carId) {

return leaseRepo.findByCarId(carId);

}

public List<Lease> getLeasesForCustomer(String leasedBy) {

return leaseRepo.findByLeasedBy(leasedBy);

}

}

CarController.java

package com.trimble.cars.controller;

import com.trimble.cars.model.Car;

import com.trimble.cars.model.Lease;

import com.trimble.cars.service.CarService;

import com.trimble.cars.service.LeaseService;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api")

public class CarController {

private final CarService carService;

private final LeaseService leaseService;

public CarController(CarService carService, LeaseService leaseService) {

this.carService = carService;

this.leaseService = leaseService;

}

// Car owner registers a car

@PostMapping("/car/register")

public ResponseEntity<Car> registerCar(@RequestBody Car car) {

Car newCar = carService.registerCar(car);

return ResponseEntity.ok(newCar);

}

// Get car details by id

@GetMapping("/car/{id}")

public ResponseEntity<Car> getCar(@PathVariable Long id) {

Car car = carService.getCar(id);

return ResponseEntity.ok(car);

}

// Get lease history for a car

@GetMapping("/car/{id}/leases")

public ResponseEntity<List<Lease>> getCarLeaseHistory(@PathVariable Long id) {

List<Lease> leases = leaseService.getLeasesForCar(id);

return ResponseEntity.ok(leases);

}

// End customer starts a lease

@PostMapping("/lease/start")

public ResponseEntity<Lease> startLease(@RequestParam Long carId, @RequestParam String leasedBy) {

Lease lease = leaseService.startLease(carId, leasedBy);

return ResponseEntity.ok(lease);

}

// Admin: Get all cars

@GetMapping("/admin/cars")

public ResponseEntity<List<Car>> getAllCars() {

List<Car> cars = carService.getAllCars();

return ResponseEntity.ok(cars);

}

// Admin: End a lease

@PostMapping("/admin/lease/end")

public ResponseEntity<Lease> endLease(@RequestParam Long leaseId) {

Lease lease = leaseService.endLease(leaseId);

return ResponseEntity.ok(lease);

}

// Admin: Get leases for a customer

@GetMapping("/admin/leases")

public ResponseEntity<List<Lease>> getCustomerLeases(@RequestParam String leasedBy) {

List<Lease> leases = leaseService.getLeasesForCustomer(leasedBy);

return ResponseEntity.ok(leases);

}

}

CarServiceTest.java

package com.trimble.cars.service;

import com.trimble.cars.model.Car;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.context.SpringBootTest;

import static org.junit.jupiter.api.Assertions.\*;

@SpringBootTest

public class CarServiceTest {

@Autowired

private CarService carService;

@Test

public void testRegisterCar() {

Car car = new Car("Alice", "Honda Civic");

Car savedCar = carService.registerCar(car);

assertNotNull(savedCar.getId());

assertEquals("Honda Civic", savedCar.getModel());

assertEquals("Ideal", savedCar.getStatus());

}

}