```java
package com.ridersharing;


import jakarta.persistence.*;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.time.LocalDateTime;
import java.util.List;

// Main Spring Boot Application
@SpringBootApplication
public class RideSharingBackend {
    public static void main(String[] args) {
        SpringApplication.run(RideSharingBackend.class, args);
    }
}

// Entity: User
@Entity
@Table(name = "users")
class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String phoneOrEmail;
    private String password;

    // Getters and Setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getPhoneOrEmail() { return phoneOrEmail; }
    public void setPhoneOrEmail(String phoneOrEmail) { this.phoneOrEmail = phoneOrEmail; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
}

// Entity: RideRequest
@Entity
@Table(name = "ride_requests")
class RideRequest {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String pickupLocation;
    private String destination;
    private LocalDateTime requestTime;

    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;

    // Getters and Setters
```

```
59      public Long getId() { return id; }
60      public void setId(Long id) { this.id = id; }
61      public String getPickupLocation() { return pickupLocation; }
62      public void setPickupLocation(String pickupLocation) { this.pickupLocation =
    pickupLocation; }
63      public String getDestination() { return destination; }
64      public void setDestination(String destination) { this.destination = destination;
    }
65      public LocalDateTime getRequestTime() { return requestTime; }
66      public void setRequestTime(LocalDateTime requestTime) { this.requestTime =
    requestTime; }
67      public User getUser() { return user; }
68      public void setUser(User user) { this.user = user; }
69  }
70
71  // Entity: ScheduledRide
72  @Entity
73  @Table(name = "scheduled_rides")
74  class ScheduledRide {
75      @Id
76      @GeneratedValue(strategy = GenerationType.IDENTITY)
77      private Long id;
78
79      private String pickupLocation;
80      private String destination;
81      private LocalDateTime scheduledTime;
82      private String groupName;
83
84      @ManyToOne
85      @JoinColumn(name = "user_id")
86      private User user;
87
88      // Getters and Setters
89      public Long getId() { return id; }
90      public void setId(Long id) { this.id = id; }
91      public String getPickupLocation() { return pickupLocation; }
92      public void setPickupLocation(String pickupLocation) { this.pickupLocation =
    pickupLocation; }
93      public String getDestination() { return destination; }
94      public void setDestination(String destination) { this.destination = destination;
    }
95      public LocalDateTime getScheduledTime() { return scheduledTime; }
96      public void setScheduledTime(LocalDateTime scheduledTime) { this.scheduledTime =
    scheduledTime; }
97      public String getGroupName() { return groupName; }
98      public void setGroupName(String groupName) { this.groupName = groupName; }
99      public User getUser() { return user; }
100     public void setUser(User user) { this.user = user; }
101 }
102
103 // Repository: UserRepository
104 interface UserRepository extends JpaRepository<User, Long> {
105     User findByPhoneOrEmail(String phoneOrEmail);
106 }
107
108 // Repository: RideRequestRepository
109 interface RideRequestRepository extends JpaRepository<RideRequest, Long> {
110     List<RideRequest> findByUserId(Long userId);
111 }
112
113 // Repository: ScheduledRideRepository
```

```
interface ScheduledRideRepository extends JpaRepository<ScheduledRide, Long> {
    List<ScheduledRide> findByUserId(Long userId);
}

// DTO: LoginRequest
class LoginRequest {
    private String phoneOrEmail;
    private String password;

    public String getPhoneOrEmail() { return phoneOrEmail; }
    public void setPhoneOrEmail(String phoneOrEmail) { this.phoneOrEmail = phoneOrEmail; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
}

// DTO: RideRequestDTO
class RideRequestDTO {
    private String pickupLocation;
    private String destination;

    public String getPickupLocation() { return pickupLocation; }
    public void setPickupLocation(String pickupLocation) { this.pickupLocation = pickupLocation; }
    public String getDestination() { return destination; }
    public void setDestination(String destination) { this.destination = destination; }
}

// DTO: ScheduleRideDTO
class ScheduleRideDTO {
    private String pickupLocation;
    private String destination;
    private LocalDateTime scheduledTime;
    private String groupName;

    public String getPickupLocation() { return pickupLocation; }
    public void setPickupLocation(String pickupLocation) { this.pickupLocation = pickupLocation; }
    public String getDestination() { return destination; }
    public void setDestination(String destination) { this.destination = destination; }
    public LocalDateTime getScheduledTime() { return scheduledTime; }
    public void setScheduledTime(LocalDateTime scheduledTime) { this.scheduledTime = scheduledTime; }
    public String getGroupName() { return groupName; }
    public void setGroupName(String groupName) { this.groupName = groupName; }
}

// Controller: AuthController
@RestController
@RequestMapping("/api/auth")
class AuthController {
    private final UserRepository userRepository;

    public AuthController(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @PostMapping("/login")
    public ResponseEntity<?> login(@RequestBody LoginRequest loginRequest) {
```

```
169         User user =
    userRepository.findByPhoneOrEmail(loginRequest.getPhoneOrEmail());
170         if (user == null || !user.getPassword().equals(loginRequest.getPassword()))
    {
171             return ResponseEntity.badRequest().body("Invalid credentials");
172         }
173         return ResponseEntity.ok("Login successful for user: " +
    user.getPhoneOrEmail());
174     }
175 }
176
177 // Controller: RideController
178 @RestController
179 @RequestMapping("/api/rides")
180 class RideController {
181     private final UserRepository userRepository;
182     private final RideRequestRepository rideRequestRepository;
183     private final ScheduledRideRepository scheduledRideRepository;
184
185     public RideController(UserRepository userRepository, RideRequestRepository
    rideRequestRepository, ScheduledRideRepository scheduledRideRepository) {
186         this.userRepository = userRepository;
187         this.rideRequestRepository = rideRequestRepository;
188         this.scheduledRideRepository = scheduledRideRepository;
189     }
190
191     @PostMapping("/request")
192     public ResponseEntity<?> requestRide(@RequestBody RideRequestDTO rideRequestDTO,
    @RequestParam Long userId) {
193         User user = userRepository.findById(userId).orElse(null);
194         if (user == null) {
195             return ResponseEntity.badRequest().body("User not found");
196         }
197
198         RideRequest rideRequest = new RideRequest();
199         rideRequest.setPickupLocation(rideRequestDTO.getPickupLocation());
200         rideRequest.setDestination(rideRequestDTO.getDestination());
201         rideRequest.setRequestTime(LocalDateTime.now());
202         rideRequest.setUser(user);
203
204         rideRequestRepository.save(rideRequest);
205         return ResponseEntity.ok("Ride requested successfully");
206     }
207
208     @PostMapping("/schedule")
209     public ResponseEntity<?> scheduleRide(@RequestBody ScheduleRideDTO
    scheduleRideDTO, @RequestParam Long userId) {
210         User user = userRepository.findById(userId).orElse(null);
211         if (user == null) {
212             return ResponseEntity.badRequest().body("User not found");
213         }
214
215         ScheduledRide scheduledRide = new ScheduledRide();
216         scheduledRide.setPickupLocation(scheduleRideDTO.getPickupLocation());
217         scheduledRide.setDestination(scheduleRideDTO.getDestination());
218         scheduledRide.setScheduledTime(scheduleRideDTO.getScheduledTime());
219         scheduledRide.setGroupName(scheduleRideDTO.getGroupName());
220         scheduledRide.setUser(user);
221
222         scheduledRideRepository.save(scheduledRide);
223         return ResponseEntity.ok("Ride scheduled successfully");
```

```
224     }
225
226     @GetMapping("/user/{userId}/requests")
227     public ResponseEntity<List<RideRequest>> getUserRideRequests(@PathVariable Long
    userId) {
228         return ResponseEntity.ok(rideRequestRepository.findByUserId(userId));
229     }
230
231     @GetMapping("/user/{userId}/schedules")
232     public ResponseEntity<List<ScheduledRide>> getUserScheduledRides(@PathVariable
    Long userId) {
233         return ResponseEntity.ok(scheduledRideRepository.findByUserId(userId));
234     }
235 }
236
```