1. Coding Standards

1. java

1. class, interface, enums

- pascal case
    - eg - GuestDao, HostDto

2. Enums

- should be appended with Enum.
- should be decalred as seprate files

3. method,interface,fields

- camel case
    - eg - guestDao, hostDto

4. final fields , final static fields, enum constants

- SCREAMING_SNAKE_CASE
    - eg- public enum RoleEnum{HOST, GUEST, ADMIN}

5. packages

- package name - snakecase and all lower and singular
    - base_package - com.rentmycar
    - **all packages**

```
com.rentmycar.exception_handler
com.rentmycar.security
com.rentmycar.custom_exception
com.rentmycar.dao
com.rentmycar.dto
com.rentmycar.entity
com.rentmycar.aspect
com.rentmycar.service
```

6. Exceptions and status codes

- **if already registerd** -
    - SC 409 , conflict
    - **exception** -
        - ConflictException("user with given email and mobile already registered");
        - ConflictException("user with given email already registered");

        - ConflictException("user with given mobile already registered");
- **if password and confirm password doesn't match** -
    - SC 400 , bad request
    - **exception** - ConstraintViolationException("password mismatch", null);
- **if expiry date is before creation date**
    - SC 400 , bad request
    - **exception** - ConstraintViolationException("expiry should come after issue date of dl ", null);

2. mysql

- all table name should be snake case and lower and singular
- all column names must be snake case and lower
    - eg- user_id
- **foreign key must be 'table_name' + '_' + 'id'**
- 'user' table -> id (from mapped super class)
- 'adress' table -> user_id (foreign key to user(id))
    - user 1 <------ * address

3. git

1. **to careate a branch and switch to that branch**

```
git switch -C <feature-name>
git switch -C feature/create-guest-entity
```

- ***branch creation coding standard***
    - Feature Branc hes (feature/):
        - Purpose: For developing new features or enhancements.
            - Naming Example: feature/add-user-profile, feature/implement-searchfunctionality
    - Bugfix Branches (bugfix/ or fix/):
        - Purpose: For fixing bugs or issues in the codebase.
        - Naming Example: bugfix/fix-login-error, fix/missing-footer-on-homepage
    - Documentation Branches (docs/):
        - Purpose: For updates or changes to documentation.
        - Naming Example: docs/update-readme, docs/improve-api-docs
    - Chore Branches (chore/):
        - Purpose: For maintenance tasks that don’t directly affect functionality, such as refactoring or updating dependencies.
        - Naming Example: chore/update-dependencies, chore/refactor-codebase

2. **to check the all the branches and current branch**

```
git branch
```

3. **to add the changes to local repo**

```
git add .
```

4. **to commit the changes to local repo**

```
git commit -m "feat: add guest entity"
```

- ***commit creation coding standard***
  - **feat** : New features or enhancements
    - **Example** - feat: add user profile page
    -
  - **fix** : Bug fixes
    - **Example** - fix: correct header image rendering
    -
  - **docs**: Documentation changes
    - **Example** - docs: update README with new setup instructions
    -
  - **style**: Formatting, missing semi-colons, etc. (no code change)
    - **Example** - style: format code according to eslint
    -
  - **refactor**: Code changes that neither fix bugs nor add features
    - **Example** - refactor: improve variable naming in utils.js
    -
  - **test**: Adding or updating tests
    - **Example** - test: add unit tests for user login functionality
    -
  - **chore**: Changes to the build process or auxiliary tools/libraries
    - **Example** - chore: update dependencies

5. **to push the changes to remote branch**

```
git push -u origin feature/create-guest-entity
```

6. **to pull the changes to main branch**

```
git switch main
git fetch
```

```
git pull origin main
```

2. api descptions

1. **user APIs**

1. UserController (@RequestMapping=/user)

1.Login Guest

- **URL** - http://host:port/user/login
- **Method** - POST
- **payload** -

```
    public class SignInRequestDto {

    @NotEmpty(message = "Email must be supplied !")
    @Email(message = "Invalid Email format")
    private String email;
    @Pattern(regexp = "((?=.*\\d)(?=.*[a-z])(?=.*[#@$*]).{5,20})", message =
"Blank or Invalid password!!!!")
    private String password;

    private UserRoleEnum roleEnum;
}
```

- **Successful Resp** - SC 201

```
public class SignInResponseDto extends BaseDto {

    private String firstName;
    private String lastName;
    private String email;
    private String mobile;
    private UserRoleEnum roleEnum;

}
```

```
Swagger Response
{
  "id": 1,
  "createdOn": "2024-08-07",
  "updatedOn": "2024-08-07T12:34:56",
  "firstName": "John",
```

```
  "lastName": "Doe",
  "email": "doe@example.com",
  "mobile": "1234567890",
  "roleEnum": "ADMIN"
}
```

- **Error resp** - SC 400 ,
  - **if isdeleted = 0** - mesg - your account is inactive
  - **if user is not found** -mesg - user not found
  -

---

1. register User with basic details

- **URL** - http://host:port/user/register_basic
- **Method** - POST
- **payload** -

```
@Setter
@Getter
public class RegisterUserReqDto {

    private Long id;

    @NotBlank(message = "firstName must not be blannk")
    String firstName; // VARCHAR(20),NOT NULL

    @NotBlank(message = "last name must have a value")
    String lastName; // VARCHAR(20),NOT NULL

    @Pattern(regexp = "^[6-9]\\d{9}$", message = "Invalid mobile number. It should
be exactly 10 digits long and start with 6, 7, 8, or 9.")
    String mobile; // VARCHAR(12),UNIQUE NOT NULL

    @NotBlank
    @Email(message = "invalid email id ")
    String email; // VARCHAR(25),UNIQUE NOT NULL

    @NotBlank
    @Pattern(regexp = "((?=.*\\d)(?=.*[a-z])(?=.*[#@$*]).{5,20})", message =
"Blank or Invalid password!!!!")
    String password; // VARCHAR(70), NOT NULL

    @NotBlank
    @Pattern(regexp = "((?=.*\\d)(?=.*[a-z])(?=.*[#@$*]).{5,20})", message =
"Blank or Invalid password!!!!")
    String confirmPassword; // VARCHAR(70), NOT NULL

    @NotNull
```

```
    private UserRoleEnum roleEnum;
}
```

- **Successful Resp** - SC 201 HttpStatus.CREATED

```
@Getter
@Setter
public class RegisterUserResDto extends BaseDto {
    private Long id;
    String firstName; // VARCHAR(20),NOT NULL
    String lastName; // VARCHAR(20),NOT NULL
    String mobile; // VARCHAR(12),UNIQUE NOT NULL
    String email; // VARCHAR(25),UNIQUE NOT NULL
    private UserRoleEnum roleEnum;
}
```

```
{
  "id": 1,
  "createdOn": "2024-08-08",
  "updatedOn": "2024-08-08T10:24:23.556208",
  "firstName": "asheesh",
  "lastName": "patel",
  "mobile": "6720003609",
  "email": "user@gmailff.com",
  "roleEnum": "HOST"
}
```

- **Error resp** -
  - **if already registerd** -
    - SC 409 , conflict
    - **exception** -
      - ConflictException("user with given email and mobile already registered");
      - ConflictException("user with given email already registered");
      - ConflictException("user with given mobile already registered");
  - **if password and confirm password doesn't match** -
    - SC 400 , bad request
    - **exception** - ConstraintViolationException("password mismatch", null);
  - **if expiry date is before creation date**
    - SC 400 , bad request
    - **exception** - ConstraintViolationException("expiry should come after issue date of dl ", null);

---

2. register User with driving license

- **URL** - http://host:port/user/register_with_dl
- **Method** - POST
- **payload** -

```
public class RegisterUserWithDlReqDto extends RegisterUserReqDto {

    @NotNull
    DrivingLicenseDto drivingLicenseDto;
}
```

```
{
 "firstName": "Jane",
  "lastName": "Smith",
  "mobile": "9876543215",
  "email": "smith@example.com",
  "password": "another@456",
  "confirmPassword": "another@456",
  "roleEnum": "GUEST",
  "drivingLicenseDto": {
    "drivingLicenseNo": "XYZ987554323",
    "issueDate": "2024-08-08",
    "expirationDate": "2026-08-08",
    "licenseClassEnum": "B"
  }
}
```

- **Successful Resp** - SC 201 HttpStatus.CREATED

```
@Data
public class RegisterUserWithDlResDto extends RegisterUserResDto {

    DrivingLicenseDto drivingLicenseDto;

}
```

```
//Response body
{
  "id": 2,
  "createdOn": "2024-08-08",
  "updatedOn": "2024-08-08T23:39:40.042376",
  "firstName": "Jane",
  "lastName": "Smith",
  "mobile": "9876543215",
  "email": "smith@example.com",
  "roleEnum": "GUEST",
  "drivingLicenseDto": {
```

```
    "id": 2,
    "drivingLicenseNo": "XYZ987554323",
    "issueDate": "2024-08-08",
    "expirationDate": "2026-08-08",
    "licenseClassEnum": "B"
  }
}
```

- **Error resp** -
  - **if already registerd** -
    - SC 409 , conflict
    - **exception** -
      - ConflictException("user with given email and mobile already registered");
      - ConflictException("user with given email already registered");
      - ConflictException("user with given mobile already registered");
  - **if password and confirm password doesn't match** -
    - SC 400 , bad request
    - **exception** - ConstraintViolationException("password mismatch", null);
  - **if expiry date is before creation date**
    - SC 400 , bad request
    - **exception** - ConstraintViolationException("expiry should come after issue date of dl ", null);

1. **guest APIs**

1. GuestController (@RequestMapping=/guest)

3.Get guest details (basic details + driving license)

- **URL** - http://host:port/guest/profile/{guestId}
- **Method** - Get
- **Successful Resp** - SC 201

```
public class GuestDetailsResponseDto extends BaseDto{

    private String firstName;

    private String lastName;

    private String mobile;

    private String email;

    private String password;

    private DrivingLicenseDto drivingLicenseDto;
}
```

```
public class DrivingLicenseDto{

    private Long id;

    private String drivingLicenseNo;

    private LocalDate issueDate;

    private LocalDate expirationDate;

    private LicenseClassEnum licenseClassEnum;
}
```

```
// Swagger Response
{
  "id": 1,
  "createdOn": "2024-08-07",
  "updatedOn": "2024-08-07T12:34:56",
  "firstName": "John",
  "lastName": "Doe",
  "mobile": "1234567890",
  "email": "doe@example.com",
  "password": "secure#123",
  "drivingLicenseDto": {
    "id": 1,
    "drivingLicenseNo": "ABC123456789",
    "issueDate": "2023-08-07",
    "expirationDate": "2025-08-07",
    "licenseClassEnum": "A"
  }
}
```

- **Error resp** -

4.Update Guest Basic Details By GuestId

- **URL** - http://host:port/guest/update/{guestId}
- **Method** - PUT
- **payload** - GuestUpdateDto (firstName,lastName,email,password)
- **Successful Resp** - SC 201 GuestResponseDto + mesg (ApiResponse)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

5.Delete Guest

- **URL** - http://host:port/guest/delete/{hostId}
- **Method** - PATCH

- **Successful Resp** - SC 201 GuestResponseDto + mesg (ApiResponse)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

6.Update Driving License By GuestId

- **URL** - http://host:port/guest/update/{guestId}/{dl_update}
- **Method** - PUT
- **payload** - GuestUpdateDto (firstName,lastName,email,password,drivingLicenseNo,issueDate,expirationDate,licenseClass)
- **Successful Resp** - SC 201 GuestResponseDto + mesg (ApiResponse)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

---

2. CarController (@RequestMapping=/car)

1.Get CarCards By City,pickupDateTime,dropOffDateTime - public api

- **URL** - http://host:port/car/get_cars_by_city?city=value,pickupDateTime,dropOffDateTime
- **Method** - GET
- **Successful Resp** - SC 201 * Successful Resp - SC 201 ArrayList CarsCardDetailsDto(brand,model,transmissionTypeEnum,fuelTypeEnum,seatingCapacity,noOfTrips,carPricePerHr,carPricePerDay)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

2.Get CarCards Details By CarId(Car + Features)

- **URL** - http://host:port/car/get_specific_car_details/{carId}
- **Method** - GET
- **Successful Resp** - SC 201 CompleteCarDetailsDto + mesg (ApiResponse) *ArrayList CarsCardDetailsDto(brand,model,transmissionTypeEnum,fuelTypeEnum,seatingCapacity,noOfTrips,carPricePerHr,carPricePerDay)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

---

3. BookingController (@RequestMapping=/booking)

1.Add Booking by guestId, CarId, AddressId of guest

- **URL** - http://host:port/booking/add_booking/gues_id/{guestId}/car_id/{carId}/address_id/{addressId}
- **Method** - POST
- **payload** - AddBookingDto(pickupDateTime,dropOffDateTime,double ammount)
- **Successful Resp** - SC 201 HostResponseDto + mesg (ApiResponse)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

2.update booking after payment is made

- **URL** - http://host:port/booking/booking_id/{booking_id}
- **Method** - Patch
- **payload** - PaymentRequestDto(ammount,cardNO, cardHoldersName, expiry Date, CVV)

- **Successful Resp** - SC 201 PaymentResponse(Staus,txn time, txnId, ammount)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

3.get upcomming bookings by guest id

- **URL** - http://host:port/booking/upcomming_booking/{guestId}
- **Method** - Get
- **Successful Resp** - SC 201 BookedCarDto(brand, model, transmissionTypeEnum, seatingCapacity, kmDriven, pickupDateTime, dropOffDateTime)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

1. get past bookings by guest id

- **URL** - http://host:port/booking/past_booking/{guestId}
- **Method** - Get
- **Successful Resp** - SC 201 BookedCarDto(brand, model, transmissionTypeEnum, seatingCapacity, kmDriven, pickupDateTime, dropOffDateTime)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

2. **host APIs**

1. HostController(@RequestMapping=/host)

1.Register Host

- **URL** - http://host:port/host/register
- **Method** - POST
- **payload** - RegisterHostDto (firstName,lastName,Mobile,email,password)
- **Successful** Resp - SC 201 HostResponseDto + mesg (ApiResponse)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

2.get Host by host id

- **URL** - http://host:port/host/{hostId}
- **Method** - GET
- **Successful** Resp - SC 201 HostProfileDto(firstName, lastName, email, mobile, password, ArrayList + mesg (ApiResponse)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

3.Update Host By HostId

- **URL** - http://host:port/host/update/{hostId}
- **Method** - PUT
- **payload** - HostUpdateDto (firstName,lastName,mobile,email,password)
- **Successful** Resp - SC 201 HostResponseDto + mesg (ApiResponse)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

---

2. CarController(@RequestMapping=/car)

3.Get CarCards By HostId

- **URL** - http://host:port/car/{hostId}
- **Method** - GET
- **Successful** Resp - SC 201 ArrayList CarsCardDetailsDto(brand,model,transmissionTypeEnum,fuelTypeEnum,seatingCapacity,noOfTrips,carPricePerHr,carPricePerDay)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

2.Add Car By HostId

- **URL** - http://host:port/car/add_car/{hostId}/{addressId}
- **Method** - POST
- **payload** - RegisterCarDto (brand,model,fuelTypeEnum,kmDriven,seatingCapacity,registrationNo,spareTyreCount,transmissionTypeEnum,CarPricingDto,CarFeatureDto) CarPricingDto(pricePerHr,pricePerDay,securityDeposit) CarFeatureDto(hasUsbCharger,hasBluetooth,hasPowerSteering,hasAbs,hasAc,hasAirBags)
- **Successful** Resp - SC 201 CarResponseDto + mesg (ApiResponse)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

5.Get complete Details of Specific Car and it's Address by CarId

- **URL** - http://host:port/car/{hostId}/car_address/{carId}
- **Method** - GET
- **Successful** Resp - SC 201 ArrayList CarsCardDetailsDto(brand,model,transmissionTypeEnum,fuelTypeEnum,seatingCapacity,noOfTrips,carPricePerHr,carPricePerDay)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

6.Upadate Car and its address by hostId , carId

- **URL** - http://host:port/car/{carId}/update_car/{hostId}
- **Method** - PUT
- **payload** - CarUpdateDto (fuelType, kmDriven,fuelMeter,SpareTyre,spareTyreCount, CarPricingDto, AddressDto) CarPricingDto(pricePerDay, pricePerHr, deposit) AddressDto (country,state,city,pincode,area,houseNo)
- **Successful** Resp - SC 201 CarResponseDto + mesg (ApiResponse)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

---

3. AddressController(@RequestMapping=/address)

1.Add Address By UserId

- **URL** - http://host:port/user/{userId}/add_address
- **Method** - POST
- **payload** - AddressDto (country,state,city,pincode,area,houseNo)
- **Successful** Resp - SC 201 AddressResponseDto + mesg (ApiResponse)

- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

3.Update Address by UserId

- **URL** - http://host:port/user/{userId}/address/{addressId}
- **Method** - PUT
- **payload** - AddressDto (country,state,city,pincode,area,houseNo)
- **Successful** Resp - SC 201 AddressResponseDto + mesg (ApiResponse)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

---

4. BookingController(@RequestMapping=/booking)

3.get upcomming bookings by host id

- **URL** - http://host:port/booking/upcomming_booking/{guestId}
- **Method** - Get
- **Successful** Resp - SC 201 BookedCarDto(brand, model, transmissionTypeEnum, seatingCapacity, kmDriven, pickupDateTime, dropOffDateTime)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

1. get past bookings by host id

- **URL** - http://host:port/booking/past_booking/{guestId}
- **Method** - Get
- **Successful** Resp - SC 201 BookedCarDto(brand, model, transmissionTypeEnum, seatingCapacity, kmDriven, pickupDateTime, dropOffDateTime)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

3. **admin APIs**

. AdminController(@RequestMapping=/admin)

---

1. GuestController (@RequestMapping=/guest)

3.get all guests

- **URL** - http://host:port/guest/get_all_guests
- **Method** - Get
- **Successful Resp** - SC 201 ArrayList
  - GetAllGuestDto (firstName,lastName,Mobile,email)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

---

1. HostController(@RequestMapping=/host)

---

2. CarController(@RequestMapping=/car)

3. Anootated Entities

```
package com.rentmycar.entity;

@Setter
@Getter
@ToString
@MappedSuperclass
public class BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(updatable = false)
    @CreationTimestamp
    private LocalDate createdOn;

    @UpdateTimestamp
    private LocalDateTime updatedOn;

}
```

```
package com.rentmycar.entity;

@Entity
@Getter
@Setter
@ToString
@Table(name = "user", uniqueConstraints = {
        @UniqueConstraint(columnNames = {"email","mobile","roleEnum"})})
public class User extends BaseEntity {

    @Column(length = 25, nullable = false)
    private String firstName;

    @Column(length = 25, nullable = false)
    private String lastName;

    @Column(length = 12, nullable = false)
    private String mobile;

    @Column(length = 20, nullable = false)
    private String email;

    @Column(length = 70, nullable = false)
    private String password;

    @Column(columnDefinition = "boolean default false", nullable = false)
```

```
    private Boolean isDeleted;

    @Enumerated(EnumType.STRING)
    @Column(length = 5)
    private UserRoleEnum roleEnum;

//
**********************************************************************************
*******

    // User 1<---->* Address
    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)
// mandatory , otherwise

// ,MappingException !
    private List<Address> addressList = new ArrayList<>();

    public void addAddress(Address address) {
        addressList.add(address);
        address.setUser(this);
    }

    public void deleteAddress(Address address) {
        addressList.remove(address);
        address.setUser(null);
    }

//
**********************************************************************************
*********

    // Guest 1<---->* Booking
    @OneToMany(mappedBy = "guest", cascade = CascadeType.ALL, orphanRemoval =
true)
    private List<Booking> bookingList = new ArrayList<>();

    public void addBooking(Booking booking) {
        bookingList.add(booking);
        booking.setGuest(this);
    }

//
**********************************************************************************
**********

    // Guest 1 <------> *Review
    @OneToMany(mappedBy = "guest", cascade = CascadeType.ALL, orphanRemoval =
true)
    private List<Review> reviewList = new ArrayList<Review>();

    public void addReview(Review review) {
        reviewList.add(review);
        review.setGuest(this);
    }
```

```
//**********************************************************************************
**************

    // User 1 <------> 1 DrivingLicense
    @OneToOne(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)
    private DrivingLicense drivingLicense;

    public void adddrivingLicense(DrivingLicense drivingLicense) {
        if (drivingLicense == null) {
            throw new RuntimeException("driving license is null");
        }
        this.drivingLicense = drivingLicense;
        drivingLicense.setUser(this);
    }

//
**********************************************************************************
**********

    // Host 1 <---------> * CarHostAddressPricing
    @OneToMany(mappedBy = "host", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<CarHostAddressPricing> carHostAddressPricingList = new
ArrayList<CarHostAddressPricing>();

    public void addCarHostAddressPricing(CarHostAddressPricing
carHostAddressPricing) {
        this.carHostAddressPricingList.add(carHostAddressPricing);
        carHostAddressPricing.setHost(this);
//**********************************************************************************
***************
    }
```

```
package com.rentmycar.entity;

public enum UserRoleEnum {
    GUEST,ADMIN,HOST
}
```

```
package com.rentmycar.entity;

@Entity
@Table
@Getter
@Setter
@ToString

public class Address extends BaseEntity {
```

```
    @Column(name = "adr_line1", length = 200, nullable = false)
    private String adrLine1; // varchar(100) not null

    @Column(name = "adr_line2", length = 200)
    private String adrLine2; // varchar(100)

    @Column(length = 20, nullable = false)
    private String country; // varchar(20),not null

    @Column(length = 20, nullable = false)
    private String state; // varchar(20),not null

    @Column(length = 20, nullable = false)
    private String city; // varchar(20),not null

    @Column(nullable = false, columnDefinition = "char(10)")
    private String pincode; // char(10),not null

    @Column(columnDefinition = "boolean default false", nullable = false)
    private Boolean isDeleted;

//*********************************************************************************
******************
    // many -> Address * <----> 1 User
    @ManyToOne(fetch = FetchType.LAZY) // mandatory
    @JoinColumn(name = "guest_id", nullable = false)
    private User user;

//*********************************************************************************
******************

}
```

```
package com.rentmycar.entity;

@Getter
@Setter
@ToString
@Entity
@Table
public class Booking extends BaseEntity {
    @Column(nullable = false)
    LocalDateTime pickUp; // DateTime,not null

    @Column(nullable = false)
    LocalDateTime dropOff; // DateTime,not null

    @Enumerated(EnumType.STRING)
    @Column(nullable = false, length = 10)
    @ColumnDefault("'PENDING'")
```

```
    BookingStatusEnum bookingStatusEnum; // not null,default - PENDING

    @Column(nullable = false)
    Double amount; // double,not null

    @Column(nullable = false)
    LocalDateTime paymentDateTime; // DateTime

    @Column(nullable = false, columnDefinition = "char(20)")
    String transactionId; // char (#### 20.

//*************************************************************************************
*********************

    // Booking * <---> 1 Guest
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "guest_id", nullable = false)
    private User guest;

//*************************************************************************************
*********************

    // Booking * -------> 1 Address
    @ManyToOne
    @JoinColumn(nullable = false)
    private Address guestAddress;

//*************************************************************************************
*********************

    // Booking * <-----------> 1 CarHostAddrssPricing
    @ManyToOne(fetch = FetchType.LAZY)
    private CarHostAddressPricing carHostAddressPricing;

//*************************************************************************************
*********************
}
```

```
package com.rentmycar.entity;

public enum BookingStatusEnum {
    CANCEL, PENDING, SUCCESS,FAILED
}
```

```
package com.rentmycar.entity;

@Table(name = "car")
@Entity
```

```
public class Car extends BaseEntity {

    @Column(length = 20, nullable = false)
    private String brand; // varchar(20),not null

    @Column(length = 20, nullable = false)
    private String model; // varchar(20),not null

    @Enumerated(EnumType.STRING)
    @Column(length = 10)
    private FuelTypeEnum fuelTypeEnum; // varchar(10), not null

    @Column(columnDefinition = "TINYINT", nullable = false )
    private int seatingCapacity; // tinyint(1-6),not null

//*************************************************************
    // Car * ------> 1 CarFeatures
    // many cars can have same features
    @ManyToOne
    private CarFeatures carFeatures;

//*************************************************************
}
```

```
package com.rentmycar.entity;

public enum FuelTypeEnum {
    PETRO,DIESEL,ELECTRIC,CNG
}
```

```
package com.rentmycar.entity;

@ToString
@Setter
@Getter
@Entity
@Table(name = "car_features")
public class CarFeatures extends BaseEntity{

    @Column(columnDefinition = "boolean default false")
    private Boolean hasUsbCharger;  // Boolean,default-false

    @Column(columnDefinition = "boolean default false")
    private Boolean hasBluetooth;   // Boolean,default-false

    @Column(columnDefinition = "boolean default false")
    private Boolean hasPowerSteering;   // Boolean,default-false
```

```
    @Column(columnDefinition = "boolean default false")
    private Boolean hasAirBags;     // Boolean,default-false

    @Column(columnDefinition = "boolean default false")
    private Boolean hasAbs;     // Boolean,default-false

    @Column(columnDefinition = "boolean default false")
    private Boolean hasAc;  // Boolean,default-false

}
```

```
package com.rentmycar.entity;

@Table
@Entity
@Getter
@Setter
public class CarHostAddressPricing extends BaseEntity {
    int kmDriven; // int,not null

    @Column(columnDefinition = "TINYINT")
    int fuelMeter; // tinyint(1-10),not null

    @Column(length = 20)
    String carImage; // varchar(20),

    @Column
    int noOfTrips;

    @Column(columnDefinition = "BOOLEAN DEFAULT TRUE")
    Boolean isAvailable; // Boolean,default-true

    @Column(columnDefinition = "BOOLEAN DEFAULT FALSE")
    Boolean isDeleted; // Booleane,default-false

    @Column(columnDefinition = "BOOLEAN DEFAULT FALSE")
    Boolean isApproved; // Boolean,default-false

    @Column(length = 20, nullable = false)
    String registrationNo; // varchar(20),not null

    @Column(columnDefinition = "TINYINT NOT NULL")
    int spareTyreCount; // tinyint,not null

//*************************************************************************

    // CarHostAddressPricing * ---------> 1 Car
    @ManyToOne
    @JoinColumn(nullable = false)
    private Car car;
```

```
//************************************************************************

    // CarHostAddressPricing * -------> 1 CarPricing
    @ManyToOne
    @JoinColumn(nullable = false)
    private CarPricing carPricing;;

//************************************************************************

    // CarHostAddressPricing * --------> 1 Address
    @ManyToOne
    @JoinColumn(nullable = false)
    private Address address;

//************************************************************************

    // CarHostAddressPricing * <------------> 1 Host
    @ManyToOne
    @JoinColumn(nullable = false)
    private User host;

//************************************************************************

    // CarHostAddressPricing 1 <-------------> * Booking
    @OneToMany(mappedBy = "carHostAddressPricing", cascade = CascadeType.ALL,
orphanRemoval = true)
    private List<Booking> bookingList = new ArrayList<Booking>();

    public void addBooking(Booking booking) {
        bookingList.add(booking);
        booking.setCarHostAddressPricing(this);
    }

//************************************************************************

}
```

```
package com.rentmycar.entity;

@ToString
@Getter
@Setter
@Entity
@Table(name = "car_pricing")
public class CarPricing extends BaseEntity{

    @Column(nullable = false)
    Double pricePerHr;  // Boolean,not null

    @Column(nullable = false)
    Double pricePerDay;     // Boolean,not null
```

```
    @Column(nullable = false)
    Double securityDeposit;     // double,not null
}
```

```
package com.rentmycar.entity;

@Getter
@Setter
@ToString
@Entity
@Table(name = "driving_license")
public class DrivingLicense extends BaseEntity {

    @Column(name = "license_no", length = 25, unique = true, nullable = false)
    private String drivingLicenseNo;

    @Column(name = "license_issue_date", nullable = false)
    private LocalDate issueDate;

    @Column(name = "license_expiry_date", nullable = false)
    private LocalDate expirationDate;

    @Enumerated(EnumType.STRING)
    @Column(length = 10, nullable = false)
    private LicenseClassEnum licenseClassEnum;

//**************************************************************************************
******************
    // DrivingLicense 1 <----> 1 User
    @OneToOne(fetch = FetchType.LAZY) // mandatory
    @JoinColumn(name = "user_id", nullable = false)
    private User user;

//**************************************************************************************
******************

}
```

```
package com.rentmycar.entity;

public enum LicenseClassEnum {

    A, // ("Light Motor Vehicle"),
    B, // ("Motorcycles up to 500cc"),
    C, // ("Heavy Motor Vehicle"),
    D// ("Transport Vehicle")
}
```

```java
package com.rentmycar.entity;

@Entity
@Table(name = "review")
@Data
public class Review extends BaseEntity{
    @Column(nullable = false)
    private byte rating; // tinyint,not-null

    @Column(length = 50)
    private String reviewText; // varchar(50)


// ******************************************************************************
    // Review * <-----------> 1 Guest
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "guest_id", nullable = false)
    private User guest;

// ******************************************************************************

    // Review * -----------> 1 CarHostAddressPricing
    @ManyToOne
    @JoinColumn(nullable = false)
    private CarHostAddressPricing carHostAddressPricing;

//******************************************************************************

}
```

```java
// YET TO BE DECIDED
@Entity
public class Transaction {
    Double ammount; // double
    PaymentStatusEnum paymentStatusEnum;
    DateTime paymentDate; // DateTime
}
```

4 . ERDiagram