**DFTF\_MAS\_M13035\_CAR\_SERVICE**

**V1.0**

**Proddhututri Ushasree**

**Manikanda Prabhu S**

**Kousik Raj S**

**Java Mode 2 Training – Final Hackathon**

Table of Contents

**USER STORIES4**

**USE CASE DIAGRAM6**

**DATABASE** **DESIGN7**

User7

Routes7

Booking8

**UI SCREENS9**

Home Page9

Admin Login10

Manage Routes11

Add Routes12

View Bookings13

User Login14

User Login Error-115

User Login Error -216

Register17

Available Routes18

User Booking19

My Bookings20

**REST API DESIGN21**

User Service21

Route Service22

Booking Service24

**CLASS DIAGRAM26**

User26

Routes27

Booking28

**SEQUENCE DIAGRAM29**

Routes-Create Route29

Edit Route30

Booking31

**TASK DESCRIPTION32**

# User Stories

**CS\_01: Welcome Page**: User can view the description about the Car Servicing. In this page user has two option one is register hyperlink and other is login hyperlink. If user click the register link it will redirect to the registration page. If user click the login link it will redirect to the login page. NFR: Performance - Response time should be less than a second. SonarQube - Maintainability, Reliability, Security should be in green.

**CS\_02: Register a User:** User will click on register link and provide the necessary details. All fields are mandatory in this page. After entering valid values and click of "Register" button show the "User registered successfully" msg. Error Messages: "You have entered invalid details" or against each field we can show the message. NFR: Performance - Response time should be less than a second. SonarQube - Maintainability, Reliability, Security should be in green.

**CS\_03: Login Page:** User will click on login link and provide the username and password. username and password is mandatory. After entering valid values and click of "Login" button show the "User Logged in successfully" msg. Error Messages: "You have entered invalid details" or against each field we can show the message. If username is not in the database it shows “username doesn’t exist”. If username and password are admin it will redirect to admin page. NFR: Performance - Response time should be less than a second. SonarQube - Maintainability, Reliability, Security should be in green.

**CS\_04: Home Page:** User can view requester link and Service link. If user want to book service, user will click the service hyperlink. If user want to check Request status, user will click the Request status hyperlink. NFR: Performance - Response time should be less than a second. Sonar Qube - Maintainability, Reliability, Security should be in green.

**CS\_05: Requester Page:** Requester can view all the requests that were placed and pending status.On clicking request Id user should be able to view the status of request and Service Details. NFR: Performance - Response time should be less than a second. Sonar Qube - Maintainability, Reliability, Security should be in green.

**CS\_06: Bulk Update:** Requester will click on Bulk requesters link should be able to view all the requests that were booked and pending status of service booked,the request details need to be updated through Approve/Reject.NFR: Performance - Response time should be less than a second. SonarQube - Maintainability, Reliability, Security should be in green.

DOD: Any given point of time, user should have the option to go for home page, by clicking home.

**CS\_07: Service Book Page:**User can check details of service centers details and availability and compare prices of servicing. Clicking on Book Service Link it will be redirected to web service page to enter mandatory fields. Error Messages: Should validate the datatype constraints for each field and update highlighted mandatory fields. NFR: Performance - Response time should be less than a second. SonarQube - Maintainability, Reliability, Security should be in green.

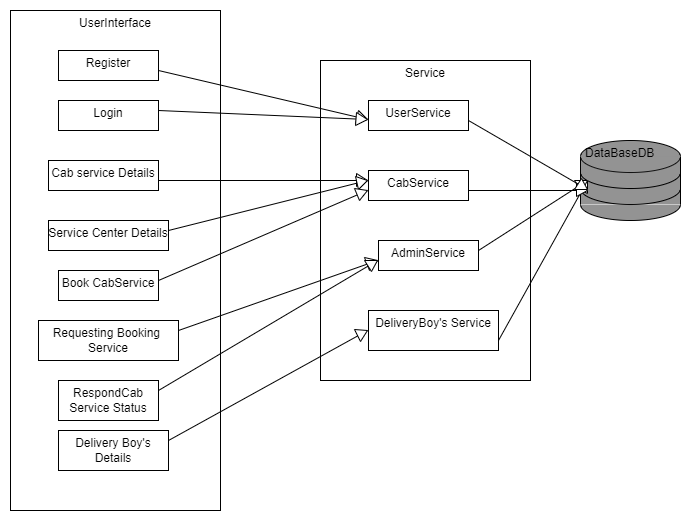
**CS\_08: Service Center for servicing:** After accepting generate service center details and delivery boy’s details and user can print the request. After entering valid values show the "Your request has been successfully raised" msg. Error Messages: "You have entered invalid details" or against each field we can show the message. NFR: Performance - Response time should be less than a second. SonarQube - Maintainability, Reliability, Security should be in green.

DOD: Any given point of time, user should have the option to go for home page, by clicking

**CS\_09: Admin Page:** Admin can view both Service list,Userlist and request status. In both table admin can update the status as approved/rejected. NFR: Performance - Response time should be less than a second. SonarQube - Maintainability, Reliability, Security should be in green.

DOD: Any given point of time, user should have the option to go for home page, by clicking home

USE CASE DIAGRAM



Database Design

User

Schema

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| USERID | Int | Primary key , Auto Increment |
| FIRST NAME | Varchar(15) | Not null |
| LAST NAME | Varchar(12) | Not null |
| AGE | int | Not null |
| PASSWORD | Varchar(30) | Not null |
| CONFIRM PASSWORD | Varchar(30) | Not null |
| MOBILE NUMBER | Varchar(30) | Not null , Unique |
| GENDER | Varchar(30) | Not null |
| DATE OF BIRTH | Varchar(20) | Not null |

### Sample Data

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| USERID | FIRST  NAME | LAST NAME | AGE | PASSWORD | CONFIRM PASSWORD | MOBILE  NUMBER | GENDER | DATE OF BIRTH |
| 1 | HARRY | H | harry  @gmail.com | harry@123 | harry@123 | 1234567890 | 25 | 5/1/2020 |

## **Cab Center**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| SERVICE ID | **int** | Primary key , Auto Increment |
| SERVICE TYPE | Varchar(30) | Not null |
| SERVICE STARTDATE | Varchar(30) | Not null |
| SERICE BOOKENDDATE | Varchar(30) | Not null |
| ESTIMATED COST | **int** | Not null |
| SERVICE STATUS | Varchar(30) | Not null |
| SERVICECENTER ID | int | Not null |
| DELIVERYBOY ID | int | Not null |

Sample

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SERVICE ID | SERVICE TYPE | SERVICE STARTDATE | SERICE BOOKENDDATE | ESTIMATED COST | SERVICE STATUS | SERVICE  CENTER ID | DELIVERY  BOY ID |
| 1 | Oil filte**r** | **12-07-2021** | 11-7-1999 | 1000 | Approve | 1 | 1 |

## **Cab Center**

|  |  |  |
| --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **CONSTRAINT** |
| CENTER ID | Int | Primary key, Auto increment |
| CENTER NAME | Varchar(30) | Not null |
| CENTER ADDRESS | Varchar(30) | Not null |
| CENTER AVAILABILITY | Varchar(20) | Not null |

### Sample

|  |  |  |  |
| --- | --- | --- | --- |
| CENTER ID | CENTER NAME | CENTER ADDRESS | CENTER AVAILABILITY |
| 1 | BALAJI | CHENNAI | YES |

## **Vehicle**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| VEHICLE ID | Int | Primary key |
| VEHICLE TYPE | Varchar(20) | Not null |
| VEHICLE NAME | Varchar(20) | Not null |
| VEHICLE COLOR | Varchar(20) | Not null |
| USERID | Int | Not null |

Sample

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **VEHICLEID** | **VEHICLE TYPE** | **VEHICLE NAME** | **VEHICLE COLOR** | **USERID** |
| 1 | PETROL | HYUNDAI | BLUE | 1 |

UI SCREENS

1.HOMEPAGE

**WELCOME TO CAR SERVICE!**

CAR SERVICE BOOKING

LOGIN

HOME

CAR SERVICE BOOKING

Java Mode 2 Training – Final Case Study

ADMIN LOGIN

Server URL: <http://localhost:8200/login>

Server Request: {”adminname”:”admin”, “password" :"admin"}

Server Response: {“msg”:“success”} – http status -ok

ADMINNAME:

PASSWORD:

CAB SERVICE APPLICATION

CCAR SERVIEC APPLICATION

HOME

WELCOME TO ADMIN

LOGIN

Java Mode 2 Training – Final Case Study

MANAGE ROUTES

Server URL: <http://localhost:8200/Routes>

Server Request: GET <http://localhost:8200/Routes>

Serve Response:[{ “serviceId”:”1”, “source”: “Chennai”, “destination”: “Vellore”,“totalKilometers”:”300km”, “cost”:”2000.00”, “cabsAvailable”:”3”, “cabType”:”Regular” } ] – http status - ok

CAR SERVICE

MANAGE STATUS

PENDING STATUS

USERLIST

SERVICE LIST

HOME

LOGIN

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SERVICE ID | SERVICE TYPE | SERVICE STARTDATE | SERVICE END  DATE | BOOKEDDATE | ESTIMATED COST | SERVICE STATUS | SERVICE  CENTER ID | DELIVERYBOY ID |
|  |  |  |  |  |  |  |  |  |

Java Mode 2 Training – Final Case Study

ADD SERVICE DETAILS

Server URL: <http://localhost:8200/Routes>

Server Request: GET <http://localhost:8200/Routes>

Serve Response:[{ “cabId”:”1”, “source”: “Chennai”, “destination”: “Vellore”, “totalKilometers”:”300km”, “cost”:”2000.00”, “cabsAvailable”:”3”, “cabType”:”Regular” } ] – http status – ok

SERVICE ID:

SERVICE TYPE

SERVICE STARTDATE

SERVICE END DATE

BOOKEDDATE

ESTIMATED COST

SERVICE STATUS

SERVICECENTER ID

DELIVERYBOY ID

**CAR SERVICE**

LOGIN

ADD ROUTES

SUBMIT

Java Mode 2 Training – Final Case Study

VIEW BOOKING

After successful SUBMIT below is the screen to be shown

Server URL: http://localhost:8200/Booking

Server Request: GET <http://localhost:8200/Booking>

Server Response: [{ “bookingId”:”1”, “source”: “Chennai”, “destination”: “Vellore”, “date”:”1-11-2021”, “time”:”5.00 PM”, “totalKilometers”:”300km”, “cost”:”2000.00”, “cabsAvailable”:”3”, “cabType”:”Regular” } ] – http status - ok

**CAR SERVICE**

LOGOUT

APPROVED STATUS

PENDING STATUS

USER LIST

SERVICE LIST

HOME

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SERVICE ID | SERVICE TYPE | SERVICE STARTDATE | SERVICE END DATE | BOOKED  DATE | ESTIMATED COST | SERVICE STATUS | SERVICE  CENTER ID | DELIVERY  BOY ID |
| 1 | OIL FILTER | 11-11-2020 | 12-11-2020 | 10-11-2021 | 1000 | yes | 1 | 1 |

Java Mode 2 Training – Final Case Study

AFTER THIS ADMIN WILL LOGOUT AND THEN USER WILL LOGIN

USER LOGIN

INITIAL

Server URL: <http://localhost:8200/login>

Server Request: {”username”:”user”, “password" :"user"}

Server Response: {“msg”:“success”} – http status -ok

CAB SERVICE APPLICATION

Login

<Username>

<password>

Login

Reset

Java Mode 2 Training – Final Case Study

New User? Register

LOGIN

HOME

USER LOGIN ERROR-1

CAB SERVICE APPLICATION

Login

<Username>

<password>

Login

Reset

Java Mode 2 Training – Final Case Study

New User? SignUp

LOGIN

HOME

WELCOME USER!!

Username not provided

Password not provided

USER LOGIN ERROR-2

Server URL: <http://localhost:8200/login>

Server Request: {”username”:”user”, “password" :"user"}

Server Response: {“msg”:“ “User Account does not exits”} – http status -not ok

CAB SERVICE APPLICATION

Login

<Username>

<password>

Login

Reset

Java Mode 2 Training – Final Case Study

New User? SignUp

User Account does not exists

LOGIN

HOME

REGISTER USER

USER REGISTRATION

If there is a new user

Server URL: <http://localhost:8200/User>

Server Request: :[{ “firstname”:” keerthu”, “lastname”: “rajkumar”, “age”: “20”, “gender”:”F”, “contactnumber”:”9003013959”, “date of join”:”1/2/2021” } ] – http status – ok

Server Response: : {“msg”:“ “successfully registered”} – http status - ok

**CAR SERVICE**

FIRSTNAME: LASTNAME:

USERID: PASSWORD:

DATE OF BIRTH AGE:

GENDER: CONTACT NUMBER:

FEMALE MALE OTHER

CAB BOOKING APPLICATION

LOGIN

HOME

REGISTER

SUBMIT

Java Mode 2 Training – Final Case Study

AVAILABLE ROUTES

Server URL: http://localhost:8200/Routes

Server Request: GET http://localhost:8200/Routes

Server Response: [{“Source”:“ “ , “Destination”: “ ” },] – http status - ok

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **USER ID** | FIRSTNAME | LASTNAME: | USERID | PASSWORD | DATE OF BIRTH | AGE | GENDER | CONTACT  NUMBER |
| 1 |  |  |  |  |  |  |  |  |

CAR SERVICE APPLICATION

LOGOUT

SERVICE LIST

USER LIST

HOME

AVAILABLE ROUTES

SUBMIT

Java Mode 2 Training – Final Case Study

Java Mode 2 Training – Final Case Study

USER BOOKING

Server URL: http://localhost:8200/Booking

Server Request: GET http://localhost:8200/Booking

Server Response:[{ “bookingId”:”1”, “source”: “Chennai”, “destination”: “Vellore”, “date”:”1-11-2021”, “time”:”5.00 PM”, “totalKilometers”:”300km”, “cost”:”2000.00”, “cabsAvailable”:”3”, “cabType”:”Regular” } ] – http status - ok

CAB SERVICE APPLICATION

LOGOUT

MY BOOKINGS

AVAILABLE ROUTES

HOME

BOOKING

CENTERID:

CENTERNAME:

CENTERADDRESS:

CENTERAVAILABILITY: (non-editable)

SERVICEID:

SUBMIT

Java Mode 2 Training – Final Case Study

MY BOOKING

Server URL: http://localhost:8200/Booking

Server Request: GET http://localhost:8200/Booking

Server Response:[{ “bookingId”:”001”, “source”: “Chennai”, “destination”: “Vellore”, “date”:”1-11-2021”, “time”:”5.00 PM”, “totalKilometers”:”300km”, “cost”:”2000.00”, “cabsAvailable”:”3”, “cabType”:”Regular” } ] – http status – ok

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CENTERID: | CENTERNAME: | CENTERADDRESS: | CENTER  AVAILABILITY: | SERVICEID: |
| 1 | BALAJI | Vellore | MON-FRI | 1 |

CAB SERVICE APPLICATION

AVAILABLE ROUTES

HOME

LOGOUT

MY BOOKING

SUBMIT

Java Mode 2 Training – Final Case Study

## **REST API design: User Service**

### Entity objects (Models – tables in database)

**UserService**

**public** **class** CarService {

@Id

**private** **int** ServiceId;

**private** String VehicleName;

**private** String ServiceName;

**private** String ServiceStartDate;

**private** String ServiceEndDate;

**private** String BookedDate;

**private** **float** EstimatedCost;

**private** **boolean** Status;

**private** String EstimateDeliveryDate;

**private** **int** UserId;

**private** **int** DeliveryBoyId;

}

### Controllers

@RestController

@RequestMapping("/api/v1/")

**public** **class** CarServiceController {

@Autowired

CabServiceService carService;

@Autowired

CabServiceRepository cabRepo;

//get all CarService

@GetMapping("cabservices")

**public** List<CarService> getAllCabService(){

**return** carService.getfindAllCabService();

}

//create CarService

@PostMapping("cabservices")

**public** ResponseEntity<CarService> create(@RequestBody CarService cabservice){

**return** **new** ResponseEntity<CarService>( carService.saveCabService(cabservice),HttpStatus.***CREATED***);

}

//get CarService By ServiceId

@GetMapping("cabservices/{ServiceId}")

**public** ResponseEntity<CarService> cabservice(@PathVariable **int** ServiceId){

**return** **new** ResponseEntity<CarService>(carService.findById(ServiceId),HttpStatus.***OK***) ;

}

//update CarService rest

@PutMapping("cabservices/{ServiceId}")

**public** ResponseEntity<CarService> update(@PathVariable **int** ServiceId,@RequestBody CarService cabservice);

**return** **new** ResponseEntity<CarService>(updateCab,HttpStatus.***OK***);

}

//delete CarService Rest

@DeleteMapping("cabservices/{ServiceId}")

**public** ResponseEntity<Boolean> delete(@PathVariable **int** ServiceId){

**return** **new** ResponseEntity<Boolean>(deleteCab,HttpStatus.***OK***);

}

}

Services

@Service

public class CabServiceServiceImpl implements CabServiceService {

@Autowired

CabServiceRepository cabRepo;

@Override

public List<CarService> getfindAllCabService() {

return cabRepo.findAll();

}

@Override

public CarService saveCabService(CarService cabservice) {

return cabRepo.save(cabservice);

}

@Override

public CarService findById(int ServiceId) {

return cabRepo.findById(ServiceId).get();

}

@Override

public CarService save(CarService cabservice) {

return cabRepo.save(cabservice);

}

@Override

public boolean delete(int ServiceId) {

cabRepo.deleteById(ServiceId);

return true;

}

}

}

### Respository

@Repository

**public** **interface** CabServiceRepository **extends** JpaRepository<CarService, Integer> {

}

## **REST API design: CABCENTERService**

### Entity objects (Models – tables in database)

MODEL

**public** **class** CabCenter {

@Id

**private** **int** centerId;

**private** String centerName;

**private** String centerAddress;

**private** String centerAvailability;

}

Controller

@RestController

@RequestMapping("/api/cc/")

**public** **class** CabCenterController {

@Autowired

CarCenterService carService;

@Autowired

CabCenterRepository carRepo;

//get all CabCenter

@GetMapping("carcenters")

**public** List<CabCenter> getAllCarCenter(){

**return** carService.getfindAllCarCenter();

}

//create CabCenter

@PostMapping("carcenter")

**public** ResponseEntity<CabCenter> create(@RequestBody CabCenter carcenter){

**return** **new** ResponseEntity<CabCenter>( carService.saveCarCenter(carcenter),HttpStatus.***CREATED***);

}

//get CabCenter By ServiceId

@GetMapping("carcenters/{centerId}")

**public** ResponseEntity<CabCenter> carcenter(@PathVariable **int** centerId){

**return** **new** ResponseEntity<CabCenter>(carService.findById(centerId),HttpStatus.***OK***) ;

}

//update CabCenter rest

@PutMapping("carcenters/{centerId}")

**public** ResponseEntity<CabCenter> update(@PathVariable **int** centerId,@RequestBody CabCenter carcenter){

**return** **new** ResponseEntity<CabCenter>(updateCar,HttpStatus.***OK***);

}

//delete CabCenter Rest

@DeleteMapping("carcenters/{centerId}")

**public** ResponseEntity<Boolean> delete(@PathVariable **int** centerId){

**return** **new** ResponseEntity<Boolean>(deleteCar,HttpStatus.***OK***);

}

}

Service

@Service

**public** **class** CarCenterServiceImpl **implements** CarCenterService {

@Autowired

CabCenterRepository carRepo;

@Override

**public** List<CabCenter> getfindAllCarCenter() {

**return** carRepo.findAll();

}

@Override

**public** CabCenter saveCarCenter(CabCenter carcenter) {

**return** carRepo.save(carcenter);

}

@Override

**public** CabCenter findById(**int** centerId) {

**return** carRepo.findById(centerId).get();

}

@Override

**public** CabCenter save(CabCenter carcenter) {

**return** carRepo.save(carcenter);

}

@Override

**public** **boolean** delete(**int** centerId) {

carRepo.deleteById(centerId);

**return** **true**;

}

}

Repository

@Repository

**public** **interface** CabCenterRepository **extends** JpaRepository<CabCenter, Integer> {

}

## **REST API design: DELIVERY’S BOY ID**

### Entity objects (Models – tables in database)

**public** **class** DeliveryBoyDetails {

@Id

**private** **int** deliveryBoyId;

**private** String deliveryBoyName;

**private** String deliveryBoyNumber;

**public** **int** getDeliveryBoyId() {

**return** deliveryBoyId;

}

Controller

 @RestController

@RequestMapping("/api/dboy/")

public class DeliveryBoyController {

@Autowired

DeliveryBoyService dService;

@Autowired

DeliveryBoyRepository dboyRepo;

//get all DeliveryBoyDetails

@GetMapping("dservices")

public List<DeliveryBoyDetails> getAllDeliveryBoyDetails(){

return dService.getfindAllDeliveryBoyDetails();

}

//create DeliveryBoyDetails

@PostMapping("dservice")

public ResponseEntity<DeliveryBoyDetails> create(@RequestBody DeliveryBoyDetails deliveryboy){

return new ResponseEntity<DeliveryBoyDetails>( dService.saveDeliveryBoyDetails(deliveryboy),HttpStatus.CREATED);

}

//get DeliveryBoyDetails By deliveryBoyId

@GetMapping("dservices/{deliveryBoyId}")

public ResponseEntity<DeliveryBoyDetails> dservice(@PathVariable int deliveryBoyId){

return new ResponseEntity<DeliveryBoyDetails>(dService.findById(deliveryBoyId),HttpStatus.OK) ;

}

//update DeliveryBoyDetails rest

@PutMapping("dservices/{deliveryBoyId}")

public ResponseEntity<DeliveryBoyDetails> update(@PathVariable int deliveryBoyId,@RequestBody DeliveryBoyDetails deliveryboy){

DeliveryBoyDetails dboy=dService.findById(deliveryBoyId)

return new ResponseEntity<DeliveryBoyDetails>(updateDboy,HttpStatus.OK);

}

//delete DeliveryBoyDetails Rest

@DeleteMapping("dservices/{deliveryBoyId}")

public ResponseEntity<Boolean> delete(@PathVariable int deliveryBoyId){

return new ResponseEntity<Boolean>(deleteDboy,HttpStatus.OK);

}

}

Service

**public** **class** DeliveryBoyServiceImpl **implements** DeliveryBoyService {

@Autowired

DeliveryBoyRepository dboyRepo;

@Override

**public** List<DeliveryBoyDetails> getfindAllDeliveryBoyDetails() {

**return** dboyRepo.findAll();

}

@Override

**public** DeliveryBoyDetails saveDeliveryBoyDetails(DeliveryBoyDetails deliveryboy) {

**return** dboyRepo.save(deliveryboy);

}

@Override

**public** DeliveryBoyDetails findById(**int** deliveryBoyId) {

**return** dboyRepo.findById(deliveryBoyId).get();

}

@Override

**public** DeliveryBoyDetails save(DeliveryBoyDetails deliveryboy) {

**return** dboyRepo.save(deliveryboy);

}

@Override

**public** **boolean** delete(**int** deliveryBoyId) {

dboyRepo.deleteById(deliveryBoyId);

**return** **true**;

}

Repository

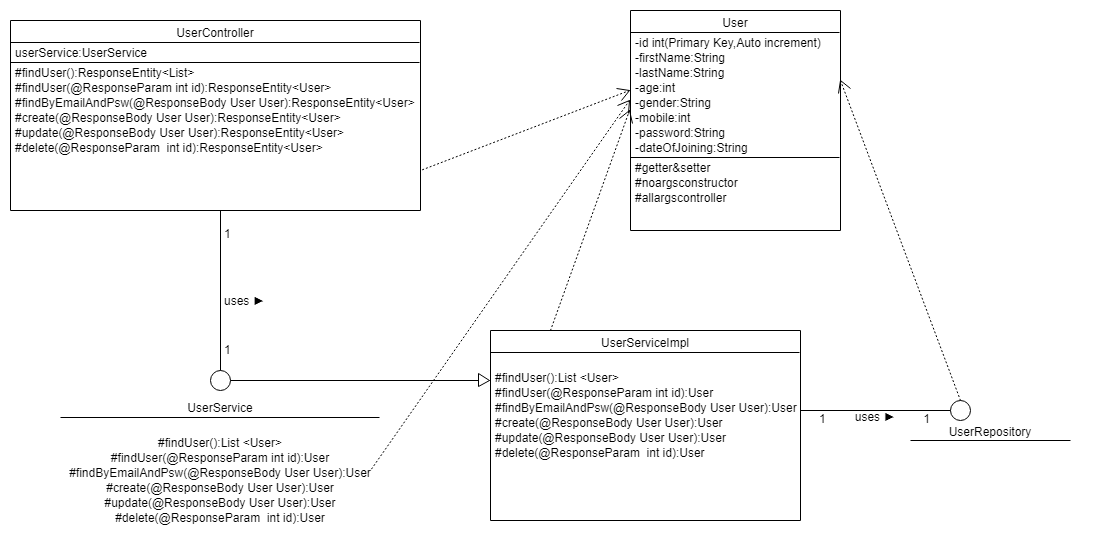
@Repository

**public** **interface** DeliveryBoyRepository **extends** JpaRepository<DeliveryBoyDetails, Integer> {

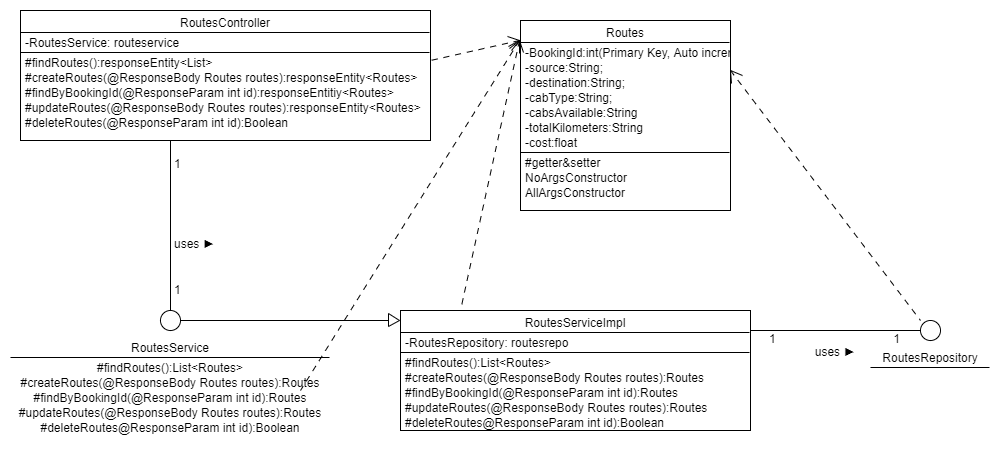
}

CLASS DIAGRAM

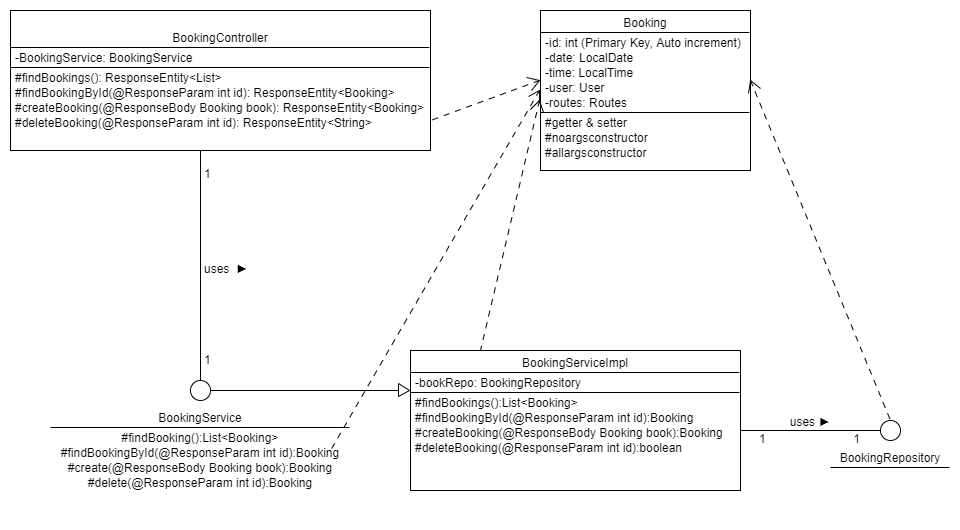
USER



ROUTES

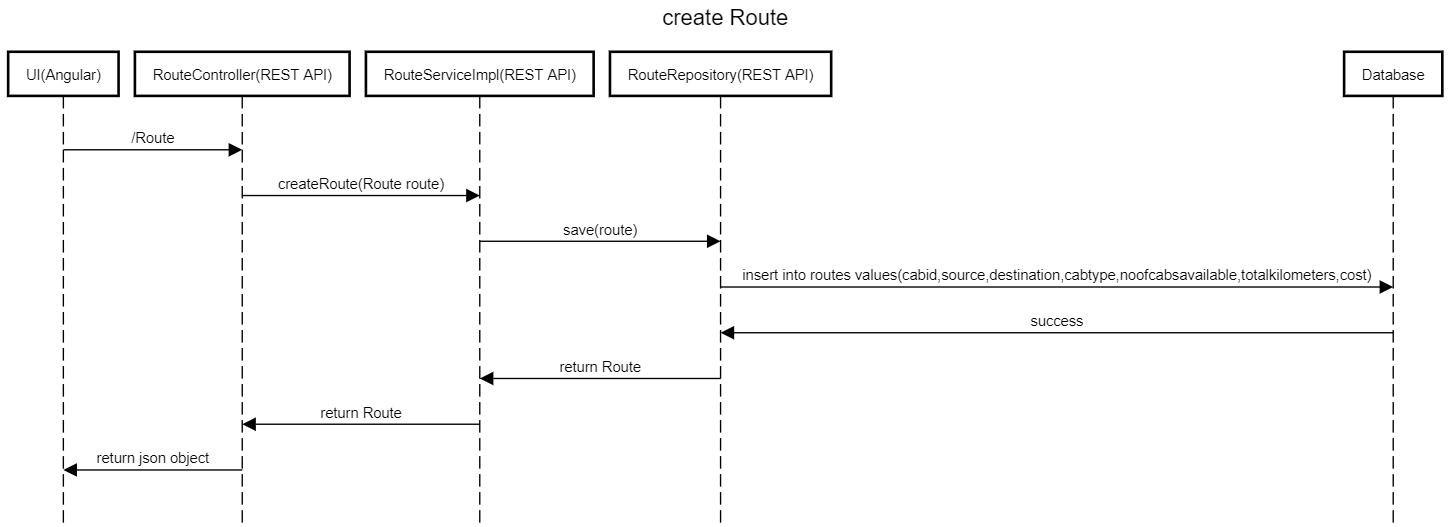


BOOKING

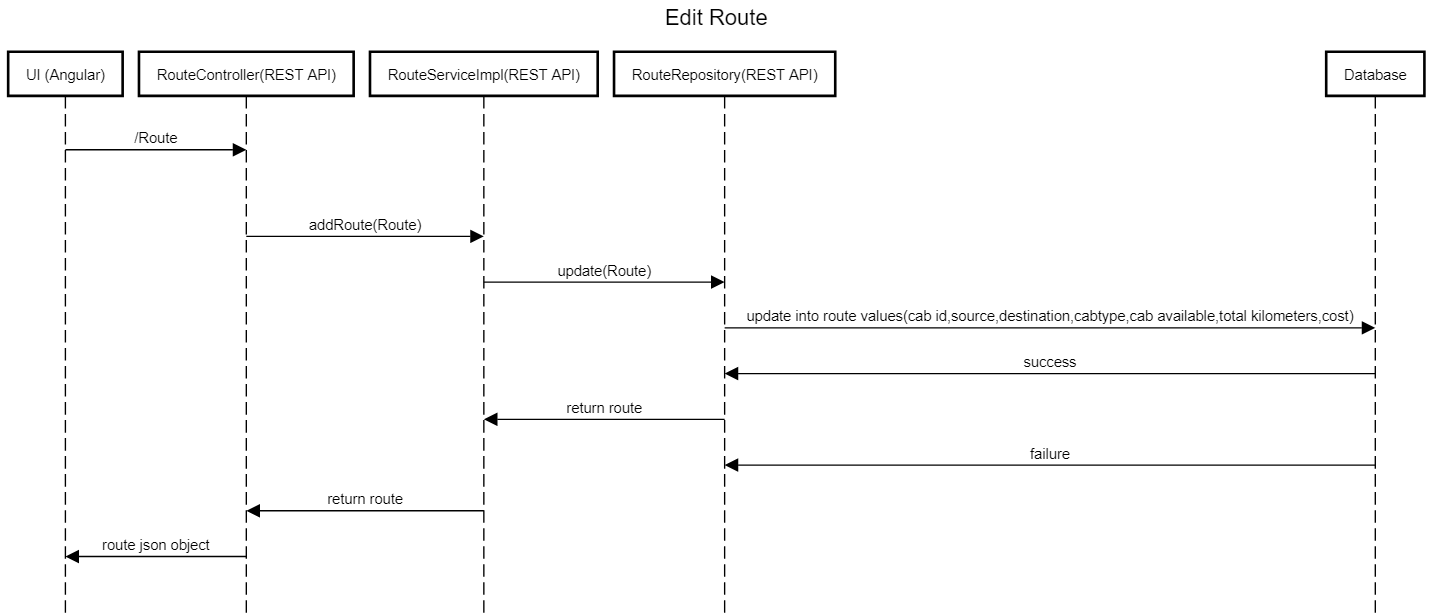


SEQUENCE DIAGRAM

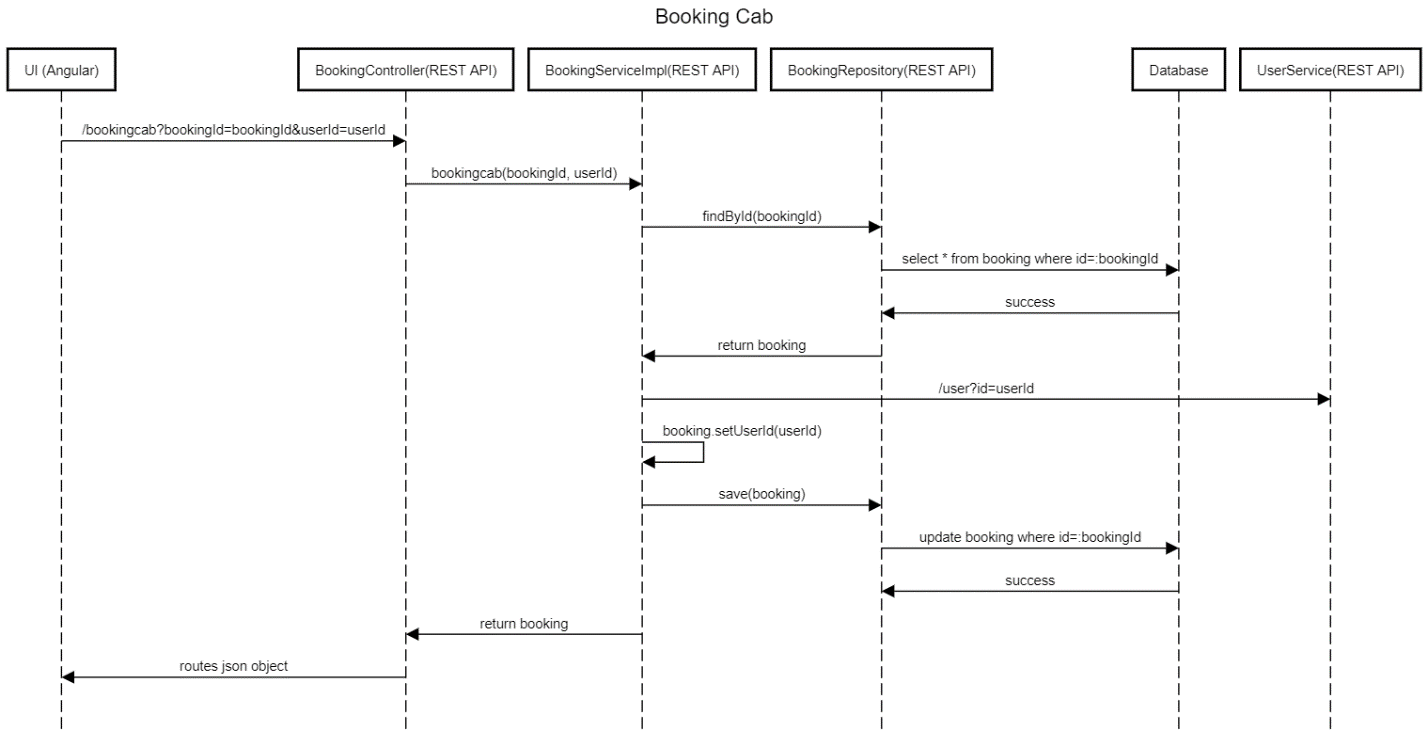
ROUTE-CREATE ROUTE



EDIT ROUTE



BOOKING



|  |  |
| --- | --- |
| UI/SERVICE | DEVELOPERS |
| UI-ANGULAR | PRASANNA RANI.S,NEELA.K,KEERTHANA.R |
| USER SERVICE | KEERTHANA.R |
| ROUTES SERVICE | PRASANNA RANI.S |
| BOOKING SERVICE | NEELA.K |

TASK DISTRIBUTION

THANKYOU