Milestone Assessment 3

Create a Springboot based RESTAPI application for the below scenarios.

A company wants to make survey on various mobile handset of various Brands. Brands name like Samsung, Apple, Redmi, OnePlus etc. Each brand has multiple mobiles. Each mobile has attributes like id, name, price. Each mobile must belongs to a particular brand. A mobile can't belongs to more than one brand.

Based on the above scenarios write a Springboot REST api based application for the following functionalities.

1. Create Brand . (Brand has attribute like id, name).

2. Create Mobile and assign it to a Brand.

3. Fetch all Brands and their corresponding mobiles based on Brand name in ascending order.

4. Update a particular mobile's price by supplying the Brand name, mobile name and price as path variable. After successfully update show the updated value in the postman.

5. Write an custom exception for usecase 4 or question 4 , if the brand name is not present then application should throw an custom exception message " No such Brand there in database" .

Follow proper industry standard folder structure like (Controller, entity, service, repository, exception folder)

Skill: Springboot RestAPI

JPA maven Postman

Code:

/src/main/java/com/mindtree/milestone3/Milestone3Application.java

```
package com.mindtree.milestone3;

import java.util.ArrayList;
import java.util.List;

import com.mindtree.milestone3.model.Brand;
import com.mindtree.milestone3.model.Mobile;
import com.mindtree.milestone3.service.BrandService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```java
@SpringBootApplication
public class Milestone3Application implements CommandLineRunner{

    @Autowired
    BrandService service;

    public static void main(String[] args) {
        SpringApplication.run(Milestone3Application.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        Brand brand = new Brand("Samsung", 101);
        Mobile Mobile1 = new Mobile(1, "J7 DUO", 15000.00);
        Mobile Mobile2 = new Mobile(2, "Galaxy A32", 14000.00);
        Mobile Mobile3 =new Mobile(3, "Galaxy Monster33", 20000.00);
        List<Mobile> Mobiles = new ArrayList<>();
        Mobiles.add(Mobile1);
        Mobiles.add(Mobile2);
        Mobiles.add(Mobile3);
        brand.setMobiles(Mobiles);
        service.createBrand(brand);

    }

}
```

/src/main/java/com/mindtree/milestone3/controller

```java
package com.mindtree.milestone3.controller;

import java.util.Arrays;
import java.util.List;

import com.mindtree.milestone3.model.Brand;
import com.mindtree.milestone3.model.Mobile;
import com.mindtree.milestone3.service.BrandService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class BrandController {
```

```
    @Autowired
    BrandService service;

    // fetch all brand and corresponding mobiles based on brand name in ascending order
    @GetMapping("/brands")
    public ResponseEntity<List<Brand>> getBrands() {
        return new ResponseEntity<List<Brand>>(service.getAllBrands(), HttpStatus.OK);
    }

    //get brand
    @GetMapping("/brands/{name}")
    public ResponseEntity<Brand> getBrand(@PathVariable String name) {
        return new ResponseEntity<Brand>(service.getBrand(name), HttpStatus.FOUND);
    }

    // create brand
    @PostMapping("/brands")
    public ResponseEntity<Brand> createBrand(@RequestBody Brand brand) {
        return new ResponseEntity<Brand>(service.createBrand(brand), HttpStatus.CREATED);
    }

    // get all mobiles
    @GetMapping("/brands/{name}/mobiles")
    public ResponseEntity<List<Mobile>> getMobile(@PathVariable String name) {
        return new ResponseEntity<List<Mobile>>(service.retrieveMobile(name),
HttpStatus.FOUND);
    }

    // Create mobile and assign to a brand
    @PostMapping("/brands/{name}/mobiles")
    public ResponseEntity<Brand> createMobile(@PathVariable String name , @RequestBody
Mobile[] mobiles) {
        List<Mobile> mobile = Arrays.asList(mobiles);
        return new ResponseEntity<Brand>(service.createMobile(name, mobile),
HttpStatus.CREATED);
    }

    // update mobile price
    @PutMapping("/brands/{bname}/mobiles/{mname}/{cost}")
    public ResponseEntity<Brand> updateRoomCost(@PathVariable String bname, @PathVariable
String mname,
            @PathVariable double cost) {
        return new ResponseEntity<Brand>(service.updateMobileCost(bname, mname, cost),
HttpStatus.OK);
    }

}
```

/src/main/java/com/mindtree/milestone3/model

Brand.java

```
package com.mindtree.milestone3.model;
```

```java
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
//import javax.annotation.Generated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;

@Entity
public class Brand {
    @Id
    private String name;
    private int id;



    @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true, targetEntity =
Mobile.class)
    @JoinColumn(name = "brand_id", referencedColumnName = "name")
    private List<Mobile> mobiles;

    public Brand(String name,int id) {
        this.name = name;
        this.id=id;
    }

    public Brand() {

    }

    public int getId() {
        return id;
    }
    public void setId(int id){
        this.id=id;
    }

    public String getName() {
        return name;
    }


    public void setName(String name) {
        this.name = name;
    }

    public List<Mobile> getMobiles() {
        return mobiles;
    }

    public void setMobiles(List<Mobile> mobiles) {
```

```
        this.mobiles = mobiles;
    }

    @Override
    public String toString() {
        return "Brand [id=" + id + ", name=" + name + "]";
    }
}
```

Mobile.java

```
package com.mindtree.milestone3.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Mobile {
    @Id
    private String name;
    private int id;
    private double price;


    public Mobile() {
    }

    public Mobile(int id,String name, double price) {
        this.name = name;
        this.id = id;
        this.price = price;
    }



    public void setId(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }


    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

```
    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

}
```

/src/main/java/com/mindtree/milestone3/service

```
package com.mindtree.milestone3.service;

import java.util.List;
import java.util.Optional;

import com.mindtree.milestone3.exception.BrandNameNotFoundException;
import com.mindtree.milestone3.model.Brand;
import com.mindtree.milestone3.model.Mobile;
import com.mindtree.milestone3.repository.BrandRepo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;

@Service
public class BrandService {

    @Autowired
    BrandRepo repo;

    public List<Brand> getAllBrands() {
        Sort sort = Sort.by(Sort.Direction.ASC, "name");
        return repo.findAll(sort);
    }

    // get brand name with mobile
    public Brand getBrand(String name) {
        Optional<Brand> brand = repo.findById(name);
        if (!brand.isPresent()) {
            throw new BrandNameNotFoundException("No such Brand there in database");
        }
        Brand brand2 = brand.get();
        return brand2;

    }

    // create brand
    public Brand createBrand(Brand brand) {
        Brand brand1 = repo.save(brand);
        return brand1;
```

```
    }

    // create mobile
    public Brand createMobile(String name, List<Mobile> mobile) {
        Optional<Brand> Brand = repo.findById(name);
        if (!Brand.isPresent()) {
            throw new BrandNameNotFoundException("No such Brand there in database");
        }
        Brand Brand2 = Brand.get();

        Brand2.getMobiles().addAll(mobile);
        Brand Brand1 = repo.save(Brand2);
        return Brand1;
    }

    // get all mobile datas
    public List<Mobile> retrieveMobile(String name) {
        Optional<Brand> brand = repo.findById(name);
        if (!brand.isPresent()) {
            throw new BrandNameNotFoundException("No such Brand there in database");
        }
        Brand brand2 = brand.get();
        return brand2.getMobiles();
    }

    // update cost
    public Brand updateMobileCost(String bname, String mname, double cost) {
        Optional<Brand> brand = repo.findById(bname);
        if (!brand.isPresent()) {
            throw new BrandNameNotFoundException("No such Brand there in database");
        }
        Brand brand2 = brand.get();
        List<Mobile> mobile = brand2.getMobiles();
        for (Mobile mobile2 : mobile) {
            if (mobile2.getName() == mname) {
                mobile2.setPrice(cost);
            }
        }
        repo.save(brand2);
        return brand2;
    }
}
```

/src/main/java/com/mindtree/milestone3/repository

```
package com.mindtree.milestone3.repository;

import com.mindtree.milestone3.model.Brand;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
```

```java
@Repository
public interface BrandRepo extends JpaRepository<Brand, String>  {

}
```

/src/main/java/com/mindtree/milestone3/exception

BrandNameNotFoundException.java

```java
package com.mindtree.milestone3.exception;

public class BrandNameNotFoundException extends RuntimeException {

    public BrandNameNotFoundException(String message) {
        super(message);
    }

}
```

GlobalException.java

```java
package com.mindtree.milestone3.exception;

import java.util.Date;
import java.util.LinkedHashMap;
import java.util.Map;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import
org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

@ControllerAdvice
public class GlobalException extends ResponseEntityExceptionHandler {
    @ExceptionHandler(value = BrandNameNotFoundException.class)
    public ResponseEntity<Object> HotelIdNotFoundException(BrandNameNotFoundException
exception) {
        Map<String, Object> body = new LinkedHashMap<>();
        body.put("timestamp", new Date());
        body.put("status", HttpStatus.NOT_FOUND.value());
        body.put("message", exception.getMessage());
        return new ResponseEntity<Object>(body, HttpStatus.NOT_FOUND);
    }

}
```

/src/main/resources/application.properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57InnoDBDialect
spring.datasource.url=jdbc:mysql://localhost:3306/milestone3
spring.datasource.username=root
spring.datasource.password=Sourav100
logging.level.org.springframework.web: DEBUG
```

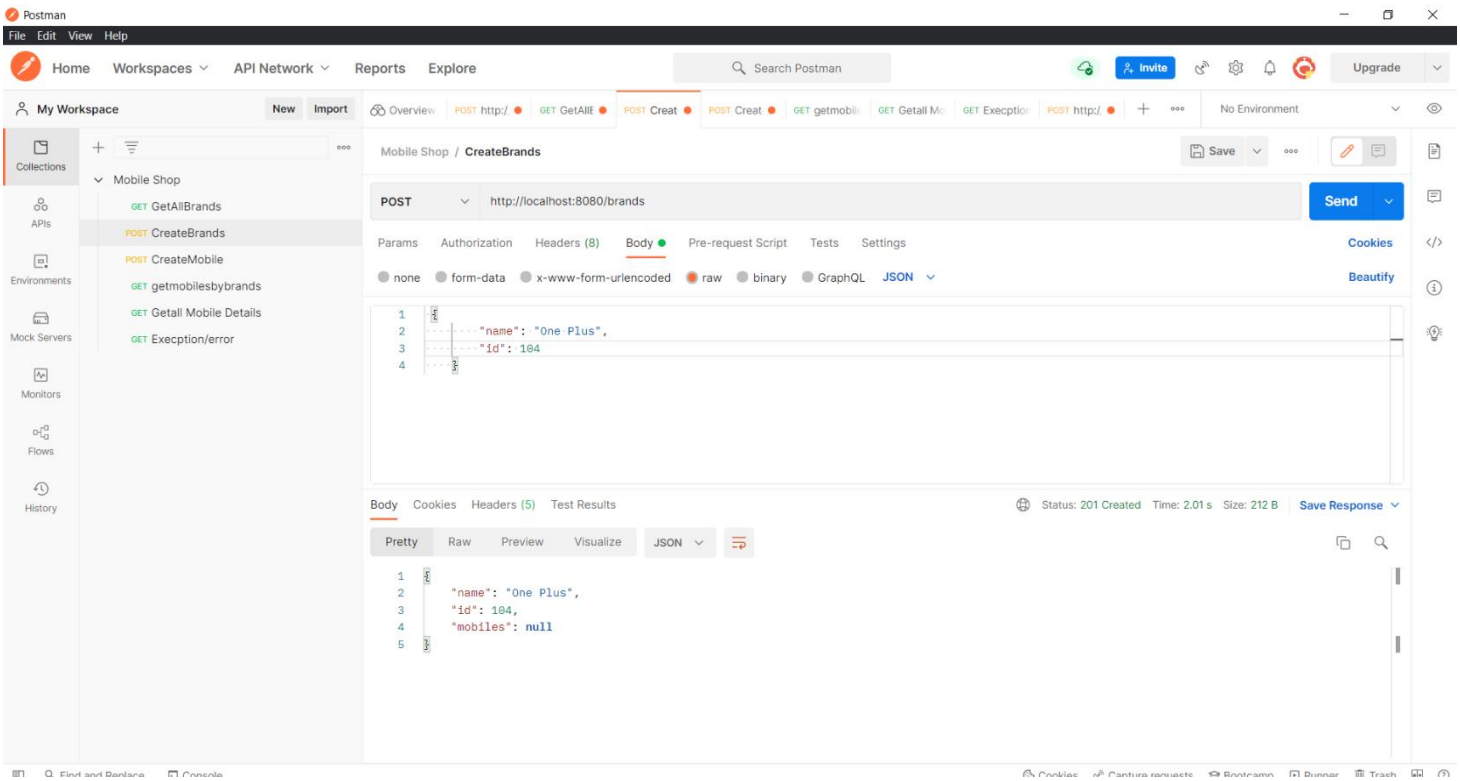
/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.7</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.mindtree</groupId>
    <artifactId>milestone3</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>milestone3</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
```

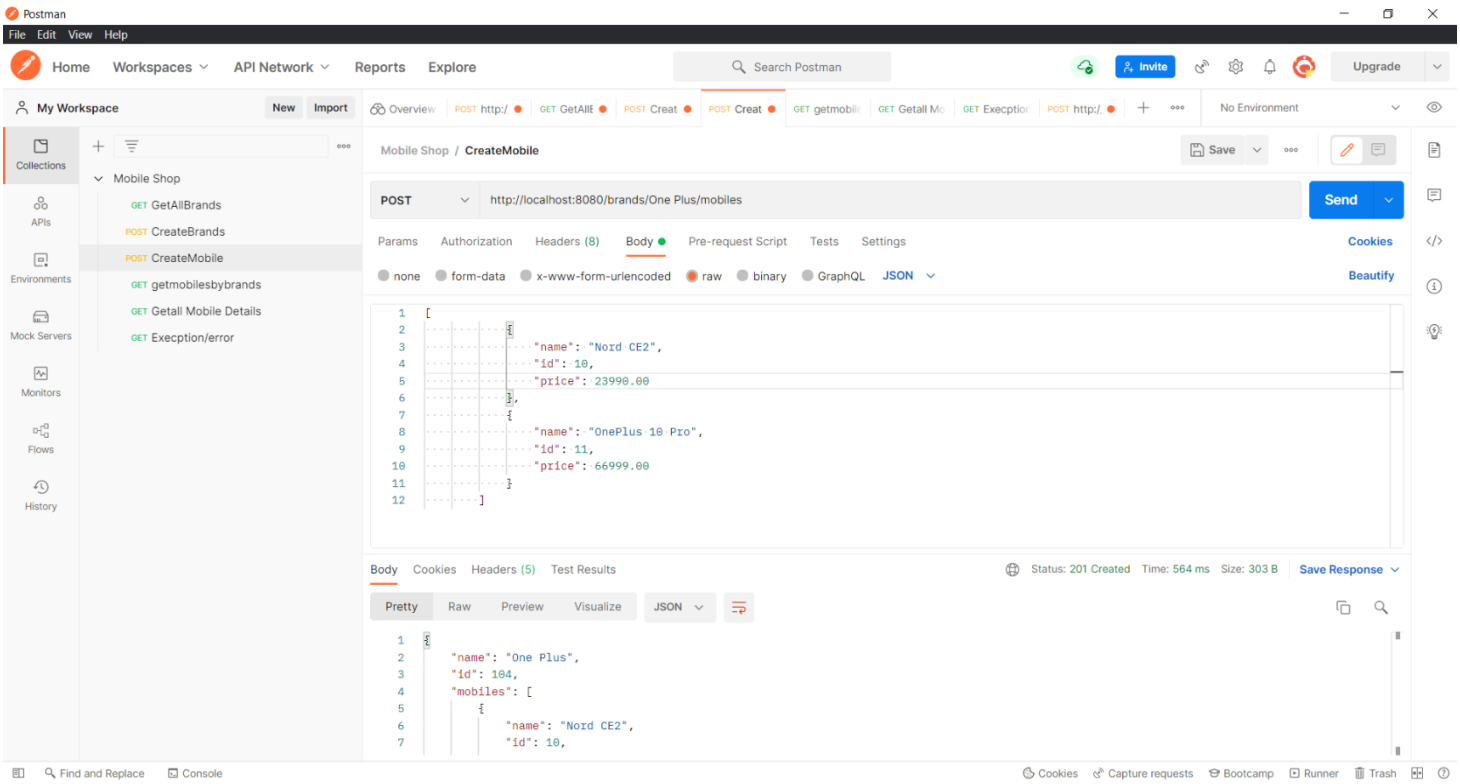
```
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Output:

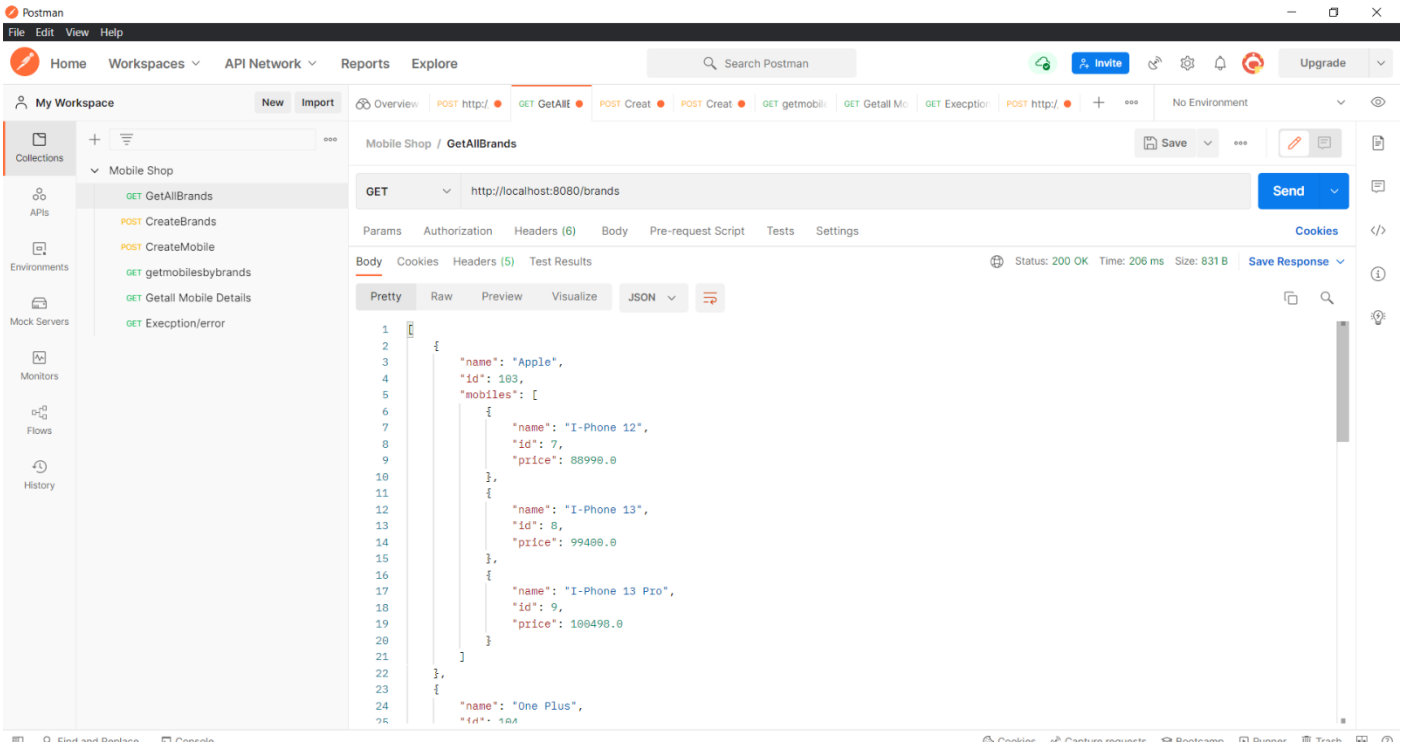
1. Create Brand . (Brand has attribute like id, name).



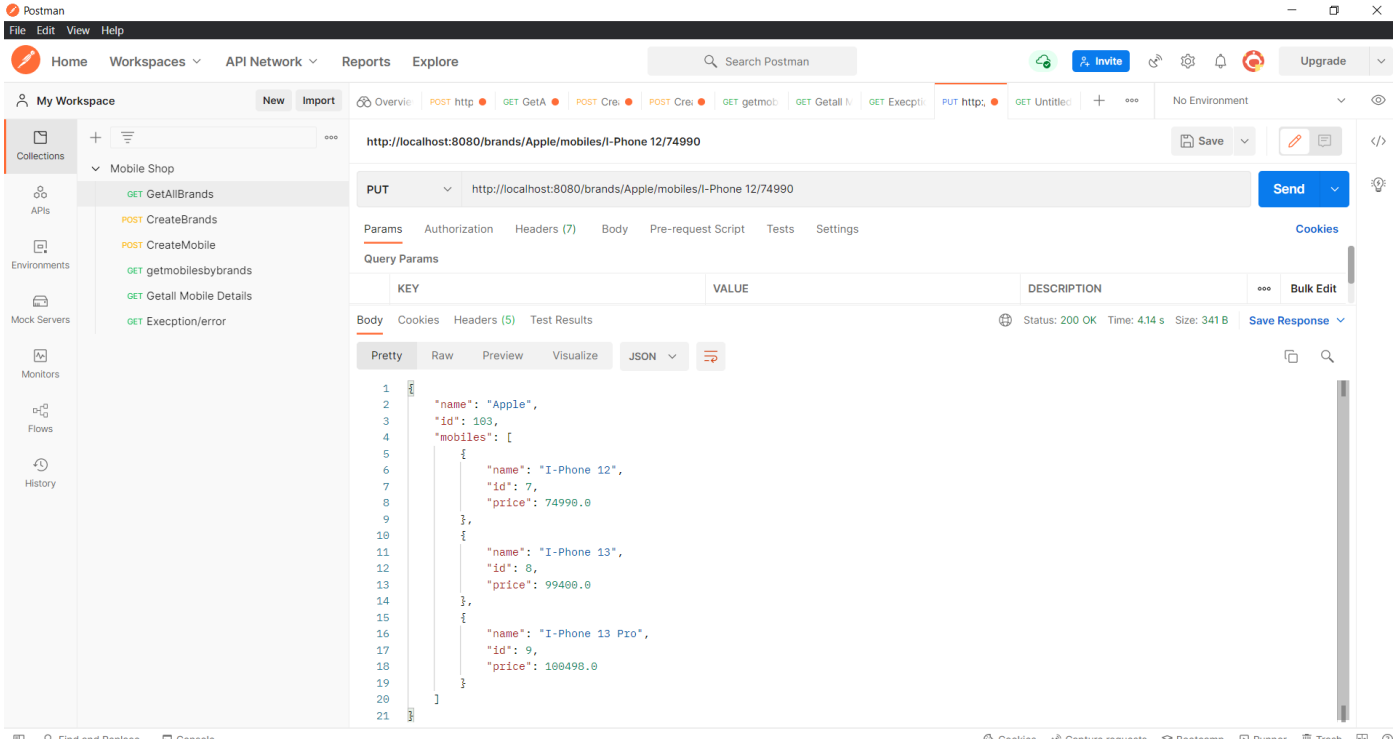
2. Create Mobile and assign it to a Brand.



3. Fetch all Brands and their corresponding mobiles based on Brand name in ascending order.



4. Update a particular mobile's price by supplying the Brand name, mobile name and price as path variable. After successfully update show the updated value in the postman.



5. Write an custom exception for usecase 4 or question 4 , if the brand name is not present then application should throw an custom exception message " No such Brand there in database"