5. Develop a Spring Boot application that provides RESTful APIs for performing CRUD operations on Mobile information. The application should use Spring Data JPA and Hibernate for data access to MySQL database that stores Mobile details such as model name, model number, device operating system, manufacturer, and country of origin. JSON may be considered as the return value of the API. Encapsulate error handling feature for situations such as record not found. Demonstrate APIs working using curl command and Postman API testing tool.

Mobile.java

```
package com.example.demo;

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

@Entity
public class Mobile {

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	Long id;
	String name;
	int model;
	String osystem;

	public Mobile() {
		// TODO Auto-generated constructor stub
	}

	public Long getId() {
		return id;
	}

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

	public String getName() {
		return name;
	}

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

	public int getModel() {
		return model;
	}

	public void setModel(int model) {
		this.model = model;
```

```
	}

	public String getOsystem() {
		return osystem;
	}

	public void setOsystem(String osystem) {
		this.osystem = osystem;
	}

}
```

MobileRepository.java

```
package com.example.demo;

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

@Repository
public interface MobileRepository extends JpaRepository<Mobile,
Long> {

}
```

MobileManager.java

```
package com.example.demo;

import java.util.List;

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

@Service
public class MobileManager {

	@Autowired
	MobileRepository repository;

	public List<Mobile> getAllMobile(){
		return repository.findAll();
	}

	public Mobile getMobileById(Long id) {
		return repository.findById(id).get();
	}

	public void saveMobile(Mobile mobile) {
		repository.save(mobile);
	}

	public void deleteMobile(Long id) {
```

```
		repository.deleteById(id);
	}

	public Boolean exists(Long id) {
		return repository.existsById(id);
	}
}
```

AppController.java

```
package com.example.demo;

import java.util.List;
import java.util.NoSuchElementException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
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 AppController {

	@Autowired
	MobileManager manager;

	@GetMapping("/mobile")
	public List<Mobile> listAllMobile() {
		return manager.getAllMobile();
	}

	@GetMapping("/mobile/{id}")
	public ResponseEntity getMobile(@PathVariable Long id) {
		try {
			Mobile mobile = manager.getMobileById(id);
			return new
ResponseEntity<Mobile>(mobile,HttpStatus.OK);
			}
			catch(NoSuchElementException e) {
				return new
ResponseEntity<Mobile>(HttpStatus.NOT_FOUND);
			}
	}

	@PostMapping("/mobile")
	public void saveMobile(@RequestBody Mobile mobile) {
		manager.saveMobile(mobile);
	}
```

```
	@PutMapping("/mobile")
	public void updateMobile(@RequestBody Mobile mobile) {
		manager.saveMobile(mobile);
	}


	//Using <Mobile> Template for ResponseEntity is Optional here
	@DeleteMapping("/mobile/{id}")
	public ResponseEntity<Mobile> deleteMobile (@PathVariable Long
id) {

		if(manager.exists(id))
		{
			manager.deleteMobile(id);
			return new ResponseEntity<Mobile>(HttpStatus.OK);
		}
		else {
			return new
ResponseEntity<Mobile>(HttpStatus.NOT_FOUND);
		}
	}
}
```

Edit the **application.properties** file available under **/src/main/resources** to add the following three lines (change the database name, username and password for MySQl server accordingly)

```
spring.datasource.url=jdbc:mysql://localhost:3306/spring
spring.datasource.username=root
spring.datasource.password=password
```

MySql server database

- **Logging in to mysql**

  ```
  mysql -u -p student

  password: student
  ```

- **select the database for use**

  ```
  use student;
  ```

- **Query to create the table in MySql**

  ```
  create table mobile (id integer(4) primary key auto_increment,
  name varchar(50), model varchar(50), osystem varchar(50));
  ```

- **Query to describe the table schema**

  ```
  desc mobile;
  ```

- **Query to select and display all the rows and columns of the table**

```
select * from mobile;
```