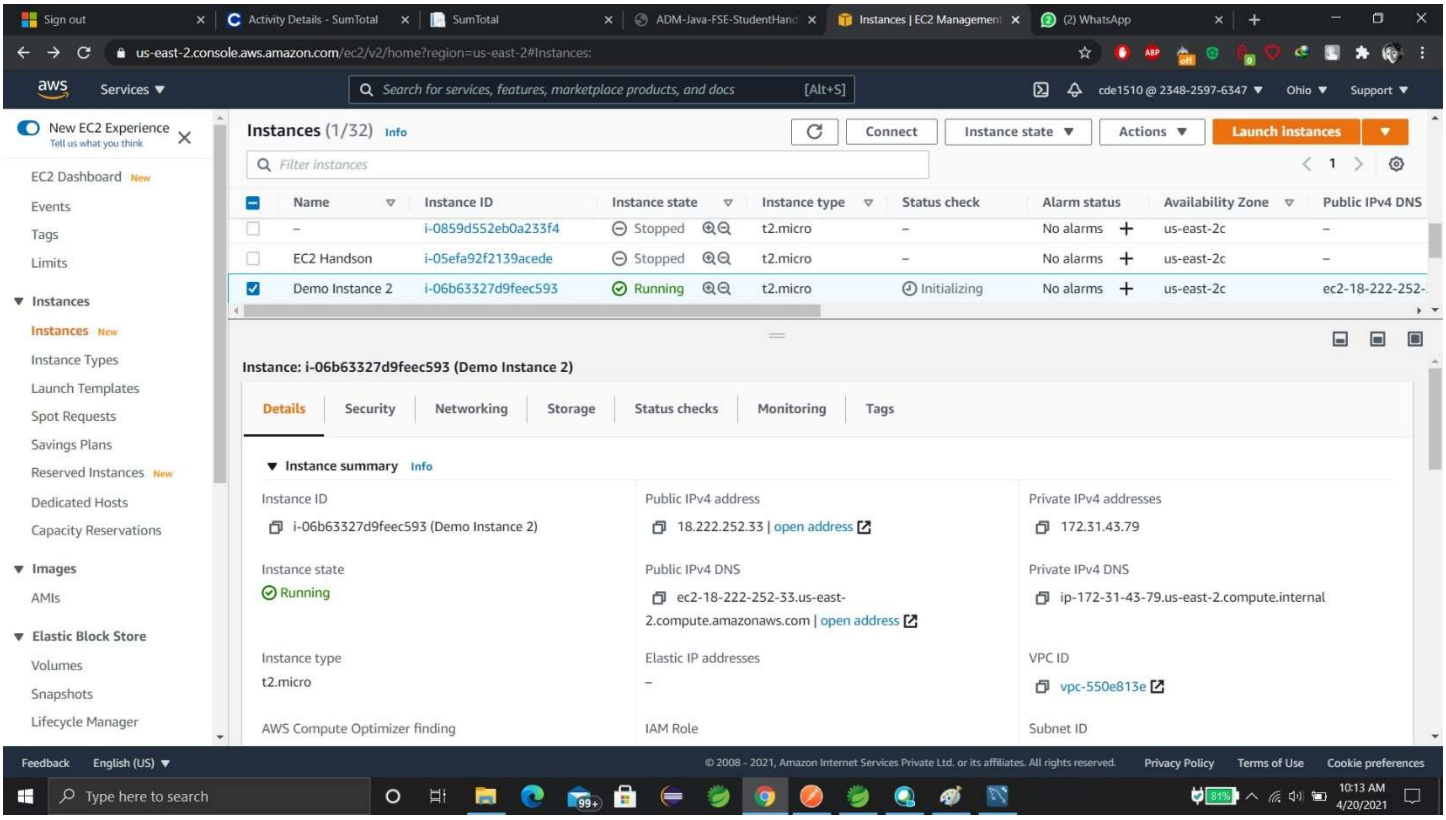
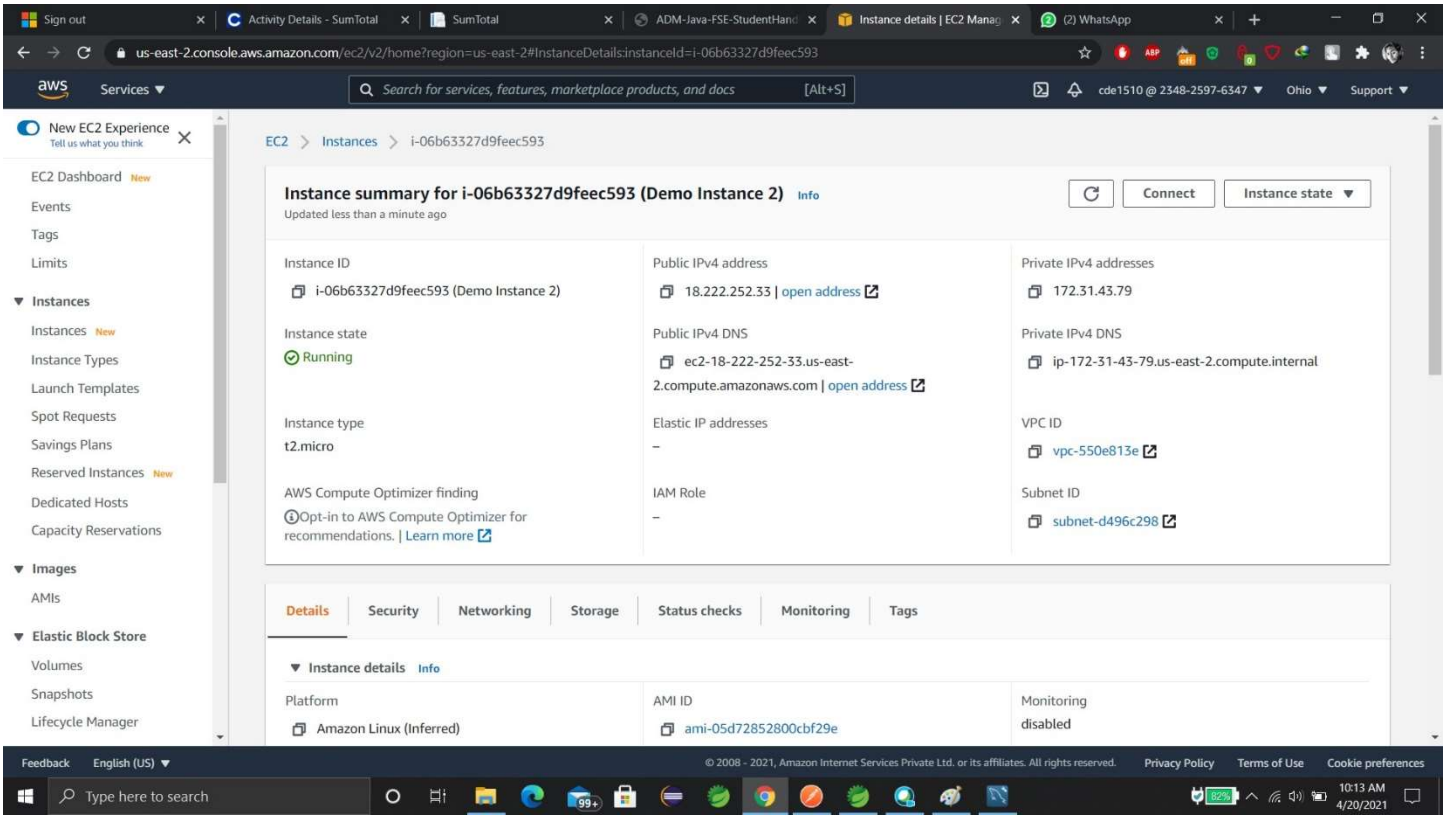
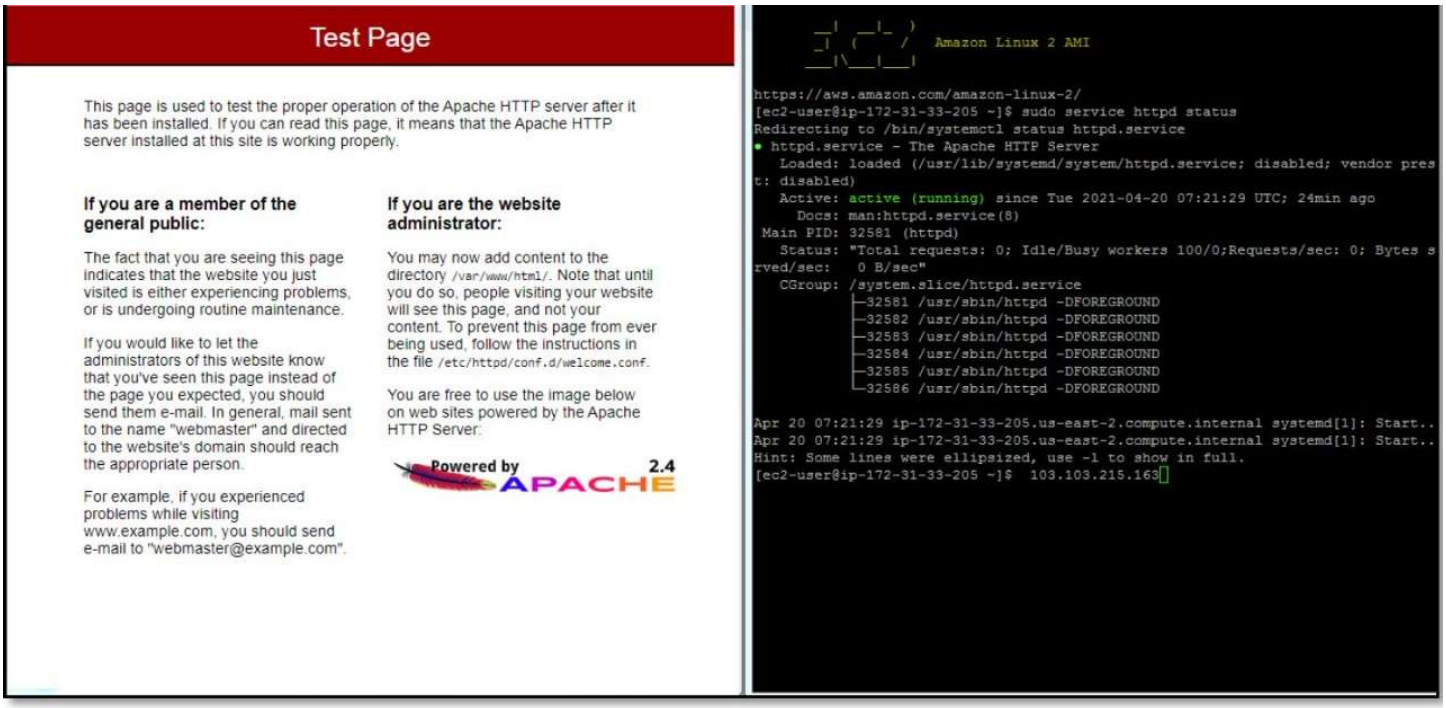
AWS HandsOn

EC2 Handson

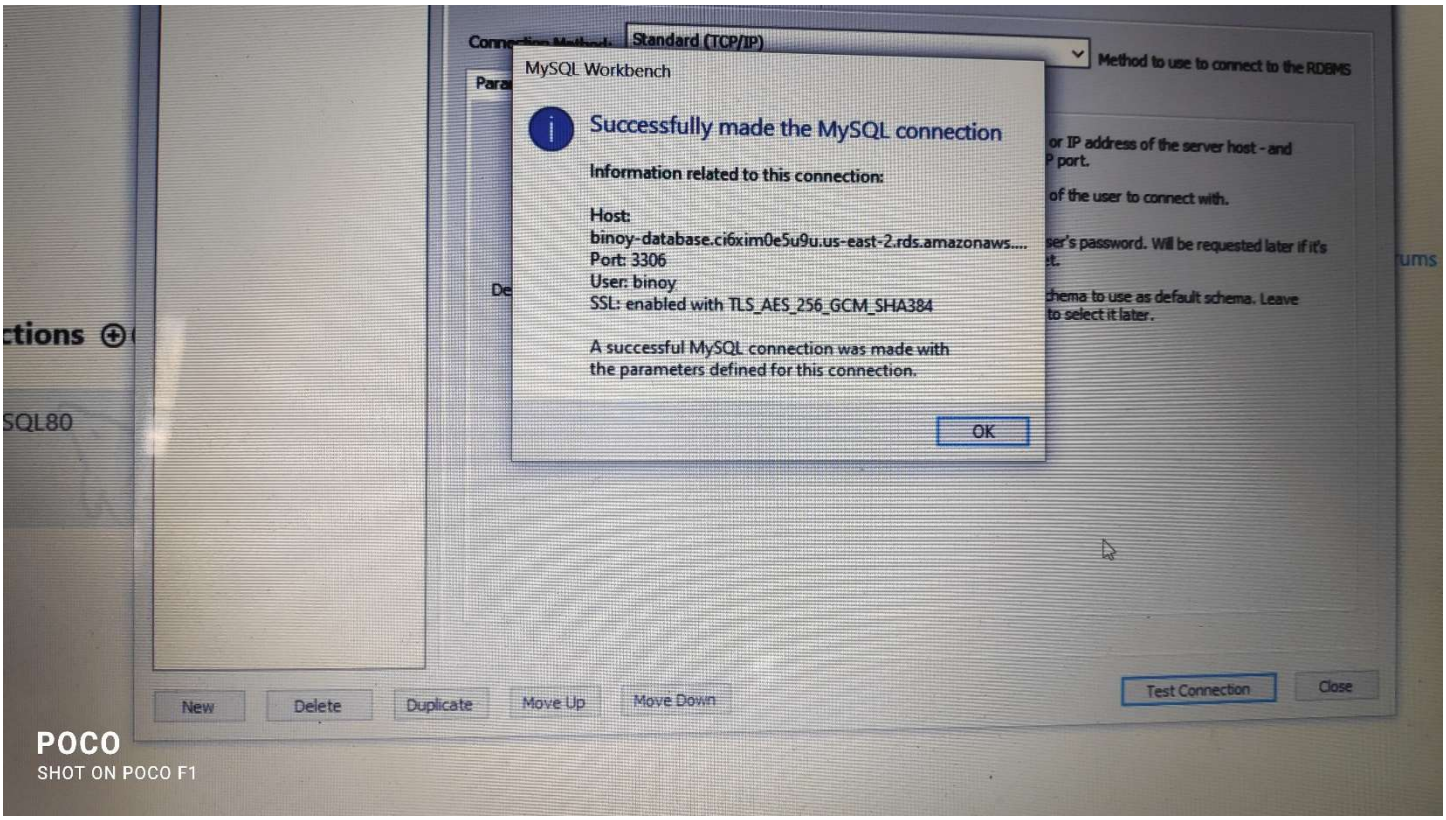


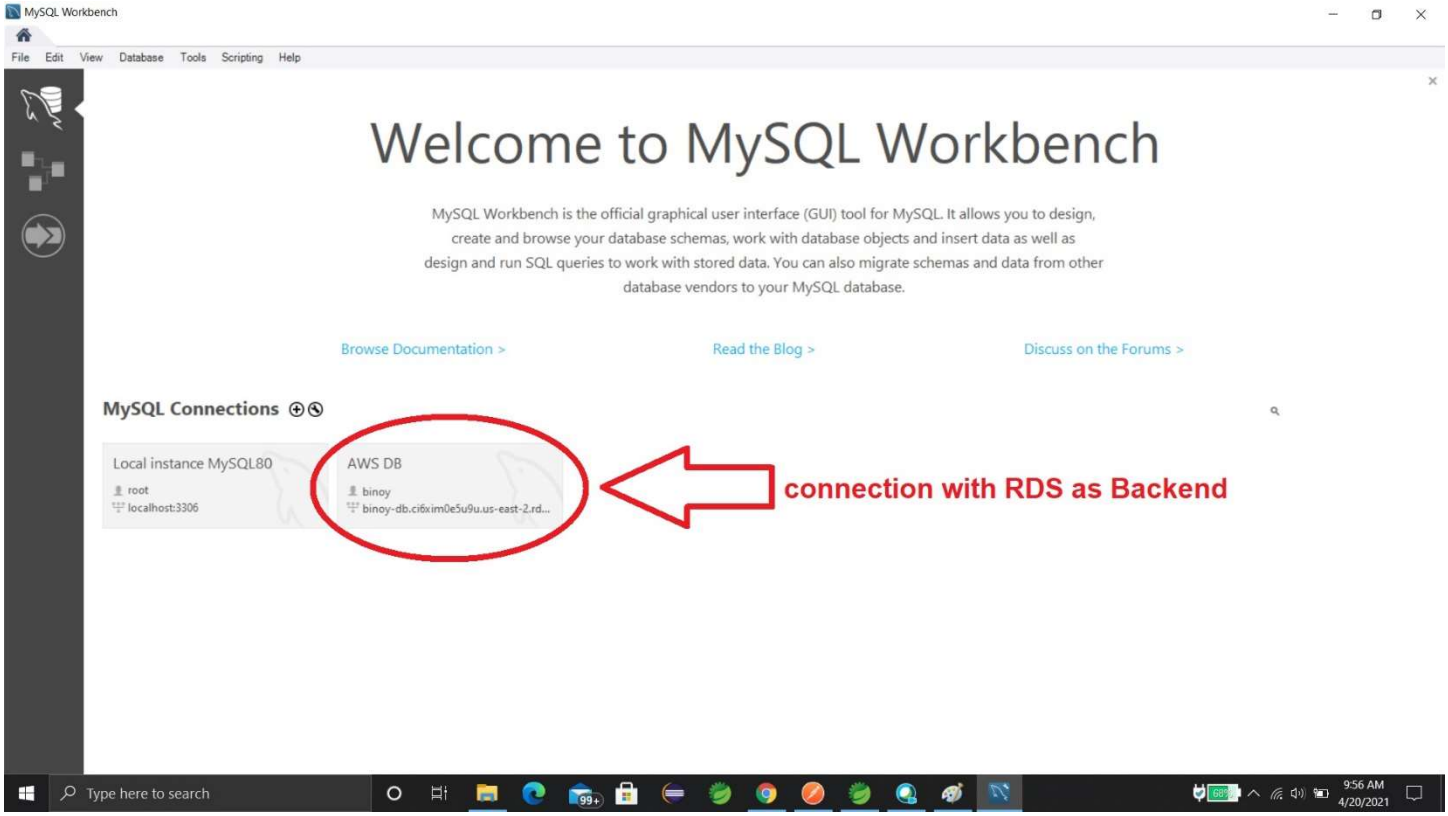




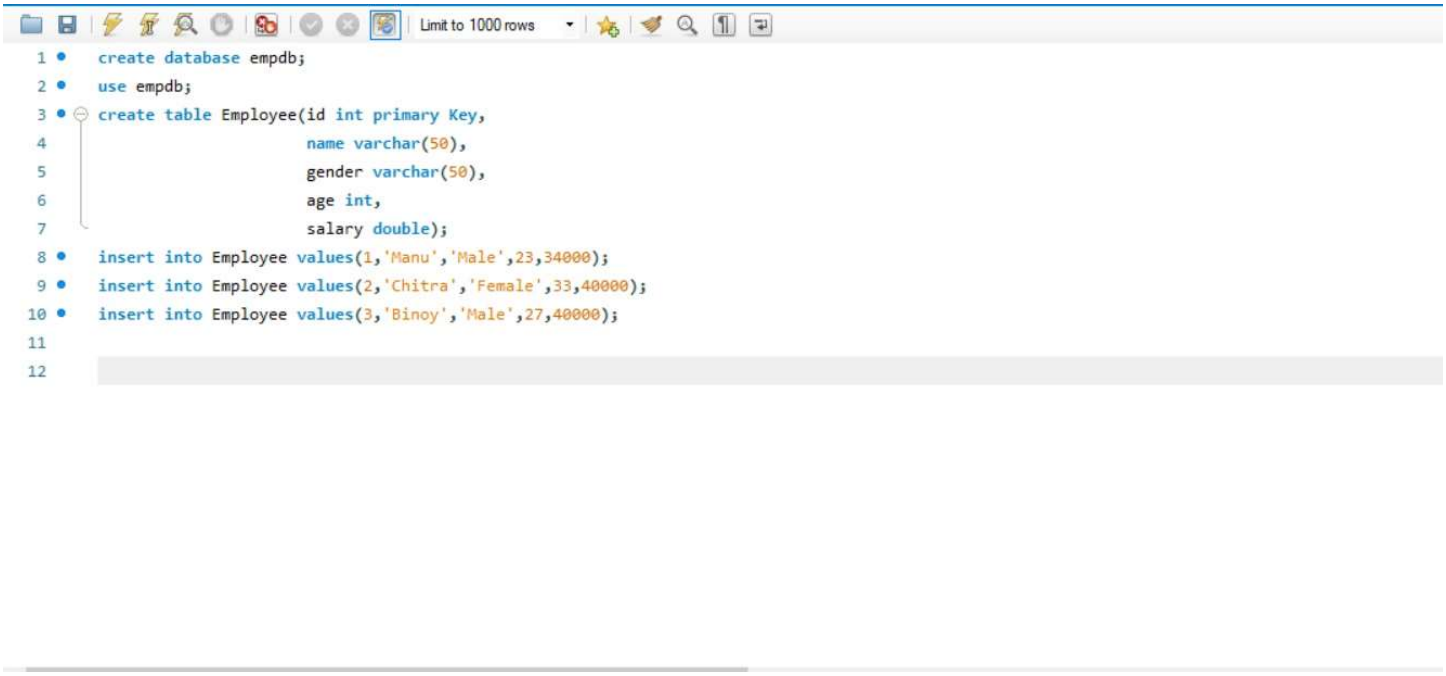
Type the public IPV4 address on the browser url bar and we should get the above shown screen

RDS_HandsOn





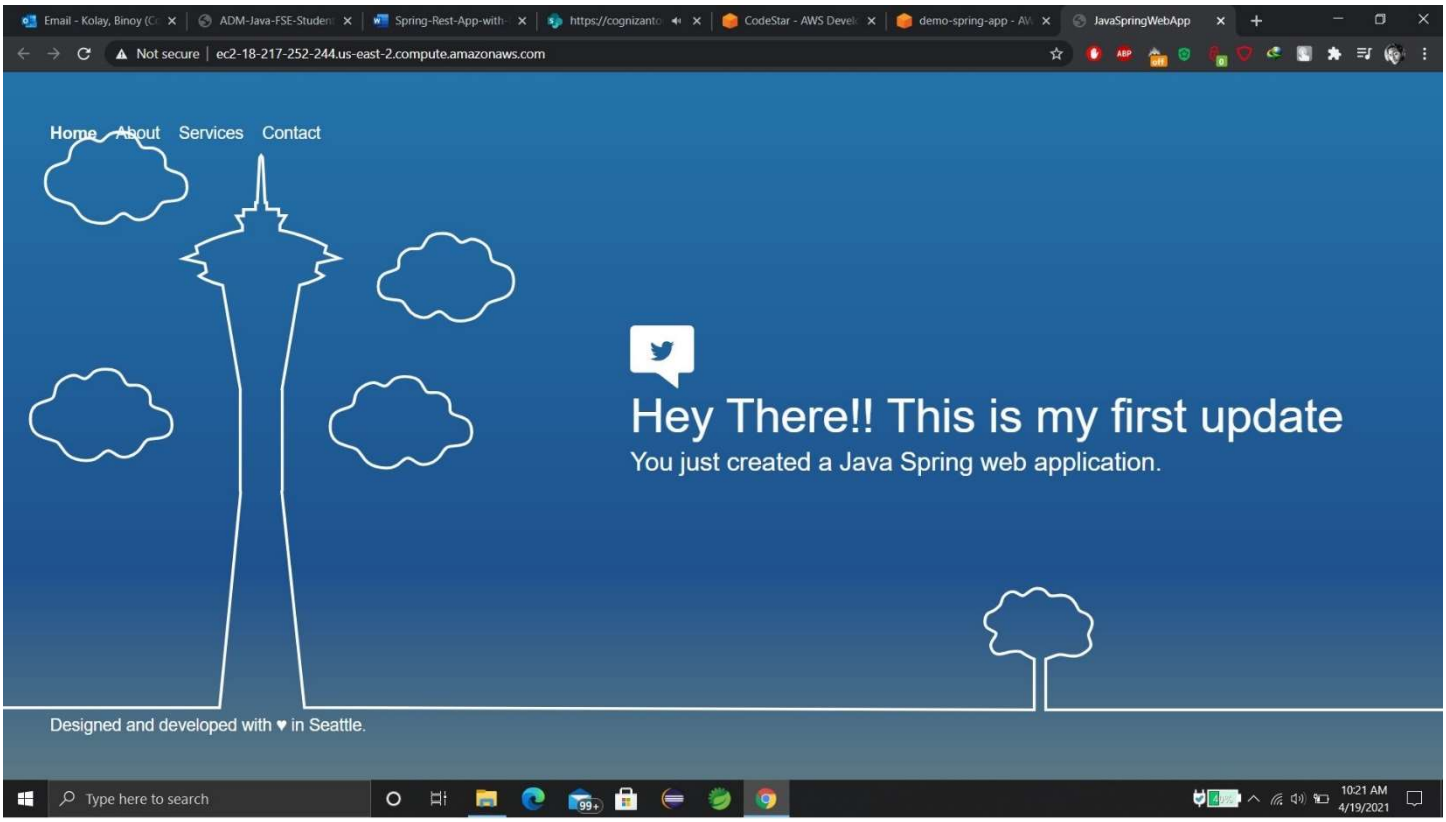
Data in RDS



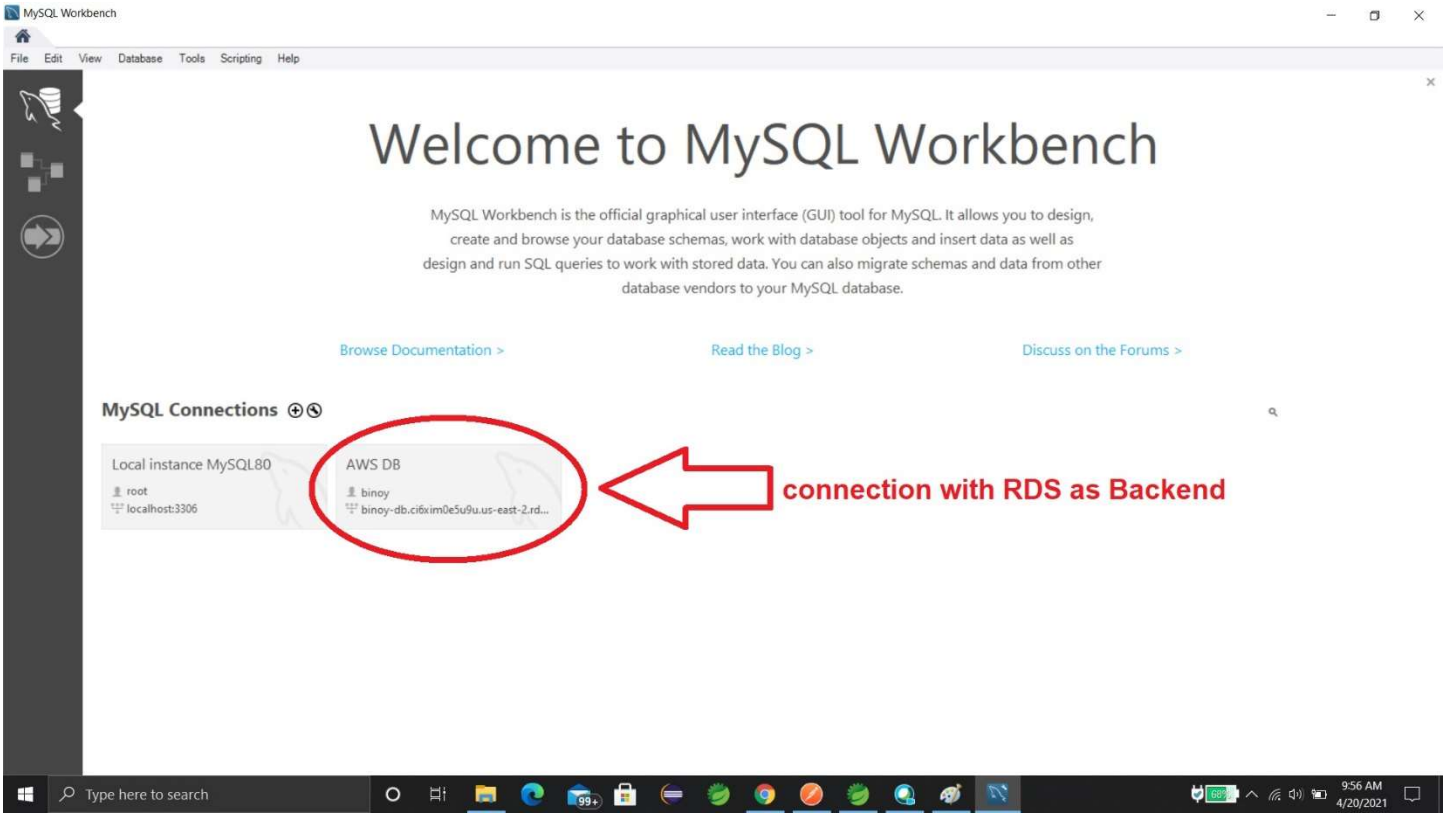
```
create database empdb;
use empdb;
create table Employee(id int primary Key,
                      name varchar(50),
                      gender varchar(50),
                      age int,
                      salary double);
insert into Employee values(1,'Manu','Male',23,34000);
insert into Employee values(2,'Chitra','Female',33,40000);
insert into Employee values(3,'Binoy','Male',27,40000);
```

CI/CD HandsOn

Output:-



Spring-Rest-with-RDS-Backend HandsOn



```
create database empdb;
use empdb;
create table Employee(id int primary Key,
                      name varchar(50),
                      gender varchar(50),
                      age int,
                      salary double);
insert into Employee values(1,'Manu','Male',23,34000);
insert into Employee values(2,'Chitra','Female',33,40000);
insert into Employee values(3,'Binoy','Male',27,40000);
```

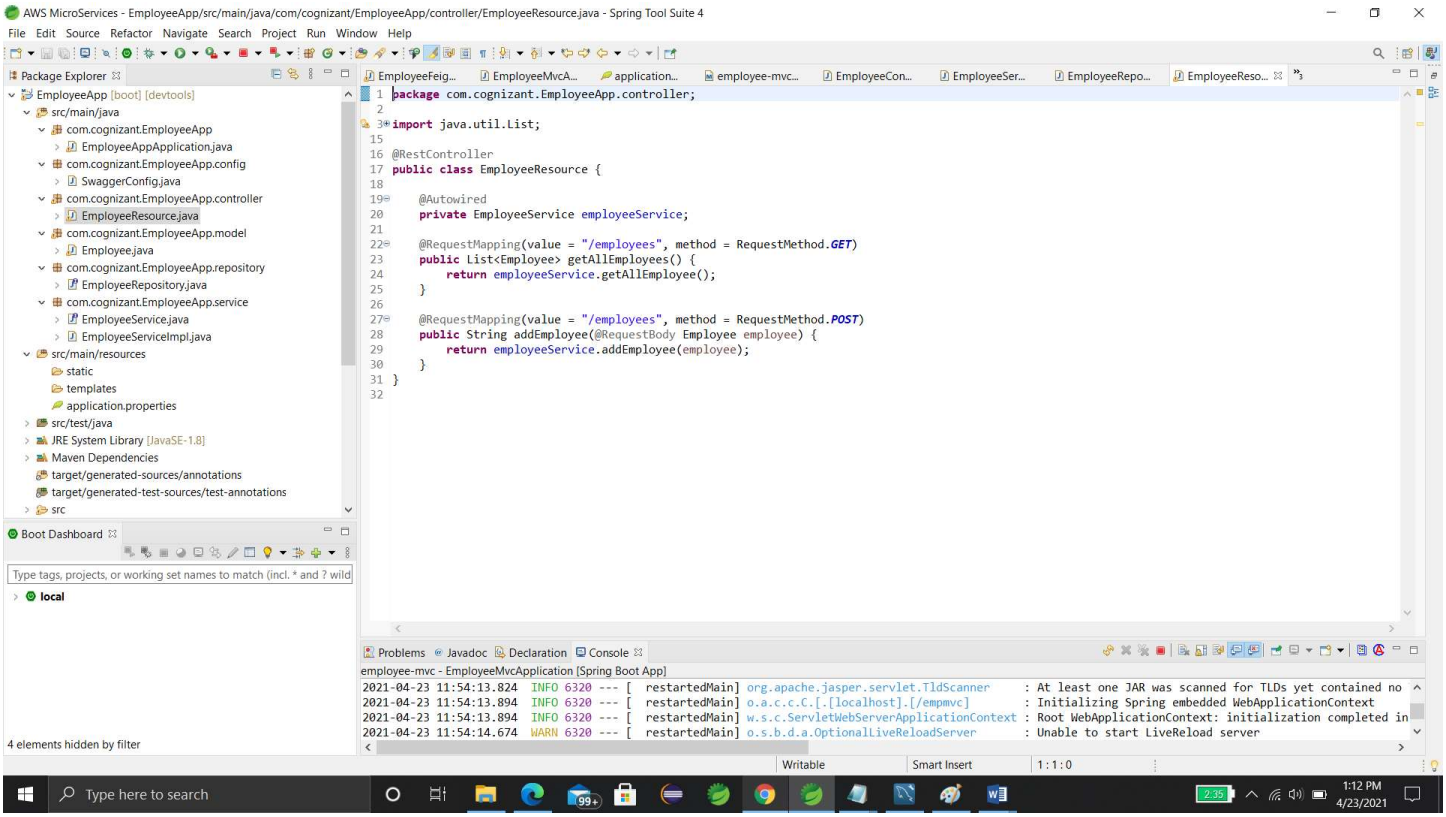
```
package com.cognizant.controller;

import java.util.List;

@RestController
@RequestMapping("/employees")
public class EmployeeController {

    @Autowired
    EmployeeService employeeService;

    @GetMapping
    public List<Employee> getAllEmployee(){
        return employeeService.getAllEmployee();
    }

    @PostMapping
    public String addEmployee(@RequestBody Employee employee) {
        return employeeService.addEmployee(employee);
    }

    @DeleteMapping("/{id}")
    public String deleteEmployee(@PathVariable int id) {
        return employeeService.deleteEmployee(id);
    }

    @PutMapping
    public String updateEmployee(@RequestBody Employee employee) {
        return employeeService.updateEmployee(employee);
    }

    @GetMapping("/{id}")
    public Employee getEmployeeBy(@PathVariable int id) {
        return employeeService.getEmployeeBy(id);
    }
```

```
2021-04-23 11:57:48.223 DEBUG 7112 --- [nio-8080-exec-3] org.hibernate.SQL                        : select employee0_.id as id1_0_0_, employee0_.age as age
2021-04-23 11:57:48.228 TRACE 7112 --- [nio-8080-exec-3] o.h.type.descriptor.sql.BasicBinder       : binding parameter [1] as [INTEGER] - [4]
2021-04-23 11:57:48.247 TRACE 7112 --- [nio-8080-exec-3] o.h.type.descriptor.sql.BasicExtractor     : extracted value ([age2_0_0_] : [INTEGER]) - [32]
```

We have created a “employee” microservice to test the RDS Database.

Swagger HandsOn

Step 1:- Create a simple RESTful service using Spring BOOT



Step-2:- Add Dependencies

```
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>3.0.0</version>
        </dependency>

        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>3.0.0</version>
        </dependency>
    </dependencies>

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

Swagger Dependencies

Step 3:- Create a Swagger configuration class

MicroServices - EmployeeApp/src/main/java/com/cognizant/EmployeeApp/config/SwaggerConfig.java - Spring Tool Suite 4

```java
package com.cognizant.EmployeeApp.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {


    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2).select()
                .apis(RequestHandlerSelectors.basePackage("com.cognizant.EmployeeApp")).build();
    }

}
```

```
2021-04-20 09:48:07.626  INFO 18696 --- [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.4.30.Final
2021-04-20 09:48:08.247  INFO 18696 --- [  restartedMain] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Fi
2021-04-20 09:48:08.595  INFO 18696 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2021-04-20 09:48:10.793  INFO 18696 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2021-04-20 09:48:10.886  INFO 18696 --- [  restartedMain] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.MySQ
2021-04-20 09:48:13.993  INFO 18696 --- [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hi
2021-04-20 09:48:14.022  INFO 18696 --- [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence
2021-04-20 09:48:14.201  INFO 18696 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2021-04-20 09:48:16.417  WARN 18696 --- [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. There
2021-04-20 09:48:17.465  INFO 18696 --- [  restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecuto
2021-04-20 09:48:18.179  INFO 18696 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context
2021-04-20 09:48:19.046  INFO 18696 --- [  restartedMain] c.c.EmployeeApp.EmployeeAppApplication   : Started EmployeeAppApplication in 20.848 seconds (JV
2021-04-20 09:48:32.542  INFO 18696 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherSer
2021-04-20 09:48:32.542  INFO 18696 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2021-04-20 09:48:32.548  INFO 18696 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 4 ms
```

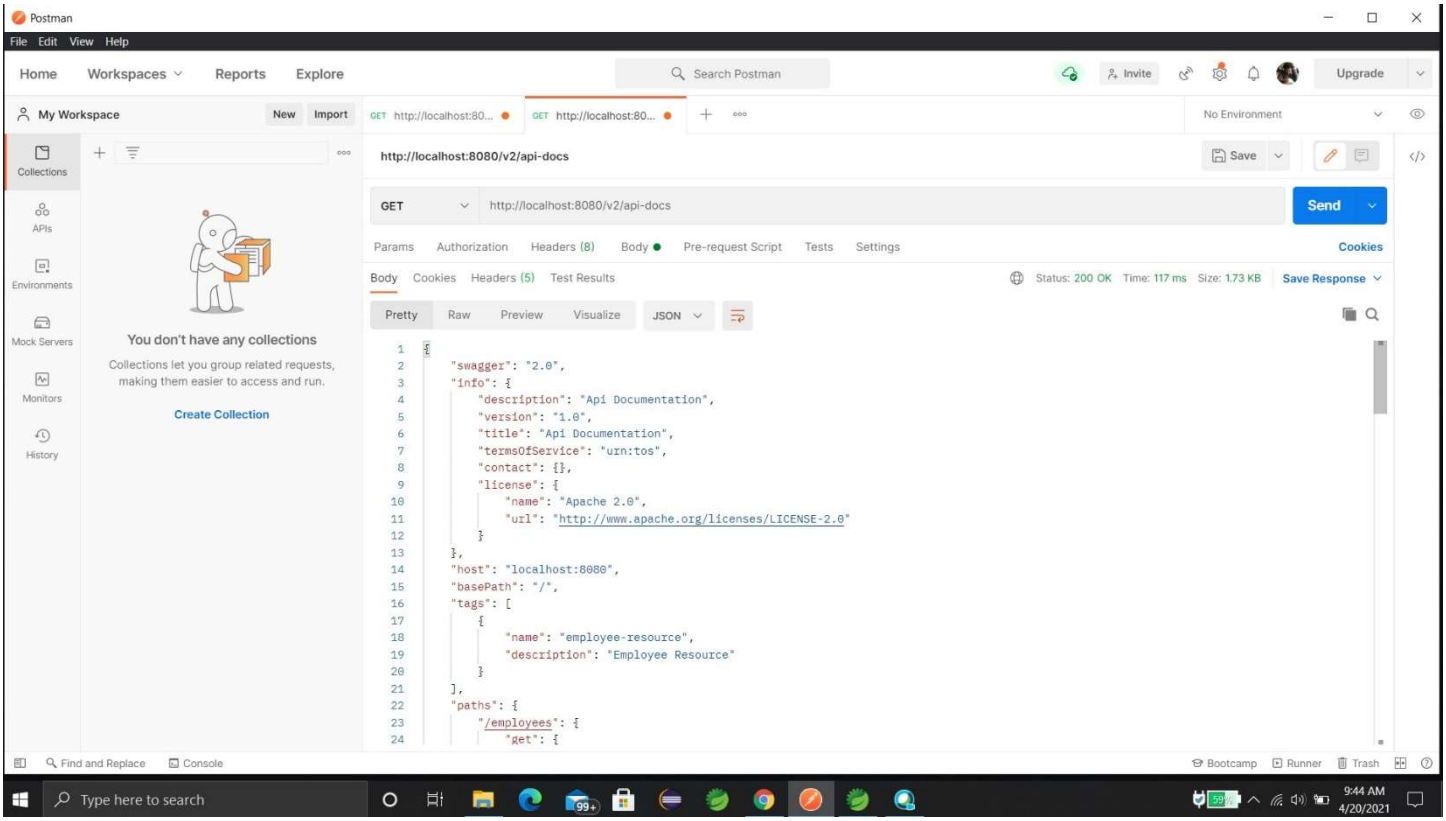
Step 4:- use Swagger specific annotations to customize the descriptions of model class and the properties.

MicroServices - EmployeeApp/src/main/java/com/cognizant/EmployeeApp/model/Employee.java - Spring Tool Suite 4

```java
package com.cognizant.EmployeeApp.model;

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

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@ApiModel(description = "Employee class which is acting as the DTO")
public class Employee {

    @Id
    private int id;
    @ApiModelProperty(notes = "Name should contain only Alphabets")
    private String name;
    private String gender;
    @ApiModelProperty(notes = "Age should be between 18 and 60")
    private int age;
    private double salary;
}
```

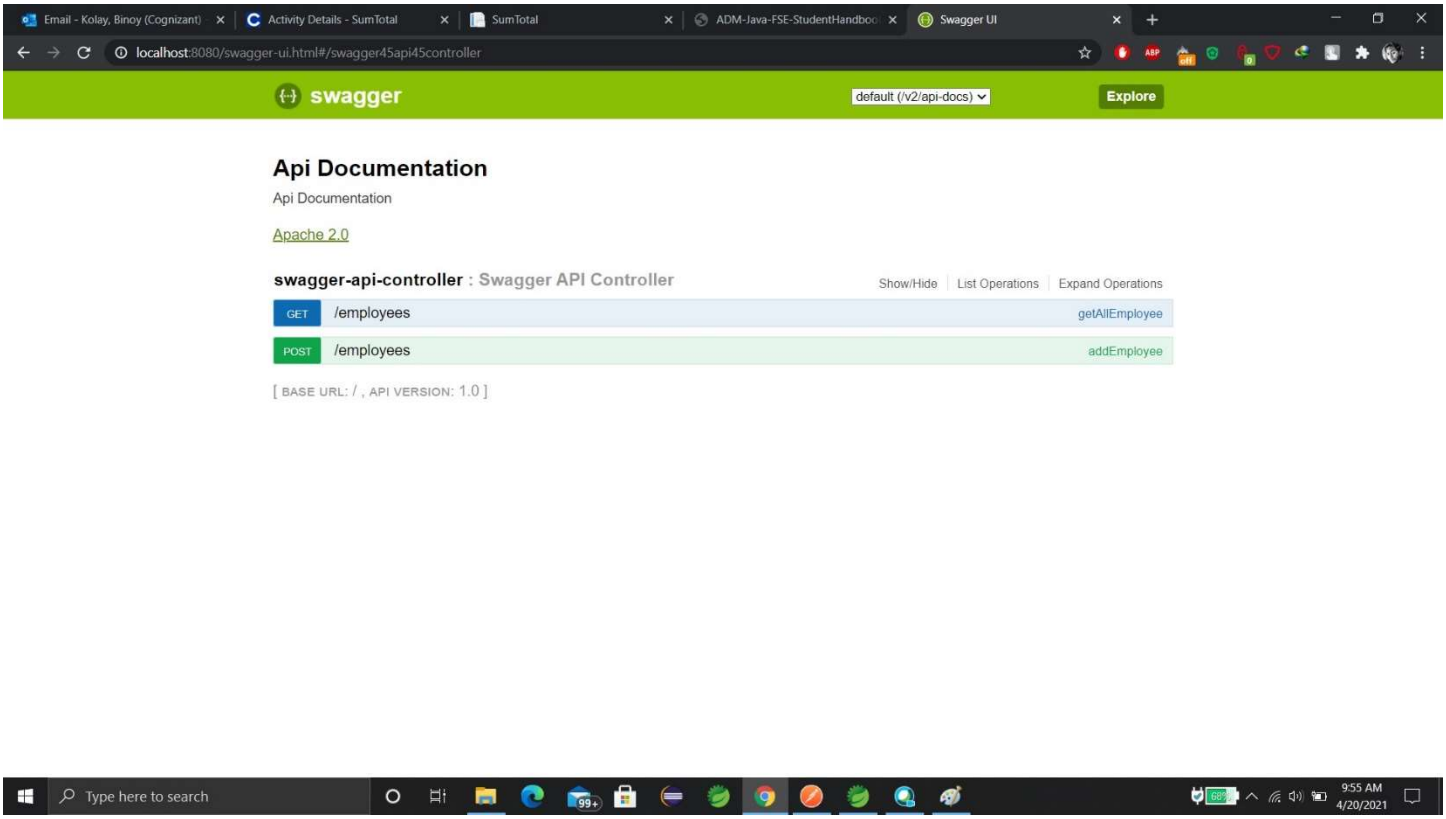
Swagger specific annotations to customize the descriptions of model class and the properties

"**localhost:8080/v2/api-docs**" and you can see the complete API documentation of your service.



Now, hit the URL in your web browser and see the Swagger API functionalities.

http://localhost:8080/swagger-ui.html



Spring MVC Client For Spring REST Service

Note:- We have already created a microservice(employee) in our local System. Now, we are just creating another microservice which will consume the rest service of our previous employee microservice.

First we have to add “openfeign” dependency

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</artifactId>
    </dependency>

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

Feign Client dependecy

EmployeeRepository.java | EmployeeFeign.java | MvcController.java | application.properties

```
package com.cts.feign;

import java.util.List;

@FeignClient(name = "emp-spring", url = "${feign.url-employee-spring}")
public interface EmployeeFeign {

    @GetMapping
    public List<Employee> getAllEmployee();

    @GetMapping("/{id}")
    public Employee getEmployeeByID(@PathVariable int id);

    @DeleteMapping("/{id}")
    public String deleteEmployee(@PathVariable int id);

    @PostMapping
    public String addEmployee(@RequestBody Employee employee);

    @PutMapping
    public String updateEmployee(@RequestBody Employee employee);
}
```

Declaring this interface as Feign Client

Note:-The *value* argument passed in the ***@FeignClient*** annotation is a mandatory, arbitrary client name, while with the *url* argument, we specify the API base URL.

Furthermore, since this interface is a Feign client, we can use the Spring Web annotations to declare the APIs that we want to reach out to.

EmployeeRepository.java | EmployeeFeign.java | MvcController.java | application.properties | EmployeeMvcApplication.java

```
1  package com.cts;
2
3  import org.springframework.boot.SpringApplication;
6
7  @SpringBootApplication
8  @EnableFeignClients
9  public class EmployeeMvcApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(EmployeeMvcApplication.class, args);
13     }
14
15 }
16
```

With this annotation, we enable component scanning for interfaces that declare they are Feign clients.

With this annotation, we enable component scanning for interfaces that declare they are Feign clients.

Output:-

output:-