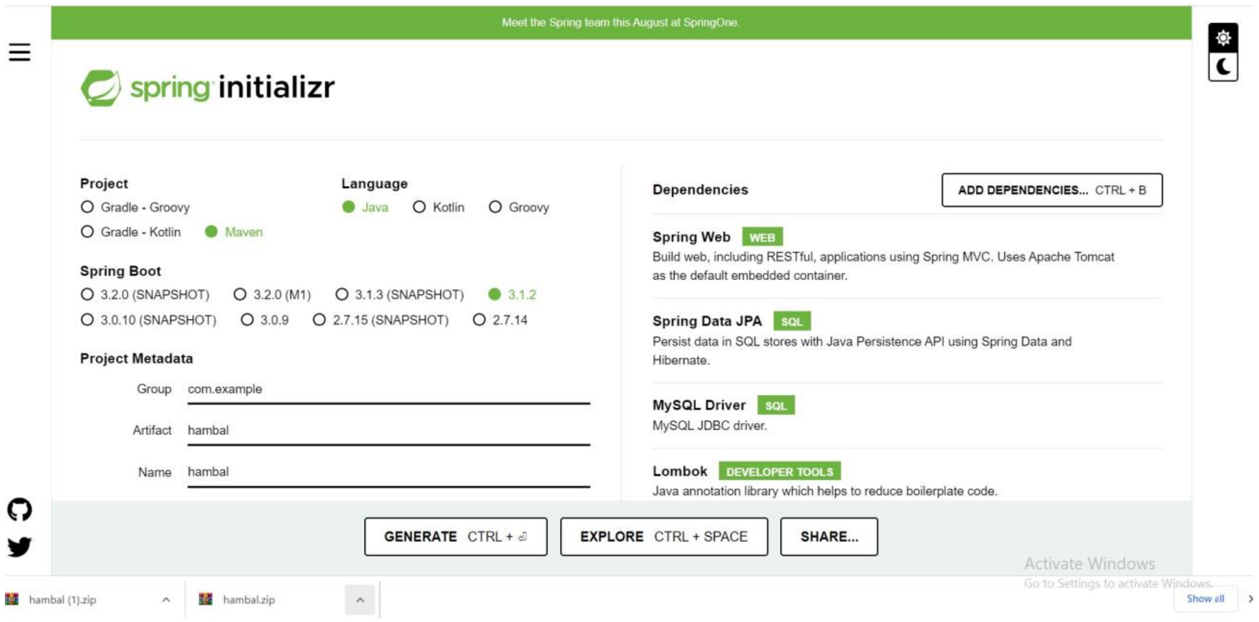
# Markup Test 2023

a) Create Spring Boot Project

Write a step to create a project using spring initializer or using maven commands

## Steps to create a project using spring initializer

- ➢ Search in browser by write spring initializer
- ➢ Select spring initializer
- ➢ Select in Project,Language and Spring boot
- ➢ Add dependences
- ➢ Generate



Project
Gradle - Groovy
Gradle - Kotlin [✓ Maven]

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.2.0 (SNAPSHOT) 3.2.0 (M1) 3.1.3 (SNAPSHOT) [✓ 3.1.2]
3.0.10 (SNAPSHOT) 3.0.9 2.7.15 (SNAPSHOT) 2.7.14

Project Metadata

| | |
|---|---|
| Group | com.example |
| Artifact | hambal |
| Name | hambal |

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC driver.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

b) Maven Dependencies

```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>3.1.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>hambal</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>hambal</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>17</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-jdbc</artifactId>
        </dependency>
        <dependency>
           <groupId>org.springframework.boot</groupId>
           <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
           <groupId>com.mysql</groupId>
           <artifactId>mysql-connector-j</artifactId>
           <scope>runtime</scope>
        </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>
    </dependencies>

    <build>
        <plugins>
           <plugin>
              <groupId>org.springframework.boot</groupId>
              <artifactId>spring-boot-maven-plugin</artifactId>
              <configuration>
                 <excludes>
                    <exclude>
                       <groupId>org.projectlombok</groupId>
                       <artifactId>lombok</artifactId>
                    </exclude>
                 </excludes>
              </configuration>
           </plugin>
        </plugins>
    </build>

</project>
```

c) Configure MySQL/PostgreSQL Database: Using application.properties or application.yml file for database connectivity, create a properties file.

```
spring.datasource.url=jdbc:mysql://localhost:3306/employee_db
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

# Logging SQL queries (optional but helpful for debugging)
spring.jpa.show-sql=true
```

d) Create Models

e) Write a code for employee model

```
package com.example.hambal;

import javax.persistence.*;

@Entity
@Table(name = "employees")
public class Employee {

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

    @Column(unique = true, nullable = false)
    private String employeeNumber;

    @Column(nullable = false)
    private String firstName;

    @Column(nullable = false)
    private String lastName;

    @Column(unique = true, nullable = false)
    private String email;
}
```

f) Employee Repository: Write a code for employee repository

```
package com.example.hambal;

import com.example.hambal.Employee;
import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRep extends JpaRepository<Employee, Long> {
    Employee findByEmployeeNumber(String employeeNumber);
    Employee findByEmail(String email);
}
```

g) Services Layer: Write a code for employee service layer

```
package com.example.hambal.service;

import com.example.hambal.Employee;
```

```java
public interface EmpService {
    Employee createEmployee(Employee employee);
    Employee getEmployeeById(Long id);
}
```

h) Controller Layer: Write a code for GET,POST,DELETE,PUT and GET by ID

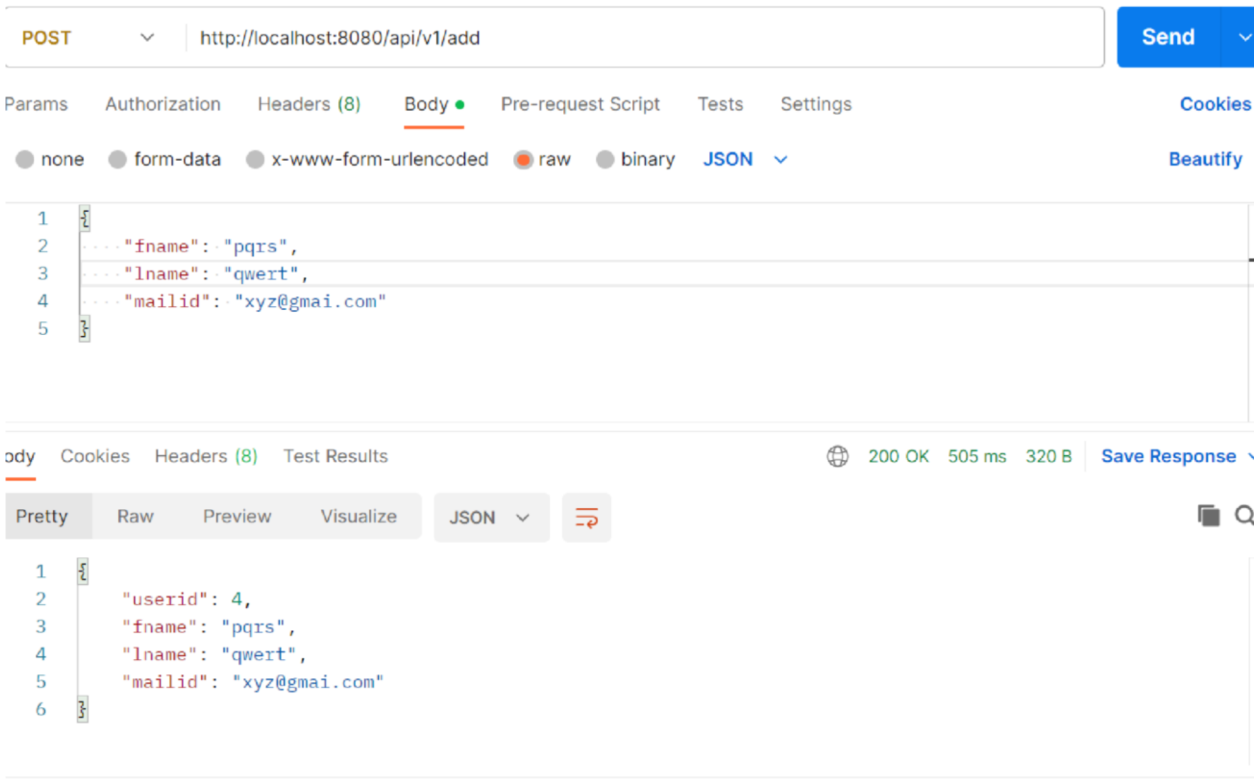
```java
package com.example.demo.controller;

import com.example.demo.model.Employee;
import com.example.demo.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/employees")
public class EmplController {

    @Autowired
    private EmpService employeeService;

    @PostMapping
    public ResponseEntity<Employee> createEmployee(@RequestBody
Employee employee) {
        Employee createdEmployee =
employeeService.createEmployee(employee);
        return ResponseEntity.ok(createdEmployee);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Employee> getEmployeeById(@PathVariable Long
id) {
        Employee employee = employeeService.getEmployeeById(id);
        if (employee == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(employee);
    }
}
```

i) Create anEmployee Application class which will be used as Main Class.

```java
package com.example.hambal;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class EmpServiceApp {

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

j) Show how you can test your API using API client.

1) POST



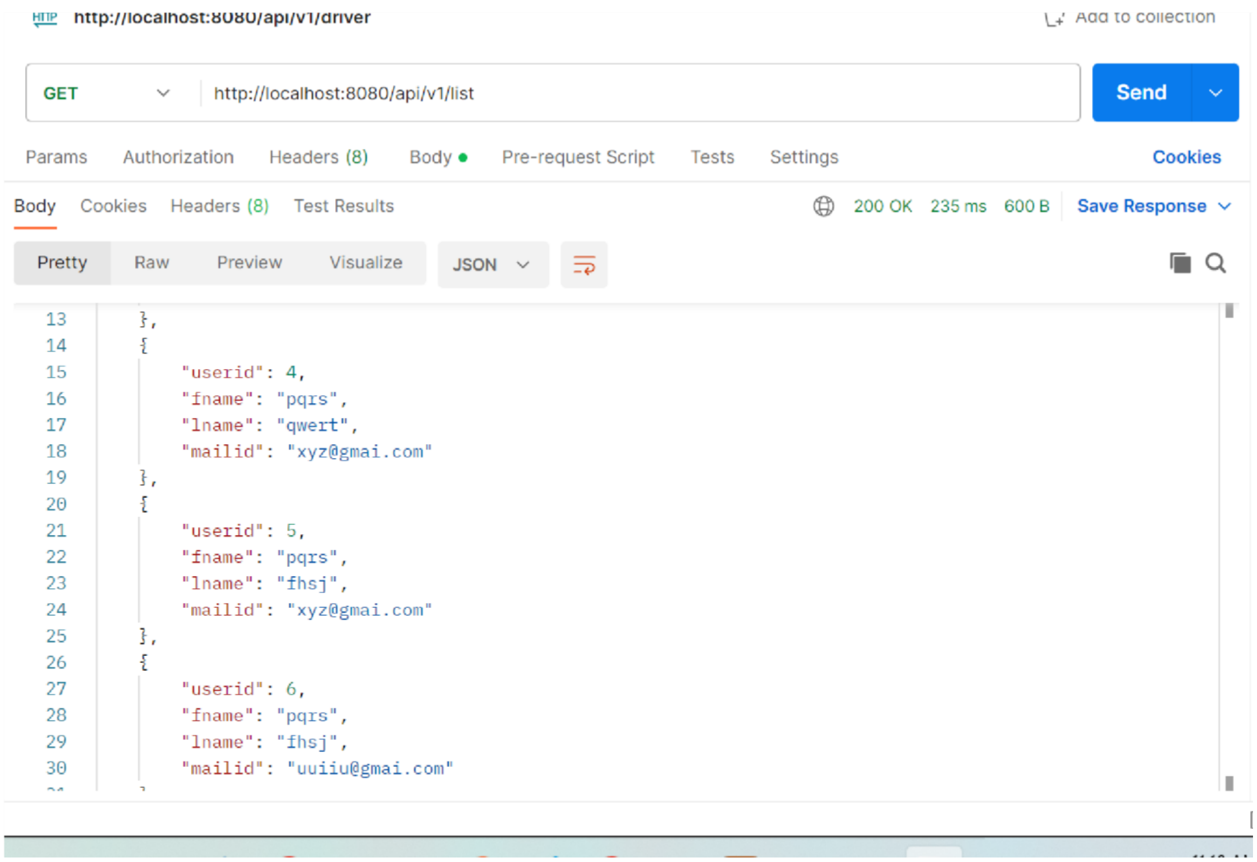
POST http://localhost:8080/api/v1/add

```
{
    "fname": "pqrs",
    "lname": "qwert",
    "mailid": "xyz@gmai.com"
}
```

200 OK 505 ms 320 B

```
{
    "userid": 4,
    "fname": "pqrs",
    "lname": "qwert",
    "mailid": "xyz@gmai.com"
}
```

2) GET ALL



http://localhost:8080/api/v1/list

200 OK 235 ms 600 B

```
    },
    {
        "userid": 4,
        "fname": "pqrs",
        "lname": "qwert",
        "mailid": "xyz@gmai.com"
    },
    {
        "userid": 5,
        "fname": "pqrs",
        "lname": "fhsj",
        "mailid": "xyz@gmai.com"
    },
    {
        "userid": 6,
        "fname": "pqrs",
        "lname": "fhsj",
        "mailid": "uuiiu@gmai.com"
```