TESTING

Junit

Pom.xml

```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>2.6.3</version>
            <relativePath /> <!-- lookup parent from repository -->
      </parent>
      <groupId>com.insurance</groupId>
      <artifactId>management</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <name>management</name>
      <description>Demo project for Spring Boot</description>
      <properties>
            <java.version>11</java.version>
      </properties>
      <dependencies>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter</artifactId>
            </dependency>
            <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>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-jdbc</artifactId>
            </dependency>
            <dependency>
                  <groupId>org.springframework.session</groupId>
```

```xml
                        <artifactId>spring-session-core</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.session</groupId>
                        <artifactId>spring-session-jdbc</artifactId>
                </dependency>
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-java</artifactId>
                        <scope>runtime</scope>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <scope>test</scope>
                </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-tomcat</artifactId>
                        <scope>provided</scope>
                </dependency>
        </dependencies>

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

</project>
```

ManagementApplicationTests.java

```java
package com.insurance.management;

import static org.assertj.core.api.Assertions.assertThat;
import static org.junit.Assert.assertNotEquals;
import static org.junit.Assert.assertNotNull;
import static org.junit.jupiter.api.Assertions.assertEquals;

import java.util.Date;
import java.util.List;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import com.insurance.management.model.Approvals;
import com.insurance.management.model.Category;
import com.insurance.management.model.Disapproved;
import com.insurance.management.model.Policies;
import com.insurance.management.model.Queries;
import com.insurance.management.model.User;
import com.insurance.management.repository.ApprovalsRepository;
import com.insurance.management.repository.ApprovedRepository;
import com.insurance.management.repository.CategoryRepository;
import com.insurance.management.repository.DisapprovedRepository;
import com.insurance.management.repository.PoliciesRepository;
import com.insurance.management.repository.QueriesRepository;
import com.insurance.management.repository.UserRepository;

@SpringBootTest
class ManagementApplicationTests {
	@Autowired
	private ApprovalsRepository approvalsRepository;
	@Autowired
	private ApprovedRepository approvedRepository;
	@Autowired
	private DisapprovedRepository disapprovedRepository;
	@Autowired
	private CategoryRepository categoryRepository;
	@Autowired
	private PoliciesRepository policiesRepository;
	@Autowired
	private QueriesRepository queriesRepository;
	@Autowired
	private UserRepository userRepository;

	@Test
	public void addPolicy() {
		Policies policies = new Policies();
		policies.setPolicyId(47);
		policies.setPolicyName("retirement");
		policies.setCategory("life");
		policies.setAmount(200000.00);
		policies.setTenureInYears(5);
		policiesRepository.save(policies);
		assertNotNull(policiesRepository.findById(47).get());
	}

	@Test
	public void addUser() {
```

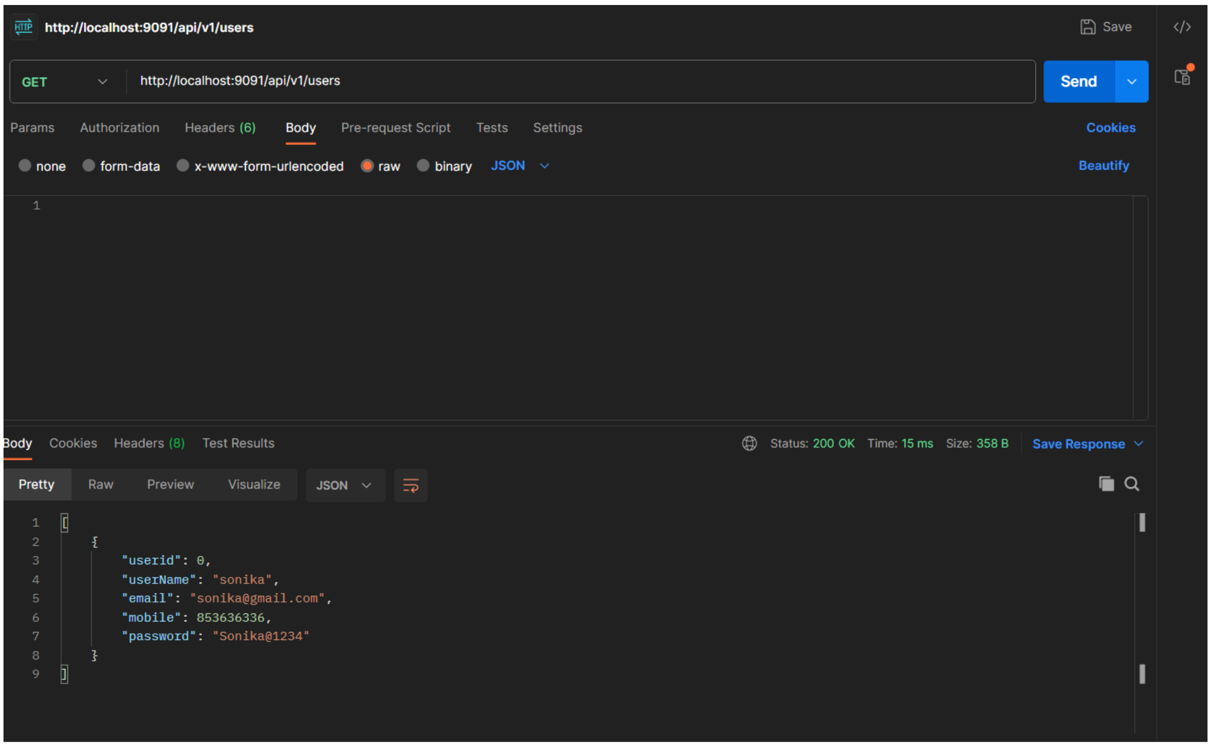
```
                User user = new User();
                user.setUserid(5);
                user.setUserName("sonika");
                user.setMobile(7563425678L);
                user.setPassword("sonika123");
                user.setEmail("sonika@123");
                userRepository.save(user);
                assertNotNull(userRepository.findById("sonika").get());
        }

        @Test
        public void addQueries() {
                Queries queries = new Queries();
                queries.setUserName("sonika");
                queries.setQueryId(5);
                queries.setQuestion("policies");
                queries.setAnswer("insurance");
                queriesRepository.save(queries);
                assertNotNull(queriesRepository.findById("sonika").get());
        }

        @Test
        public void addCategory() {
                Category policycategory = new Category();
                policycategory.setCategory("life");
                categoryRepository.save(policycategory);
                assertNotNull(categoryRepository.findById("life").get());
        }

        @Test
        public void addApprovels() {
                Approvals approvals = new Approvals();
                approvals.setRequestId(8);
                approvals.setPolicyId(10);
                Date date = new Date();
                approvals.setDate(date);
                approvals.setStatus("pending");
                approvals.setUserName("sonika");
                approvalsRepository.save(approvals);
                assertNotNull(approvalsRepository.findById("sonika").get());
        }

        @Test
        public void addApproved() {
                com.insurance.management.model.Approved approved = new
com.insurance.management.model.Approved();
                approved.setPolicyId(5);
                approved.setRequestId(2);
                Date date = new Date();
                approved.setDate(date);
                approved.setStatus("Success");
                approved.setUserName("sonika");
                approvedRepository.save(approved);
                assertNotNull(approvedRepository.findById("sonika").get());

        }

        @Test
        public void adddisApproved() {
                Disapproved disapproved = new Disapproved();
                disapproved.setPolicyId(10);
```

```java
        disapproved.setRequestId(2);
        Date date = new Date();
        disapproved.setDate(date);
        disapproved.setStatus("Success");
        disapproved.setUserName("sonika");
        disapprovedRepository.save(disapproved);
        assertNotNull(disapprovedRepository.findById("sonika").get());

    }

    @Test
    public void policies() {
        List<Policies> list = policiesRepository.findAll();
        assertThat(list).size().isGreaterThan(0);
    }

    @Test
    public void user() {
        List<User> list = userRepository.findAll();
        assertThat(list).size().isGreaterThan(0);
    }

    @Test
    public void AllQueries() {
        List<Queries> list = queriesRepository.findAll();
        assertThat(list).size().isGreaterThan(0);
    }

    @Test
    public void categories() {
        List<Category> list = categoryRepository.findAll();
        assertThat(list).size().isGreaterThan(0);
    }

    @Test
    public void AllApprovals() {
        List<Approvals> list = approvalsRepository.findAll();
        assertThat(list).size().isGreaterThan(0);
    }

    @Test
    public void AllDisapproved() {
        List<Disapproved> list = disapprovedRepository.findAll();
        assertThat(list).size().isGreaterThan(0);
    }

    @Test
    public void AllApproved() {
        List<com.insurance.management.model.Approved> list =
approvedRepository.findAll();
        assertThat(list).size().isGreaterThan(0);
    }

    @Test
    public void getPolicy() {
        Policies policies = policiesRepository.findById(2).get();
        assertEquals(10, policies.getTenureInYears());
    }

    @Test
    public void getUser() {
```

```
		User user = userRepository.findById("sonika").get();
		assertEquals("sonika", user.getUserName());
	}

	@Test
	public void Approval() {
		Approvals approvals =
approvalsRepository.findById("sonika").get();
		assertEquals(8, approvals.getRequestId());
	}

	@Test
	public void getQuery() {
		Queries queries = queriesRepository.findById("sonika").get();
		assertEquals("what", queries.getQuestion());
	}

	@Test
	public void category() {
		Category category = categoryRepository.findById("life").get();
		assertEquals("life", category.getCategory());
	}

	@Test
	public void Approved() {
		com.insurance.management.model.Approved approved =
approvedRepository.findById("sonika").get();
		assertEquals(5, approved.getPolicyId());
	}

	@Test
	public void Disapproved() {
		Disapproved disapproved =
disapprovedRepository.findById("sonika").get();
		assertEquals(10, disapproved.getPolicyId());
	}

	@Test
	public void updatePolicy() {
		Policies policies = policiesRepository.findById(5).get();
		policies.setPolicyName("sonika");
		policies.setCategory("life");
		policies.setAmount(200000.00);
		policies.setTenureInYears(5);
		policiesRepository.save(policies);
		assertNotEquals(500000.00,
policiesRepository.findById(5).get().getAmount());
		assertNotEquals("retirement",
policiesRepository.findById(5).get().getPolicyName());
		assertNotEquals("motor",
policiesRepository.findById(5).get().getCategory());
		assertNotEquals(9,
policiesRepository.findById(5).get().getTenureInYears());
	}

	@Test
	public void updateUser() {
		User user = userRepository.findById("sonika").get();
		user.setUserName("varsh");
		user.setPassword("varsh1234");
		user.setMobile(8888888888L);
		user.setEmail("varsh@123");
		userRepository.save(user);

		assertEquals("sonika",
userRepository.findById("sonika").get().getUserName());
		assertNotEquals(7563425679L,
userRepository.findById("sonika").get().getMobile());
		assertNotEquals("sonika1234",
userRepository.findById("sonika").get().getPassword());
		assertNotEquals("sonika@123",
userRepository.findById("sonika").get().getPassword());
	}

	@Test
	public void updateQuery() {
		Queries queries = queriesRepository.findById("sonika").get();
		queries.setQuestion("what");
		queries.setAnswer("Hi");
		queriesRepository.save(queries);
		assertNotEquals("who",
queriesRepository.findById("sonika").get().getQuestion());
		assertNotEquals("yes",
queriesRepository.findById("sonika").get().getAnswer());
	}

	@Test
	public void deleteCategory() {
		categoryRepository.deleteById("life");
		assertThat(categoryRepository.existsById("life")).isFalse();
	}

}
```

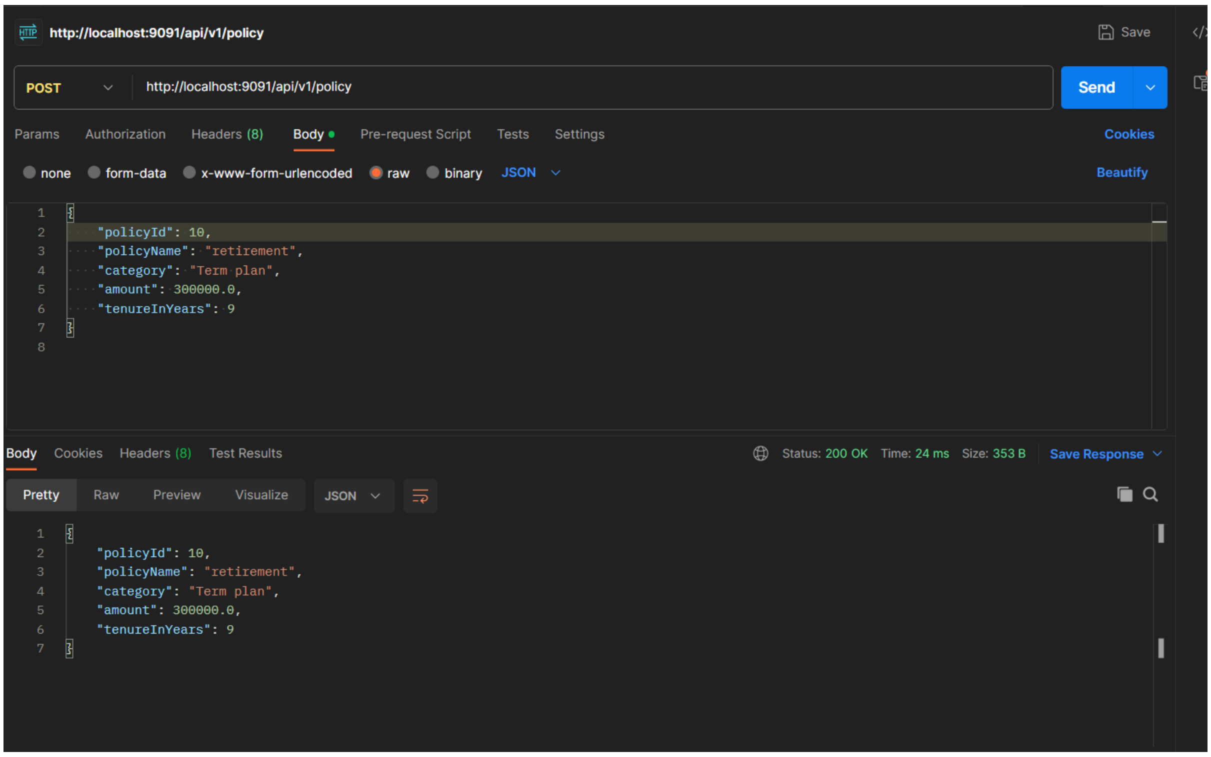
Postman Testing

Get Method



```
[
    {
        "userid": 0,
        "userName": "sonika",
        "email": "sonika@gmail.com",
        "mobile": 853636336,
        "password": "Sonika@1234"
    }
]
```

Post method



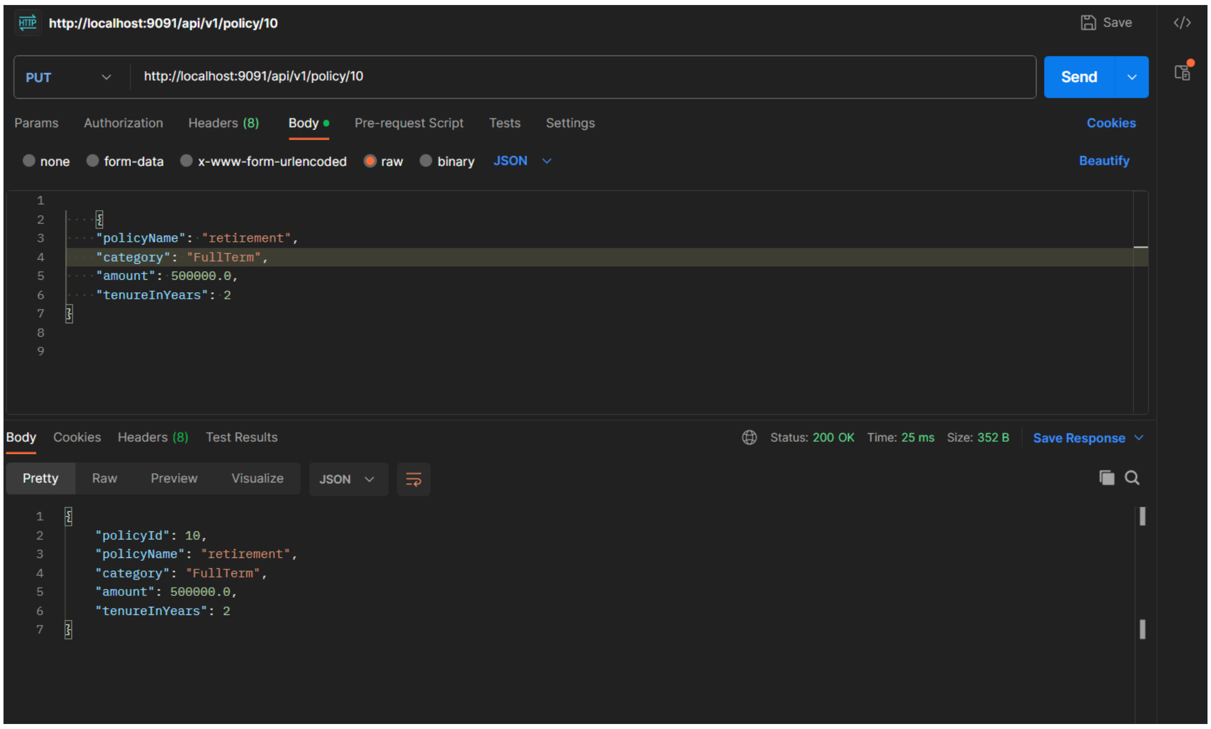
POST http://localhost:9091/api/v1/policy

```
{
    "policyId": 10,
    "policyName": "retirement",
    "category": "Term plan",
    "amount": 300000.0,
    "tenureInYears": 9
}
```

Status: 200 OK Time: 24 ms Size: 353 B

```
{
    "policyId": 10,
    "policyName": "retirement",
    "category": "Term plan",
    "amount": 300000.0,
    "tenureInYears": 9
}
```

Put method



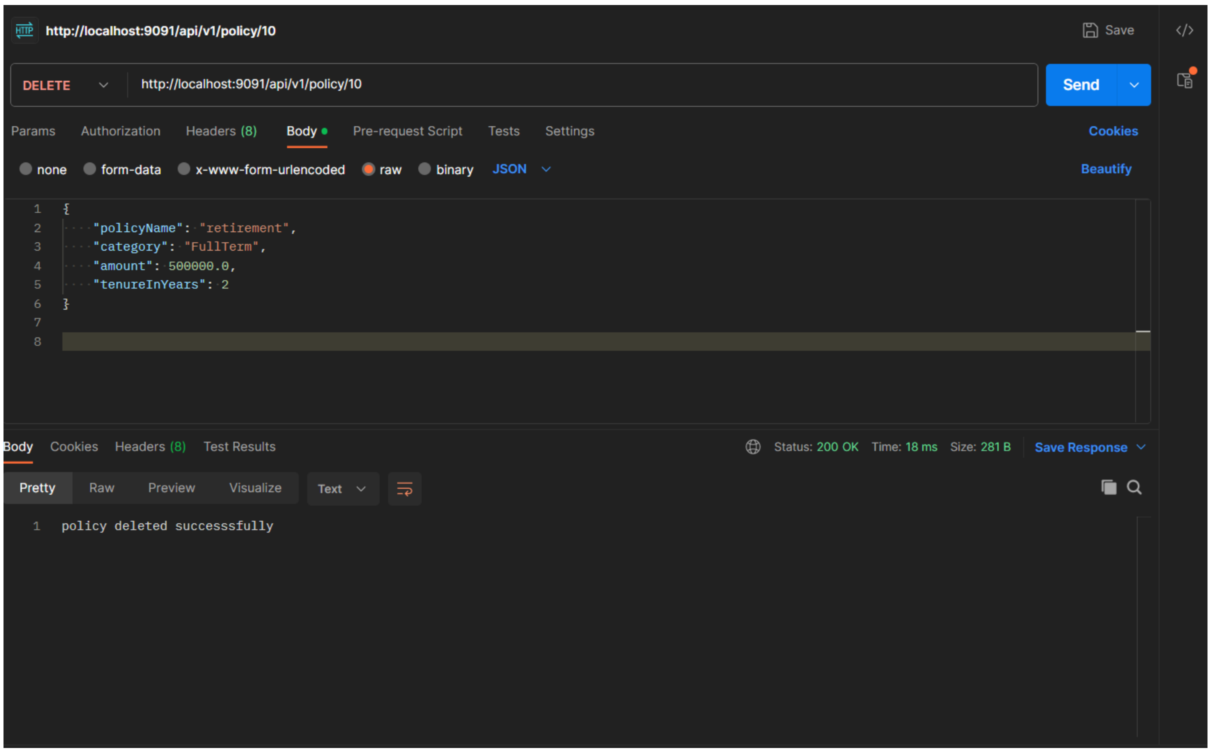
PUT http://localhost:9091/api/v1/policy/10

```

    {
    "policyName": "retirement",
    "category": "FullTerm",
    "amount": 500000.0,
    "tenureInYears": 2
}
```

Status: 200 OK Time: 25 ms Size: 352 B

```
{
    "policyId": 10,
    "policyName": "retirement",
    "category": "FullTerm",
    "amount": 500000.0,
    "tenureInYears": 2
}
```

Delete Method



```
{
    "policyName": "retirement",
    "category": "FullTerm",
    "amount": 500000.0,
    "tenureInYears": 2
}
```

```
policy deleted successsfully
```