

# Lab Exercise 6- CRUD API Testing for Spring Boot Project Using Postman & Newman

---

## 1. Objective

In this lab, you will:

1. Use the provided Spring Boot CRUD API (User API).
  2. Test all endpoints using Postman.
  3. Add automated test scripts in Postman.
  4. Export Postman collection.
  5. Run API tests using Newman.
  6. Generate test reports.
- 

## 2. Prerequisites

Install the following:

1. **Java 17 or 21**
2. **Maven**
3. **Postman (Desktop)**

Download: <https://www.postman.com/downloads>

4. **NodeJS**

Download: <https://nodejs.org>

5. **Newman**
6. **Newman HTML Extra Reporter (optional)**

---

### 3. Install Newman & Tools

#### 3.1 Verify Node Installation

```
node -v
```

```
npm -v
```

#### 3.2 Install Newman

```
npm install -g newman
```

#### 3.3 Install HTML report generator

```
npm install -g newman-reporter-htmlextra
```

---

### 4. Spring Boot API (Your Attached Project)

Your project contains:

**User model**

**UserService (in-memory store)**

**UserController:**

Endpoints:

Method	URL	Description
GET	/api/users	List all users
GET	/api/users/{id}	Get user by ID
POST	/api/users	Create user
PUT	/api/users/{id}	Update user

---

## 5. Start the Spring Boot API

Option 1: Via IDE

Run SimpleapiApplication class.

API base URL:

```
http://localhost:8080/api/users
```

---

## 6. Testing the API in Postman

Open Postman.

Create a new **Collection** named:

```
Simple-Spring-Boot-API-Testing
```

---

### 6.1 Test 1 – List Users (GET)

URL:

```
GET http://localhost:8080/api/users
```

Expected Response: List of two users (Alice and Bob seeded by service).

**Add Tests in Postman → Tests tab:**

```
pm.test("Status code is 200", function () {  
    pm.response.to.have.status(200);  
});  
  
pm.test("Response should be array", function () {  
    pm.expect(pm.response.json()).to.be.an("array");  
});
```

---

## 6.2 Test 2 – Get User by ID (GET)

URL:

```
GET http://localhost:8080/api/users/1
```

**Tests:**

```
pm.test("User found", function () {  
    pm.response.to.have.status(200);  
});  
  
pm.test("User name is Alice", function () {  
    var json = pm.response.json();  
    pm.expect(json.name).to.exist;  
});
```

---

## 6.3 Test 3 – Create User (POST)

URL:

```
POST http://localhost:8080/api/users
```

Body → raw → JSON:

```
{  
    "name": "Hitesh",  
    "email": "hitesh@test.com"  
}
```

**Tests:**

```
pm.test("User created successfully", function () {  
    pm.response.to.have.status(201);  
});  
  
pm.test("Response contains an ID", function () {  
    var json = pm.response.json();  
    pm.expect(json.id).to.exist;  
});
```

---

**6.4 Test 4 – Update User (PUT)**

URL:

```
PUT http://localhost:8080/api/users/1
```

Body:

```
{  
    "name": "Alice Updated",  
    "email": "alice.updated@example.com"  
}
```

**Tests:**

```
pm.test("User updated", function () {  
    pm.response.to.have.status(200);  
});
```

---

---

## 7. Export Postman Collection

1. Click the collection name: “Simple Spring Boot API Testing”
2. Click “Export”
3. Choose **Collection v2.1**
4. Save as:

```
springboot-simpleapi.postman_collection.json
```

---

## 8. Running Automated Tests Using Newman

Navigate to folder where your collection is saved.

Run:

```
newman run springboot-simpleapi.postman_collection.json
```

Expected output:

- All CRUD tests run
  - Summary shown in terminal
- 

## 9. Generate Newman HTML Report

Run:

```
newman run springboot-simpleapi.postman_collection.json -r htmlextra
```

Output report:

```
newman/report.html
```

Open this in any browser.