* **POSTMAN  
  Test API on Postman**

1. Open Postman and create a new request.
2. Set Method: Choose GET, POST, PUT, or DELETE based on the API endpoint.
3. Enter URL: Example for fetching employees:

**1.POST**

URL[: http://localhost:9092/employees/saveEmp](:%20http:/localhost:9092/employees/saveEmp)  
Body:(JSON)

{

  "name": "Supriya Belge",

  "email": "supp@example.com",

  "department": "IT\_dev",

  "salary": 80000,

  "joiningDate": "2024-02-12"  
 }

Add multiple data using above method.

**2.GET**

URL: <http://localhost:9092/employees/showAll>

We get all the data /retrieve the data.

**3.PUT**

URL: <http://localhost:9092/employees/updateEmp/3>

Body:(JSON)

{

  "name": "Supriya Belge",

  "email": "suppbelge@example.com",

  "department": "IT\_dev",

  "salary": 80000,

  "joiningDate": "2024-02-12"  
 }

Update the data using above method.

**4.DELETE**

URL: <http://localhost:9092/employees/deleteById/3>

We delete the data using above method.

* **H2-Console**

<http://localhost:9092/h2-console>

Use password and username provided in **application. Properties**,

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.h2.console.enabled=true

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update

**Easy to Set Up** – No external database required  
**Visual Interface** – Helps in debugging database tables

**Supports SQL Queries**

– Run SELECT, INSERT, UPDATE commands easily

* **GITBASH**

**Initialize git in project**

**Step 1:** Initialize Git in Project

1. Open the terminal in your project directory  
    **cd C:/EmployeeManagementSystem2/EmployeesManagementSystem**
2. Run: This initializes an empty Git repository.  
    **git init**

**Step 2:** Add a. gitignore File (Recommended)

Since your project is a **Spring Boot application**, create a. gitignore file to exclude unnecessary files.

**git add .gitignore**

**Step 3:** Create a New GitHub Repository

1. Go to [GitHub](https://github.com/) and **log in**.
2. Click on **+ (New Repository)** .
3. Give your repository a name (e.g., EmployeeManagementSystem).
4. Click **Create Repository**.

**Step 4: Add Remote Repository and Push Code**

After creating the repository, GitHub will provide you with a remote URL.  
Run these commands in your terminal:  
 # Add all files

**git add** .

# Commit changes

**git commit -m "Initial commit"**

# Add GitHub repository as remote (Replace with your actual repo URL)

**git remote add origin** **https://github.com/Supriy1234/EmployeeManagementSystem.git**

{https://github.com/YOUR\_USERNAME/EmployeeManagementSystem.git}

# Push to GitHub

**git branch -M main  
git push -u origin main**

**Step 5:** Share the Repository Link

Once the push is successful, go to your **GitHub repository page** and **copy the URL**. Share this link with others so they can **clone your project** using:

<https://github.com/Supriy1234/EmployeeManagementSystem.git>

* **Use a Personal Access Token (PAT) for Git Authentication**

Since GitHub no longer supports password authentication, we must use a **Personal Access Token** instead.

**1.Generate a Personal Access Token (PAT)**

1. **Go to GitHub Token Settings**
2. **Click** "Generate new token (classic)".
3. **Set the Expiration Date**:
   * Choose a time limit (e.g., **90 days**)
   * **Avoid "No Expiration" for security reasons**
4. **Select Required Permissions (Scopes)**:
   * ✅ repo → **Full control of private repositories**
   * ✅ workflow → (Optional, for CI/CD workflows)
5. **Click**"Generate Token" and **Copy it Immediately** (you won’t see it again!).

**2️. Use the Token Instead of a Password**

**git push -u origin main**

Git will ask for your GitHub **username** and **password**.

* **For username**: Enter your **GitHub username**
* **For password**: **Paste your PAT instead of your password**

Username: Supriy1234  
Password: Token Generated