USER STORIES

a) Hotel Booking is an online Hotel room booking application that helps the users to book a room for staying at particular place across Karnataka. This application allows users to log in for booking a room. Users can search for the room at a hotel for a specific location. Once found, user can check the availability of a room for specific dates. Users can book a hotel for required date. Once booked, user can get the booking details. Identify and write the user stories for this application.

REGISTRATION

<u>1. SIGN-UP:</u>

As an unauthorized user, I want to sign up for the Hotel Booking application through a sign-up form, so that I can access to book a room.

Acceptance Criteria:

- While signing up-Use Name, Username, Email, and Password and Confirm Password.
- If sign up is successful, it will get automatically logged in.
- If I sign up with an incorrect detail which are specified in step1, I will receive an error message for incorrect information.
- If we are trying to sign up with an existing email address, we will receive an error message saying "email exists."

<u>2. LOGIN</u>

As an authorized user, I want to log in for Hotel Booking application, so that I can have access to the application.

Acceptance Criteria:

- While logging in, Username and password are required.
- After successful log in, it will be redirected to the main page.
- If we are trying to login with incorrect username or password, then error message will be displayed as "invalid login”.

3. SEARCHING A ROOM

As an authorized user, I want to search for a room in Hotel Booking application, so that I can book a room in a specific location.

Acceptance Criteria:

- While searching, Valid location should be specified.
- Checking for a room at specific date always should be current date and ahead of the current date.

4. BOOKING ROOM

As an authorized user, I want to book a room in Hotel Booking application, so that I can reserve the room in a specific location and date.

Acceptance Criteria:

- While Booking, accommodation should be allotted according to the room size.
- One should select the valid payment method based on the price of reserved room.
- After successful payment one should get the booking details to registered mobile number and Email id.

5. Logout

As an authorized user, I want to log out of Hotel Booking application, so that I can prevent unauthorized access of my profile.

Acceptance Criteria:

- When I log out of my account, I will be redirected to the log-in page.

b) Flipkart is an online shopping application that helps its users to buy variety of authentic products. This application allows users to log in for buying products. Users can search for a product, sort the product list based on rating or price. Users can select the items and add them to the cart. Once the selection is done, users can go to the cart page for payment. Identify and write the user stories for this application.

REGISTRATION

<u>1. SIGN-UP:</u>

As a shopper, I want to sign up for the Flipkart application through a New user form, so that I can get access to buy a variety of products.

Acceptance Criteria:

- While signing up-Valid Phone Number/Email Id and OTP/Password.
- If sign up is successful, it will get automatically logged in.
- If I am trying to sign up with an invalid phone number/Email Id, I will receive an error message to enter a valid information.
- If we are trying to sign up with an existing phone number/Email Id, we will receive an error message saying "you are already registered."

<u>2. LOGIN</u>

As an authorized shopper, I want to log in for Flipkart application, so that I can have access to the application for searching and buying products.

Acceptance Criteria:

- While logging in, Phone number/Email Id and OTP/Password are required.
- After successful log in, it will be redirected to the main page.
- If we are trying to login with incorrect mobile number/Email Id or OTP/Password, then error message will be displayed as "invalid credentials”.

3. VIEW A LIST OF PRODUCTS

As a Shopper I want to view a list of products so I can select some items to purchase.

Acceptance Criteria:

- See a thumbnail image for each product
- Click to view details for product
- Add to cart from detail page
- Search for a product
- View products by category

4. REVIEW A CART

As a Shopper I want to review my cart so I can make adjustments prior to checkout

Acceptance Criteria:

- View quantities and items in the cart
- See a total cost before tax and shipping
- Remove items
- Adjust quantity of items
- Click to navigate to a product detail page

5. CHECK OUT

As a Shopper I want to check out so I can get my products shipped to me.

Acceptance Criteria:

- Trigger checkout from any page, if there are items in the cart
- Enter a shipping address
- Enter a billing address
- Enter a credit card number
- Show total including tax and shipping before finalizing.
- Show Confirmation message after finalizing
- Verify payment through our payment processor

6. REVIEW ORDERS

As a Shopper I want to review my orders so I can see what I have purchased in the past.

Acceptance Criteria:

- View a list of open and completed orders
- See the status of the order
- Navigate to the details of the order
- Include a tracking number if the order is shipped but not delivered
- Contact customer service about an order from the details page

7. LOGOUT

As a Shopper, I want to log out of Flipkart application, so that I can prevent unauthorized access of my profile.

Acceptance Criteria:

- When I log out of my account, I will be redirected to the log-in page.

c) Swiggy is an online food ordering application that helps its users to buy variety of authentic food items. This application allows users to log in for ordering food. Users can search for their favourite food based on rating or price. Users can select the items and add to the cart. Once the selection made go to payment page and make payment. write the user stories for this application.

REGISTRATION

<u>1. SIGN-UP:</u>

As a foodie, I want to sign up for Swiggy application through a New user form, so that I can get access to order food of my favourite.

Acceptance Criteria:

- While signing up-Valid Phone Number/Email Id and OTP/Password. If sign up is successful, it will get automatically logged in.
- If I am trying to sign up with an invalid phone number/Email Id, I will receive an error message to enter a valid information.
- If we are trying to sign up with an existing phone number/Email Id, we will receive an error message saying "you are already registered."

<u>2. LOGIN</u>

As an authorized customer, I want to login for application, so that I can have access to the application for searching and ordering food.

Acceptance Criteria:

- While logging in, Phone number/Email Id and OTP/Password are required. After successful log in, it will be redirected to the main page.
- If we are trying to login with incorrect mobile number/Email Id or OTP/Password, then error message will be displayed as "invalid credentials".

3. ORDER CREATION

As a customer, I should be able to browse through the menu and look at various food restaurants and along with their price. As a customer, I should be able to select items from the menu and add them to cart.

As a customer, I should have cart containing all the choosen items.

As a customer, I should be able to remove items from my cart or increase item count.

As a customer, I should be able to cancel my entire order. As a customer, I should be able to view the items bill for my order along with price of each item.

As a customer, I should be able to see the listing of restaurants selling food items.

Acceptance Criteria:

- Categorized menu with prices is visible and enabled with selection choices as soon as the customer chooses items, the order is created in the database and is visible to the customer.
- See a thumbnail image for each product
- Click to view details for product
- Add to cart from detail page
- Search for an item
- View food item by category

4. ORDER COMPLETION

As a customer, I should be able to provide feedback for service and the food.

Acceptance Criteria:

- All the feedbacks are recorded in database for further improvement.

5. LOGOUT

As a customer, I want to log out of application, so that I can prevent unauthorized access of my profile.

Acceptance Criteria:

- When I log out of my account, I will be redirected to the log-in page.

d) Book My Show is an online movie ticket booking application that helps its user to book movie tickets by logging in. Users can find their movie from the listings. After booking is confirm the details are sent to user. Identify and write the user stories for this application.

REGISTRATION

1. SIGN-UP:

As an unauthorized user, I want to sign up for the Book My Show application through a sign-up form, so that I can access to movies list.

Acceptance Criteria:

- While signing up-Use Name, Username, Email, and Password and Confirm Password.
- If sign up is successful, it will get automatically logged in.
- If I sign up with an incorrect detail which are specified in step1, I will receive an error Message for incorrect information.
- If we are trying to sign up with an existing email address, we will receive an error Message saying "email exists."

2. LOGIN:

As an authorized user, I want to log in for Book My Show application, so that I can have access to the application.

Acceptance Criteria:

- While logging in, Username and password are required.
- After successful log in, it will be redirected to the main page.
- If we are trying to login with incorrect username or password, then error message will be displayed as "invalid login".

3. SEARCHING A MOVIE

As an authorized user, I want to search for a movie in Book My Show application, so that I can book a movie ticket in a specific theater.

Acceptance Criteria:

- While searching, Valid theater should be specified.
- Checking for availability of a movie ticket on specific date always should be current date and ahead of the current date.

4. BOOKING TICKET

As an authorized user, I want to book a ticket in Book My Show application, so that I can reserve the seat in a specific theater and date.

Acceptance Criteria:

- While Booking, accommodation should be allotted according to the room size.
- One should select the valid payment method based on the price of reserved room
- After successful payment one should get the booking details to registered mobile Number and E-mail id.

5. LOGOUT

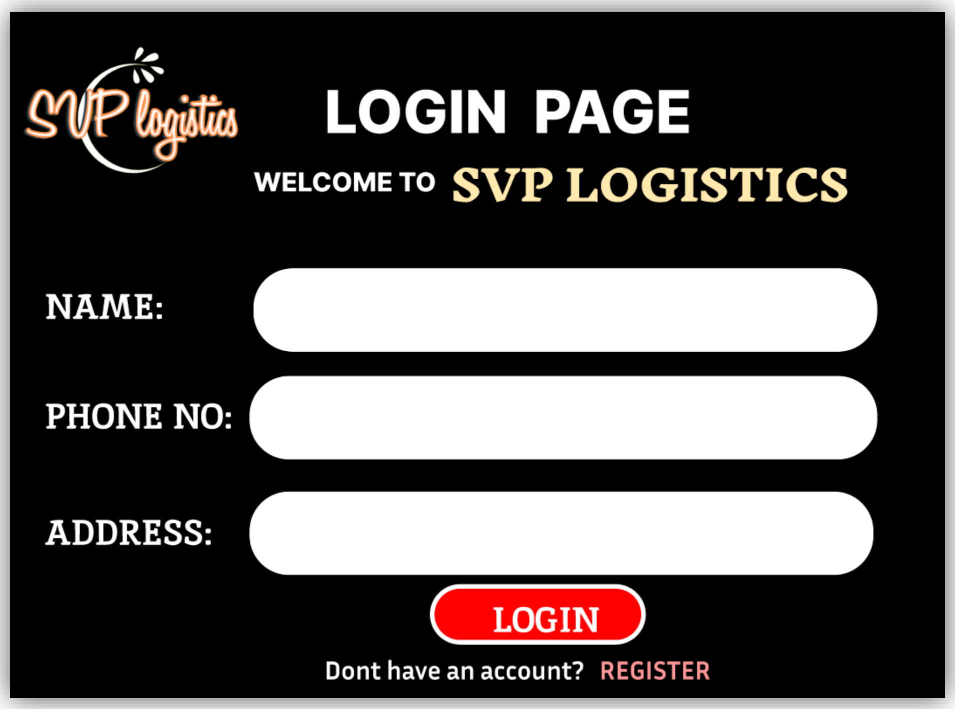
As an authorized user, I want to log out of application, so that I can prevent unauthorized access of my profile.

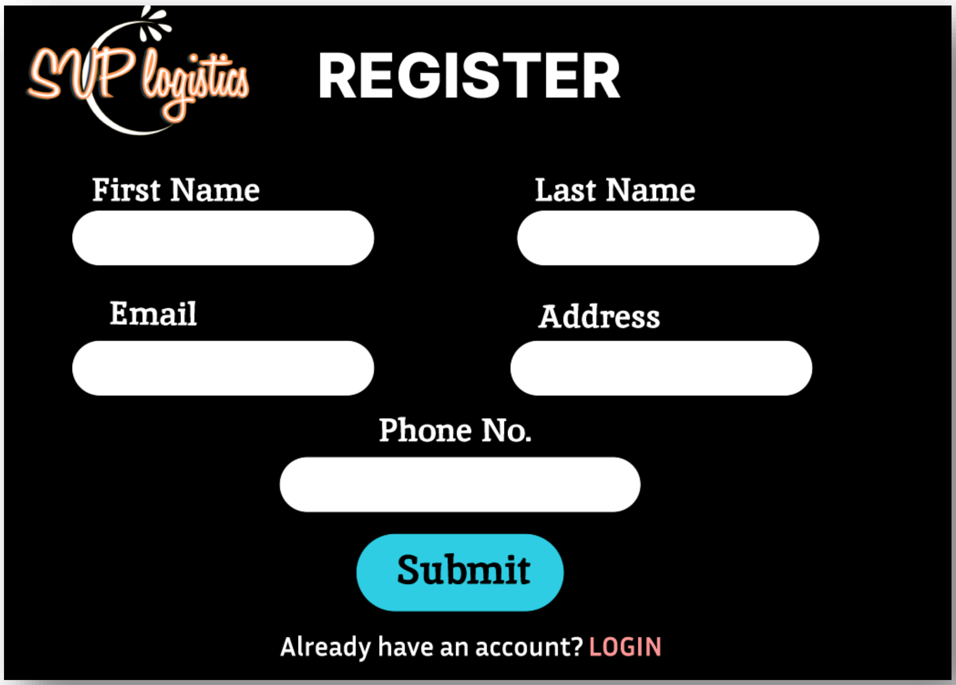
Acceptance Criteria:

- When I log out of my account, I will be redirected to the log-in page.

EXPERIMENT –   Date :

DESIGN A LOGIN PAGE AND REGISTRATION PAGE IN FIGMA





CREATE REPOSITORY NAMED MINI PROJECT 1 & PUSH THE SAME TO GITHUB

How to install Git on your computer?

1. Windows: For Windows users, you can download and install Git using the Git for Windows package, also known as Git Bash.

- Visit the official Git website: https://git-scm.com/
- Download the Git for Windows installer.
- Run the installer executable file.
- Follow the installation wizard's instructions. You can generally choose the default options unless you have specific preferences.
- During installation, you'll be prompted to select components. Make sure to include the Git Bash terminal emulator.
- Complete the installation process.

2. Linux (Ubuntu as an example): You can use the package manager to install Git on Linux distributions. Here's how to do it on Ubuntu:

- Open Terminal.
- Update your package list:

  sudo apt update
- Install Git:

  sudo apt install git

3. Verify Installation: After installation, you can verify that Git is installed by opening a terminal/command prompt and typing:

```
git --version
```

This should display the installed version of Git.

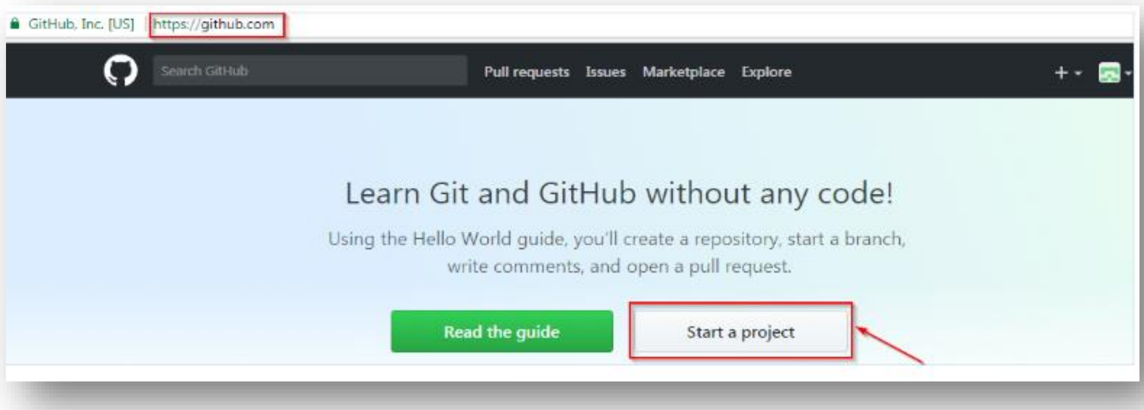
HOW TO CREATE A GITHUB REPOSITORY NAMED MINIPROJECT-1?

A repository is a storage space where your project lives. It can be local to a folder on your computer, or it can be a storage space on GitHub or another online host. You can keep code files, text files, images or any kind of a file in a repository.

Steps to create a GitHub repository:

- Go to the link: https://github.com/. Fill the sign up form and click on "Sign up for Github".
- Click on "Start a new project".

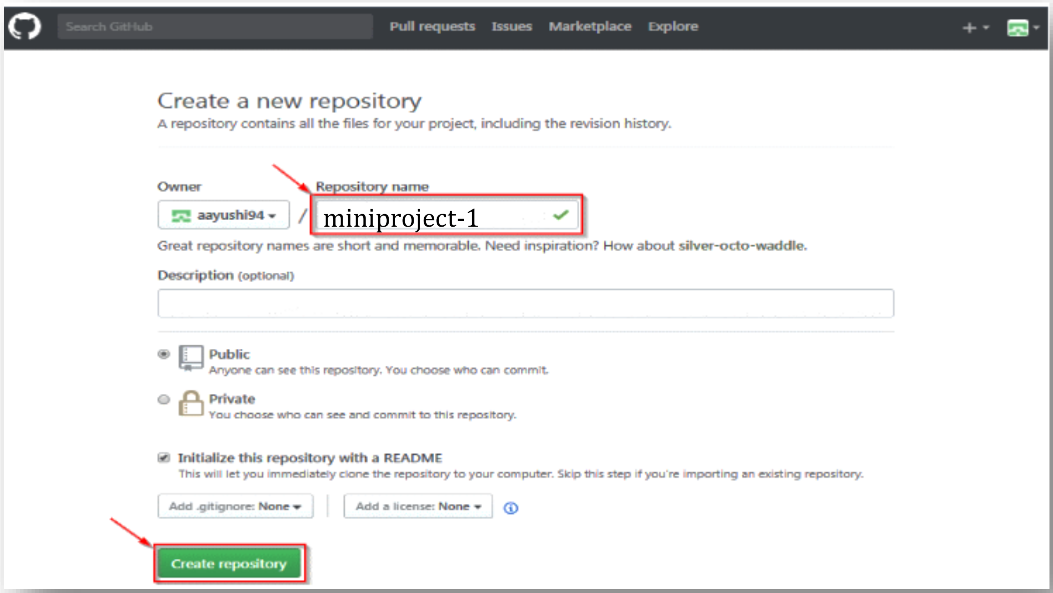
Refer to the below screenshot to get a better understanding.



Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

- Enter any repository name and click on "Create Repository". You can also give a description to your repository (optional).



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: aayushi94 / Repository name: miniproject-1

Great repository names are short and memorable. Need inspiration? How about silver-octo-waddle.

Description (optional)

● Public — Anyone can see this repository. You choose who can commit.

○ Private — You choose who can see and commit to this repository.

✓ Initialize this repository with a README — This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Create repository

- Now, if you noticed by default a GitHub repository is public which means that
- anyone can view the contents of this repository.
- Private repository is a paid version.

Create folder Select git bash/windows powershell

Change directory to folder

git init will create a new local GIT repository.

- **git init**

git config can be used to set user-specific configuration values like email, username, file format, and so on.

- **git config --global user.name yourname**
- **git config --global user.email youremail@example.com**

git status displays the list of changed files together with the files that are yet to be staged or committed.

- **git status**

To connect the local repository to a remote server, use the command below

- **git remote add origin <host-or-remoteURL>**
- **Ex:- git remote add origin https://github.com/Navya-Hebbar/Logistics_Website.git**

Meanwhile, the following command will delete a connection to a specified remote repository:

- g**it remote rm <name-of-the-repository>**

git add is used to add files to the staging area. For example, the basic Git following command will index the temp.txt file:

- **git add <temp.txt>**

git rm can be used to remove files from the index and the working directory.

- **git rm filename.txt**

git commit will create a snapshot of the changes and save it to the git directory.

- **git commit –m "Message to go with the commit here"**

git push is used to send local commits to the master branch of the remote repository. Here's the basic code structure:

- **git push origin <master>**

git clone is used to copy a repository. If the repository lies on a remote server, use:

- **git clone username@host:/path/to/repository**

Conversely, run the following basic command to copy a local repository:

- **git clone /path/to/repository**
- **Ex :- git clone https://github.com/Navya-Hebbar/Logistics_Website.git**

git pull merges all the changes present in the remote repository to the local working directory.

- **git pull –v repository-link**
- **Ex :- git pull -v https://github.com/Navya-Hebbar/Logistics_Website.git**

git show is a command used to view information about any git object.

- **git show**

git log is used to see the repository's history by listing certain commit's details. Running the command will get you an output that looks like this:

commit

- 15f4b6c44b3c8344caasdac9e4be13246e21sadw

  Author: Alex Hunter <alexh@gmail.com>;

  Date: Mon Oct 1 12:56:29 2016 -0600

git reset command will reset the index and the working directory to the last git commit's state.

- **git reset --hard HEAD**

git branch will list, create, or delete branches. For instance, if you want to list all the branches present in the repository, the command should look like this:

- **git branch new_branchname**

git checkout creates branches and helps you to navigate between them. For example, the following basic command creates a new branch and automatically switches you to it:
command

- **git checkout -b <branch-name>**

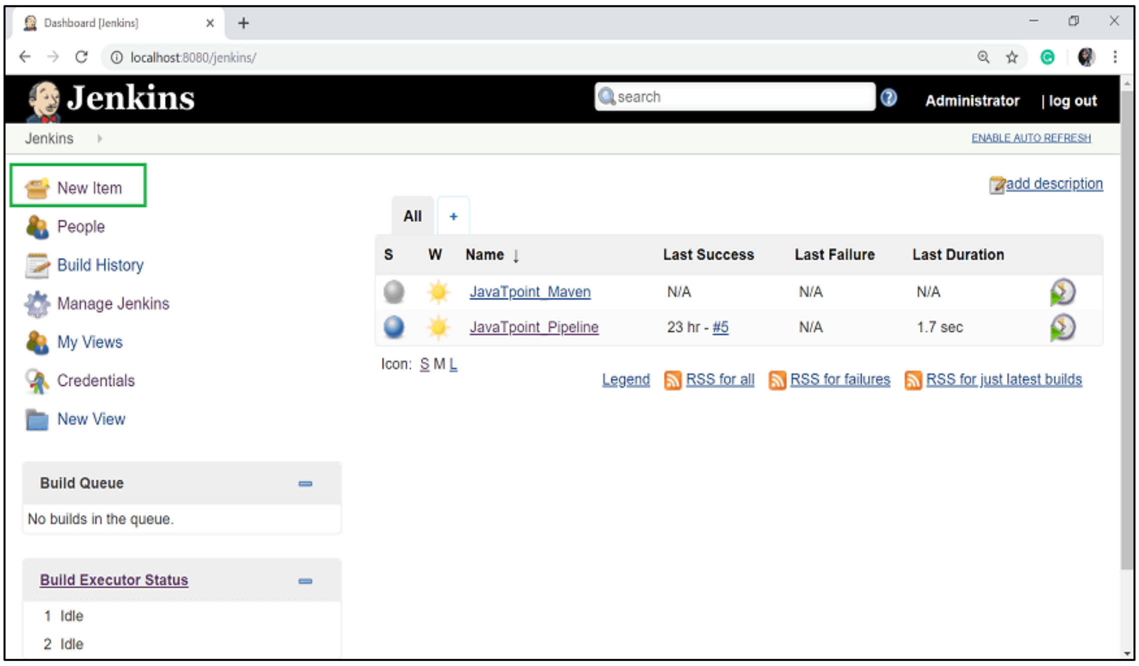
git merge is used to merge a branch into the active one.

- **git merge <branch-name>**

USE JENKINS WHICH IS A CI/CD TOOL TO BUILD AND TEST CODE BY INTEGRATING IT WITH GITHUB

Jenkins - Setup Build Jobs

Step 1: Go to the Jenkins dashboard and click on the New Item.

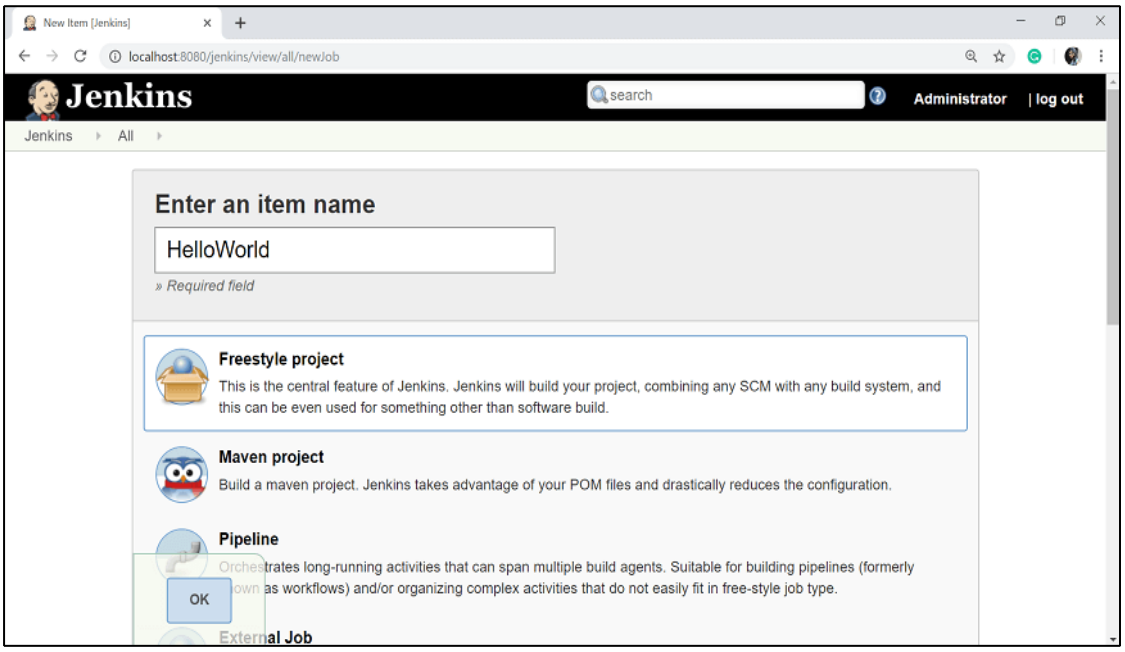


S	W	Name ↓	Last Success	Last Failure	Last Duration
		JavaTpoint_Maven	N/A	N/A	N/A
		JavaTpoint_Pipeline	23 hr - #5	N/A	1.7 sec

No builds in the queue.

1 Idle
2 Idle

Step 2: In the next page, enter the item name, and select the 'Freestyle project' option. And click OK. Here, my item name is HelloWorld.



Enter an item name

HelloWorld

» Required field

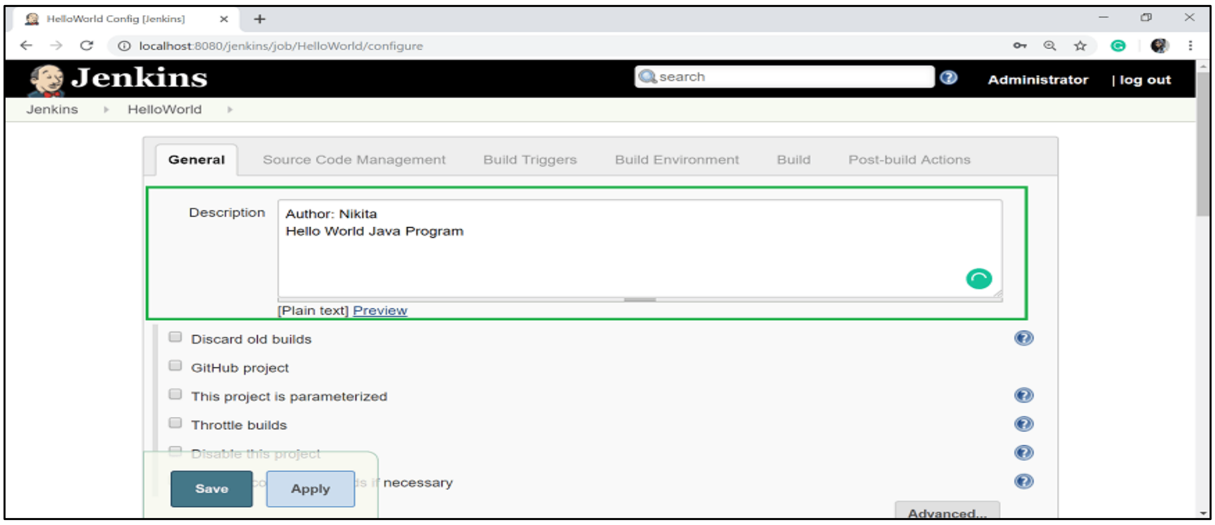
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

External Job

Step 3: When you enter the OK, you will get a configuration page. Enter the details of the project in the Description section.



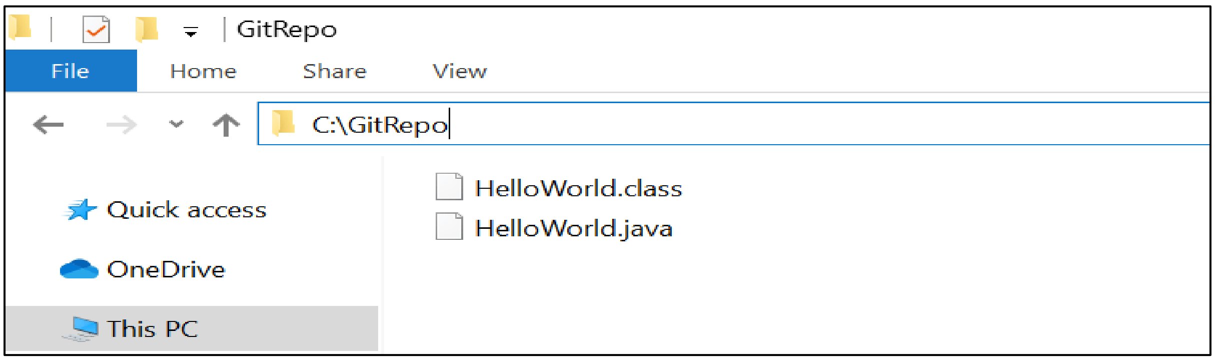
Description: Author: Nikita
Hello World Java Program
[Plain text] Preview

- Discard old builds
- GitHub project
- This project is parameterized
- Throttle builds
- Disable this project
- necessary

Step 4: On the Source Code Management section, select the **Git** option, and specify the Repository URL.

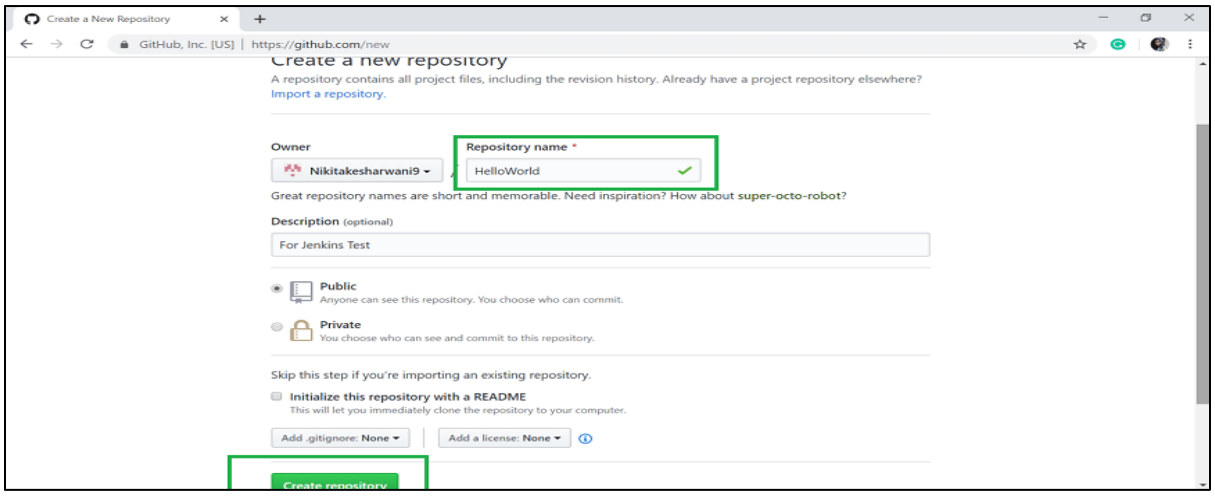
To do that you should have proper github setup on your system. To do the github setup:

- First, you have to create a project in java. Here, I created a simple **HelloWorld** program and saved it to one folder i.e. C:\GitRepo. Compile the HelloWorld.java file.



HelloWorld.class
HelloWorld.java

- Now create a project in your GitHub account and give the Repository name. Here my repository name is HelloWorld.



Create a New Repository

GitHub, Inc. [US] | https://github.com/new

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: Nikitakesharwani9 / Repository name *: HelloWorld

Great repository names are short and memorable. Need inspiration? How about super-octo-robot?

Description (optional): For Jenkins Test

● Public — Anyone can see this repository. You choose who can commit.

○ Private — You choose who can see and commit to this repository.

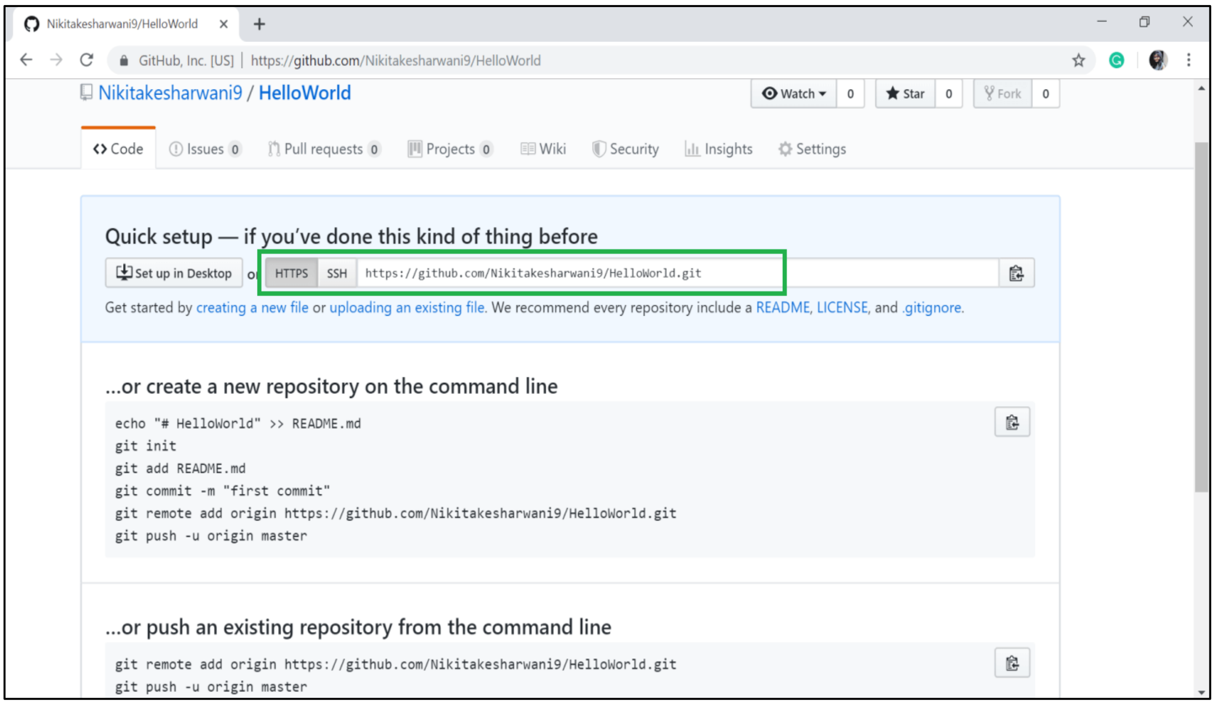
Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README — This will let you immediately clone the repository to your computer.

Add .gitignore: None | Add a license: None

Create repository

- Click on **Create repository**.



Nikitakesharwani9/HelloWorld

GitHub, Inc. [US] | https://github.com/Nikitakesharwani9/HelloWorld

Nikitakesharwani9 / HelloWorld

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security | Insights | Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/Nikitakesharwani9/HelloWorld.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

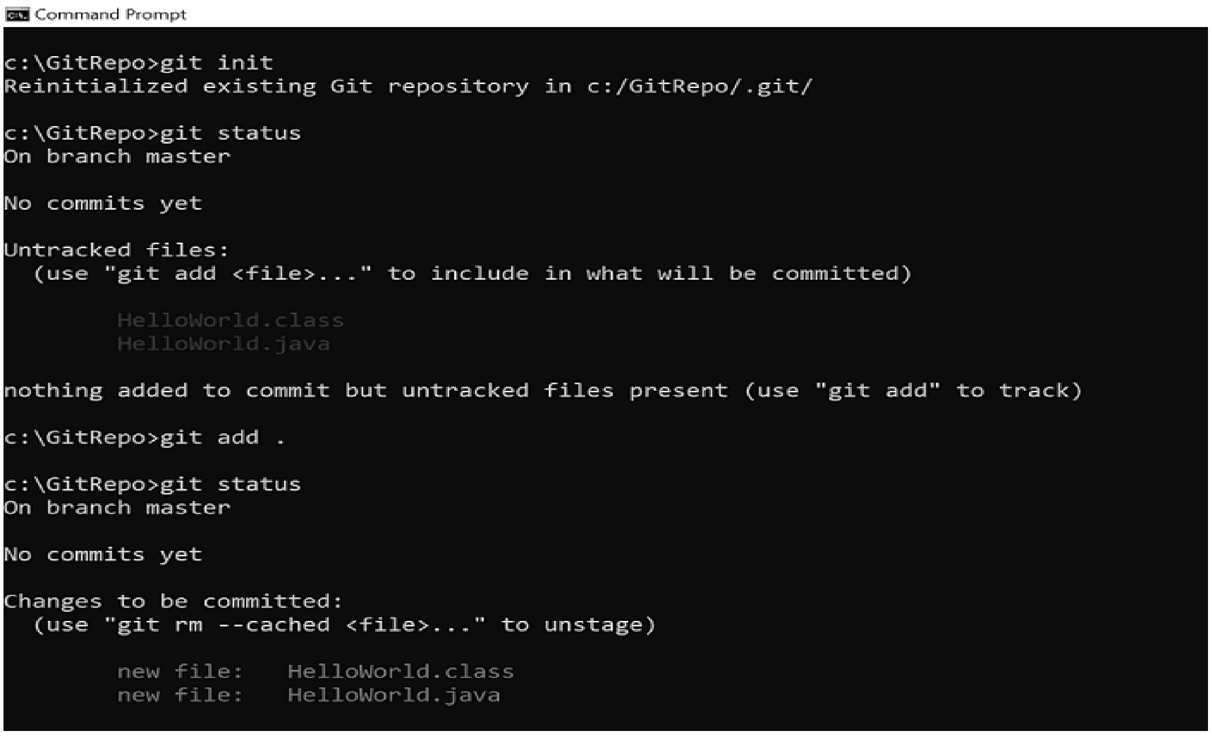
```
echo "# HelloWorld" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/Nikitakesharwani9/HelloWorld.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/Nikitakesharwani9/HelloWorld.git
git push -u origin master
```

- Your repository is created. Copy the repository URL. My repository URL is: https://github.com/dineshbalgi/Jenkins.git

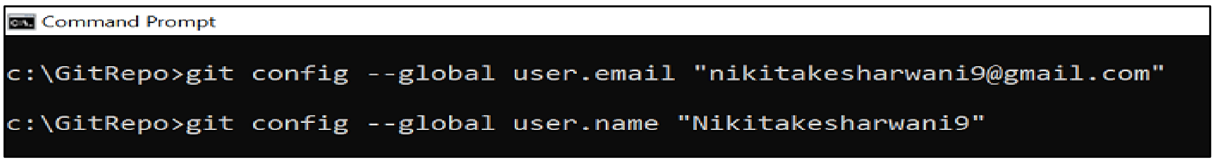
- Open the command prompt in your Windows and go to the path where your java file is created.
- Then run the following command.
  1. git init
  2. git status
  3. git add .
  4. git status



```
c:\GitRepo>git init
Reinitialized existing Git repository in c:/GitRepo/.git/

c:\GitRepo>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        HelloWorld.class
        HelloWorld.java

nothing added to commit but untracked files present (use "git add" to track)

c:\GitRepo>git add .

c:\GitRepo>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   HelloWorld.class
        new file:   HelloWorld.java
```

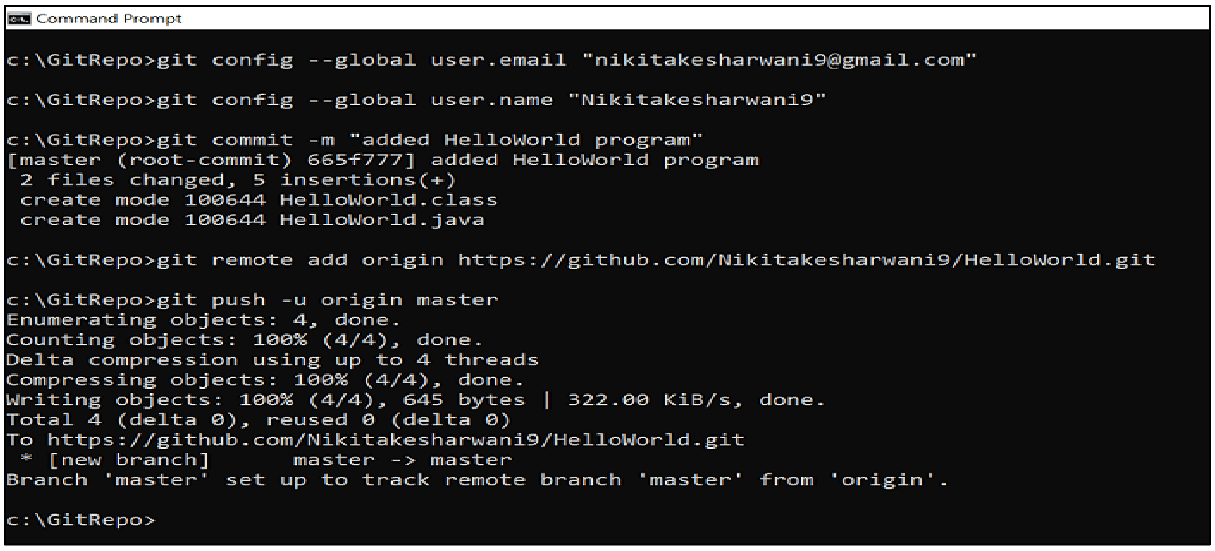
- Configure your GitHub account in your system.

1. git config --global user.email "your@email"
2. git config --global user.name "username"



```
c:\GitRepo>git config --global user.email "nikitakesharwani9@gmail.com"

c:\GitRepo>git config --global user.name "Nikitakesharwani9"
```

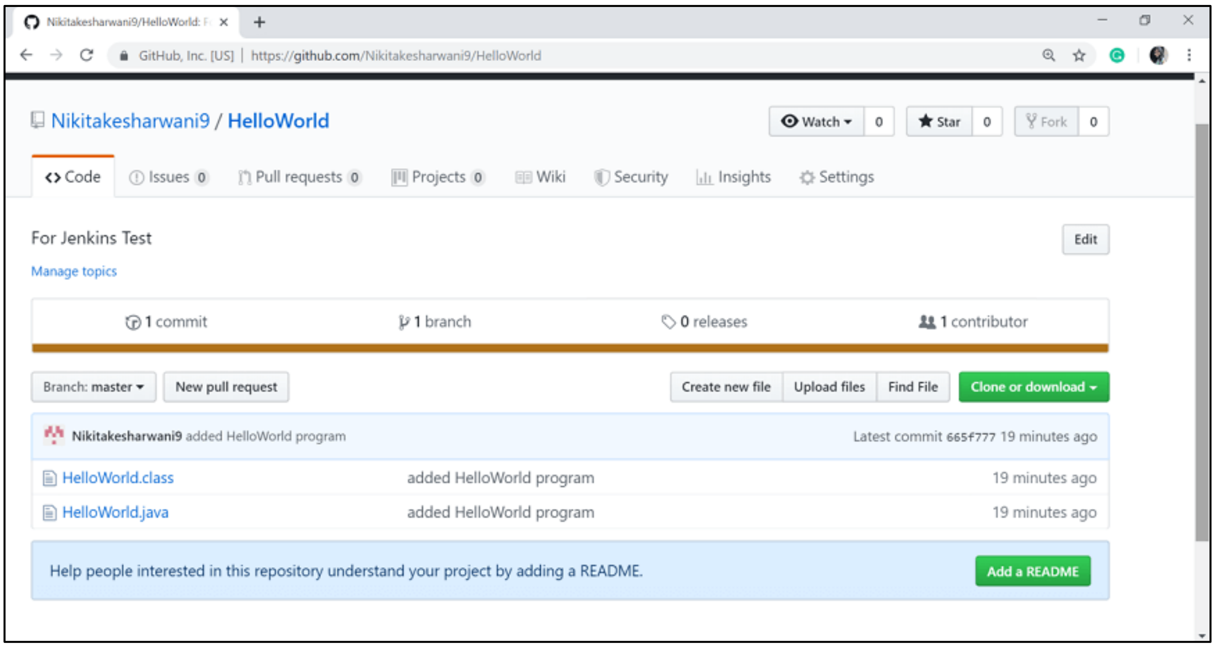
- Commit it and add the repository URL.
  1. git commit -m "added HelloWorld program"
  2. git remote add origin https://github.com/Nikitakesharwani9/HelloWorld.git
  3. git push -u origin master



Command Prompt

```
c:\GitRepo>git config --global user.email "nikitakesharwani9@gmail.com"

c:\GitRepo>git config --global user.name "Nikitakesharwani9"

c:\GitRepo>git commit -m "added HelloWorld program"
[master (root-commit) 665f777] added HelloWorld program
 2 files changed, 5 insertions(+)
 create mode 100644 HelloWorld.class
 create mode 100644 HelloWorld.java

c:\GitRepo>git remote add origin https://github.com/Nikitakesharwani9/HelloWorld.git

c:\GitRepo>git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 645 bytes | 322.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/Nikitakesharwani9/HelloWorld.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

c:\GitRepo>
```

- Now, when you refresh your GitHub account, the helloWorld file will be added in your repository.



Nikitakesharwani9 / HelloWorld

For Jenkins Test

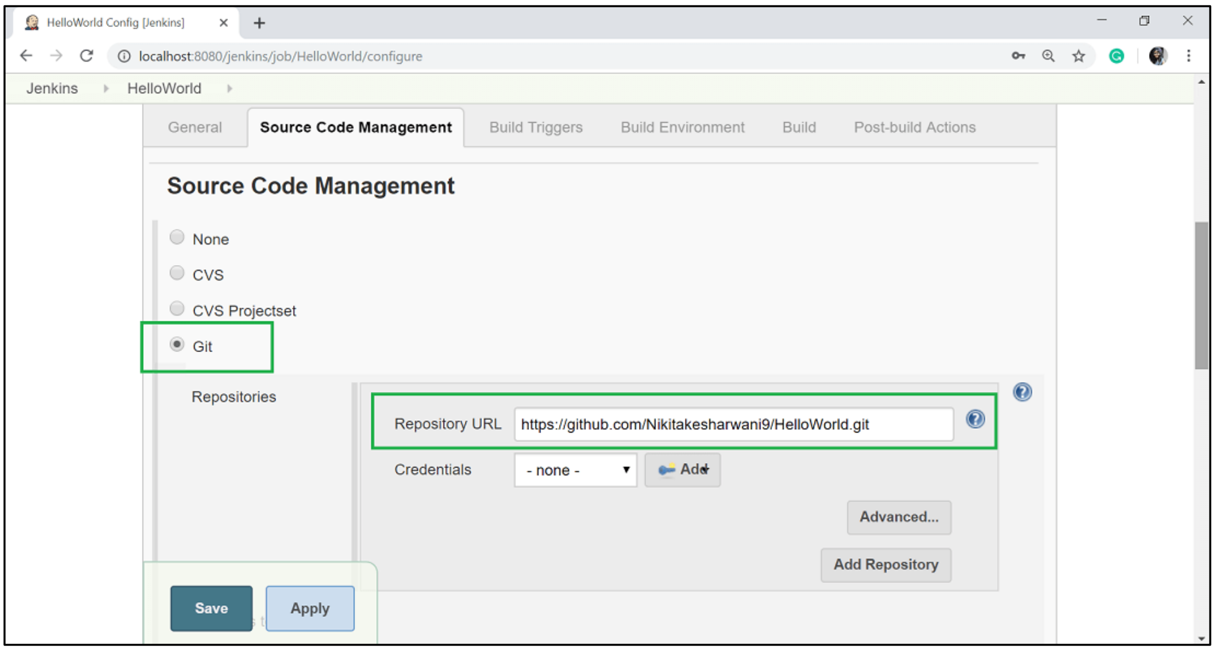
1 commit | 1 branch | 0 releases | 1 contributor

Nikitakesharwani9 added HelloWorld program — Latest commit 665f777 19 minutes ago

HelloWorld.class	added HelloWorld program	19 minutes ago
HelloWorld.java	added HelloWorld program	19 minutes ago

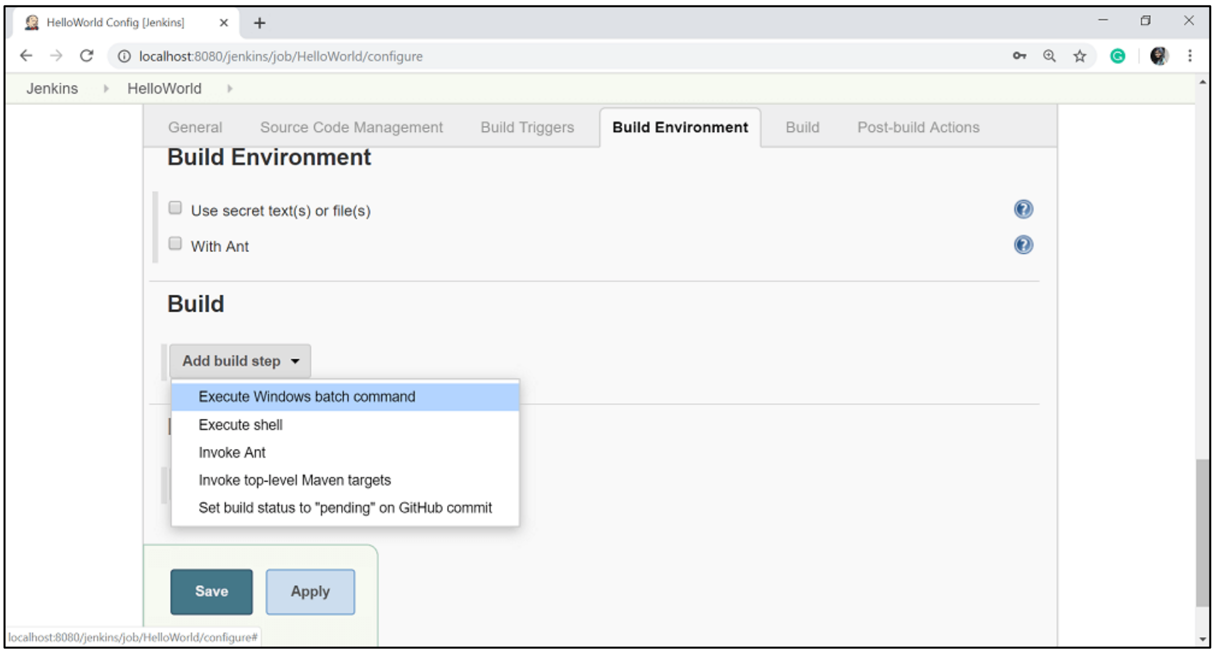
Help people interested in this repository understand your project by adding a README.

Step 5: Add the Repository URL in the **Source Code Management** section.



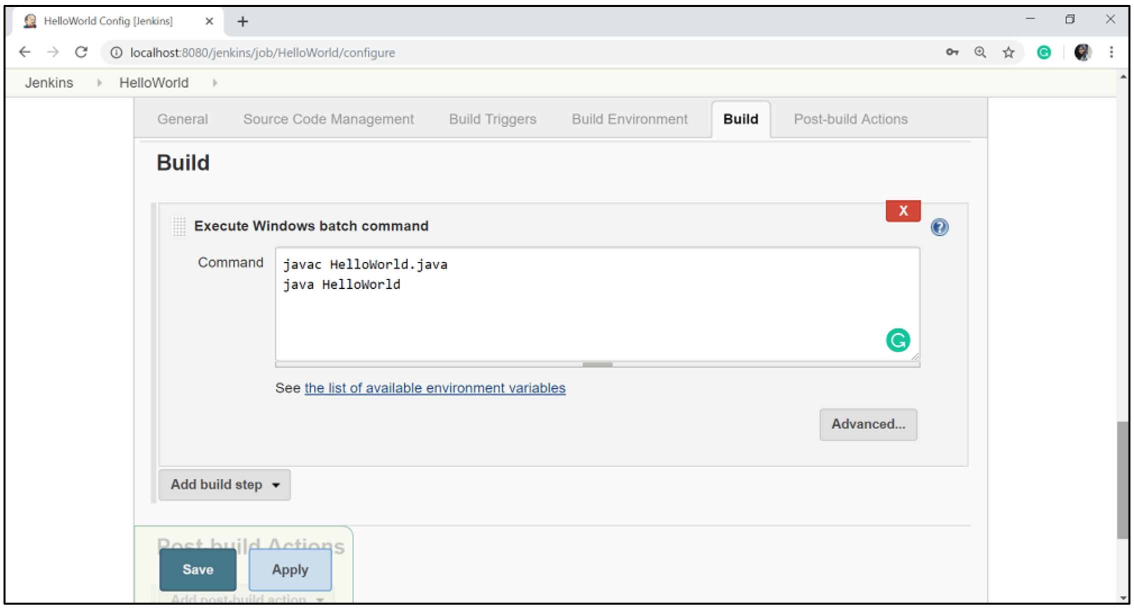
You can also use a local repository. And if your GitHub repository is private, Jenkins will first validate your login credentials with GitHub and only then access the source code from your GitHub repository.

Step 6: Now, it is time to build the code. Click on "**Add build step**" and select the "**Execute Windows batch command**".


HelloWorld Config [Jenkins]
localhost:8080/jenkins/job/HelloWorld/configure
Jenkins
HelloWorld
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build Environment
Use secret text(s) or file(s)
With Ant
Build
Add build step
Execute Windows batch command
Execute shell
Invoke Ant
Invoke top-level Maven targets
Set build status to "pending" on GitHub commit
Save
Apply
localhost:8080/jenkins/job/HelloWorld/configure#

Step 7: Enter the following command to compile the java code.

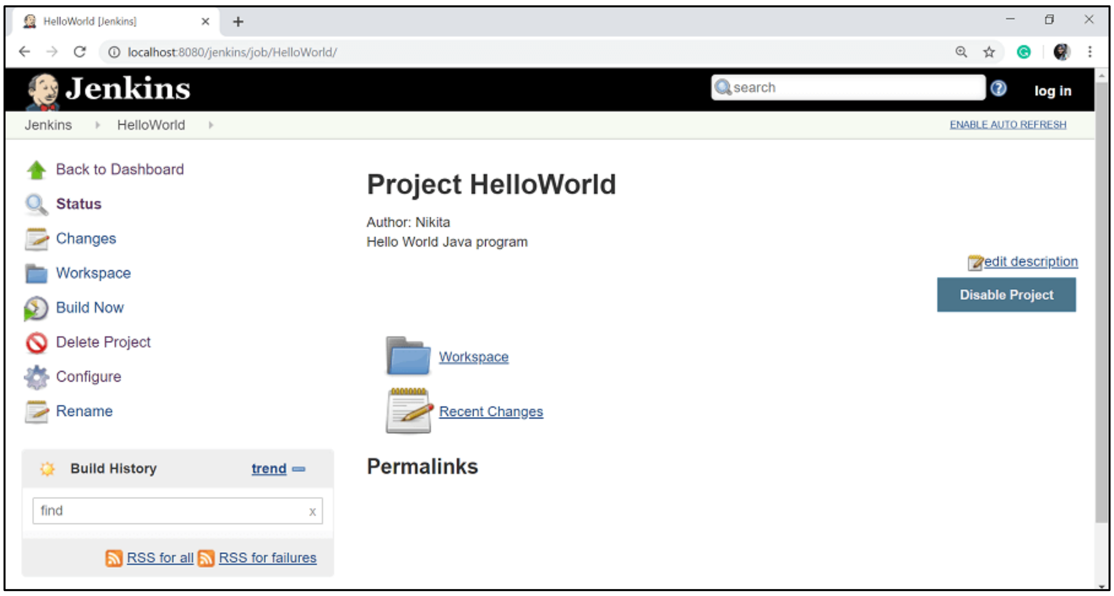
1. javac HelloWorld.java
2. java HelloWorld



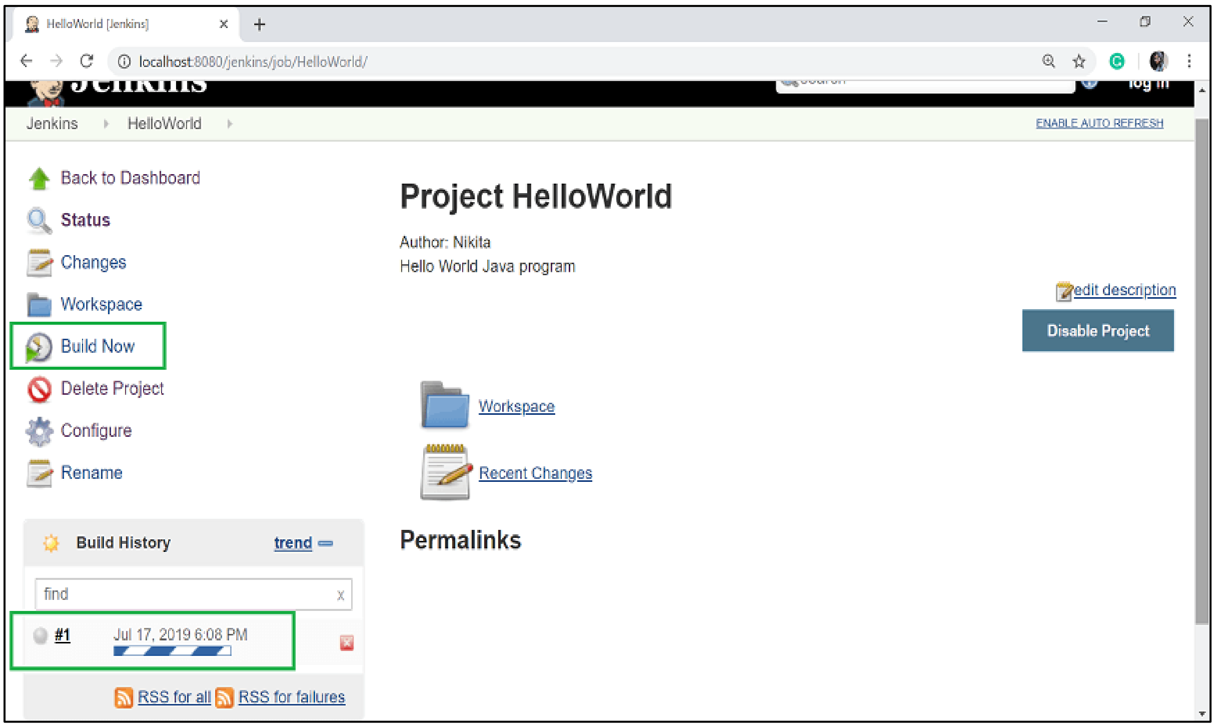
```
javac HelloWorld.java
java HelloWorld
```

Step 8: Click Apply and then Save button.

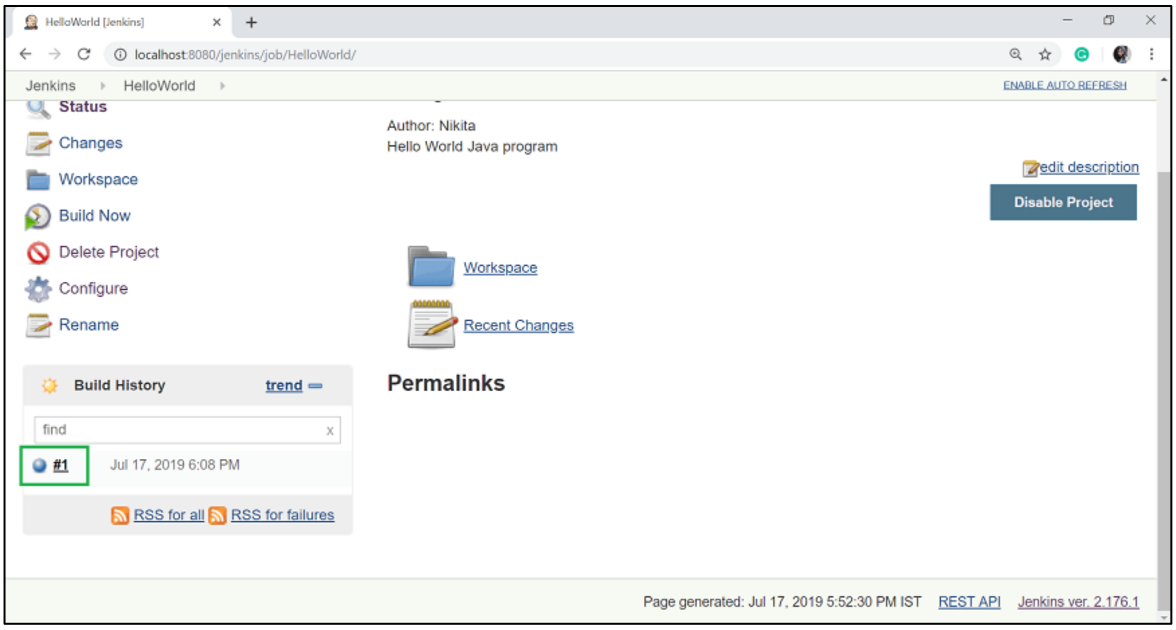
Step 9: Once you saved the configuration, then now can click on **Build Now** option.



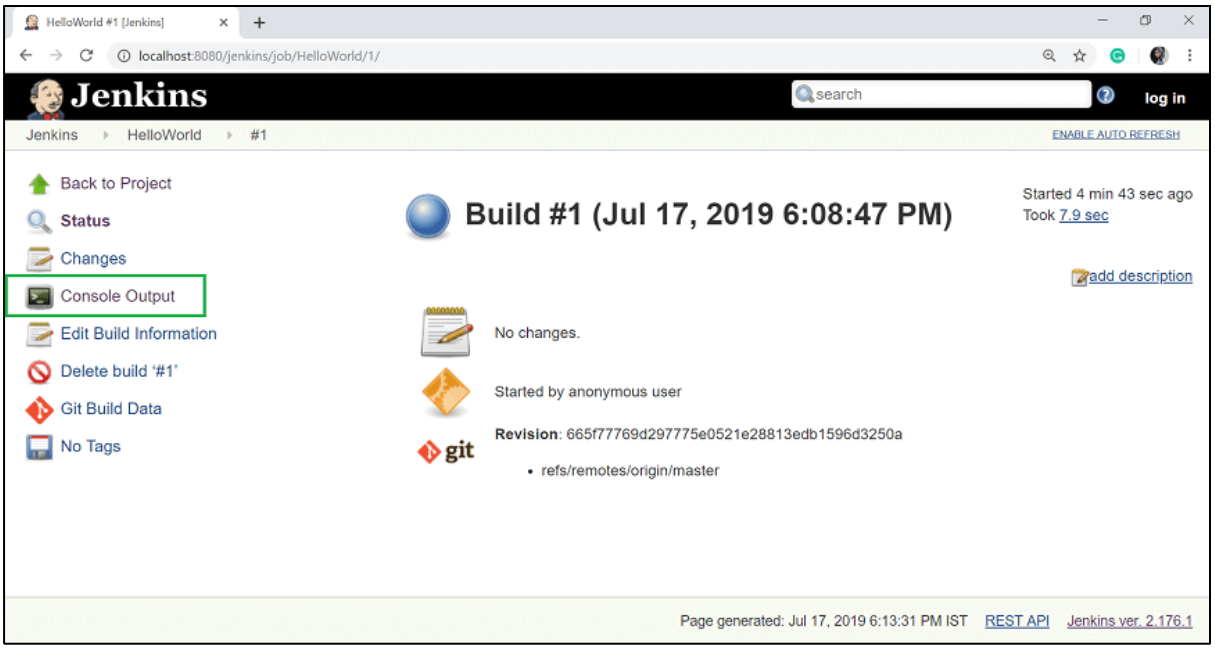
Step 10: After clicking on **Build Now**, you can see the status of the build on the Build History section.



Once the build is completed, a status of the build will show if the build was successful or not. If the build is failed then it will show in red color. Blue symbol is for success.



Click on the build number **#1** in the **Build History section** to see the details of the build.



HelloWorld #1 [Jenkins]

localhost:8080/jenkins/job/HelloWorld/1/

Jenkins › HelloWorld › #1

Back to Project
Status
Changes
Console Output
Edit Build Information
Delete build '#1'
Git Build Data
No Tags

Build #1 (Jul 17, 2019 6:08:47 PM)

Started 4 min 43 sec ago
Took 7.9 sec

add description

No changes.

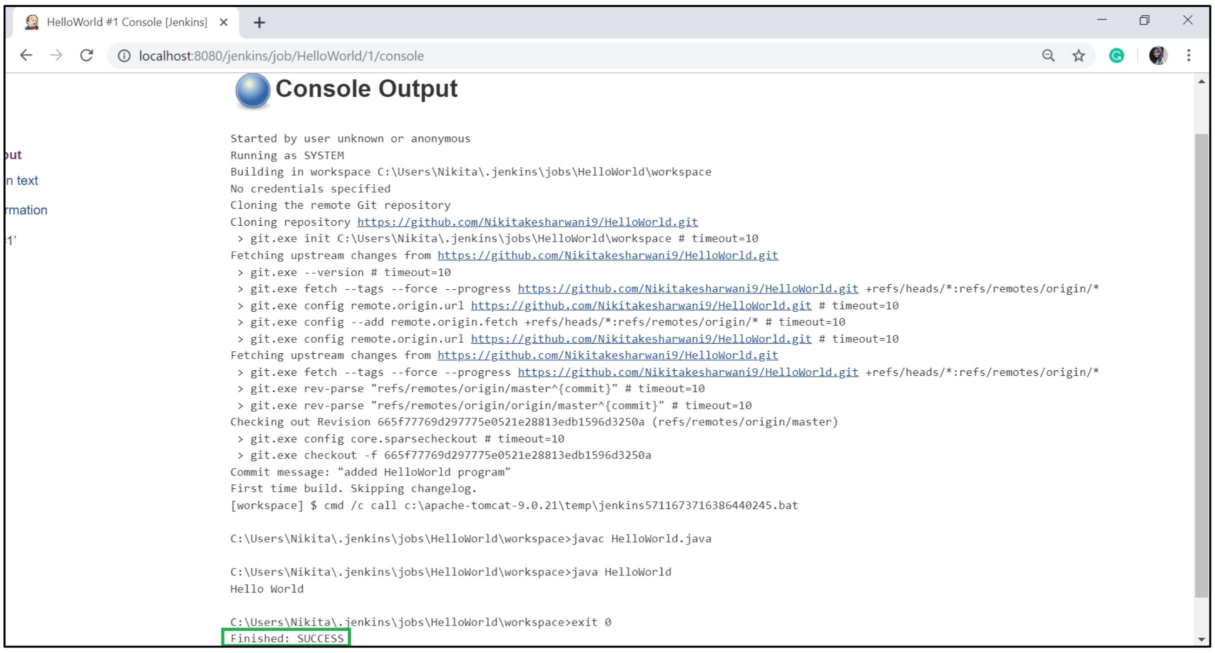
Started by anonymous user

Revision: 665f77769d297775e0521e28813edb1596d3250a

- refs/remotes/origin/master

Page generated: Jul 17, 2019 6:13:31 PM IST REST API Jenkins ver. 2.176.1

Step 11: Click on **Console Output** from the left side of the screen to see the status of the build you run. It should show the success message.



HelloWorld #1 Console [Jenkins]

localhost:8080/jenkins/job/HelloWorld/1/console

Console Output

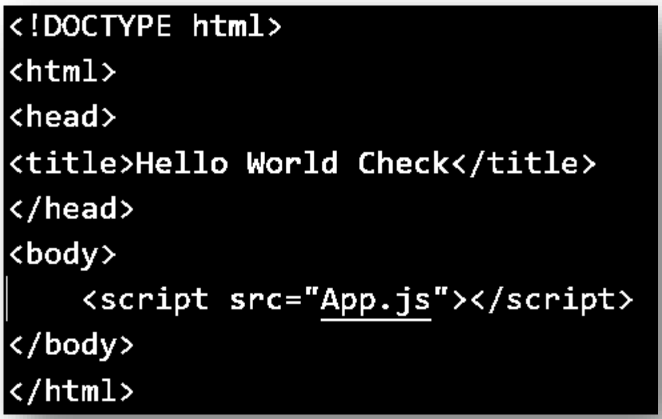
```
Started by user unknown or anonymous
Running as SYSTEM
Building in workspace C:\Users\Nikita\.jenkins\jobs\HelloWorld\workspace
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/Nikitakesharwani9/HelloWorld.git
 > git.exe init C:\Users\Nikita\.jenkins\jobs\HelloWorld\workspace # timeout=10
Fetching upstream changes from https://github.com/Nikitakesharwani9/HelloWorld.git
 > git.exe --version # timeout=10
 > git.exe fetch --tags --force --progress https://github.com/Nikitakesharwani9/HelloWorld.git +refs/heads/*:refs/remotes/origin/*
 > git.exe config remote.origin.url https://github.com/Nikitakesharwani9/HelloWorld.git # timeout=10
 > git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe config remote.origin.url https://github.com/Nikitakesharwani9/HelloWorld.git # timeout=10
Fetching upstream changes from https://github.com/Nikitakesharwani9/HelloWorld.git
 > git.exe fetch --tags --force --progress https://github.com/Nikitakesharwani9/HelloWorld.git +refs/heads/*:refs/remotes/origin/*
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
 > git.exe rev-parse "refs/remotes/origin/origin/master^{commit}" # timeout=10
Checking out Revision 665f77769d297775e0521e28813edb1596d3250a (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 665f77769d297775e0521e28813edb1596d3250a
Commit message: "added HelloWorld program"
First time build. Skipping changelog.
[workspace] $ cmd /c call c:\apache-tomcat-9.0.21\temp\jenkins5711673716386440245.bat

C:\Users\Nikita\.jenkins\jobs\HelloWorld\workspace>javac HelloWorld.java

C:\Users\Nikita\.jenkins\jobs\HelloWorld\workspace>java HelloWorld
Hello World

C:\Users\Nikita\.jenkins\jobs\HelloWorld\workspace>exit 0
Finished: SUCCESS
```

EXPERIMENT –   Date :

TYPESCRIPT "HELLO WORLD" PROGRAM IN WEB BROWSER

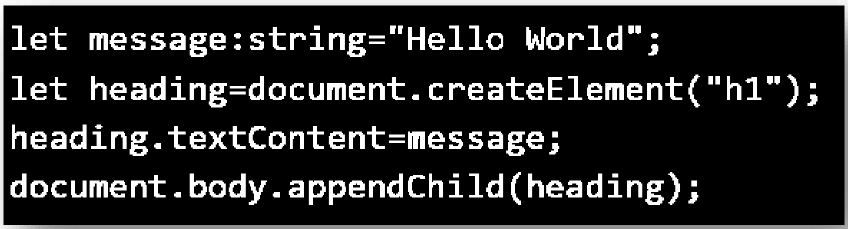
Follow these steps to create a web page that shows Hello World message on web browser

- Create a new HTML file called index.html and include script tag mentioned below



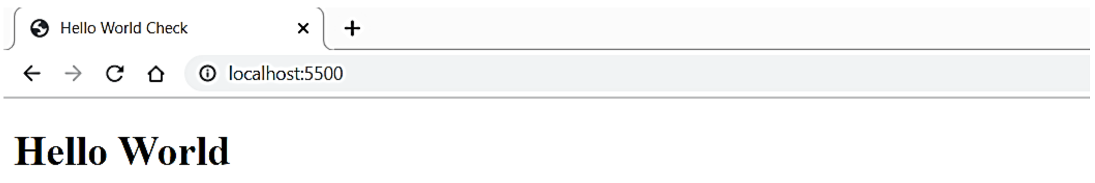
```
<!DOCTYPE html>
<html>
<head>
<title>Hello World Check</title>
</head>
<body>
    <script src="App.js"></script>
</body>
</html>
```

- Create App.ts file and add below code



```
let message:string="Hello World";
let heading=document.createElement("h1");
heading.textContent=message;
document.body.appendChild(heading);
```

- Compile the App.ts file using below command in the terminal
    - ➢ **tsc App.ts**
- If tsc command doesn't work install npm typescript module using below command in terminal
    - ➢ **npm install -g typescript**
- Typescript compiler will generate a new App.js file
- Open index.html file and load it using live server

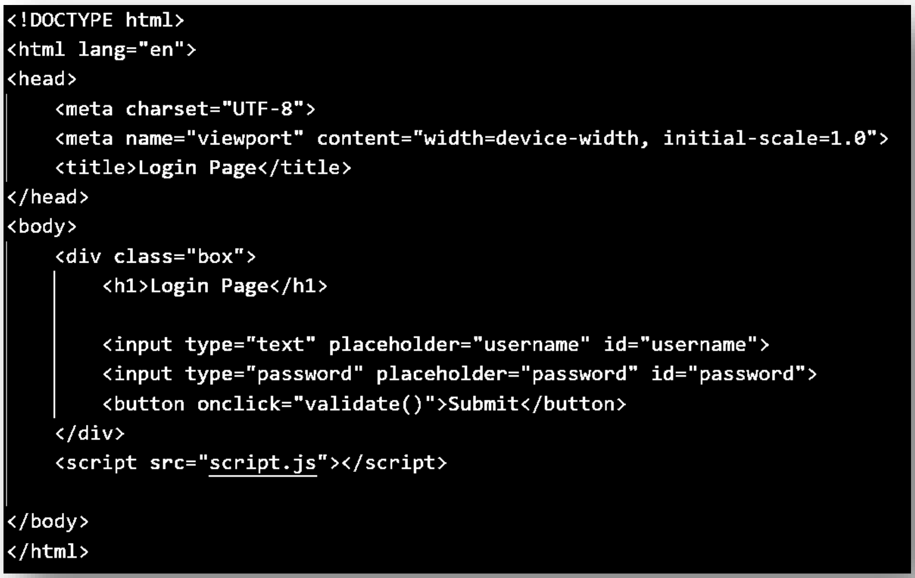


```
Hello World
```

VALIDATION FORM – JAVASCRIPT

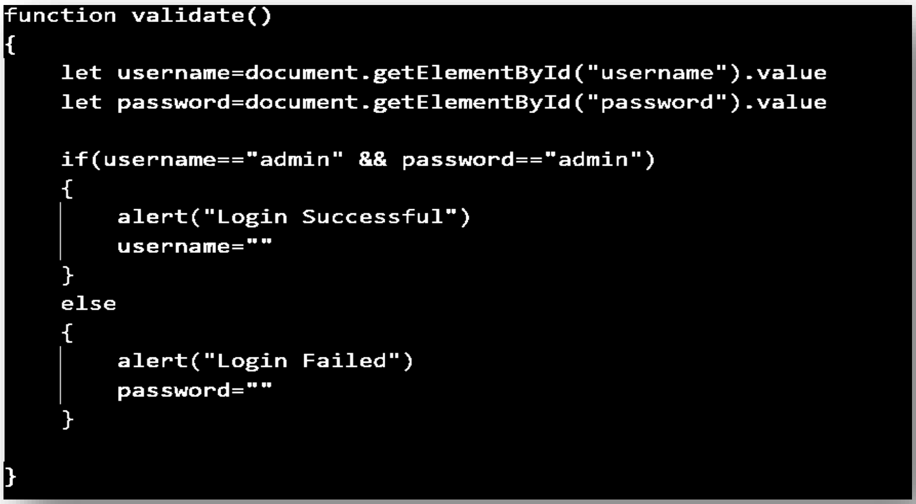
VALIDATING USER’S CREDENTIALS USING JAVASCRIPT

- Create index.html file and write the below code



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login Page</title>
</head>
<body>
    <div class="box">
        <h1>Login Page</h1>

        <input type="text" placeholder="username" id="username">
        <input type="password" placeholder="password" id="password">
        <button onclick="validate()">Submit</button>
    </div>
    <script src="script.js"></script>

</body>
</html>
```

- Create script.js file and write the below code:



```
function validate()
{
    let username=document.getElementById("username").value
    let password=document.getElementById("password").value

    if(username=="admin" && password=="admin")
    {
        alert("Login Successful")
        username=""
    }
    else
    {
        alert("Login Failed")
        password=""
    }

}
```

- Open index.html and load it using live server
- On the web browser enter credentials & click on submit and check whether login successful message is displayed

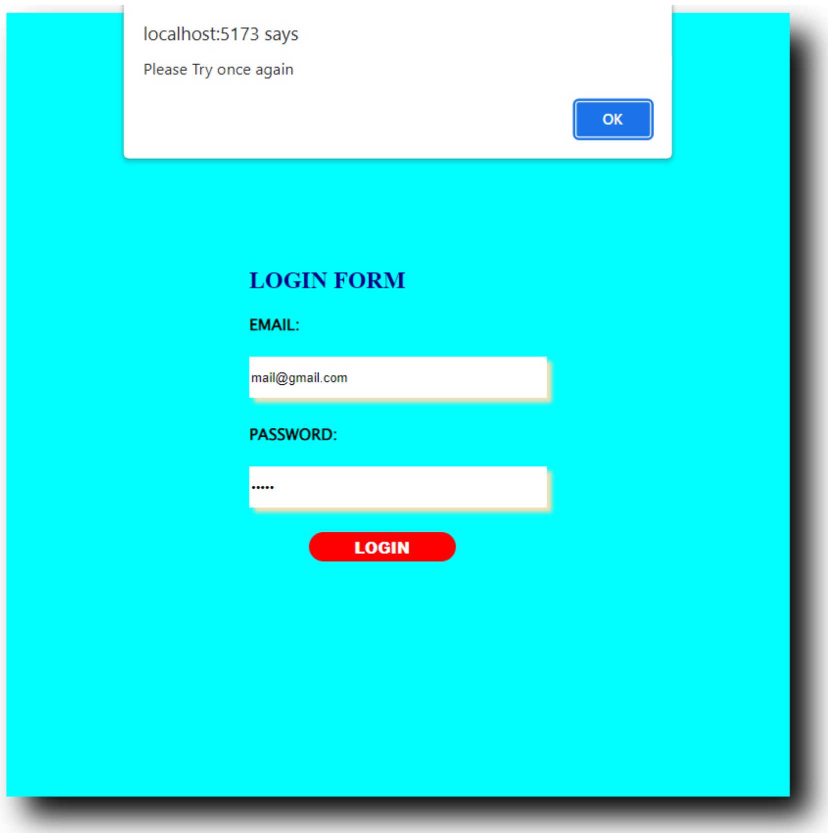
DESIGN A LOGIN FORM USING REACT JS

Login.js

```
import React,{useState} from 'react';

import './Login.css'

const Login=()=>

{

   const [email,setEmail]=useState('')

   const [password,setPassword]=useState('')

   const handleSubmit=(e)=>

{

      e.preventDefault();

      if(!email || !password)

      {alert("Enter email and password"); return}

      if(email==="svpkumta@gmail.com" && password==="admin")

      {

         alert("Login Successful");

       }

      else

      {

         alert("Please Try once again")

       }

   }
return(
      <>
      <div className='login'>
      <form onSubmit={handleSubmit}>
      <h2 style={{fontWeight:'bolder',fontFamily:'serif',color:'darkblue'}}>
       LOGIN FORM
     </h2>

      EMAIL:<br/><input type="email" name="email" placeholder="Enter your
      Email id" onChange={(e)=>setEmail(e.target.value)}/><br/>

      PASSWORD: <br/><input type="password" name="password"
     placeholder="Enter your Password"  onChange={(e)=>
      setPassword(e.target.value)}/>

      <br/><button onClick={handleSubmit}>LOGIN</button>
      </form>
      </div>
      </>
   )
}
export default Login;
```

Login.css

```css
body{

    display: flex;
    justify-content: center;
    align-items: center;
    }

.login
{
    box-shadow: 20px 20px 30px black;
    width: 800px;
    height: 800px;
    background-color:aqua ;
    display: flex;
    align-items: center;
    justify-content: center;
    font-weight: bold;
    font-family:'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida
    Sans Unicode', Geneva, Verdana, sans-serif;
}

input
{
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 300px;
    height: 40px;
    margin-top:20px;
    border: none;
    box-shadow: 5px 5px 4px wheat;
}
button
{
    border-radius: 50px;
    border: none;
    background-color: red;
    color: white;
    font-weight: bolder;
    font-size: medium;
    height: 30px;
    width: 150px;
    margin-left: 20%;
    cursor: pointer;
}
```



localhost:5173 says

Please Try once again

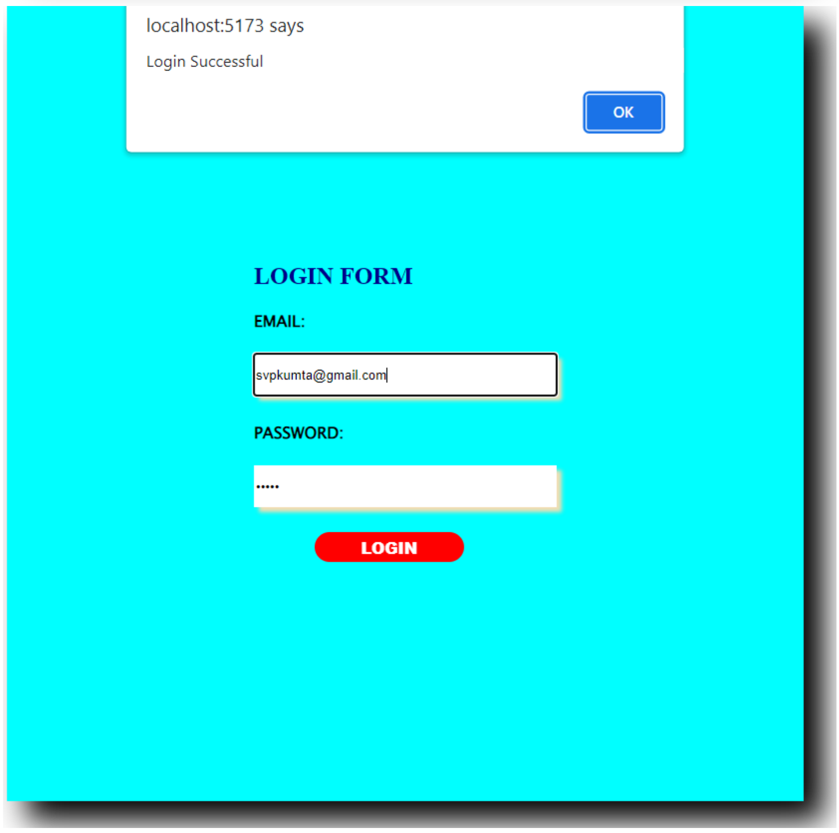
LOGIN FORM

EMAIL:

mail@gmail.com

PASSWORD:

•••••



localhost:5173 says

Login Successful

LOGIN FORM

EMAIL:

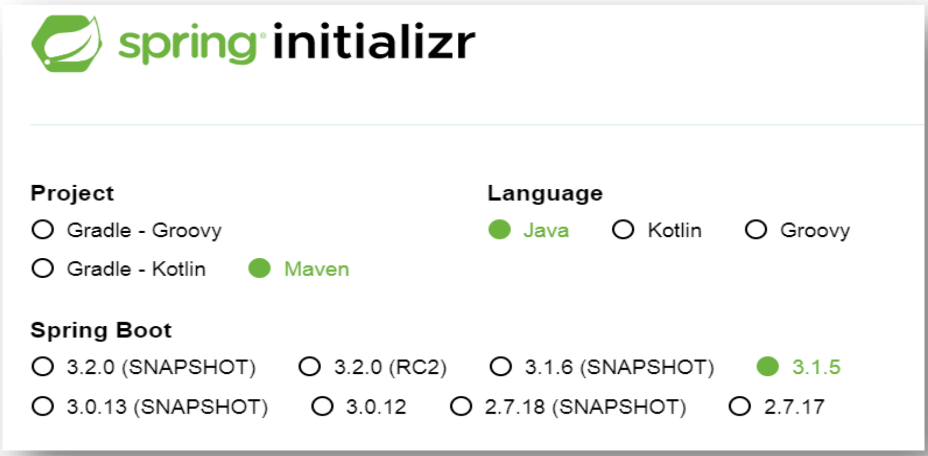
svpkumta@gmail.com

PASSWORD:

•••••

CREATE A SPRING BOOT APPLICATION USING SPRING BOOT API TO MAINTAIN THE DETAILS OF THE BOOK

- Go to https://start.spring.io/
- Select Maven project

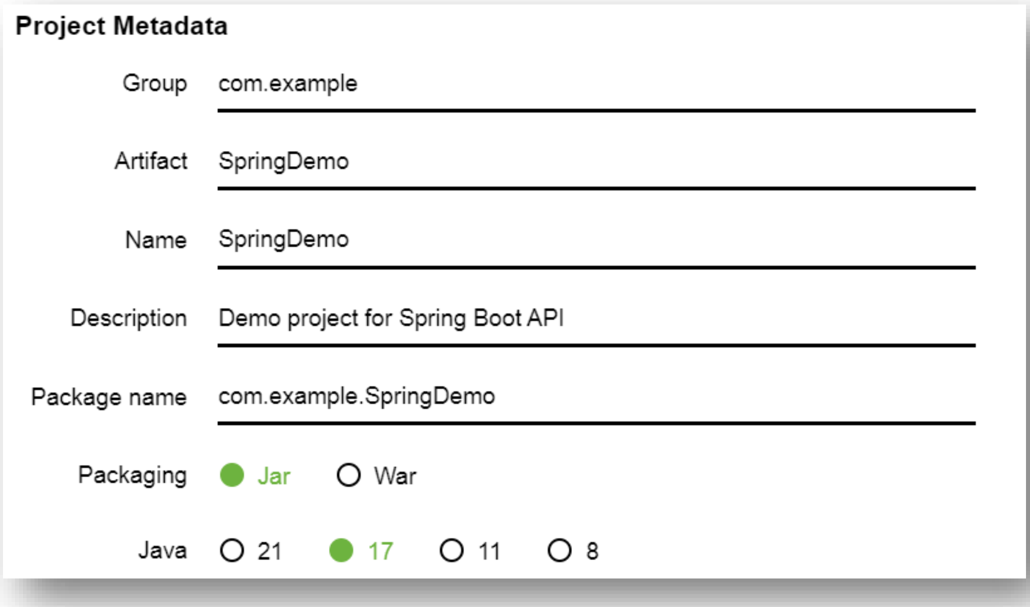

spring initializr

Project
○ Gradle - Groovy
○ Gradle - Kotlin [✓ Maven]

Language
[✓ Java] ○ Kotlin ○ Groovy

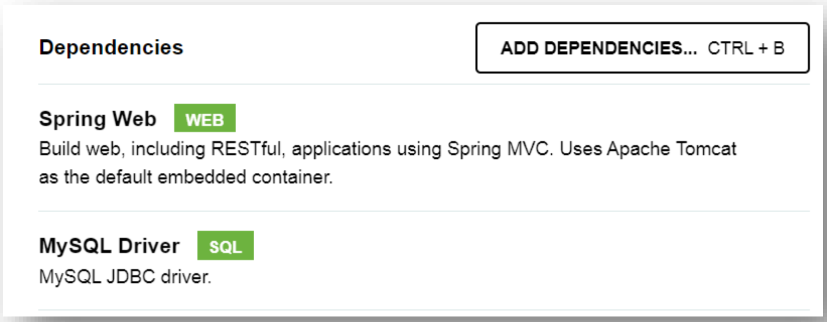
Spring Boot
○ 3.2.0 (SNAPSHOT) ○ 3.2.0 (RC2) ○ 3.1.6 (SNAPSHOT) [✓ 3.1.5]
○ 3.0.13 (SNAPSHOT) ○ 3.0.12 ○ 2.7.18 (SNAPSHOT) ○ 2.7.17

- Provide Group , Artifact , Description for your project and select the Java version


Project Metadata

Group	com.example
Artifact	SpringDemo
Name	SpringDemo
Description	Demo project for Spring Boot API
Package name	com.example.SpringDemo
Packaging	[✓ Jar] ○ War
Java	○ 21 [✓ 17] ○ 11 ○ 8

- Add the Dependency : Spring Web , MySQL Driver



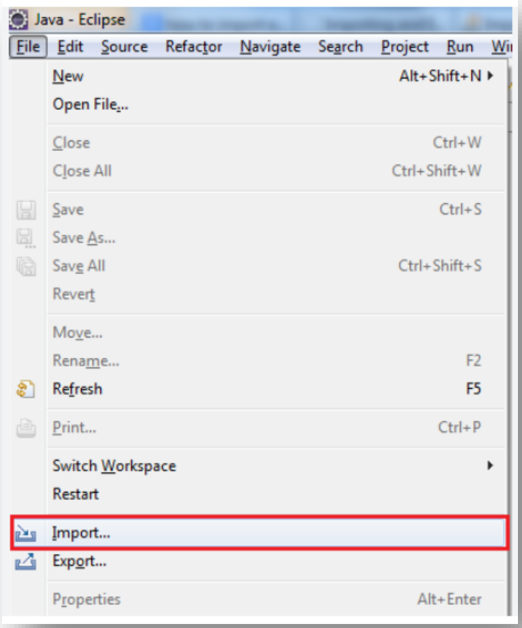
Dependencies

ADD DEPENDENCIES... CTRL + B

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

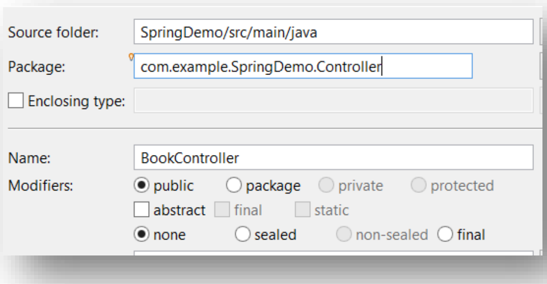
MySQL Driver SQL
MySQL JDBC driver.

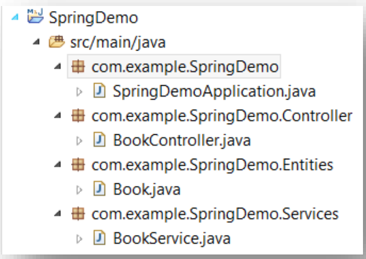
- Click on Generate , The zip file gets downloaded
- Extract the zip file
- Open Code Editor and import the extracted Package



- Open Code Editor and import the extracted Package
- Once when the package gets imported , in src/main/resources –> application.properties write the code **server.port=8100**
- In src/main/java click on the auto generated package -> create a class BookController and mention its package name as at Package end .Controller click finish

- In the same way create another class named Book.java inside Entities package
- And BookServices.java inside Services package


Source folder: SpringDemo/src/main/java
Package: com.example.SpringDemo.Controller
☐ Enclosing type:
Name: BookController
Modifiers: [✓ public] package private protected
☐ abstract ☐ final ☐ static
[✓ none] sealed non-sealed final


```
SpringDemo
  src/main/java
    com.example.SpringDemo
      SpringDemoApplication.java
    com.example.SpringDemo.Controller
      BookController.java
    com.example.SpringDemo.Entities
      Book.java
    com.example.SpringDemo.Services
      BookService.java
```

❖ Book.java

```
package com.example.SpringDemo.Entities;

public class Book
{
	private int id;
	private String title;
	private String author;
	public int getId()
	{return id;}
	public void setId(int id)
	{this.id = id;}
	public String getTitle()
	{return title;}
	public void setTitle(String title)
	{this.title = title;}
	public String getAuthor()
	{return author;}
	public void setAuthor(String author)
	{this.author = author;}
	public Book(int id, String title, String author)
	{
		super();
		this.id = id;
		this.title = title;
		this.author = author;
	}
	public Book() {}

}
```

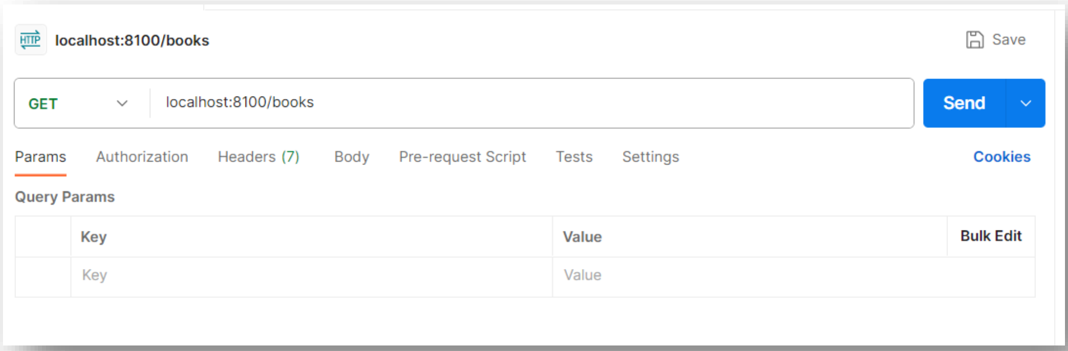
❖ BookService.java

```
package com.example.SpringDemo.Services;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
import org.springframework.stereotype.Component;
import com.example.SpringDemo.Entities.Book;

@Component
public class BookService
{
     private static List <Book> list=new ArrayList<>();
     static
     {list.add(new Book(101,"java","James"));
     list.add(new Book(102,"python","Guido Van Rossum"));}

     public List<Book> getBooks()
     {return list;}

     public Book getBookById(int id)
       {Book book=null;
        book=list.stream().
        filter(e->e.getId()==(id)).findFirst().get();
        return book;}

       public void addBook(Book b)
       {list.add(b);}

       public void deleteBook(int bid)
       {list=list.stream().
       filter(book->book.getId()!=bid)
       .collect(Collectors.toList());}

       public void updateBook(Book book,int bookId)
       {list.stream().map(b->{if(b.getId()==bookId)
                 {
                  b.setTitle(book.getTitle());
                  b.setAuthor(book.getAuthor());
                 }
                  return b;}).collect(Collectors.toList());
           }
       }
```

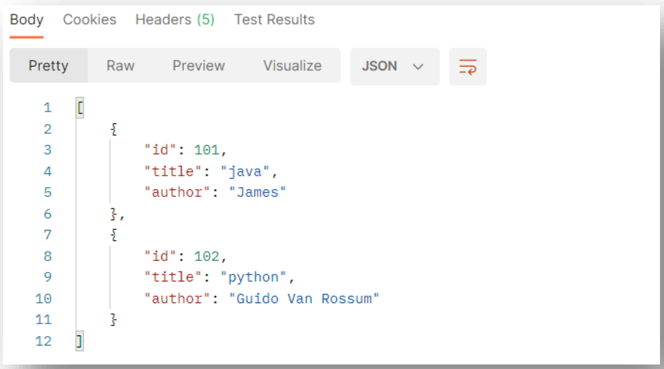
❖ BookController.java

```
package com.example.SpringDemo.Controller;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.example.SpringDemo.Entities.Book;
import com.example.SpringDemo.Services.BookService;

@RestController
public class BookController
{
	@Autowired
	private BookService bookService;

	@GetMapping("/books/{id}")
	public Book getBook(@PathVariable("id") int id)
	{return bookService.getBookById(id);}


	@GetMapping("/books")
	public List <Book> getBooks()
	{return this.bookService.getBooks();}

	@PostMapping("/books")
	public Book addBook(@RequestBody Book book)
	{this.bookService.addBook(book);
	 return book;}

	@DeleteMapping("/books/{bookId}")
	public void deleteBook(@PathVariable("bookId")int bookId)
	{this.bookService.deleteBook(bookId);}

	@PutMapping("books/{bookId}")
	public Book updateBook(@RequestBody Book  book,
	@PathVariable("bookId")int bookId)
	{this.bookService.updateBook(book,bookId);
	return book;}
}
```

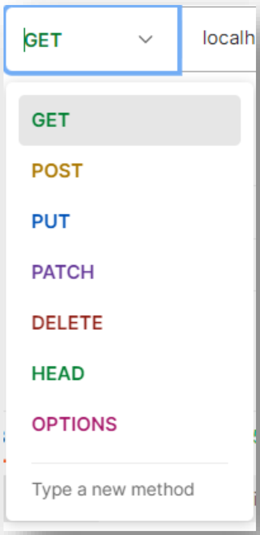
- Once when all the coding is done , run the main file i.e. SpringDemoApplication.java
- The application starts in the specified port i.e. 8100
- Install Postman which is a standalone tool that exercises web APIs by making HTTP requests from outside the service



- GET method is used to display the records stored
- Likewise POST is used to add new data , PUT to update the existing data , DELETE to delete the existing data



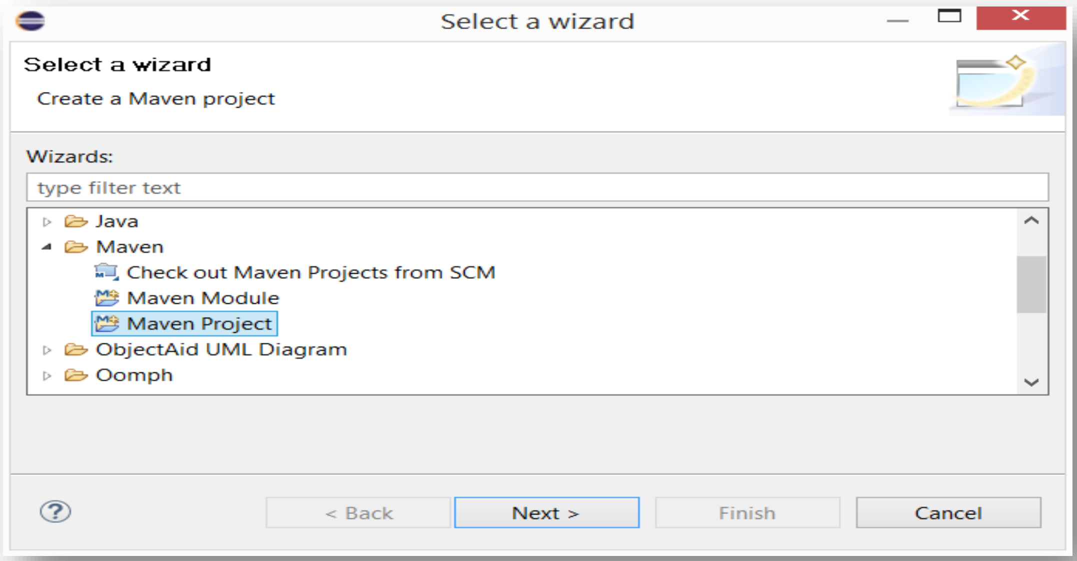
```
[
    {
        "id": 101,
        "title": "java",
        "author": "James"
    },
    {
        "id": 102,
        "title": "python",
        "author": "Guido Van Rossum"
    }
]
```



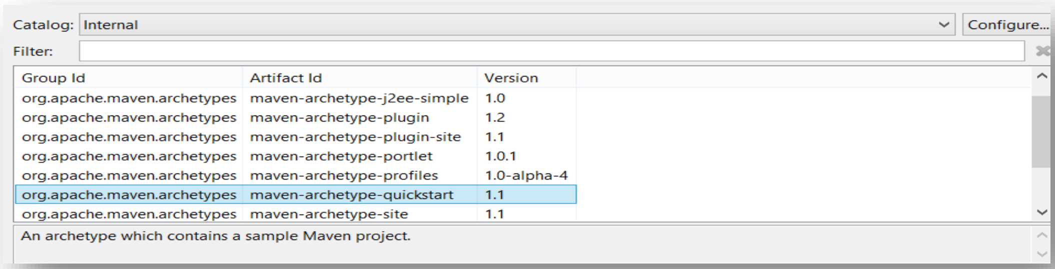
EXPERIMENT –   Date :

CREATE A SPRING APPLICATION USING DEPENDENCY INJECTION

- Open Eclipse , Go to file -> new -> other -> Maven -> Maven Project

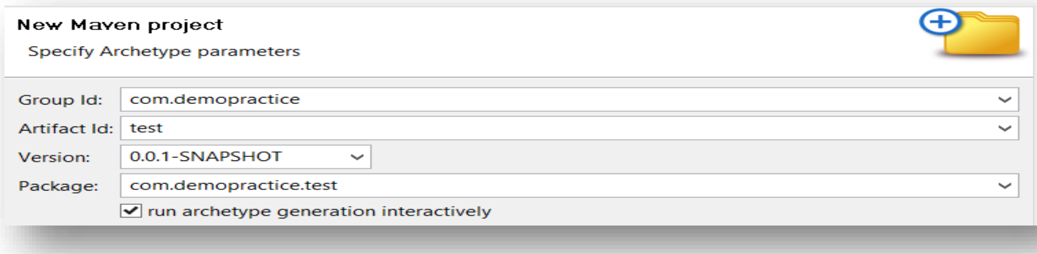


- Select the maven-archetype-quickstart artifact Id and specify the group id and artifact id for the project


Catalog: Internal
Filter:

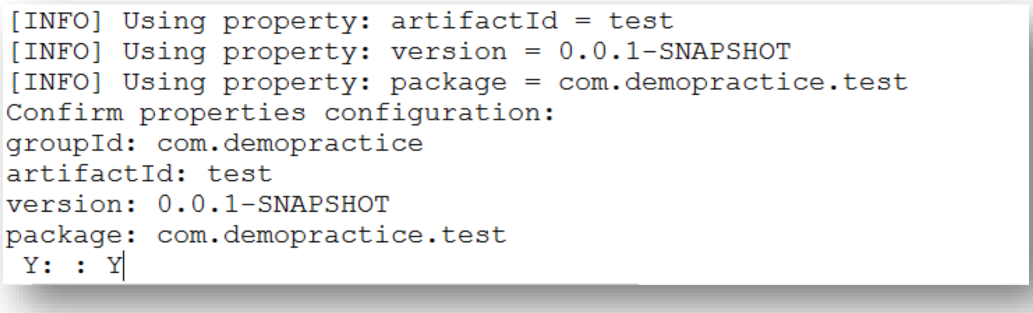
Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-j2ee-simple	1.0
org.apache.maven.archetypes	maven-archetype-plugin	1.2
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.1
org.apache.maven.archetypes	maven-archetype-site	1.1

An archetype which contains a sample Maven project.


New Maven project
Specify Archetype parameters

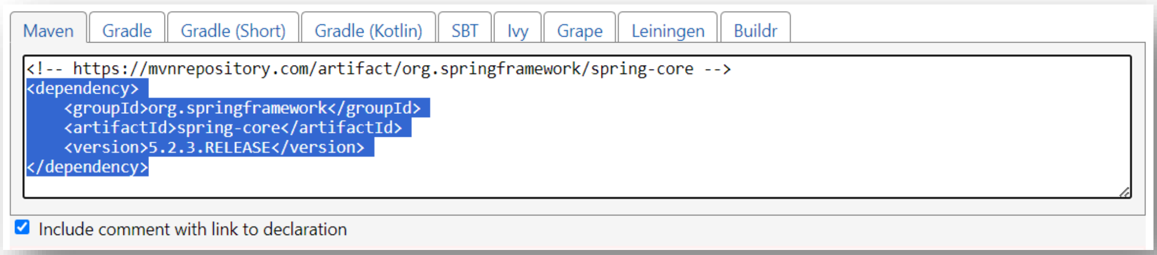
Group Id: com.demopractice
Artifact Id: test
Version: 0.0.1-SNAPSHOT
Package: com.demopractice.test
[✓] run archetype generation interactively

- Type Y in the console which means Yes to proceed further which is responsible to create a project with specified name



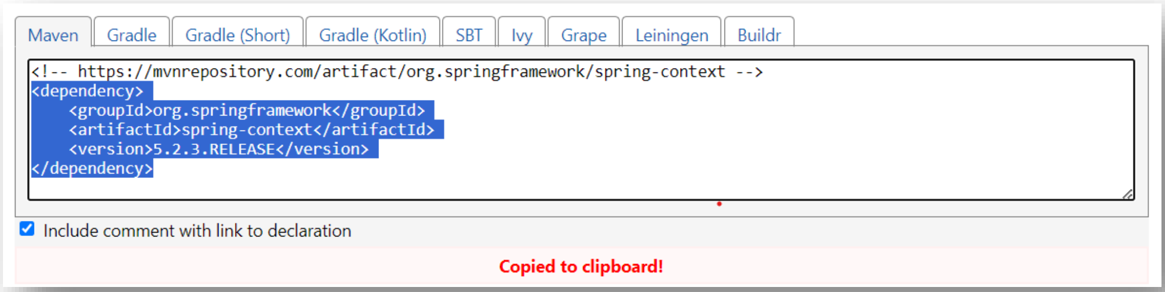
```
[INFO] Using property: artifactId = test
[INFO] Using property: version = 0.0.1-SNAPSHOT
[INFO] Using property: package = com.demopractice.test
Confirm properties configuration:
groupId: com.demopractice
artifactId: test
version: 0.0.1-SNAPSHOT
package: com.demopractice.test
 Y: : Y
```

- Go to google
- Search spring core maven and click on https://mvnrepository.com/artifact/org.springframework/spring-core
- Click on required dependency and copy the spring-core and spring-context code



```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>5.2.3.RELEASE</version>
</dependency>
```

☑ Include comment with link to declaration

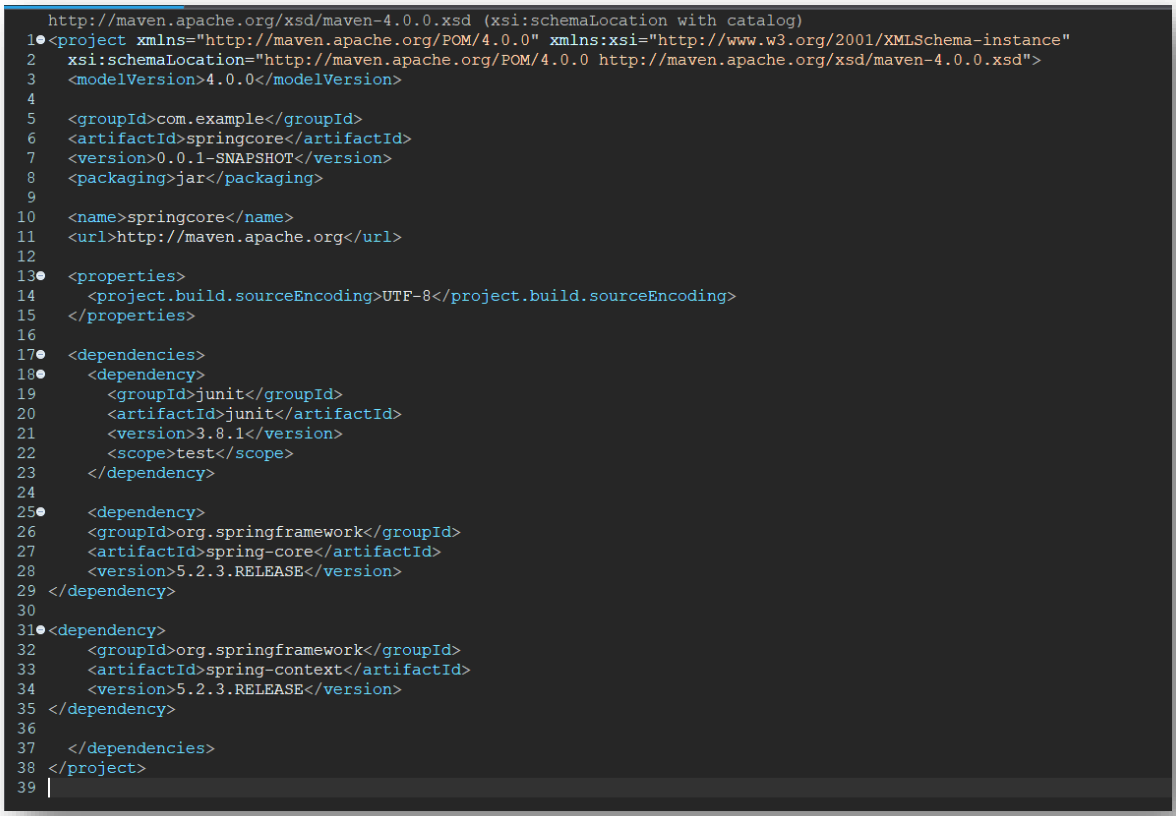


```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.3.RELEASE</version>
</dependency>
```

☑ Include comment with link to declaration

Copied to clipboard!

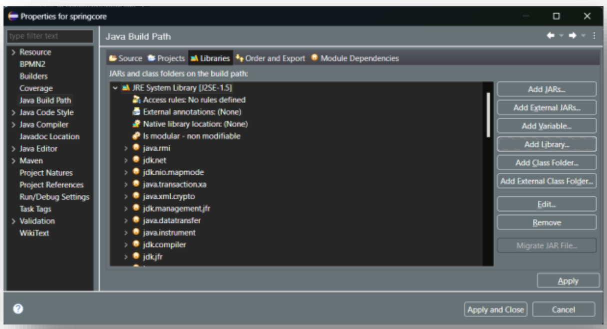
- Paste this in pom.xml file which is present in the created project



```
http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>springcore</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>springcore</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>5.2.3.RELEASE</version>
</dependency>

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.3.RELEASE</version>
</dependency>

  </dependencies>
</project>
```

➢ <u>BUILD JAVA PATH</u>

- Right click on project -> Properties
- Click on Java Build Path -> Remove JRE System Library [J2SE-1.5] -> Add Library -> JRE System Library
- Click on workspace

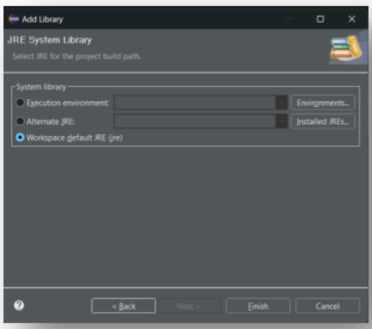


Properties for springcore

Java Build Path

JARs and class folders on the build path:

```
JRE System Library [J2SE-1.5]
    Access rules: No rules defined
    External annotations: (None)
    Native library location: (None)
    Is modular - non modifiable
  > java.rmi
  > jdk.net
  > jdk.nio.mapmode
  > java.transaction.xa
  > java.xml.crypto
  > jdk.management.jfr
  > java.datatransfer
  > java.instrument
  > jdk.compiler
  > jdk.jfr
```



Add Library

JRE System Library

Select JRE for the project build path.

System library

Execution environment:

Alternate JRE:

[✓ Workspace default JRE (jre)]

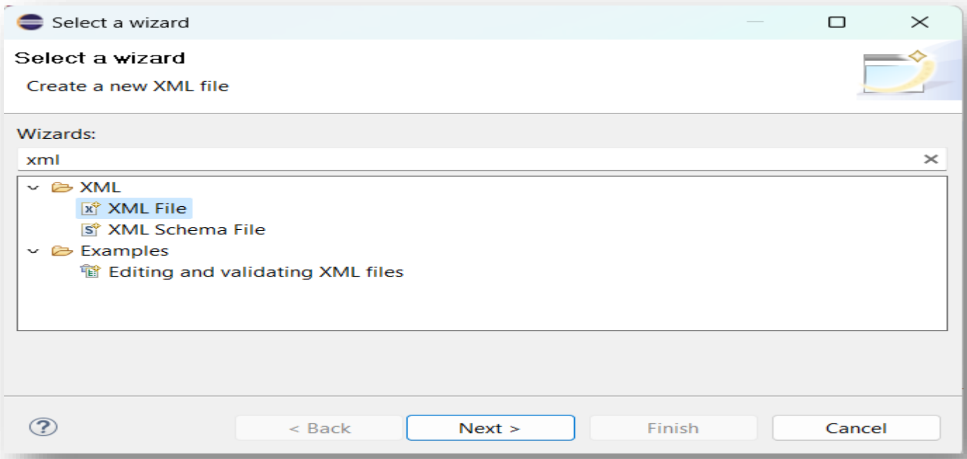
- Finish -> Apply and close

- In the project go to “src/main/java” there in package create a class as student.

Student.java

```
package com.example.springcore;

public class Student {

private int studentId;

private String studentName;

private String studentAddress;

public int getStudentId()

{return studentId;}

public void setStudentId(int studentId)

{this.studentId = studentId;}

public String getStudentName()

{return studentName;}

public void setStudentName(String studentName)

{this.studentName = studentName;}

public String getStudentAddress()

{return studentAddress;}

public void setStudentAddress(String studentAddress)

{this.studentAddress = studentAddress;}

public Student(int studentId, String studentName, String studentAddress) {

super();

this.studentId = studentId;

this.studentName = studentName;

this.studentAddress = studentAddress;}

public Student()

{super();}
@Override

public String toString()

{return "Student [studentId=" + studentId + ", studentName=" +

studentName + ", studentAddress=" + studentAddress+ "]";}}
```

- In the “src/main/java” right click on it and create xml file -> Select other
- Select XML file and give name as config.xml



- Go the browser and search spring 5 documentation pdf and copy the bean code and paste it in config.xml file ,
- Now add property and set in the bean section

config.xml

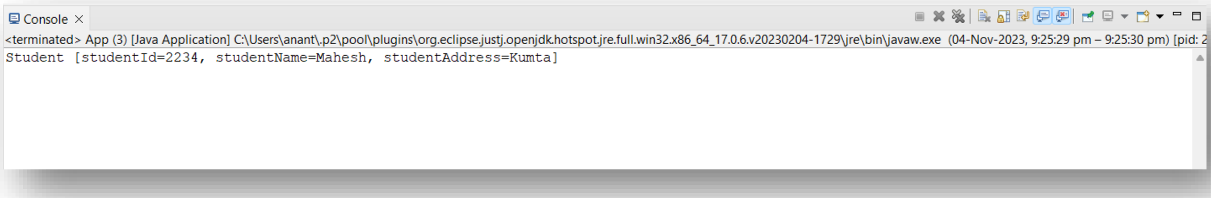
```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<!--this is our beans-->

<bean class=" com.example.springcore.Student" name="student1">

<property name="studentId">
```

```
<value>2234</value>

</property>

<property name="studentName">

<value>Mahesh</value>

</property>

<property name="studentAddress">

<value>Kumta</value>

</property>

</bean>

</beans>
```

App.java

```
package com.example.springcore;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main( String[] args )

{System.out.println( "Hello World!" );

ApplicationContext context=new
ClassPathXmlApplicationContext("config.xml");

Student student1=(Student)context.getBean("student1");

System.out.println(student1);}}
```

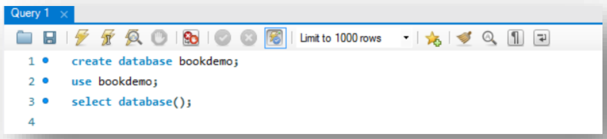
- Run the App.js file to get the output of the code



```
<terminated> App (3) [Java Application] C:\Users\anant\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre\bin\javaw.exe  (04-Nov-2023, 9:25:29 pm – 9:25:30 pm) [pid: 2
Student [studentId=2234, studentName=Mahesh, studentAddress=Kumta]
```

CREATE A SPRING BOOT DATA JPA TO MAINTAIN BOOK DETAILS AND PERFORM CRUD OPERATIONS AND SAVE THE DETAILS IN DATABABSE USING MYSQL

- Create a Maven project from Spring Initilizr , add dependencies Spring Web,Spring Data Jpa and MySQL Driver
- Install Workbench and create a database in a particular port & use it

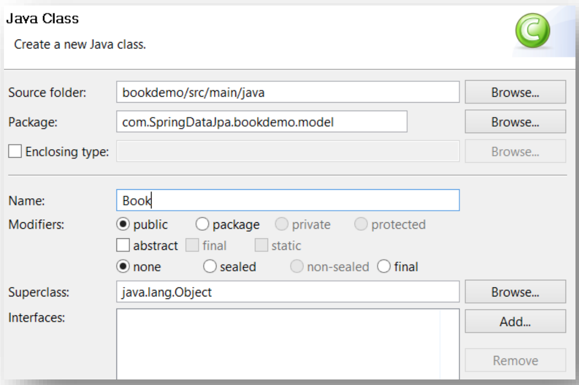


```
create database bookdemo;
use bookdemo;
select database();
```

- Write the below code in src/main/resources -> **application.properties**

```
spring.datasource.name=bookdemo
spring.datasource.url=jdbc:mysql://localhost:3307/bookdemo
spring.datasource.username=root
spring.datasource.password=student
spring.jpa.properties.hibernate.dialect=
org.hibernate.dialect.MySQL57Dialect
spring.datasource.dbcp2.driver-class-name=
com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
```

- Right click on the main package which is auto generated create a class Book.java and mention its package name as .model at the end of already displayed package



Java Class
Create a new Java class.

Source folder: bookdemo/src/main/java
Package: com.SpringDataJpa.bookdemo.model
Enclosing type: [ ]
Name: Book
Modifiers: [✓ public] package private protected
abstract final static
[✓ none] sealed non-sealed final
Superclass: java.lang.Object
Interfaces:

❖ <u>Book.java</u>

```java
package com.SpringDataJpa.bookdemo.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="books")
public class Book
{
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private long bookId;
	private String bookName;
	private String bookAuthor;
	public long getBookId()
	{return bookId;}
	public void setBookId(long bookId)
	{this.bookId = bookId;}
	public String getBookName()
	{return bookName;}
	public void setBookName(String bookName)
	{this.bookName = bookName;}
	public String getBookAuthor()
	{return bookAuthor;}
	public void setBookAuthor(String bookAuthor)
	{this.bookAuthor = bookAuthor;}

	@Override
	public String toString()
	{return "Book [bookId=" + bookId + ", bookName=" +
	bookName + ", bookAuthor=" + bookAuthor + "]";}

}
```

- Create another package named Repository by clicking on main package and an interface inside repository package i.e. BookRepository.java
- In the same way create another class named BookController.java inside controller package

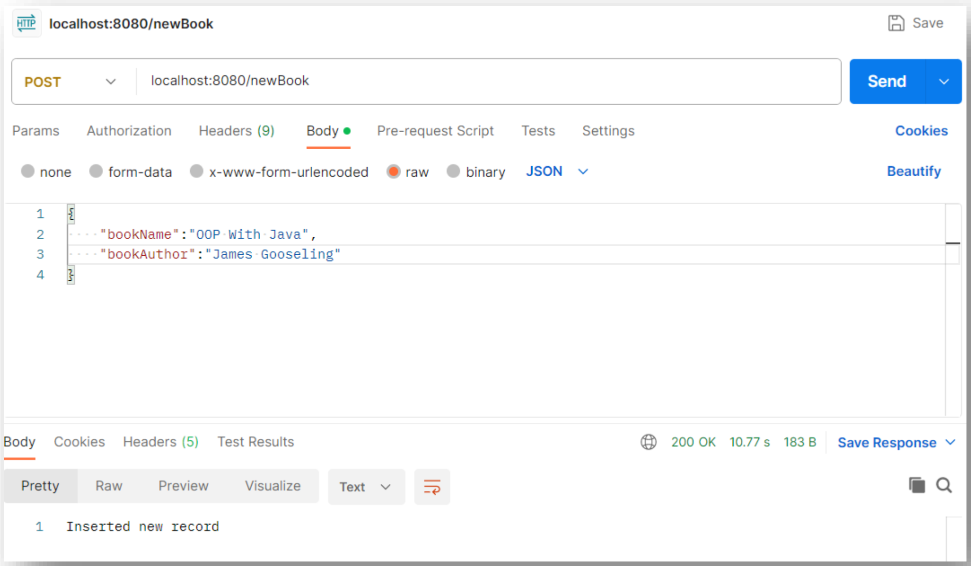
❖ BookRepository.java

```
package com.SpringDataJpa.bookdemo.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.SpringDataJpa.bookdemo.model.Book;

public interface BookRepository extends
JpaRepository<Book, Long>
{}
```

❖ BookController.java

```
package com.SpringDataJpa.bookdemo.controller;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
import com.SpringDataJpa.bookdemo.model.Book;
import com.SpringDataJpa.bookdemo.repository.BookRepository;

@RestController
public class BookController
{
	@Autowired
	BookRepository bookRepo;

	@PostMapping("/newBook")
	public String insertBook(@RequestBody Book book)
	{bookRepo.save(book);
	return "Inserted new record";}

	@PostMapping("/manyBook")
	public String insertBook(@RequestBody List<Book> book)
	{bookRepo.saveAll(book);
	return "Inserted new records";}

	@GetMapping("/getAllBook")
	public List <Book> getBook()
	{return (List<Book>)bookRepo.findAll();}

}
```

- Run the main file and check whether the table is created in the workbench or not using the query show tables;
- To insert a single data go to Postman application type the localhost in which the tomcat server is started select POST method

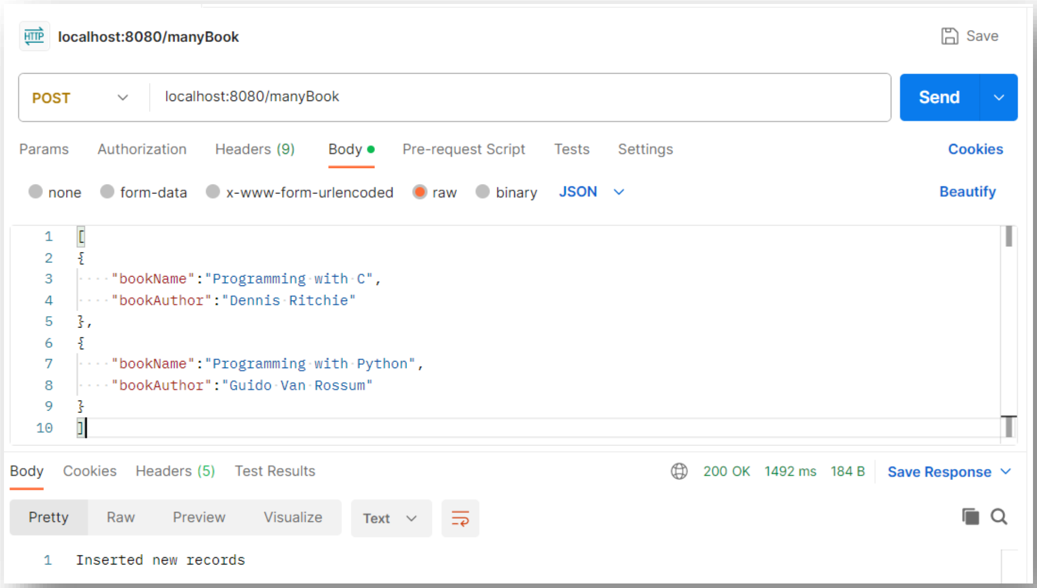

localhost:8080/newBook
Save
POST
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary JSON
Beautify
Body Cookies Headers (5) Test Results
200 OK 10.77 s 183 B
Save Response
Pretty Raw Preview Visualize Text

localhost:8080/newBook

```
{
    "bookName":"OOP With Java",
    "bookAuthor":"James Gooseling"
}
```

```
Inserted new record
```

- To insert a multiple data go to Postman application type the localhost in which the tomcat server is started select POST method and provide the data in an array

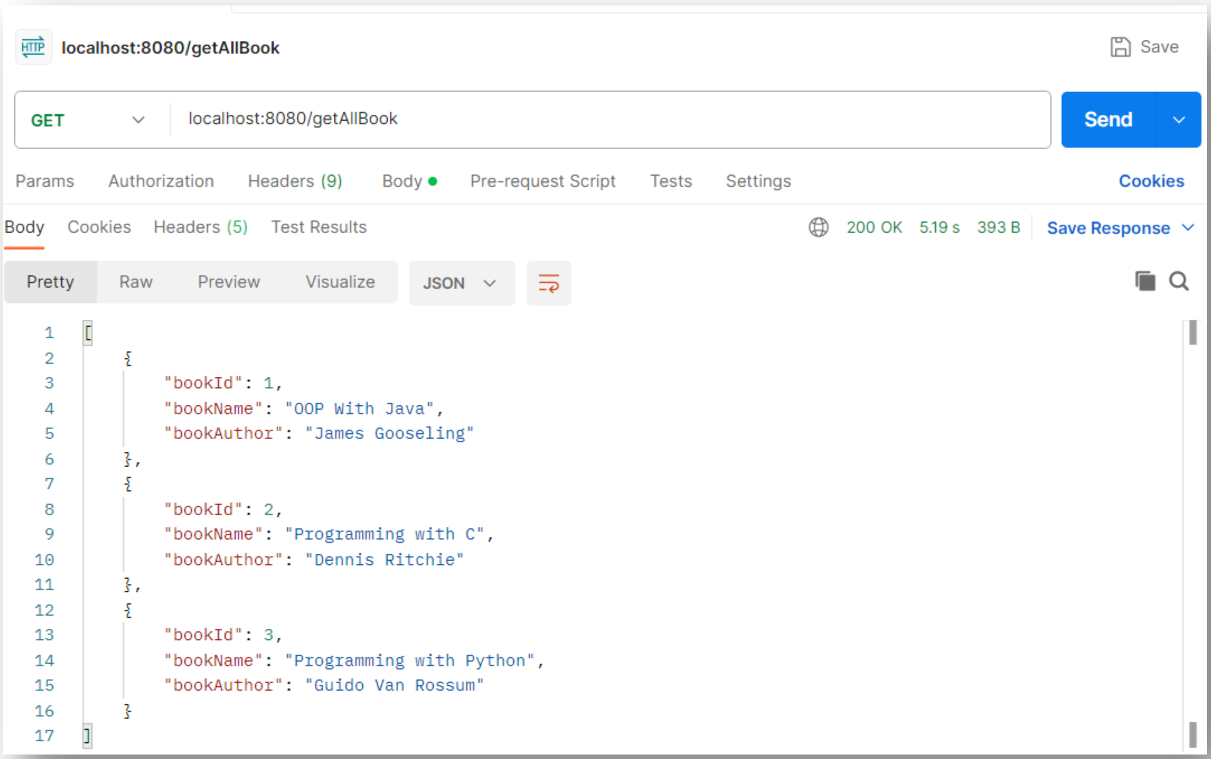

localhost:8080/manyBook
Save
POST
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary JSON
Beautify
Body Cookies Headers (5) Test Results
200 OK 1492 ms 184 B
Save Response
Pretty Raw Preview Visualize Text

localhost:8080/manyBook

```
[
{
    "bookName":"Programming with C",
    "bookAuthor":"Dennis Ritchie"
},
{
    "bookName":"Programming with Python",
    "bookAuthor":"Guido Van Rossum"
}
]
```

```
Inserted new records
```

- To confirm whether the data is inserted or not you can type the query **select * from books;** in workbench or perform **GET** operation in postman



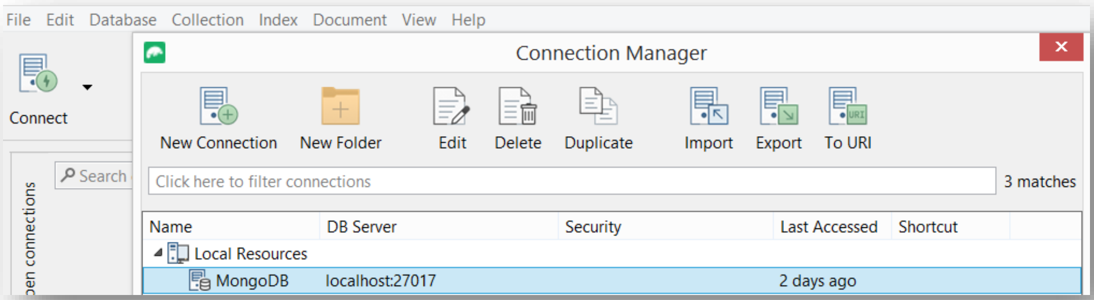
GET localhost:8080/getAllBook

200 OK 5.19 s 393 B

```
[
    {
        "bookId": 1,
        "bookName": "OOP With Java",
        "bookAuthor": "James Gooseling"
    },
    {
        "bookId": 2,
        "bookName": "Programming with C",
        "bookAuthor": "Dennis Ritchie"
    },
    {
        "bookId": 3,
        "bookName": "Programming with Python",
        "bookAuthor": "Guido Van Rossum"
    }
]
```

CREATE A SPRING BOOT DATA JPA TO MAINTAIN BOOK DETAILS AND PERFORM CRUD OPERATIONS AND SAVE THE DETAILS IN DATABASE USING MONGODB

- Create a Maven project from Spring Initilizr , add dependencies Spring Web,Spring Data Jpa , Spring Boot DevTools , Spring Data MongoDB
- Install Studio 3T application from browser and run the application -> click on connect and add new connection manually & connect to default port 27017 -> Open IntelliShell

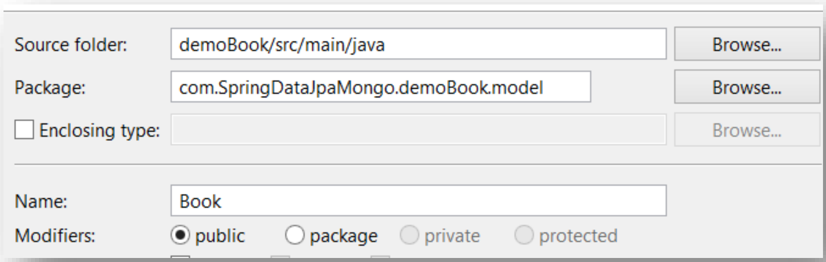


Name	DB Server	Security	Last Accessed	Shortcut
Local Resources				
MongoDB	localhost:27017		2 days ago	

- Create a database with the database name as mentioned in application.properties using query -> **use demoBook**
- Write the below code in src/main/resources -> **application.properties**

```
spring.data.mongodb.database=demoBook
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
server.port=1770
```

- Right click on the main package which is auto generated create a class Book.java and mention its package name as .model at the end of already displayed package


Source folder: demoBook/src/main/java
Package: com.SpringDataJpaMongo.demoBook.model
Enclosing type:
Name: Book
Modifiers: [✓ public] package private protected

❖ Book.java

```
package com.SpringDataJpa.bookdemo.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

package com.SpringDataJpaMongo.demoBook.model;

import org.springframework.data.mongodb.core.mapping.Document;

import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Document(collection="bookscheck")
public class Book
{
          @Id
          @GeneratedValue(strategy=GenerationType.IDENTITY)
          private long bookId;
          private String bookName;
          private String bookAuthor;
          public long getBookId()
          {return bookId;}
          public void setBookId(long bookId)
          {this.bookId = bookId;}
          public String getBookName()
          {return bookName;}
          public void setBookName(String bookName)
          {this.bookName = bookName;}
          public String getBookAuthor()
          {return bookAuthor;}
          public void setBookAuthor(String bookAuthor)
          {this.bookAuthor = bookAuthor;}

          @Override
          public String toString()
          {return "Book [bookId=" + bookId + ", bookName=" +
          bookName + ", bookAuthor=" + bookAuthor + "]";}


}
```

- Create another package named Repository by clicking on main package and an interface inside repository package i.e. BookRepository.java

- In the same way create another class named BookController.java inside controller package

❖ BookRepository.java

```
package com.SpringDataJpaMongo.demoBook.repository;

import
org.springframework.data.mongodb.repository.MongoRepository;

import com.SpringDataJpaMongo.demoBook.model.Book;

public interface BookRepository extends
MongoRepository<Book, Long>{}
```

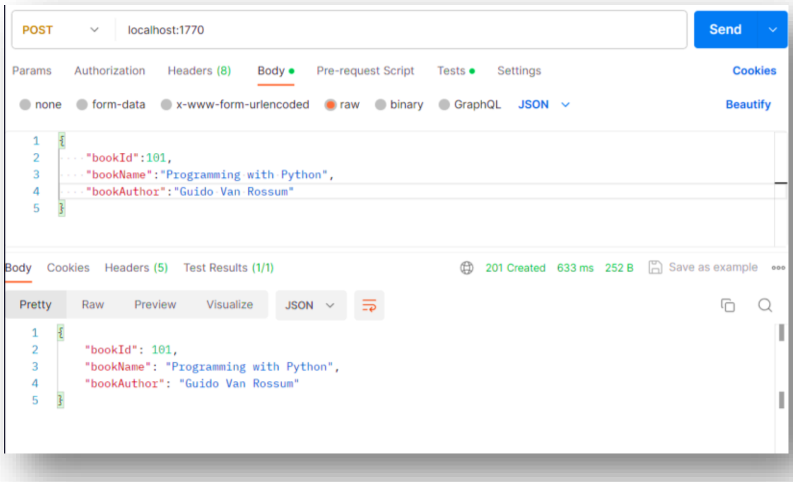
❖ BookController.java

```
package com.SpringDataJpaMongo.demoBook.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.SpringDataJpaMongo.demoBook.model.Book;
import
com.SpringDataJpaMongo.demoBook.repository.BookRepository;

@RestController
public class BookController
{

     @Autowired
     BookRepository bookRepo;

     @GetMapping
     public ResponseEntity<List<Book>> getAllBook()
     {List<Book> books=bookRepo.findAll();
     return new ResponseEntity<>(books,HttpStatus.OK);}


     @GetMapping("/{bookId}")
     public ResponseEntity<Book> getBookById(@PathVariable
     Long bookId)
     {     Book book=bookRepo.findById(bookId).orElse(null);
          if(book==null)
          {return new ResponseEntity<>(HttpStatus.NOT_FOUND);}
          return new ResponseEntity<>(book,HttpStatus.OK);}

     @PostMapping
     public ResponseEntity<Book> createBook
    (@RequestBody Book book)
     {Book createdBook=bookRepo.save(book);
      return new
      ResponseEntity<>(createdBook,HttpStatus.CREATED);}


     @PutMapping("/{bookId}")
     public ResponseEntity<Book> updateBook(@PathVariable Long
     bookId, @RequestBody Book book)
    {Book existingBook=bookRepo.findById(bookId).orElse(null);
          if(existingBook==null)
          {return new ResponseEntity<>(HttpStatus.NOT_FOUND);}
          existingBook.setBookName(book.getBookName());
          existingBook.setBookAuthor(book.getBookAuthor());
          bookRepo.save(existingBook);
    return new ResponseEntity<>( existingBook HttpStatus.OK);}

     @DeleteMapping("/{bookId}")
     public ResponseEntity<Void> deleteBook(@PathVariable Long
     bookId)
     {Book book=bookRepo.findById(bookId).orElse(null);
          if(book==null)
          {return new ResponseEntity<>(HttpStatus.NOT_FOUND);}
          bookRepo.delete(book);
          return new ResponseEntity<>(HttpStatus.NO_CONTENT);
     }}
```

- Run the Application file by adding (exclude={DataSourceAutoConfiguration.class}) after @SpringBootApplication annotation

- Open Postman to perform CRUD operations
  Type the localhost as specified in application.properties

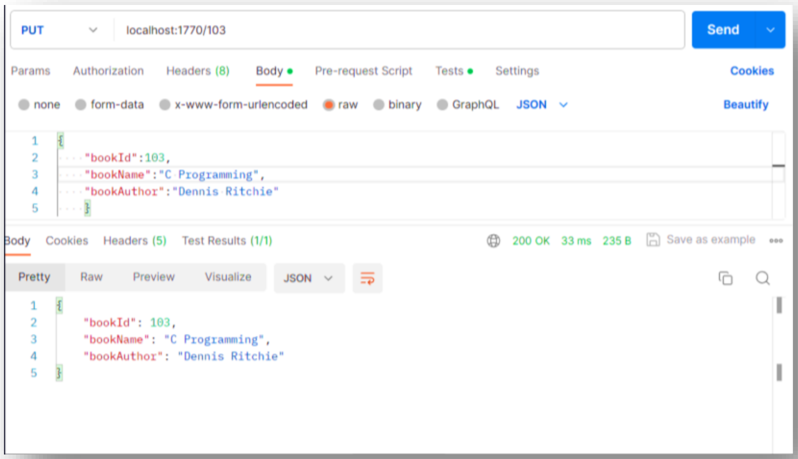
➢ To insert new data select POST



```
{
    "bookId":101,
    "bookName":"Programming with Python",
    "bookAuthor":"Guido Van Rossum"
}
```

```
{
    "bookId": 101,
    "bookName": "Programming with Python",
    "bookAuthor": "Guido Van Rossum"
}
```

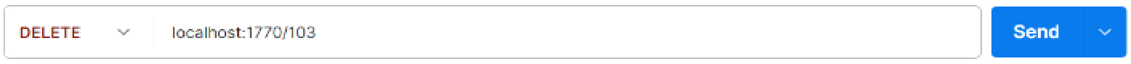
➢ To modify the existing data select PUT



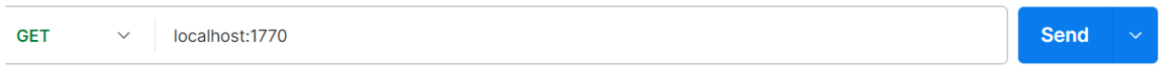
```
{
    "bookId":103,
    "bookName":"C Programming",
    "bookAuthor":"Dennis Ritchie"
    }
```

```
{
    "bookId": 103,
    "bookName": "C Programming",
    "bookAuthor": "Dennis Ritchie"
}
```

➢ To delete the existing data select DELETE



➢ To view all the data select GET method



➢ To check whether the data is inserted in the collection or not write query **db.bookscheck.find()** in Studio 3T IntelliShell

TEST SPRING BOOT DATA JPA PROJECT USING JUNIT TEST

- Go to “src/test/java” and their write the code in BookdataApplicationsTests.java

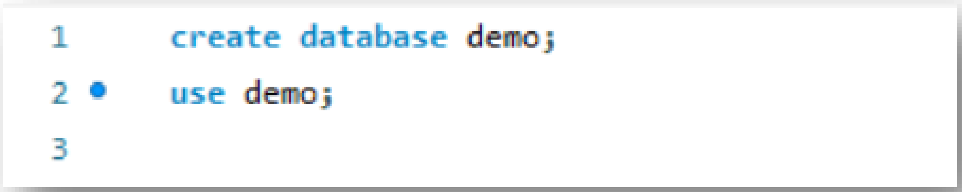
```
package com.example.bookdata1;
import static org.assertj.core.api.Assertions.assertThat;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotEquals;
import static org.junit.jupiter.api.Assertions.assertNotNull;
import java.util.List;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.example.bookdata1.Repository.BookRepository;
import com.example.bookdata1.model.Book;

@SpringBootTest
class Bookdata1ApplicationTests
{
	@Autowired
	BookRepository bookRepo;
	@Test
	public void testCreate()
	{
		Book b=new Book();
		b.setBookId(2);
		b.setbookAuthor("Mahesh");
		b.setBookName("varity");
		bookRepo.save(b);
		assertNotNull(bookRepo.findById(1).get());
	}

	@Test
	public void testSingle()
	{
		Book bk=bookRepo.findById(1).get();
		assertEquals("Life in woods",bk.getBookName());
	}

	@Test
	public void testReadAll()
	{
		List<Book>list=bookRepo.findAll();
		assertThat(list).size().isGreaterThan(0);
	}
@Test
public void testUpdate() {
      Book bk=bookRepo.findById(1).get();
      bk.setbookAuthor("Dinesh");
      bookRepo.save(bk);
      assertNotEquals("Ashwin",bookRepo.findById(1).get().
      getbookAuthor());
}

@Test
public void testDelete() {
      bookRepo.deleteById(2);
      assertThat(bookRepo.existsById(5)).isFalse();}
}
```

- **CONNECT TO THE MYSQL WORK BENCH**

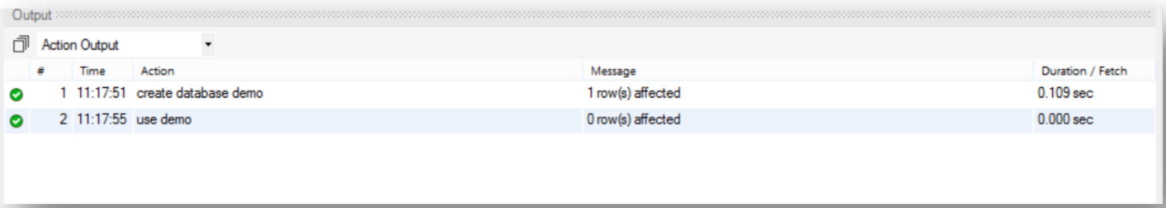
- Open MySQL workbench and give username and password
- Then create database demo using query

  create database demo;
- ctrl+enter to run the command
- to switch to the demo database write below query

  use demo;
- **Select * from books;** -> To select data from table "books".
- After writing the code run the BookdataApplicationsTests.java file



```
create database demo;
use demo;
```



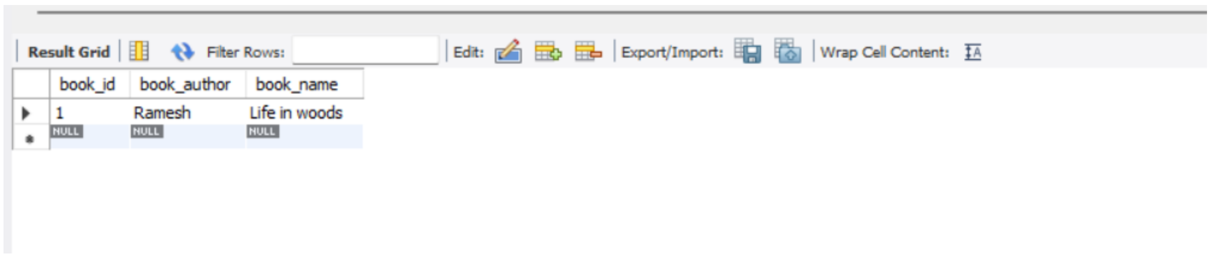
Output

Action Output

	#	Time	Action	Message	Duration / Fetch
✓	1	11:17:51	create database demo	1 row(s) affected	0.109 sec
✓	2	11:17:55	use demo	0 row(s) affected	0.000 sec

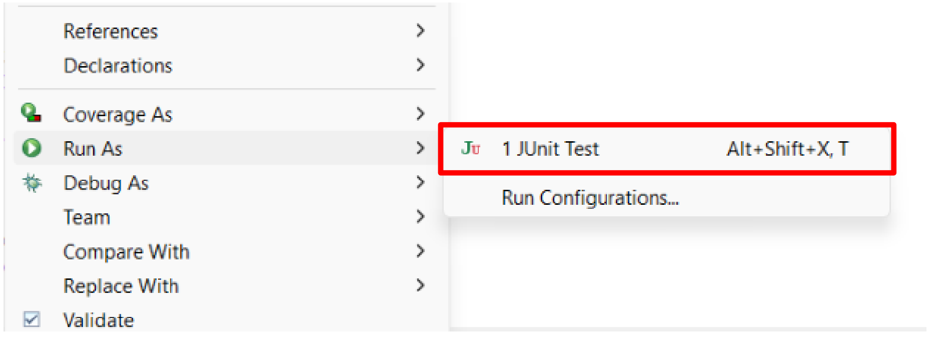
- Select & right click on testCreate in the code and run as Java Unit perform same for other codes too

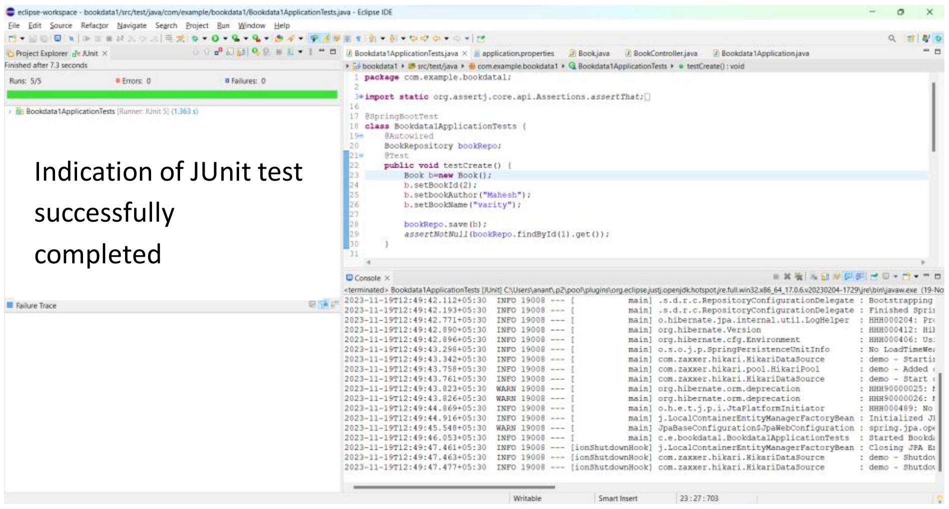
Output in MySQL



	book_id	book_author	book_name
▶	1	Ramesh	Life in woods
*	NULL	NULL	NULL

- Right click on BookdataApplicationsTests.java and run as JUnit Test




Finished after 7.3 seconds
Runs: 5/5
Errors: 0
Failures: 0
Bookdata1ApplicationTests [Runner: JUnit 5] (1.363 s)
Failure Trace

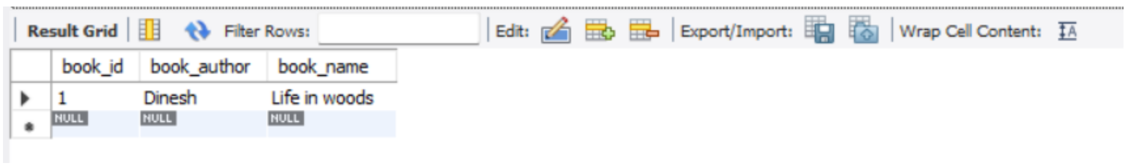
Indication of JUnit test successfully completed

```java
package com.example.bookdata1;

import static org.assertj.core.api.Assertions.assertThat;

@SpringBootTest
class Bookdata1ApplicationTests {
    @Autowired
    BookRepository bookRepo;
    @Test
    public void testCreate() {
        Book b=new Book();
        b.setBookId(2);
        b.setbookAuthor("Mahesh");
        b.setBookName("varity");

        bookRepo.save(b);
        assertNotNull(bookRepo.findById(1).get());
    }
```

```
<terminated> Bookdata1ApplicationTests [JUnit] C:\Users\anant\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre\bin\javaw.exe (19-No
2023-11-19T12:49:42.112+05:30  INFO 19008 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping
2023-11-19T12:49:42.193+05:30  INFO 19008 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Sprin
2023-11-19T12:49:42.771+05:30  INFO 19008 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Pro
2023-11-19T12:49:42.890+05:30  INFO 19008 --- [           main] org.hibernate.Version                    : HHH000412: Hib
2023-11-19T12:49:42.896+05:30  INFO 19008 --- [           main] org.hibernate.cfg.Environment            : HHH000406: Us
2023-11-19T12:49:43.298+05:30  INFO 19008 --- [           main] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWea
2023-11-19T12:49:43.342+05:30  INFO 19008 --- [           main] com.zaxxer.hikari.HikariDataSource       : demo - Starti
2023-11-19T12:49:43.758+05:30  INFO 19008 --- [           main] com.zaxxer.hikari.pool.HikariPool        : demo - Added
2023-11-19T12:49:43.761+05:30  INFO 19008 --- [           main] com.zaxxer.hikari.HikariDataSource       : demo - Start
2023-11-19T12:49:43.823+05:30  WARN 19008 --- [           main] org.hibernate.orm.deprecation            : HHH90000025: 
2023-11-19T12:49:43.826+05:30  WARN 19008 --- [           main] org.hibernate.orm.deprecation            : HHH90000026: 
2023-11-19T12:49:44.869+05:30  INFO 19008 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No
2023-11-19T12:49:44.916+05:30  INFO 19008 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JP
2023-11-19T12:49:45.548+05:30  WARN 19008 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.ope
2023-11-19T12:49:46.053+05:30  INFO 19008 --- [           main] c.e.bookdata1.Bookdata1ApplicationTests  : Started Bookda
2023-11-19T12:49:47.461+05:30  INFO 19008 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA En
2023-11-19T12:49:47.463+05:30  INFO 19008 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : demo - Shutdow
2023-11-19T12:49:47.477+05:30  INFO 19008 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : demo - Shutdow
```

- To check whether the test is properly executed write the query **SELECT * FROM book;** in MySQL Workbench

OUTPUT :

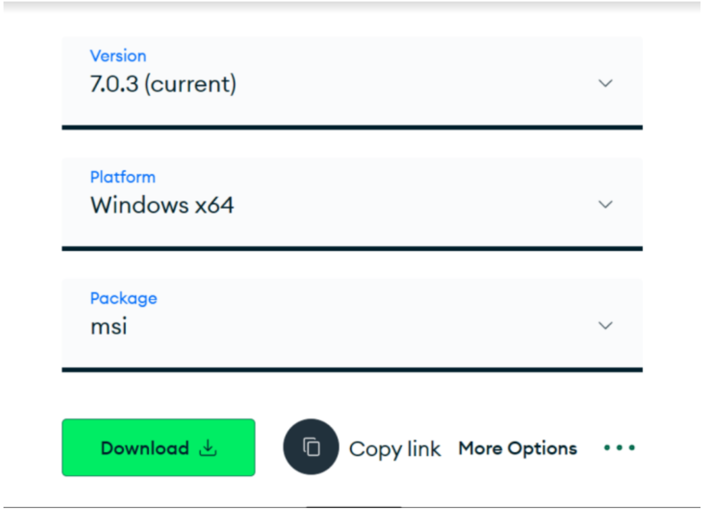


	book_id	book_author	book_name
▶	1	Dinesh	Life in woods
*	NULL	NULL	NULL

EXPERIMENT –   Date :

INSTALL MONGODB , MONGOSH & MONGODB COMPASS AND TRY THE BASIC COMMANDS

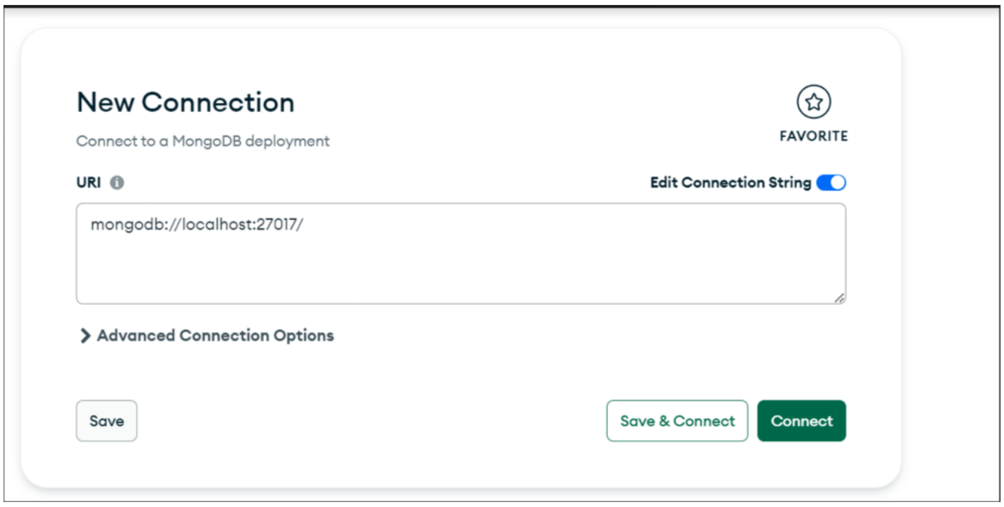
- Install Mongo DB from https://www.mongodb.com/try/download/community


Version
7.0.3 (current)

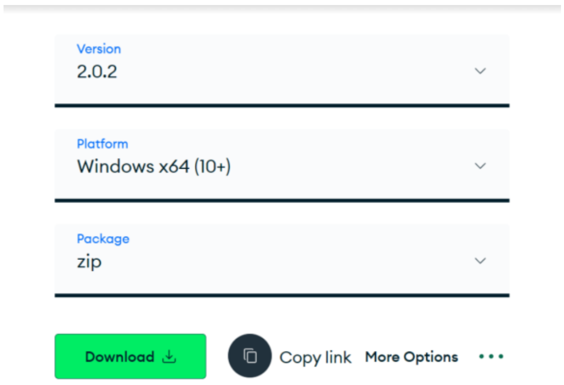
Platform
Windows x64

Package
msi

- Open the downloaded msi file and run
- Click next -> agree terms and conditions -> choose setup type : complete -> next -> next -> check the box : install mongodb compass -> install
- When the installation is complete it opens MongoDB Compass
- Click on save and connect & provide a specific connection name


New Connection
Connect to a MongoDB deployment
URI
Edit Connection String
mongodb://localhost:27017/

- Install Mongosh shell from https://www.mongodb.com/try/download/shell


Version
2.0.2
Platform
Windows x64 (10+)
Package
zip

- Extract the zip file to setup mongosh

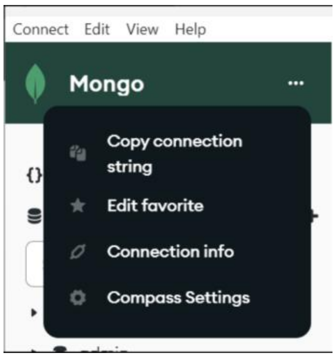
➢ **EDIT SYSTEM ENVIRONMENT VARIABLE**

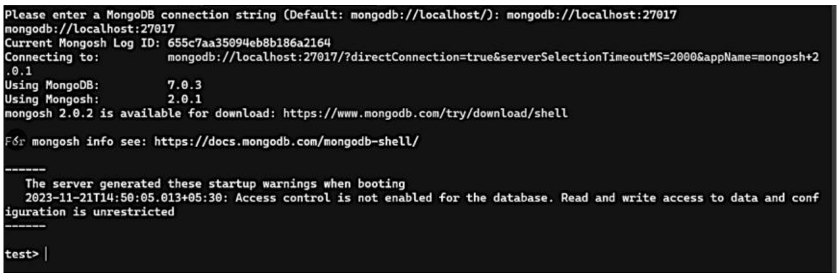
- search edit system variables -> environment variables-> select path in system variables -> edit -> new -> paste the path of mongoDB bin file and Mongosh bin file -> OK

  C:\Program Files\MongoDB\Server\7.0\bin

  C:\mongosh-2.0.1-win32-x64\bin

- to check version of mongosh open cmd and type mongosh –version
- Open mongosh (shell) : when it asks to enter connection string
  copy the connection string from MongoDB compass and paste it in the shell **mongodb://localhost:27017**

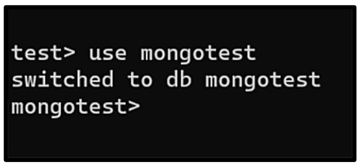



```
Please enter a MongoDB connection string (Default: mongodb://localhost/): mongodb://localhost:27017
mongodb://localhost:27017
Current Mongosh Log ID: 655c7aa35094eb8b186a2164
Connecting to:          mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.0.1
Using MongoDB:          7.0.3
Using Mongosh:          2.0.1
mongosh 2.0.2 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-11-21T14:50:05.013+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------

test> |
```

1. The use Query

MongoDB use DATABASE_NAME is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

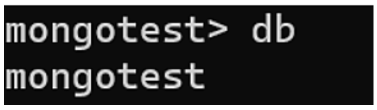
use mongotest



```
test> use mongotest
switched to db mongotest
mongotest>
```

2. To check your currently selected database

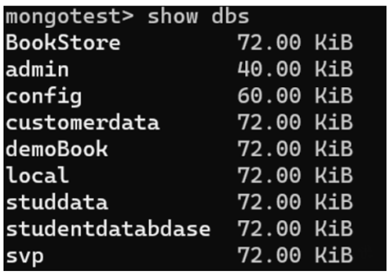
db



```
mongotest> db
mongotest
```

3. To check database list

show dbs

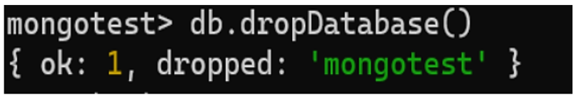


```
mongotest> show dbs
BookStore           72.00 KiB
admin               40.00 KiB
config              60.00 KiB
customerdata        72.00 KiB
demoBook            72.00 KiB
local               72.00 KiB
studdata            72.00 KiB
studentdatabdase    72.00 KiB
svp                 72.00 KiB
```

4. The dropDatabase Query

MongoDB db.dropDatabase() command is used to drop a existing database.

db.dropDatabase()

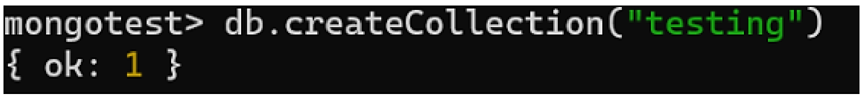


```
mongotest> db.dropDatabase()
{ ok: 1, dropped: 'mongotest' }
```

5. The createCollection() Query

MongoDB db.createCollection(name, options) is used to create collection.

db.createCollection("testing")

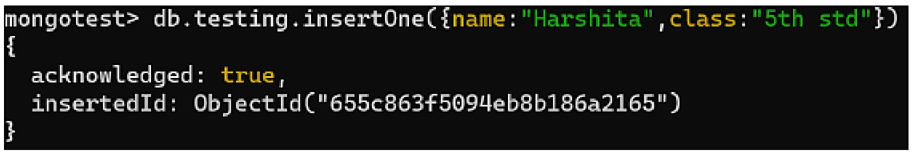


```
mongotest> db.createCollection("testing")
{ ok: 1 }
```

6. **The insertOne() Query**

   If you need to insert only one document into a collection you can use this method.

   db.testing.insertOne({name:"Harshita",class:"5th std"})

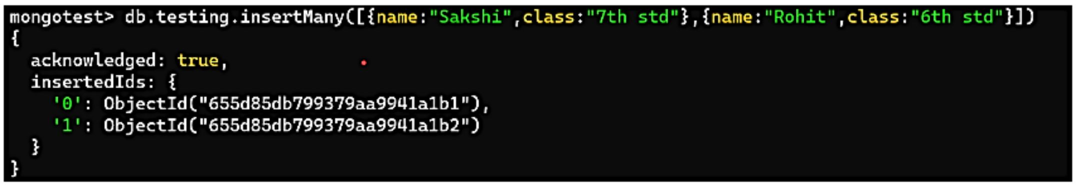


```
mongotest> db.testing.insertOne({name:"Harshita",class:"5th std"})
{
  acknowledged: true,
  insertedId: ObjectId("655c863f5094eb8b186a2165")
}
```

7. **The insertMany() Query**

   You can insert multiple documents using the insertMany() method. To this method you need to pass an array of documents.

   db.testing.insertMany([{name:"Sakshi",class:"7th std"},{name:"Rohit",class:"6th std"}])

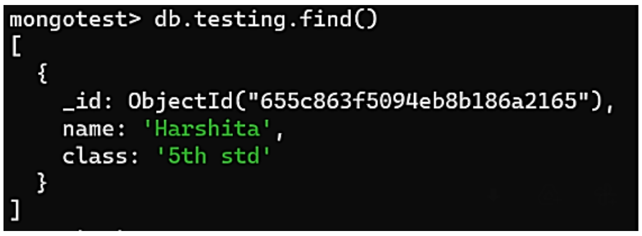


```
mongotest> db.testing.insertMany([{name:"Sakshi",class:"7th std"},{name:"Rohit",class:"6th std"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("655d85db799379aa9941a1b1"),
    '1': ObjectId("655d85db799379aa9941a1b2")
  }
}
```

8. **The find() Query**

   To query data from MongoDB collection, you need to use MongoDB's find() method.

   db.testing.find()



```
mongotest> db.testing.find()
[
  {
    _id: ObjectId("655c863f5094eb8b186a2165"),
    name: 'Harshita',
    class: '5th std'
  }
]
```

9. **The pretty() Query**

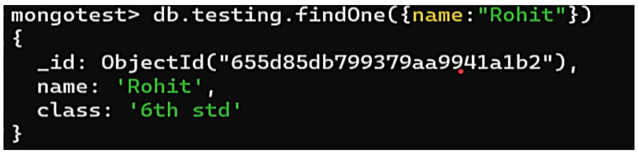
   To display the results in a formatted way, you can use pretty() method.

   db.testing.find().pretty()

10. **The findOne() Query**

   findOne() method returns only one document.

   db.testing.findOne({name:"Rohit"})

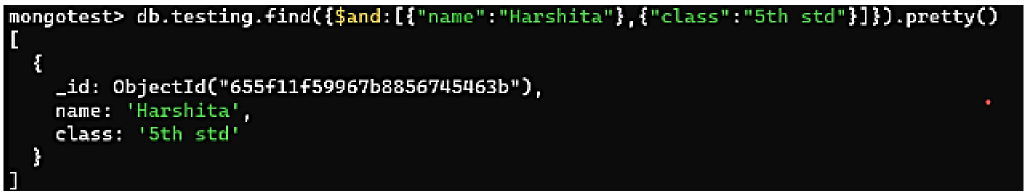


```
mongotest> db.testing.findOne({name:"Rohit"})
{
  _id: ObjectId("655d85db799379aa9941a1b2"),
  name: 'Rohit',
  class: '6th std'
}
```

11. The AND Query

To query documents based on the AND condition, you need to use $and keyword.

db.testing.find({$and[{“name”:”Harshita”},{“class”:“5th std”}]})

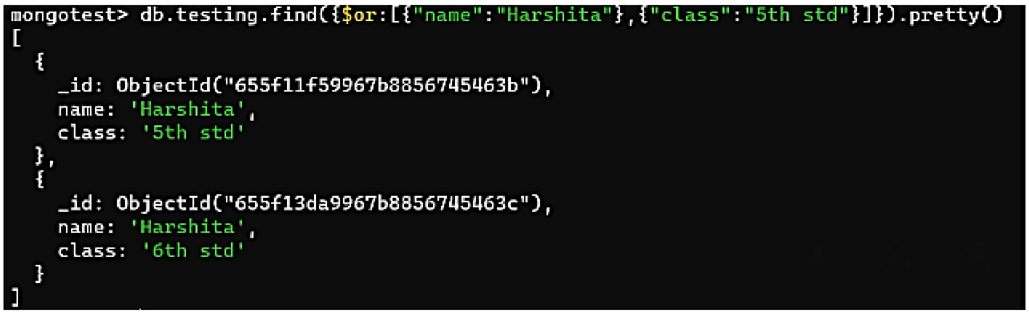


```
mongotest> db.testing.find({$and:[{"name":"Harshita"},{"class":"5th std"}]}).pretty()
[
  {
    _id: ObjectId("655f11f59967b8856745463b"),
    name: 'Harshita',
    class: '5th std'
  }
]
```

12.The OR Query

To query documents based on the OR condition, you need to use $or keyword.

db.testing.find({$or[{“name”:”Harshita”},{“class”:“5th std”}]})

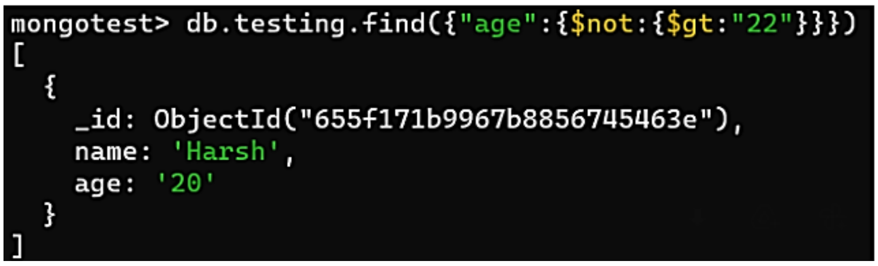


```
mongotest> db.testing.find({$or:[{"name":"Harshita"},{"class":"5th std"}]}).pretty()
[
  {
    _id: ObjectId("655f11f59967b8856745463b"),
    name: 'Harshita',
    class: '5th std'
  },
  {
    _id: ObjectId("655f13da9967b8856745463c"),
    name: 'Harshita',
    class: '6th std'
  }
]
```

13.The NOT Query

To query documents based on the NOT condition $not keyword is used

db.testing.find({“age”:{$not:{gt:“22”}}})

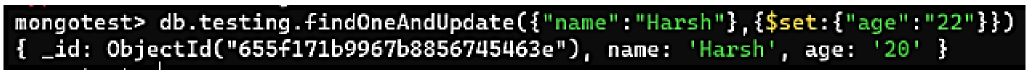


```
mongotest> db.testing.find({"age":{$not:{$gt:"22"}}})
[
  {
    _id: ObjectId("655f171b9967b8856745463e"),
    name: 'Harsh',
    age: '20'
  }
]
```

14.findOneAndUpdate Query

The findOneAndUpdate() method updates the values in the existing document.

db.testing.findAndUpdate({“name”:“Harsh”},{$set:{“age”:“22”}})

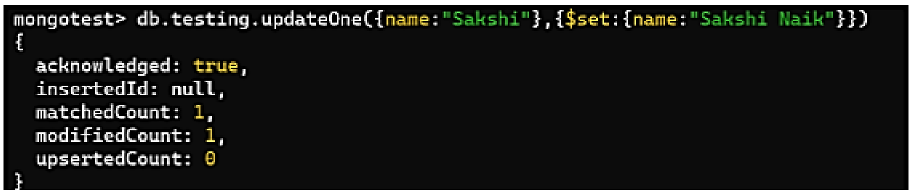


```
mongotest> db.testing.findOneAndUpdate({"name":"Harsh"},{$set:{"age":"22"}})
{ _id: ObjectId("655f171b9967b8856745463e"), name: 'Harsh', age: '20' }
```

15.The updateOne Query

This method updates a single document which matches the given filter.

db.testing.updateOne({name:”Sakshi”},{$set:{name:”Sakshi Naik”}})

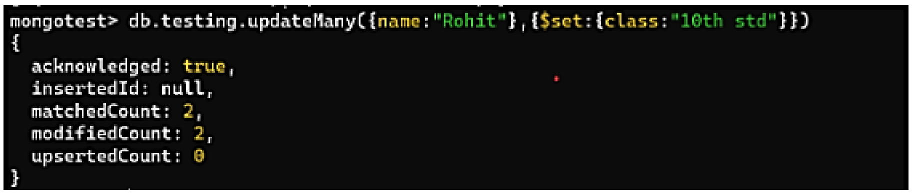


```
mongotest> db.testing.updateOne({name:"Sakshi"},{$set:{name:"Sakshi Naik"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

16.The updateMany Query

The updateMany() method updates all the documents that matches the given filter.

db.testing.updateMany({name:”Rohit”},{$set{class:”10th std”}})

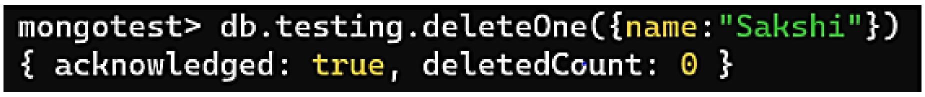


```
mongotest> db.testing.updateMany({name:"Rohit"},{$set:{class:"10th std"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

17.The deleteOne Query

MongoDB’s deleteOne() method is used to delete a single document from the collection.

db.testing.deleteOne({name:”Sakshi”})

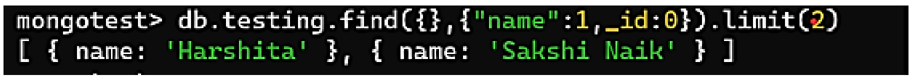


```
mongotest> db.testing.deleteOne({name:"Sakshi"})
{ acknowledged: true, deletedCount: 0 }
```

18.The limit Query

To limit the records in MongoDB, you need to use limit() method. The method accepts one number type argument, which is the number of documents that you want to be displayed.

db.testing.find({},{“name”:1,_id:0}).limit(2)

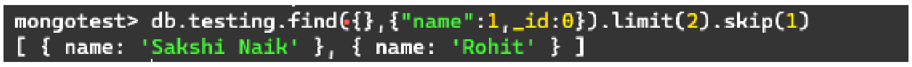


```
mongotest> db.testing.find({},{"name":1,_id:0}).limit(2)
[ { name: 'Harshita' }, { name: 'Sakshi Naik' } ]
```

19.The skip Query

used to skip the number of documents.

db.testing.find({},{“name:1”,_id:0}).limit(2).skip(1)



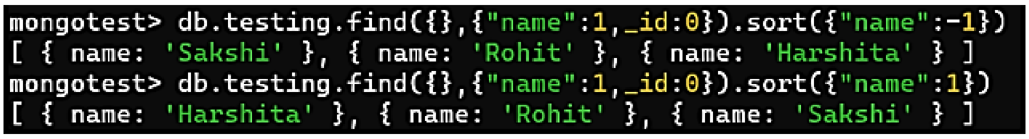
```
mongotest> db.testing.find({},{"name":1,_id:0}).limit(2).skip(1)
[ { name: 'Sakshi Naik' }, { name: 'Rohit' } ]
```

20.The sort Query

To sort documents in MongoDB, you need to use sort() method. The method accepts a document containing a list of fields along with their sorting order. To specify sorting order 1 and -1 are used. 1 is used for ascending order while -1 is used for descending order.

db.testing.find({},{“name”:1,_id:0}).sort({“name”:-1})

db.testing.find({},{“name”:1,_id:0}).sort({“name”:1})

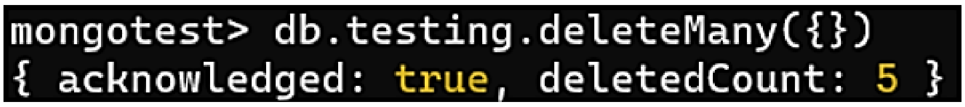


```
mongotest> db.testing.find({},{"name":1,_id:0}).sort({"name":-1})
[ { name: 'Sakshi' }, { name: 'Rohit' }, { name: 'Harshita' } ]
mongotest> db.testing.find({},{"name":1,_id:0}).sort({"name":1})
[ { name: 'Harshita' }, { name: 'Rohit' }, { name: 'Sakshi' } ]
```

21.The deleteMany Query

MongoDB’s deleteMany() method is used to delete multiple document from the collection.

db.testing.deleteMany({})

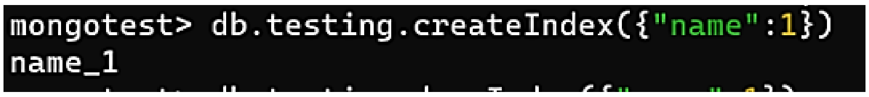


```
mongotest> db.testing.deleteMany({})
{ acknowledged: true, deletedCount: 5 }
```

22.The createIndex Query

improves the performance of data retrieval operations in databases and other data storage systems

db.testing.createIndex({“name”:1})

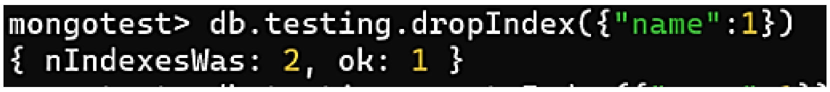


```
mongotest> db.testing.createIndex({"name":1})
name_1
```

23.The dropIndex Query

You can drop a particular index using the dropIndex() method of MongoDB.

db.testing.dropIndex({“name”:1})

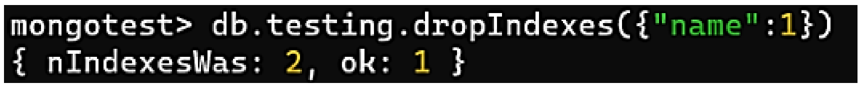


```
mongotest> db.testing.dropIndex({"name":1})
{ nIndexesWas: 2, ok: 1 }
```

24.The dropIndexes Query

This method deletes multiple (specified) indexes on a collection.

db.testing.dropIndexes({“name”:1})

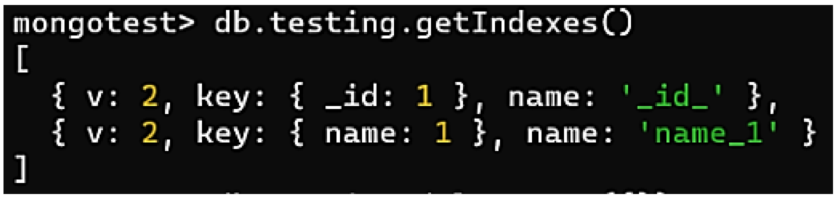


```
mongotest> db.testing.dropIndexes({"name":1})
{ nIndexesWas: 2, ok: 1 }
```

25.The getIndexes Query

This method returns the description of all the indexes int the collection.

db.testing.getIndexes()

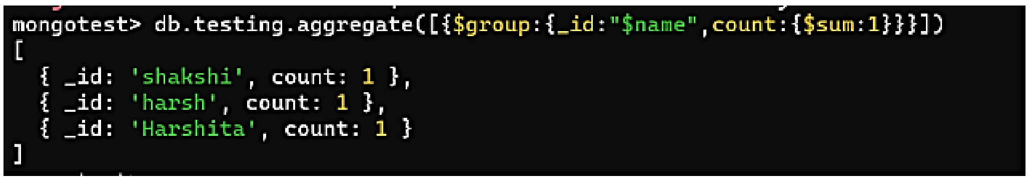


```
mongotest> db.testing.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1' }
]
```

26.The aggregate Query

You can use aggregation operations to group values from multiple documents together

db.testing.aggregate([{$group:{_id:“$name”,count:{$sum:1}}}])



```
mongotest> db.testing.aggregate([{$group:{_id:"$name",count:{$sum:1}}}])
[
  { _id: 'shakshi', count: 1 },
  { _id: 'harsh', count: 1 },
  { _id: 'Harshita', count: 1 }
]
```