Experiment – 01

Creating a project plan and product backlog for project and User story creation.

1. **Creating project plan**
    - o With project planning and project management running smoothly, IT departments can deliver more projects for the same budgets, and most properly managed projects are completed by their due time/date.
    - o Planning for every project is very individual and specific.
    - o However, several major steps are going to be similar for many projects of the businesses.

Step #1: Write down project scope of work

- The first step is to craft a scope statement and a general outline for the project. Creating the outline and the scope is usually the first step towards writing technical documentation and specs.
- Certainly, documentation will be changed in the project's scope, but initially, it is created by stakeholders and a project manager.
- The first step also contains developing the purpose and objectives of the project as well as the business justification for the project and is a basis for the entire future work.

Step #2: Organize deliverables into a work breakdown structure (WBS)

- When it comes to implementing project ideas effectively and promptly, there is a need to prepare deliverables.
- In turn, making deliverables is a part of the WBS that is the work breakdown structure for the project.
- A manager is the one to create the WBS according to the requirements of the customer, which need to include project scope, project tasks, and tasks split into simpler subtasks.

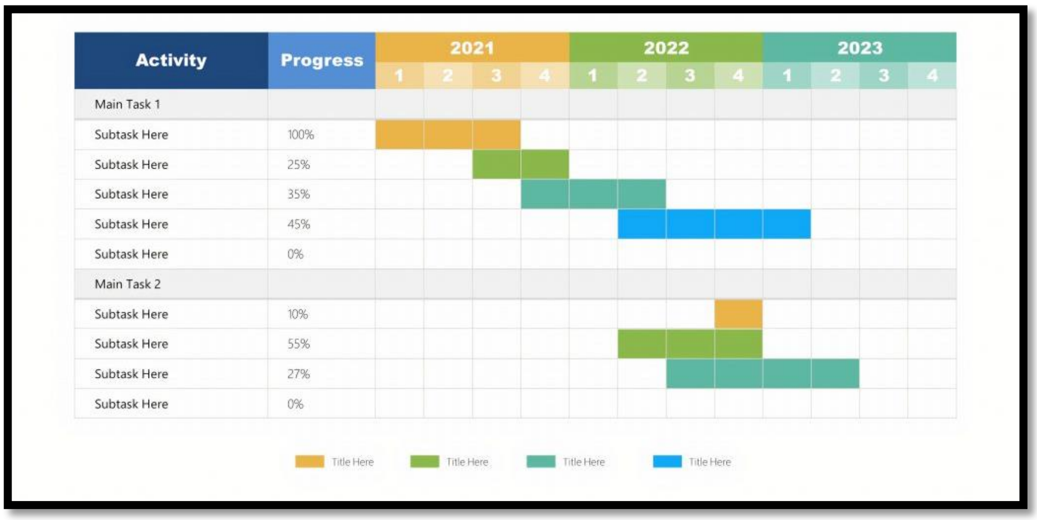
STEP #3: Get to know your dependencies

- When you already have the parts of your project to do with the WBS, it is time to create a to-do list for every subtask.
- At this point, many managers often ignore dependencies, and this might create bottlenecks in the project.
- To avoid bottlenecks, there is a need to plan tasks, taking in mind their dependencies on other tasks and subtasks

STEP #4: Create a timeline

- Gantt chart is a very convenient tool to use when it comes to planning the milestones and the dependencies.

- With a Gantt chart, you can plan time and tasks in a way that at any given moment you can see the status your project is in.
- The example of a Gantt chart is represented in the picture below.



<u>STEP #5 Plan budget</u>

- It goes without saying, before starting the project, there is a need to estimate the total cost that you can afford.
- Ideally, there should be several estimations, depending on the timeline and features added to the project.
- For example, you might want to have the lowest and the highest total cost of your project at hand so you can estimate the project from several points of view.
- Potentially, this helps to be more realistic with the budget and determine the features that are a must for the project and the ones that are auxiliary but desired.

2. **<u>Creating product Backlog</u>**
    - According to the official Scrum Guide, the product backlog is "...an ordered list of everything that is known to be needed in the product."
    - The product backlog sits outside of the sprint loop (meaning it contains work that will not be completed during the sprint) but informs how your sprint will be planned.
    - The product backlog is composed of feedback from:
        - ❖ The development team
        - ❖ Customer
        - ❖ Stakeholders

Follow these steps to start your scrum product backlog:

1. Add ideas to the backlog

Stakeholders will typically be approaching you with ideas for product improvements

2. Get clarification

Once you're approached by a stakeholder with a product addition or fix, make sure you understand:

- The reason behind the addition or fix
- The amount of value it contributes to the product as a whole
- The specifications of the item

3. Prioritize

- The backlog should have clearly defined, high-priority items at the top and vague items that are not a priority at the bottom.
- If an item has no value, it should not be added to the backlog.

4. Update the backlog regularly

- The backlog is a living document; make sure you're constantly prioritizing, refining, and keeping the backlog up to date.
- You may have hundreds of items in your backlog as ideas for product improvements are suggested.
- Some of these items may be discarded, but many of them will begin making their way up the backlog for further refinement and, ultimately, development.

3. **Creating User Stories**
    - A user story is a small unit of work in an Agile workflow.
    - It is a short, written explanation of a particular user's need and how it can be fulfilled.
    - User stories follow a simple template.
    - The chosen user story format will outline the "who," "what," and "why" of a particular requirement.
        - ❖ Who wants something?
        - ❖ What do they want?
        - ❖ Why do they want it?
    - The following template is one of the most common:

As [persona], I want to [action], so that I can [benefit]."

Step 1: Outline acceptance criteria

- The definition of done is the set of criteria that needs to be fulfilled for your user story to be considered complete.
- Define the specific acceptance criteria for each user story and use it as a checklist.

Step 2: Decide on user personas

- Conduct extensive user research by creating surveys, hosting focus groups, and reading user forums.
- Analyze your data and search for patterns to identify your key personas.

Step 3: Create tasks

- Break your user story down into numerous tasks to make it more manageable. If it is a complex requirement, you can also add subtasks.
- Include detailed descriptions, so your team is aligned on what each task requires.

Step 4: Map stories

- Use story mapping to structure work in a large process.
- In this case, your user stories will take the form of ordered steps.

Step 5: Request feedback

- Speak to users and potential customers to find out what they want.
- Ask them for their opinions on existing products or if they have suggestions for new features.
- Incorporate this feedback into your user story.

4. **User stories:**
   - **User Story 1 Summary: Customer registration functionality**
   - **Description:**
     - ❖ AS A customer
     - ❖ I WANT to have registration functionality
     - ❖ SO THAT I can successfully resist
   - **Scope:**
     - ❖ Build a registration page
     - ❖ Customer validation
     - ❖ Customer should be able to change the phone number
     - ❖ It should work in all the browser
     - ❖ It should also work in mobile
   - **Pre-condition:**
     - ❖ Customer should have email and phone number

- **Acceptance criteria:**
    - ❖ Scenario1: Customer can successfully resister
        1. "Given" I am on registration page
        2. "And" I give valid customer name and phone number
        3. "And" I check on sing in
        4. "Then" I will successfully resister
    - ❖ Scenario2: Customer cannot successfully resister
        1. "Given" I am on registration page
        2. "And" I give invalid customer name and phone number
        3. "Then" I will get an error message as "registration failed incorrect customer name"

- **User Story 2 Summary: Customer checking availability**
- **Description:**
    - ❖ AS A customer
    - ❖ I WANT to be able to check availability of hall
    - ❖ SO THAT I can check the available halls
- **Scope:**
    - ❖ Build an available checking page
    - ❖ It should be only inside the Karnataka
    - ❖ Customer should be able to check the available halls in their location
- **Pre-condition:**
    - ❖ Customer should have nearest halls in their location
- **Acceptance criteria:**
    - ❖ Scenario1: Customers can successfully check availability of hall in their location
        1. "Given" I am on check available of hall page
        2. "And" I give location and date
    - ❖ Scenario2: Customer can't successfully check availability of hall in their location
        1. "Given" I am on check available of hall page
        2. "And" I give wrong location
        3. "Then" I will get the error message as the location is invalid

- **Summary: Customer booking details**
- **Description:**

- ❖ AS A customer
- ❖ I WANT to block the hall
- ❖ SO THAT I can get the booking details

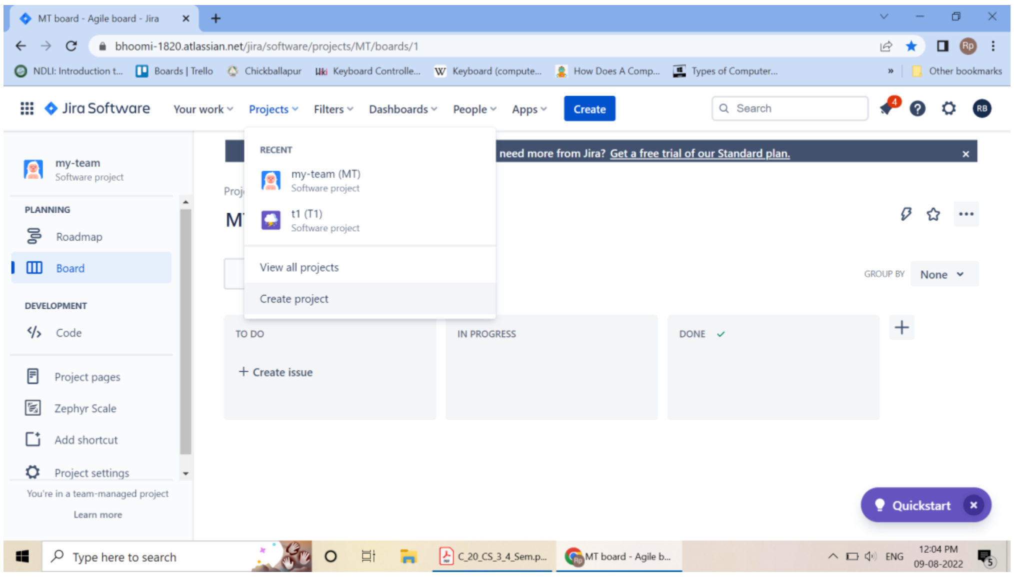
- **Scope**
  - ❖ build a booking details page
  - ❖ it should be able to see after the booking also
  - ❖ customer should be able to change details if their want
- **Pre-condition**
  - ❖ Customers must fill all the information given in the booking details
- **Acceptance criteria**
  - ❖ Scenario 1: customer can successfully get the booking details
    1. "Given" I am on the booing details page
    2. "And" I fill the details
    3. "And" I have also blocked the hall
    4. "Then" I will successfully get the booing details
  - ❖ Scenario 2: customer will not get the booking details
    1. "Given" I am on the booking details page
    2. "And" I will fill the details without blocking hall
    3. "Then" I will get an error message as the hall is not blocked yet

---

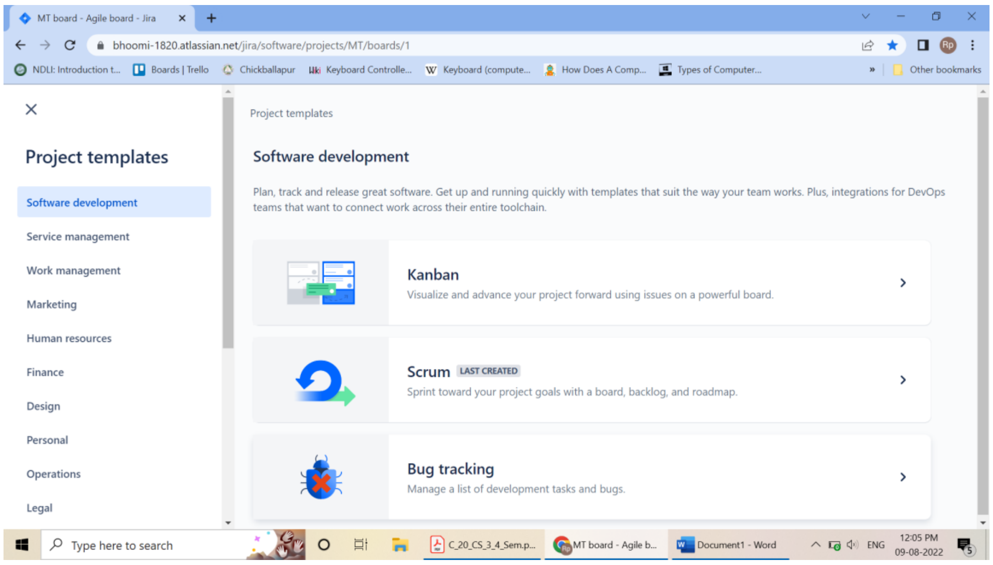
Experiment – 02

Create Sprint 1 with required user stories.

Step 1: Login to your Jira account and click on create project to create a new one.



Step 2: Choose your template.



Project templates

Software development

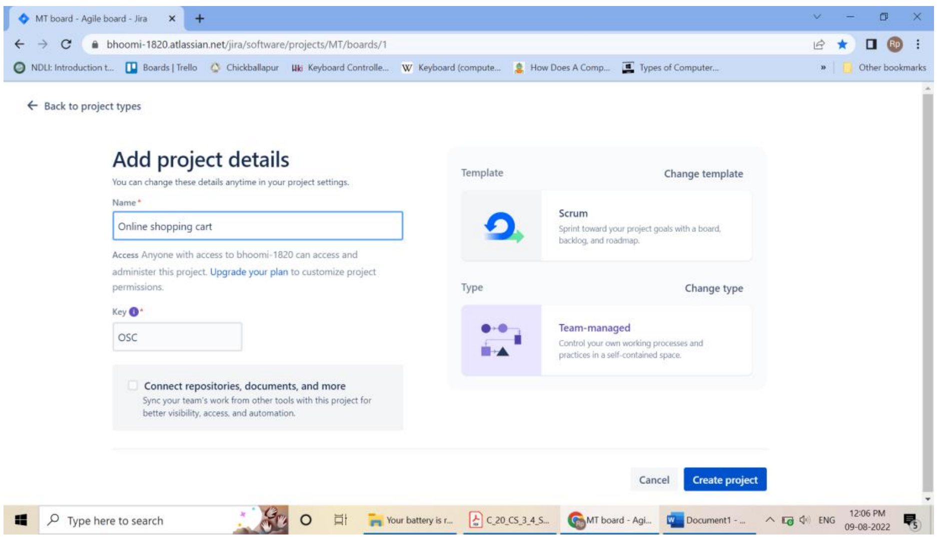
Plan, track and release great software. Get up and running quickly with templates that suit the way your team works. Plus, integrations for DevOps teams that want to connect work across their entire toolchain.

Kanban — Visualize and advance your project forward using issues on a powerful board.

Scrum — Sprint toward your project goals with a board, backlog, and roadmap.

Bug tracking — Manage a list of development tasks and bugs.

Step 3: Give a name to your project. And then click on create project.


MT board - Agile board - Jira

bhoomi-1820.atlassian.net/jira/software/projects/MT/boards/1

Back to project types

Add project details

You can change these details anytime in your project settings.

Name*
Online shopping cart

Access Anyone with access to bhoomi-1820 can access and administer this project. Upgrade your plan to customize project permissions.

Key*
OSC

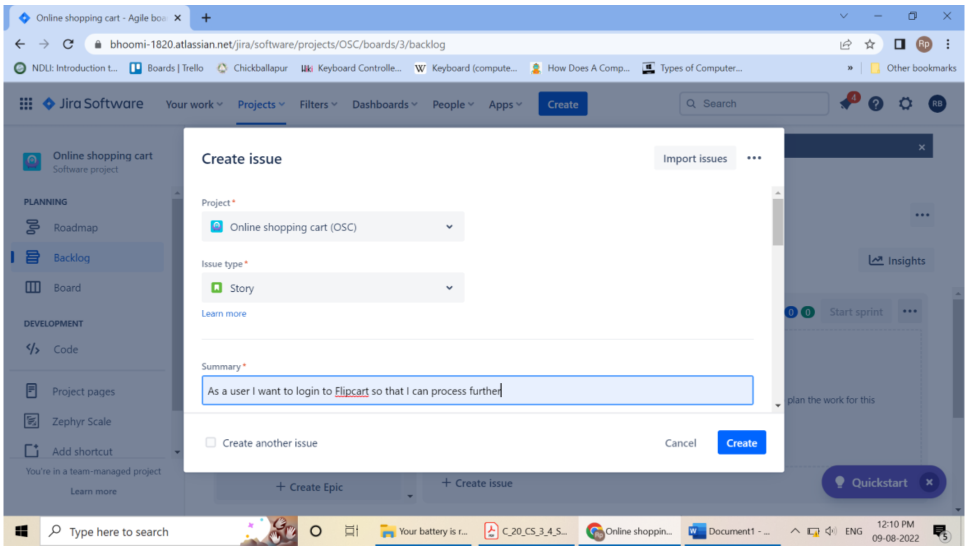
Connect repositories, documents, and more
Sync your team's work from other tools with this project for better visibility, access, and automation.

Template — Change template
Scrum
Sprint toward your project goals with a board, backlog, and roadmap.

Type — Change type
Team-managed
Control your own working processes and practices in a self-contained space.

Cancel — Create project

<u>Step 4: Now click on 'Create' to create issue type of Story. And give the summary of your user story.</u>


Online shopping cart - Agile board

bhoomi-1820.atlassian.net/jira/software/projects/OSC/boards/3/backlog

Create issue

Project*
Online shopping cart (OSC)

Issue type*
Story

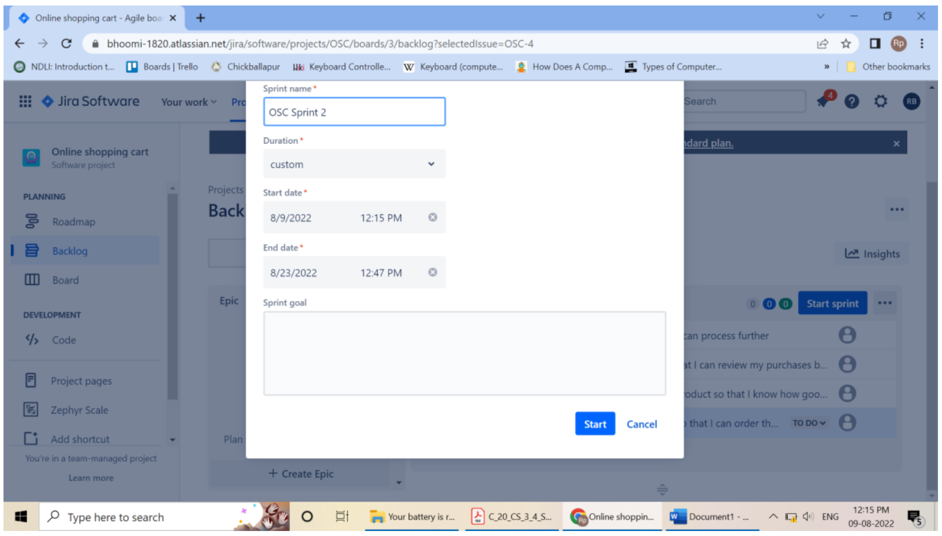
Learn more

Summary*
As a user I want to login to Flipcart so that I can process further

Create another issue

Cancel — Create

<u>Step 5: Create User Stories</u>


Online shopping cart - Agile board

bhoomi-1820.atlassian.net/jira/software/projects/OSC/boards/3/backlog?selectedIssue=OSC-4

Sprint name*
OSC Sprint 2

Duration*
custom

Start date*
8/9/2022 12:15 PM

End date*
8/23/2022 12:47 PM

Sprint goal

Start — Cancel

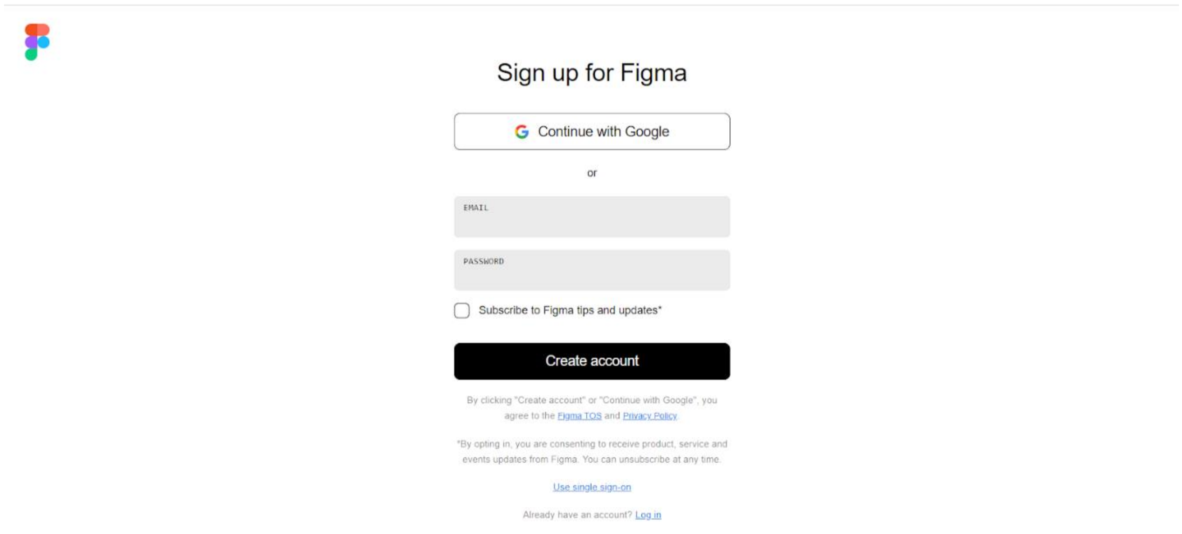
- Sprint1 is created on clicking on start

- Give a summary to your project.
- Now write a user story in Description box.
- Your story will then go into the backlog to be Assigned and auctioned by the project manager, product owner or other relevant stakeholders and click on start sprint
- Click on Board and select Insights
- Click on Insights and click “Sprint burn down” And click on Learn more.

Experiment – 03

Create UI/UX design - for created user stories (wire framing).

1. Continue with your Gmail account or login to Figma.



Sign up for Figma

Continue with Google

or

EMAIL

PASSWORD

Subscribe to Figma tips and updates*

Create account

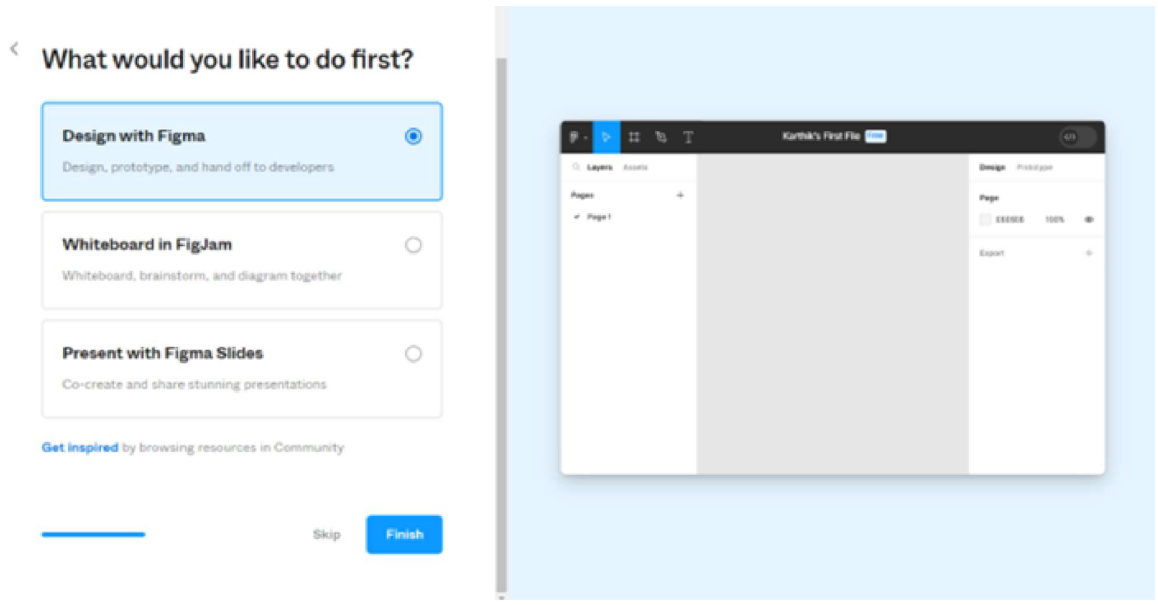
By clicking "Create account" or "Continue with Google", you agree to the Figma TOS and Privacy Policy.

*By opting in, you are consenting to receive product, service and events updates from Figma. You can unsubscribe at any time.

Use single sign-on

Already have an account? Log in

2. First create design file



What would you like to do first?

Design with Figma [✓]
Design, prototype, and hand off to developers

Whiteboard in FigJam
Whiteboard, brainstorm, and diagram together

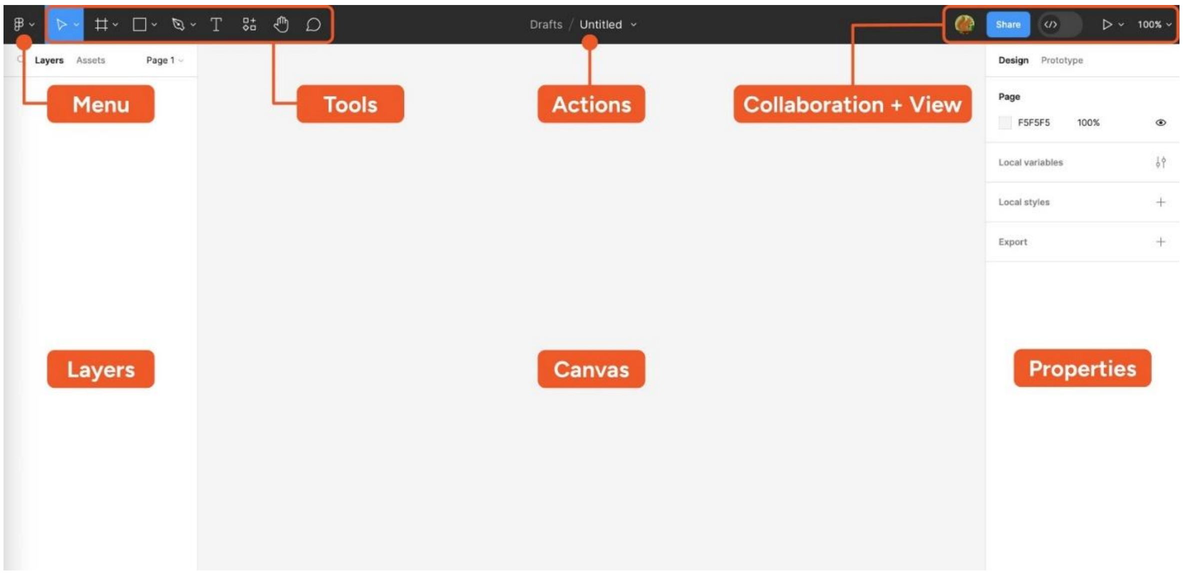
Present with Figma Slides
Co-create and share stunning presentations

Get inspired by browsing resources in Community

Skip

Finish

3. And adding elements to over design file from Figma community



4. Click on **"#"** button on the tool menu at the (Top left)
5. Depends on which size you want to use choose the screen size from the right sidebar.
6. Add background colour to the frame by clicking it and add colour from the "Fill" section in the (right panel).
7. Create text button (click on "T" text button from the Top left)
8. Click on rectangle **"☐"** button to select image from the popup menu at the (Top left)

Experiment – 04

Create repository – named mini project 1; Push and pull operation in GitHub.

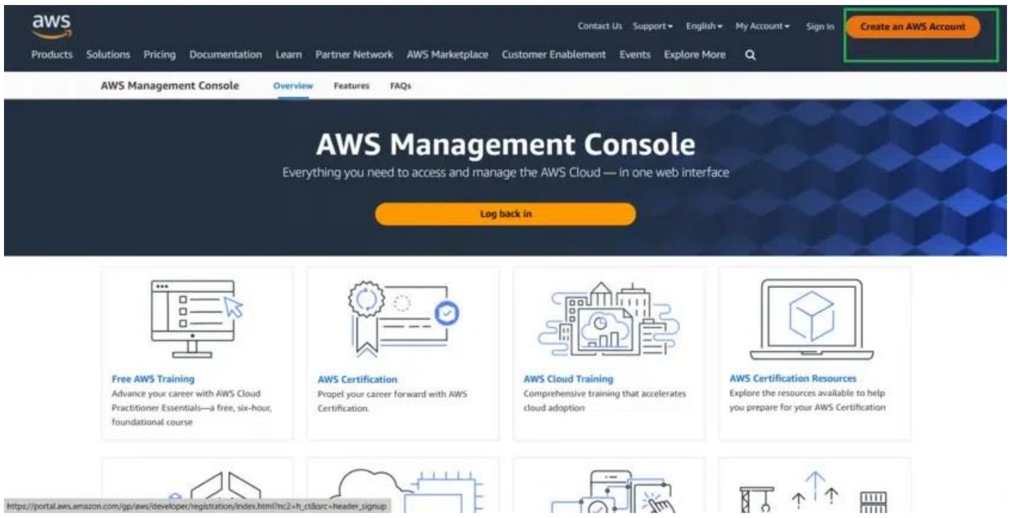
1. Browse to the official Git website: https://git-scm.com/downloads
2. Click the download link for Windows and allow the download to complete.
3. Double-click the file to extract and launch the installer.

- **Git operations**
  - ❖ Creating a repository
  - ❖ Open browser, search for GitHub Login.
  - ❖ Sign in with your username and password
  - ❖ In the upper-right corner, use the drop-down menu, and select **New repository**.
  - ❖ Give a name for your repository. For example, "hello-world".
  - ❖ Add a description of your repository. For example, "Mini Project I"
  - ❖ Click **Create repository**.
- **Push Operation:**
  - ❖ Go to add files and select upload files.
  - ❖ Choose your files then select a file or folder click on open.
  - ❖ Click on commit changes.
- **Clone or pull operation:**
  - ❖ Click on code dropdown button.
  - ❖ Click on Download Zip.

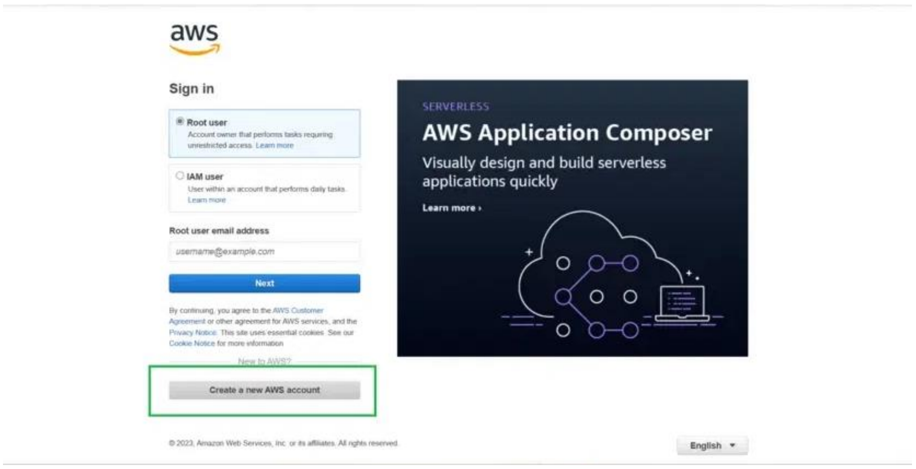
Experiment – 05

Create an account in Amazon Web Services (AWS)

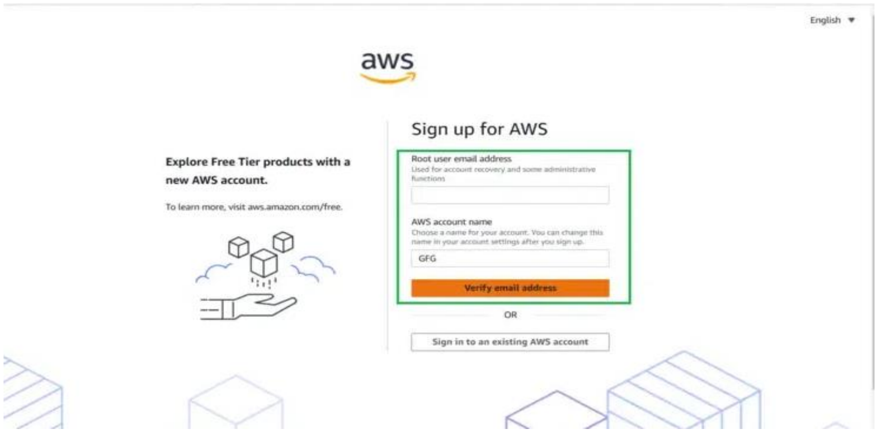
- **Step 1:** To create an account in AWS, go to AWS console and click on the sign-up button.


aws
Contact Us
Support
English
My Account
Sign In
Create an AWS Account
Products
Solutions
Pricing
Documentation
Learn
Partner Network
AWS Marketplace
Customer Enablement
Events
Explore More
AWS Management Console
Overview
Features
FAQs
AWS Management Console
Everything you need to access and manage the AWS Cloud — in one web interface
Log back in
Free AWS Training
Advance your career with AWS Cloud Practitioner Essentials—a free, six-hour, foundational course
AWS Certification
Propel your career forward with AWS Certification.
AWS Cloud Training
Comprehensive training that accelerates cloud adoption
AWS Certification Resources
Explore the resources available to help you prepare for your AWS Certification
https://portal.aws.amazon.com/gp/aws/developer/registration/index.html?nc2=h_ct&src=header_signup

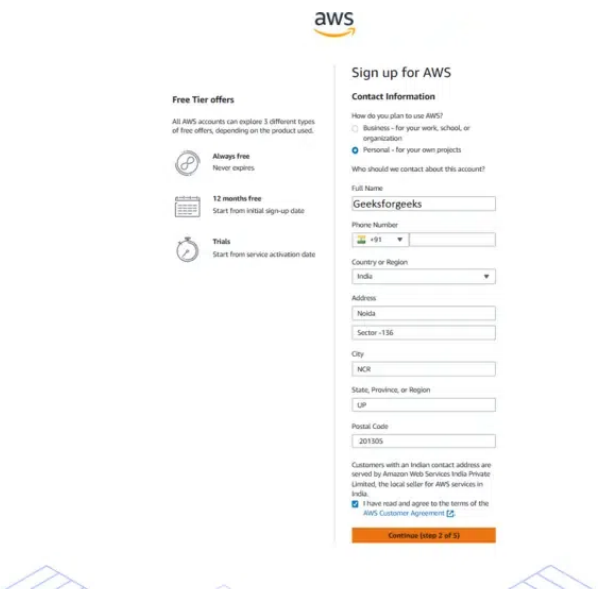
- **Step 2:** After that click on create new account.


aws
Sign in
Root user
Account owner that performs tasks requiring unrestricted access. Learn more
IAM user
User within an account that performs daily tasks. Learn more
Root user email address
username@example.com
Next
By continuing, you agree to the AWS Customer Agreement or other agreement for AWS services, and the Privacy Notice. This site uses essential cookies. See our Cookie Notice for more information.
New to AWS?
Create a new AWS account
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.
English
SERVERLESS
AWS Application Composer
Visually design and build serverless applications quickly
Learn more

- **Step 3:** Enter your e-mail address and account name, verify your account.


English
aws
Explore Free Tier products with a new AWS account.
To learn more, visit aws.amazon.com/free.
Sign up for AWS
Root user email address
Used for account recovery and some administrative functions
AWS account name
Choose a name for your account. You can change this name in your account settings after you sign up.
GFG
Verify email address
OR
Sign in to an existing AWS account

- **Step 4:** After verification, click on 'Next' and enter.
- **Step 5:** In the next step we must provide some information about ourselves like username, full name, phone number, city, etc for verification purposes. Once done, click on 'Continue'.



Free Tier offers

All AWS accounts can explore 3 different types of free offers, depending on the product used.

Always free
Never expires

12 months free
Start from initial sign-up date

Trials
Start from service activation date

Sign up for AWS

Contact Information

How do you plan to use AWS?
Business - for your work, school, or organization [✓ Personal - for your own projects]

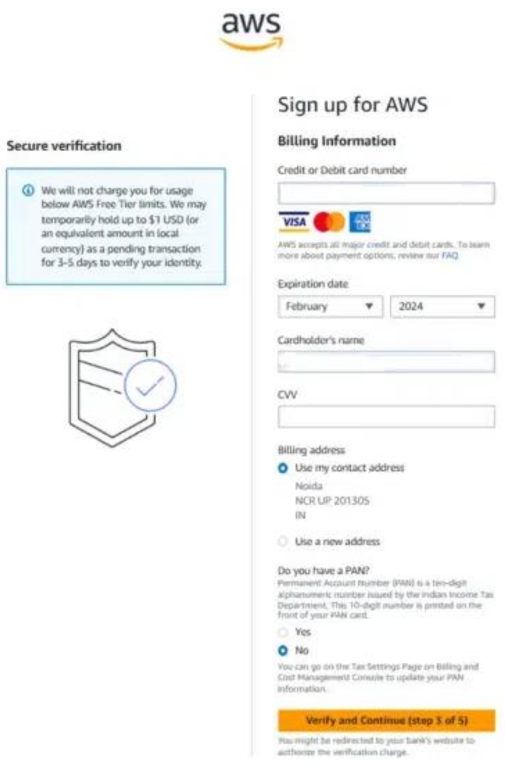
Who should we contact about this account?

Full Name	Geeksforgeeks
Phone Number	+91
Country or Region	India
Address	Noida
	Sector -136
City	NCR
State, Province, or Region	UP
Postal Code	201305

Customers with an Indian contact address are served by Amazon Web Services India Private Limited, the local seller for AWS services in India.

✓ I have read and agree to the terms of the AWS Customer Agreement.

- **Step 6:** In the next step we must provide our card details for the payment as it is mandatory.



Secure verification

We will not charge you for usage below AWS Free Tier limits. We may temporarily hold up to $1 USD (or an equivalent amount in local currency) as a pending transaction for 3-5 days to verify your identity.

Sign up for AWS

Billing Information

Credit or Debit card number

AWS accepts all major credit and debit cards. To learn more about payment options, review our FAQ.

Expiration date: February 2024

Cardholder's name

CVV

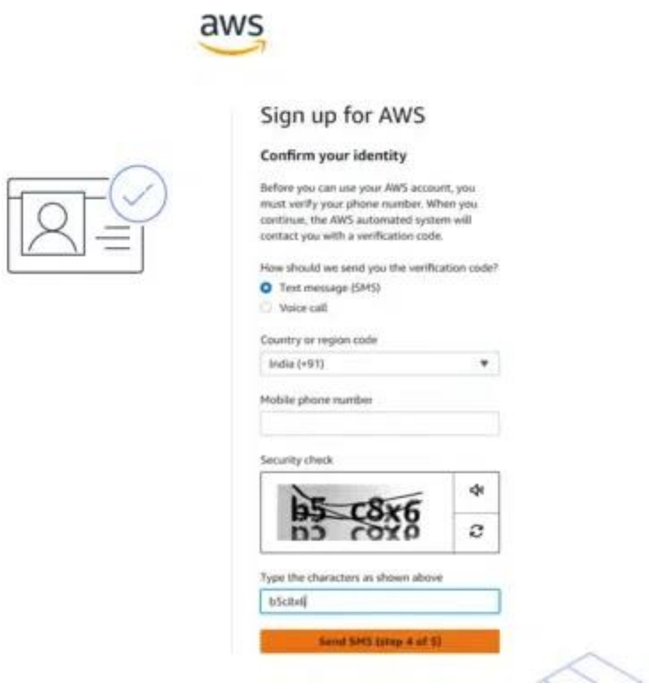
Billing address
[✓ Use my contact address]
Noida
NCR UP 201305
IN
Use a new address

Do you have a PAN?
Permanent Account Number (PAN) is a ten-digit alphanumeric number issued by the Indian Income Tax Department. This 10-digit number is printed on the front of your PAN card.
Yes [✓ No]

You can go on the Tax Settings Page on Billing and Cost Management Console to update your PAN information.

You might be redirected to your bank's website to authorize the verification charge.

- **Step 7:** After that you must confirm your identity either via text or voice call.



Sign up for AWS

Confirm your identity

Before you can use your AWS account, you must verify your phone number. When you continue, the AWS automated system will contact you with a verification code.

How should we send you the verification code?
[✓ Text message (SMS)]
Voice call

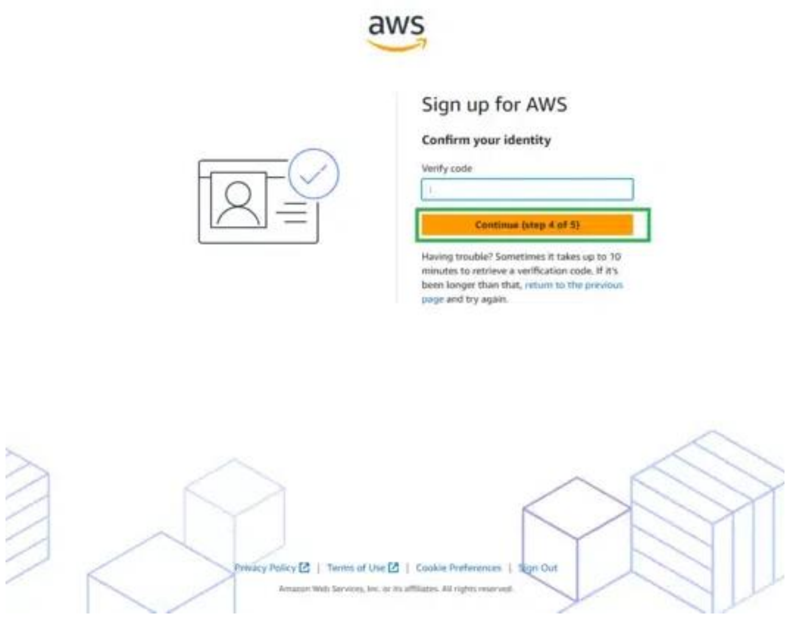
Country or region code: India (+91)

Mobile phone number:

Security check

Type the characters as shown above: b5citxl

- **Step 8:** After confirming your identity, go with the free tier and sign out.



Sign up for AWS

Confirm your identity

Verify code:

Having trouble? Sometimes it takes up to 10 minutes to retrieve a verification code. If it's been longer than that, return to the previous page and try again.

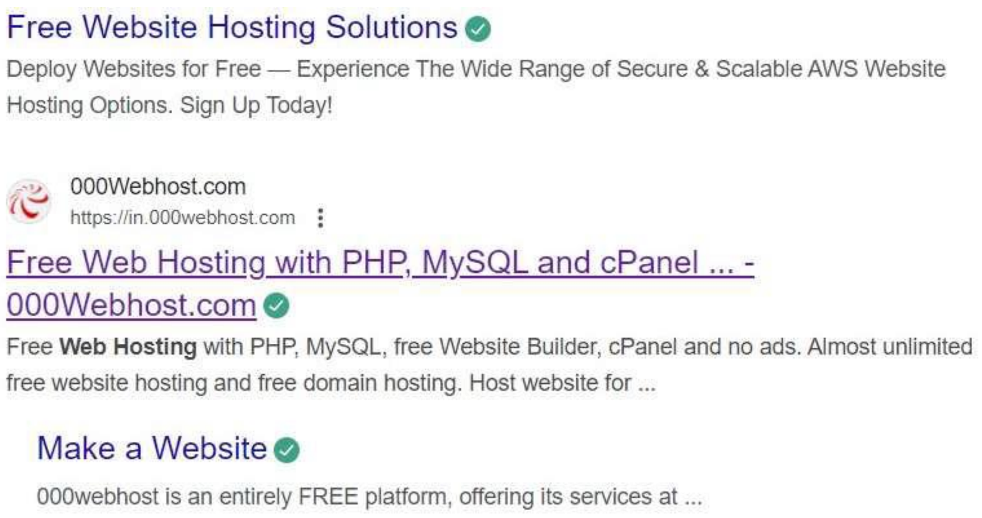
Privacy Policy | Terms of Use | Cookie Preferences | Sign Out

Amazon Web Services, Inc. or its affiliates. All rights reserved.

Experiment – 06

Creating an account and project in 000.webhost.com (from cloud service).

1. Open web browser and search for free cloud service (000webhost.com).



Free Website Hosting Solutions

Deploy Websites for Free — Experience The Wide Range of Secure & Scalable AWS Website Hosting Options. Sign Up Today!

000Webhost.com
https://in.000webhost.com

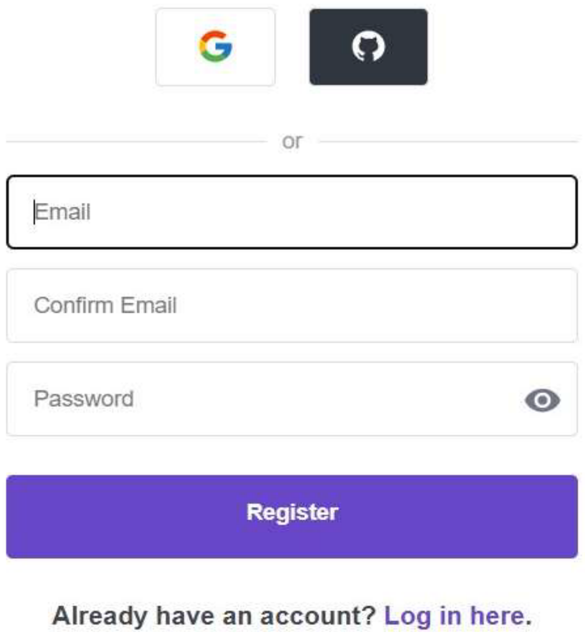
Free Web Hosting with PHP, MySQL and cPanel ... - 000Webhost.com

Free **Web Hosting** with PHP, MySQL, free Website Builder, cPanel and no ads. Almost unlimited free website hosting and free domain hosting. Host website for ...

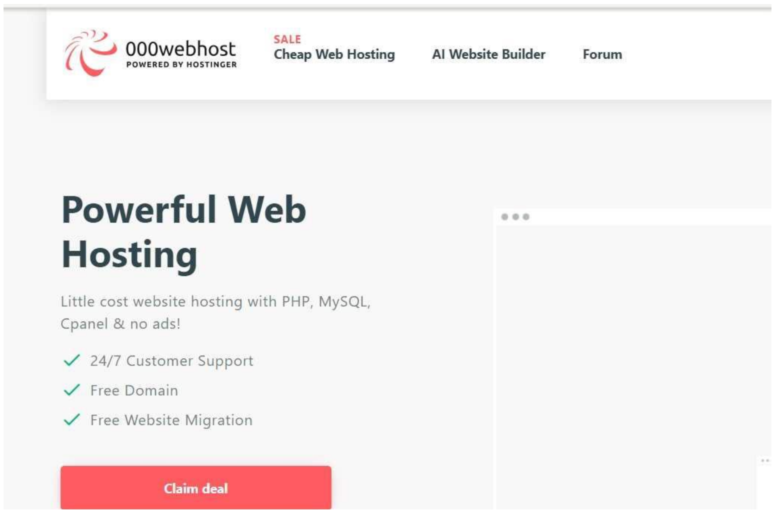
Make a Website

000webhost is an entirely FREE platform, offering its services at ...

2. Proceed sign in process through the Google account.



3. Host free website and click on Manage Website.



Powerful Web Hosting

Little cost website hosting with PHP, MySQL, Cpanel & no ads!

- 24/7 Customer Support
- Free Domain
- Free Website Migration

4. Towards left column, tools>>file manager and select upload files.
5. Select public_html and upload the existing html file.
6. After uploading, right click on the file and open.
7. Make sure the file has been uploaded.
8. Finally, we can perform the following operations
    - View the output of the webpage application
    - Download the webpage application
    - Share the webpage application through the given below
        - ➢ Webhost_link + file_name.extension
    - We can move from one directory to another.
    - Either we can copy the content of the file or by copying the whole file

Experiment – 07

Create a form like Registration form or feedback form, after submit hide create form and enable the display section using JavaScript.

Registration.html

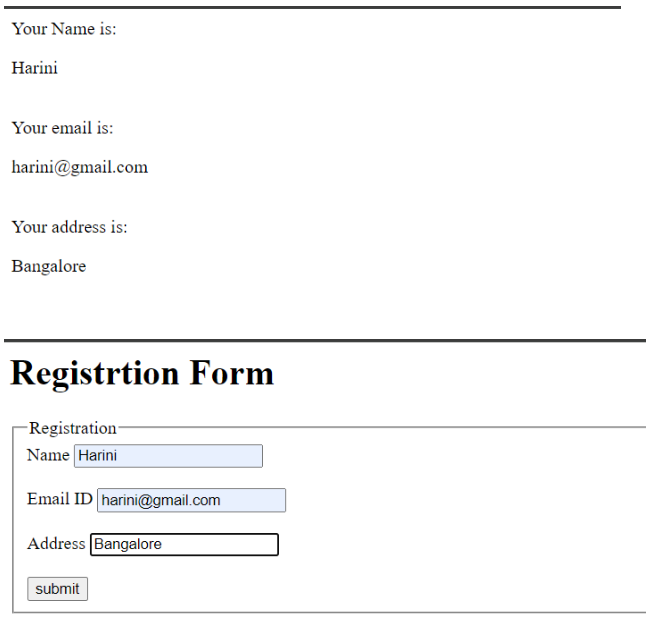
```
<html>
<head>
        <title> Registration Form</title>
        <script>
            function passvalues()
    {
                    var name = document.getElementById("name").value;
                    var email = document.getElementById("email").value;
                    var address = document.getElementById("address").value;
                    localStorage.setItem("name",name);
                    localStorage.setItem("email",email);
                    localStorage.setItem("address",address);
                    return;
            }
        </script>
    </head>
    <body>
<h1>Registrtion Form</h1>
        <form action="Details.html">
<fieldset>
    <legend>Registration</legend>
 <label> Name </label>
        <input type="text" id="name"/><br><br>
 <label> Email ID </label>
        <input type="email" id="email"/><br><br>
 <label> Address </label>
        <input type="address" id="address"/><br><br>
        <input type="submit" value="submit" onclick="passvalues()"/>
</fieldset>
        </form>
    </body>
</html>
```

Details.html

```
<html>
    <head>
        <title> Details</title>
    </head>
    <body>
<form>
    Your Name is:<p id="name"></p><br>
        Your email is:<p id="email"></p><br>
        Your address is:<p id="address"></p>
<script>
            document.getElementById("name").innerHTML =
localStorage.getItem("name");
            document.getElementById("email").innerHTML =
localStorage.getItem("email");
            document.getElementById("address").innerHTML =
localStorage.getItem("address");
```

```
        </script>
</form>

    </body>
</html>
```

Output:



```
Your Name is:

Harini

Your email is:

harini@gmail.com

Your address is:

Bangalore
```

Registrtion Form

Registration

Name	Harini
Email ID	harini@gmail.com
Address	Bangalore

Experiment – 08

Create form validation using JavaScript.

Index.html

```
<html>
<body>
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;
if (name==null || name=="")
{
  alert("Name can't be blank");
  return false;
}
else if(password.length<6)
{
  alert("Password must be at least 6 characters long.");
  return false;
  }
}
</script>
<body>
<form name="myform" method="post" action="valid.html"
onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
</body>
</html>
```

valid.html

```
<html>
<body>
<h1>Validation Successfull</h1>
</body>
</html>
```

Experiment – 09

Setting up an Environment and tools for frontend development such as installing VS Code and installing required extensions.

Step 1: Open browser and go to visual studio code official website and download the windows exe file.

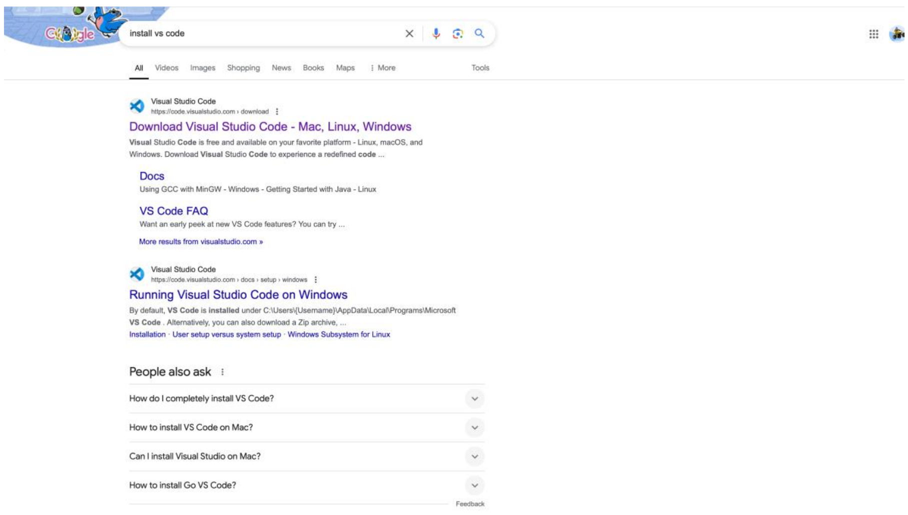
Step 2: Open The downloads folder and execute the downloaded file.

Step 3: Accept the license agreement and click on Next.

Step 4: Select the installation directory and click on Next.

Step 5: Keep clicking on Next keeping the setting default. And finally click on Next.

Step 6: Once the installation finish launch VS Code and go extensions to download the following extension – ESLint



Visual Studio Code
https://code.visualstudio.com › download

Download Visual Studio Code - Mac, Linux, Windows

Visual Studio **Code** is free and available on your favorite platform - Linux, macOS, and Windows. Download **Visual** Studio **Code** to experience a redefined **code** ...

Docs
Using GCC with MinGW - Windows - Getting Started with Java - Linux

VS Code FAQ
Want an early peek at new VS Code features? You can try ...

More results from visualstudio.com »

Visual Studio Code
https://code.visualstudio.com › docs › setup › windows

Running Visual Studio Code on Windows

By default, **VS Code** is **installed** under C:\Users\{Username}\AppData\Local\Programs\Microsoft **VS Code** . Alternatively, you can also download a Zip archive, ...
Installation · User setup versus system setup · Windows Subsystem for Linux

People also ask

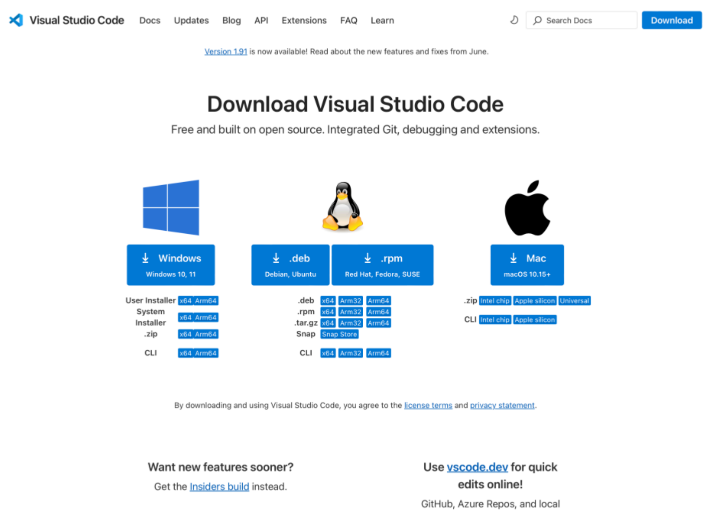
How do I completely install VS Code?

How to install VS Code on Mac?

Can I install Visual Studio on Mac?

How to install Go VS Code?

Feedback



Version 1.91 is now available! Read about the new features and fixes from June.

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

User Installer	x64 Arm64
System Installer	x64 Arm64
.zip	x64 Arm64
CLI	x64 Arm64

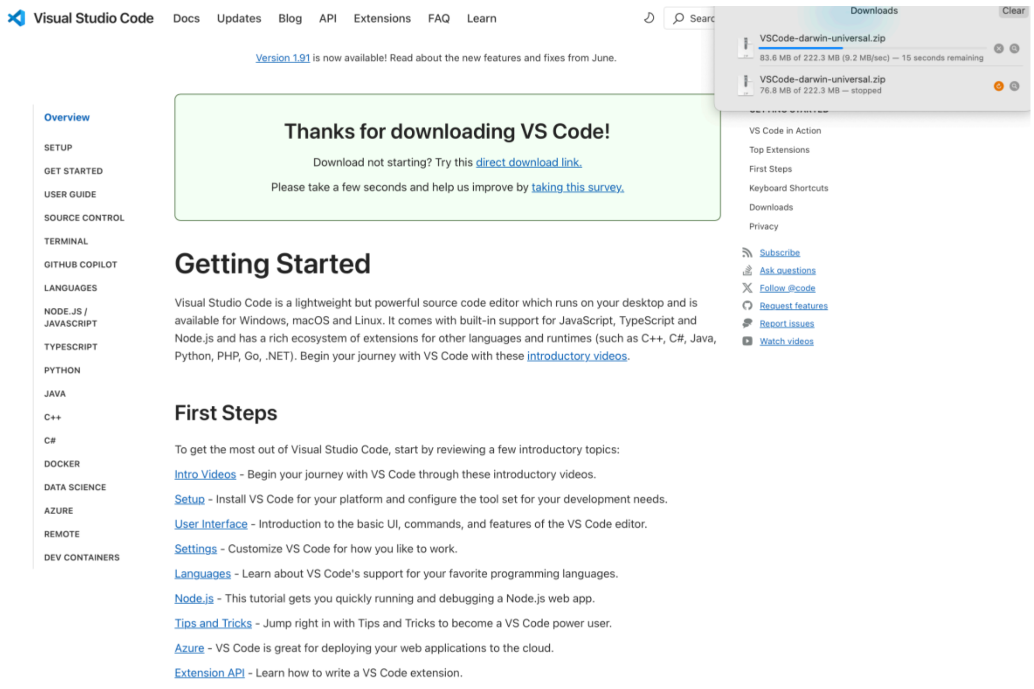
.deb	x64 Arm32 Arm64
.rpm	x64 Arm32 Arm64
.tar.gz	x64 Arm32 Arm64
Snap	Snap Store
CLI	x64 Arm32 Arm64

.zip	Intel chip Apple silicon Universal
CLI	Intel chip Apple silicon

By downloading and using Visual Studio Code, you agree to the license terms and privacy statement.

Want new features sooner?
Get the Insiders build instead.

Use vscode.dev for quick edits online!
GitHub, Azure Repos, and local



Visual Studio Code — Docs — Updates — Blog — API — Extensions — FAQ — Learn

Downloads — Clear

VSCode-darwin-universal.zip
83.6 MB of 222.3 MB (9.2 MB/sec) — 15 seconds remaining

VSCode-darwin-universal.zip
76.8 MB of 222.3 MB — stopped

Version 1.91 is now available! Read about the new features and fixes from June.

Overview
SETUP
GET STARTED
USER GUIDE
SOURCE CONTROL
TERMINAL
GITHUB COPILOT
LANGUAGES
NODE.JS / JAVASCRIPT
TYPESCRIPT
PYTHON
JAVA
C++
C#
DOCKER
DATA SCIENCE
AZURE
REMOTE
DEV CONTAINERS

Thanks for downloading VS Code!

Download not starting? Try this direct download link.

Please take a few seconds and help us improve by taking this survey.

Getting Started

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET). Begin your journey with VS Code with these introductory videos.

First Steps

To get the most out of Visual Studio Code, start by reviewing a few introductory topics:

Intro Videos - Begin your journey with VS Code through these introductory videos.

Setup - Install VS Code for your platform and configure the tool set for your development needs.

User Interface - Introduction to the basic UI, commands, and features of the VS Code editor.

Settings - Customize VS Code for how you like to work.

Languages - Learn about VS Code's support for your favorite programming languages.

Node.js - This tutorial gets you quickly running and debugging a Node.js web app.

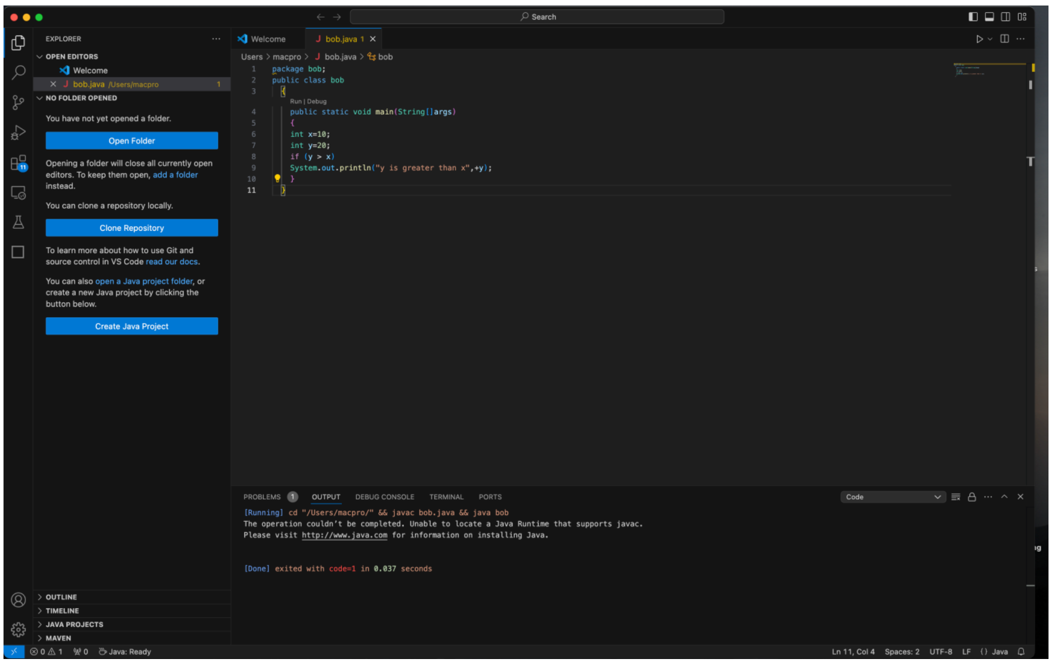
Tips and Tricks - Jump right in with Tips and Tricks to become a VS Code power user.

Azure - VS Code is great for deploying your web applications to the cloud.

Extension API - Learn how to write a VS Code extension.

VS Code in Action
Top Extensions
First Steps
Keyboard Shortcuts
Downloads
Privacy

Subscribe
Ask questions
Follow @code
Request features
Report issues
Watch videos



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

To learn more about how to use Git and source control in VS Code read our docs.

You can also open a Java project folder, or create a new Java project by clicking the button below.

```java
package bob;
public class bob
  {
    public static void main(String[]args)
    {
    int x=10;
    int y=20;
    if (y > x)
    System.out.println("y is greater than x",+y);
    }
  }
```

```
[Running] cd "/Users/macpro/" && javac bob.java && java bob
The operation couldn't be completed. Unable to locate a Java Runtime that supports javac.
Please visit http://www.java.com for information on installing Java.


[Done] exited with code=1 in 0.037 seconds
```

Experiment – 10

Create and run simple JavaScript programs.

JavaScript functions

- **Function.html**

```
<!DOCTYPE html>
<html>
<body>
<h1> Java script Function </h1>
<p> The product of 2 no: are </p>
<p id = "demo"> </p>
<script>
function myfunction (p1, p2)
{
  return p1 * p2
}
let result = myfunction (4, 3);
document.getElementById("demo").innerHTML=result;
</script>
</body>S
</html>
```

- **Arrow.html**

```
<!DOCTYPE html>
<html>
<head>
<title> Document </title>
<head>
<body>
<h1> Java Script Function </h1>
<h2> The Arrow Function </h2>
<p> message is: </p>
<p id="demo"> </p>
<script>
const message = () => "Hello world";
document.getElementById("demo").innerHTML=message();
</script>
</body>
<html>
```

- **Prompt.html**

```
<!DOCTYPE html>
<html>
<head>
<title>Document</title>
</head>
```

```
<body>
<script>
let age = prompt("How old are you?", 50);
alert("you are " + age + " year old");
</script>
</body>
</html>
```

- **Object.html**

```
<!DOCTYPE html>
<html>
<body>
<h2>Java script object</h2>
<p> using an object constructor: </p>
<p id="demo"> </p>
<script>
function Person (first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColour = eye;
}
const myFather = new Person("John", "Doe", 50, "blue");
const myMother = new Person("Sally", "Rally", 48, "green");
document.getElementById("demo").innerHTML =
"My father age is: " + myFather.age + "<br>" + "My mother age
is : " + myMother.age;
</script>
</body>
</html>
```

Experiment – 11

Create and run simple TypeScript programs.

Install TypeScript using Node.js Package Manager (npm)

Step-1 Install Node.js. It is used to setup TypeScript on our local computer.

To install Node.js on Windows, go to the following link: https://www.javatpoint.com/install-nodejs

Step-2 Install TypeScript. To install TypeScript, enter the following command in the Terminal Window.

- npm install typescript --save-dev   //As dev dependency
- npm install typescript -g   //Install as a global module

or

- npm install -g typescript
- npm install typescript@latest -g   //Install latest if you have an older version

Step-3 To verify the installation was successful, enter the command **$ tsc -v** in the Terminal Window.

Install Live server

npm install -g live-server

Create and run first program in TypeScript

1. open command prompt
2. go to d: drive(any drive)
3. d:\>**mkdir typescript**
4. d:\>**cd typescript**
5. d:\typescript>**npm install typescript --save-dev**
6. open Visual studio code
7. file-open folder-choose typescript folder from d:
8. create new file- save it as types.ts (any name.ts)
9. Write the below code and save it

10. console.log("Hello World");
11. In Visual studio code and compile the program
12. **tsc.cmd types.ts**
13. run the program
14. and retype this **node types.js** and compile to get the output
15. Observe the output

- **Types.ts**

```
let arr: number[];
arr = [1, 2, 3, 4];
console.log("Array [0]: " + arr[0]);
console.log("Array [1]: " + arr[1]);
```

To compile run the below 2 steps / codes in vsc terminal

- tsc.cmd types.ts
- node types.ts

- **Program1.ts**

```
function addTwo(a: number, b: number): number {
  return a + b;
}
console.log(addTwo(7, 8));
```

- **Program2.ts**

```
let a: number = 50;
if (a > 100) {
  console.log("a is greater than 100");
} else {
  console.log("a is lesser than 100");
}
```

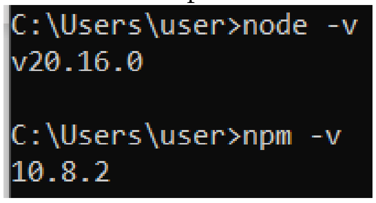
Experiment – 12

Setting up a React environment – installing Node.js; create, read and run a React application.

Step 01: Open the browser and go to nodejs.org.
Step 02: Go to downloads and download the latest MSI installer for your system.
Step 03: Open the downloads folder on your system and run the MSI file.
Step 04: Accept the license T&Cs and click on 'Next'.
Step 05: Select the installation directory and click on 'Install'.

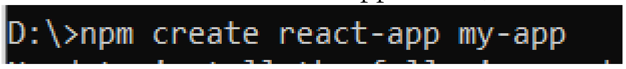
Step 06: Once installed, verify the installation with the following commands:

node -v & npm -v



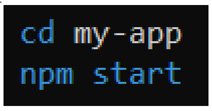
```
C:\Users\user>node -v
v20.16.0

C:\Users\user>npm -v
10.8.2
```

Step 07: To create the React application, execute the following command:



```
D:\>npm create react-app my-app
```

Step 08: To run the react application, execute the following commands:



```
cd my-app
npm start
```

Experiment – 13

Using expressions, specifying attributes, and rendering elements in React.

- By using JSX, we can use attributes with HTML elements. JSX uses the camelCase convention to specify the attributes.
- The ‘index.js’ and ‘app.js’ files come in-built along with the React application installed earlier in the previous experiment. In order to access those files and modify them, open the application with VS Code or any other IDE and navigate to the ‘src’ folder.

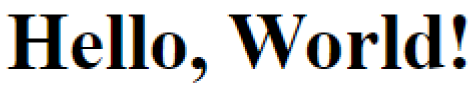
In the **index.js** file:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const root =
ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

In the **app.js** file:

```
import React, { Component } from 'react';
class App extends Component{
  render(){
    return <h1>Hello, World!</h1>
  }
}
export default App;
```

Output:



```
Hello, World!
```

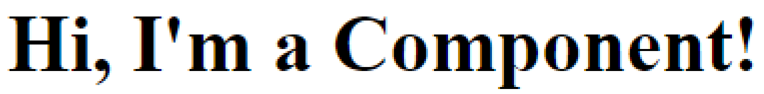
Experiment – 14

Using components, state, and props in React.

Component (Functional Component):

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function Comp(){
  return <h1>Hi, I'm a Component!</h1>
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Comp/>);
```

Output:

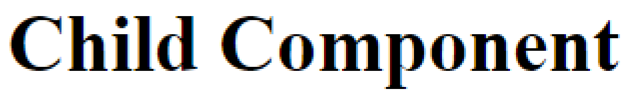


Hi, I'm a Component!

Props:

```
import React from 'react';
import ReactDOM from 'react-dom/client';

class ParentComponent extends React.Component{
  render(){
    return(<ChildComponent name = "Child Component"/>)
  }
}
const ChildComponent = (props) => {return(<h1>{props.name}</h1>)}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<ParentComponent/>);
```

Output:



Child Component

State:

```
import React from 'react';
import ReactDOM from 'react-dom/client';

class Test extends React.Component{
  constructor(props){
    super(props);
    this.state = {id: 1, name: "test"}
  }
  render(){
    return(
      <div>
        <h1>{this.state.id}</h1>
        <h1>{this.state.name}</h1>
      </div>
    )
  }
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Test/>);
```

Output:

```
1

test
```

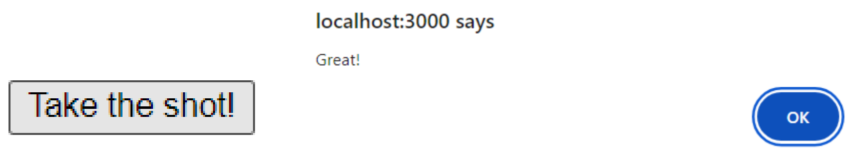
Experiment – 15

Using Lists and Keys and passing Arguments.

Passing Arguments: (Using Arrow Functions)

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function Football(){
  const shoot = (a) => {alert(a);}
  return(<button onClick={() => shoot('Great!')}>Take the
shot!</button>)
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Football/>);
```

Output:


```
localhost:3000 says
Great!
```

Components:

Rendering Multiple Components → Lists & Keys

You can build collections of elements and include them in JSX using curly braces.

- Lists are used to display the data in an ordered format and is traversed using the map() function.
- Lists are a continuous group of text or messages.

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function NumberList (props){
  const numbers = props.numbers;
  const listItems = numbers.map((numbers) => <li>{numbers}</li>)
  return <ul>{listItems}</ul>
}
const numbers = [10, 20, 30, 40, 50]

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<NumberList numbers = {numbers} />);
```

Output:

- 10
- 20
- 30
- 40
- 50

Keys:

Keys are used to identify the items that have been changed, added, or removed. They should be given inside the array to give the elements a stable entity.

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function Keys (props){
  const keys = props.keys;
  return (
    <div>
      <ul>{
          keys.map((keys) => <li key={keys.toString()}>{keys}</li>)
        }
      </ul>
    </div>
  )
}
const keys = ["Bengaluru", "Indore", "Mumbai", "Chennai"]

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Keys keys = {keys} />);
```

Output:

- Bengaluru
- Indore
- Mumbai
- Chennai

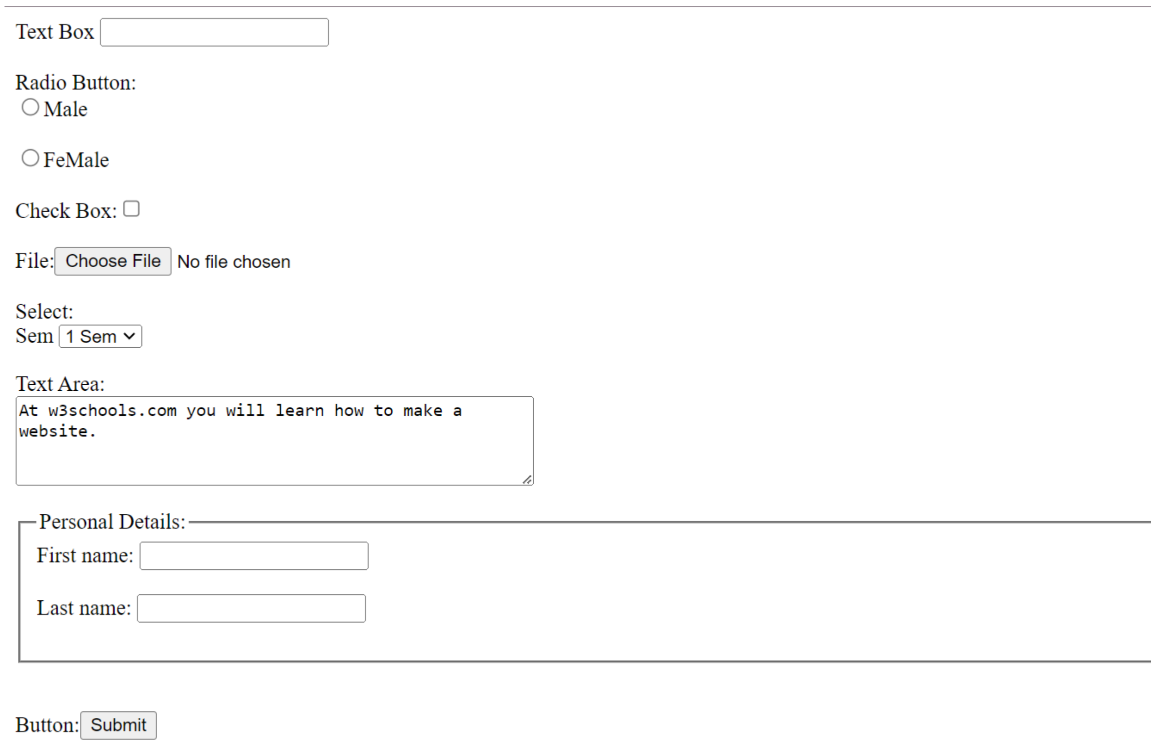
Experiment – 16

Forms - Use of HTML tags in forms like select, input, file, textarea, etc.

Form.html

```
<html>
<head>
<title>Form Elements</title>
</head>
<body>
<form>
<lable>Text Box</lable>
<input type="text" id="t1" name="name" value=""/><br><br>
Radio Button:     <br>
<input type="radio" id="r1" name="" value=""/>Male<br>      <br>
<input type="radio" id="r1" name="" value=""/>FeMale<br><br>
Check Box:<input type="checkbox" id="c1" name="" value=""/><br><br>
File:<input type="file" id="e1" name="file" value=""/><br><br>
Select:<br>
<label>Sem</label>
<select name="sem" id="sem">
  <option value="1">1 Sem</option>
  <option value="2">2 Sem</option>
</select><br><br>
Text Area:<br>
<textarea id="ta1" name="textarea" rows="4" cols="50">
At w3schools.com you will learn how to make a website.
</textarea><br><br>
<fieldset>
    <legend>Personal Details:</legend>
    <label>First name:</label>
    <input type="text" id="fname" name="fname"><br><br>
    <label>Last name:</label>
    <input type="text" id="lname" name="lname"><br><br>
  </fieldset><br><br>
Button:<input type="button" id="t1" name="" value="Submit"/><br>
</form>
</body>
</html>
```

Output:



Text Box

Radio Button:
○ Male

○ FeMale

Check Box: ☐

File: Choose File No file chosen

Select:
Sem 1 Sem

Text Area:
```
At w3schools.com you will learn how to make a
website.
```

Personal Details:

First name:

Last name:

Button: Submit

Experiment – 17

Testing Single Page Application using React (Registration Form).

Index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App/>)
```

App.js:

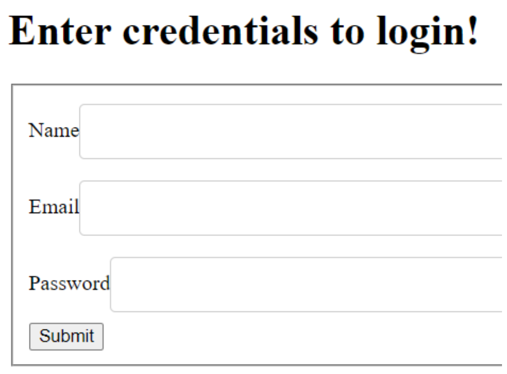
```
import { useState } from 'react';
import './App.css';
export default function App()
{

  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [submitted, setSubmitted] = useState(false);

  const handleName = (e) => {setName(e.target.value);};
  const handleEmail = (e) => {setEmail(e.target.value);};
  const handlePassword = (e) => {setPassword(e.target.value);};

  const handleSubmit = (e) => {
      e.preventDefault();
      if (name === '' || email === '' || password === '') {
      alert("Please enter all the fields");
      } else {
      setSubmitted(true);
      }
    };
  const successMessage = () => {
        if(submitted)
        return (
        <div className="success" >
            <h1>User {name} successfully registered!!</h1>
        </div>
        );
    };
  return (
    <div className="form">
      <div>
          <h1> Enter credentials to login! </h1>
      </div>
      <div className="messages">
          {successMessage()}
      </div>
      <form>
```

```
        <fieldset>
          <label className="label">Name</label>
          <input onChange={handleName} className="input"
value={name} type="text" /><br></br>
          <label className="label">Email</label>
          <input onChange={handleEmail} className="input"
value={email} type="email" /><br></br>
          <label className="label">Password</label>
          <input onChange={handlePassword} className="input"
value={password} type="password" /><br></br>
          <button onClick={handleSubmit} className="btn"
type="submit">Submit</button>
        </fieldset>
      </form>
    </div>
  );
}
```

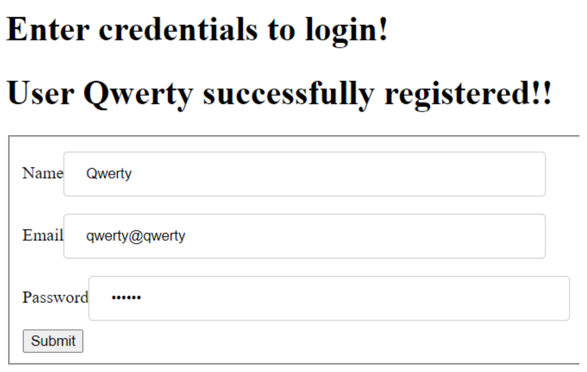
App.css

```
.input {
  width: 30%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}
```

Output:



Enter credentials to login!



Enter credentials to login!

User Qwerty successfully registered!!

Experiment – 18

Implement navigation using react router.

To add React Router in your application, run this in the terminal from the root directory of the application:

Npm i-D react-router-dom

Index.js

```
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Layout from "./pages/Layout";
import Home from "./pages/Home";
import Blogs from "./pages/Blogs";
import Contact from "./pages/Contact";
import NoPage from "./pages/NoPage";
import ccs from "./App.css";
     export default function App() {
           return (
         <BrowserRouter>
           <Routes>
             <Route path="/" element={<Layout />}>
               <Route index element={<Home />} />
               <Route path="blogs" element={<Blogs />} />
               <Route path="contact" element={<Contact />} />
               <Route path="*" element={<NoPage />} />
             </Route>
           </Routes>
         </BrowserRouter>
       );
     }
     const                          root                          =
     ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);
```

Create a folder name called pages. Within a pages create following files.

Blogs.js

```
const Blogs = () => {
      return <h1>Blog Articles</h1>;
};
export default Blogs;
```

Home.js

```
const Home = () => {
      return <h1>Home</h1>;
};
export default Home;
```

Contact.js

```
const Contact = () => {
      return <h1>Contact Me</h1>;
      };
export default Contact;
```

Layout.js

```
import { Outlet, Link } from "react-router-dom";
      const Layout = () => {
        return (
          <>
            <nav>
              <ul>
                <li>
                  <Link to="/">Home</Link>
                </li>
                <li>
                  <Link to="/blogs">Blogs</Link>
                </li>
                <li>
                  <Link to="/contact">Contact</Link>
                </li>
              </ul>
            </nav>
            <Outlet />
          </>
        )
};export default Layout;
```

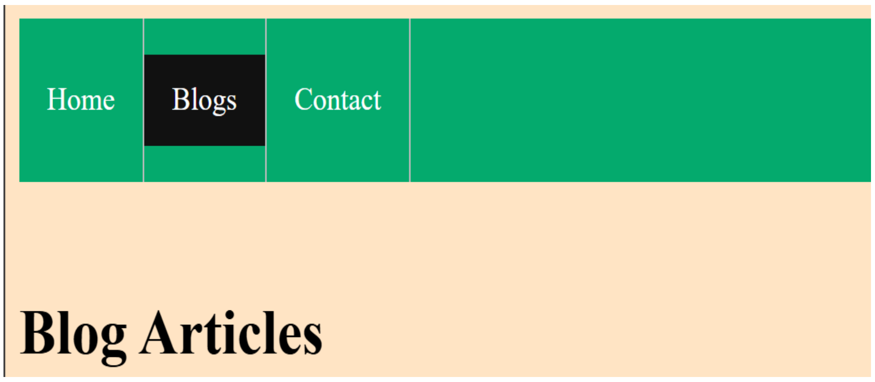
<u>NoPage.js</u>

```
const NoPage = () => {
      return <h1>404</h1>;
};
export default NoPage;
```

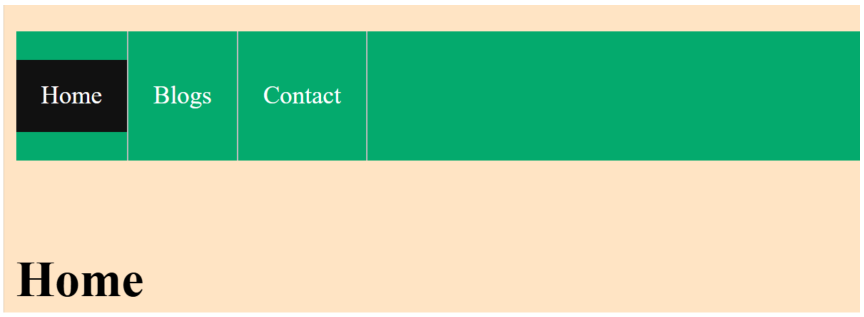
<u>App.css</u>

```
ul {
        list-style-type: none;
        margin: 0;
        padding: 0;
        overflow: hidden;
        background-color: #04AA6D;
}

li{
        float: left;
        border-right: 1px solid #bbb;
}li a {
        display: block;
        color: white;
        text-align: center;
        padding: 14px 16px;
        text-decoration: none;
}
li a:hover:not(.active) {
        background-color: #111;
}
body{
        background-color: bisque;
        align-content: center;
}
```

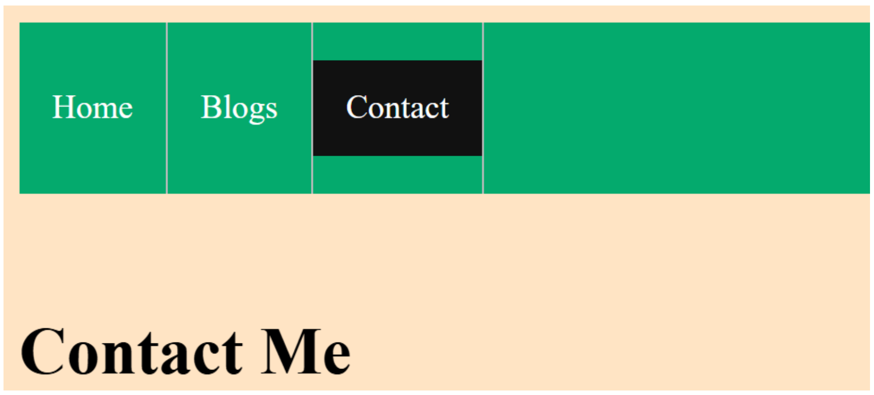
Output:



```
Home  Blogs  Contact

Blog Articles
```



```
Home  Blogs  Contact

Home
```



```
Home  Blogs  Contact

Contact Me
```

Experiment – 19

Build SPA (Add product to Product List).

App.js

```
import { useState } from "react";
import "./App.css";
function App() {
  const [list, setList] = useState([]);
  const [value, setValue] = useState("");
  const addToList = () => {
    let tempArr = list;
    tempArr.push(value);
    setList(tempArr);
    setValue("");
  };
 const deleteItem = (index) => {
 let temp = list.filter((item, i) => i !== index);
    setList(temp);
  };
  return (
    <div className="App">
        <fieldset>
        <h>Add Product to List</h><br></br>
         <input    type="text"    value={value}    onChange={(e)    =>
setValue(e.target.value)}/>
           <button     onClick={addToList}>     Click     to     Add
</button><br></br><br></br>
    <h>Product Catalog</h><br></br>
    <ol>
               {list.map((item,   i)   =>   <li   onClick={()   =>
deleteItem(i)}>{item} </li>)}
    </ol>
    <h>Click on Product to Delete</h><br></br>
    </fieldset></div>
  );}export default App;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode> <App/> </React.StrictMode>
);
```

To compile the program type **npm start** in vsc terminal

Experiment – 20

Setting up an environment for creating Java Applications. Install Java, and Java Editor (Eclipse or IntelliJ IDEA, etc). Install DBMS (MySQL/Postgre SQL).

Installing JDK Java:

Step 01: Go to the official website of Java and search for Java 17 or higher versions.
Step 02: Download the MSI installer.
Step 03: Open the downloads folder and execute the downloaded JDK file.
Step 04: Select the installation directory and click on 'Install'.

Download MySQL:

The simplest and recommended method is to download MySQL installer for your system from https://dev.mysql.com/downloads/installer. Execute the file. Select MySQL community installer MSI file.

Install MySQL:

After downloading, unzip it and double click on the MSI installer and follow the below steps:

Step 01: "Choose a Setup Type" screen: Choose the 'Full' setup type. This installs all MySQL products and features. Click on 'Next'.
Step 02: "Check requirements" screen: The installer checks if your PC has the needed requirements. If there are some failing requirements, click on each item to resolve it.
Step 03: "Installation" screen: See what products will be installed. Click 'Execute' to download and install the products. Click on 'Next' after finishing.
Step 04: "Product Configuration" screen: See what products will be configured. Click on 'MySQL Server' option to configure the MySQL server. Click on 'Next'.
Step 05: "Authentication Method" screen: Choose 'Use Story Password Encryption for Authentication'. Click on 'Next'.
Step 06: "Accounts and Roles" screen: Set a password for the root accounts. Click 'Next'.
Step 07: "Windows Service" screen: Configure the Windows services to start the server. Keep the default setup and then click on 'Next'.
Step 08: "Apply Configuration" screen: Click the 'Execute' button to apply the service configuration. Click on the 'Finish' button after completion.
Step 09: "Product Configuration" screen: See that the product configuration is completed. Keep the default settings and click on the 'Next' and 'Finish' buttons to finish the installation process.
Step 010: In the next screen, choose to configure the Router. Click on 'Next'.
Step 011: "Connect the Server" screen: Type in the root password. Click the 'Check' button to check if the connection is successful or not.

Step 012: “Apply Configuration” screen: Select the option and click the ‘Execute’ button. After finishing, clicl on ‘Finish’.

Step 013: “Installation Complete” screen: The installation is completed. Click on the ‘Finish’ button.

Install Eclipse:

Step 01: Go to eclipse.org and download the .exe file for your system.

Step 02: Open the Downloads folder and execute the downloaded eclipse executable file.

Step 03: Select the installation directory and click on ‘Install’.

Step 04: Once Eclipse is installed, click on ‘Launch’.

<u>Experiment – 21</u>

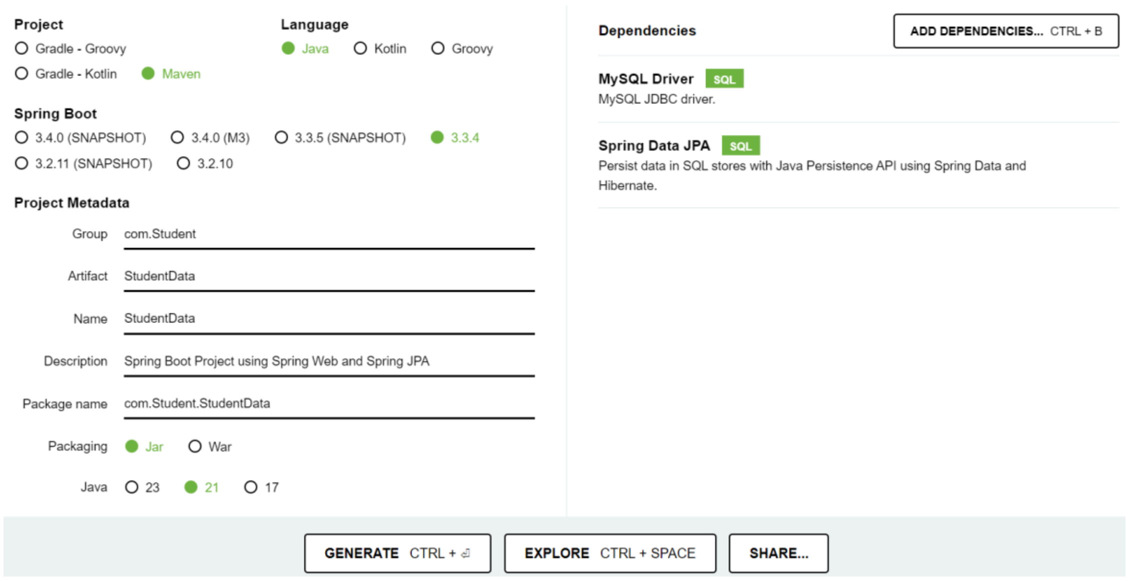
<u>Create a Spring Application using Spring Web and Spring JPA.</u>

Step 01: Open the web browser and go to start.spring.io.

Step 02: Create a project with the following information:

a. Project – Maven
b. Language – Java
c. Spring Boot Version – 3.3.4
d. Project Metadata:
    i. Group – com.Student
    ii. Artifact – StudentData
    iii. Name – StudentData
    iv. Package Name – com.Student.StudentData
    v. Packaging – Jar
    vi. Java Version – 21
e. Dependencis –
    i. MySQL Driver
    ii. Spring JPA

Step 03: Click on 'Generate' and extract the file downloaded.



Project
Gradle - Groovy
Gradle - Kotlin [✓ Maven]

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.4.0 (SNAPSHOT) 3.4.0 (M3) 3.3.5 (SNAPSHOT) [✓ 3.3.4]
3.2.11 (SNAPSHOT) 3.2.10

Project Metadata

Group	com.Student
Artifact	StudentData
Name	StudentData
Description	Spring Boot Project using Spring Web and Spring JPA
Package name	com.Student.StudentData
Packaging	[✓ Jar] War
Java	23 [✓ 21] 17

Dependencies

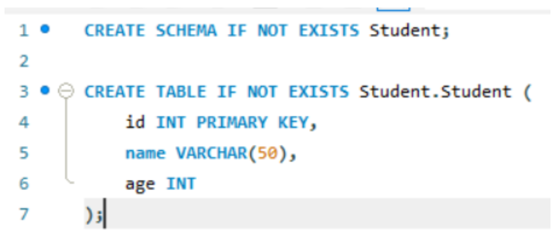
MySQL Driver SQL
MySQL JDBC driver.

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

Step 04: Launch VS Code and MySQL Workbench.

Step 05: In the Workbench, create a new Schema with the name 'Student'.

Step 06: In the 'Student' schema, create a table by the name 'Student' with id, name, and age as the columns of the table.



```
CREATE SCHEMA IF NOT EXISTS Student;

CREATE TABLE IF NOT EXISTS Student.Student (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT
);
```

Step 07: In VS Code, go to File → Open Folder → Open the folder generated and extracted earlier.

Step 08: In the project navigate to src/main/java directory a default package (com/Student) is already created with a main java Application file (StudentdataDataApplication.java).

Step 09: Create one class file named 'Student.java' (must be same as Table name) and an interface named 'StudentRepo.java'.

- In the Student.java file, write the following code:

```
package com.Student.StudentData;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Student {

    public Student(int id, String name, String age) {
        super();
        this.id = id;
        this.name = name;
        this.age = age;
    }

    public Student() {
        super();
    }

    @Id
    private int id;
    private String name;
    private String age;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getAge() {
        return age;
    }
    public void setAge(String age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + ", age=" +
age + "]";
    }
}
```

- In the StudentRepo.java file, write the following code:

```
package com.Student.StudentData;
import org.springframework.data.repository.CrudRepository;

public interface StudentRepo extends CrudRepository<Student,
Integer>{
}
```

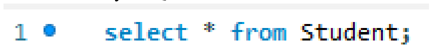
- In the main StudentDataApplication.java file, write the following code:

```
package com.Spring.StudentData;

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

@SpringBootApplication
public class StudentDataApplication {
    public static void main(String[] args) {
        ApplicationContext context =
        SpringApplication.run(StudentDataApplication.class, args);
        StudentRepo studentRepo =
context.getBean(StudentRepo.class);
        Student st = new Student();
        st.setId(1);
        st.setName("Abhigna");
        st.setAge("18");
        Student stDisplay = studentRepo.save(st);
        System.out.println(stDisplay);
    }
}
```

- Navigate to the 'resources' folder and 'application properties' file, add the following code, and modify it according to the credentials of your system.

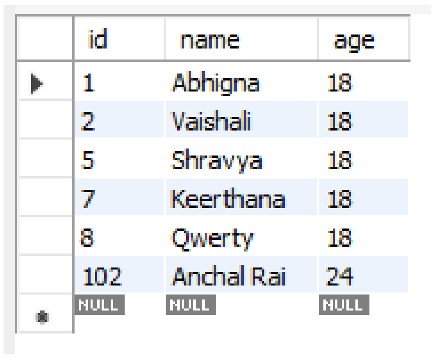
```
spring.application.name=StudentData
spring.datasource.name=Student
spring.datasource.url=jdbc:mysql://localhost/Student
spring.datasource.username=root
spring.datasource.password=Abhigna@01
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLD
ialect
```

- Install the extensions of 'Spring Initializer', 'Spring Boot Dashboard', 'Spring Boot Extension pack', and 'Spring Boot Tools'.
- To run the Application. Go to Spring Dashboard(present at left side of VS Code) in Apps > Run
- Go to MySQL workbench and execute the following query.



```
select * from Student;
```

Output:



id	name	age
1	Abhigna	18
2	Vaishali	18
5	Shravya	18
7	Keerthana	18
8	Qwerty	18
102	Anchal Rai	24
NULL	NULL	NULL

All the data added in the StudentDataApplication.java will be added to the table in the MySQL database and can be viewed there.

Experiment – 22

Install MongoDB and perform CRUD operations.

Step 1: Go to MongoDB official website

Step 2: Click on Products > MongoDB Community Edition

Step 3: Download the MongoDB Community Server for your OS

Step 4: To download the MongoDB shell click on Products > Tools > shell

Step 5: Download the zip file and extract it

Step 6: Now execute the executable file of MongoDB server after the download is finished

Step 7: Accept the license terms and agreement and click on next.

Step 8: Select complete and click Next every time keeping everything as it is. Click on Install.

Step 9: Once the installation is complete click on finish.

Step 10: Now Copy the extracted file to *C:/* drive and create subfolder with name *data/db* in the same drive.

Step 11: Now copy the bin folder path of both shell and server and add those paths to environment variables.

Step 12: Launch the command prompt and start the server by executing the command *'mongod'*

Step 13: Open another command prompt and launch the MongoDB shell using *mongosh* command

Step 14: CRUD Operation in MongoDB (Let's take an example of Student)

Step 15: Create Database use Student

Step 16: Create Collection

```
db.createCollection('studentInfo')
```

Step 17: Show Collection

```
show collections
```

Step 18: Show Database

```
show dbs
```

Step 19: Insert a single Documents

```
db.studentInfo.insertOne({id: 101, name: "Vikas Singh", age: 24, address: "Varanasi"})
```

Step 20: Insert multiple documents

```
db.studentInfo.insertOne([{id: 101, name: "Vikas Singh", age: 24, address: "Varanasi"}, {id: 102,
name: "Anchal Rai", age: 22, address: "Varanasi"}])
```

Step 21: Show documents in a collection

```
db.studentInfo.find()
```

Step 22: Show documents in a collection using filter

```
db.studentInfo.find({id: 102})
```

Step 23: Update one document

```
db.studentInfo.updateOne({}, {address: "Bangalore"})
```

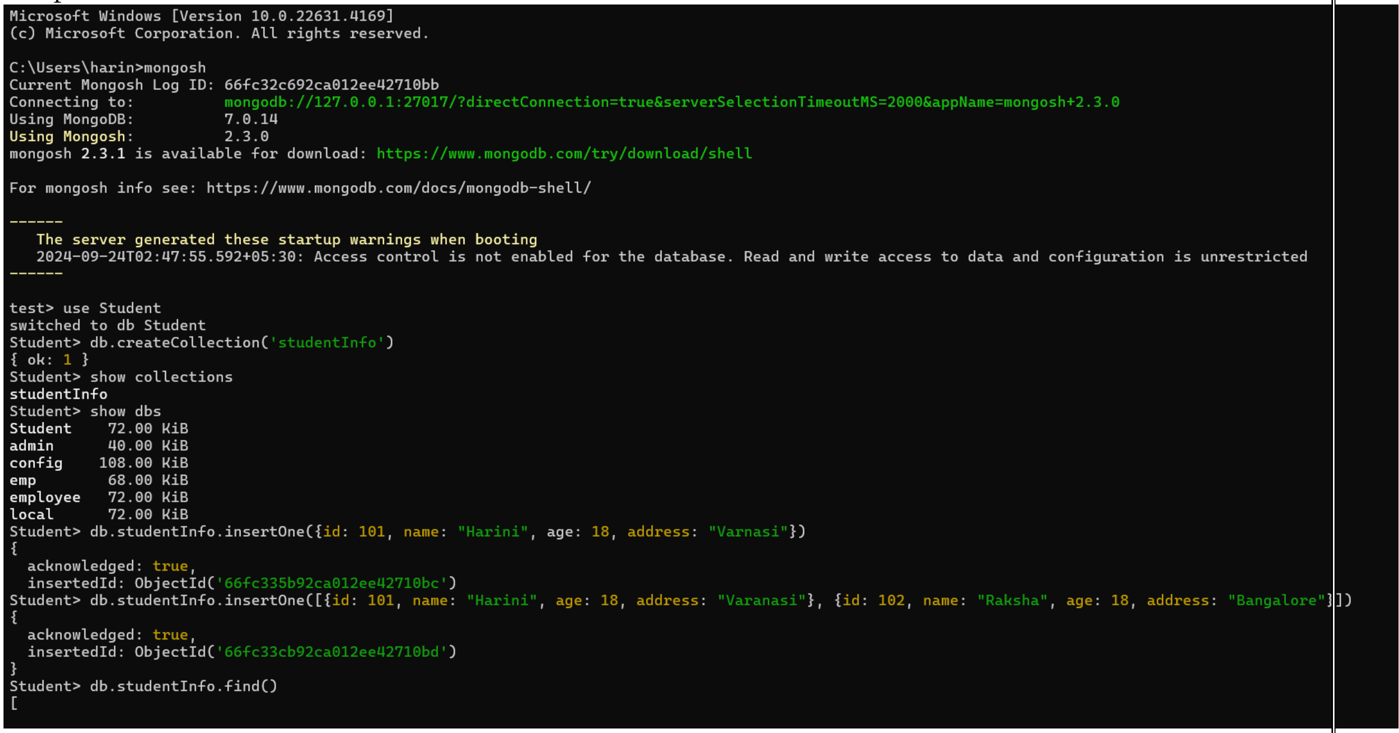
Step 25: Delete single document

```
db.studentInfo.deleteOne({address: "Bangalore"})
```

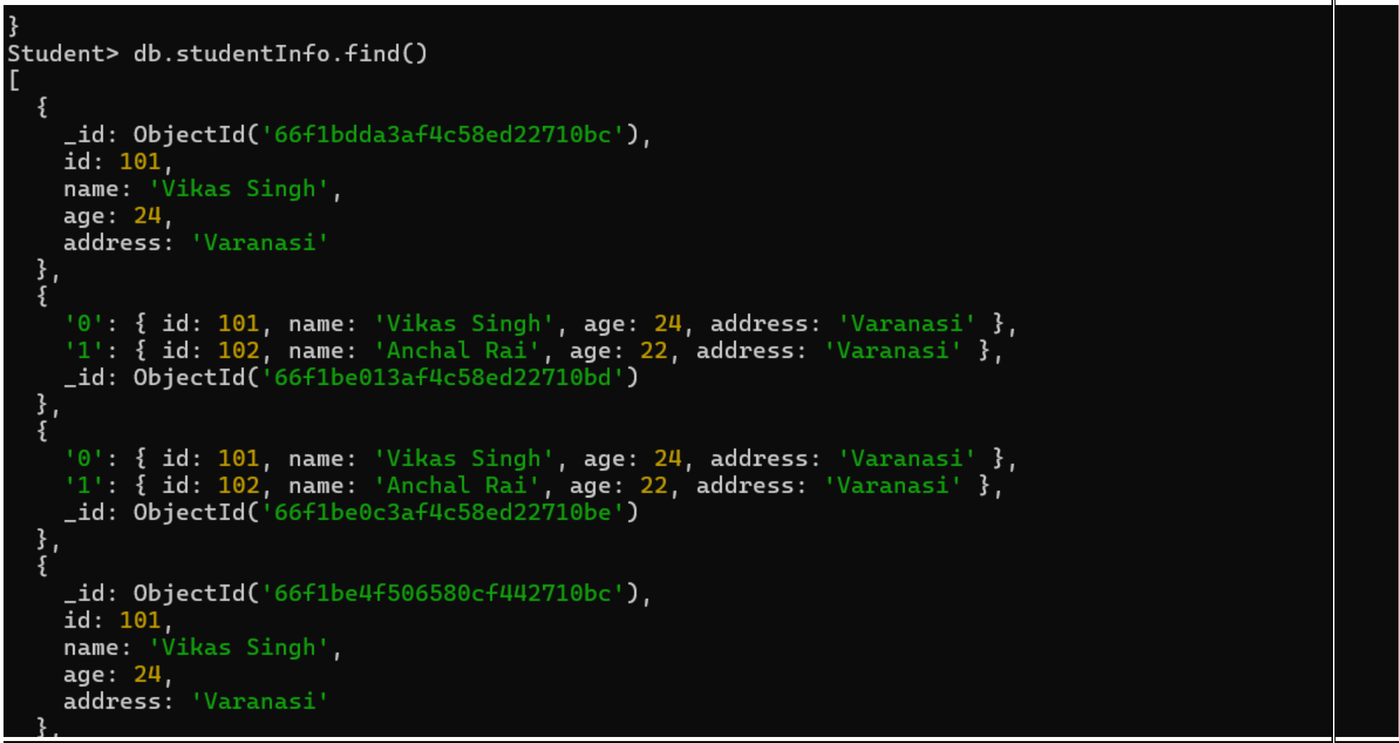
Step 26: Delete multiple documents

```
db.studentInfo.deleteMany({})
```

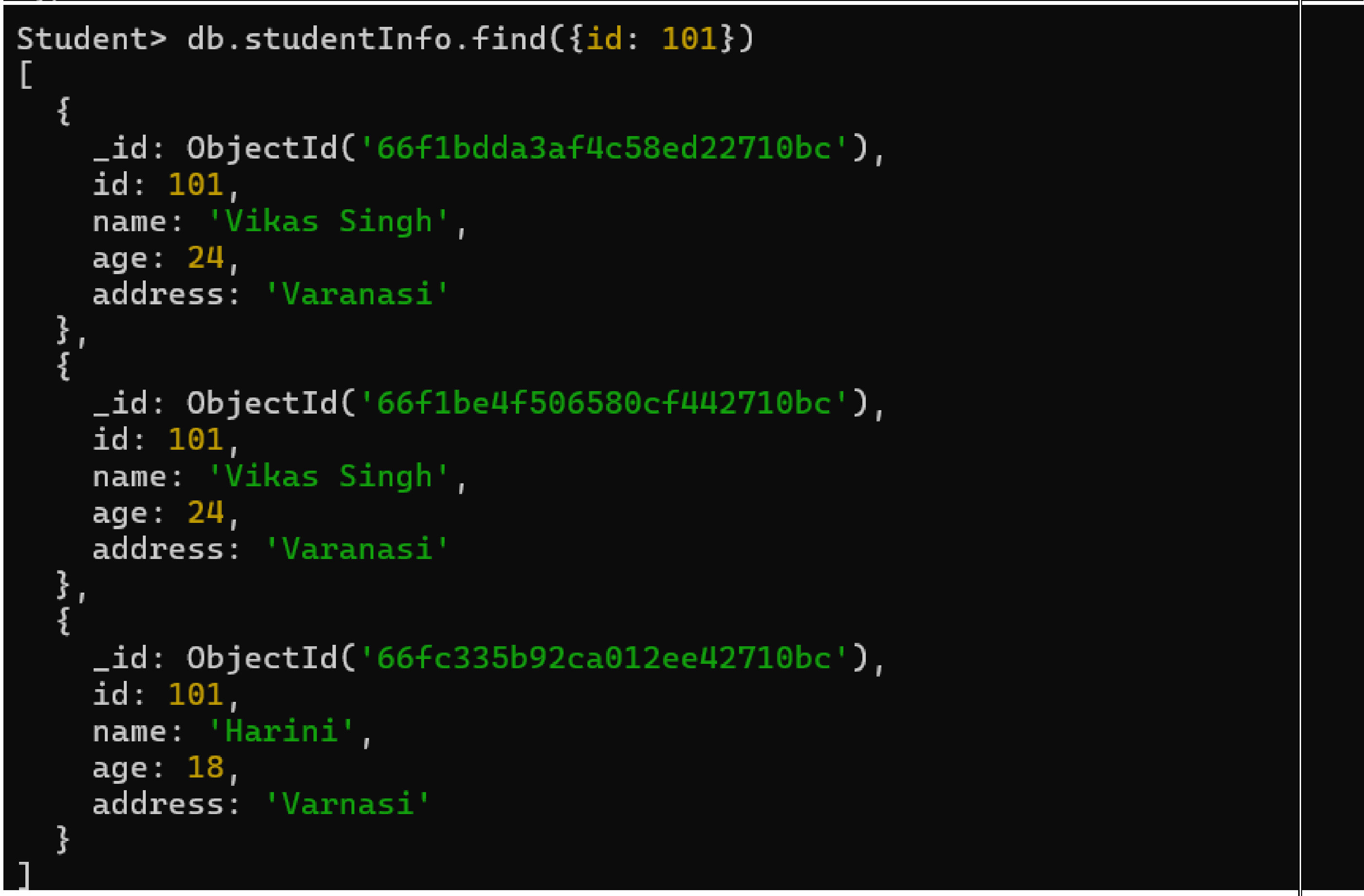
Output:



```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\harin>mongosh
Current Mongosh Log ID: 66fc32c692ca012ee42710bb
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.0
Using MongoDB:          7.0.14
Using Mongosh:          2.3.0
mongosh 2.3.1 is available for download: https://www.mongodb.com/try/download/shell

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

------
   The server generated these startup warnings when booting
   2024-09-24T02:47:55.592+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use Student
switched to db Student
Student> db.createCollection('studentInfo')
{ ok: 1 }
Student> show collections
studentInfo
Student> show dbs
Student    72.00 KiB
admin      40.00 KiB
config    108.00 KiB
emp        68.00 KiB
employee   72.00 KiB
local      72.00 KiB
Student> db.studentInfo.insertOne({id: 101, name: "Harini", age: 18, address: "Varnasi"})
{
  acknowledged: true,
  insertedId: ObjectId('66fc335b92ca012ee42710bc')
Student> db.studentInfo.insertOne([{id: 101, name: "Harini", age: 18, address: "Varanasi"}, {id: 102, name: "Raksha", age: 18, address: "Bangalore"}])
{
  acknowledged: true,
  insertedId: ObjectId('66fc33cb92ca012ee42710bd')
}
Student> db.studentInfo.find()
[
```



```
}
Student> db.studentInfo.find()
[
  {
    _id: ObjectId('66f1bdda3af4c58ed22710bc'),
    id: 101,
    name: 'Vikas Singh',
    age: 24,
    address: 'Varanasi'
  },
  {
    '0': { id: 101, name: 'Vikas Singh', age: 24, address: 'Varanasi' },
    '1': { id: 102, name: 'Anchal Rai', age: 22, address: 'Varanasi' },
    _id: ObjectId('66f1be013af4c58ed22710bd')
  },
  {
    '0': { id: 101, name: 'Vikas Singh', age: 24, address: 'Varanasi' },
    '1': { id: 102, name: 'Anchal Rai', age: 22, address: 'Varanasi' },
    _id: ObjectId('66f1be0c3af4c58ed22710be')
  },
  {
    _id: ObjectId('66f1be4f506580cf442710bc'),
    id: 101,
    name: 'Vikas Singh',
    age: 24,
    address: 'Varanasi'
  },
```



```
Student> db.studentInfo.find({id: 101})
[
  {
    _id: ObjectId('66f1bdda3af4c58ed22710bc'),
    id: 101,
    name: 'Vikas Singh',
    age: 24,
    address: 'Varanasi'
  },
  {
    _id: ObjectId('66f1be4f506580cf442710bc'),
    id: 101,
    name: 'Vikas Singh',
    age: 24,
    address: 'Varanasi'
  },
  {
    _id: ObjectId('66fc335b92ca012ee42710bc'),
    id: 101,
    name: 'Harini',
    age: 18,
    address: 'Varnasi'
  }
]
```

Experiment – 23

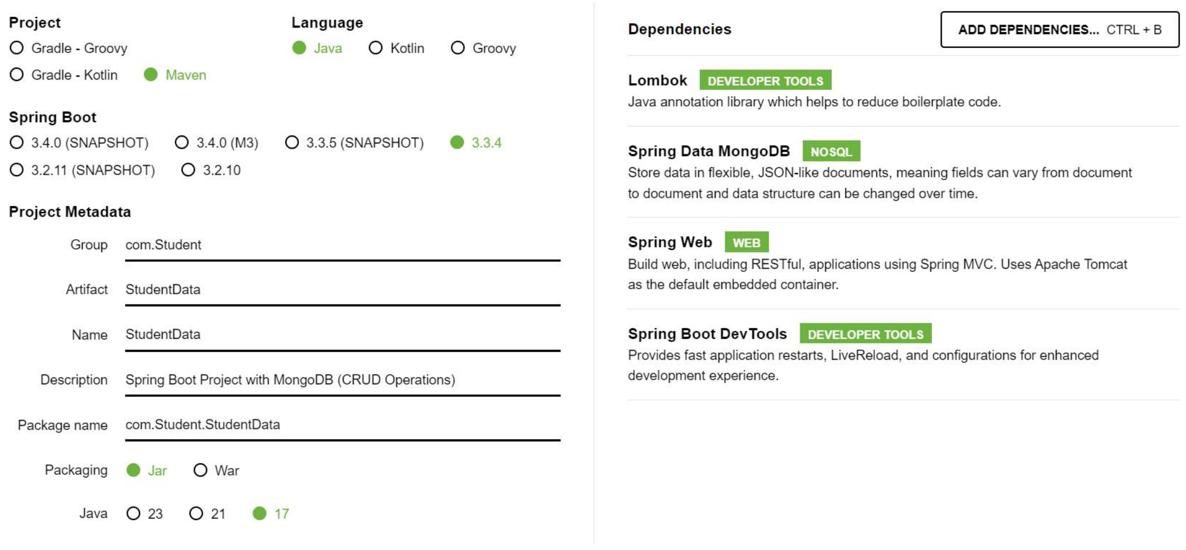
Perform CRUD operations on MongoDB through REST API using Spring Data MongoDB.

Step 01: Open the web browser and go to start.spring.io.

Step 02: Create a project with the following information:

a. Project – Maven
b. Language – Java
c. Spring Boot Version – 3.3.4
d. Project Metadata:
    i. Group – com.Student
    ii. Artifact – StudentData
    iii. Name – StudentData
    iv. Package Name – com.Student.StudentData
    v. Packaging – Jar
    vi. Java Version – 17
e. Dependencis –
    i. Spring Web
    ii. Spring Data MongoDB
    iii. Lombok
    iv. Sping Boot Dev Tools

Step 03: Click on Generate and extract the downloaded file.



Project

Gradle - Groovy Gradle - Kotlin [✓ Maven]

Language

[✓ Java] Kotlin Groovy

Spring Boot

3.4.0 (SNAPSHOT) 3.4.0 (M3) 3.3.5 (SNAPSHOT) [✓ 3.3.4] 3.2.11 (SNAPSHOT) 3.2.10

Project Metadata

Group	com.Student
Artifact	StudentData
Name	StudentData
Description	Spring Boot Project with MongoDB (CRUD Operations)
Package name	com.Student.StudentData
Packaging	[✓ Jar] War
Java	23 21 [✓ 17]

Dependencies ADD DEPENDENCIES... CTRL + B

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

Spring Data MongoDB NOSQL
Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

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

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Step 04: Open the extracted folder in VS Code

Step 05: Go to Explorer > src > main > package (com\Student\StudentData) inside that you will find a default Application file with main method called StudentDataSApplication.java.

Step 06: Create two class files named 'Student.java' and 'StudentController.java' and an interface named 'StudentRepo.java'.

➢ In the StudentDataApplication.java, type the following:

```
package com.Student.StudentData;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;

@SpringBootApplication
public class StudentDataApplication {
    public static void main(String[] args) {
        ApplicationContext context =
SpringApplication.run(StudentDataApplication.class, args);
        StudentRepo studentRepo =
context.getBean(StudentRepo.class);
        Student st = new Student();
        st.setId(1);
        st.setName("Abhigna");
        st.setAddress("Bangalore");
        Student stDisplay = studentRepo.save(st);
        System.out.println(stDisplay);
    }
}
```

➢ In the Student.java, type the following:

```
package com.Student.StudentData;

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

 import lombok.AllArgsConstructor;
 import lombok.Data;
 import lombok.NoArgsConstructor;

 @Data
 @NoArgsConstructor
 @AllArgsConstructor
 @Document(collection="studentData")
 public class Student {
 @Id
 private int id;
 private String name;
 private String address;
 }
```

➢ In the StudentController.java, type the following:

```
package com.Student.StudentData;

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.RequestBody;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class StudentController {
    @Autowired
    public StudentRepo repo;
    @PostMapping("/createData")
    public String createStudent(@RequestBody Student student){
        repo.save(student);
        return "Data added successfully.";
    }
    @GetMapping("/fetchData")
    public List<Student> fetch(){
        return repo.findAll();
    }
    @DeleteMapping("removeData/{id}")
    public String delete(@PathVariable int id) {
        repo.deleteById(id);
        return "Document deleted successfully";
    }
}
```

➢ In the StudentRepo.java, type the following:

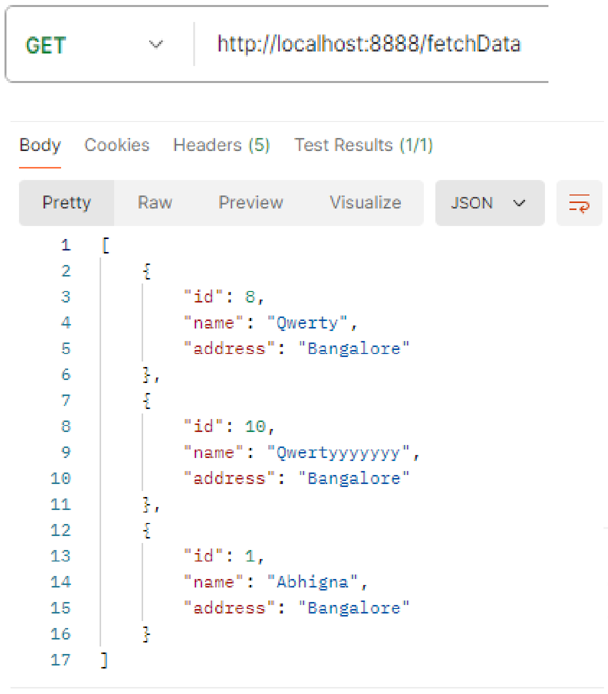
```
package com.Student.StudentData;
import org.springframework.data.mongodb.repository.MongoRepository;

public interface StudentRepo extends
MongoRepository<Student,Integer>{

}
```

➢ Navigate to the 'resources' folder and 'application properties' file, add the following code:

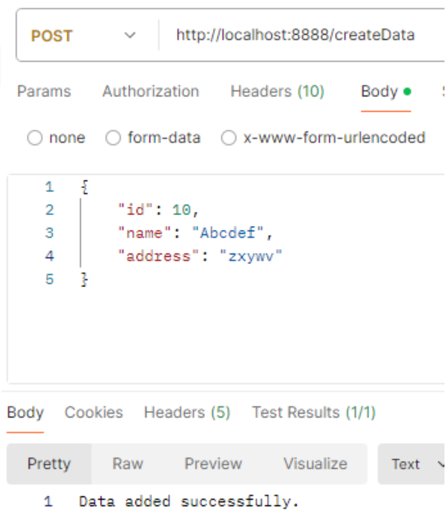
```
server.port=8888
spring.data.mongodb.host=localhost
spring.data.mongodb.database=student
```

- To run the Application. Go to Spring Boot Dashboard present at left side of the VS Code. In Apps you will find your project name → Click on Run. →The Spring Boot Application will run on the terminal opened.
- Open the MongoDB Shell and create the database ‘student’ and a collection ‘studentData’.

- Open the Postman application or install and use the Thunder Client Extension on VS Code and click on ‘New Request’.
- To test the method use the post after the endpoint URL i.e., http://localhost:PORT/label
- Click on send and cross check the document from MongoDB or on the browser by entering the URL mentioned.

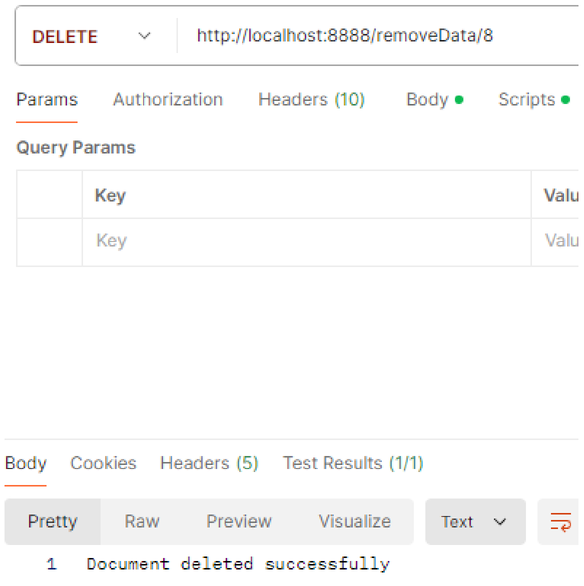


```
[
    {
        "id": 8,
        "name": "Qwerty",
        "address": "Bangalore"
    },
    {
        "id": 10,
        "name": "Qwertyyyyyyy",
        "address": "Bangalore"
    },
    {
        "id": 1,
        "name": "Abhigna",
        "address": "Bangalore"
    }
]
```

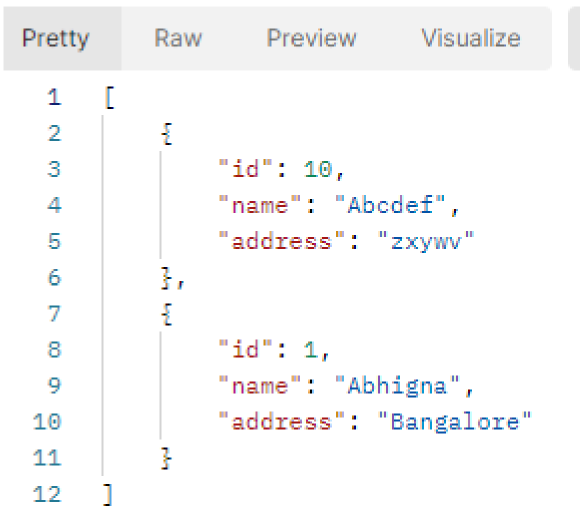


```
{
    "id": 10,
    "name": "Abcdef",
    "address": "zxywv"
}
```

```
Data added successfully.
```



```
Document deleted successfully
```



```
[
    {
        "id": 10,
        "name": "Abcdef",
        "address": "zxywv"
    },
    {
        "id": 1,
        "name": "Abhigna",
        "address": "Bangalore"
    }
]
```