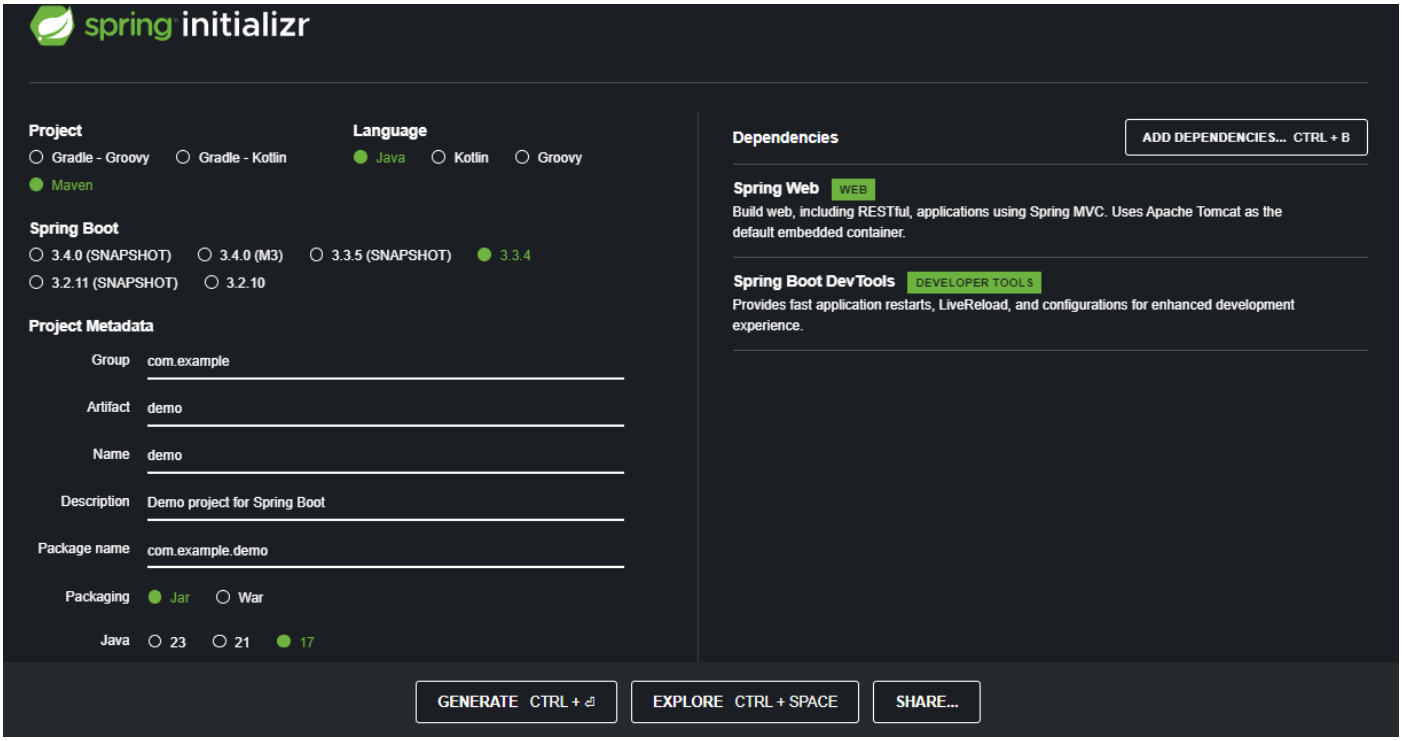
9.Spring Boot Application without security.

Step1: go to google and search for spring initialize. Visit https://start.spring.io/ website.

Step2: Choose project, language, spring Boot version. Add project metadata and dependencies as shown below.



Step3: click on generate → go to download and extract the zip file.

File – Open Projects from file system – directory - demo

Name: demo_demo

Package: com.example.demo

Name: Controller.java

package com.example.demo;

Controller.java

```
package com.example.demo;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class Controller {
@GetMapping("/spring")
public String Welcome() {
return ("<h1>Welcome to SpringBoot</h1>");
}
}
```

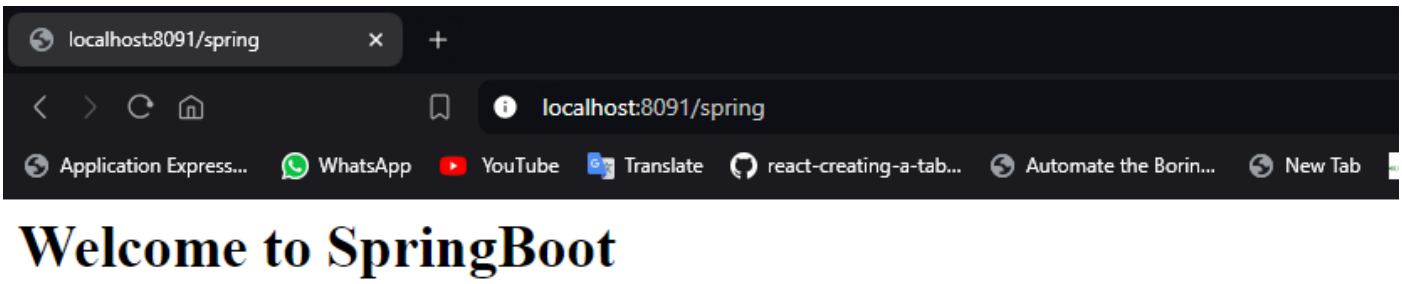
application.properties File

```
spring.application.name=demo
server.port=8091
```

Go to demo_demo → src/main/java → com.example.demo → DemoApplication.java → Run the file.

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.4)

2024-09-29T15:37:09.224+05:30  INFO 13732 --- [demo] [  restartedMain] com.example.demo.DemoApplication         : Starting DemoApplication usin
2024-09-29T15:37:09.251+05:30  INFO 13732 --- [demo] [  restartedMain] com.example.demo.DemoApplication         : No active profile set, fallin
2024-09-29T15:37:09.562+05:30  INFO 13732 --- [demo] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults ac
2024-09-29T15:37:09.567+05:30  INFO 13732 --- [demo] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related lo
2024-09-29T15:37:17.429+05:30  INFO 13732 --- [demo] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port
2024-09-29T15:37:17.511+05:30  INFO 13732 --- [demo] [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-09-29T15:37:17.512+05:30  INFO 13732 --- [demo] [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apa
2024-09-29T15:37:17.871+05:30  INFO 13732 --- [demo] [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
2024-09-29T15:37:17.882+05:30  INFO 13732 --- [demo] [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: i
2024-09-29T15:37:20.243+05:30  INFO 13732 --- [demo] [  restartedMain] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager
2024-09-29T15:37:21.118+05:30  INFO 13732 --- [demo] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running
2024-09-29T15:37:21.431+05:30  INFO 13732 --- [demo] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8090 (
2024-09-29T15:37:21.510+05:30  INFO 13732 --- [demo] [  restartedMain] com.example.demo.DemoApplication         : Started DemoApplication in 16
```

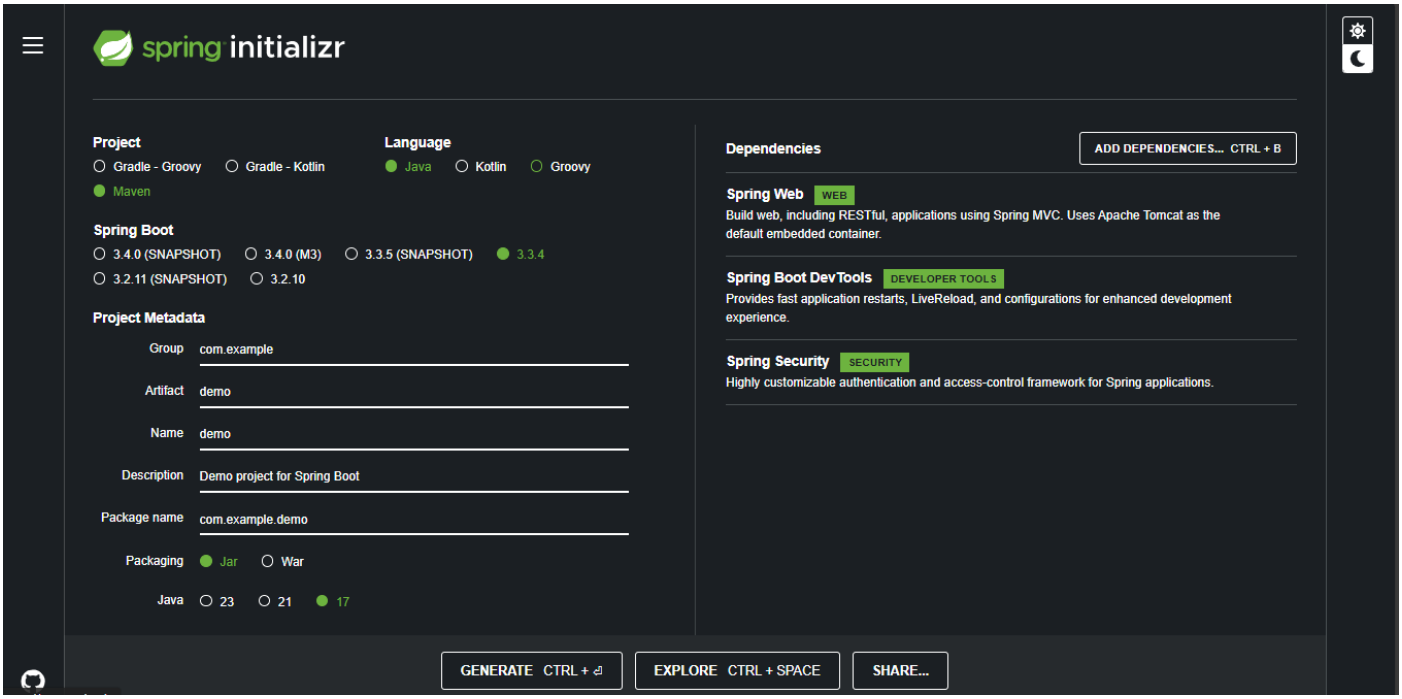
Go to any web browser →



10. Securing REST APIs with Spring Security.

Step1: go to google and search for spring initialize. Visit https://start.spring.io/ website.

Step2: Choose project, language, spring Boot version. Add project metadata and dependencies as shown below.



Step3: click on generate → go to download and extract the zip file.

File – Open Projects from file system – directory - demo

Name: demo_demo

Package: com.example.demo

Name: SecurityController.java

package com.example.demo;

SecurityController.java

```
package com.example.demo;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class SecurityController {
@GetMapping("/page")
public String Welcome() {
return ("<h1>Welcome to SpringBoot Security</h1>");
}
}
```

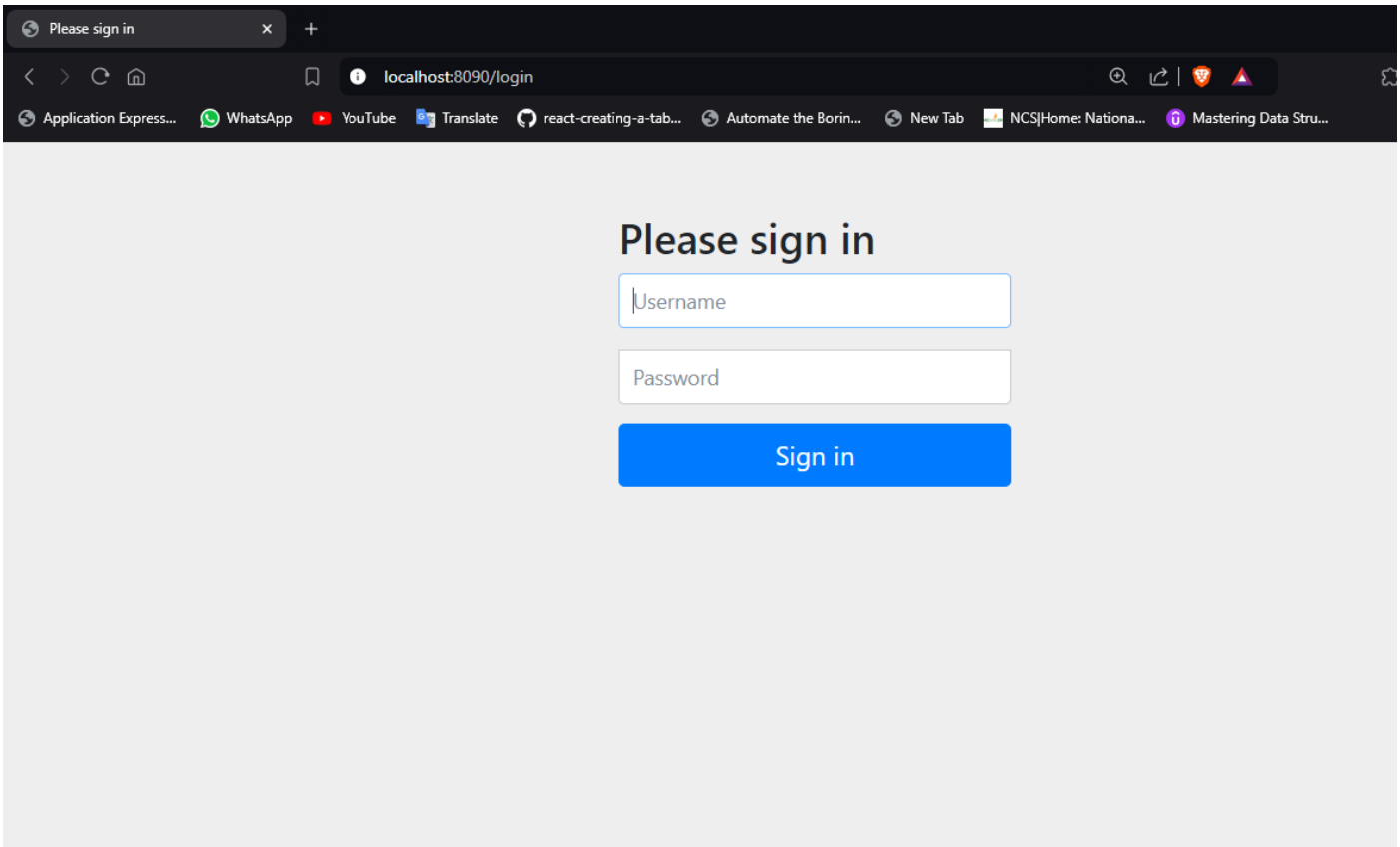
application.properties File

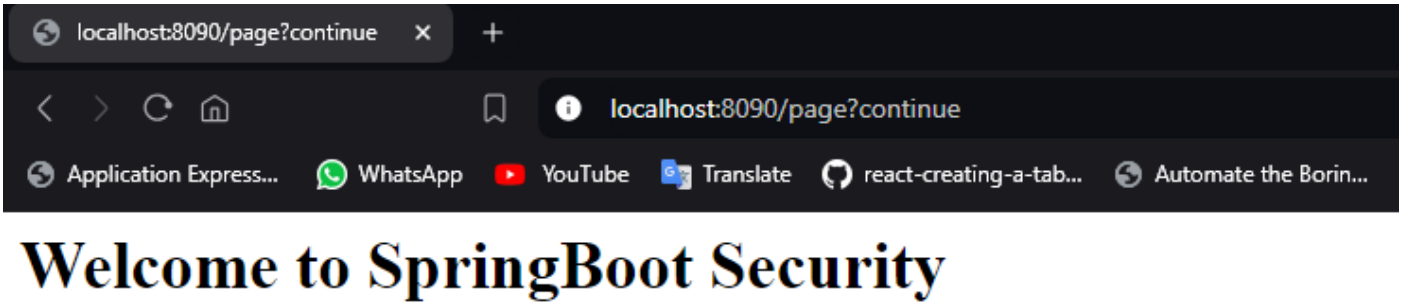
```
spring.application.name=demo
spring.security.user.name=niranjan
spring.security.user.password=murthy
server.port=8090
```

Go to demo_demo → src/main/java → com.example.demo → DemoApplication.java → Run the file.

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.4)

2024-09-29T15:37:09.224+05:30  INFO 13732 --- [demo] [  restartedMain] com.example.demo.DemoApplication         : Starting DemoApplication usin
2024-09-29T15:37:09.251+05:30  INFO 13732 --- [demo] [  restartedMain] com.example.demo.DemoApplication         : No active profile set, fallin
2024-09-29T15:37:09.562+05:30  INFO 13732 --- [demo] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults ac
2024-09-29T15:37:09.567+05:30  INFO 13732 --- [demo] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related lo
2024-09-29T15:37:17.429+05:30  INFO 13732 --- [demo] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port
2024-09-29T15:37:17.511+05:30  INFO 13732 --- [demo] [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-09-29T15:37:17.512+05:30  INFO 13732 --- [demo] [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apa
2024-09-29T15:37:17.871+05:30  INFO 13732 --- [demo] [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
2024-09-29T15:37:17.882+05:30  INFO 13732 --- [demo] [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: i
2024-09-29T15:37:20.243+05:30  INFO 13732 --- [demo] [  restartedMain] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager
2024-09-29T15:37:21.118+05:30  INFO 13732 --- [demo] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running
2024-09-29T15:37:21.431+05:30  INFO 13732 --- [demo] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8090 (
2024-09-29T15:37:21.510+05:30  INFO 13732 --- [demo] [  restartedMain] com.example.demo.DemoApplication         : Started DemoApplication in 16
```

Go to any web browser →





11. CRUD Operations on document using Mongo DB.

Creating a Table.

>db.createCollection("student")

>show tables

```
mongosh> use college
switched to db college
college> db.createCollection("student")
{ ok: 1 }
college> show tables
student
college>
```

insert() Method.

To insert data into MongoDB collection, you need to use MongoDB's insert() or save() method.

Syntax: db.COLLECTION_NAME.insert(document)

> db.student.insertOne({id:1,name:"chandru",mark:300})

```
college> db.student.insertOne({id:1,name:"chandru",mark:300})
{
  acknowledged: true,
  insertedId: ObjectId('66f930a4b115677aa62710bc')
}
college>
```

> db.student.insertMany([{id:2,name:"suman",mark:290},{id:3,name:"tejas",mark:280}])

```
}
college> db.student.insertMany([{id:2,name:"suman",mark:290},{id:3,name:"tejas",mark:280}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66f93160b115677aa62710bd'),
    '1': ObjectId('66f93160b115677aa62710be')
  }
}
college>
```

View data from Table.

db.student.find()

```
}
college> db.student.find()
[
  {
    _id: ObjectId('66f930a4b115677aa62710bc'),
    id: 1,
    name: 'chandru',
    mark: 300
  },
  {
    _id: ObjectId('66f93160b115677aa62710bd'),
    id: 2,
    name: 'suman',
    mark: 290
  },
  {
    _id: ObjectId('66f93160b115677aa62710be'),
    id: 3,
    name: 'tejas',
    mark: 280
  }
]
college>
```

Update.

db.student.updateOne({name:"chandru"},{$set:{name:"shekar",id:5}})

```
]
college> db.student.updateOne({name:"chandru"},{$set: {name:"shekar",id:5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
college>
```

Delete only one data.

db.student.deleteOne({name:"shekar"})

```
}
college> db.student.deleteOne({name:"shekar"})
{ acknowledged: true, deletedCount: 1 }
college>
```

12. Perform CRUD Operations on MongoDB through REST API using Spring Boot StarterData MongoDB.

Step 1: Create a Spring Boot project.

Step 2: Add the following dependency from spring initializer.

- Spring Web
- MongoDB
- Lombok
- DevTools

Step 3: Create 3 packages and create some classes and interfaces inside these packages

- entity
- repository

controller

Step 4: Inside the entity package create a Book.java file.

```
package com.example.MongoDB.Entity;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;
@Data
@NoArgsConstructor
@AllArgsConstructor
@Document(collection = "Book")
public class Book  {
	@Id
	private int id;
	private String bookName;
	private String authorName;
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getBookName() {
		return bookName;
	}
	public void setBookName(String bookName) {
		this.bookName = bookName;
	}
	public String getAuthorName() {
		return authorName;
	}
	public void setAuthorName(String authorName) {
		this.authorName = authorName;
	}

}
```

Step 5: Inside the repository package

```
package com.example.MongoDB.Repository;
```

```
import org.springframework.data.mongodb.repository.MongoRepository;
import com.example.MongoDB.Entity.Book;
public interface BookRepo
	extends MongoRepository<Book, Integer> {
}
```

Step 6: Inside the controller package. Inside the package create one class named as **BookController**

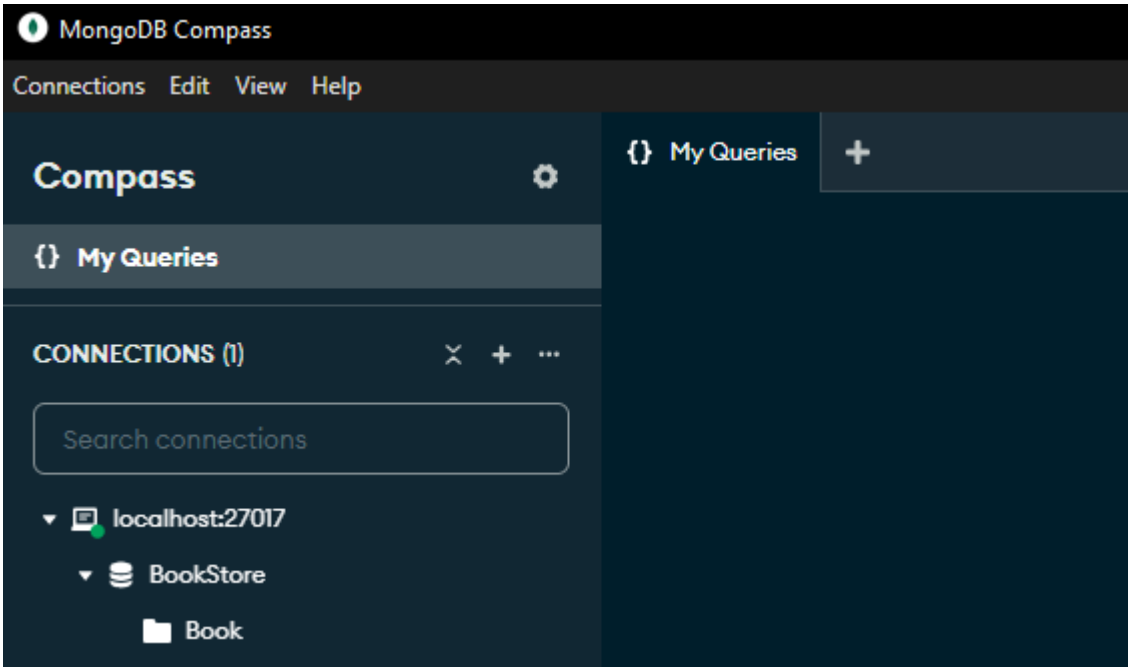
```
package com.example.MongoDB.Controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import com.example.MongoDB.Entity.Book;
import com.example.MongoDB.Repository.BookRepo;
import java.util.List;
@RestController
public class BookController {
	@Autowired
	private BookRepo repo;

@PostMapping("/addBook")
	public String saveBook(@RequestBody Book book){
		repo.save(book);
		return "Added Successfully";
	}
@GetMapping("/findAllBooks")
	public List<Book> getBooks() {
		return repo.findAll();
	}
@DeleteMapping("/delete/{id}")
	public String deleteBook(@PathVariable int id){
		repo.deleteById(id);
		return "Deleted Successfully";
	}
}
```

Step 7: Below is the code for the application.properties file

```
spring.application.name=MongoDB
server.port:8989
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=BookStore
```

Step 8: Inside the MongoDB Compass

Go to your MongoDB Compass and create a Database named **BookStore** and inside the database create a collection named **Book**



Testing the Endpoint in Postman

POST – http://localhost:8989/addBook

GET – http://localhost:8989/findAllBooks

DELETE – http://localhost:8989/delete/1

13. Test created APIs with the help of Postman.

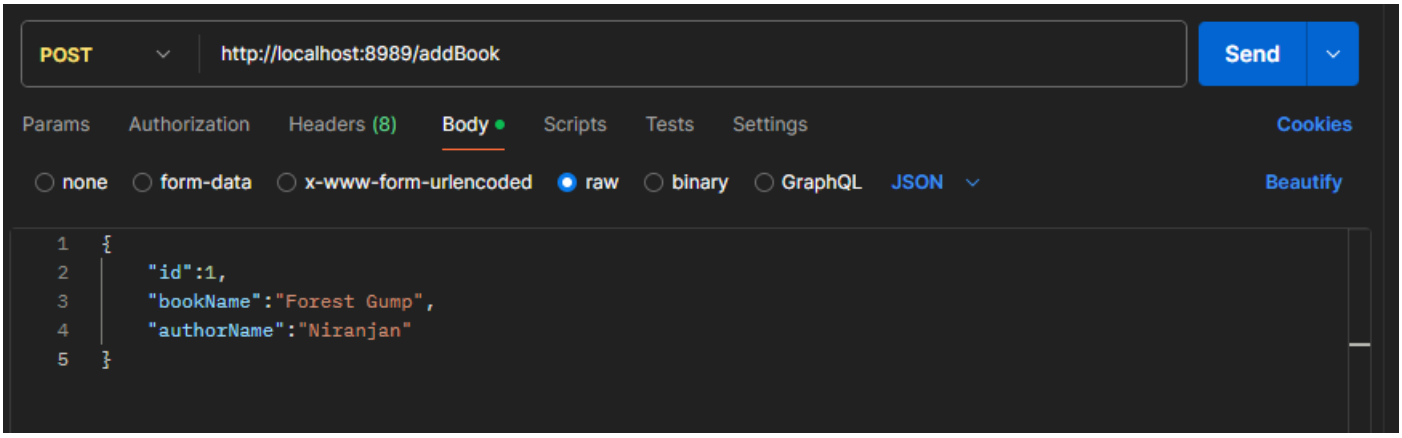
Step1: Download & Install postman from official website

https://www.postman.com/downloads/

Step2: Demonstrate Get, Post, Put, Delete methods

Get: Select Get method from dropdown list and enter the URL [http://localhost:8989/findAllBooks] → Send

Post: Select Post method from dropdown list → Click on Body, choose raw and select JSON from dropdown list and enter the URL [http://localhost:8989/addBook] → Give the input in the form of JSON and Click on Send



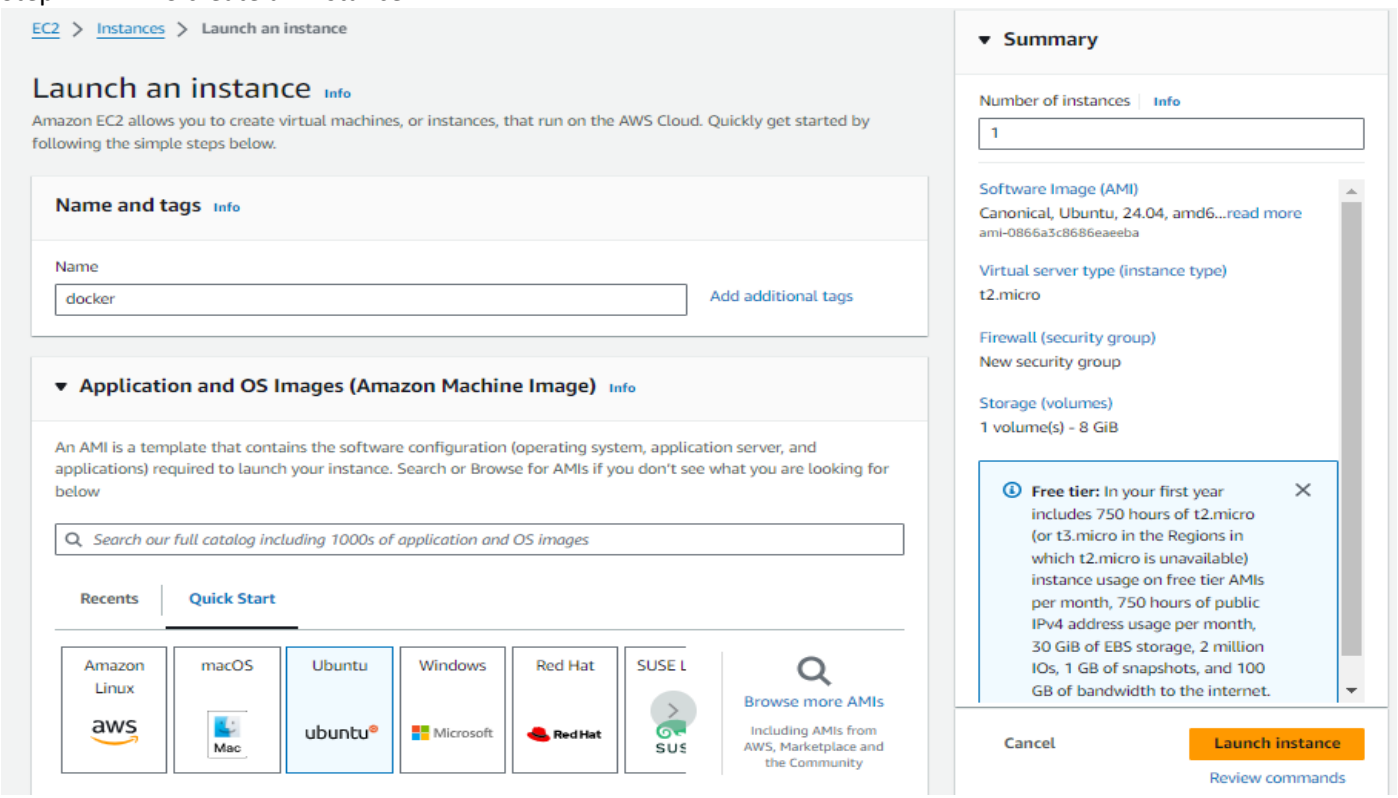
Put: Select Put method from dropdown list and enter the URL [http://localhost:8989/update/1]

Update the existing data and Click on Send

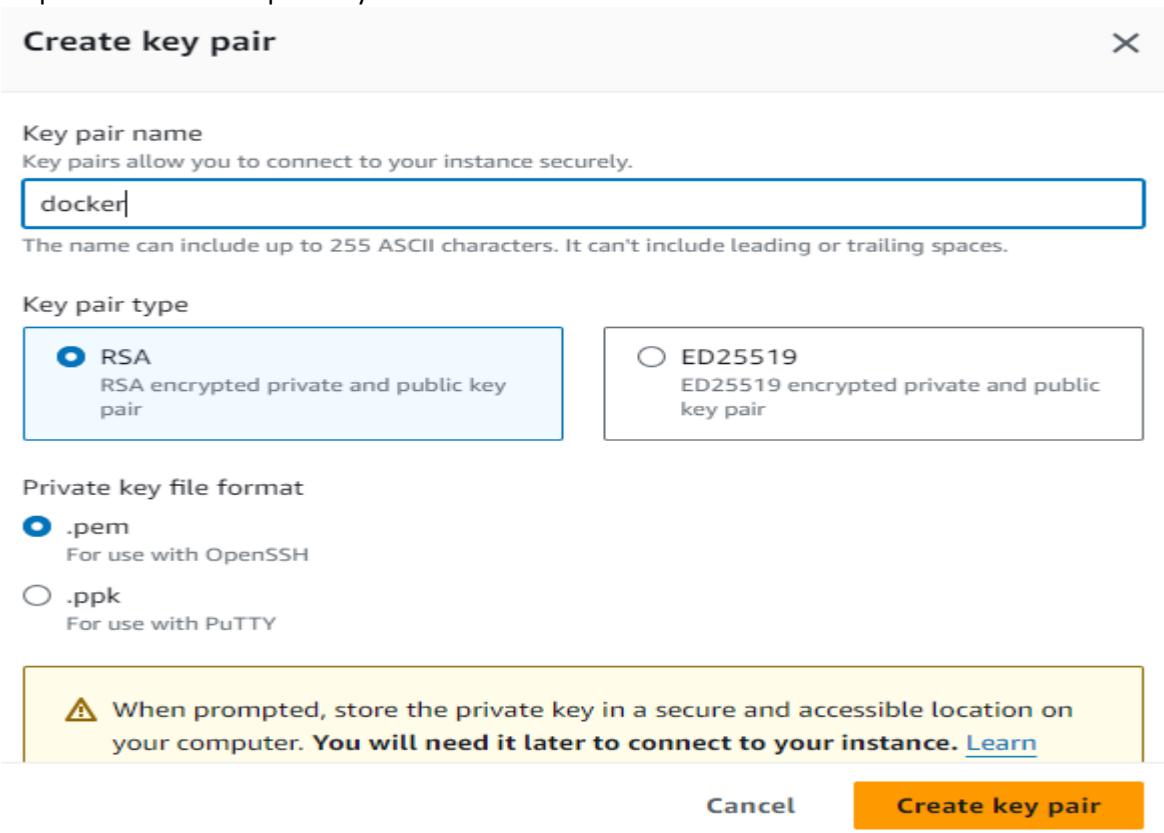
Delete: Select Delete method from dropdown list and enter the URL [http://localhost:8989/delete/1]

14.AWS

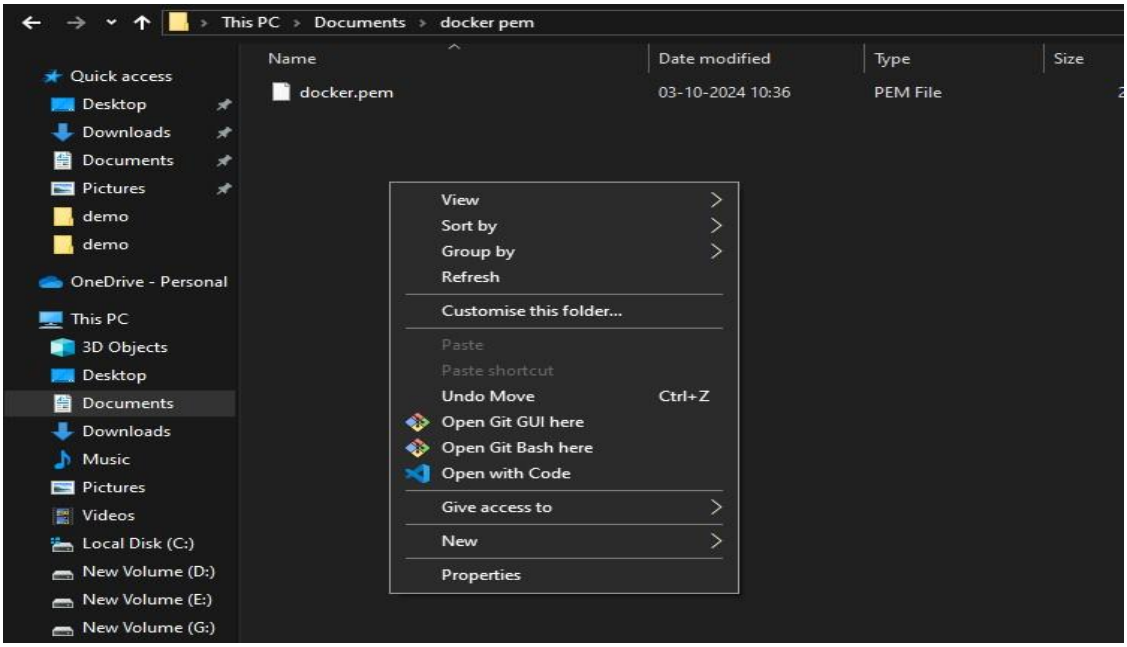
Step-1: In AWS create an instance.



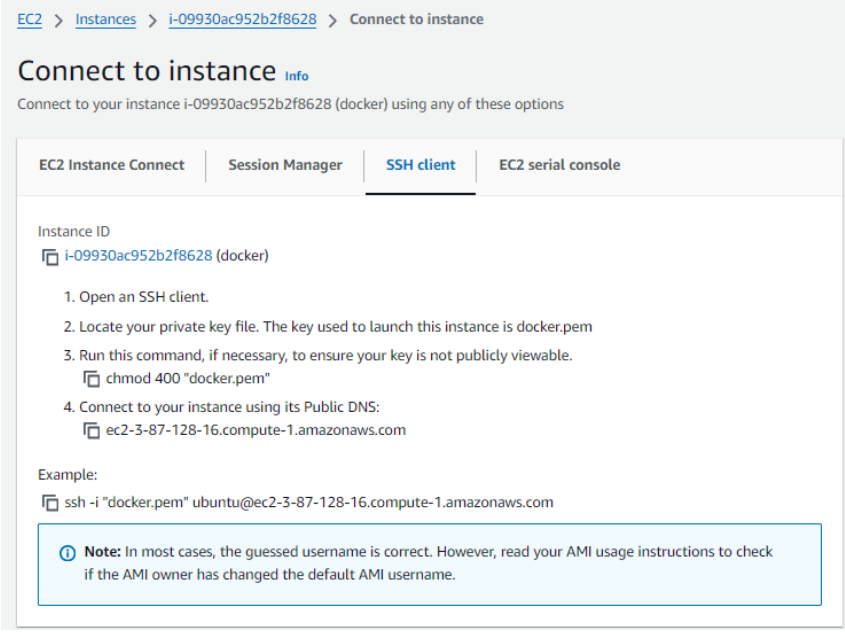
Step-2: Download the pem key.



Step-3: From there go to open bit bash here.



Step-4: First connect instance (using this command)




```
NagavallabhaShettySD@DESKTOP-A913TCK MINGW64 ~/Documents/docker pem
$ ssh -i "docker.pem" ubuntu@ec2-3-87-128-16.compute-1.amazonaws.com
The authenticity of host 'ec2-3-87-128-16.compute-1.amazonaws.com (3.87.128.16)' can't be established.
ED25519 key fingerprint is SHA256:foBoTh49bDeXf3GKn8hSACTg6H69W0fvOPK91m2wz1U.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-87-128-16.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Oct  3 05:19:57 UTC 2024

  System load:  0.0               Processes:             104
  Usage of /:   22.9% of 6.71GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for enX0: 172.31.32.84
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-32-84:~$ |
```

Step-5:

>Docker Install

sudo apt update

sudo apt install docker.io –y

>

sudo systemctl status docker

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-32-84:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-10-03 05:35:25 UTC; 13s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 1929 (dockerd)
      Tasks: 8
     Memory: 25.0M (peak: 25.6M)
        CPU: 220ms
     CGroup: /system.slice/docker.service
             └─1929 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

>

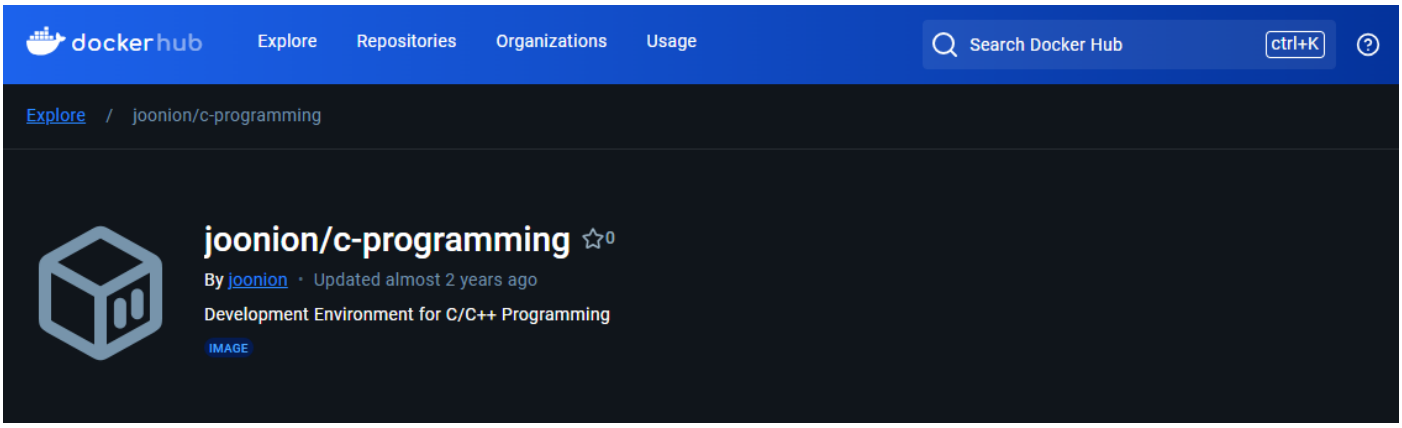
sudo systemctl start docker

>

sudo usermod -aG docker ubuntu

Till here copy the commands from [https://github.com/devopsmanu6 /Docker-Zero-to-Hero/blob/main/README.md](https://github.com/devopsmanu6/Docker-Zero-to-Hero/blob/main/README.md)

Then follow steps from https://hub.docker.com/r/joonion/c-programming (for this we need have an account in docker hub).



> sudo docker pull joonion/c-programming:latest

```
ubuntu@ip-172-31-32-84:~$ sudo docker pull joonion/c-programming:latest
latest: Pulling from joonion/c-programming
e96e057aae67: Pull complete
c98eb0cd7752: Pull complete
a226240d7de4: Pull complete
Digest: sha256:beff9ddade0df2a19e7f0449b60f65d9bdc0b9c1e7b3199fa5a8c6f39a5f331a
Status: Downloaded newer image for joonion/c-programming:latest
docker.io/joonion/c-programming:latest
ubuntu@ip-172-31-32-84:~$
```

> sudo docker images

```
ubuntu@ip-172-31-32-84:~$ sudo docker images
REPOSITORY              TAG       IMAGE ID       CREATED         SIZE
joonion/c-programming   latest    8c1f1de04eb3   22 months ago   692MB
ubuntu@ip-172-31-32-84:~$
```

> sudo docker run joonion/c-programming:latest

>sudo docker ps –a

```
ubuntu@ip-172-31-32-84:~$ sudo  docker ps -a
CONTAINER ID   IMAGE                   COMMAND   CREATED          STATUS                     PORTS     NAMES
8e36b7fecbd5   joonion/c-programming   "bash"    13 seconds ago   Exited (0) 12 seconds ago            condescending_rosalind
ubuntu@ip-172-31-32-84:~$
```

$sudo apt-get install build-essential

$sudo apt-get install git

>create c program using command

nano first.c

```
#include<stdio.h>

void main()

{

      printf("hello");

}
```

```
  GNU nano 7.2
#include<stdio.h>
void main()
{
printf("hello\n");
}
```

>To save: ctrl+s

>To exit from nano: ctrl+x

>To run the program

$gcc first.c –o first

$./first

```
helloubuntu@ip-172-31-32-84:~$ nano first.c
ubuntu@ip-172-31-32-84:~$ gcc first.c -o first
ubuntu@ip-172-31-32-84:~$ ./first
hello
ubuntu@ip-172-31-32-84:~$ |
```