WEEK – 1

1. Digital Transformation Success Story of Netflix.

- Netflix is an American subscription video on-demand over-the-top streaming service owned and operated by Netflix Incorporated company. The service primarily distributes films and television series produced by the media company of some name from various genres.
- It is available internationally in multiple languages. Netflix was launched on January 16, 2007 as a Streaming media company. To enjoy the Netflix services, customers need to subscribe theplan they want and can experience the service provided.
- Its headquarters are situated in Los Galos, California, U.S. registration is required to experiencethis service and nearly 238.39 million active users are using the service as of June 30, 2023 reports.
- Netflix's founders were Reed Hastings and Marc Randolph. Current CEO of this Netflix company is Greg Peters. Netflix was once a DVD rental service. Now Netflix has transformedinto a leading global streaming platform.
- By leveraging (maximum use) digital technology and data analytics, Netflix personalized its content recommendations and optimized it streaming quality.
- This digital technology also enhanced in creating the original content based on viewer preferences. This led to the exponential growth and dominance in the video streaming industry.

Digital Transformation: Digital transformation is a Strategic approach that aims to improve operational efficiency, enhance decision-making, and create new business opportunities by integrating traditionally separate IT and OT systems.

Eg: Automating attendance management system etc.

Information Technology: IT refers to the use of computers, networks, and other physical devices, infrastructures and processes to create, process, store, secure and exchange all forms of electronic data. The commercial use of IT encompasses both computer technology and tele communications.

Operational Technology: It is the practice of using hardware and software to control industrial equipments and it primarilyinteracts with the physical world.

Eg: Manufacturing, transportation and more.

IT/OT Convergence:

- It is the integration of information technology systems with Operational technology systems.
- IT systems are used for data centric computing while OT systems monitor events, processes,devices and make adjustments in enterprise and industrial operations.
- IT/OT Convergence involves transferring information across both IT and OT networks.
- This connectivity can be used to enhance the value of the systems to deliver and progress our organization further towards our business goal.

2. Case Study of Digital transformation through IT and OT convergences.

IKEA

A large furniture retail manufacturing company or industry decided to undergo a digital transformation by integrating its Information technology (IT) systems & Operational technology (OT) to improve;

- Productivity
- Safety &
- Overall efficiency

Digital transformation challenges faced by this company:

- **Continuous evolution of customer needs** - Customer's expectations and demands have raised even when organizations put years of effort into digital transformation, customer needs can change throughout that as they are constantly looking for more intuitive and enhanced services.
- **Budget constraints** - Another challenge of digital transformation is the high costs that come with it. As this is a huge investment, organizations need to carefully plan the budget and comeup with a strategy that will address and respond to customers and organization's needs.
- **Lack of manpower** - When an organization aims digital transformation, lack of manpower isrequired to perform different tasks at various levels. Considering how complex digital transformation strategies are, the right skill and knowledge set are required to implement the necessary changes.
- **Security concerns** - As organizations adopt remote work, digital processes and cloud-based technology, they are exposed to higher levels of risk. Consequently, they are required to implement higher security measures and improve their cyber security to defend themselves against threats. Not protecting data and other valuable assets of an organization can lead to enormous risks and negative consequences.
- **Inflation and supply chain issues** - Procurement becomes more complex during inflationaryperiods. If increased costs are to be passed on to the buyer, then demand typically falls, therefore fewer good and services may be required by producers. This decreases the inflows and outflows of that industry.

Digital transformation implementation to overcome the above difficulties:

- **Meeting the customer's needs** – Organizations should stay up-to-date with the newest trendsand design the user experience by satisfying their needs.
- **Digital transformation budget planning -** It needs lot of money and should create the financial roadmap to facilitate the processes involved. A budget should be prepared based on overall goals and needs, strategies, priorities, timelines, planned outcomes and returns of investments etc.
- **Creating job opportunities** – Due to overflow of business in furniture manufacturing industry,lack of manpower was faced because hiring had been freezed due to insufficient funds in the company. By providing employment opportunities in order to increase the manpower as per the requirement. This can overcome the matter of unemployment.
- **Digital transformation security** – Organizations should implement security controls and policies, invest in new tools and technologies, implement risk management and train employees.
- **Changing the way decisions are made -** Data and analytics have played a crucial role in IKEA's digital transformation, as they have become increasingly embedded in the company's decision-making processes. By leveraging data, the company has been able to modernize its inventory management, logistics, fulfillment, and overall supply chain operations. This has ledto the adoption of new working practices and an increased focus on agility.
- **Embracing AR and VR technologies** - IKEA has been experimenting with augmented reality(AR) and virtual reality (VR) technologies to enhance customer experiences. The company hastested VR in stores to help customers visualize how furniture fits in their homes and has acquired California-based Geo-magical Labs to develop a 3D home visualization tool. This tool aims to democratize home design, making it accessible to everyone through smartphone devices.

Final Outcomes

- **Increased operational visibility** – It requires decision making for the production process andresponse regarding production issues.
- **Enhanced productivity** – The digital integration streamlined workflows and minimized the downtime, leading to a 15% increase in overall productivity.
- **Improved quality control** – Improving the quality control by controlling the process,regarding defects and enhancing product quality.
- **Predictive maintenance** – After improvising the quality of the product, predictive maintenance is very much important in order to reduce unplanned downtime by 20%
- **Skill development** – Proper training has to be given for the required work force in order toadopt new technology and systems.

Conclusion

In conclusion, IKEA's digital transformation journey serves as an inspiring example of how alegacy retail brand can successfully adapt to changing customer needs and business environments.

By staying true to its core values while embracing new technologies and practices, IKEAhas managed to reinvent itself for the future, paving the way for continued success in the digital age.

Here is the final stage that is conclusion for this is to achieve the target in monetary terms, as well as in terms of good will. By focusing regarding production process by taking propermeasures in adopting new technologies and recruiting more manpower for the required workforce, proper quality control for controlling regarding production issues etc. this leadsto the bullish market.

Hence, customer service and satisfaction are the key to have a huge success in the business.

3. Identify the Typical Process of Workflow that can be Automated.

Nintendo:

Nintendo Co., Ltd. Is a Japanese multinational video game company headquartered in **Kyoto**. It develops, publishes and releases both video games and video game consoles. Nintendo was founded in 1889 as Nintendo Karuta by craftsman Fusajiro. Since then, Nintendo has produced some of the most successful consoles in the video game industry, such as the Game Boy, the Super Nintendo and Switch.

Here are some typical processes that could be automated using various tools with in the context of the production industry like Nintendo:

1. Conceptualization and Design:

- **Automated Idea Generation**: AI-driven algorithms can analyse popular trends, player preferences, and market data to generate game concept ideas.
- **Procedural Level Design**: Tools can automatically generate game levels using predefined rules and algorithms.

2. Pre-production:

- **Asset Generation:** AI tools can create textures, models, and other art assets based on design specifications.
- **Script Writing:** AI can assist in generating dialogues, narratives, and scripts based on the game's story and genre.

3. Production:

- **Version Control:** Automated version control systems help manage source code, assets, and ensure synchronization among team members.

- **Bug Detection**: Automated testing tools can identify bugs, glitches, and inconsistencies in the game code.
- **Build Automation:** Continuous integration tools can automatically compile, build, and distribute new game builds to the team for testing.
- **Localization:** Automated systems can assist in translating game content into multiple languages.

4. Quality Assurance:

- **Automated Testing:** AI-driven testing tools can perform regression testing, identify performance issues, and ensure consistent gameplay across different devices.
- **Playtesting AI:** AI agents can simulate player behavior to identify potential balancing issues or areas of the game that need improvement.

5. Marketing and Promotion:

- **Social Media Automation:** Tools can schedule and post updates on social media platforms, reaching out to the game's audience.
- **Data Analytics:** Automated analytics platforms can provide insights into player behavior, helping the team make informed decisions about marketing strategies.

6. Post-launch Support:

- **Player Support Chatbots:** AI-powered chatbots can provide immediate assistance to players, answering common questions and troubleshooting issues.
- **Live Operations:** Automation can help schedule in-game events, updates, and content releases to keep players engaged.

7. User Feedback and Updates:

- **Sentiment Analysis:** Automated sentiment analysis tools can process user reviews and feedback to understand player reactions.
- **Feature Updates:** Automated systems can facilitate the integration of new features, improvements, and content updates.

8. Monetization and Analytics:

- **In-game Purchases:** Automation can manage in-game purchases, virtual currency, and transactions.
- **Monetization Analytics:** Automated systems can track player spending patterns and provide insights to improve revenue strategies.

9. Security and Anti-Cheat:

- **Automated Security Checks:** Tools can continuously scan for vulnerabilities and potential security breaches in the game's code.
- **Anti-Cheat Systems:** Automated systems can detect and prevent cheating or unauthorized behaviour in multiplayer games.

10. Data Backup and Recovery:

- **Automated Backups:** Regular automated backups of game assets, code, and databases ensure data integrity and disaster recovery.

4. How to Create Project Plan and Product Backlog for project and User Story Creation.

Step 1: Open the web browser &search for Jira Login.

Step 2: You can continue with your Gmail account or login to the Jira Tool.

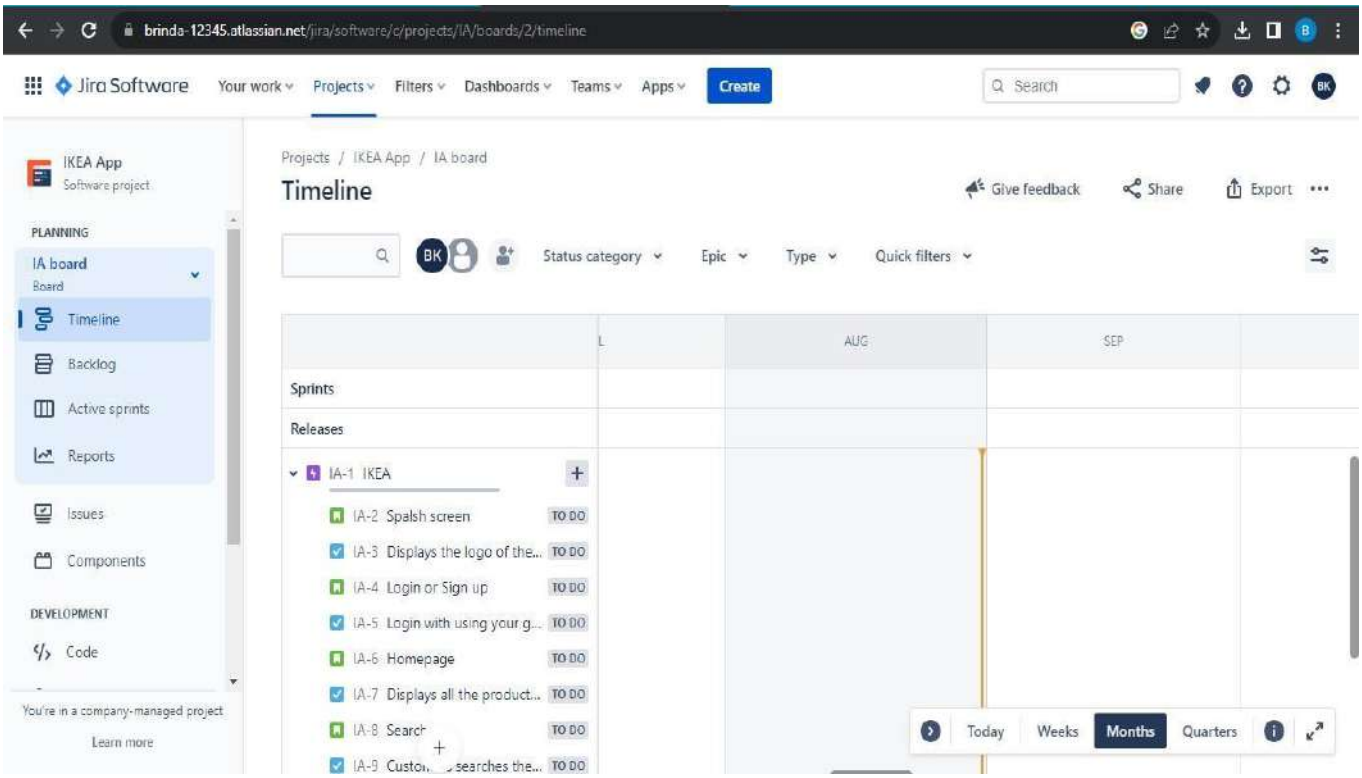
Step 3: Click on Jira software → project → create project from drop down menu.

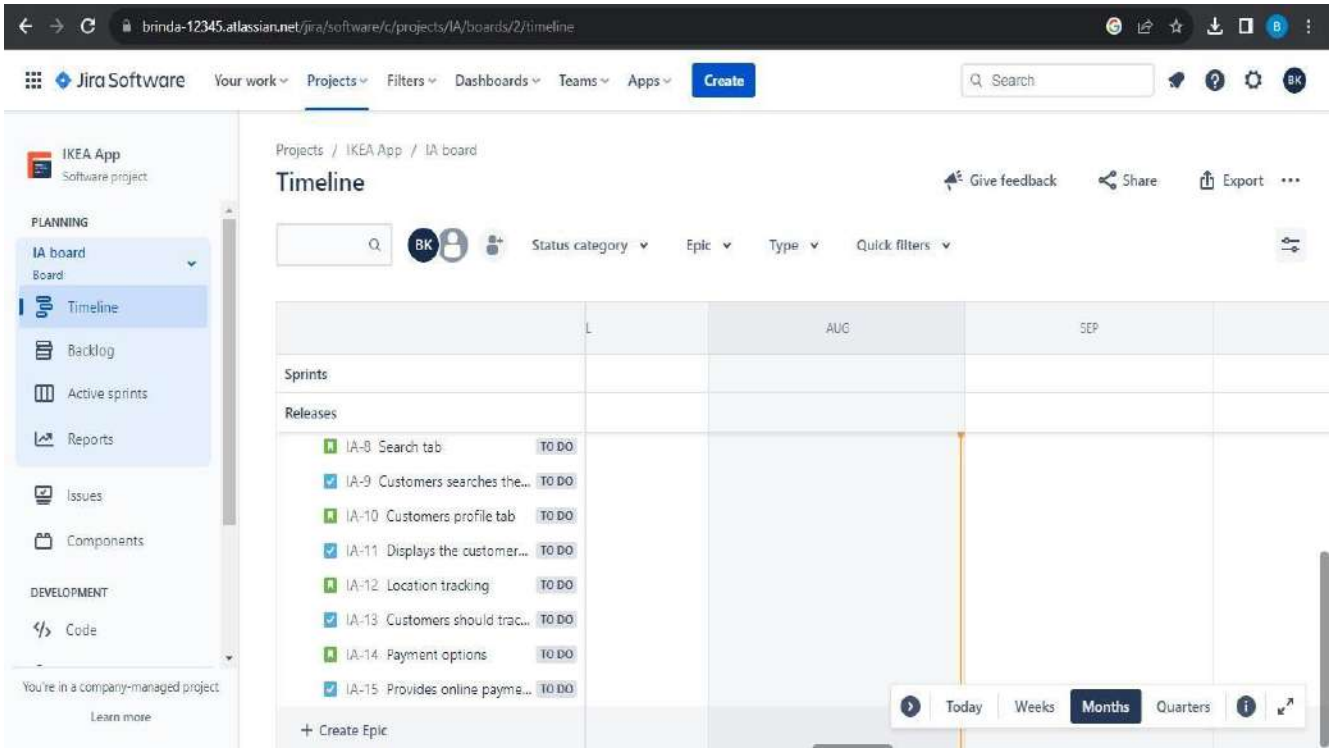
Step 4: Select **Scrum** & click on **Use template** and then click on **create**.

Step 5: Give a name to your project and Give a Description if you want.

Step 6: In Time line, create Epic name of the project with the defined user stories.

Step 7: Now go to Backlog, drag & drop the User stories and tasks and then divide that into different Sprints.

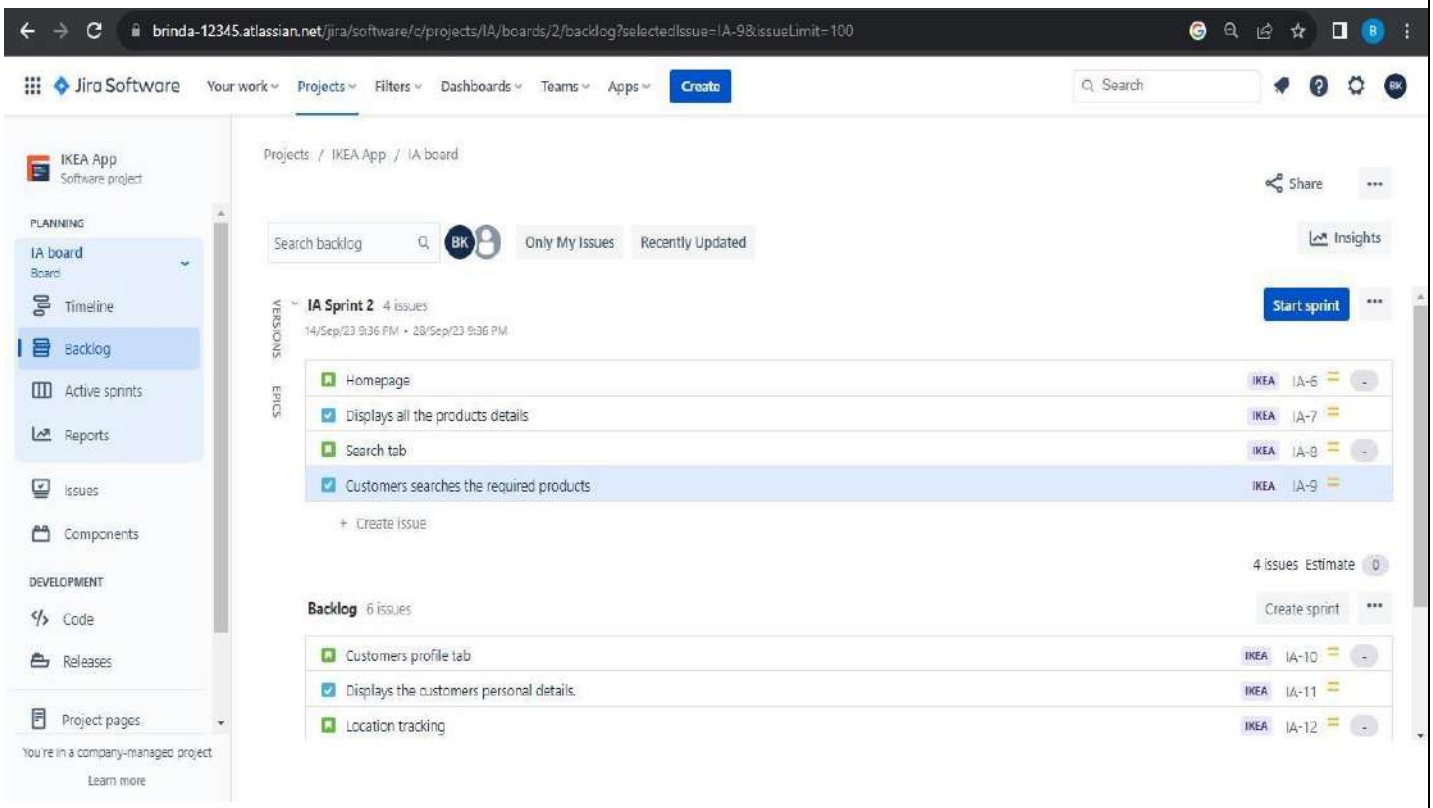


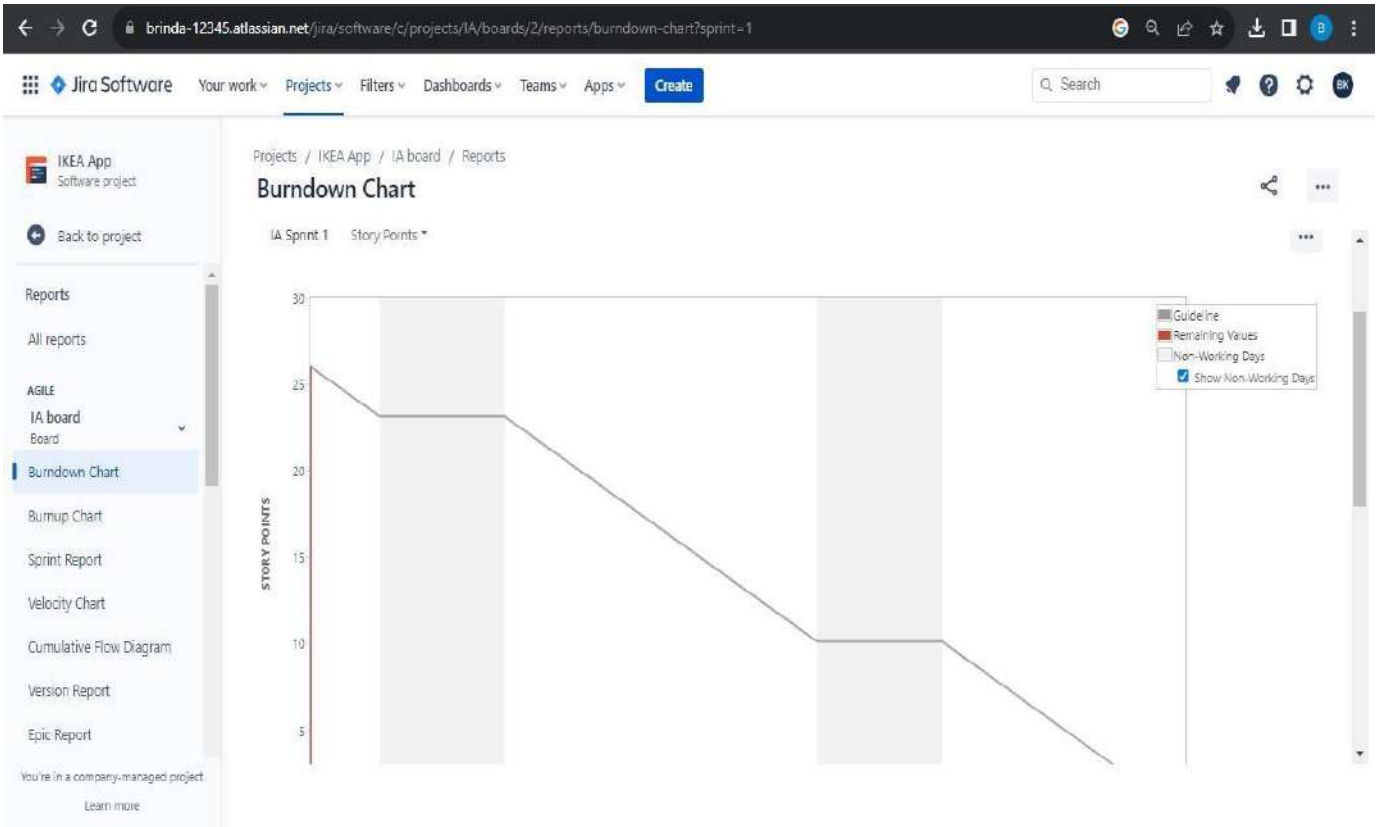


Step 8: Give the user story points to each user story and click on "Start Sprint".

Step 9: Go to Active Sprint & place the User Stories and tasks to the Done State and Click on'Start Sprint'.

Step 10: Click on the Burn Down Chart.



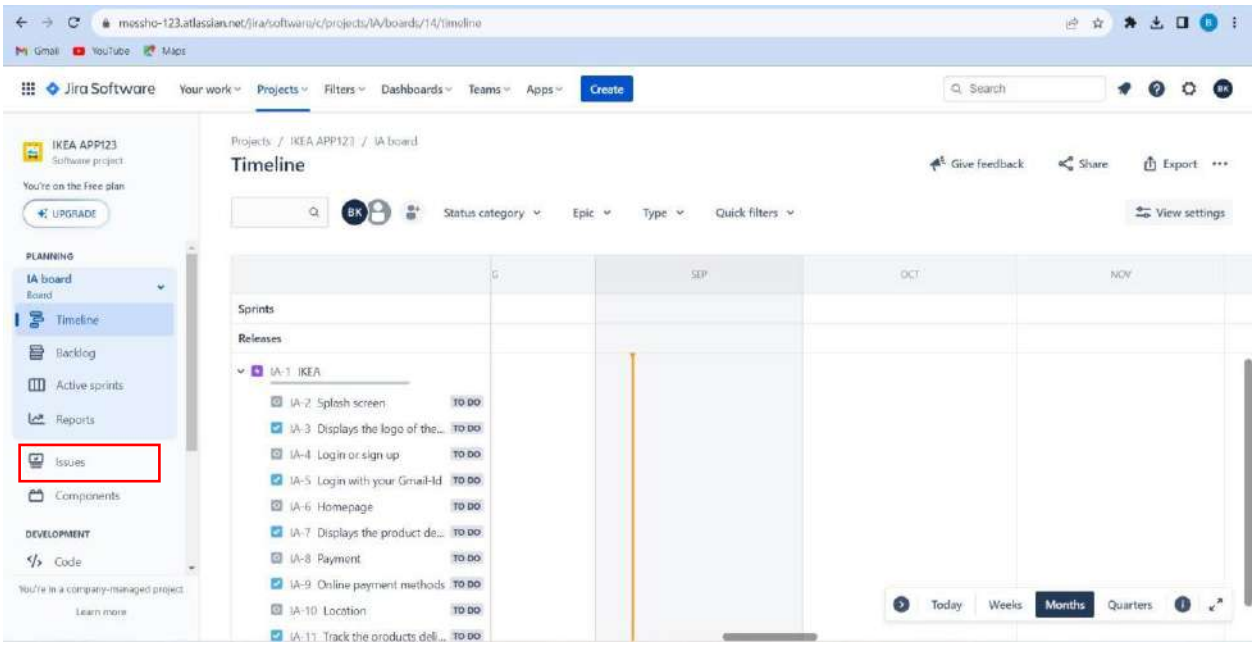

brinda-12345.atlassian.net/jira/software/c/projects/IA/boards/2/reports/burndown-chart?sprint=1
Jira Software
Your work
Projects
Filters
Dashboards
Teams
Apps
Create
Search
IKEA App
Software project
Back to project
Reports
All reports
AGILE
IA board
Board
Burndown Chart
Burnup Chart
Sprint Report
Velocity Chart
Cumulative Flow Diagram
Version Report
Epic Report
You're in a company-managed project
Learn more
Projects / IKEA App / IA board / Reports
Burndown Chart
IA Sprint 1
Story Points
STORY POINTS
30
25
20
15
10
5
Guide line
Remaining Values
Non-Working Days
Show Non-Working Days

WEEK – 2

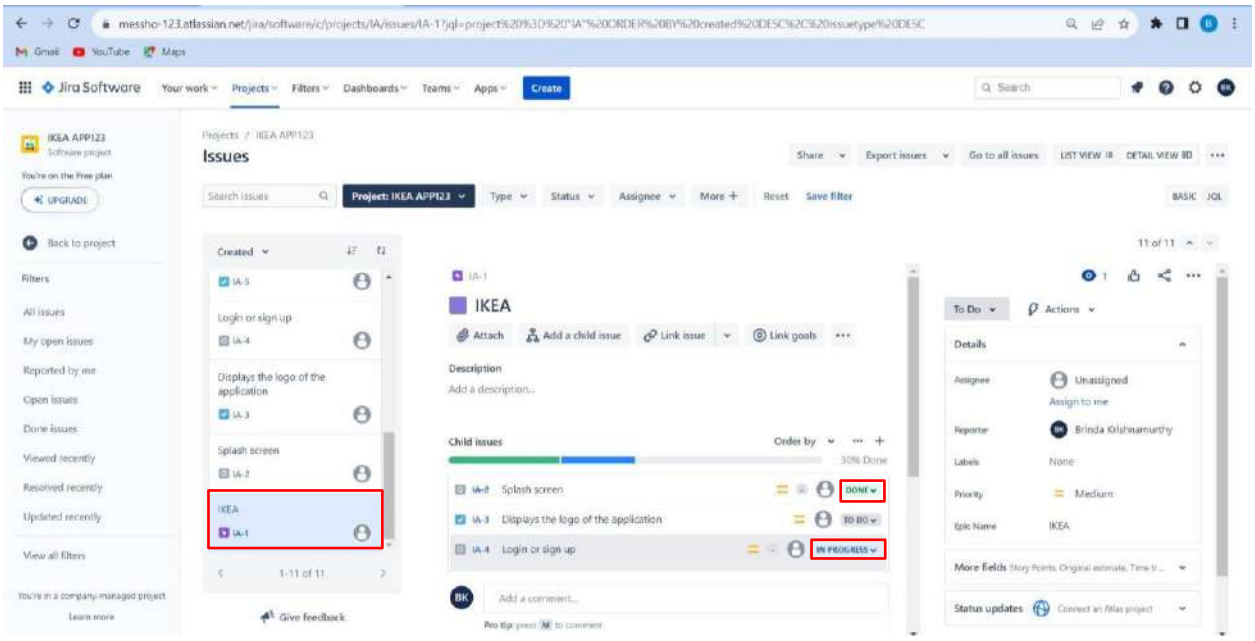
1. Creating Acceptance Criteria for the given User Stories by using Jira Software.

Steps for creating Acceptance Criteria.

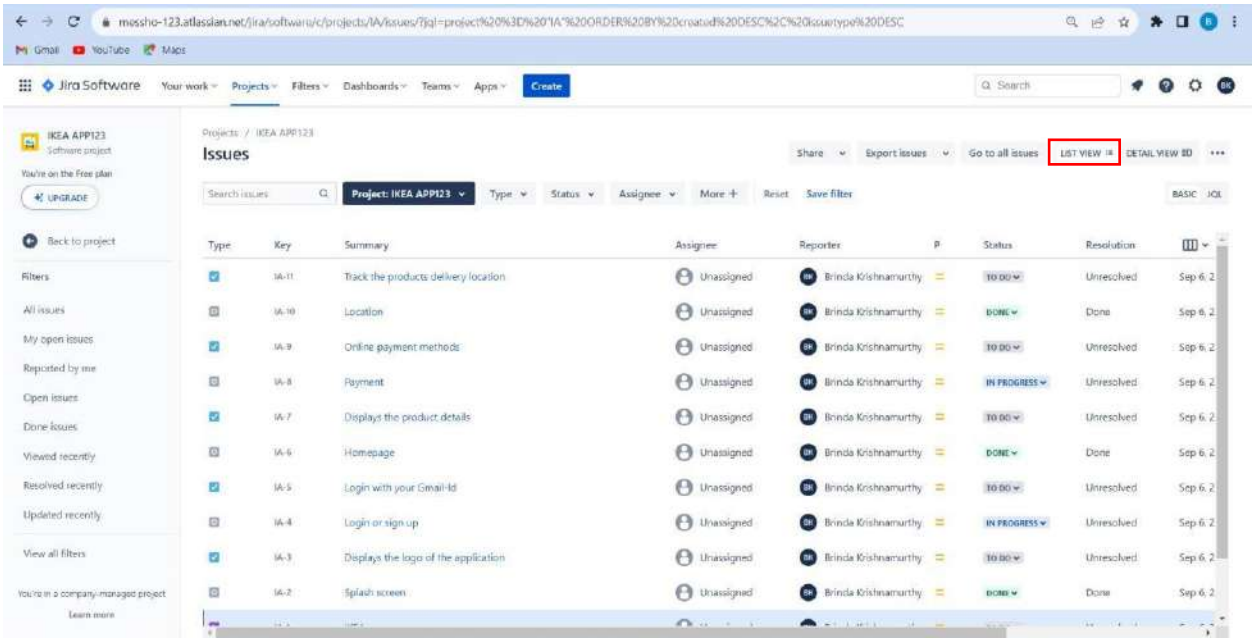
Step 1: After creating User stories with tasks in Timeline, place them in Backlog. Now go to “Issues” in the project.



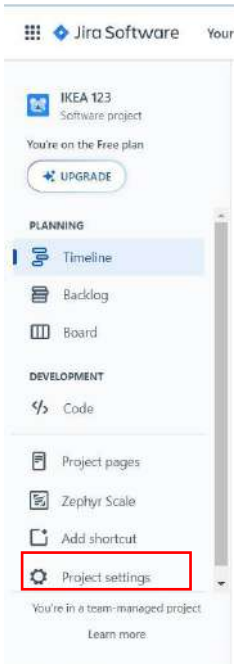
Step 2: Then select Epic and make changes in the right side of the “Child Issues” to **In progress** or **Done** statefor the given User stories and tasks.

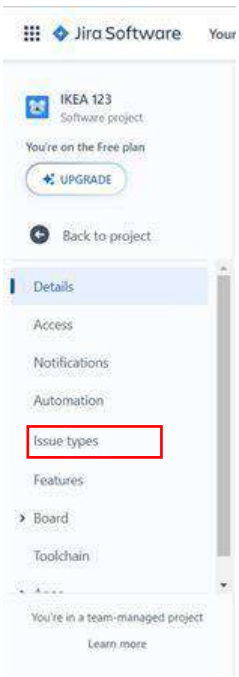


Step 3: Now Go to “List View” on the right side of the screen to view all the User stories and tasks of the Epic.

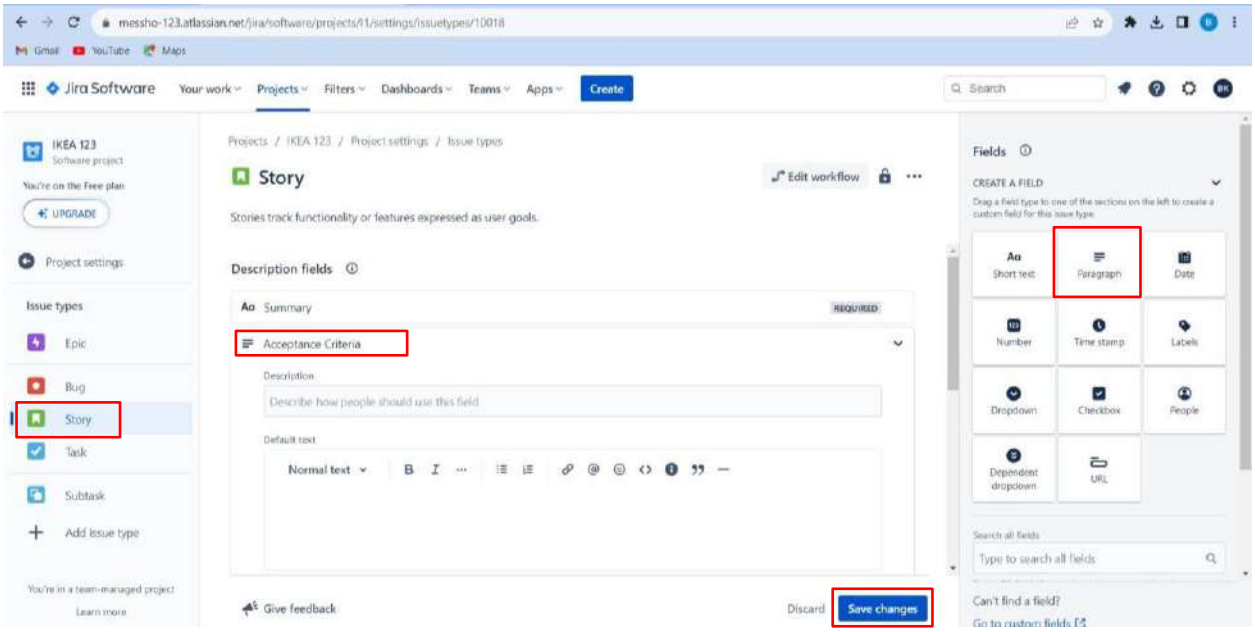


Step 4: Go to “Project Settings” and then select “Issue Types”.

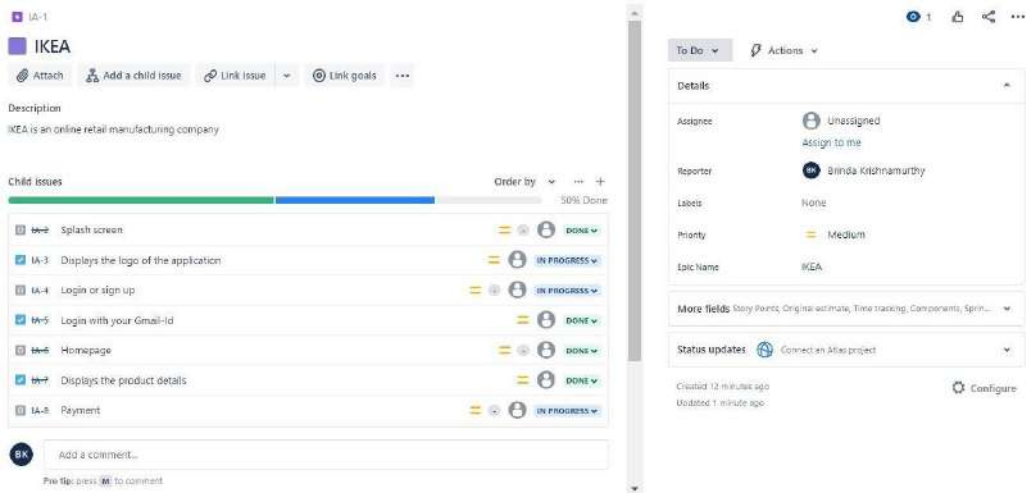




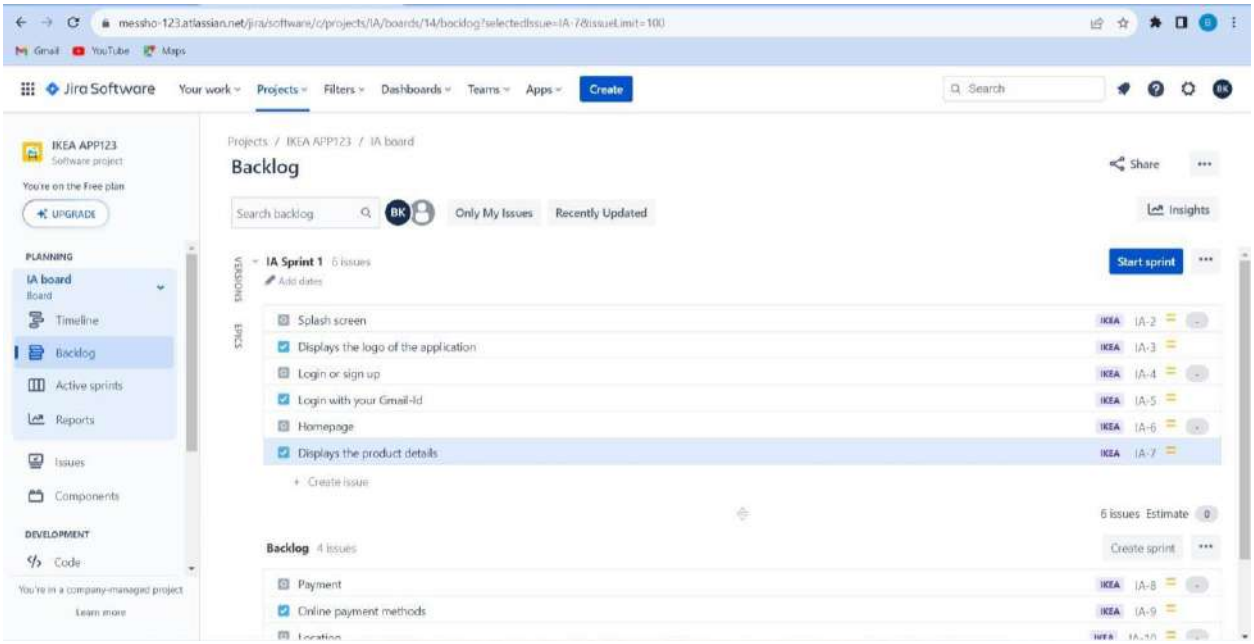
Step 5: Now select “Story” and also select the “Paragraph” in fields section and type “Acceptance Criteria”in the description box and then click on “Save Changes”.



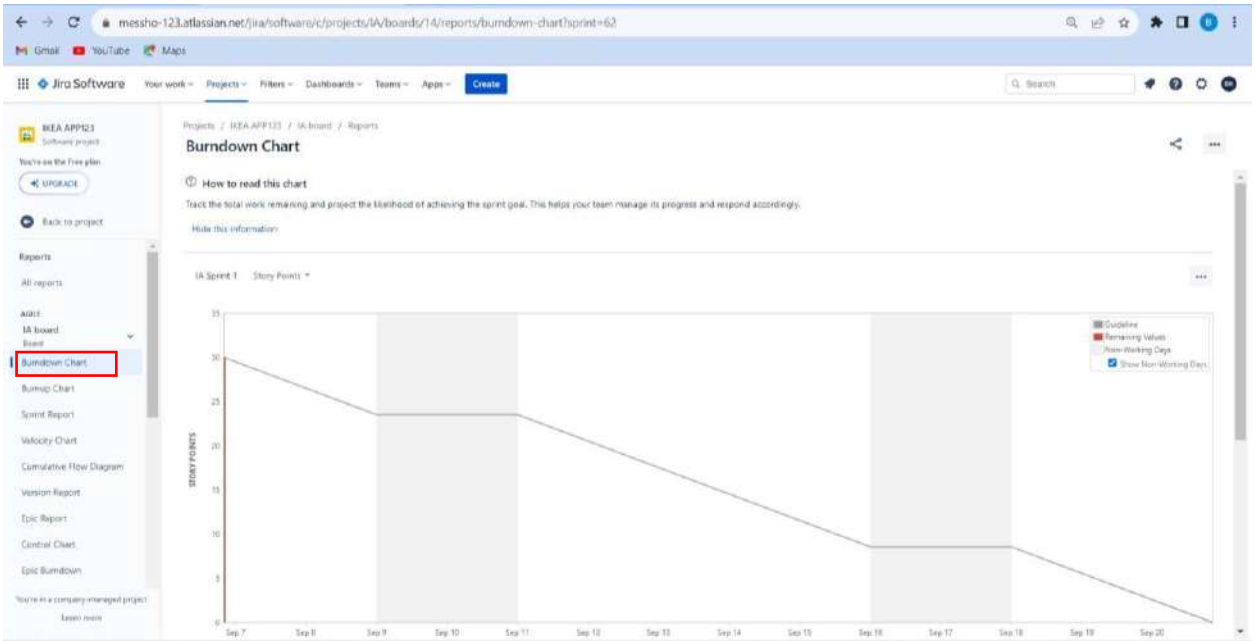
A. **Sprint Planning:** Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team. Unlike in sport, scrum encourages you to bealways sprinting so you can deliver working software, while continuously learning and improving.



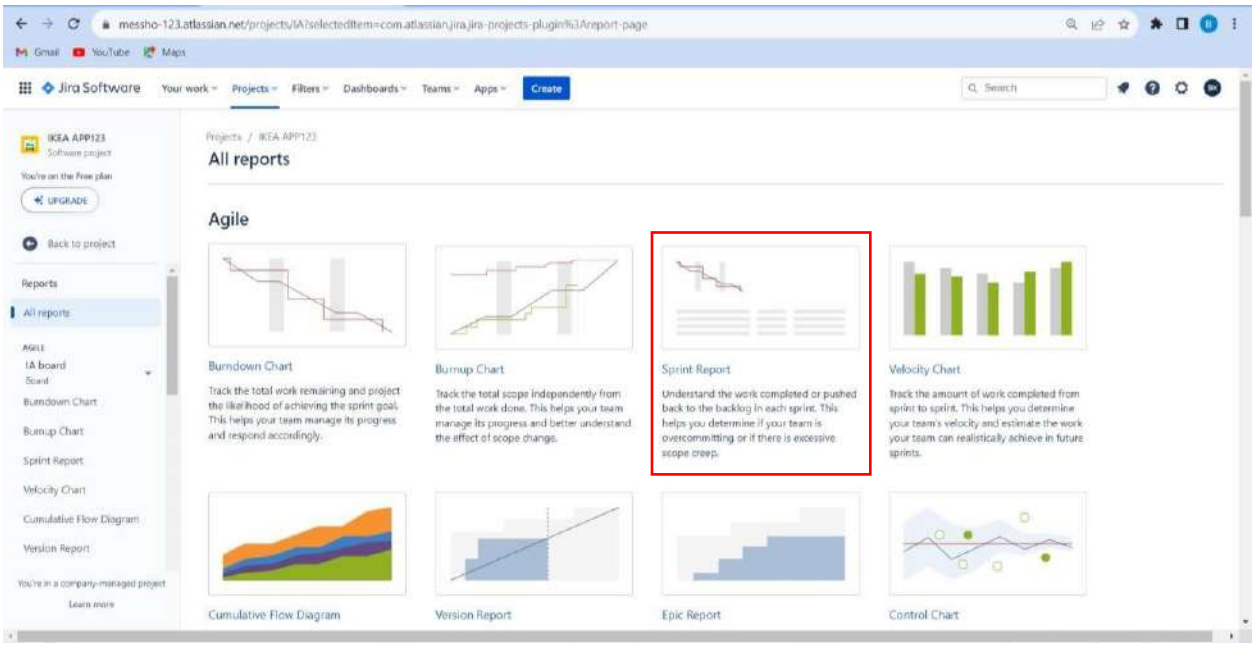
B. **Backlog Refinement:** JIRA Backlog Refinement otherwise called JIRA Backlog Grooming is a key process in Scrum. Backlog refinement is the process of reviewing, ranking, and editing your product backlog. Backlog refinement is an important tool in your product development process because it helps your development team build only the features and functionalities that the customer wants and the businessneeds. Backlog refinement is an ongoing process championed by the product owner, product managers, scrum master, and representatives from the development team.

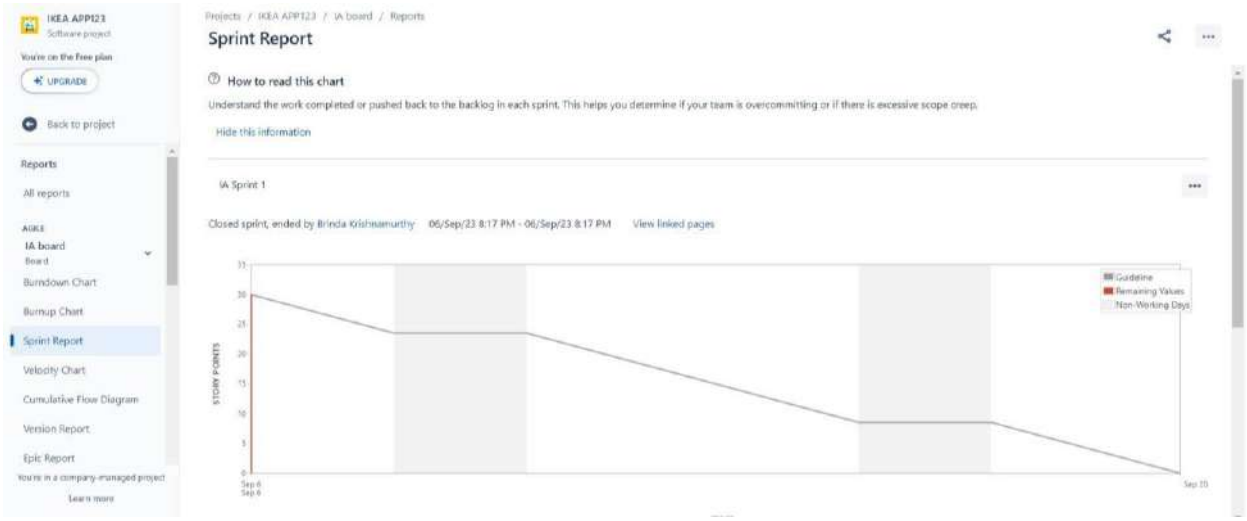


C. **Burndown Chart:** The burndown chart provides an overview of how much work remains and how fast scrum team work. As a project management and bug-tracking program, Jira improves team communication and collaboration. Use a Burndown Chart to track the total work remaining, and to projectthe likelihood of achieving the sprint goal. By tracking the remaining work throughout the iteration, a teamcan manage its progress, and respond to trends accordingly.

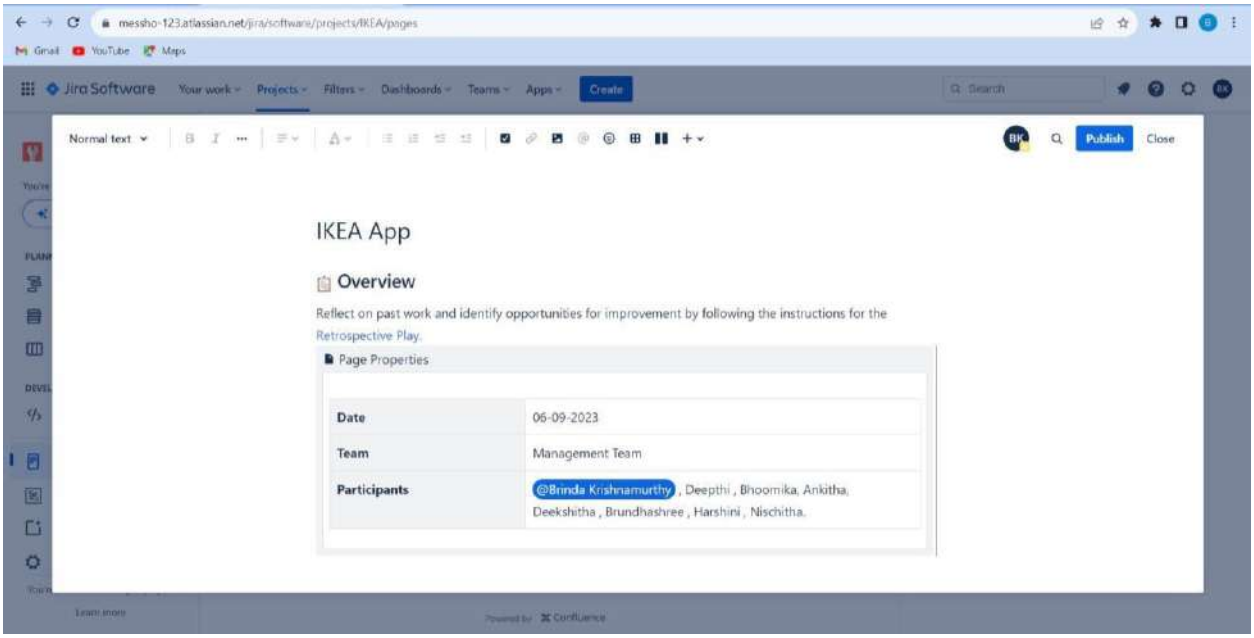


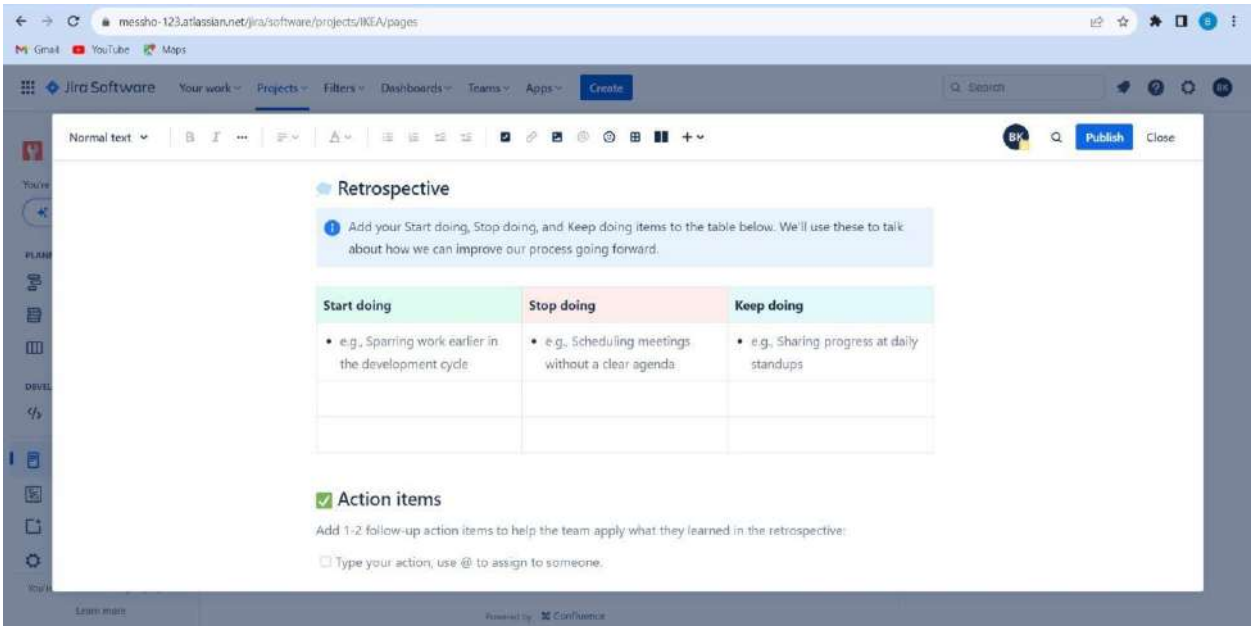
D. **Sprint Demo:** In a traditional scrum method, the sprint demo comes at the end of a sprint. A sprint demois your team's chance to show off to the business stakeholders. The work that they've done is important, and you want to share it with the people who are most excited about it. The best sprint demos are celebrations. Stakeholders and developers alike are coming together to share in the work they've accomplished. The team demonstrates fixed bugs, new features, and infrastructure necessary for future work.





E. **Sprint Retrospective:** The retrospective is a meeting that takes place at the end of a sprint. The lengthof the meeting depends on the time scale and complexity of the iteration. Retrospective Identify how to improve teamwork by reflecting on what worked, what didn't, and why. Here team members will share all sort of feedback whether it is Good or bad or it can be anonymous sometimes.





2. Design Principles for UI and UX design for the created User Stories by using Wire Framing Technology like Figma.

Figma: Figma is the leading collaborative design tool for building meaningful products. Seamlessly design, prototype, develop, and collect feedback in a single platform. Figma helps you gain control of version management. You have the ability to "fork" branches or components of the design away from the main project. Those forks can be edited, tested, and revised multiple times without affecting the rest of the project. When you're satisfied with that fork, you can safely merge it back into the overall design.

User Interface: The user interface (UI) is the point of human-computer interaction and communication ina device. User interface relates to the way in which a user interacts with a piece of software, a tool, an application, or an appliance. This includes graphical elements such as menus, buttons, and other controls essential for a product's function.

User Experience: User experience (UX) design is the process design teams use to create products that provide meaningful and relevant experiences to users. UX design involves the design of the entire process of acquiring and integrating the product, including aspects of branding, design, usability and function.

1. **Splash Screen**

2. **Login Screen**

3. **Homepage Screen**

3. Test cases for IKEA Application

1) Test cases for Splash screen page

Sl. No.	Test Steps	Test Data	Expected Result	Actual Result	Status
1.	Create an innovative User Interface	GUI should be presentable	Fascinating User interface screen	Splash Screen	Pass

2) Test cases for Login Screen

Sl. No.	Test Steps	Test Data	Expected Result	Actual Result	Status
1.	Providing many login types for the users	Users can login with their mobile no.,	Users should be able to login with no complex issues.	Login process should be achieved successfully.	Pass

		Username or Email – Id.			
2.	Providing 'Forgot password' option.	Resetting the new password.	Users should be able to reset the new password using their login details.	Resetting the new password has been achieved successfully.	Pass
3.	To create a new account	To sign in into the new account using the mobile no., Email, or with Google account.	To be able to create a new account and perform login process.	Cannot able to create a new account with the same mobile no.	Fail
4.	To enter into the Homepage screen	Search bar, innovative view of the products with the details.	Homepage should be presentable to the users	Users should be able to see the product details in a better manner.	Pass

3) Test cases for Homepage Screen

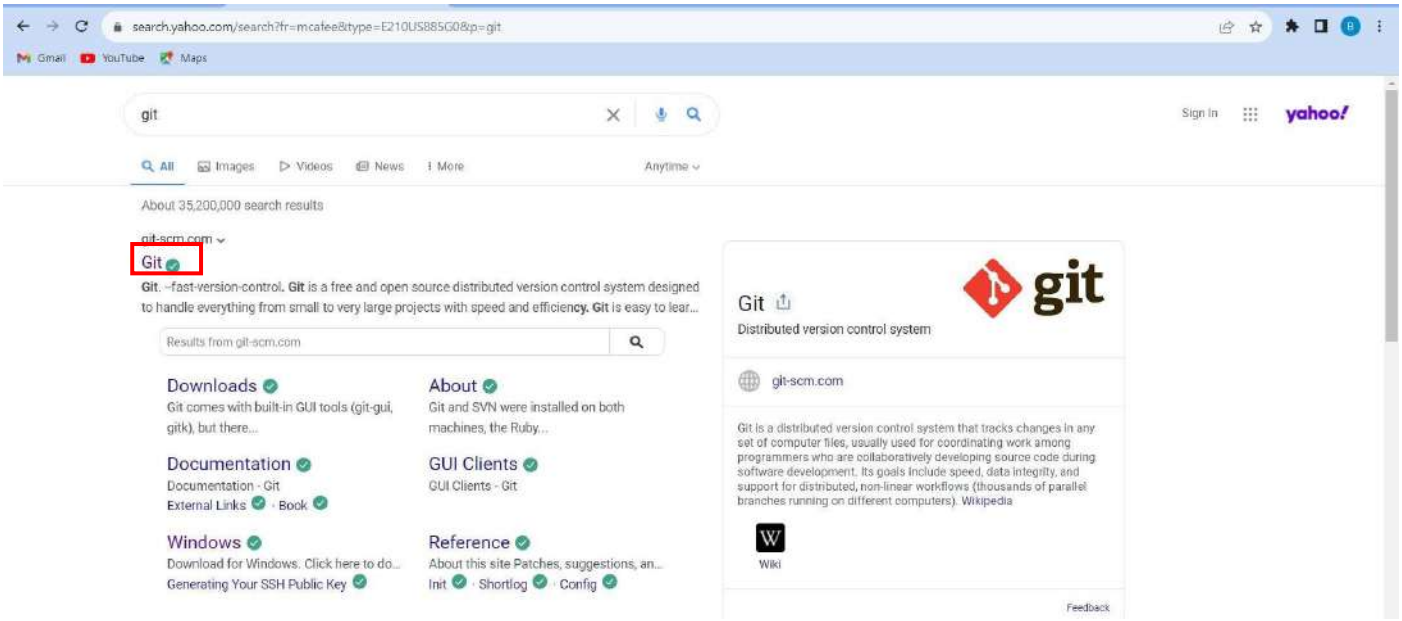
Sl. No.	Test Steps	Test Data	Expected Result	Actual Result	Status
1.	Create a Search bar based on SEO technology.	Searching of required products, materials or items.	Displaying of the products based on search results.	Searching of the available products in that platform.	Pass
2.	To check whether all the products are displayed and managed in a better way.	Updating the advertisements displayed if any changes required.	Customers should be able to see the products clearly with all the discounts offered.	Not displaying the desired search.	Fail
3.	To create a categories section to sort the products.	Doing categorization section based on the quality, durability and reviews of the users.	Categories section should be displayed with various buttons and options.	Categorization tab should be made visible in the Homepage itself.	Pass

4.	To create a user profile.	To manage and secure the User's credentials.	To be made visible in the Homepage screen with small profile icon.	Displayed on the Homepage under 'My account' name.	Pass

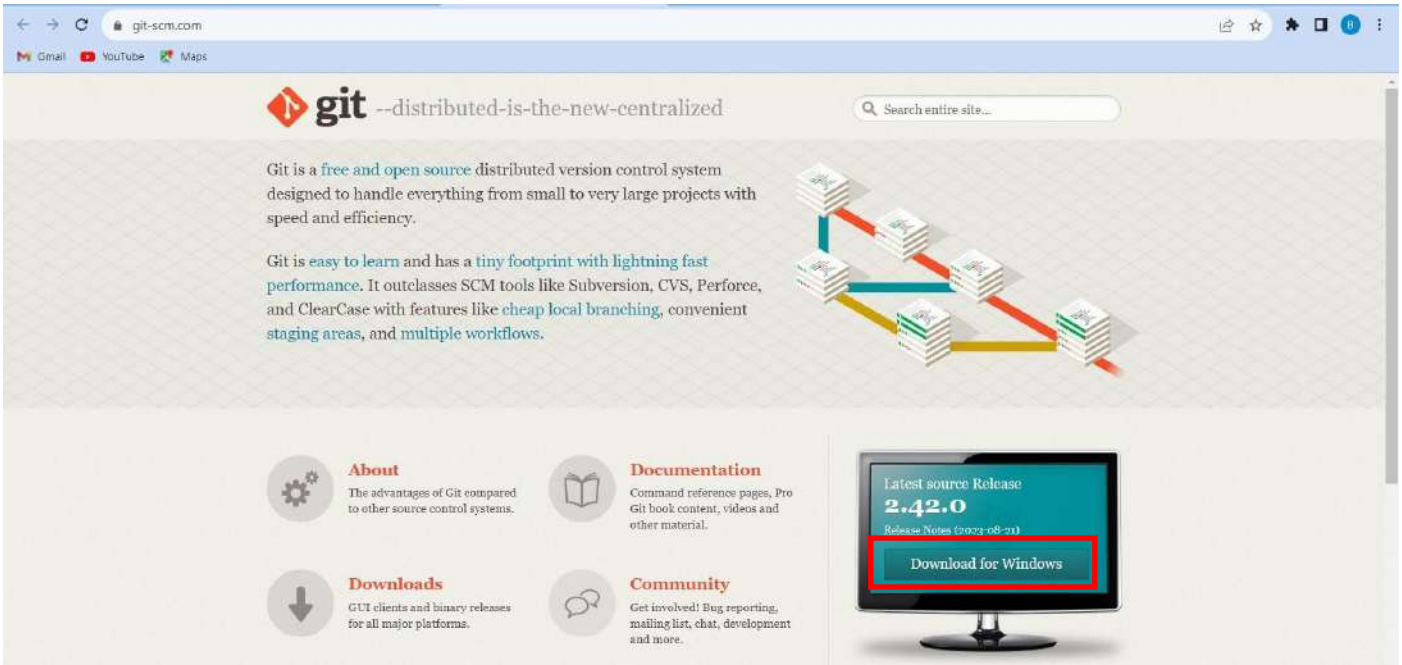
WEEK – 3

1. Git Client installation and setup

Step 1: Go to web browser and Search for ‘Git’ and tap on the first link.



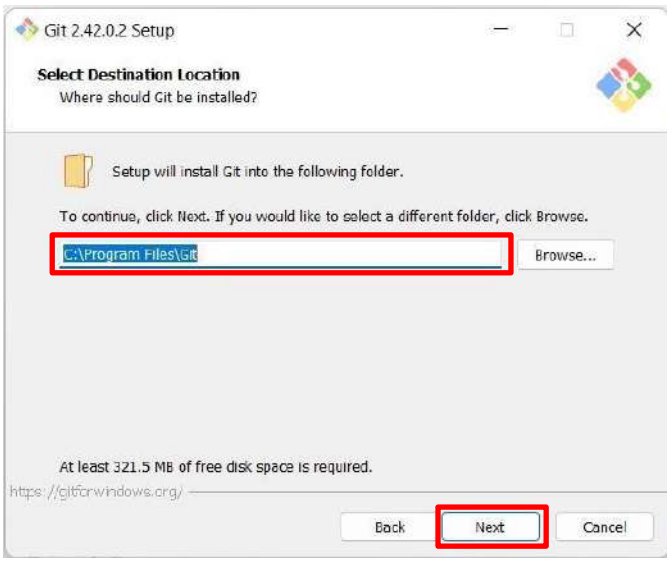
Step 2: Now click on ‘Download for Windows’.



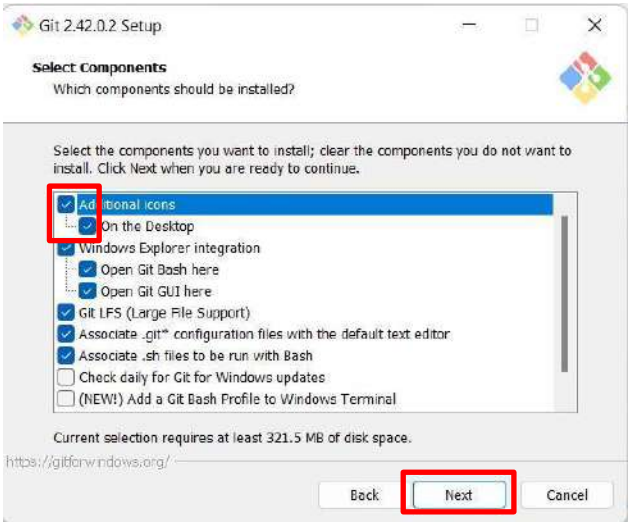
Step 3: Click on ‘Click here to Download’.

Step 4: Click on the downloaded file & click on ‘Next’.

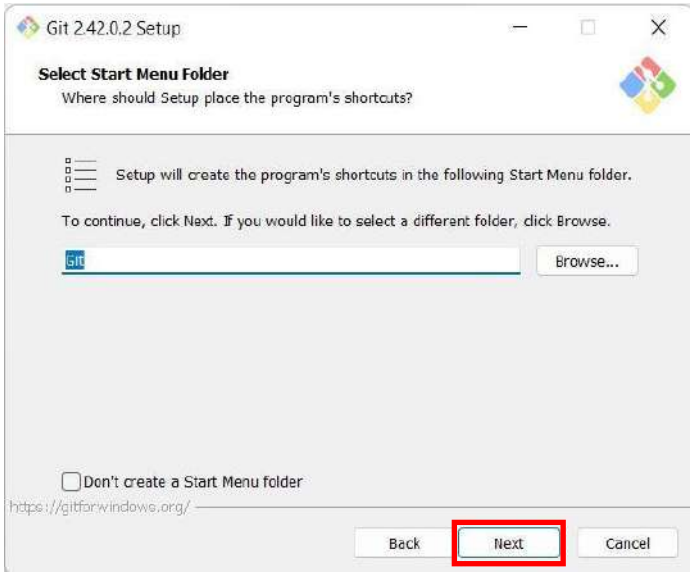
Step 5: Choose the file location to save this file & click on ‘Next’.



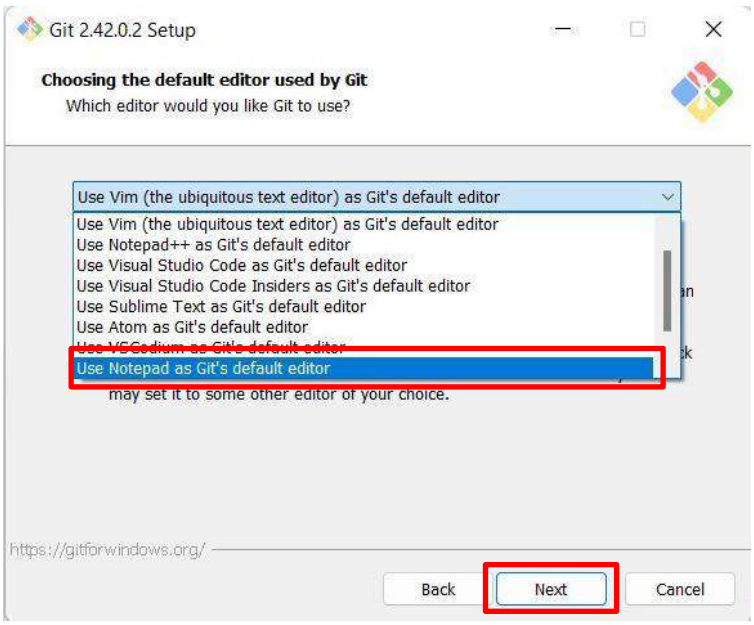
Step 6: Enable the top 2 checkboxes & click on ‘Next’.



Step 7: Then click on ‘Next’.

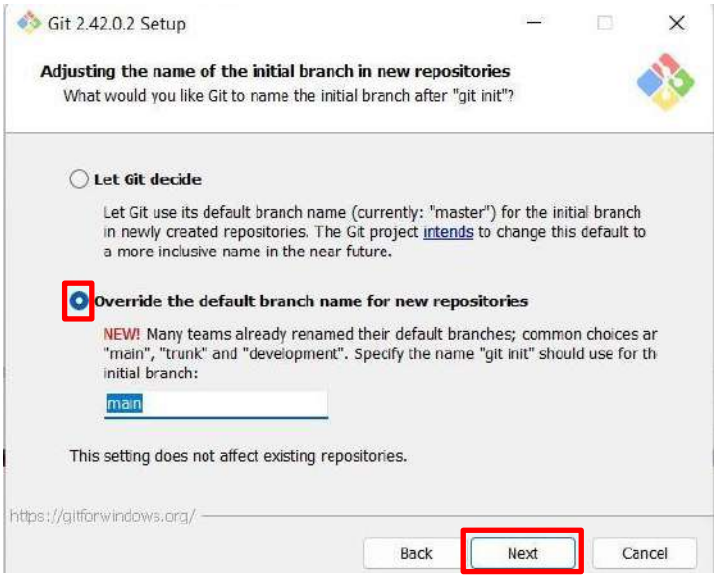


Step 8: Now select the ‘Use Notepad ad Git’s default editor’ option in the dropdown list & click on ‘Next’.

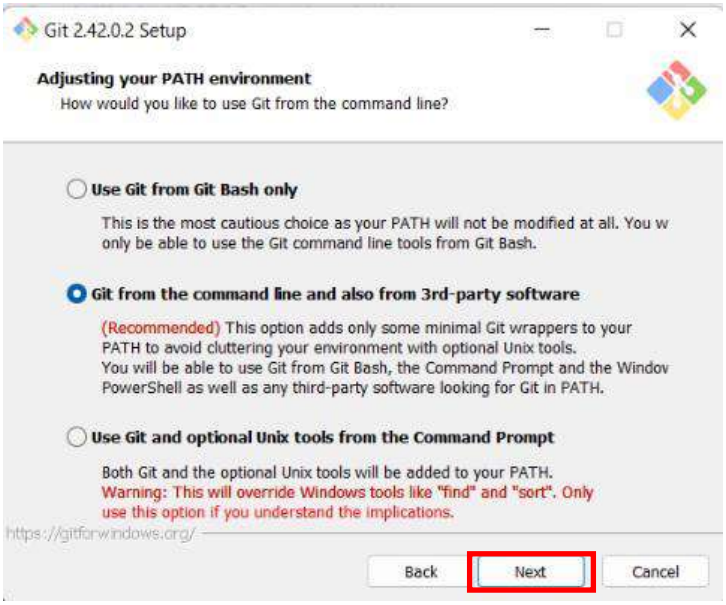


Step 9: Enable the 2nd checkbox ‘Override the default branch name for new repositories’ & click on

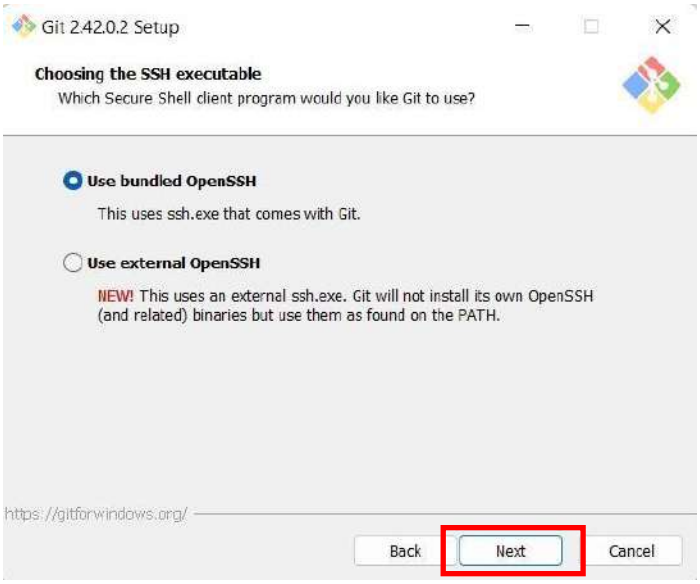
‘Next’;.



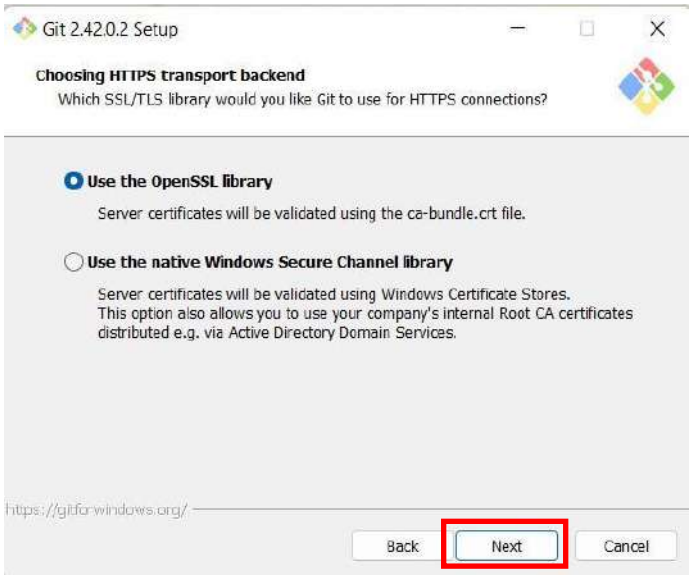
Step 10: Now click on ‘Next’.



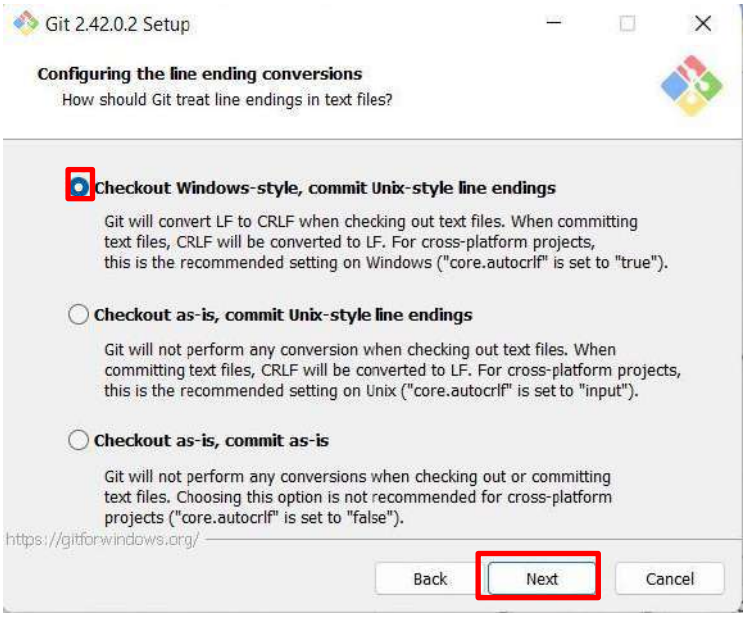
Step 11: Click on ‘Next’.



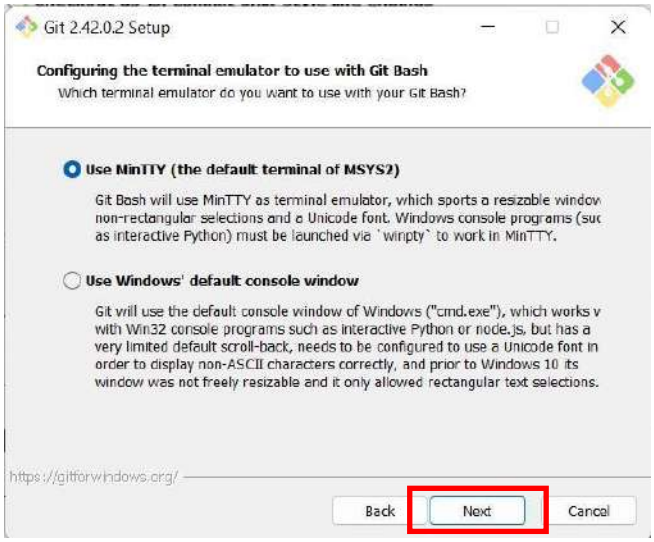
Step 12: Click on 'Next'.



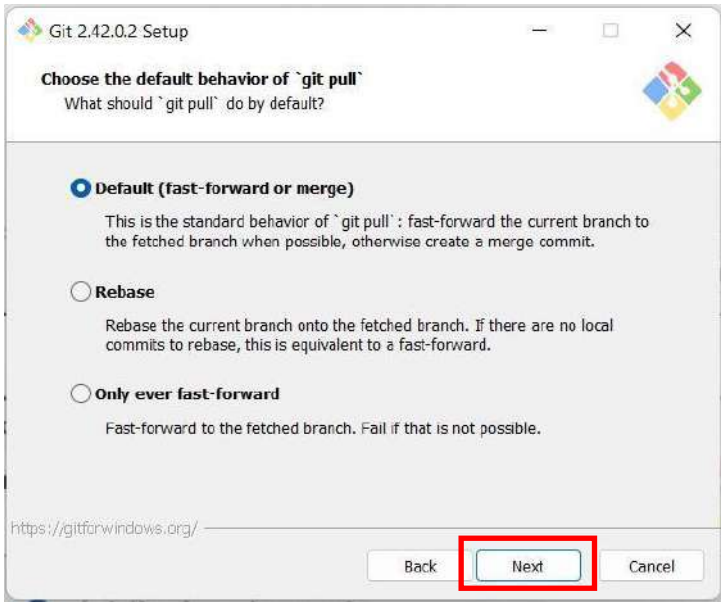
Step 13: Enable the 1st 'Checkout Windows-Style, Commit Unix-Style line endings' & click on 'Next'.



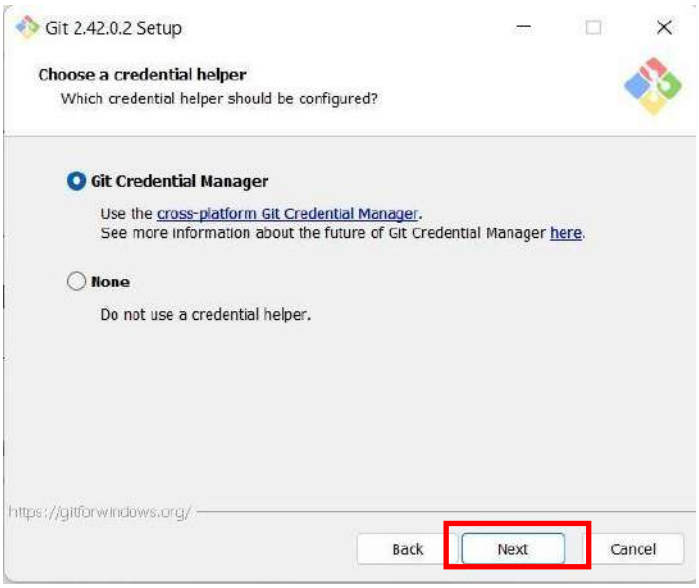
Step 14: Now enable the first option & click on ‘Next’.



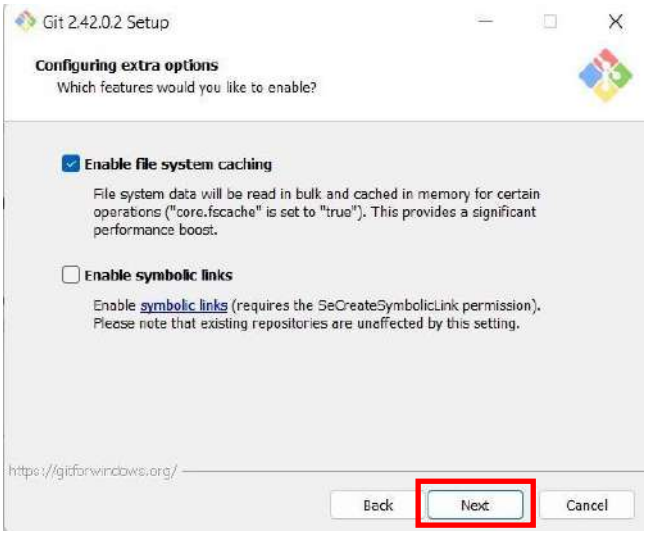
Step 15: Then enable the first option & click on ‘Next’.



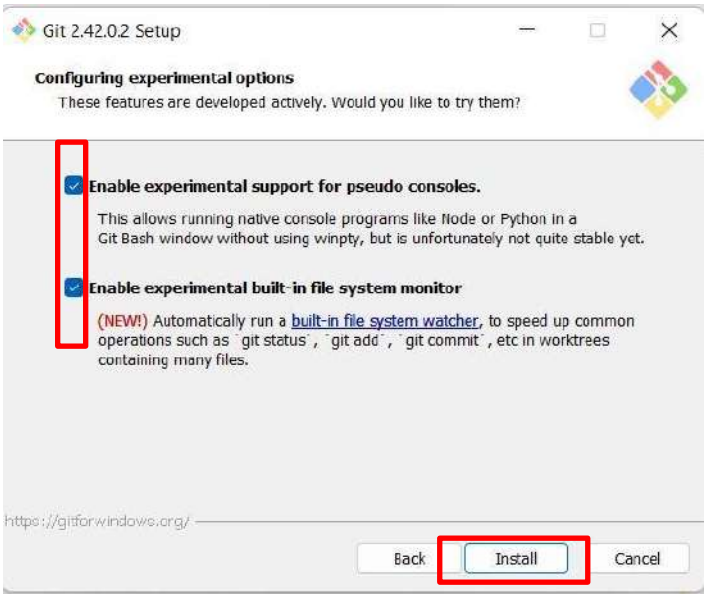
Step 16: Enable the first option & click on ‘Next’.



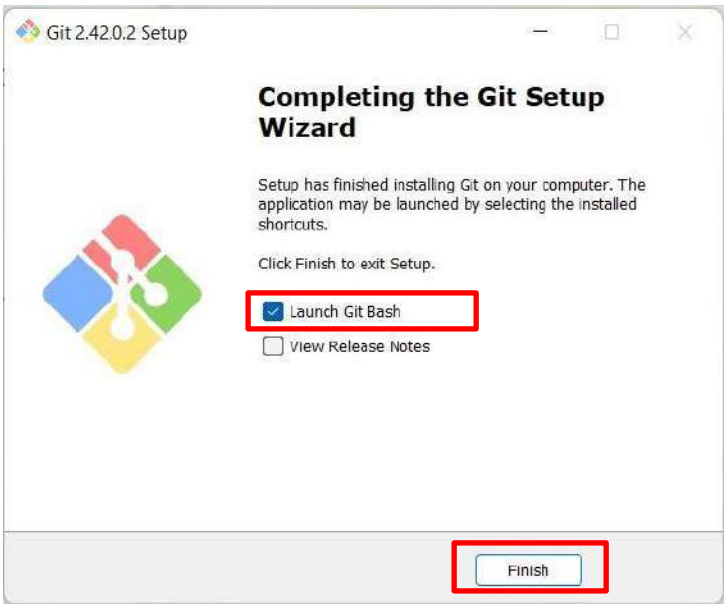
Step 17: Enable the first option & click on ‘Next’.



Step 18: Now enable both the checkboxes & click on ‘Install’.



Step 19: Then click on ‘Launch Git Bash’ & click on ‘Finish’.

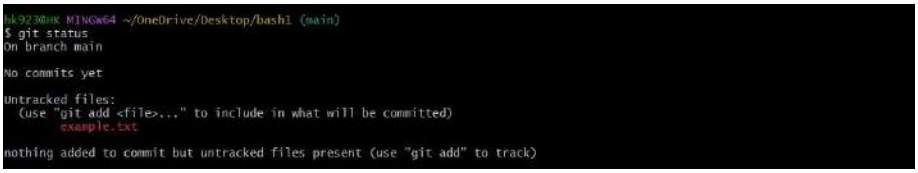


- **<u>Creating a repository</u>**

Step 20: Use the command **'git init'** to create a new repository.

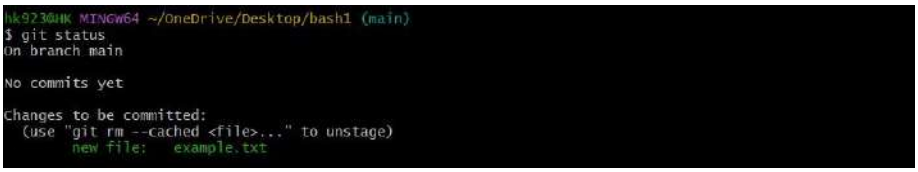
Step 21: Now enter the command **'ls -al'** to view all the created files.

Step 22: Enter the command **'git status'** to view the status.



Step 23: Enter the command **'git add example.txt'** to add a new text file.

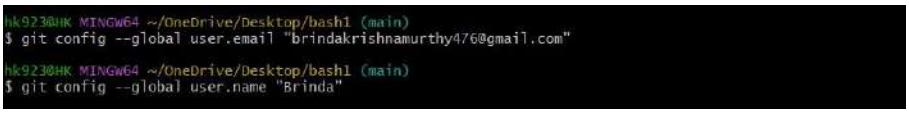
Step 24: Now enter **'git status'** to view the new text file.



Step 25: Enter the command **'git commit -m "First Commit" example.txt'** to commit the file.

Step 26: Now enter **'git config --global user.email 'brindakrishnamurthy476@gmail.com'** & **'git config**

--global user.name 'Brinda' if you've already committed a past file.



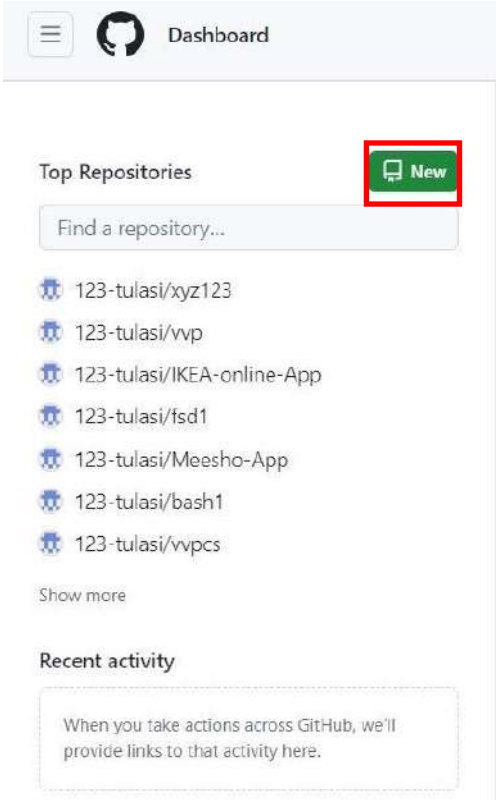
Step 27: Enter **'git commit -m "First Commit" example.txt'** to commit the file.

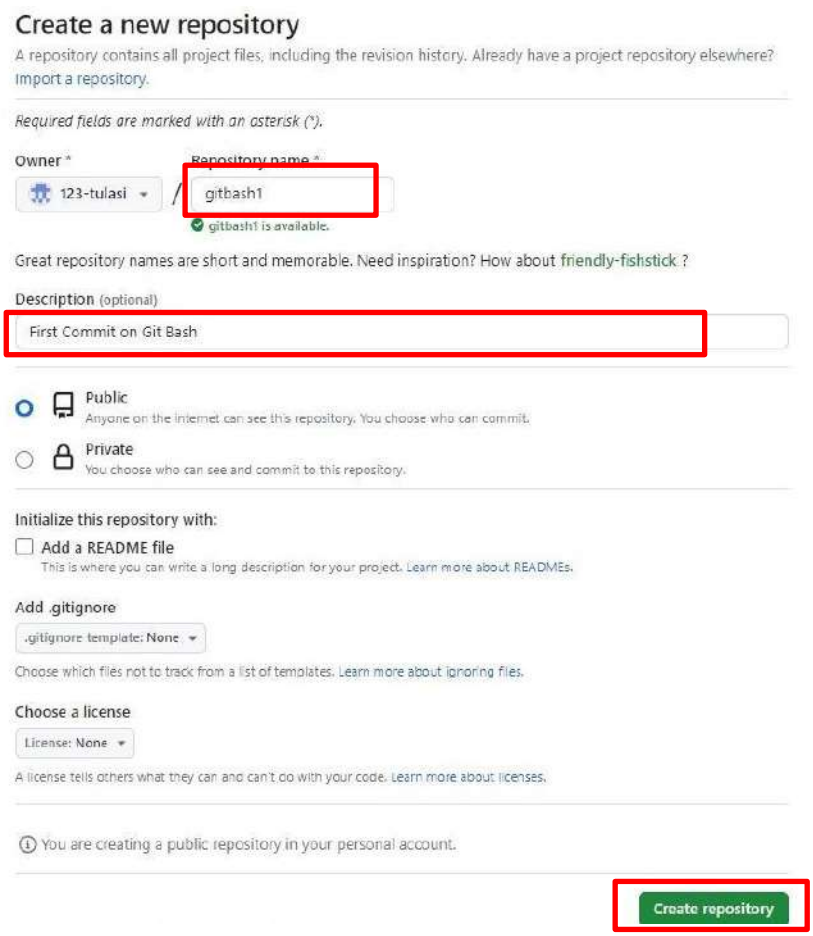
```
hk923@HK MINGW64 ~/OneDrive/Desktop/bash1 (main)
$ git commit -m "First Commit" example.txt
On branch main
nothing to commit, working tree clean
```

Step 28: Then enter **'git log'** to view the committed file.

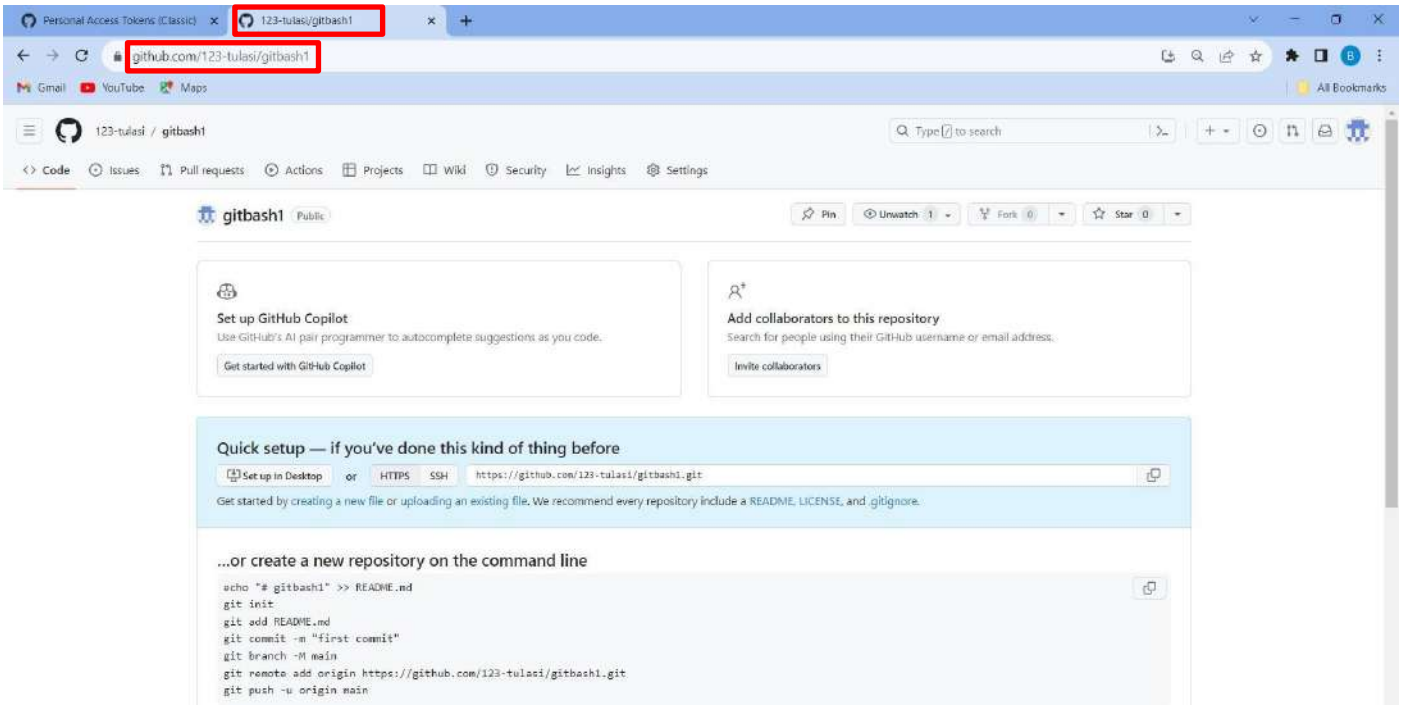
```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (main)
$ git log
commit 7201c3c4ea9f12e2f2b6cf50e25313fa1cbb503d (HEAD -> main)
Author: brinda <brindakrishnamurthy476@gmail.com>
Date:   Wed Sep 20 18:52:49 2023 +0530

    First Commit
```

Step 29: Now sign in to the GitHub account and create a new repository named **gitbash1** i.e., the givenname should match the opened Git Bash folder name. then add some description click on **'Create**

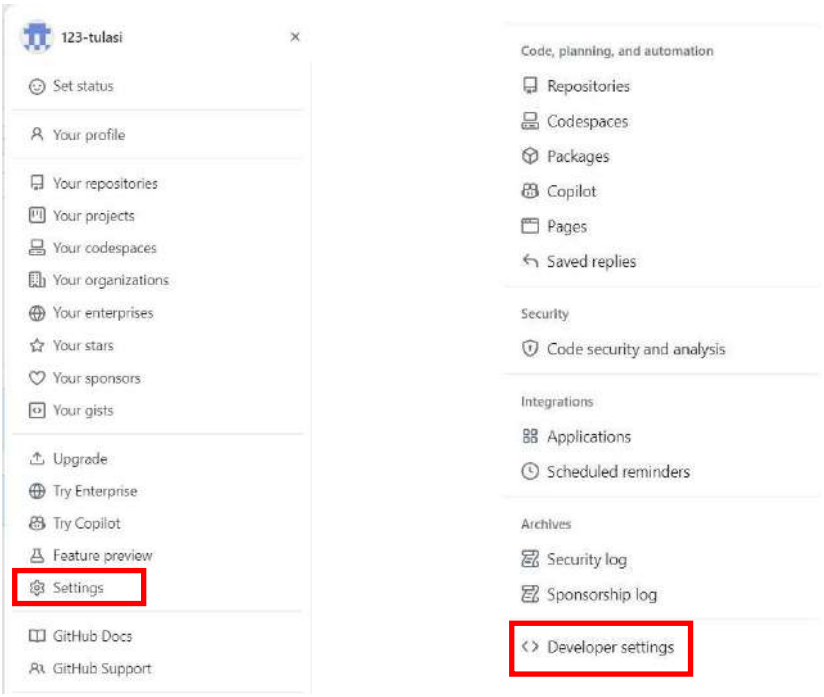




Step 30: Now copy the GitHub location and open it in New tab.

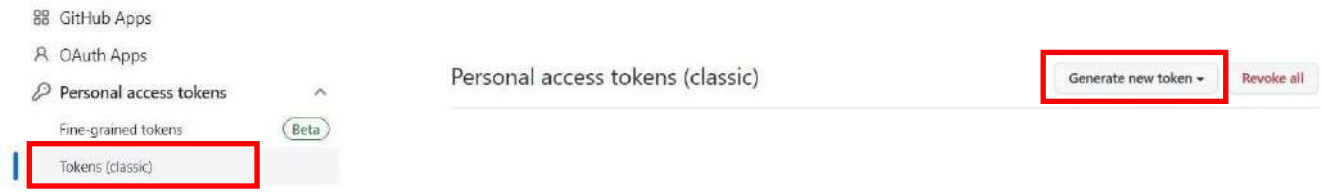


Step 31: Go to **'Settings'** & click on **'Deliverable Settings'**.

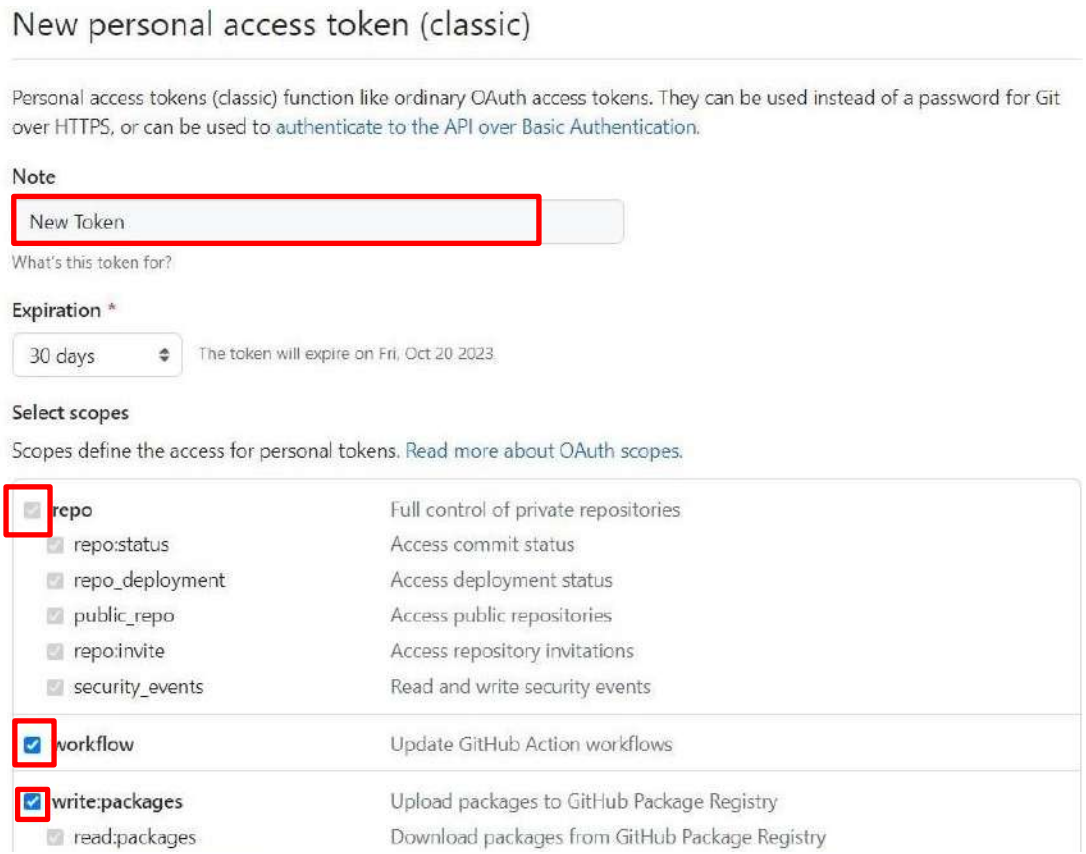


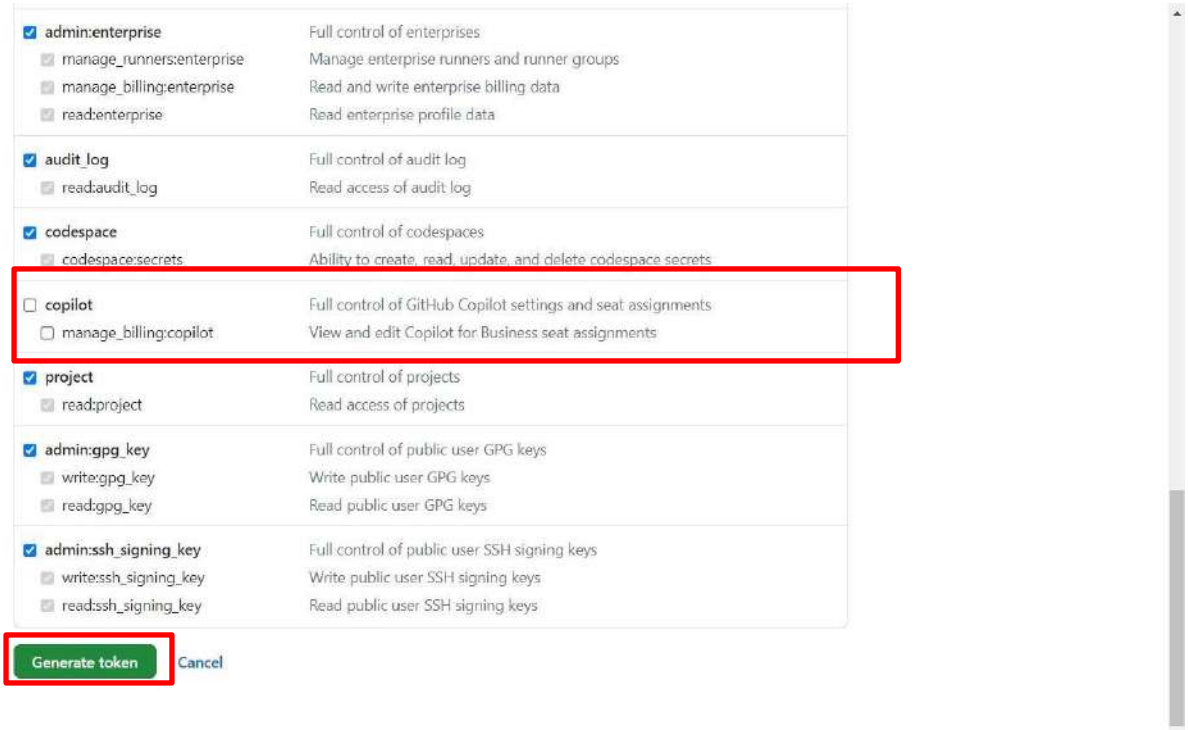
Step 32: Click on **'Personal access tokens'** & click on **'Tokens (classic)'**. And then select **'Token classic'**

under the 'Generate new Token' drop down list.

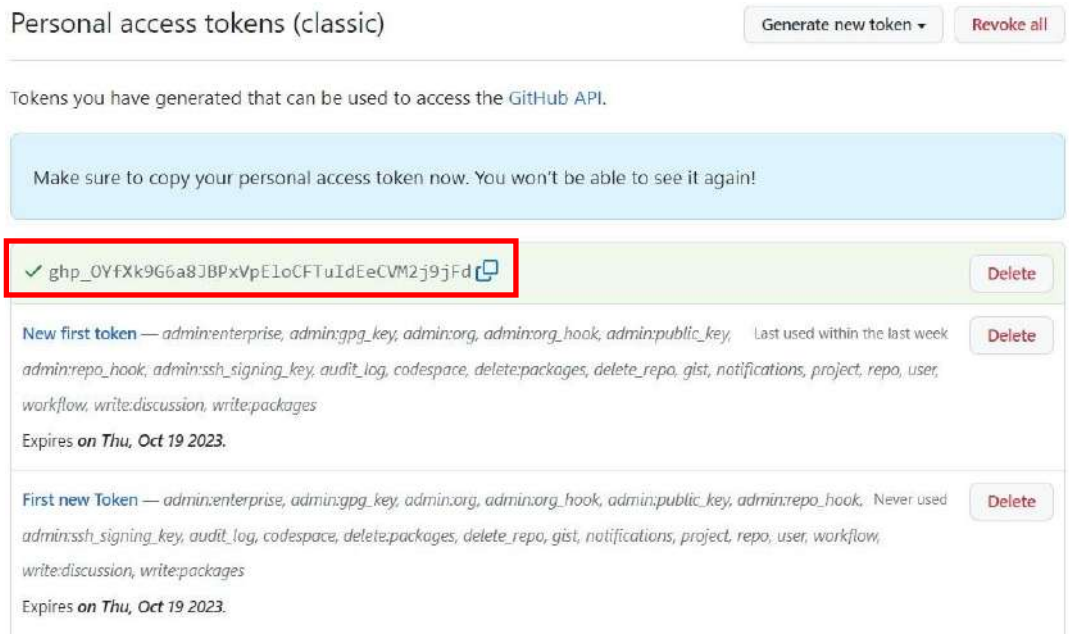


Step 33: Give a name for your Token **'New Token'** & enable all the checkboxes except **'copilot'** and thenclick on **'Generate Token'**.

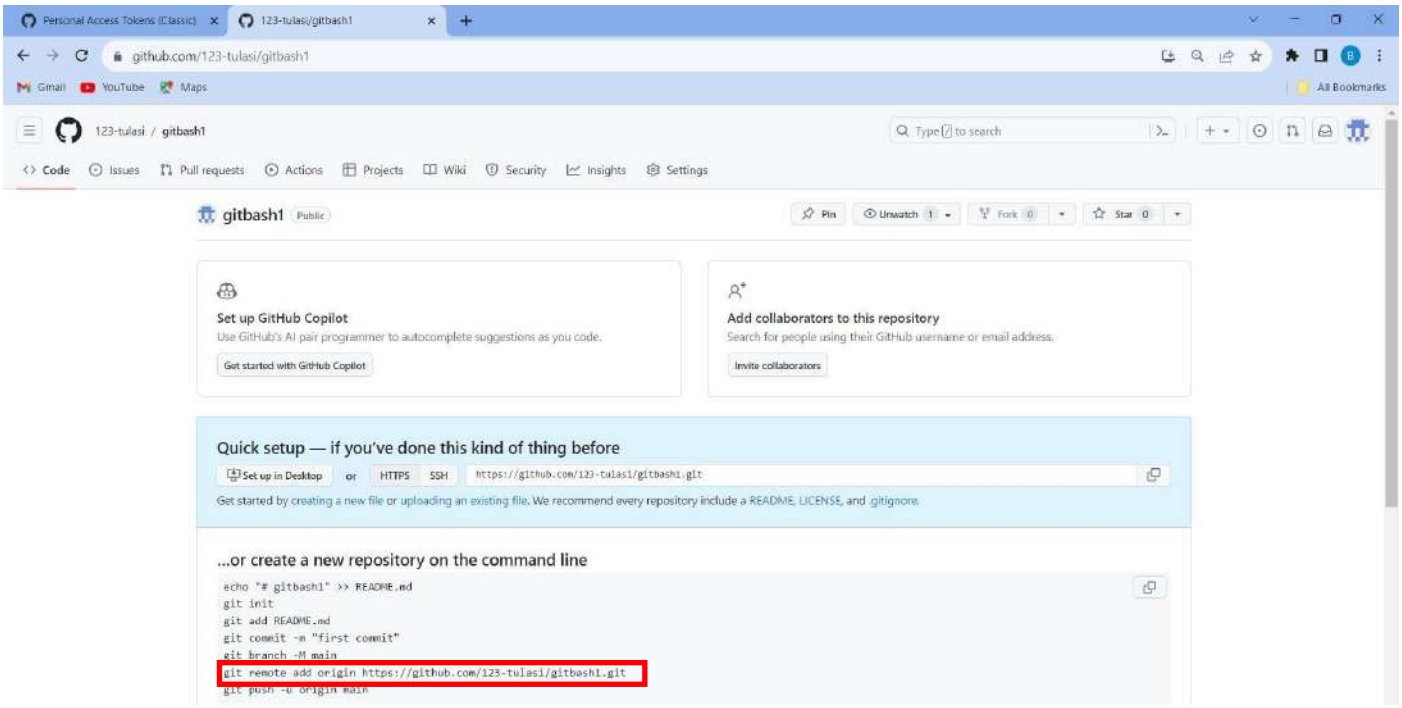




Step 34: Now copy the generated token link and paste it in the notepad.



Step 35: Also copy the below command **'git remote add origin https://github.com/tulasi/gibash1.git'** &paste that link in the same notepad.



Step 36: Now copy the first link and paste it after the **hhttps:// .....** @ & delete the **.git** command.

Step 37: Now paste the above created link in the Git Bash terminal & press Enter.

Step 38: Now again copy the **'git push -u origin main'**.

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

```
echo "# gitbash1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/123-tulasi/gitbash1.git
git push -u origin main
```

Step 39: Then paste that link into the Git Bash terminal and press Enter.

```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 242 bytes | 80.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/123-tulasi/gitbash1
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Step 40: Now type **'ls'** command to view created files.

```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (main)
$ ls
example.txt
```

- **<u>Creating a new branch and adding new text files.</u>**

Step 41: Give **'git branch newbranch1'** to create a new branch. And **'git checkout newbranchd1'** toswitch into that branch.

```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (main)
$ git branch newbranch1

hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (main)
$ git checkout newbranch1
Switched to branch 'newbranch1'
```

Step 42: Now create a new text file named **'file.txt'** and add some contents in it. Then use **'git add file.txt'**

command to add that file into the **newbranch1**.

```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (newbranch1)
$ vi file.txt

hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (newbranch1)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next time Git touches it
```

Step 43: Enter **'git commit -m "first commit" file.txt'**.

```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (newbranch1)
$ git commit -m "first commit" file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next time Git touches it
[newbranch1 870beeb] first commit
 1 file changed, 4 insertions(+)
 create mode 100644 file.txt
```

Step 44: Now enter **'git log'** to view the committed file.

```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (main)
$ git log
commit 870beebb376a3eadf46ca2ca1254d9c451e4ce79 (HEAD -> main, newbranch1)
Author: Brinda <brindakrishnamurthy476@gmail.com>
Date:   Wed Sep 20 19:53:12 2023 +0530

    first commit
```

Step 45: Then enter **'git add .'** & **'git status'** command.

```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (newbranch1)
$ git add .

hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (newbranch1)
$ git status
On branch newbranch1
nothing to commit, working tree clean
```

- ## Merging of Branches

Step 46: Enter **'git checkout main'** & type **'git merge newbranch1'** to merge the **newbranch1** to the **main**

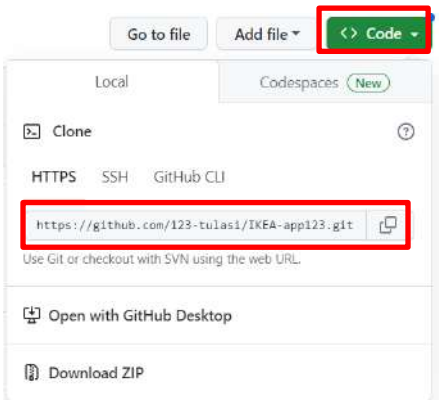
branch. And also enter **'ls'** to view merged files.

```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (newbranch1)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (main)
$ git merge newbranch1
Updating 7201c3c..870beeb
Fast-forward
 file.txt | 4 ++++
 1 file changed, 4 insertions(+)
 create mode 100644 file.txt

hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (main)
$ ls
example.txt  file.txt
```

- ## Cloning a repository from GitHub to GitBash terminal.

Step 47: Now select any repository in your GitHub account and click on **'Code'**. Then copy that **HTTPS**

location & paste into the GitBash terminal.




```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (main)
$ git clone https://github.com/123-tulasi/IKEA-app123.git
Cloning into 'IKEA-app123'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

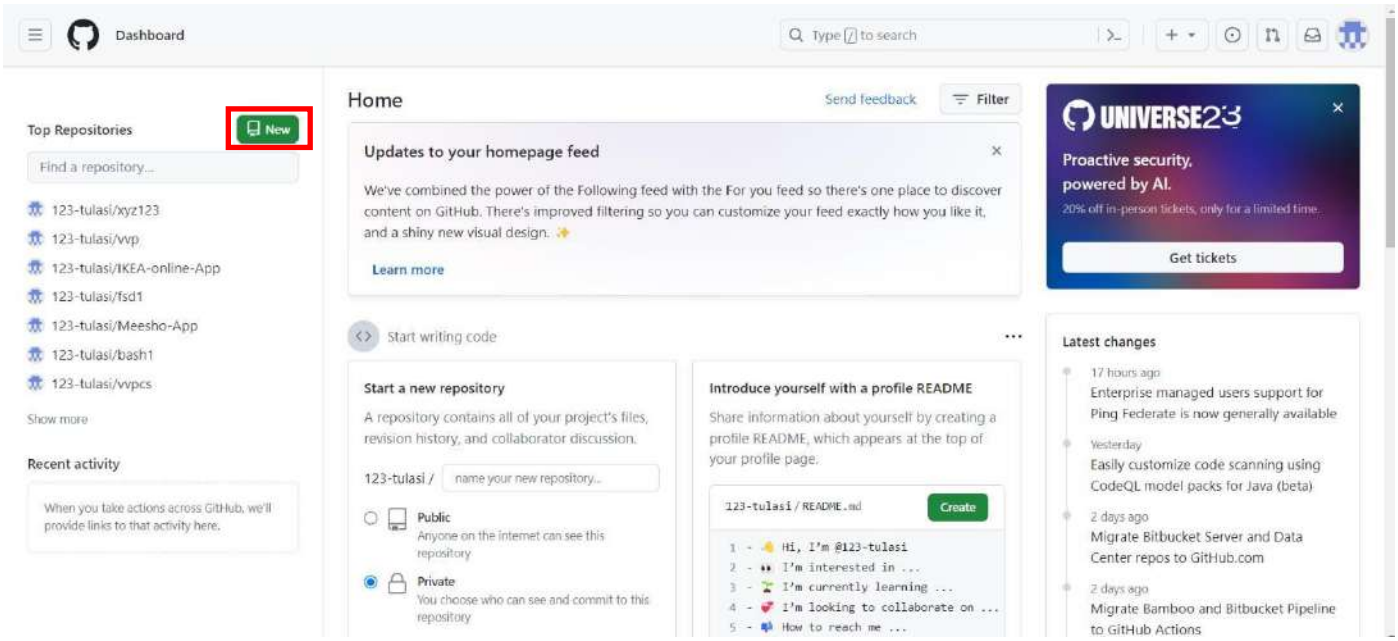
- ## Working in GitHub

GitHub: GitHub, Inc. is a platform and cloud-based service for software development and version controlusing Git, allowing developers to store and manage their code.

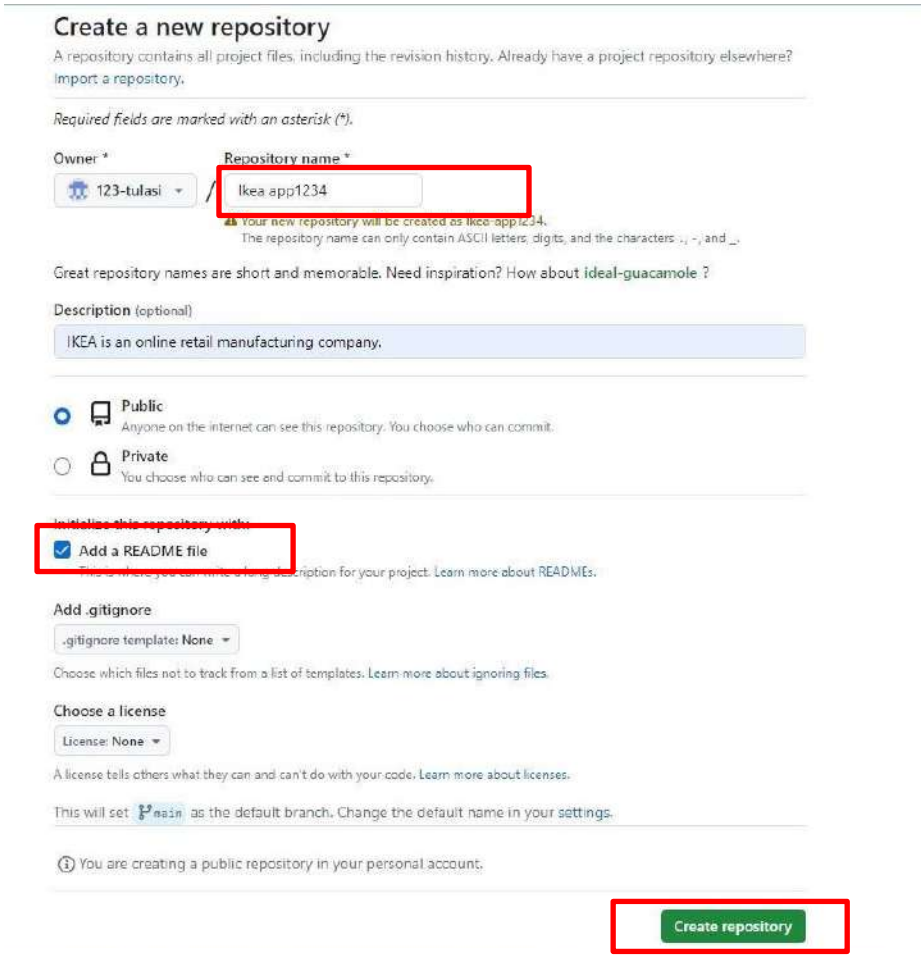
Create Git (Similar Tool) account and configure the repository.

Step 1: Go to web browser & search for **'GitHub'**. Then sign into your GitHub account. Now click on

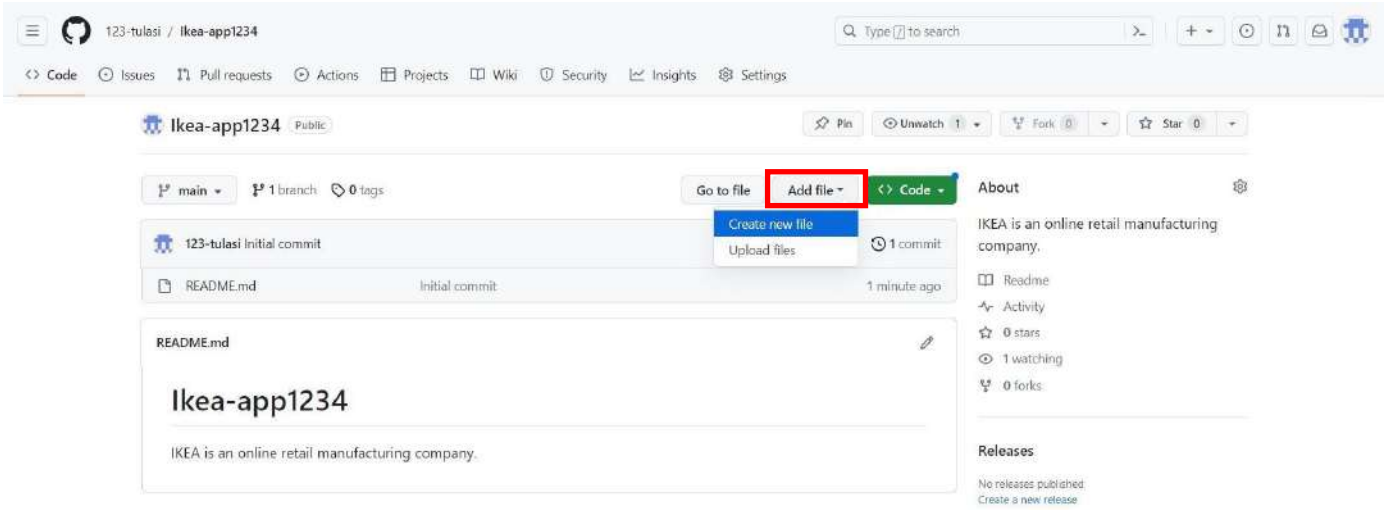
'New' to create a new repository.



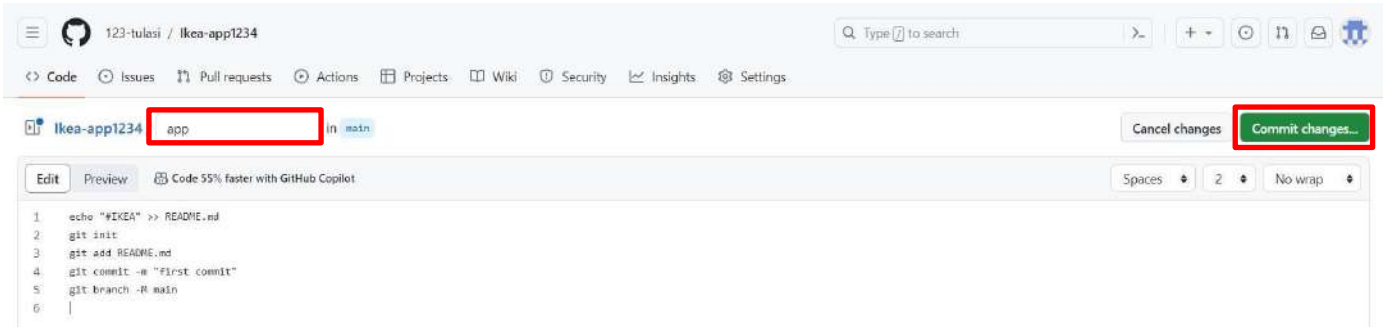
Step 2: Give a name to your repository & some description. Then enable **'Add a README file'** and clickon **'Create repository'**.



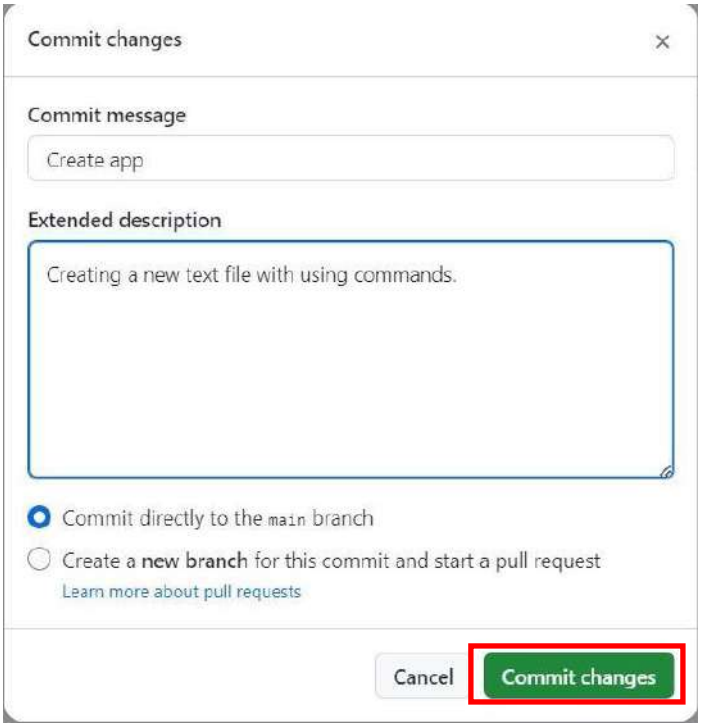
Step 3: Click on **'Add file'** and select **'Create new file'** to create a new file inside the repository.



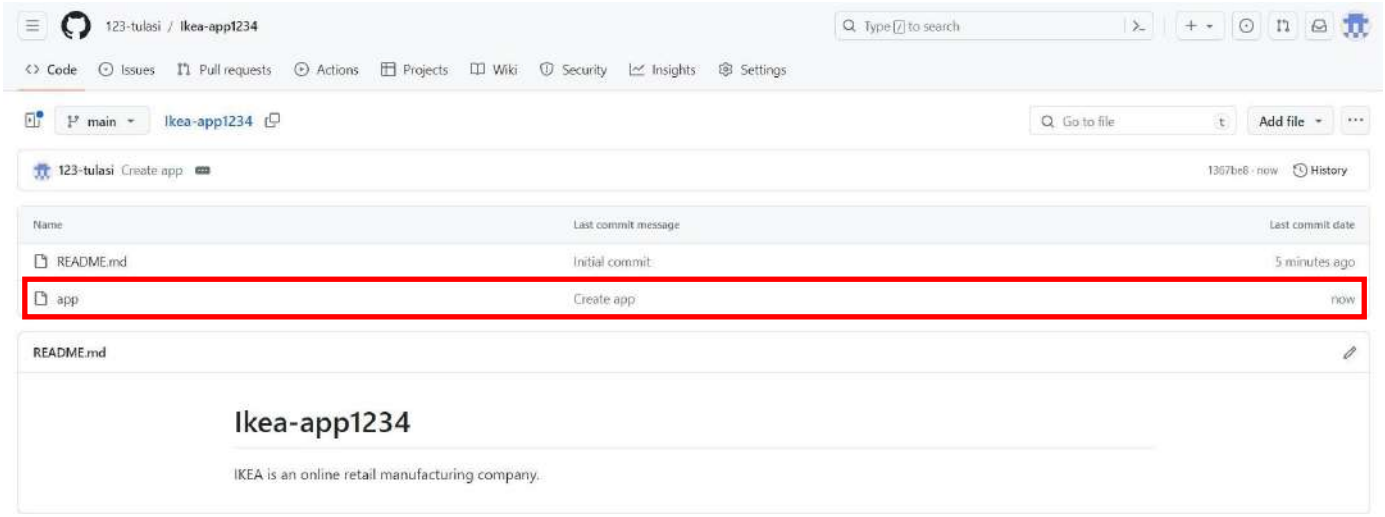
Step 4: Give a name to your new file and enter the below code and then click on **'Commit changes'**.



Step 5: Add the extended description and click on **'Commit changes'**.

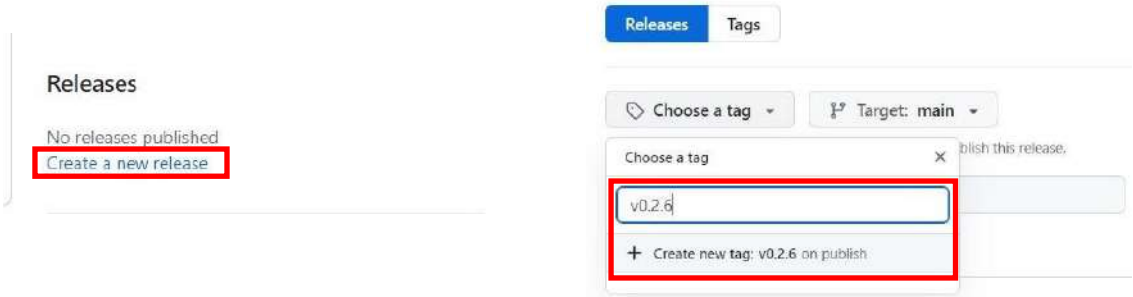


Step 6: Now you can view the newly created file.

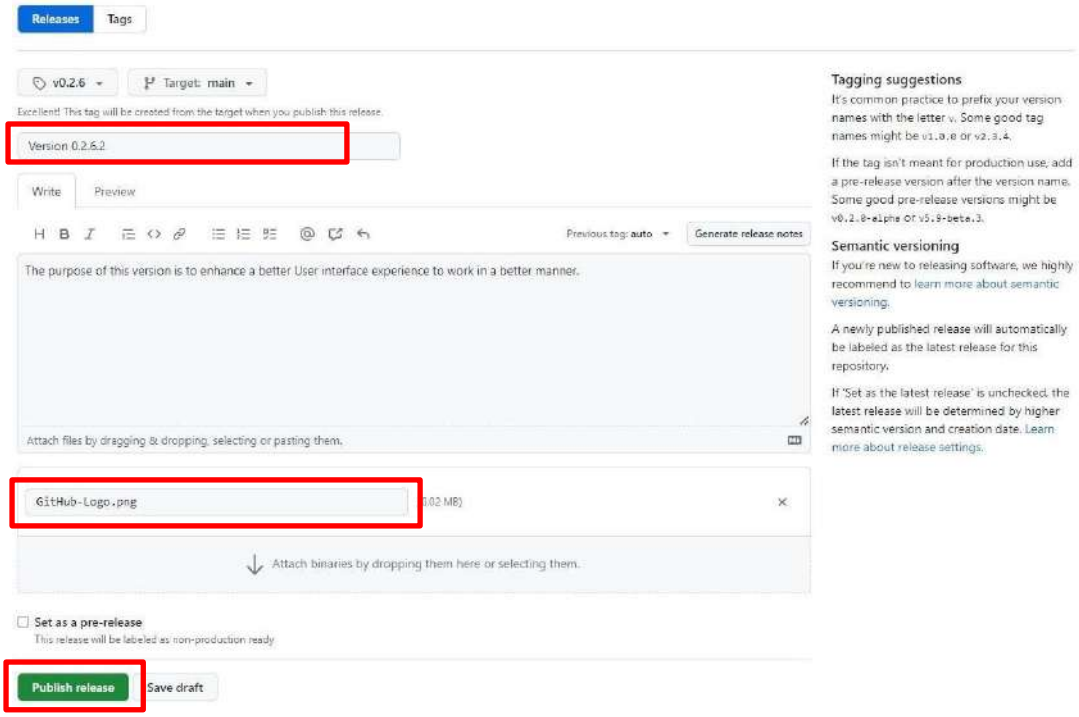


- **<u>Using Version control system.</u>**

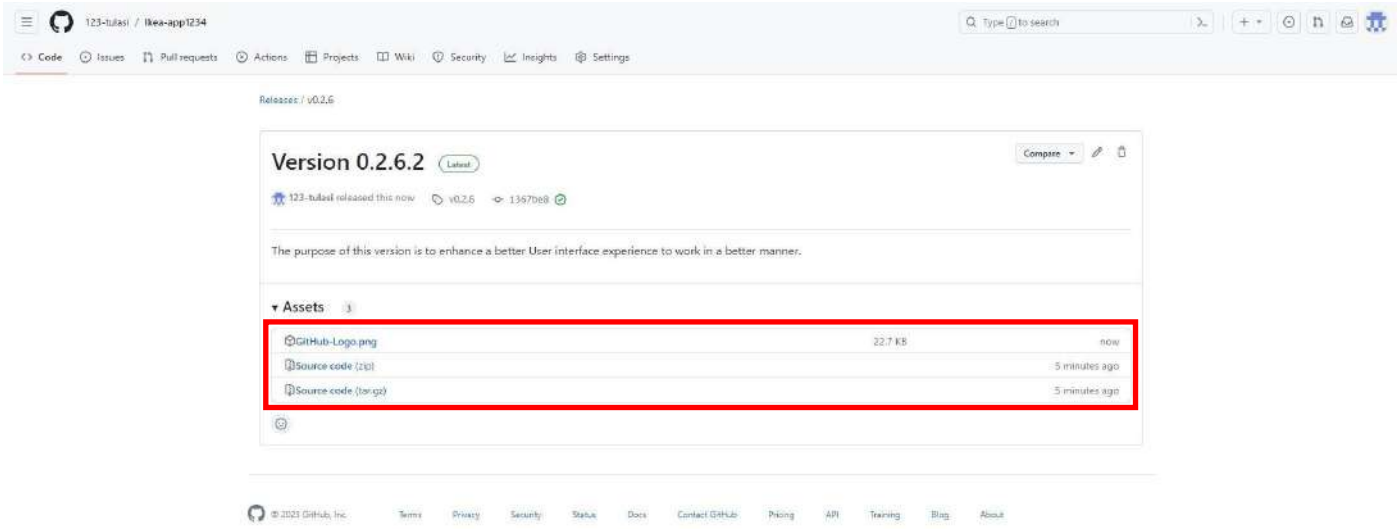
Step 7: Now click on ‘Create a new release’ in home page & choose a tag to your new release creation.



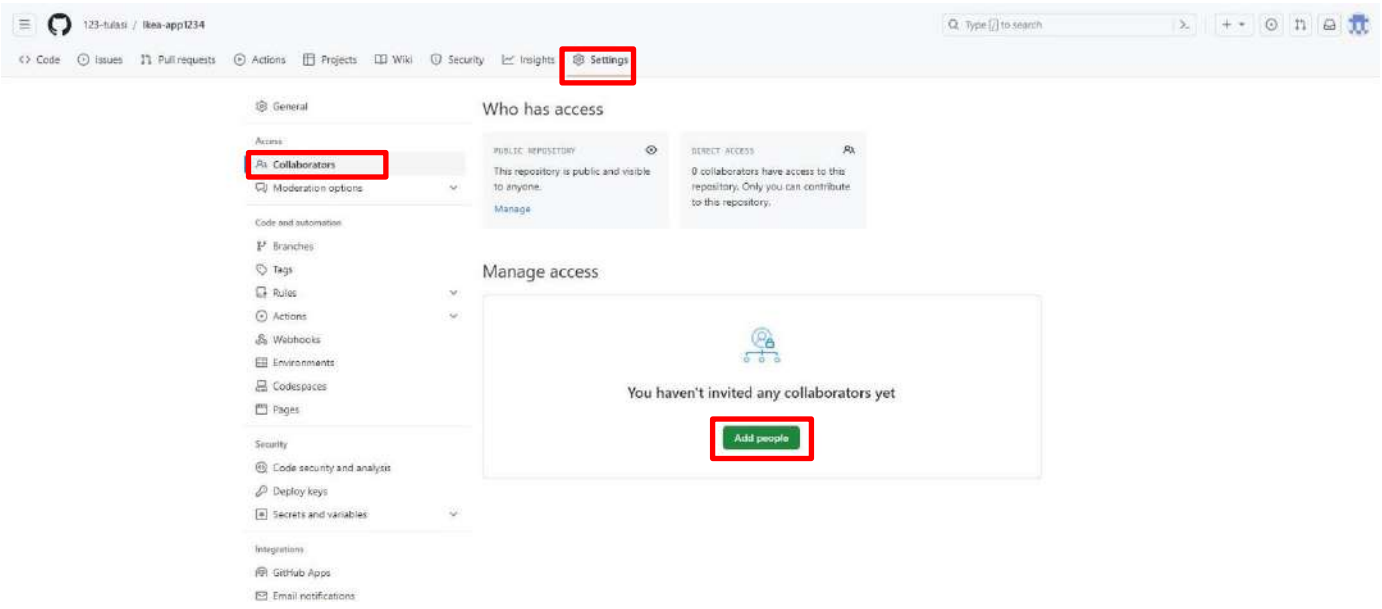
Step 8: Give a name to your **version** and enter some description for your new release and also attach somerelated files to it. Then press on **‘Publish release’**.



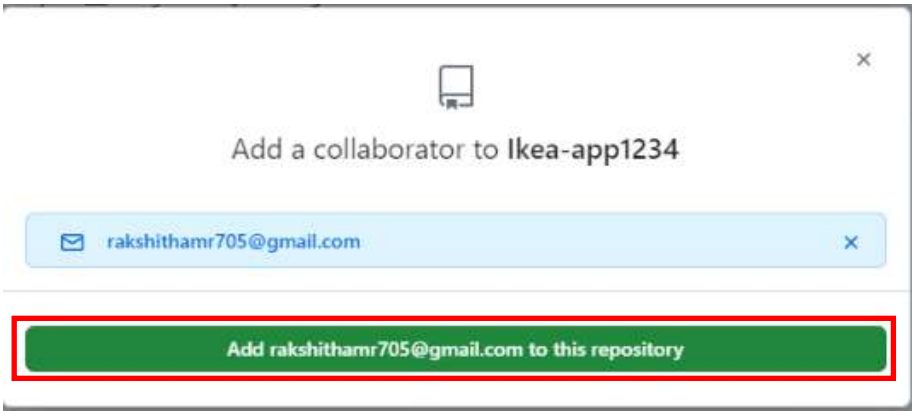
Step 9: Now you can clearly see the Version released & the attached files here.



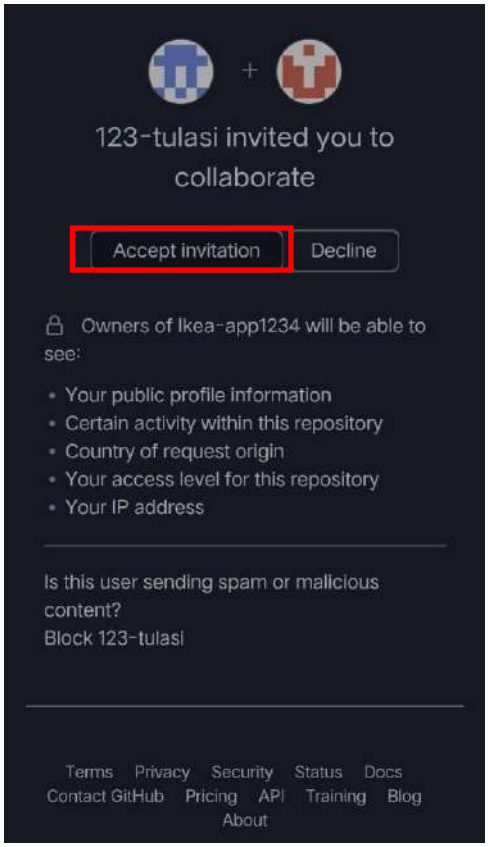
Step 10: Go to Settings Collaborators Add people.



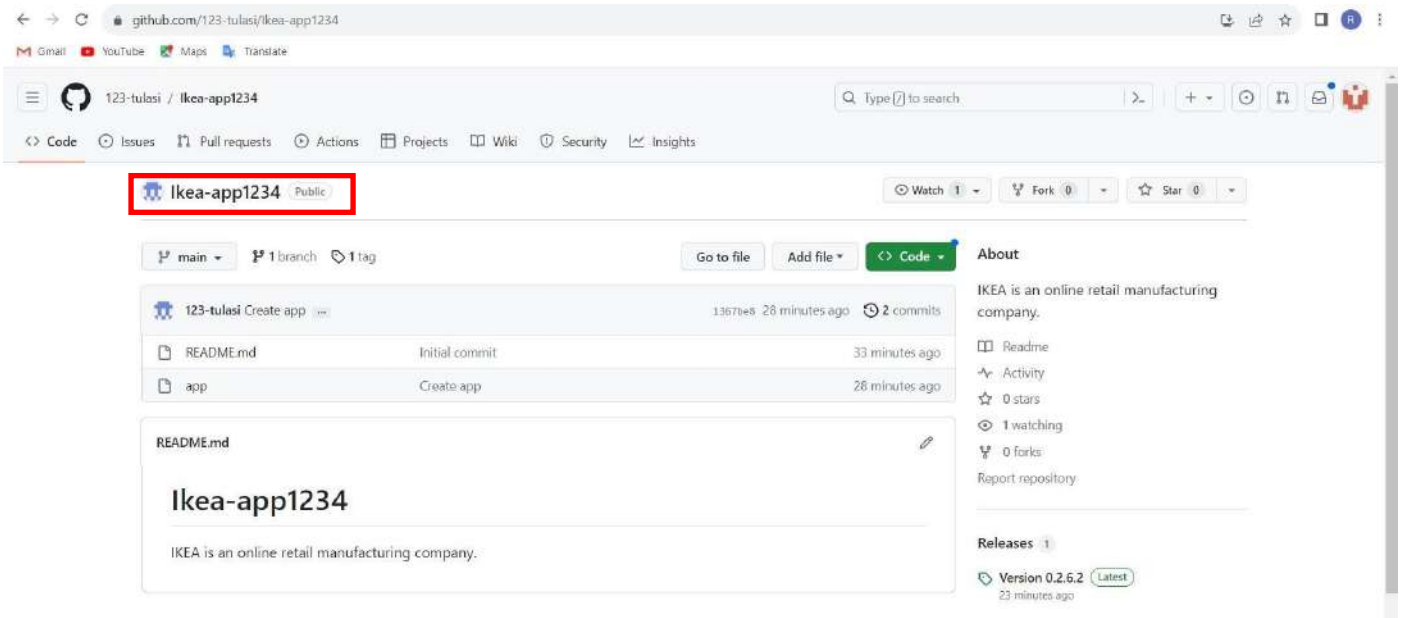
Step 11: Invite another GitHub user to collaborate with your project.



Step 12: Now the collaborator has to accept the invitation.



Step 13: Now the collaborate with the same project which is displayed here.



2. Create a Cloud account (AWS, GCB or any other service provider) and explore thefeatures.

Amazon Web Service (AWS)

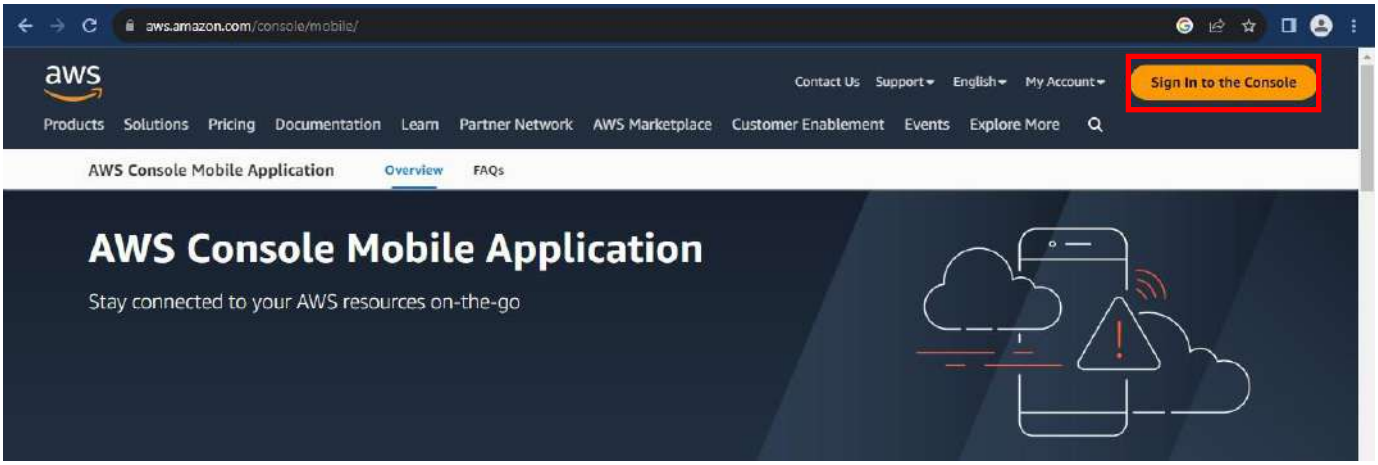
Amazon Web Services, Inc. (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay-as-you-go basis. Clients will often use this in combination with autoscaling (a process that allows a client to use more computing

in times of high application usage, and then scale down to reduce costs when there is lesstraffic).

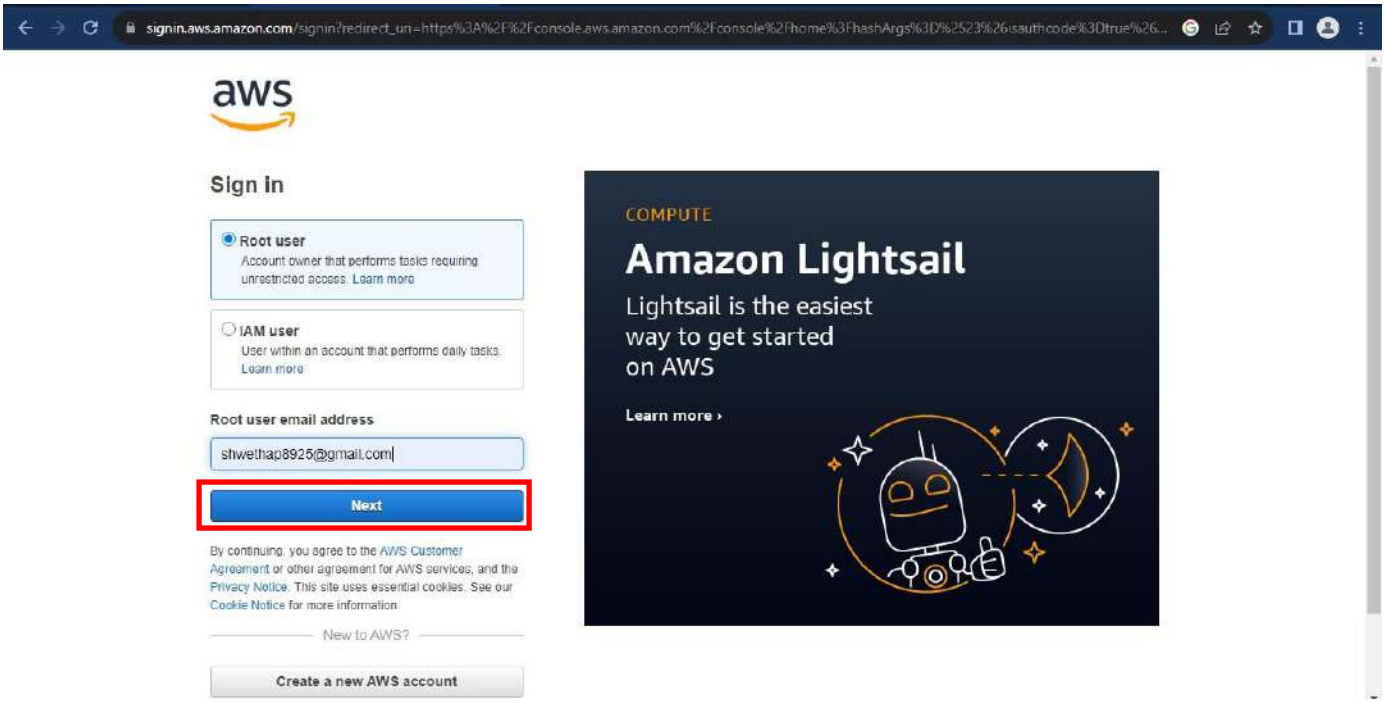
- **<u>Create and setup a virtual machine</u>.**

Step 1: Go to web browser and search for AWS console and click on first website provided and create a newAWS account by giving all the credentials required.

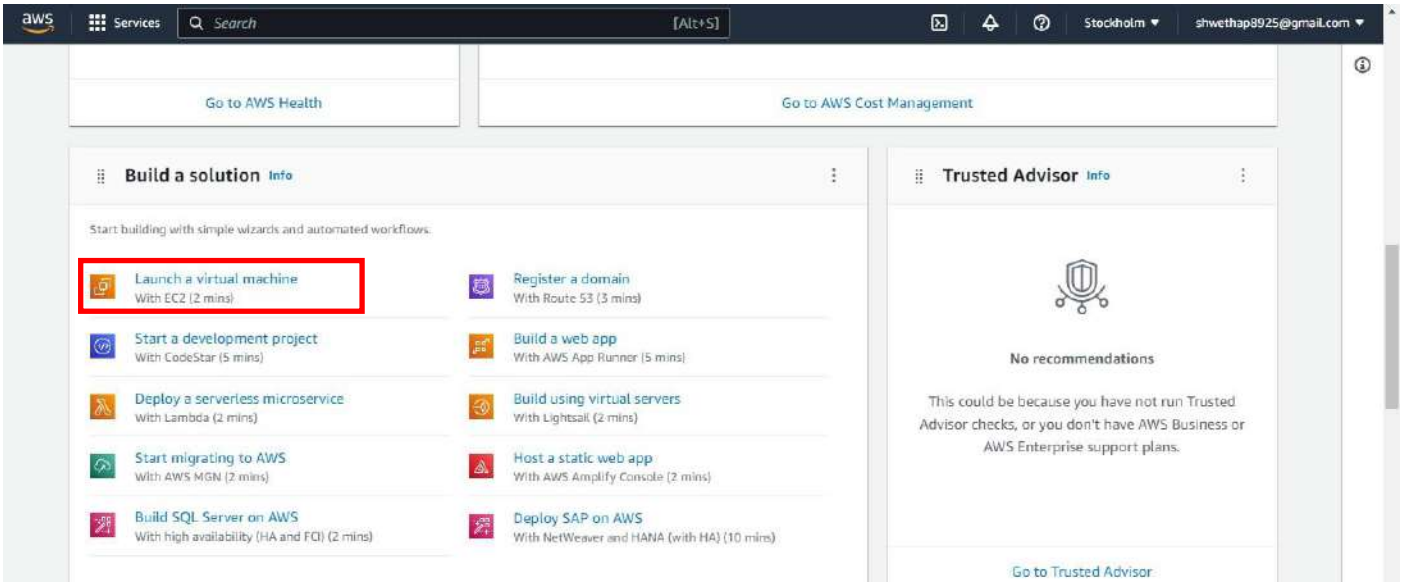
Step 2: After creating an account, click on ‘Sign in to the console’ option.



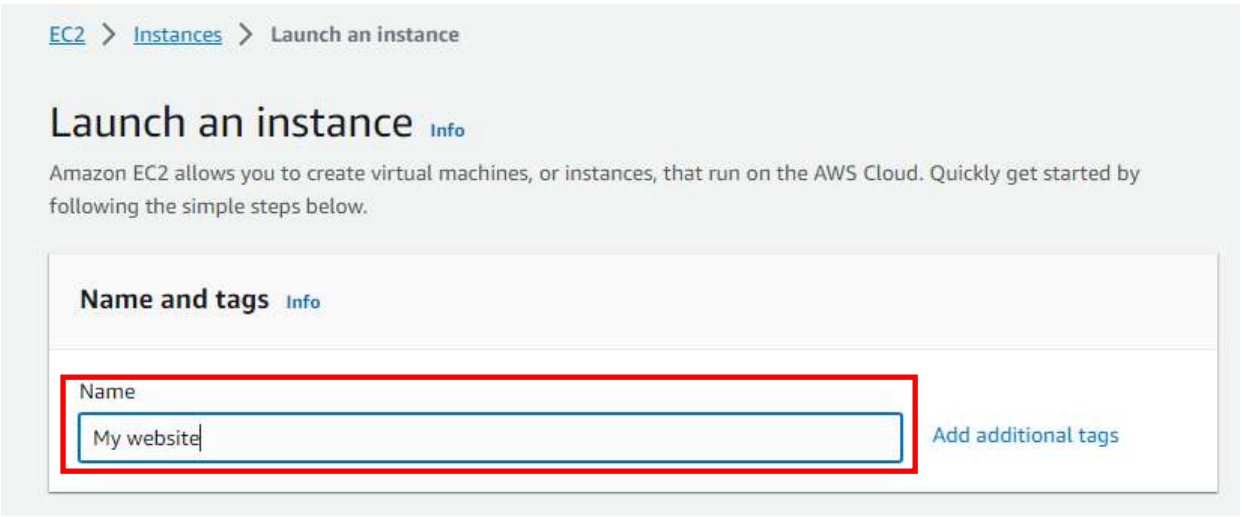
Step 3: Now select the ‘Root User’ and enter the ‘Gmail-Id’ & ‘password’ which is registered with theaccount details and then click on ‘Next’.



Step 4: In the AWS home tab, scroll down and click on ‘Launch a virtual machine’.

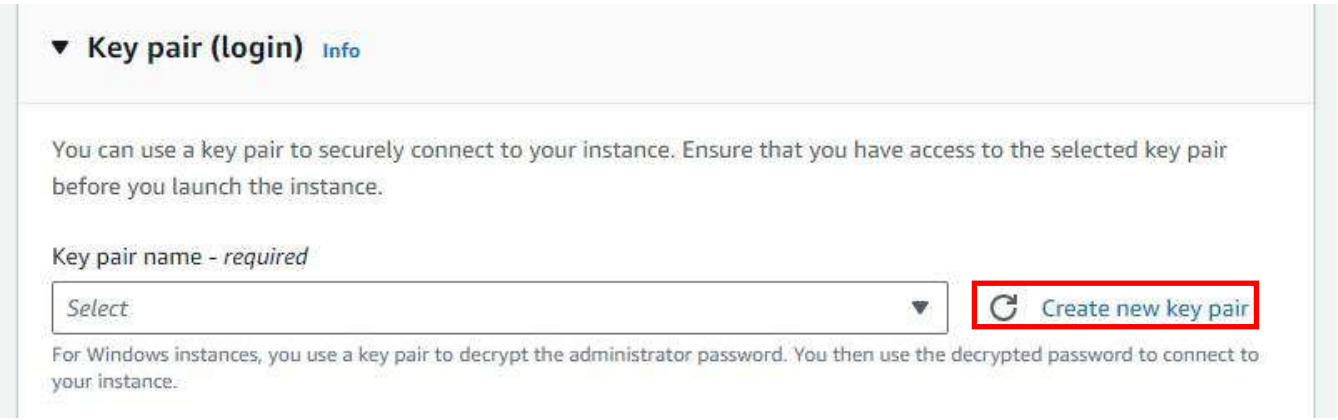


Step 5: Give any name for the tag.

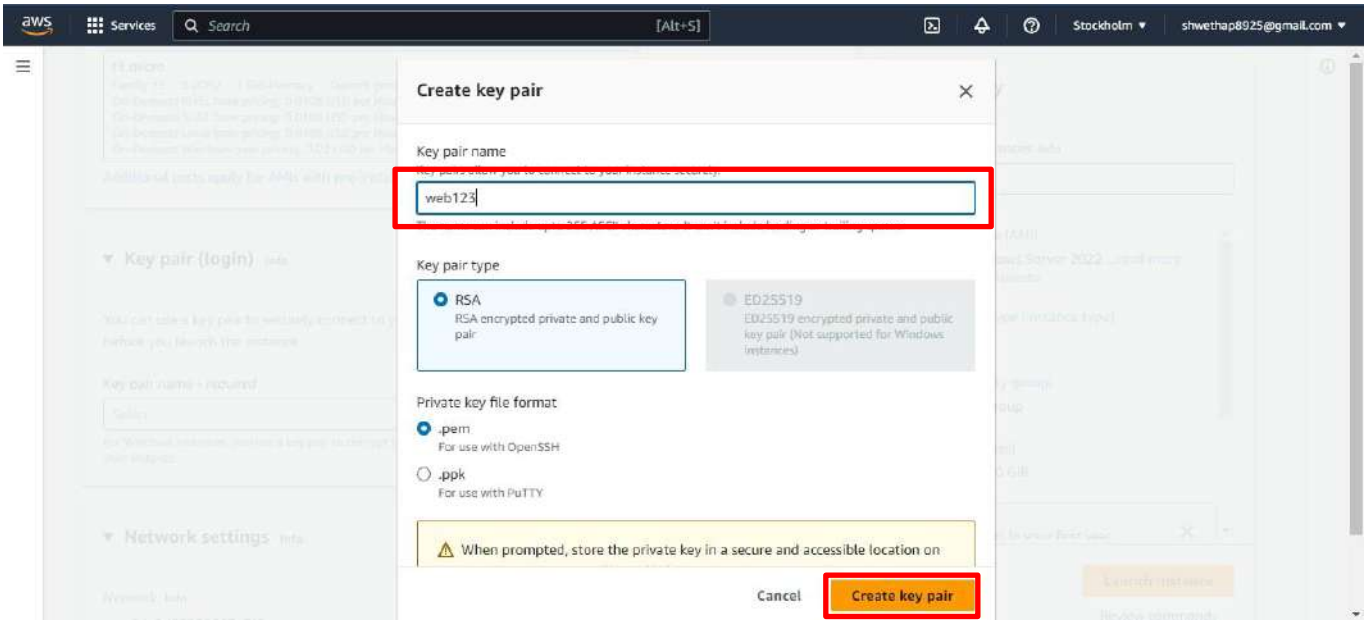


Step 6: Now select the OS as 'Windows'.

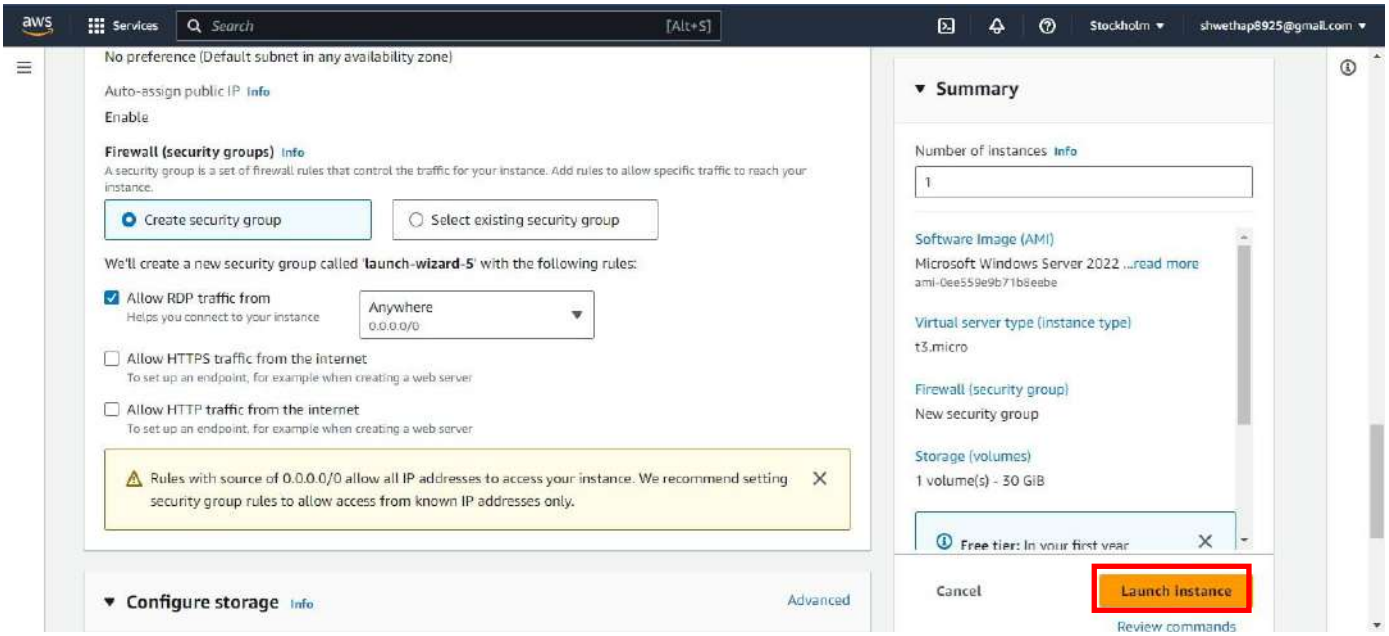
Step 7: Then click on 'Create a New key pair'.



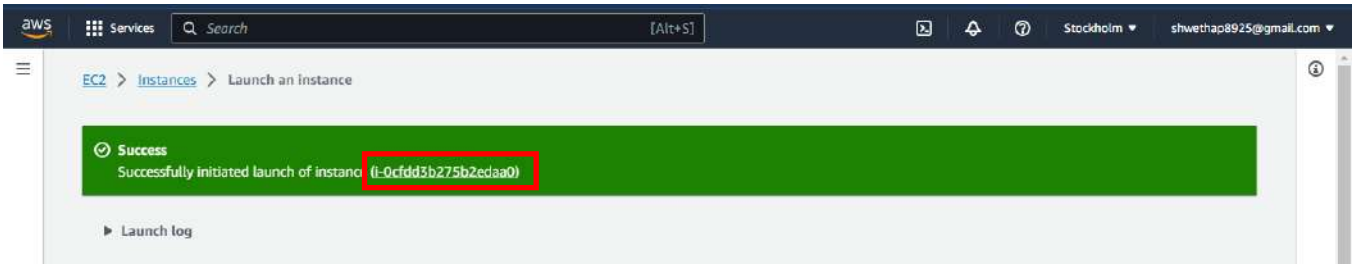
Step 8: Give a name to the key pair & click on 'Create Key pair'.



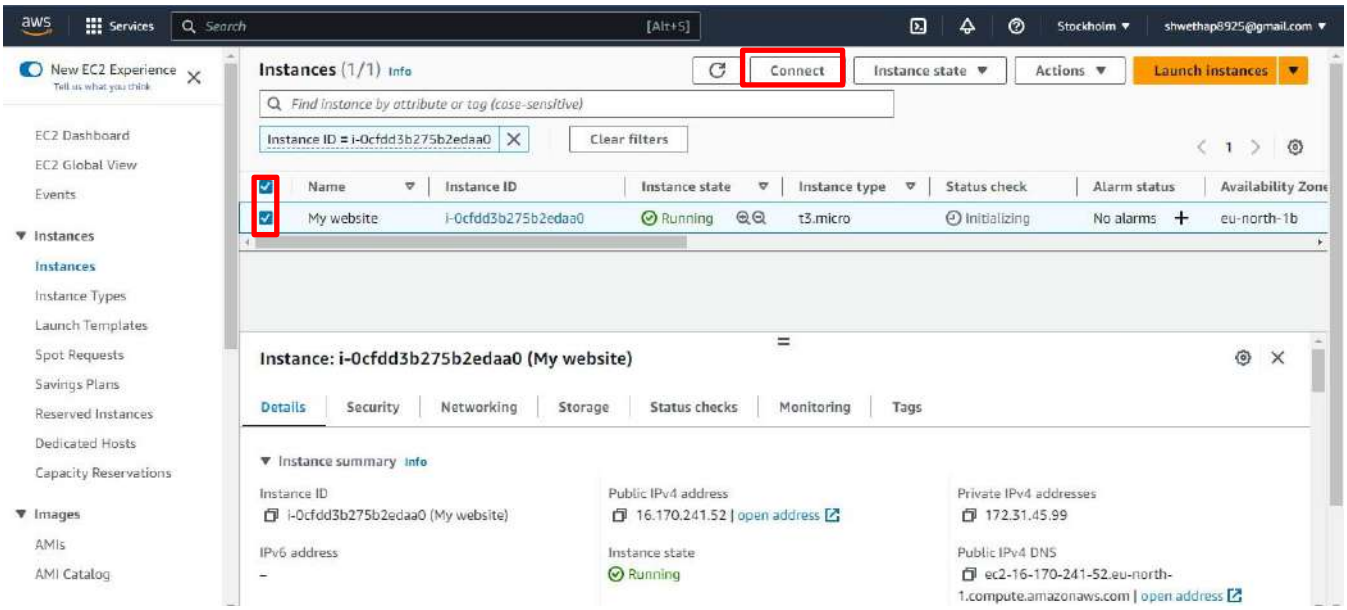
Step 9: Now click on 'Launch instance'.



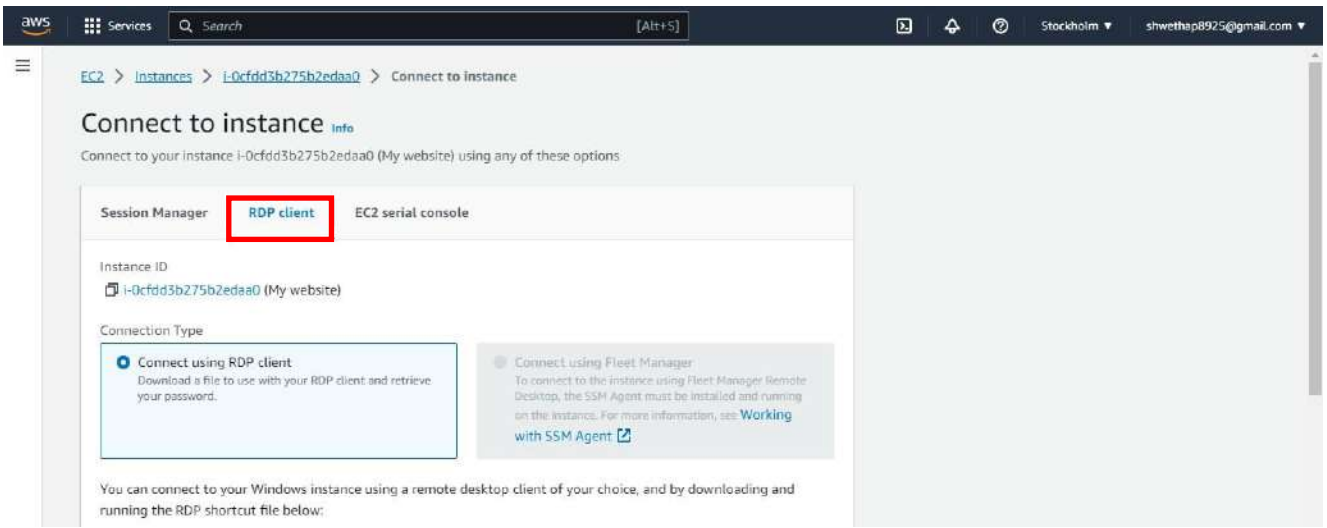
Step 10: Click on the below link after the successful installation of the Launch instance.



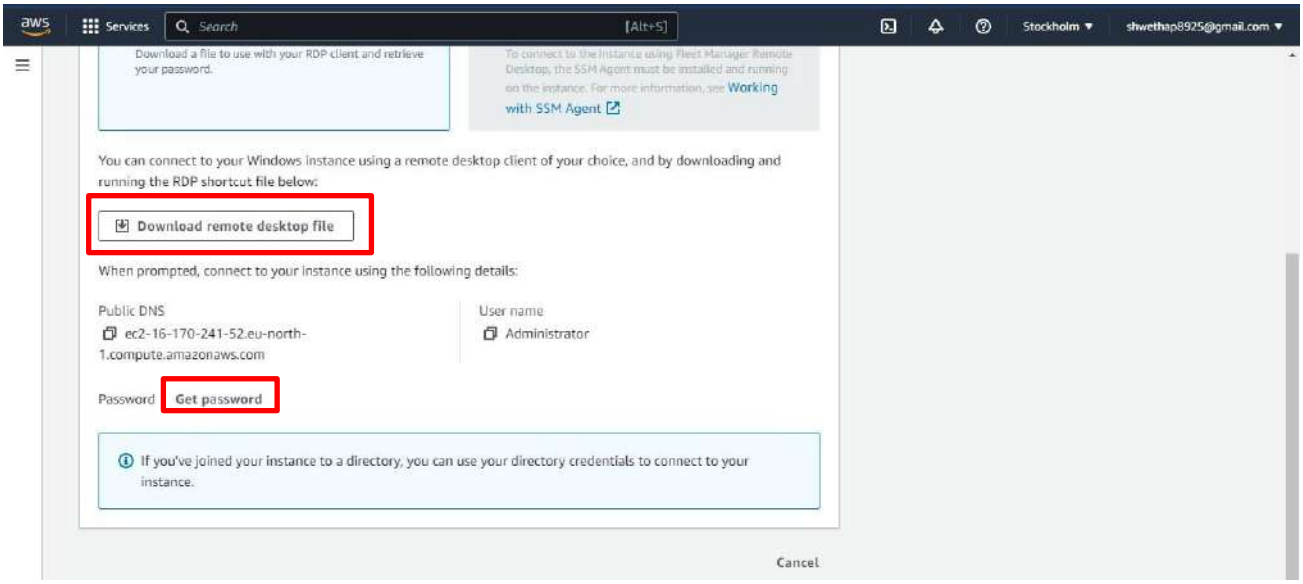
Step 11: Select the key pair created and click on ‘Connect’.



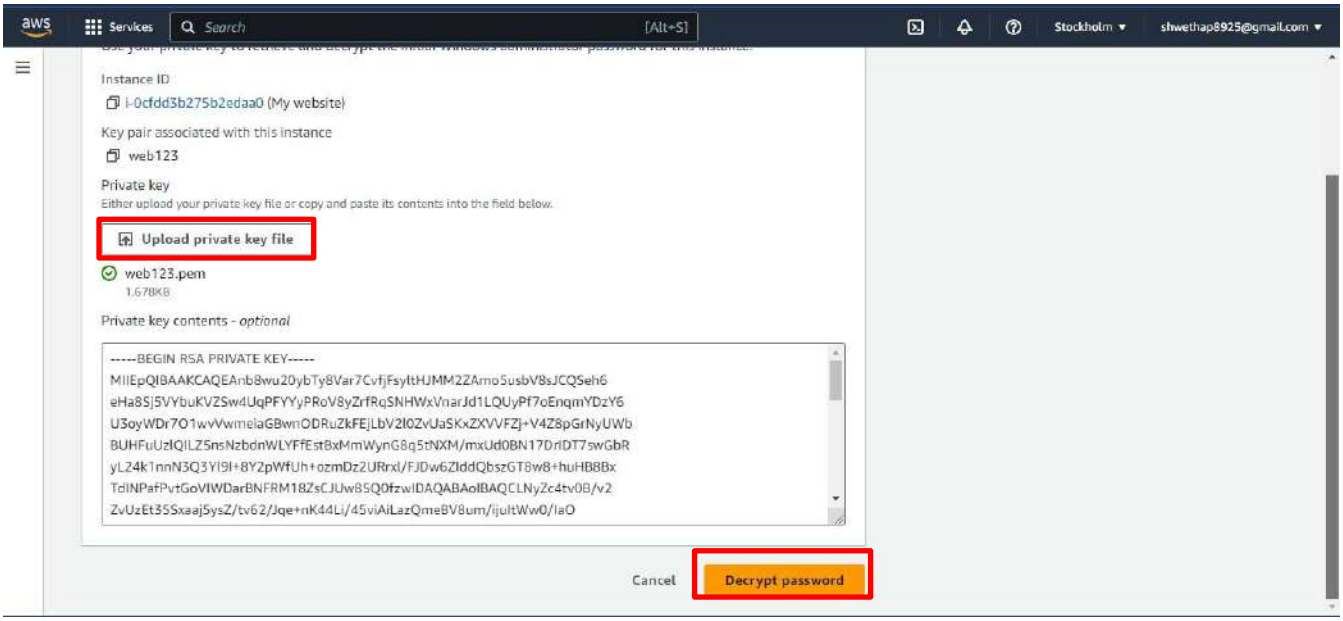
Step 12: Click on ‘RDP Client’.



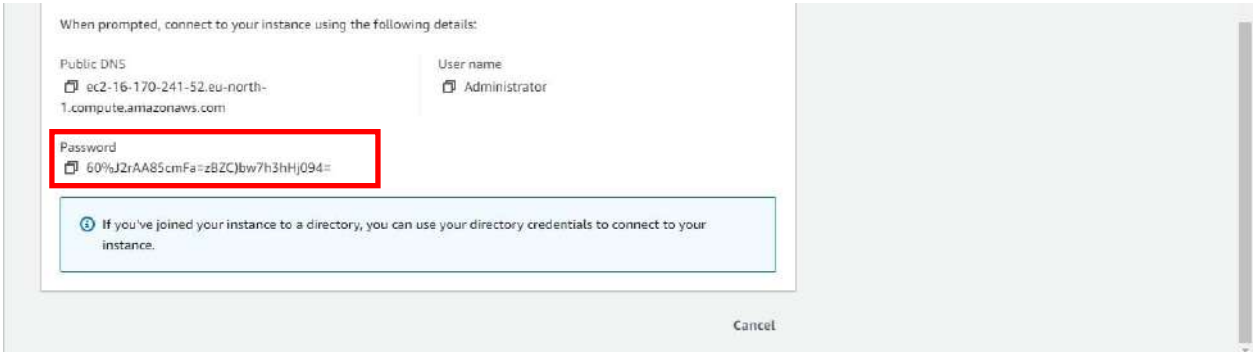
Step 13: Then click on ‘Download remote desktop file’ and click on ‘Get password’.



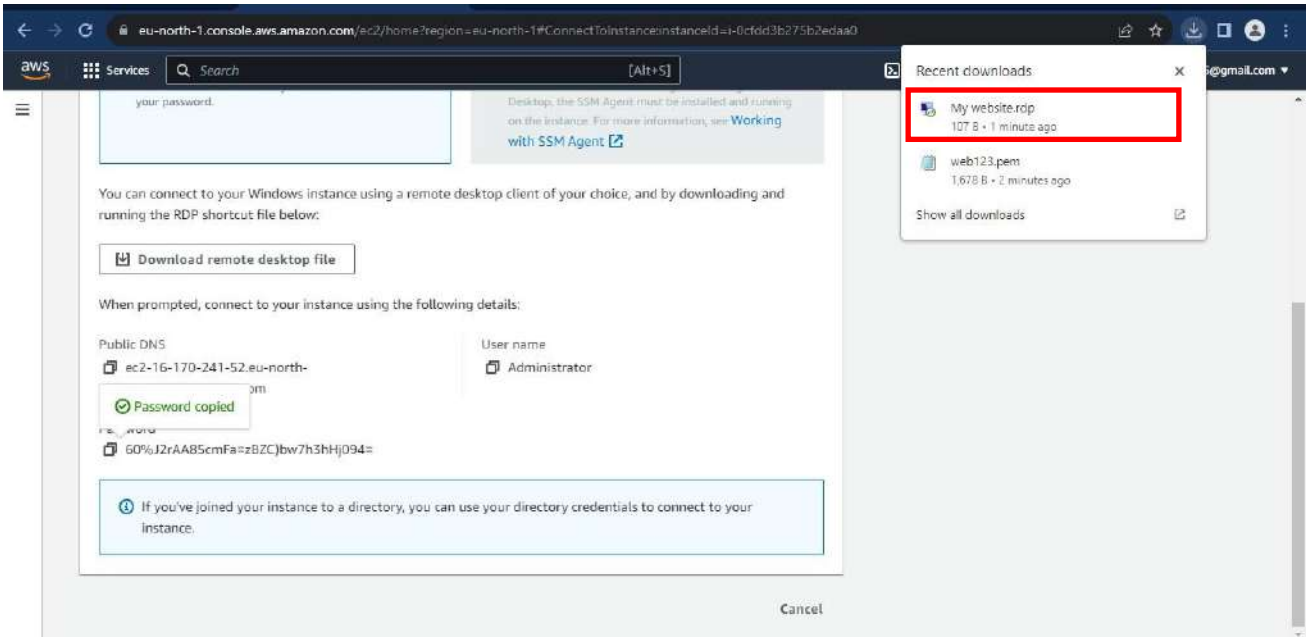
Step 14: Now click on 'Upload Private Key File' and open the key pair which was already downloaded.And click on 'Decrypt password'.



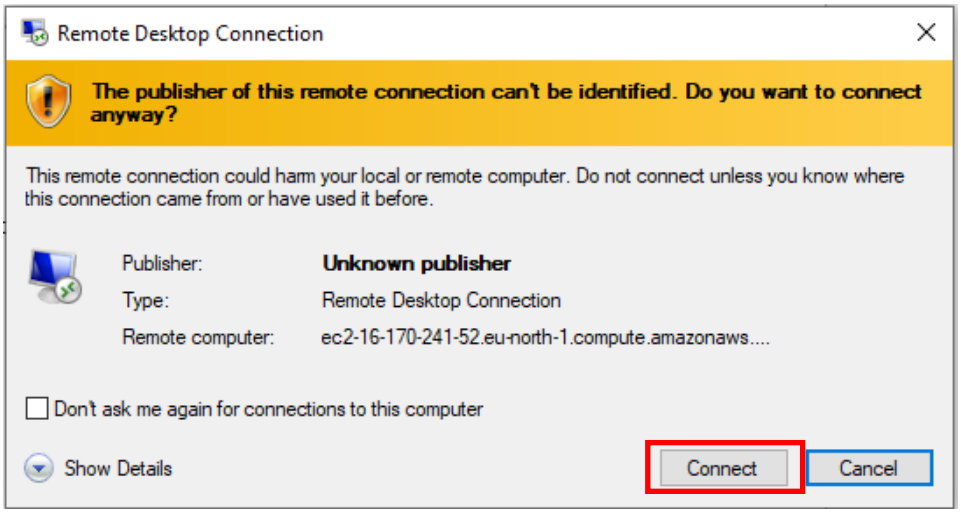
Step 15: Now copy the below password generated.



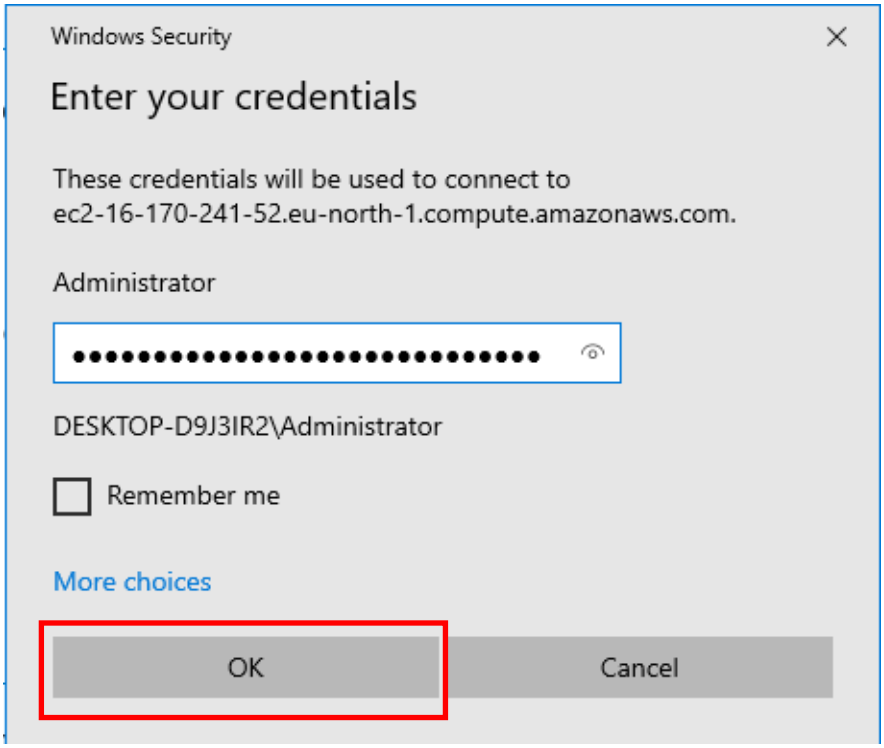
Step 16: Now open the Remote desktop file which was already downloaded.



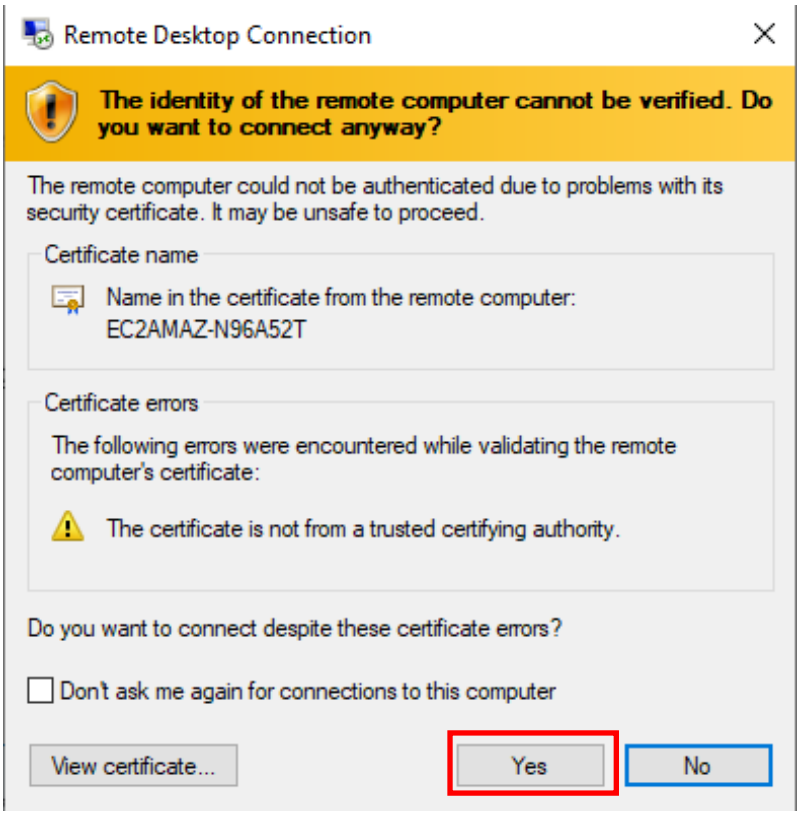
Step 17: Click on ‘Connect’.



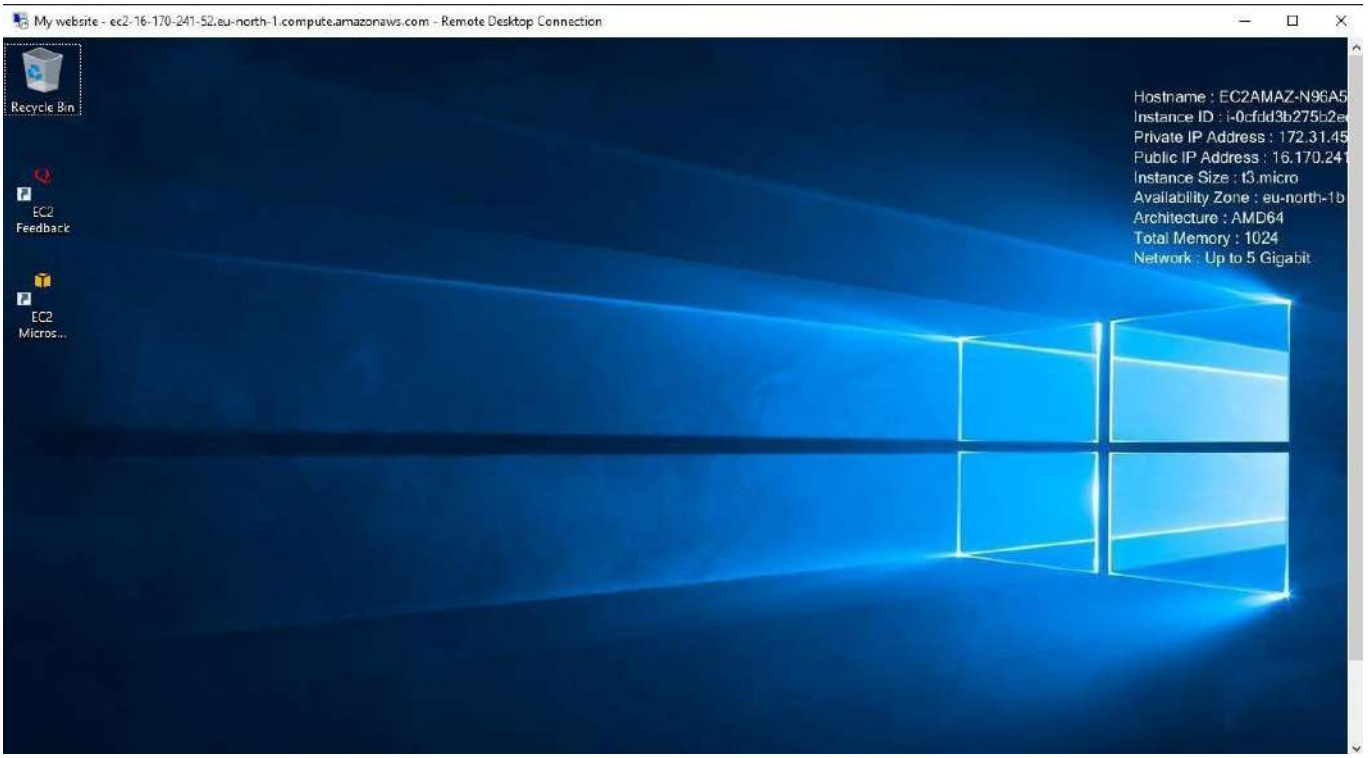
Step 18: Now paste the password which was copied in step15 and click on ‘OK’.



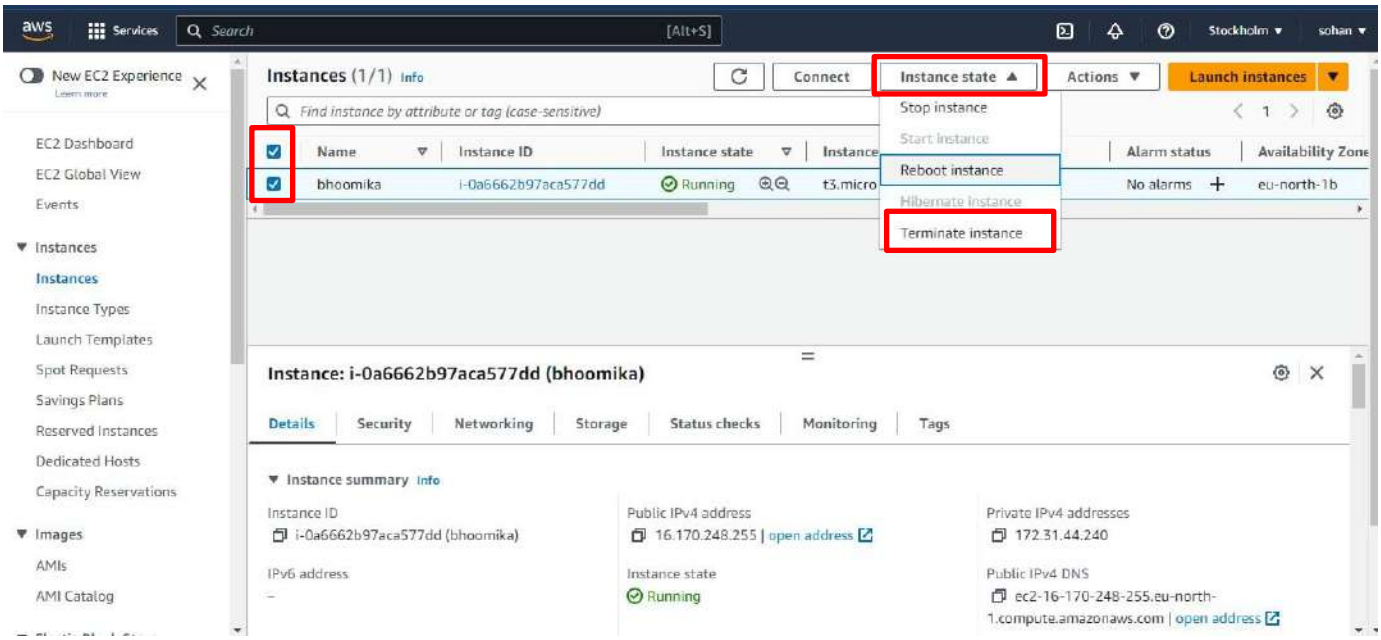
Step 19: Click on 'Yes'.

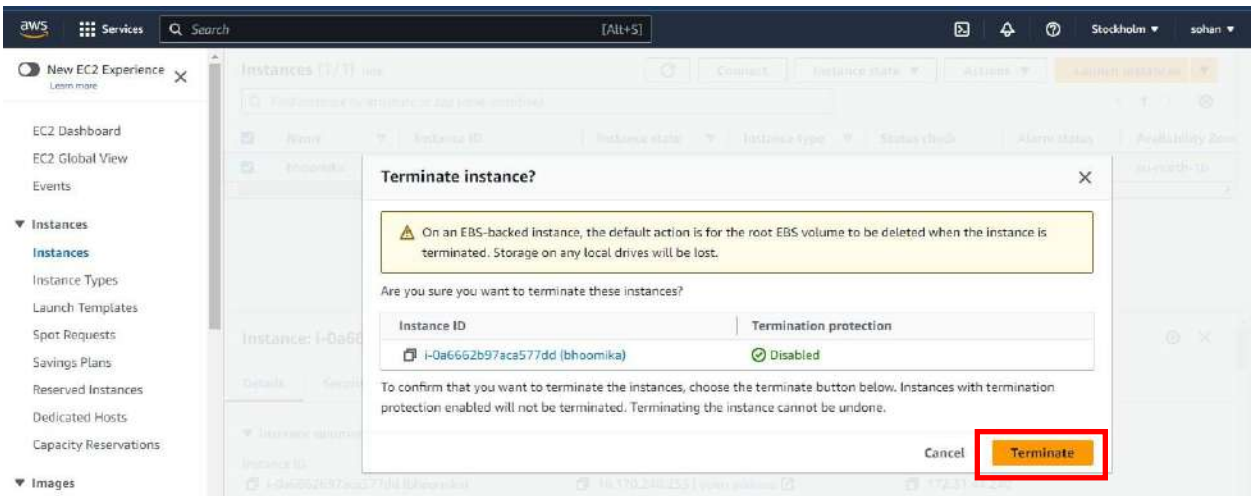


Step 20: Now the Virtual machine will be created successfully.



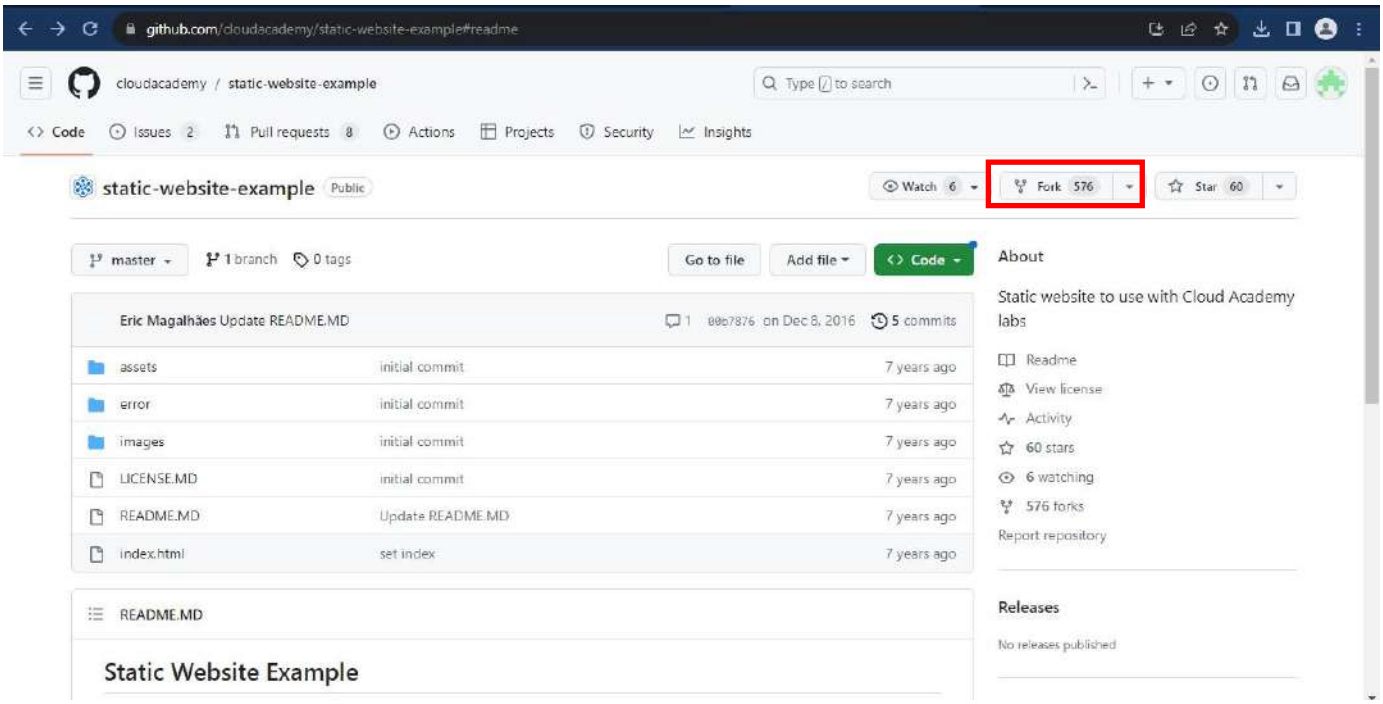
Note: Now terminate the instance created.



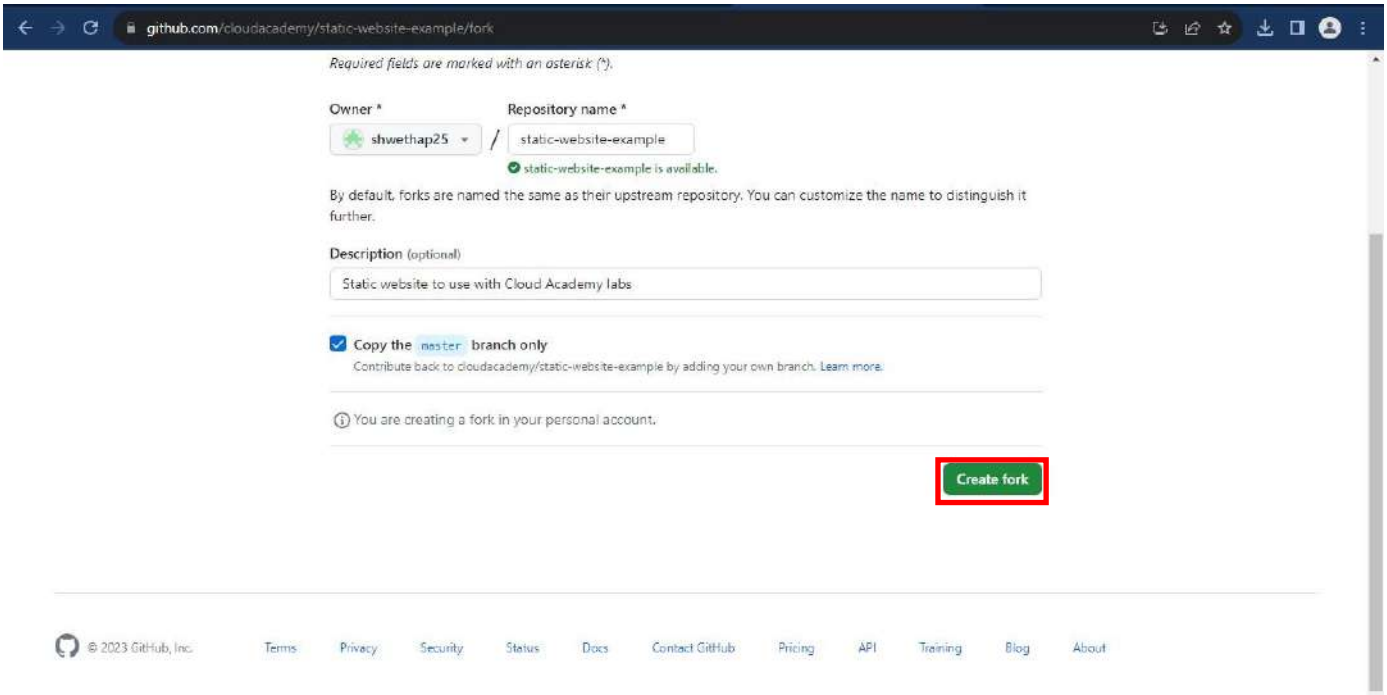


B. Create a simple Web application using Cloud Services.

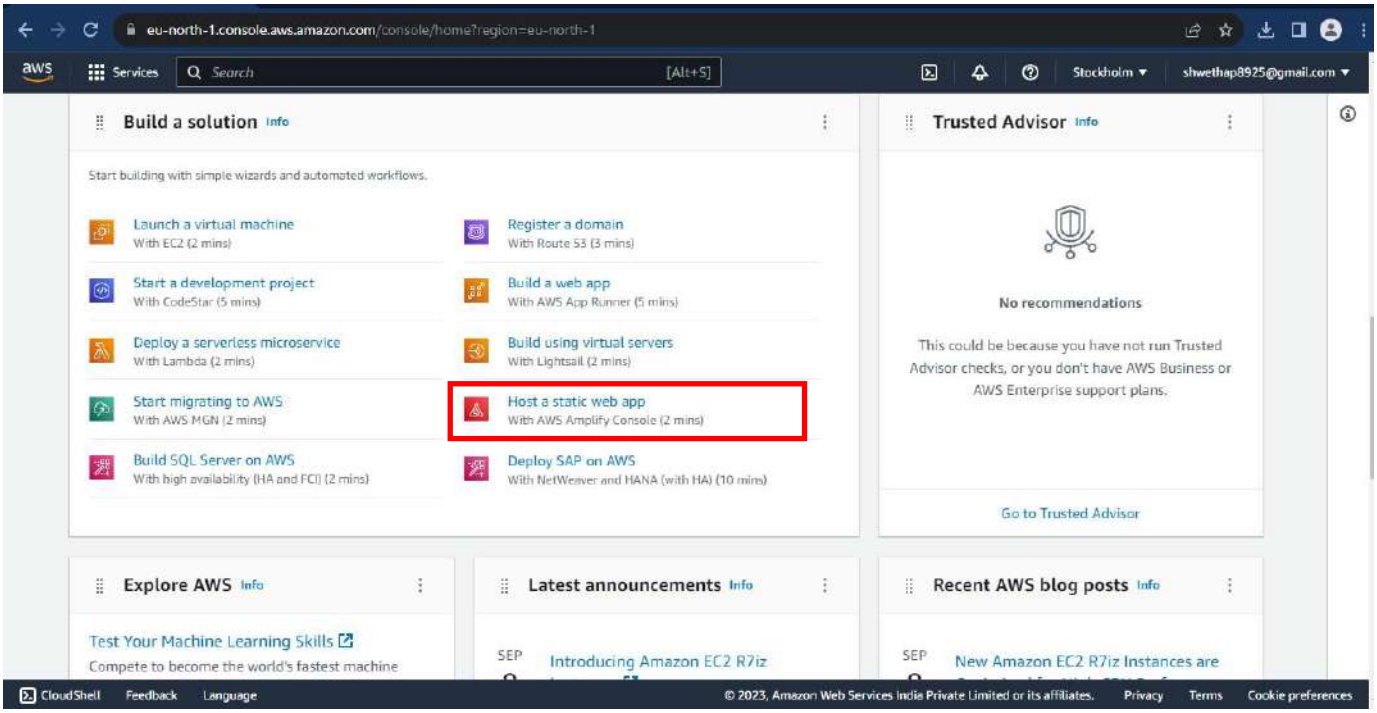
Step 21: Now sign in to the GitHub account which was registered to AWS account. Then click on this linkhttps://github.com/cloudacademy/static-website-example to create a new Fork file.



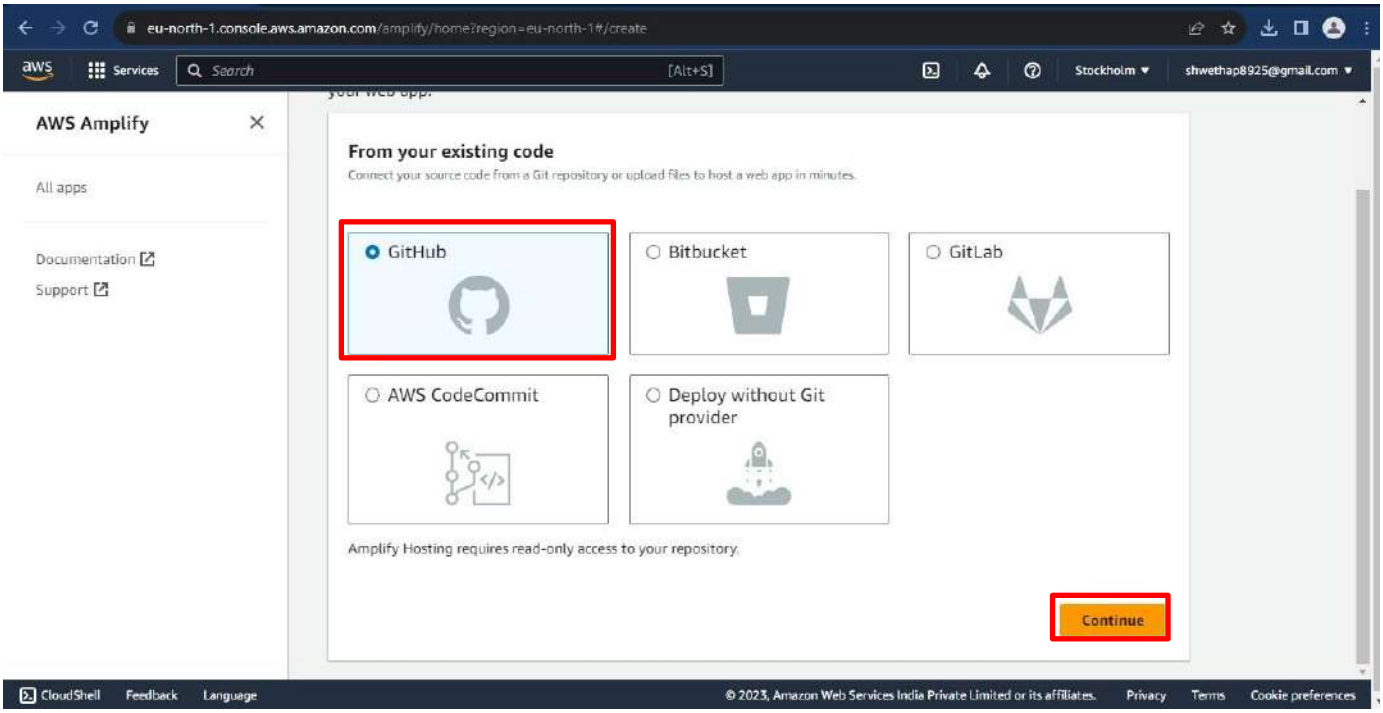
Step 22: Click on ‘Create Fork’.



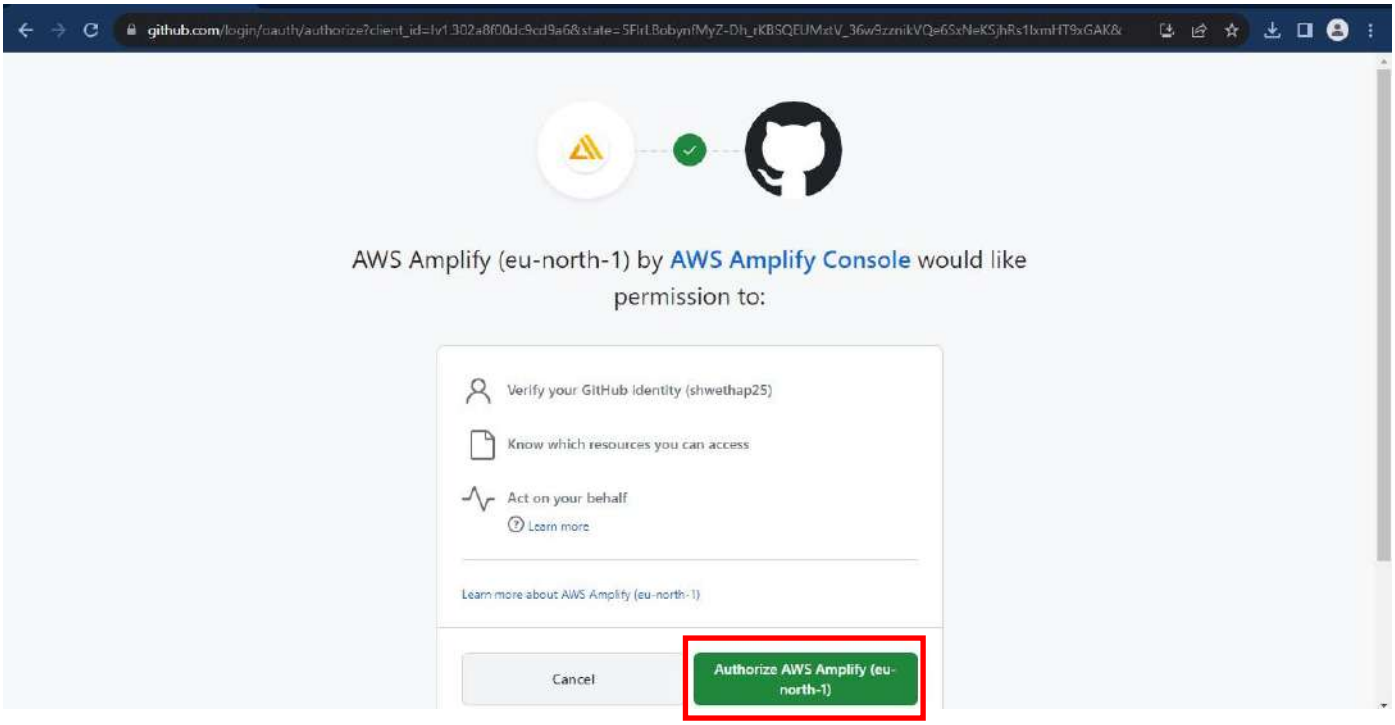
Step 23: Now go back to the AWS home screen and click on ‘Host a static web app’.



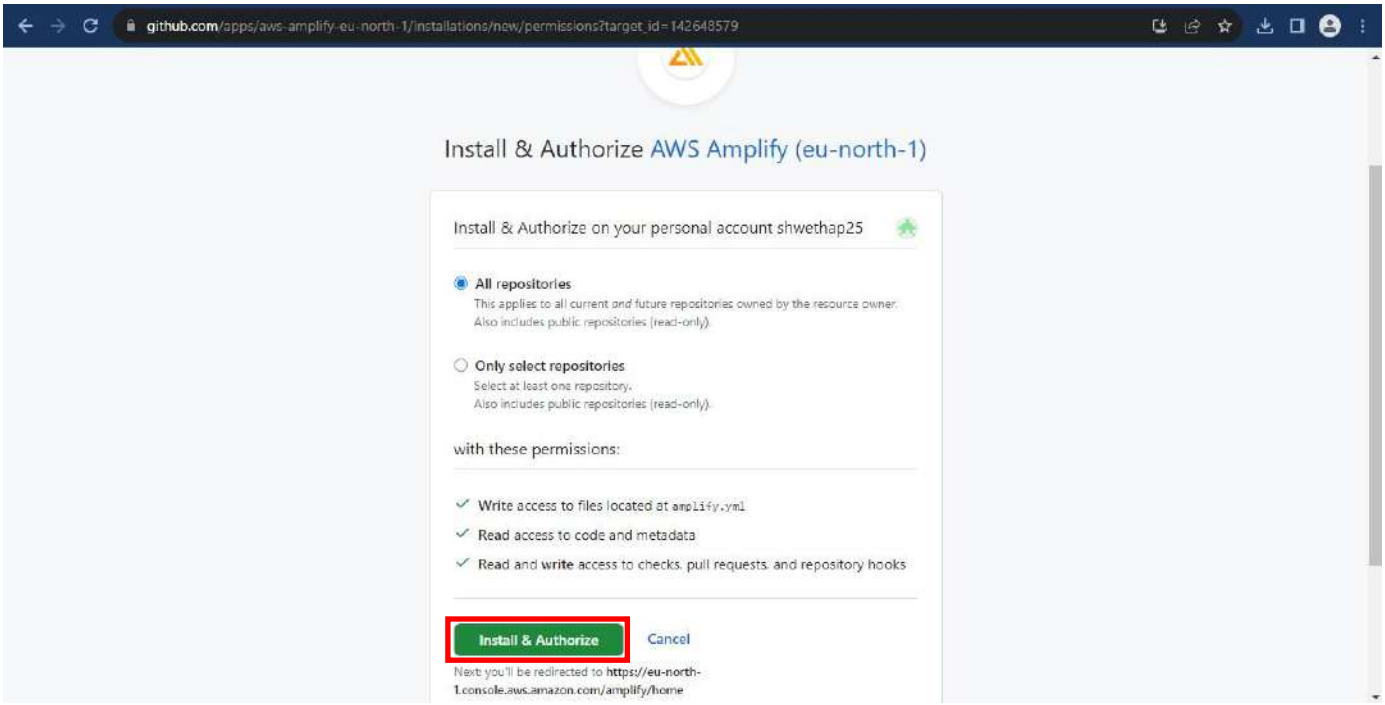
Step 24: Select ‘GitHub’ as the Host web app. And click on ‘Continue’.



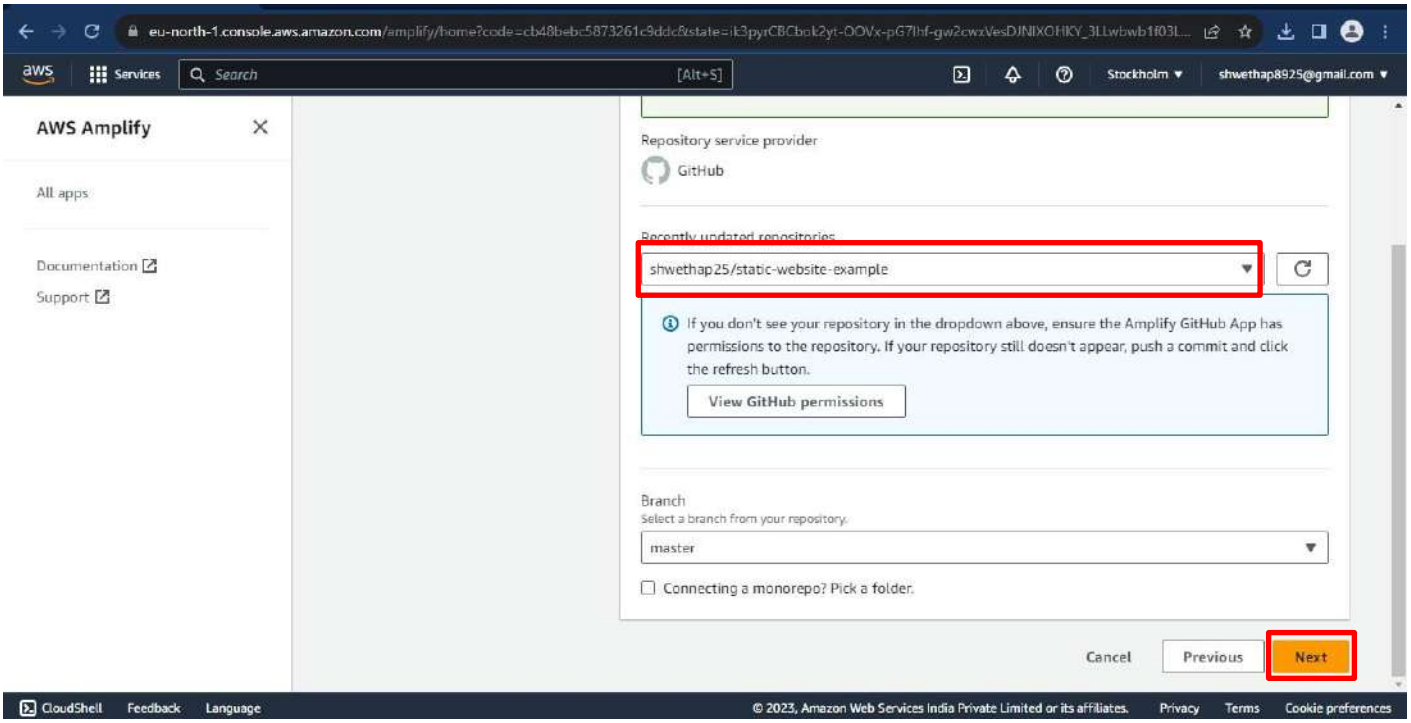
Step 25: Now click on ‘Authorize AWS Amplify (eu-north-1)’.



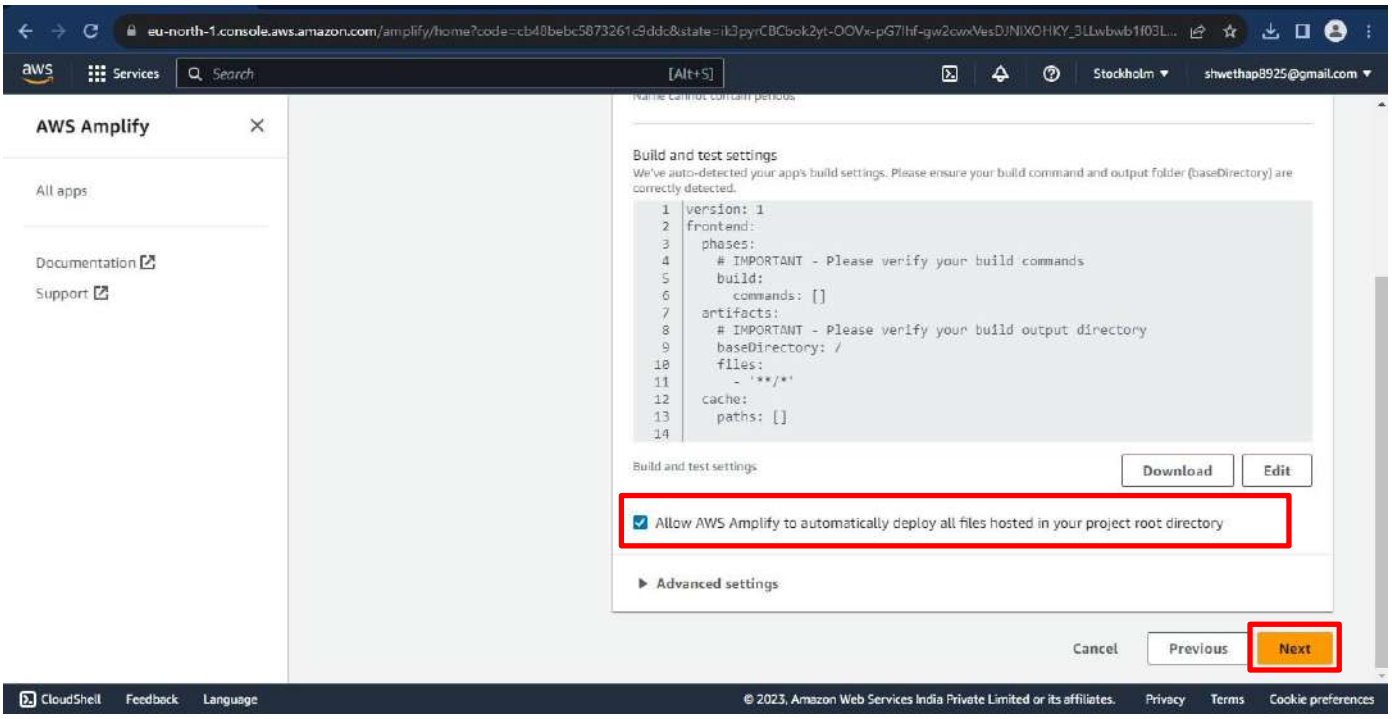
Step 26: Then click on 'Install & Authorize'.



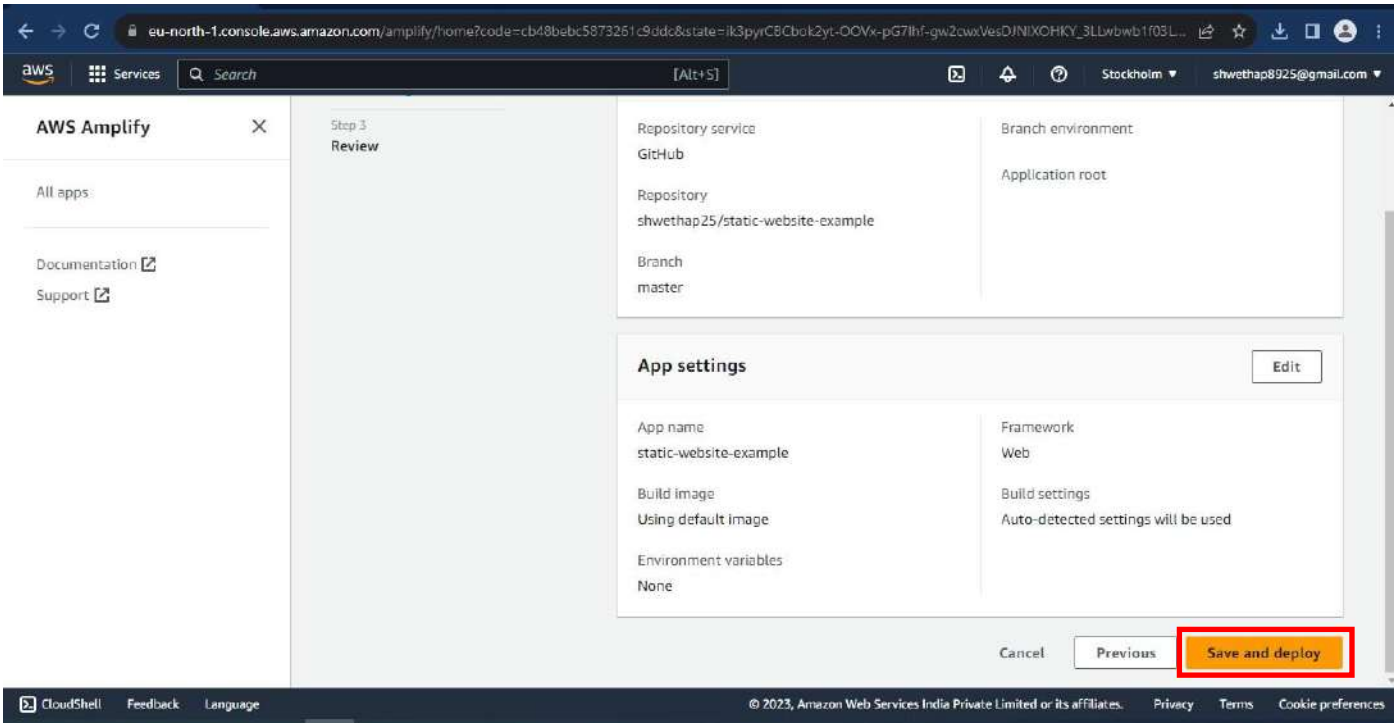
Step 27: Select the repository and click on 'Next'.



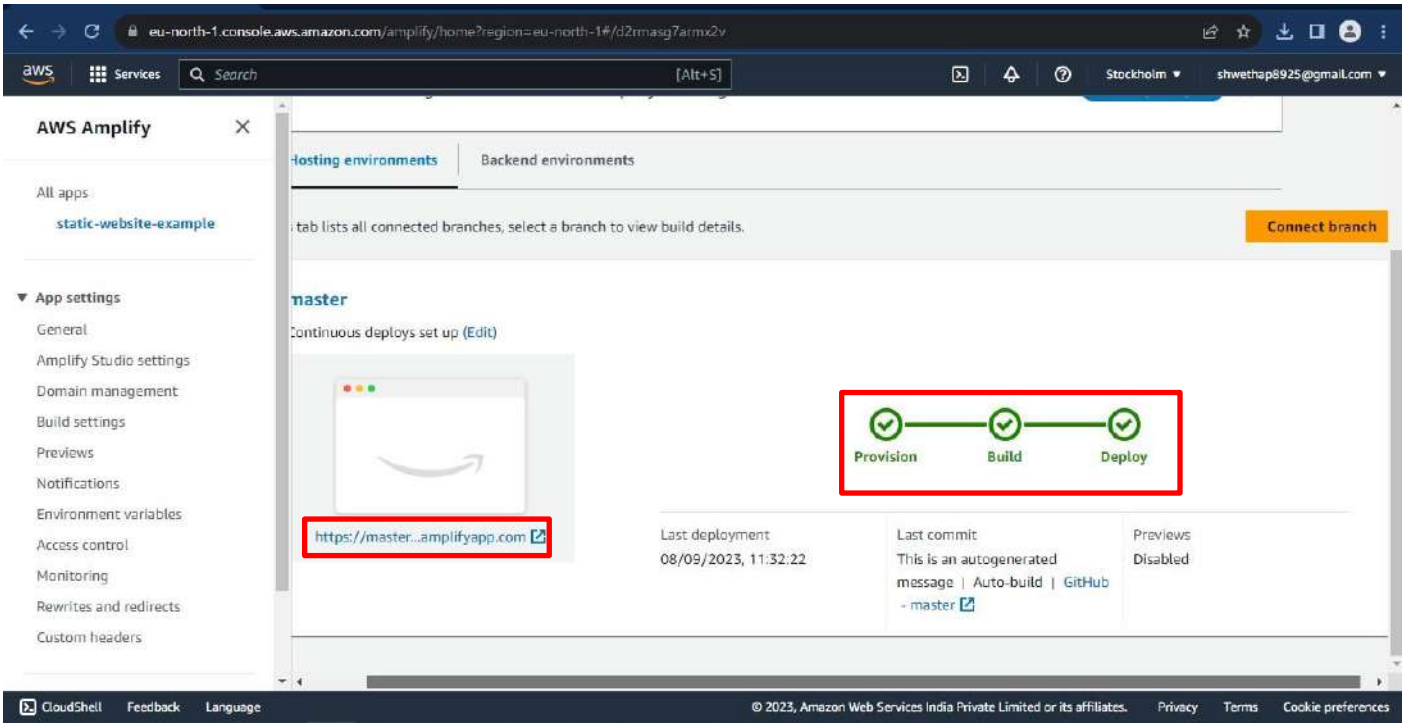
Step 28: Now enable the below checkbox and click on 'Next'.



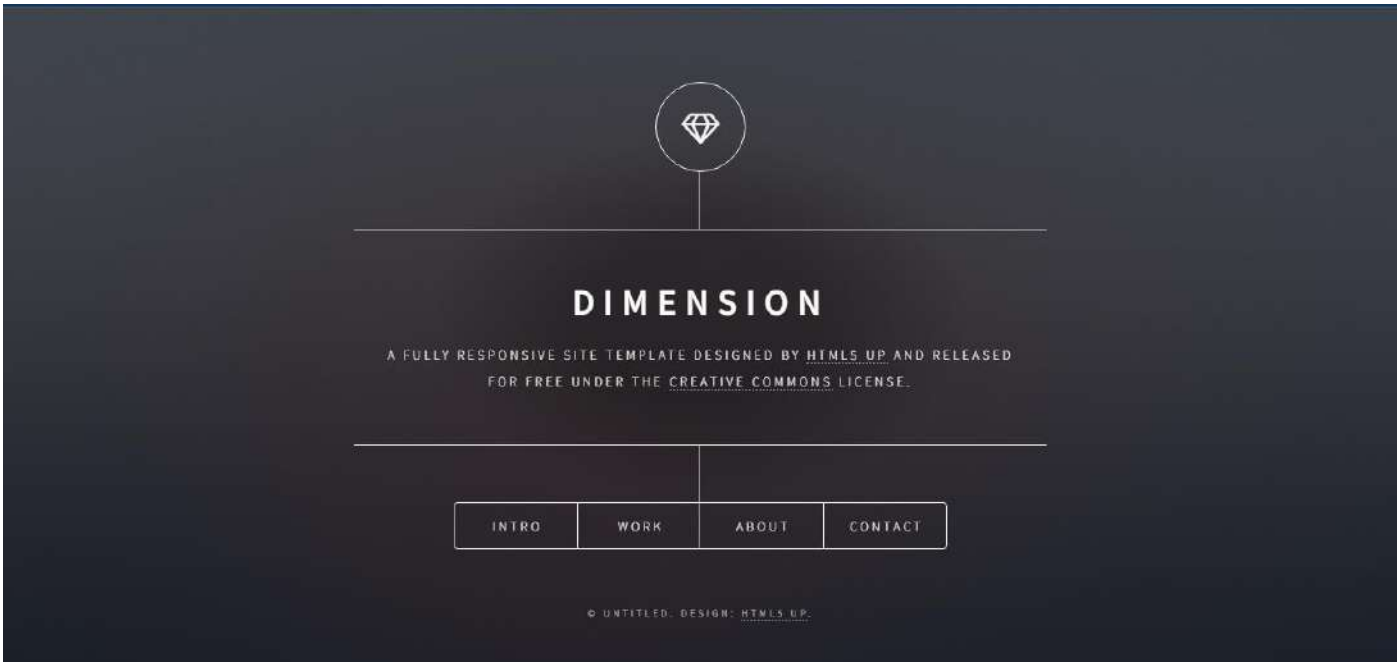
Step 29: Click on 'Save and deploy'.



Step 30: Wait until the Provision, Build and Deploy process completed. And to view the web app, click onthe below generated link.

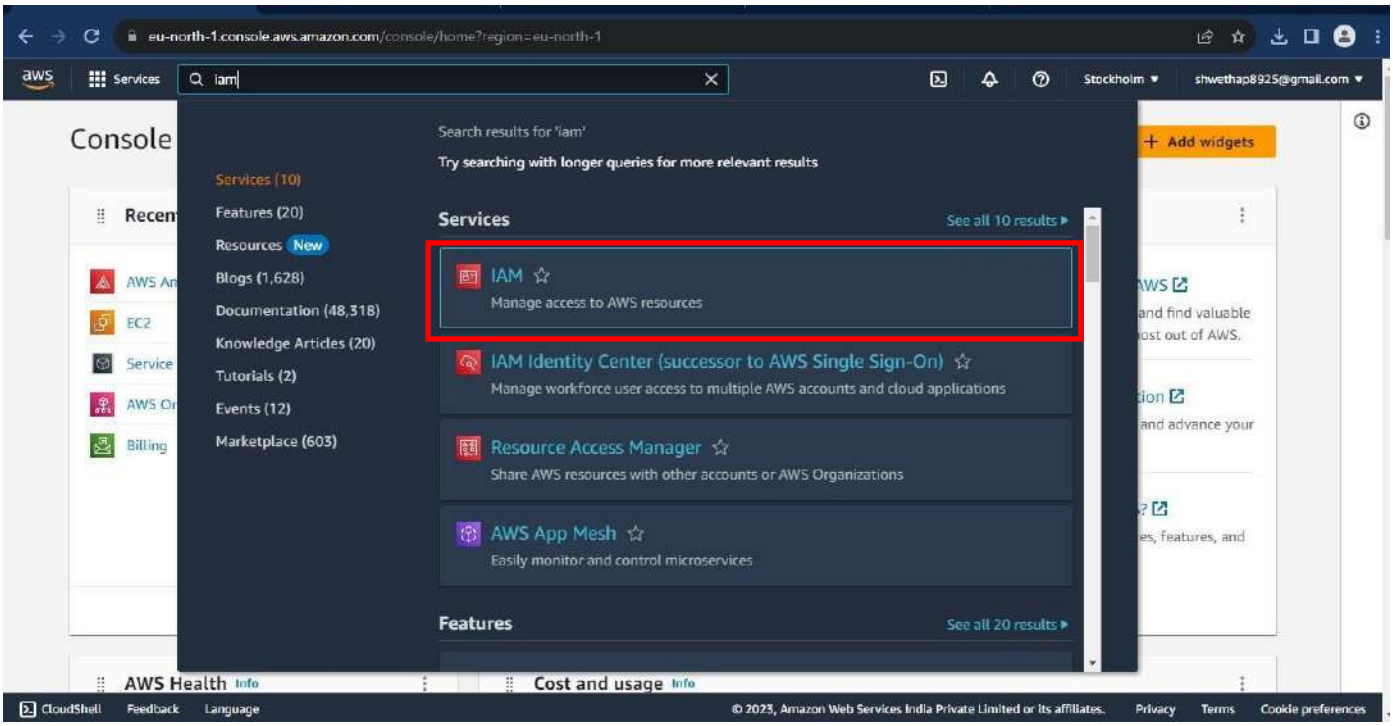


Step 31: Now we can see the web app which we have developed.

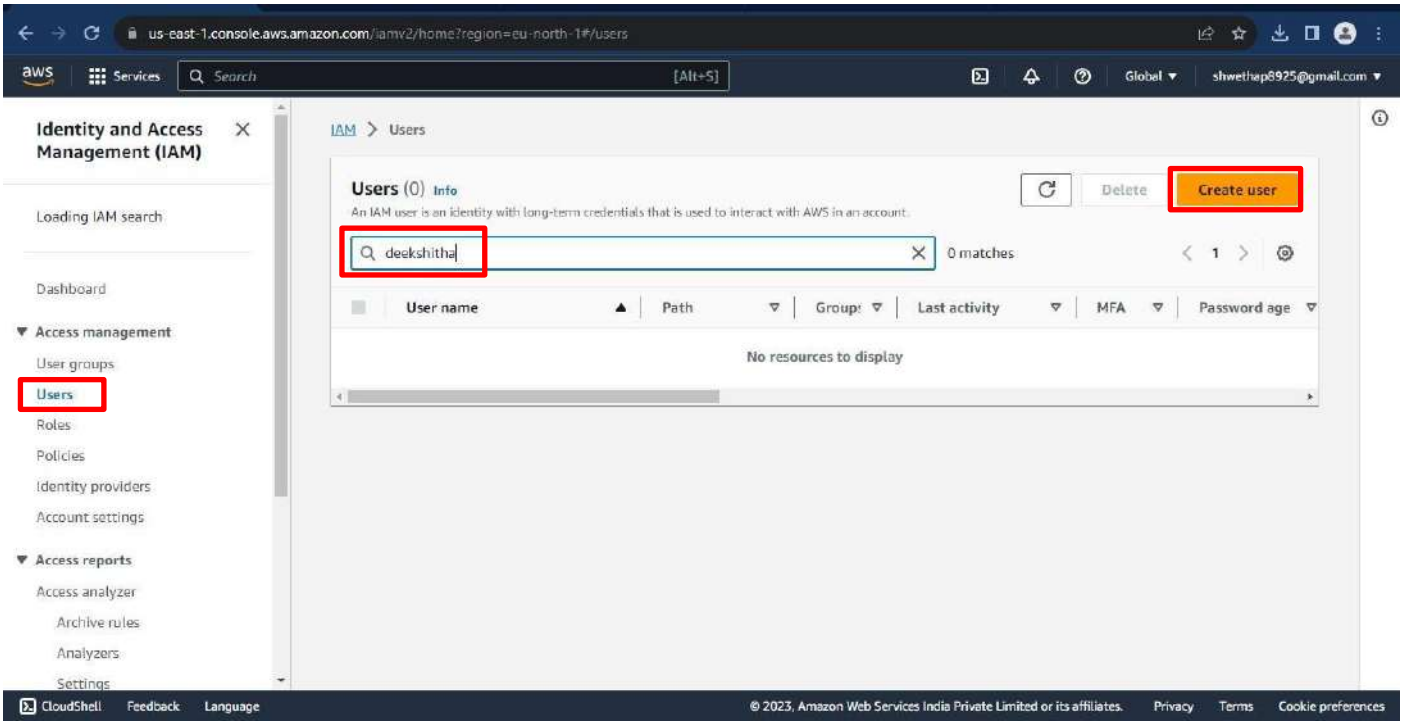


C. How to use cloud services for the user authentication flow (allowing access to otherusers by collaborating), allowing users to sign up, sign in and reset their password.

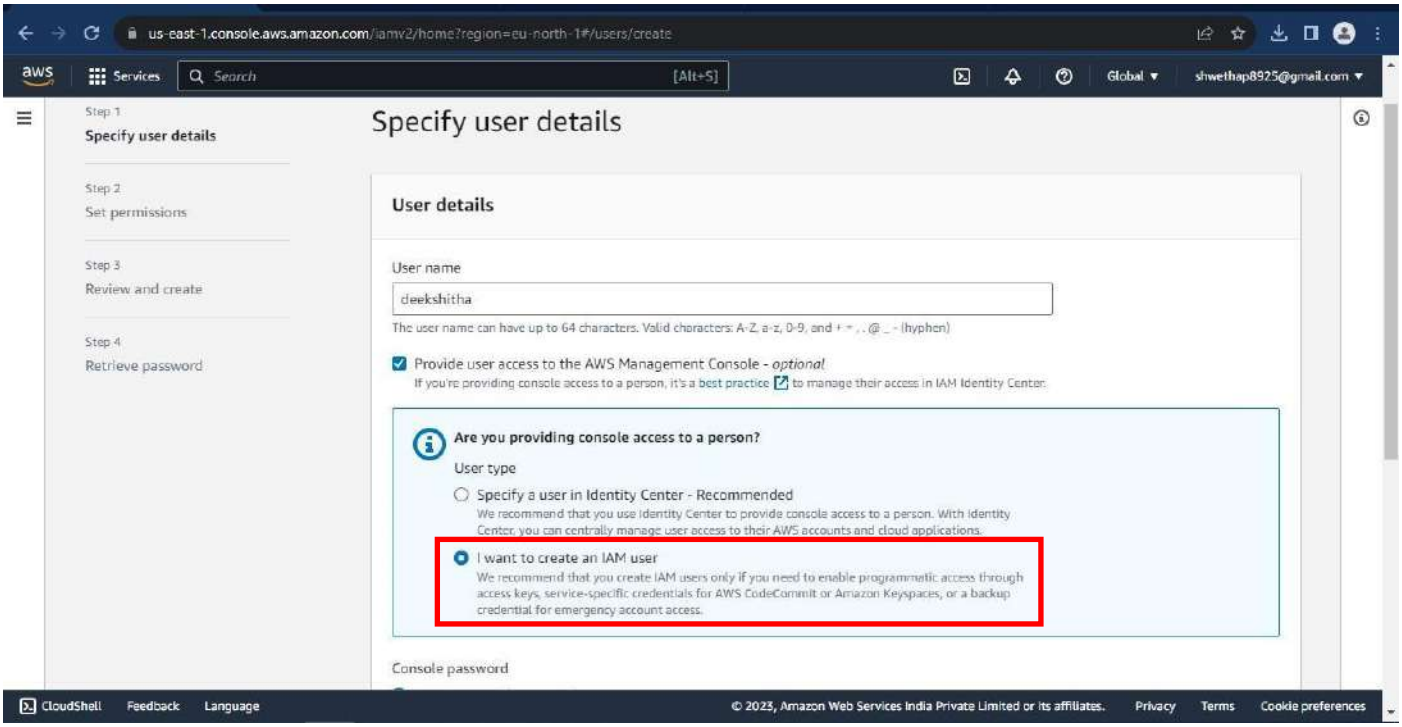
Step 32: After deploying the web app, come back to the AWS home screen and search for 'IAM' in theAWS search bar and click on the first option.



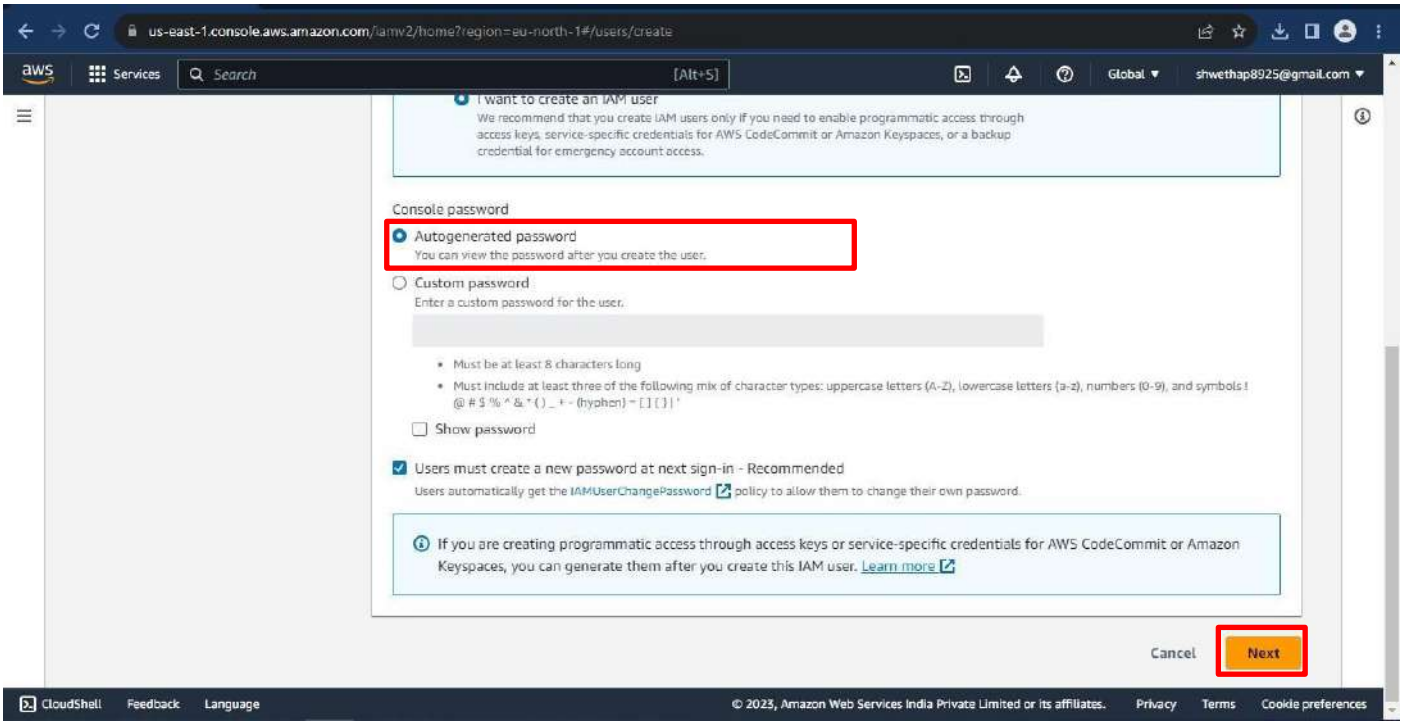
Step 33: Click on 'Users' and give the name of the User. Now click on 'Create User'.



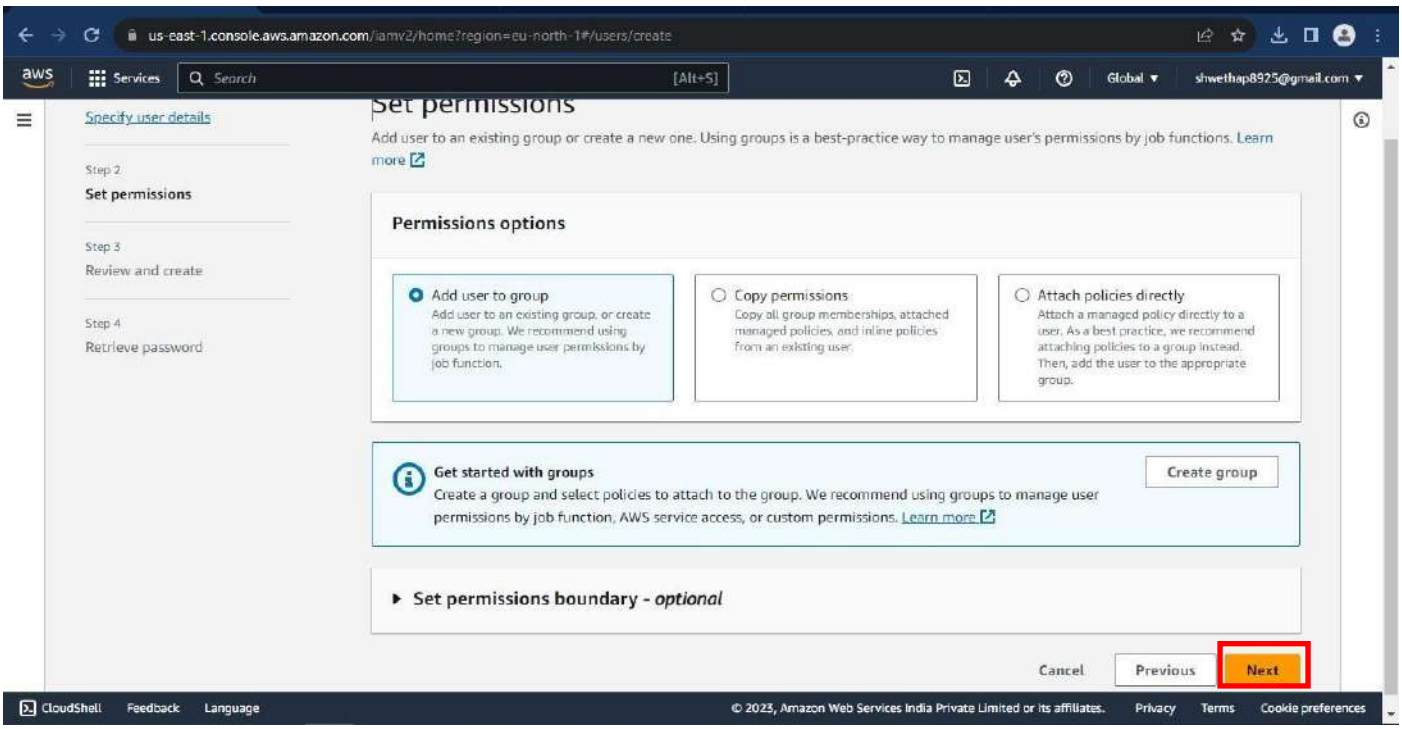
Step 34: Now select ‘I want to create an IAM user’,



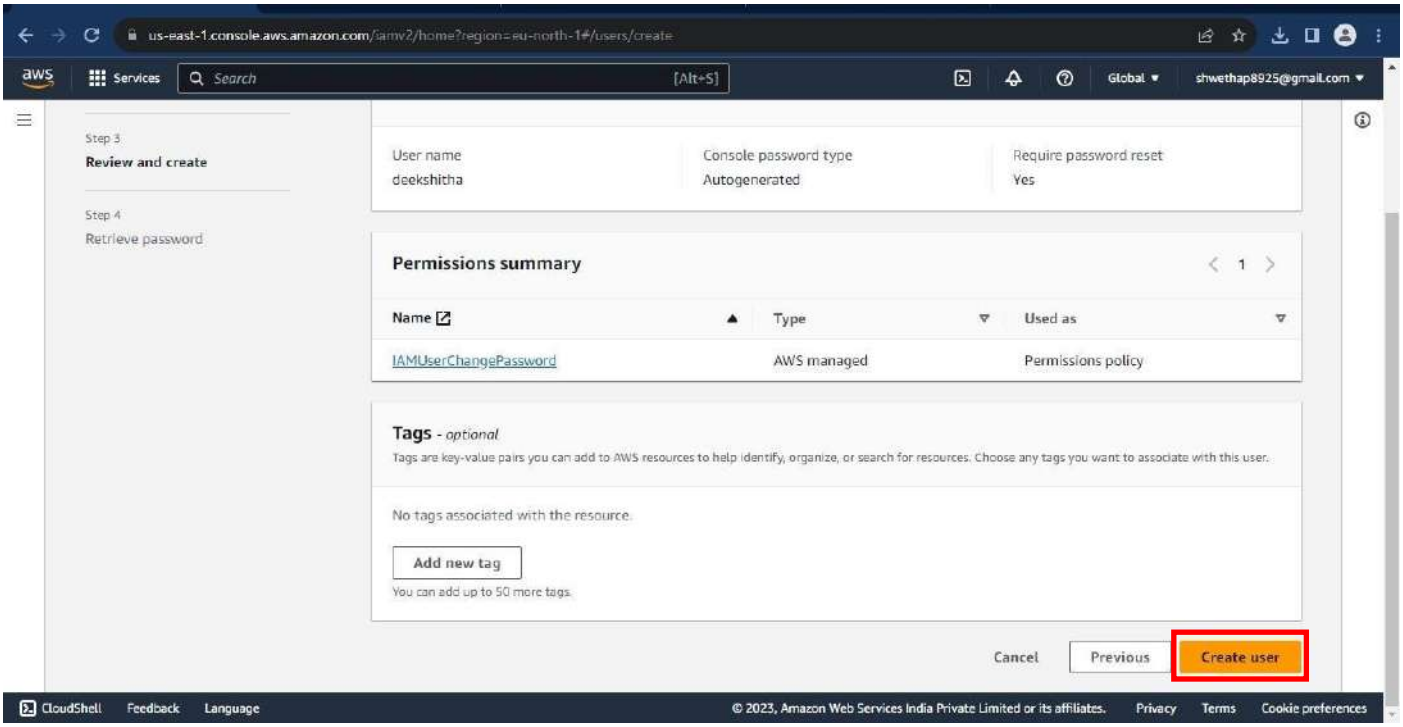
Step 35: Select ‘Autogenerated password’ and click on ‘Next’.



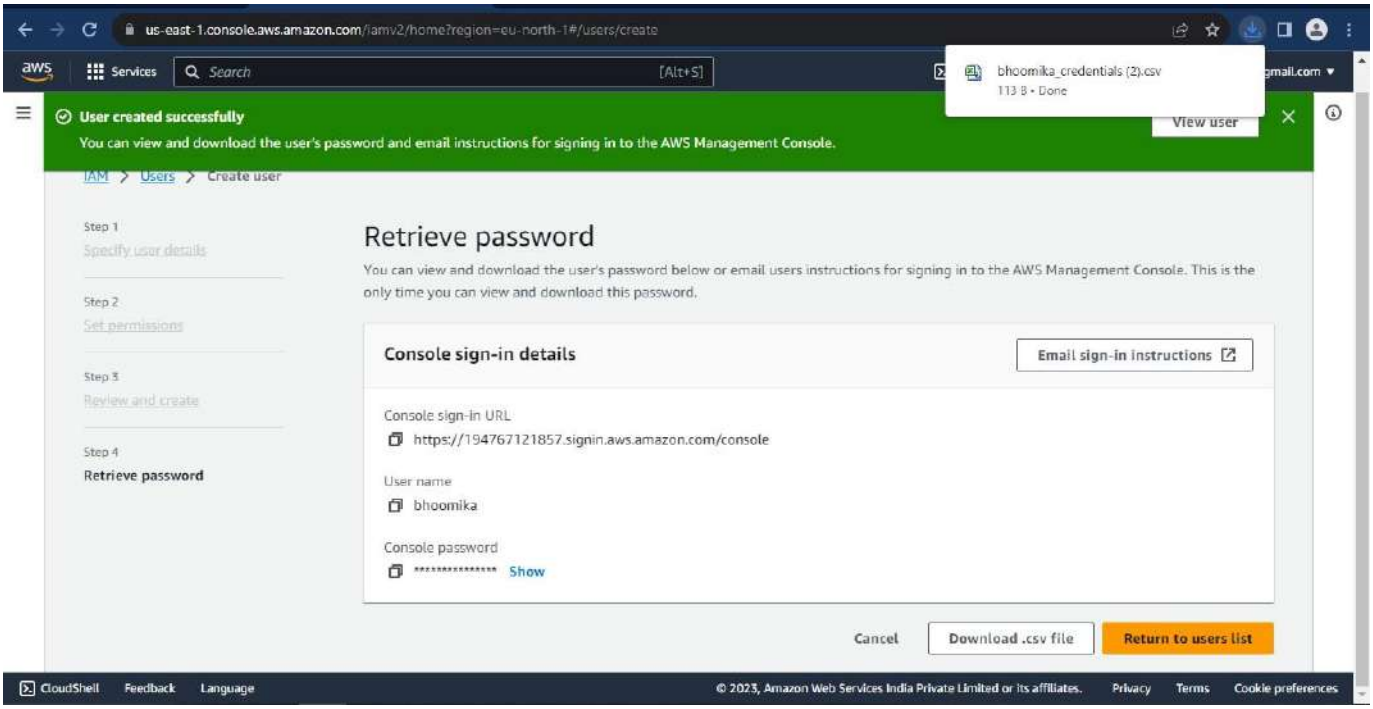
Step 36: Click on 'Next'.



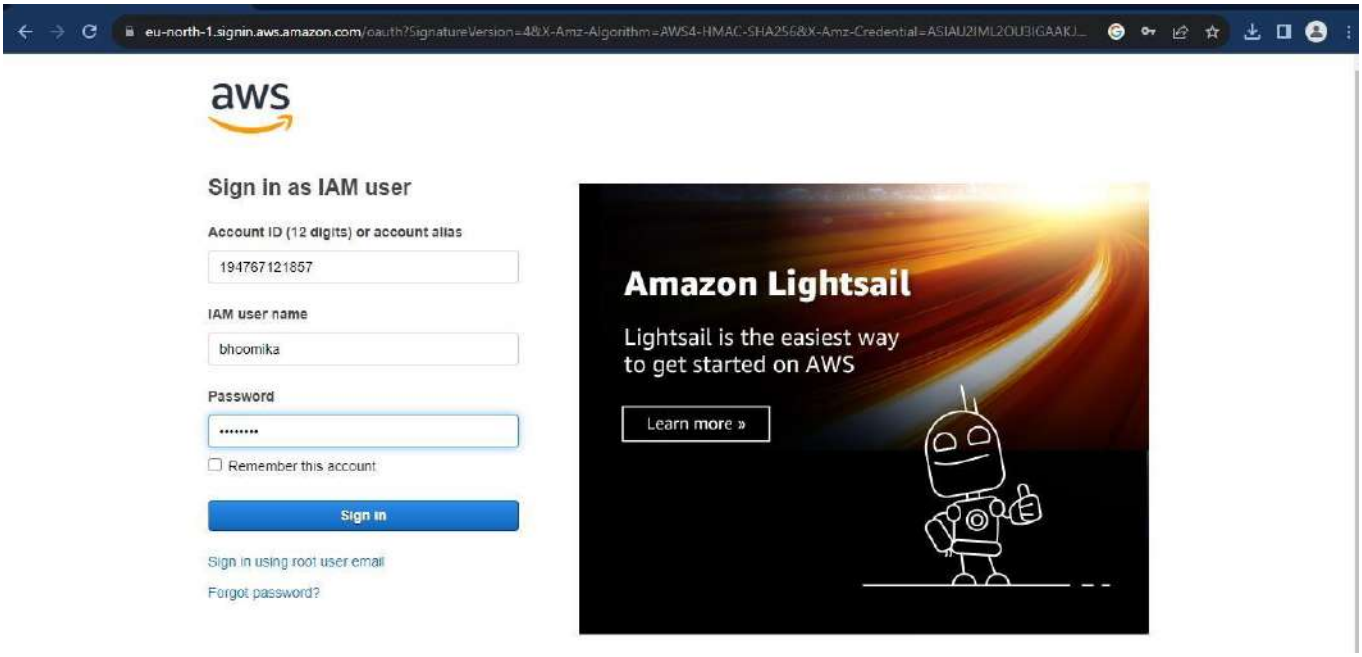
Step 37: Now click on 'Create User'.



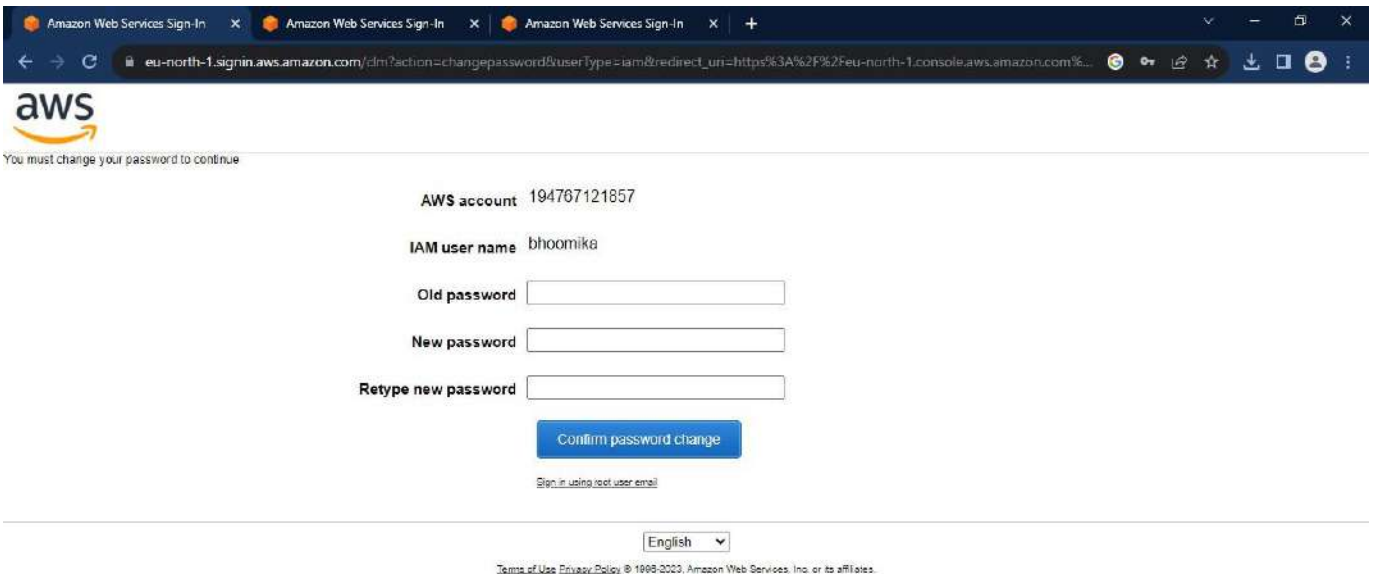
Step 38: Now copy the sign URL, Username and Password.



Step 39: Go to **IAM user** and enter the Username and password copied earlier and click on **Sign in**.



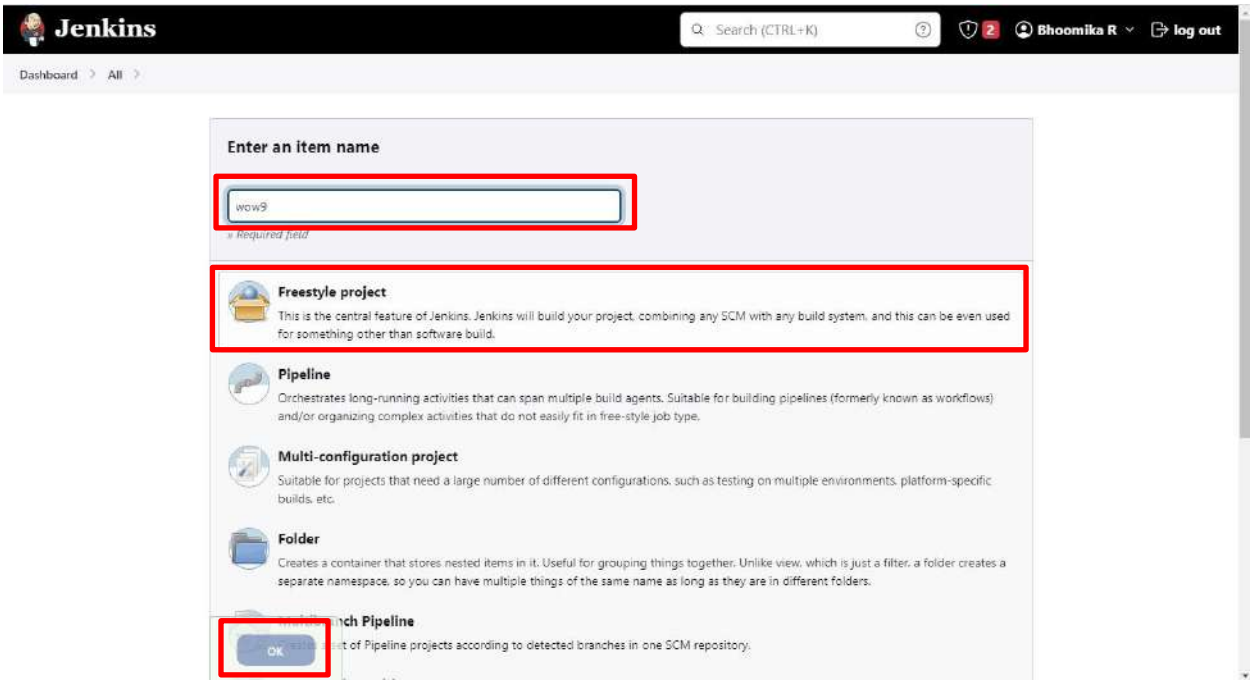
Step 40: You can also change your old password and set a new password and then click on **Confirmpassword change**.



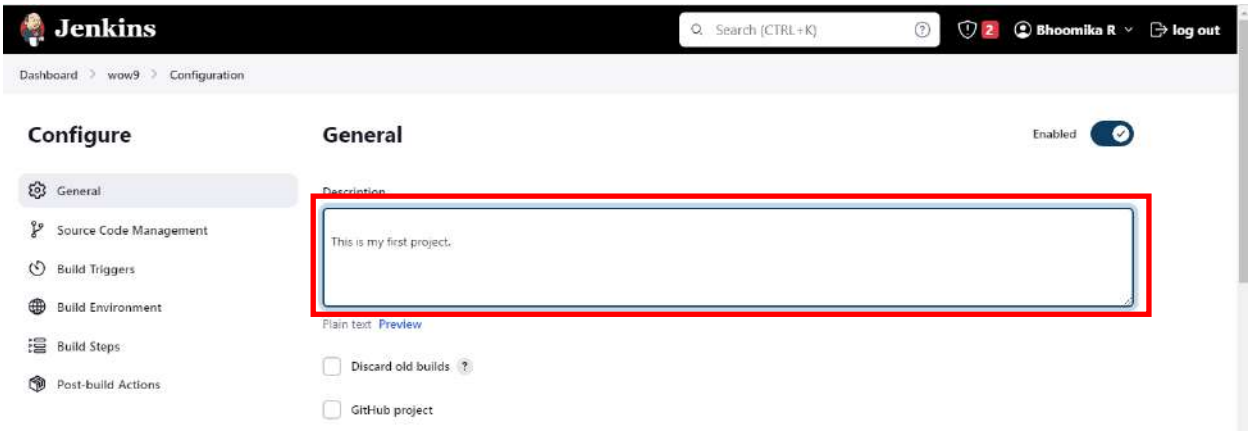
3. <u>Create a build pipeline for simple web application such as To-do app, BMI calculator, Number converter, Word Count etc.</u>

Step 1: Go to the Web browser and enter the **local host** (Jenkins port number) and press enter.

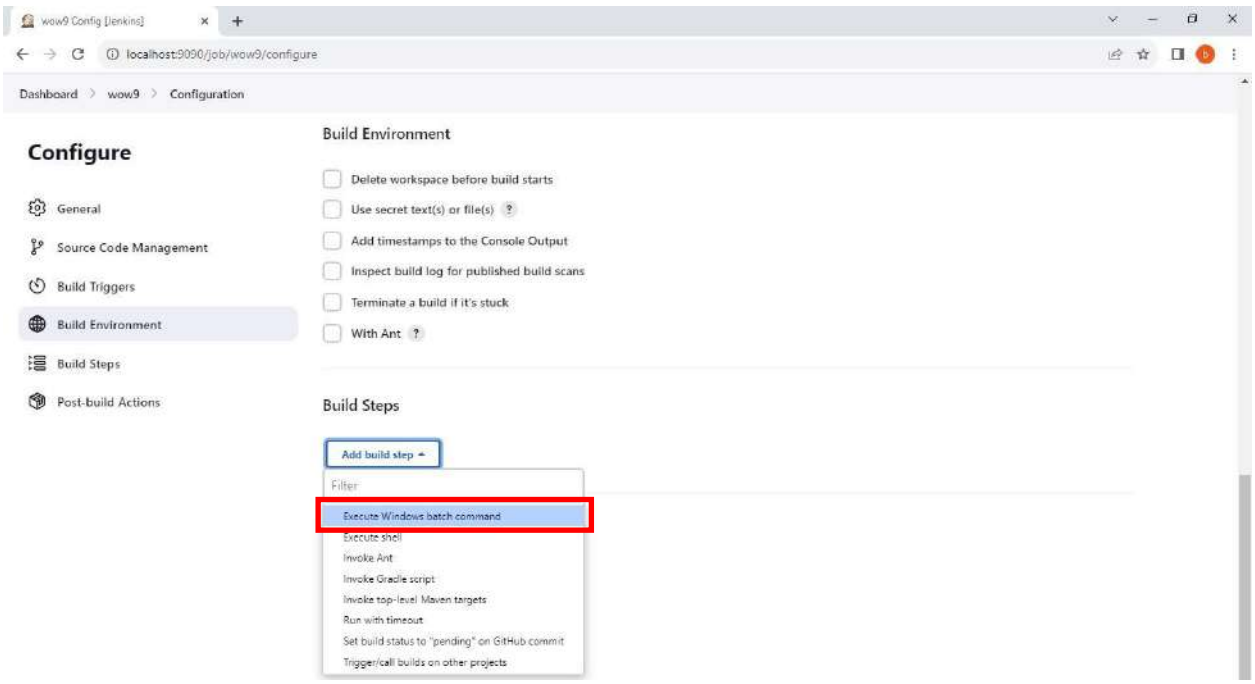
Step 2: In the Jenkins Dashboard Click on **new item** and create a free style project named **'Wow9'**.



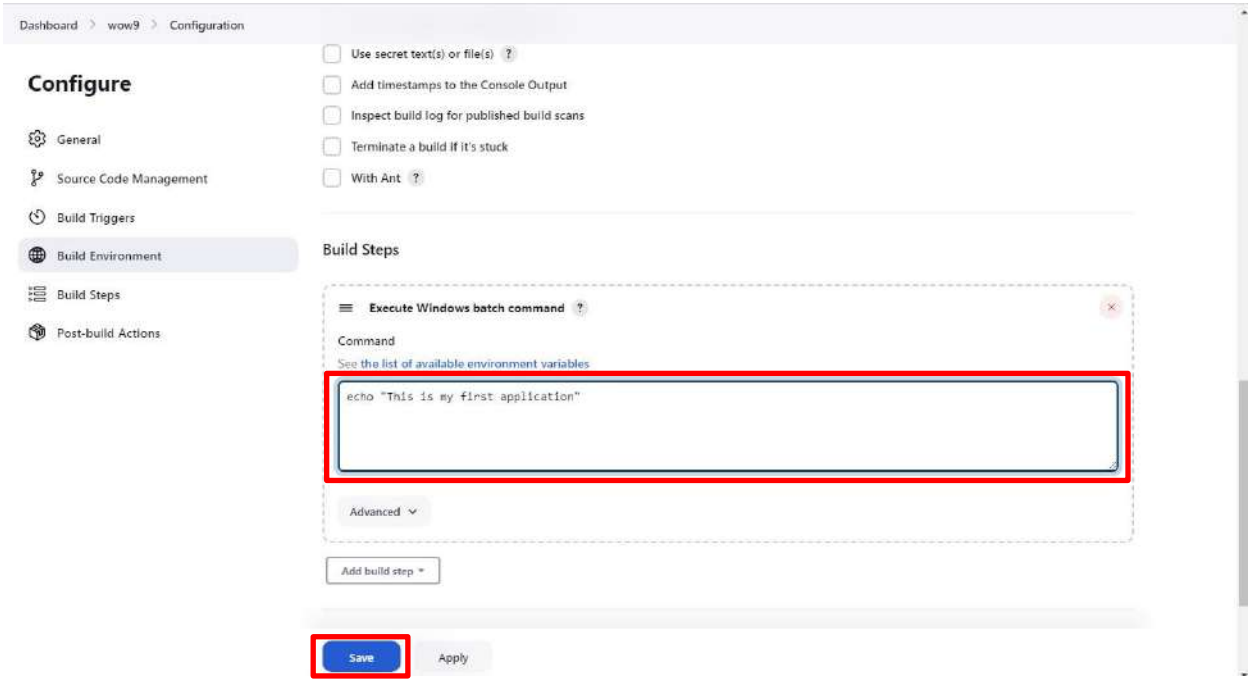
Step 3: On the next page, add the project description and scroll down to the **build steps**.

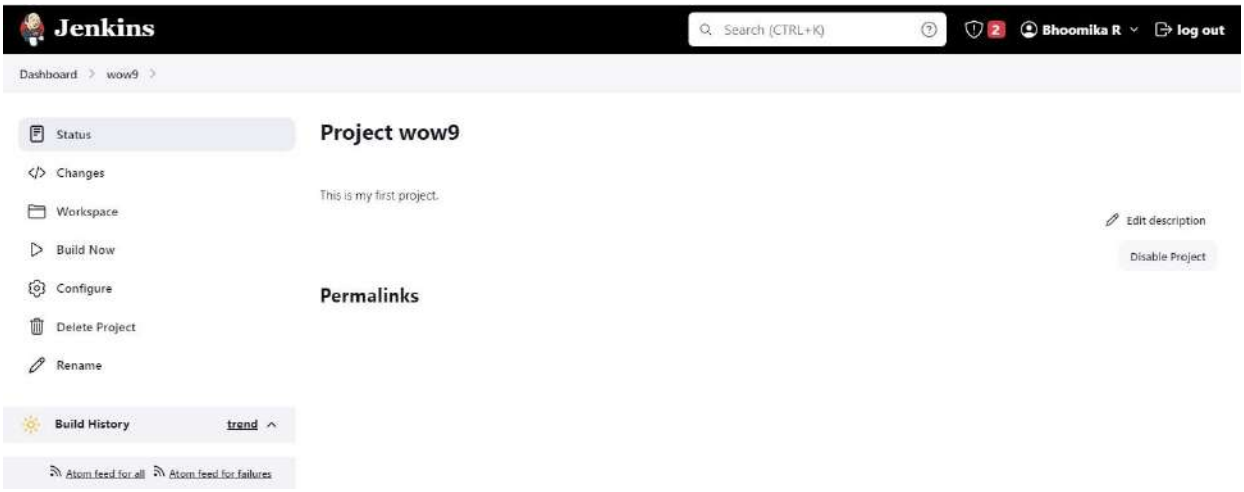


Step 4: Select **'Execute windows batch command'**.

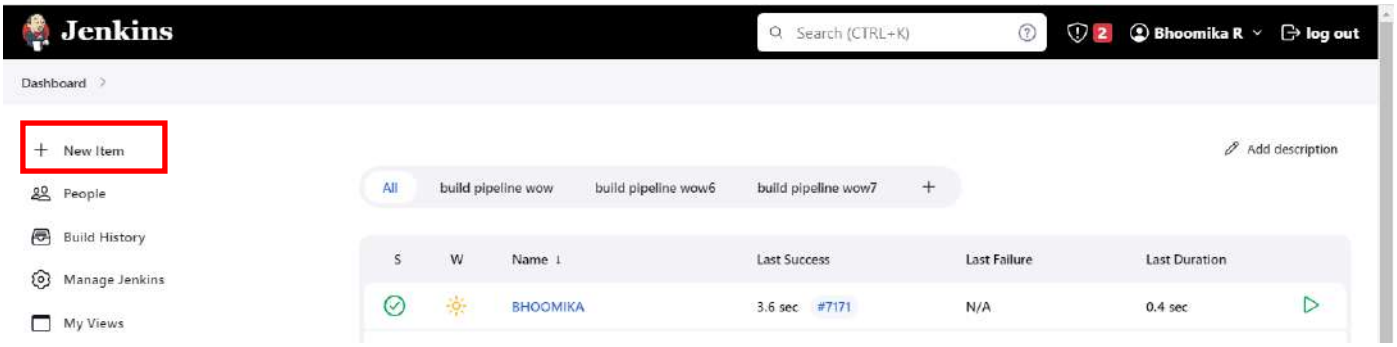


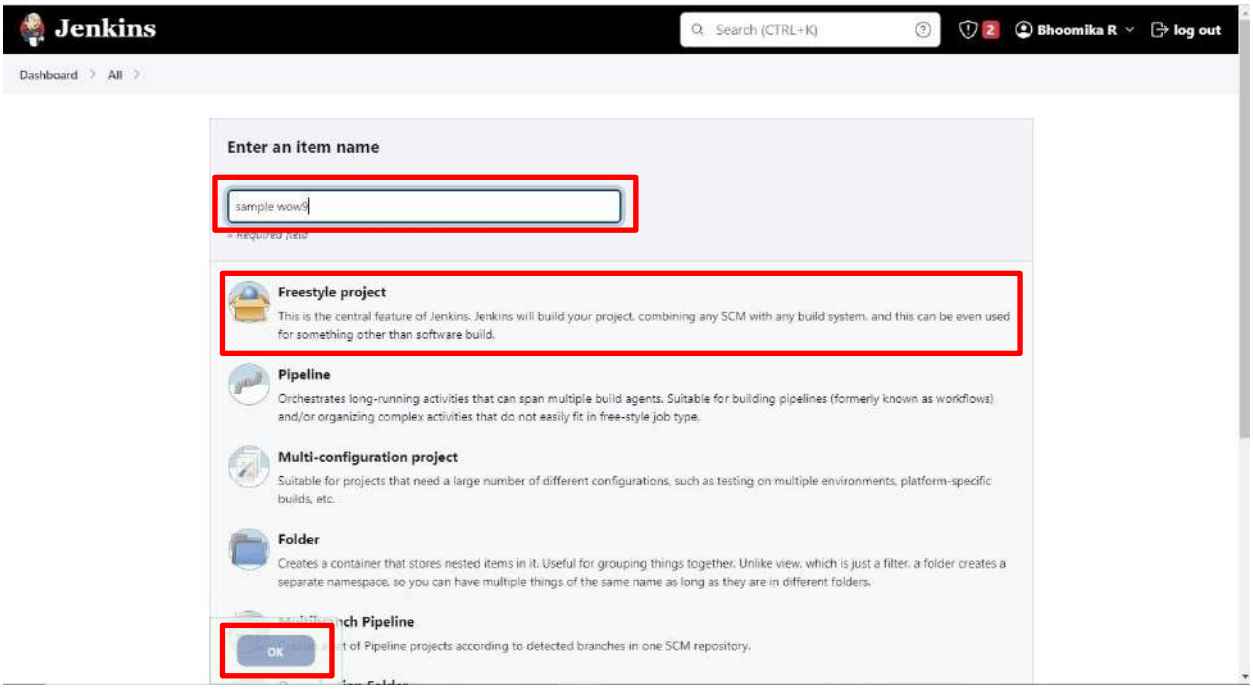
Step 5: And enter a command and click on **Save**.





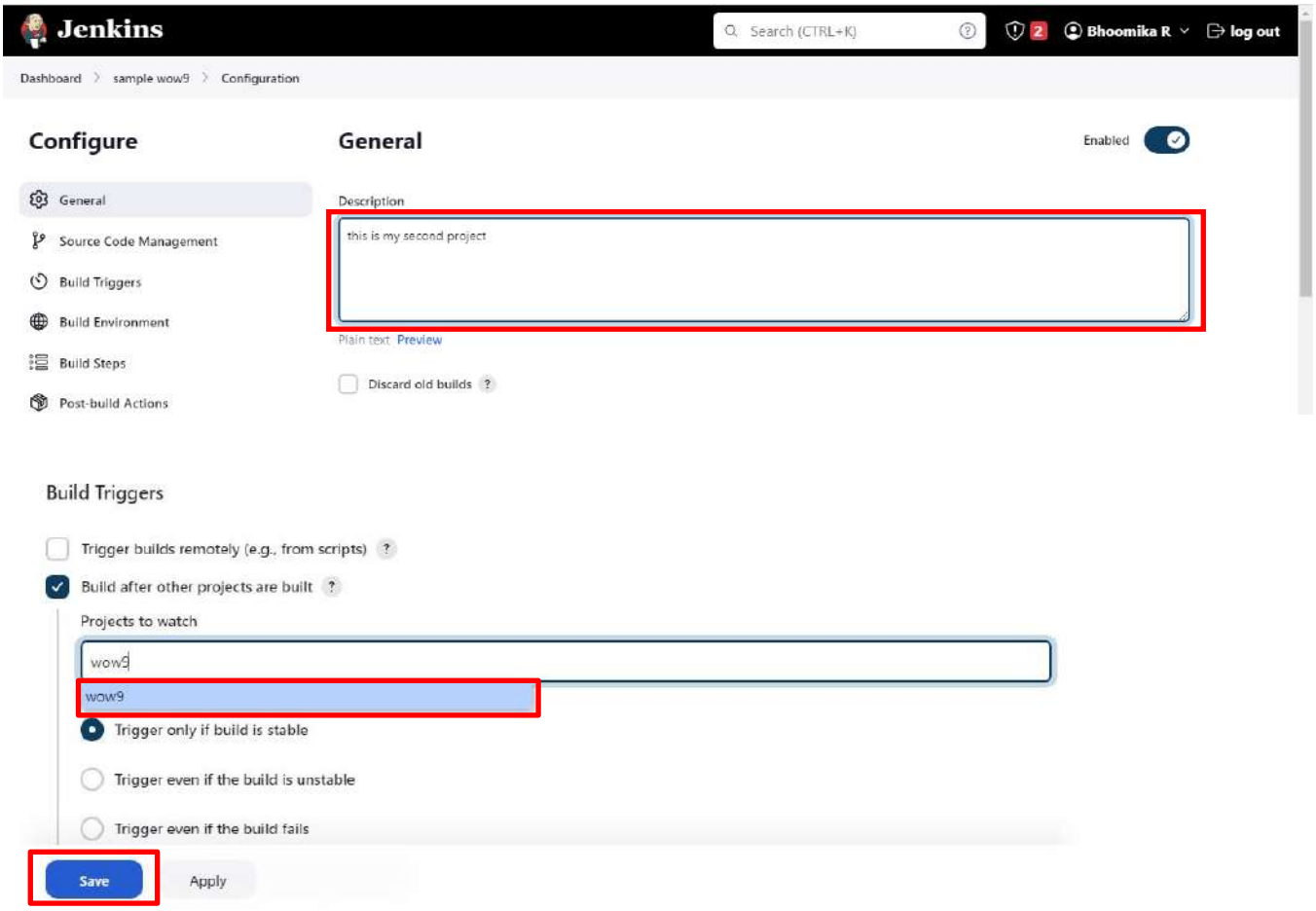
Step 6: Now go back to dashboard and create another freestyle project named **'Sample wow9'**.





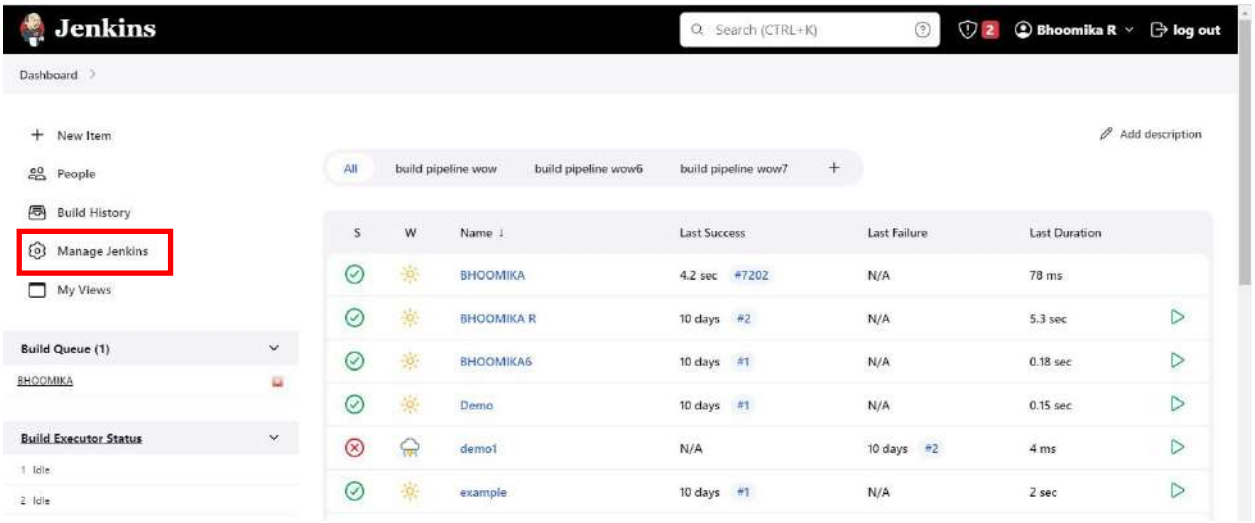
Step 7: Give some description & under the **Build Trigger tab**, Select **Build after other projects are built**

and select previous job created i.e., **Wow9**.

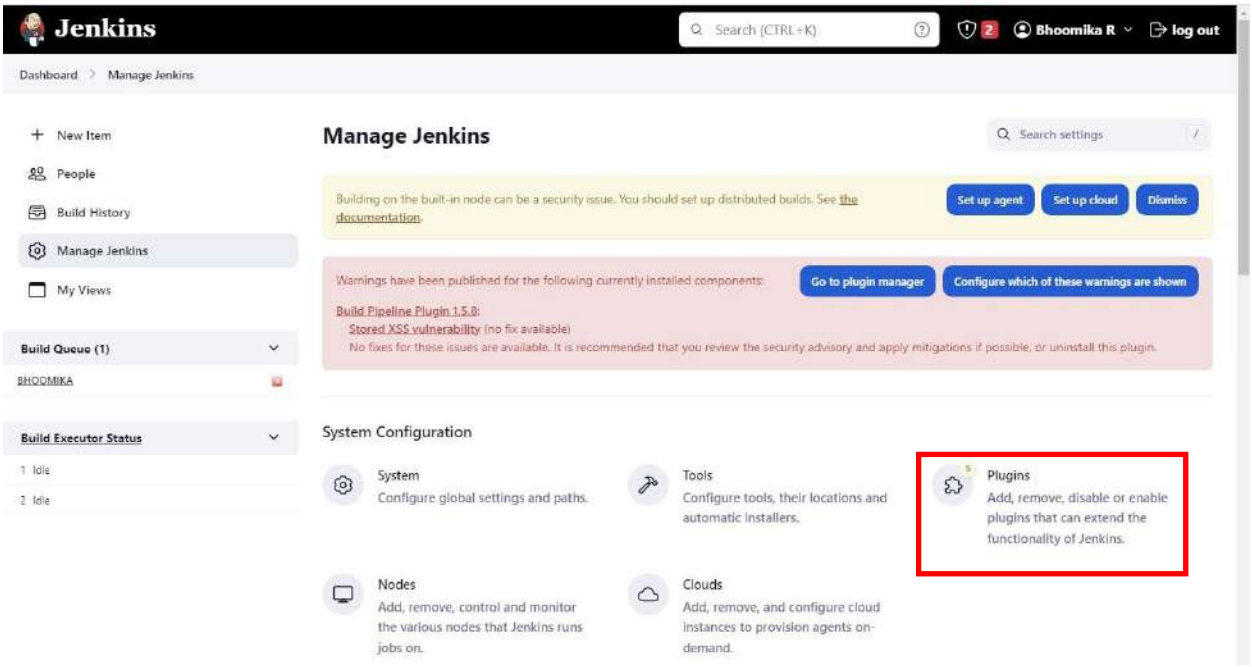


Step 8: Continue these steps for creating another three freestyle projects.

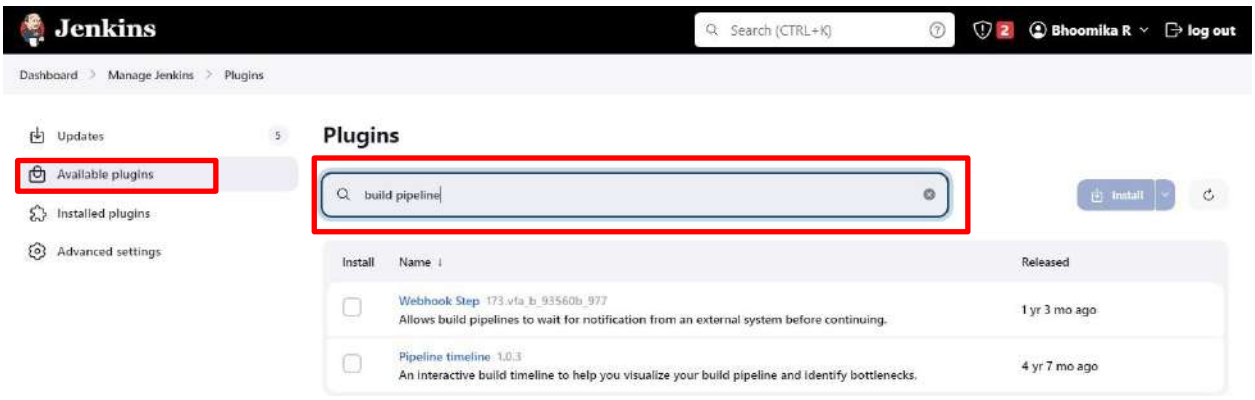
Step 9: After the creation of all four projects, Go back to dashboard and select **Manage Jenkins**.



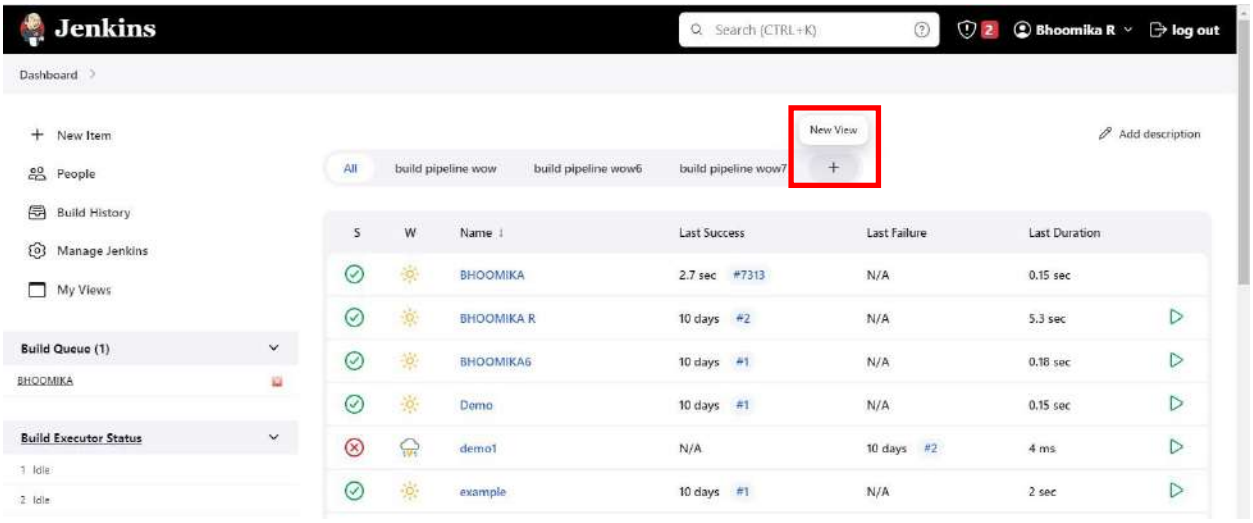
Step 10: Go to **Plugins**.



Step 11: Click on **Available Pulgins** and give a name to create a new plugin. Install the **pulgins** if it is notthere.

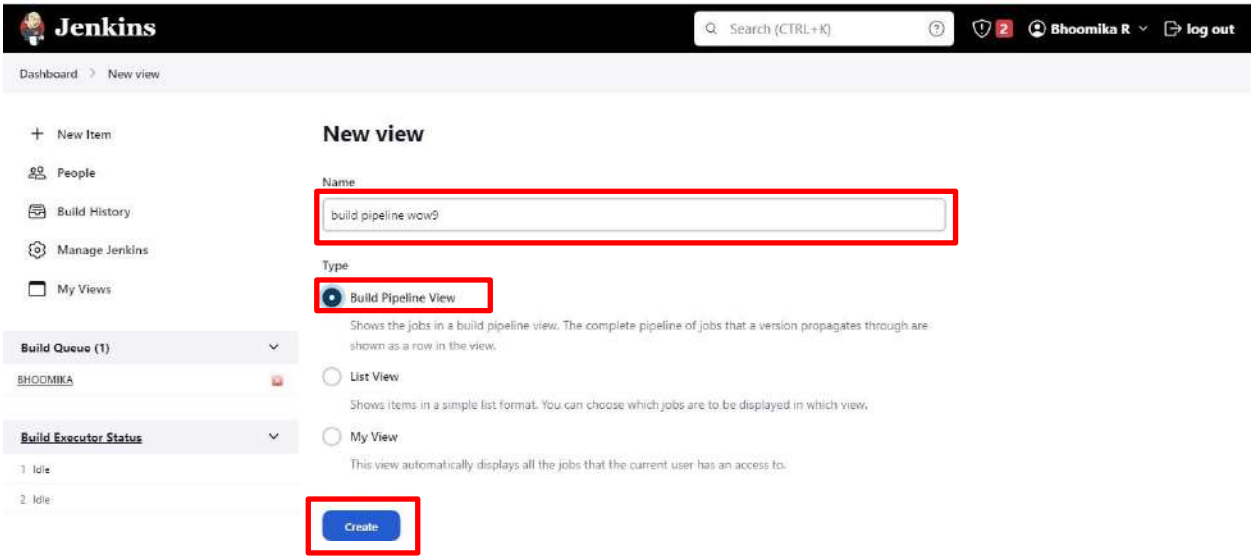


Step 12: Now Go back to the **Dashboard** & Click on **+ icon**.

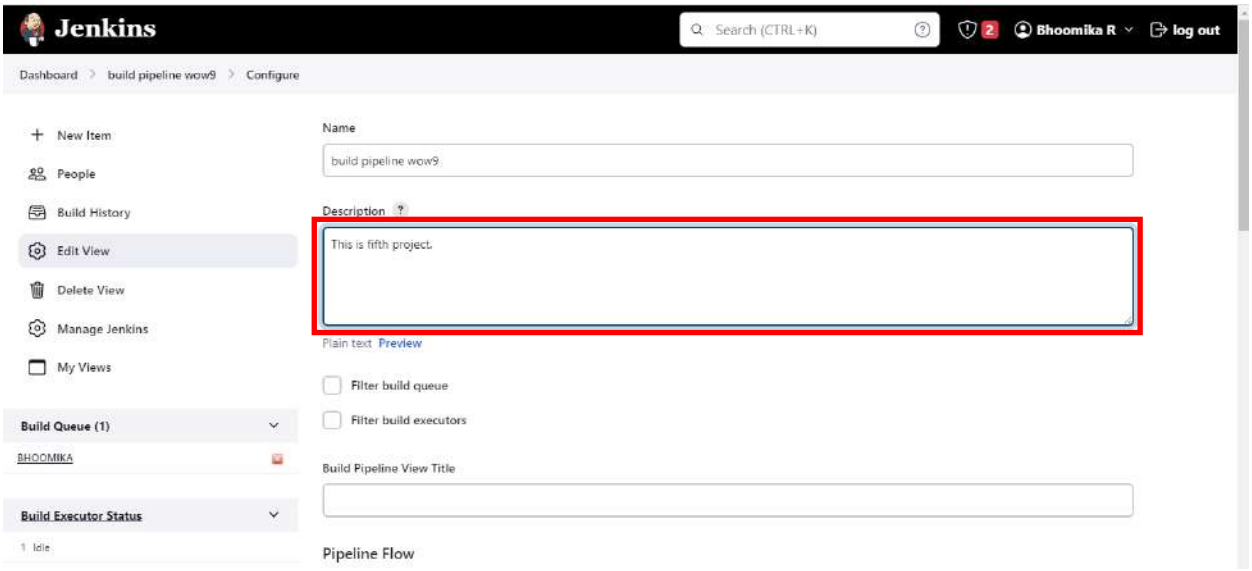


Step 13: On the next page, give a name to the pipeline and select a **Build Pipeline View** and Click on

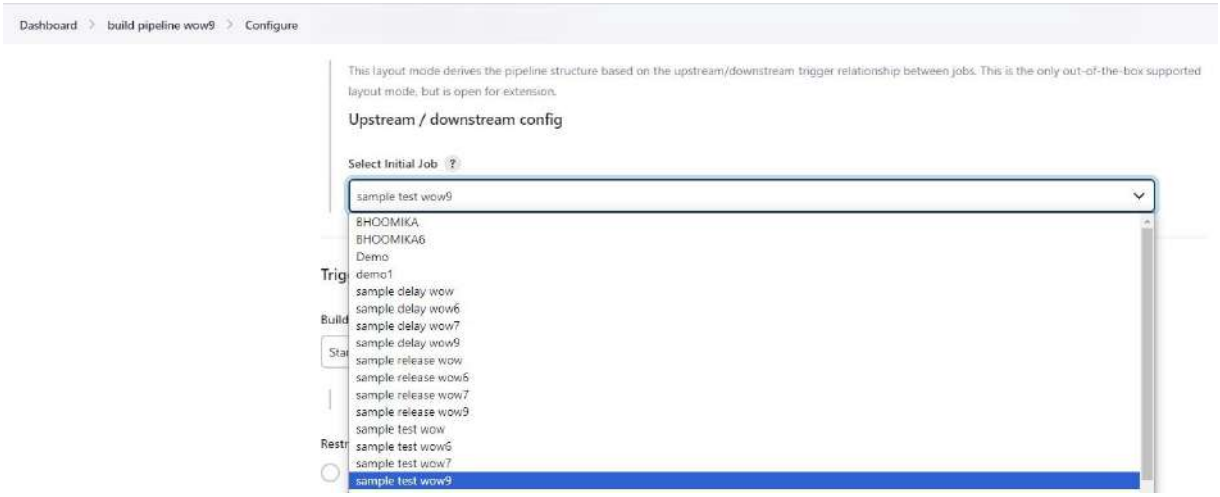
Create.



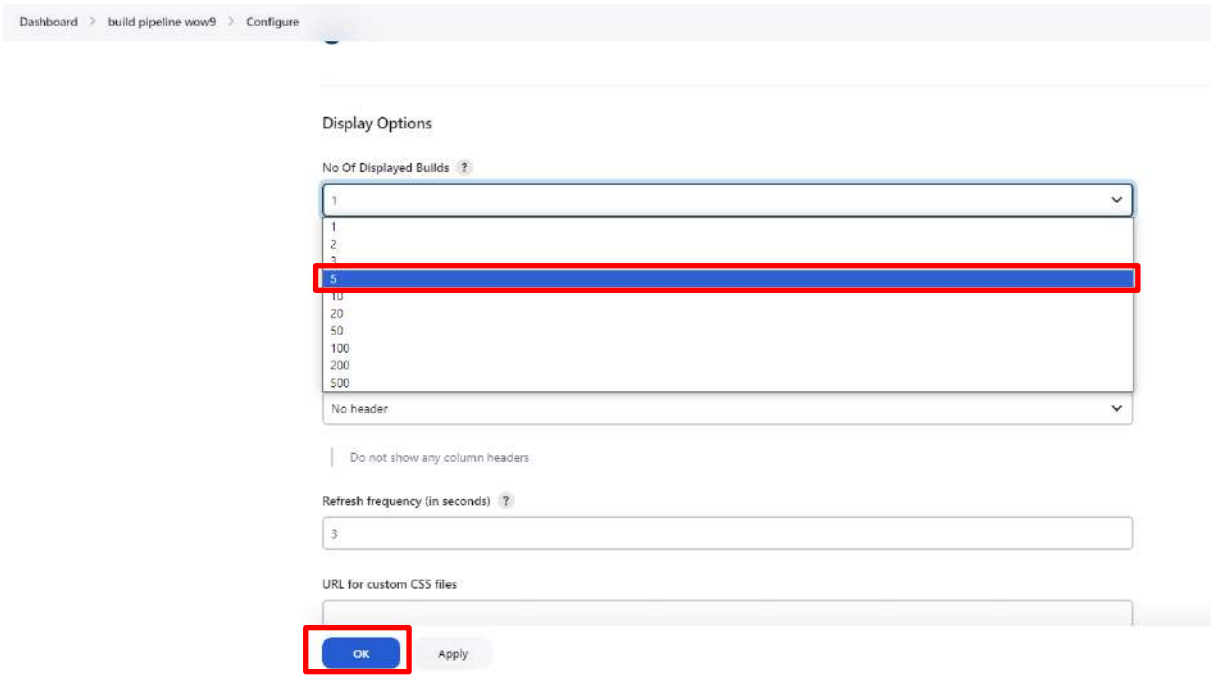
Step 14: Now add the description and scroll down to select **Initial Job Section**.



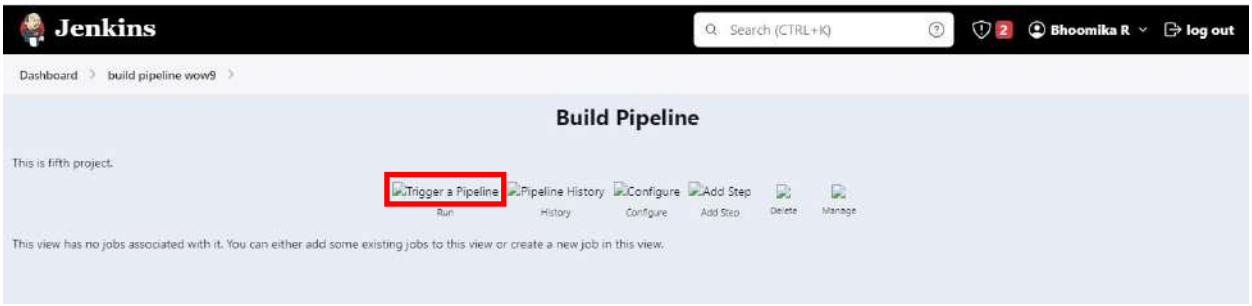
Step 15: Select the Initial job as **sampletestwow9**.



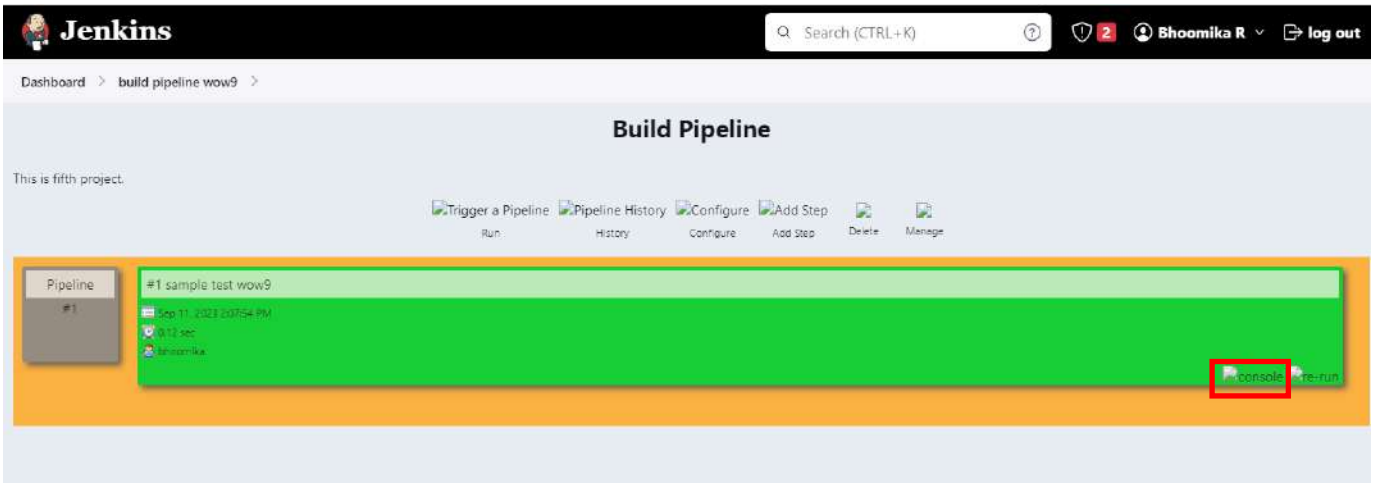
Step 16: Scroll down to **Display Section Tab** and Select the **no for Displayed build to be 5**, click on **ok**.



Step 17: Click on **Trigger a Pipeline** to View the Pipeline.

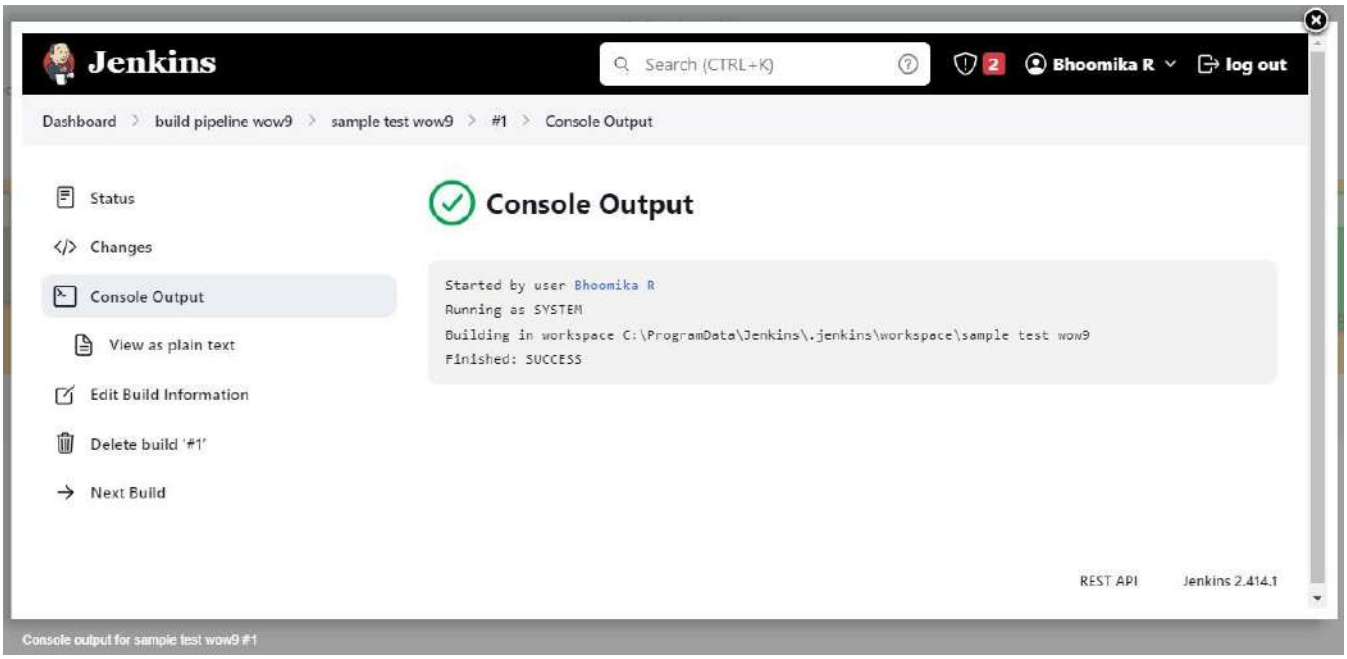


Step 18: Now you can see the pipeline Build Flow of all the jobs.



Step 19: All the Successful jobs are Shown in the green, pending in blue and failed in red. Now click on

Console Output to view the build outlook.



WEEK – 4

1. **HTML, CSS and Java Script Fundamentals (Code structure – statements, comments, variables, constants, data types, interaction, operators, comparisons, control flow, functions)**

A) Using HTML with JavaScript.

index.html
```
<html>
    <body>
        <head>
            <link rel="stylesheet" href="style.css">
        </head>
        <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><center>
    <form name="myform" method="post" action="valid.html" onsubmit="return validateform()">
    Enter your Name: <input type="text" name="name"><br><br>
    Enter your Password: <input type="password" name="password"><br><br>
    <input type="submit" value="Register">
    </form>
    </body></center>
</html>
```

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

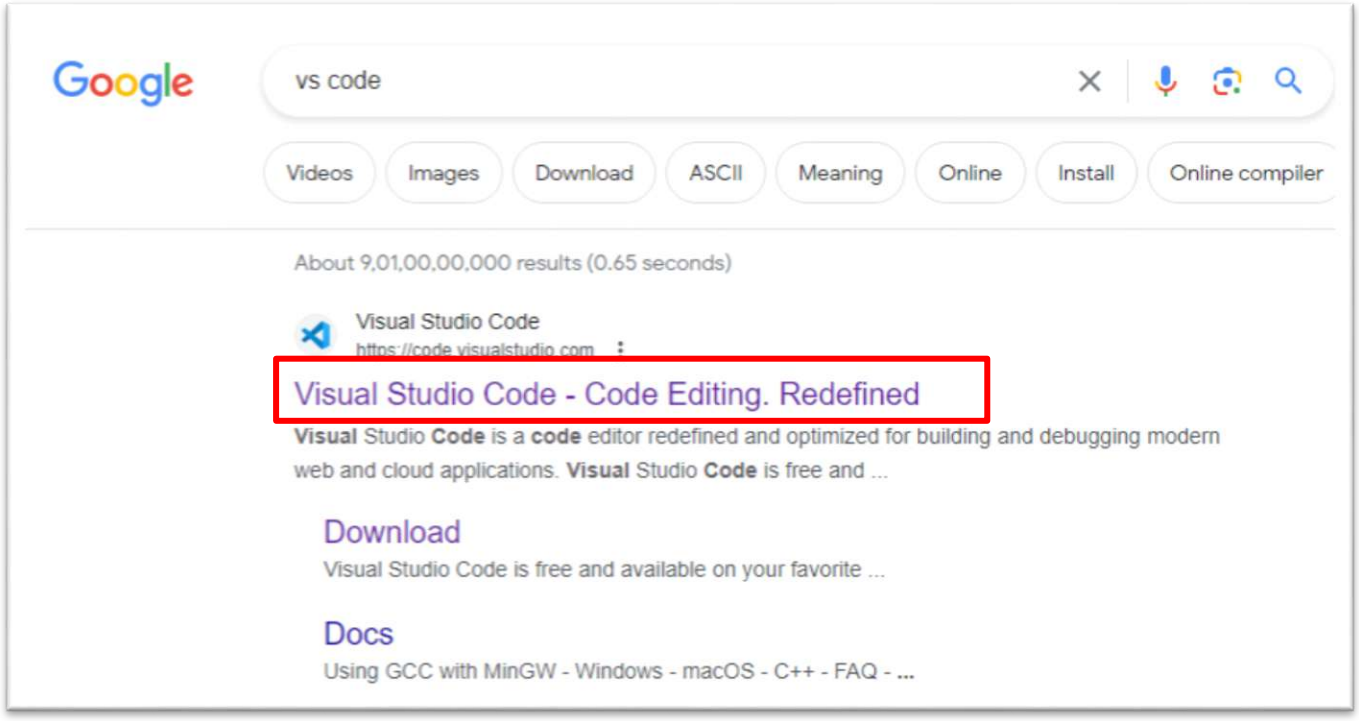
B) Using CSS file.

style.css
```
body {
    background-color: rgb(159, 203, 204);
    margin: 100px;
    padding: 50px;
}

```

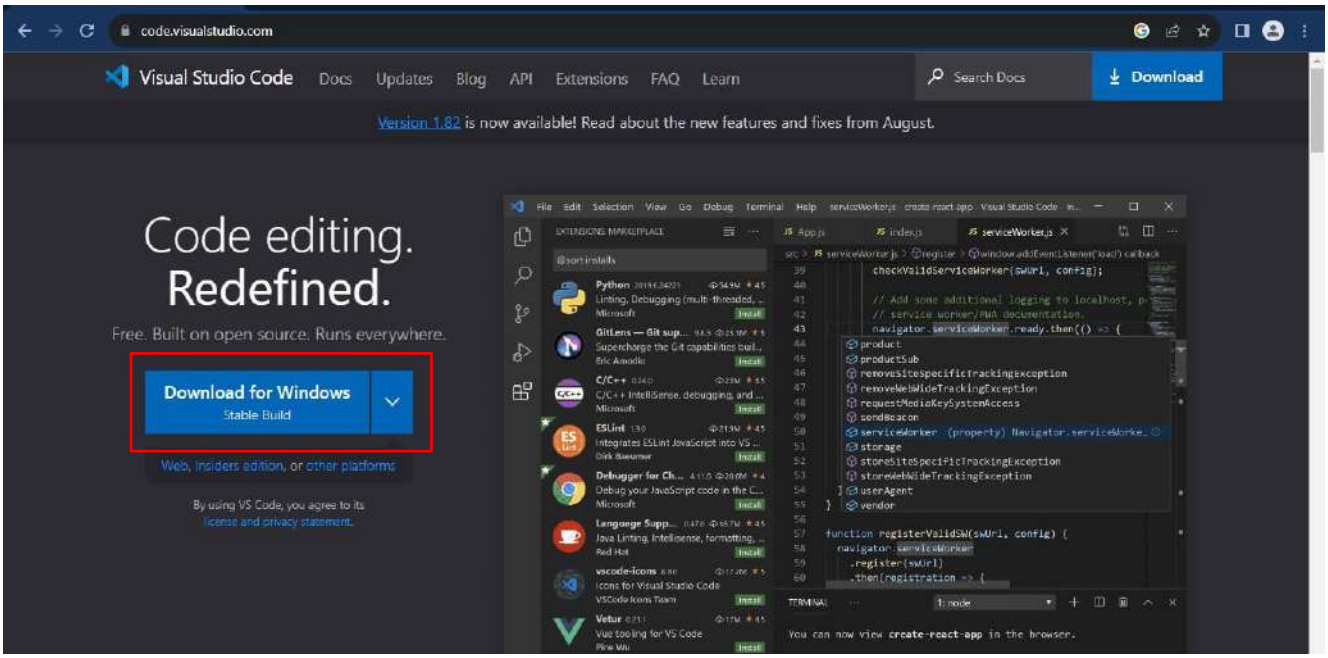
2. Setting up the Environment and tools for Front End Development.

- **Installation of VS Code.**

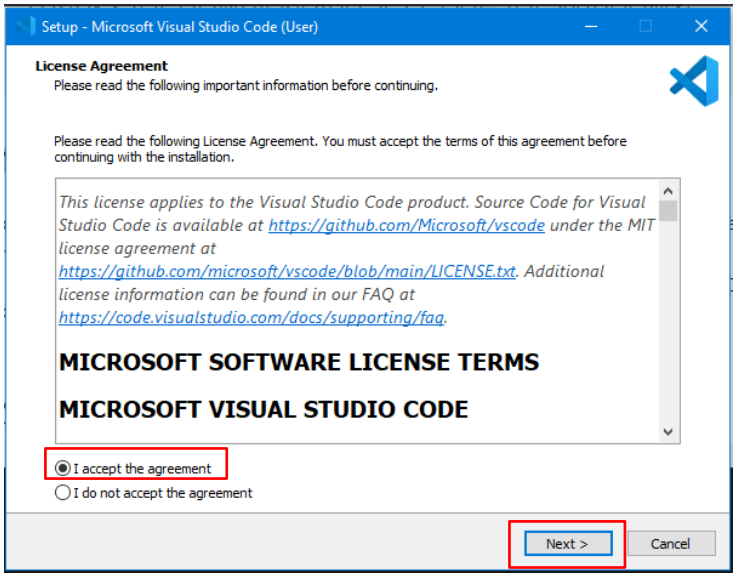
Step 1: Go to Web Browser and search for VS Code. Then click on the first link.



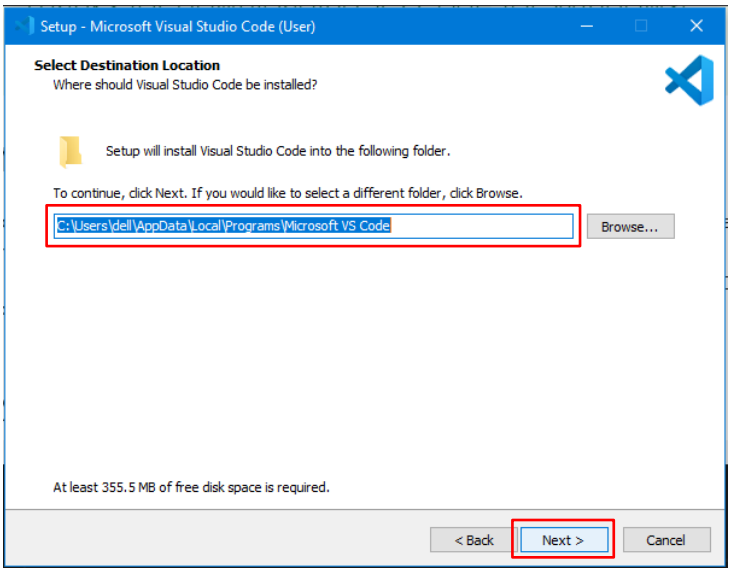
Step 2: Click on **'Download for Windows'**.



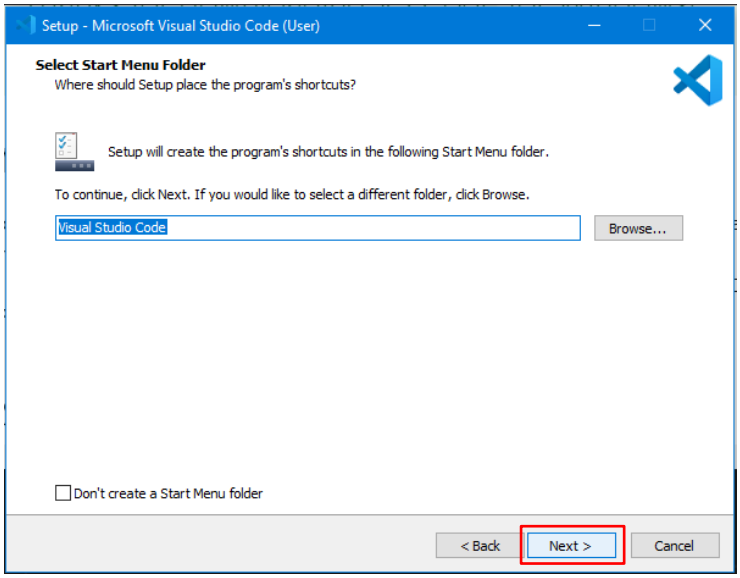
Step 3: Enable the first option **'I accept the agreement'** & click on **'Next'**.



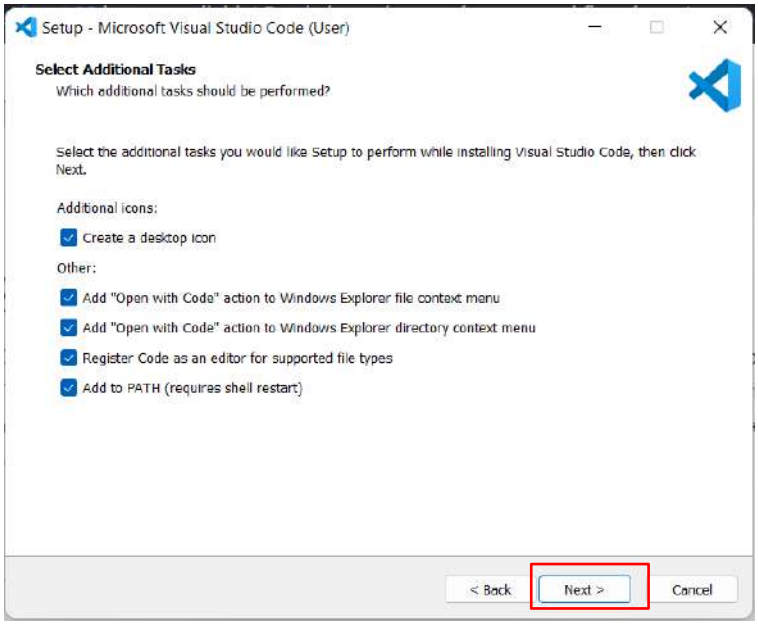
Step 4: Choose the location to save the file & click on **'Next'**.



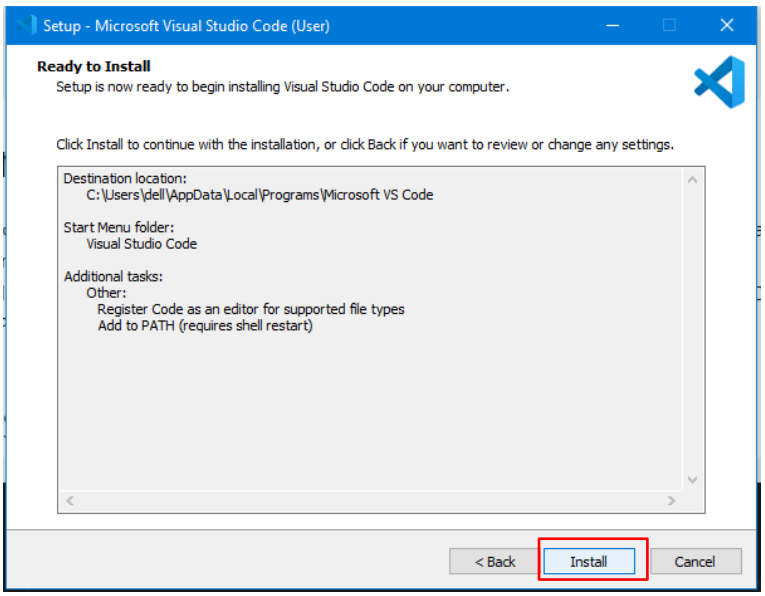
Step 5: Select a folder name or click on **'Next'**.



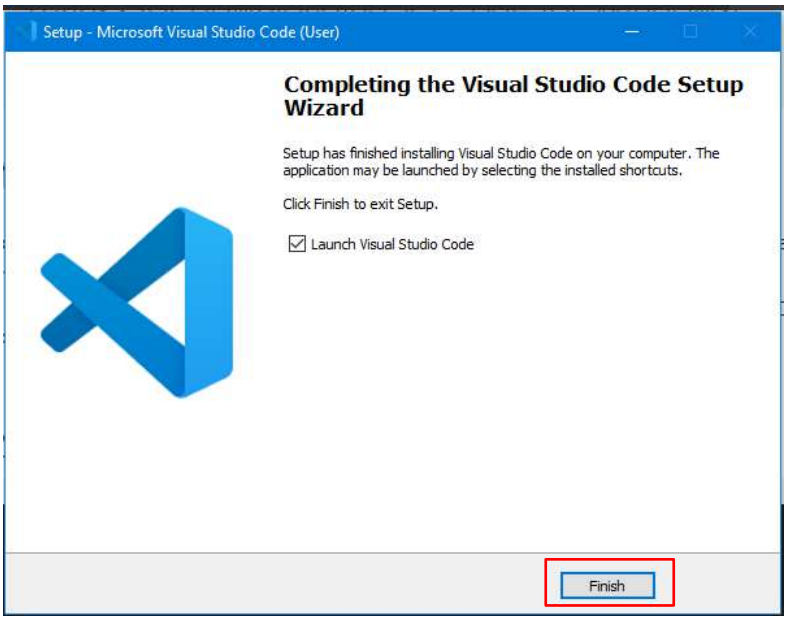
Step 6: Enable all the checkboxes and click on **'Next'**.



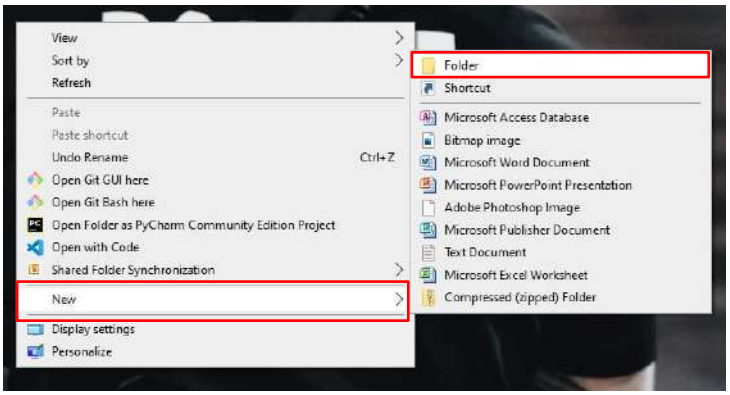
Step 7: Now click on **'Install'**.



Step 8: Enable the **'Launch Visual Studio Code'** & click on **'Finish'**.



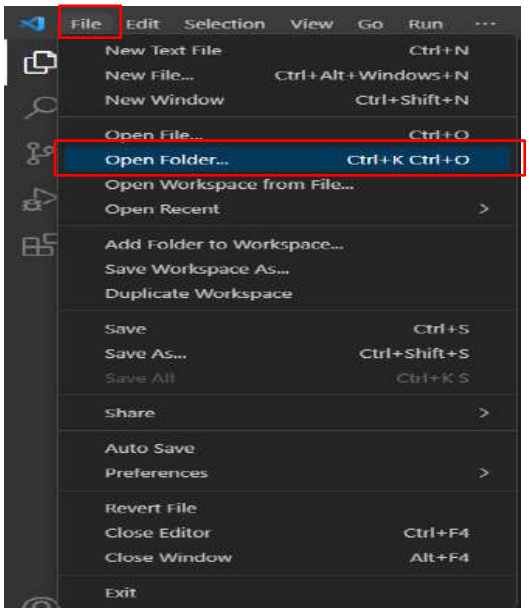
Step 9: Go to Desktop □ Right click □ New □ Folder □ Folder name.

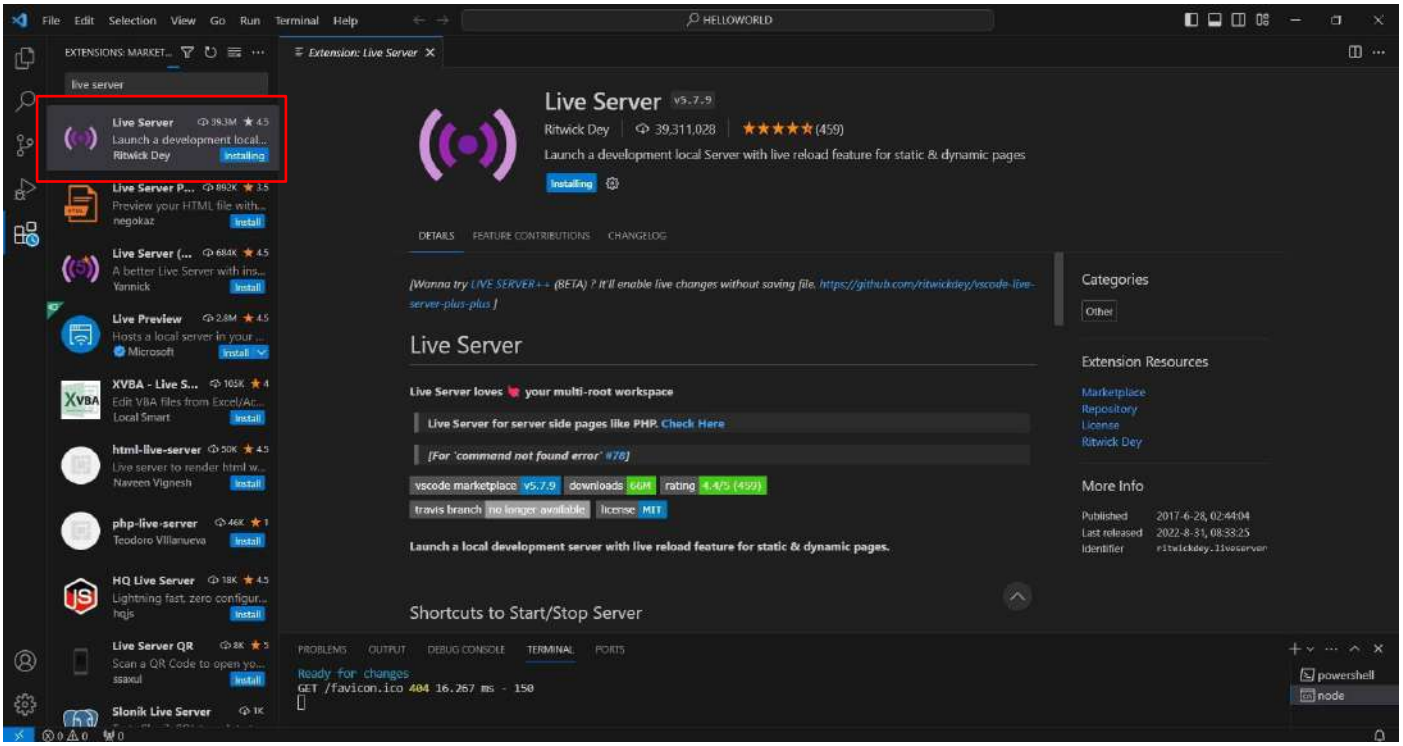


Step 10: Launch the VS Code □ click on **File** □ Open Folder □ Select the newly created folder.

- **<u>VS Code extensions</u>**

Step 11: Click on Extensions logo and type for **'Live Server'**. Then click on **'Install'** to install the server.





- **Create a simple form like Registration form, feedback form, after submit hide create form and enable the display section.**

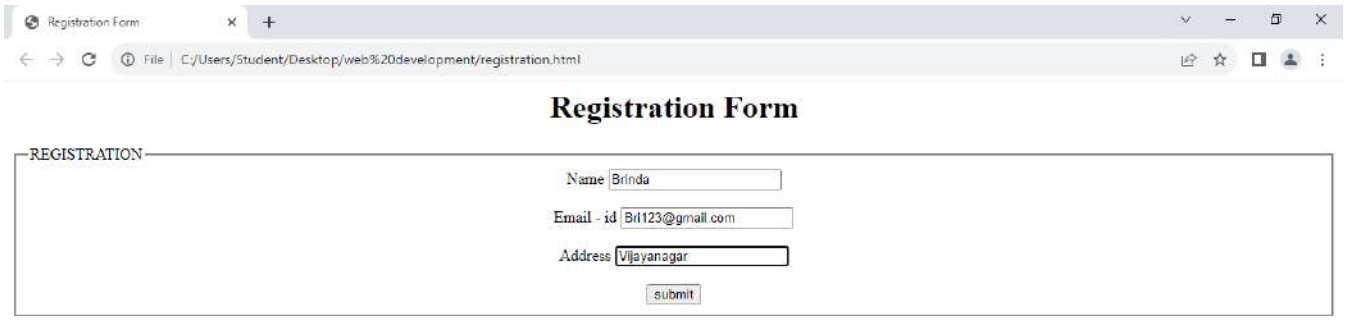
Registration Form:

```
Registration.html > 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><center>
        <h1>Registration 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()"/></center>
            </fieldset>
        </form>
    </body>
</html>
```

Details Form:

```html
<!DOCTYPE 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>
```

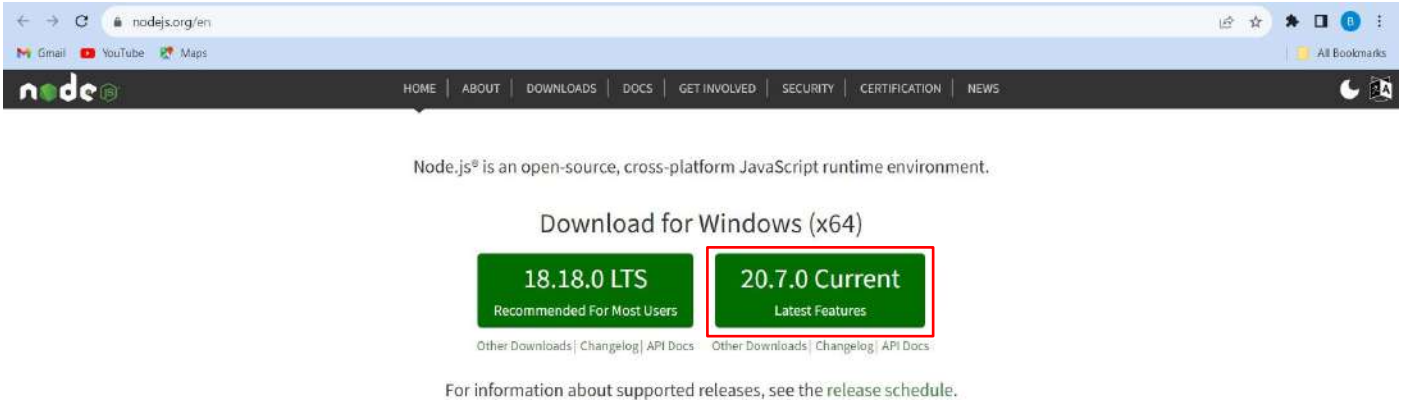
Final Output:



2. Create and run simple program in Type Script.

- **Installation of Typescript using Node.js Package Manager (npm).**

Step 1: Go to web browser & search for Node.js. Then click on ‘Download’ and download the latest version.

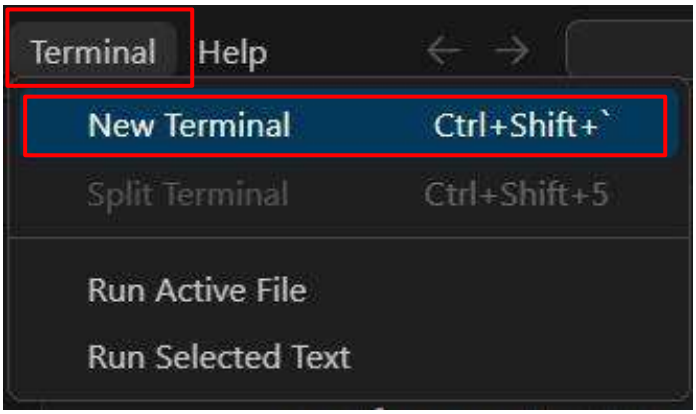


Step 2: Now enter the below commands in the Command Prompt to check the correct installation of Typescript.

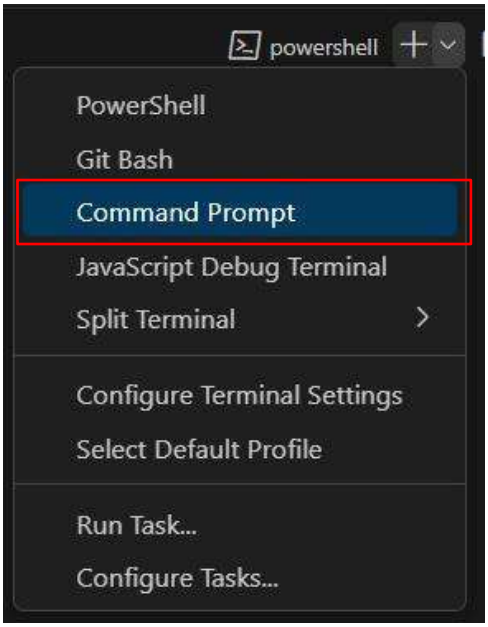
```
Microsoft Windows [Version 10.0.22000.2295]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hk923>node -v
v20.6.1

C:\Users\hk923>npm install typescript --save.dev

up to date, audited 2 packages in 3s

found 0 vulnerabilities

C:\Users\hk923>npm install typescript -g

changed 1 package in 1s

C:\Users\hk923>npm install typescript@latest -g

changed 1 package in 1s

C:\Users\hk923>tsc -v
Version 5.2.2

C:\Users\hk923>npm install -g live-server
[                ] / idealTree:send: sill placeDep node_modules/live-server ms@2.0.0 OK for: debug@2.6.9 want: 2.0.0
```

- **<u>Installation of typescript inside the VS code live server.</u>**

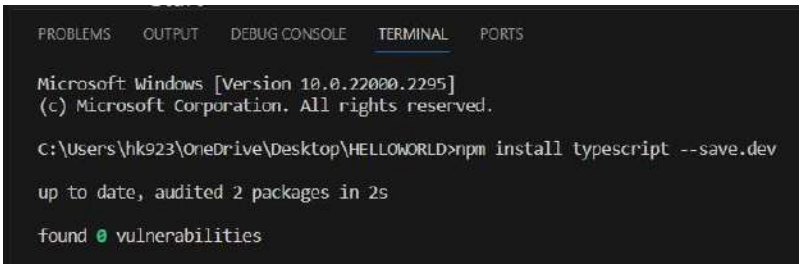
Step 3: Launch the VS code Terminal New Terminal.



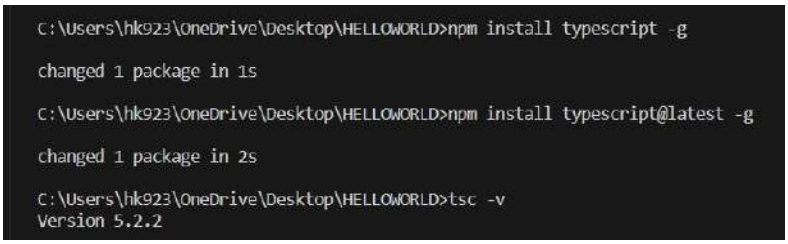
Step 4: Select the Terminal to '**Command Prompt**'.



Step 5: Enter **'npm install typescript –save.dev'** to install the typescript dependencies.

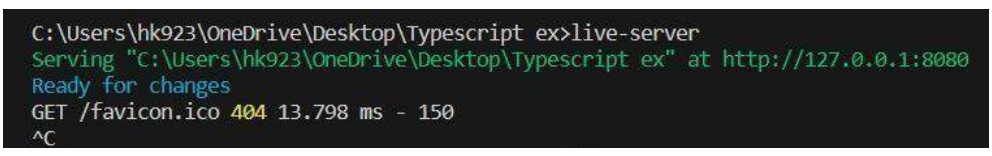


Step 6: Now enter **'npm install typescript -g'** & **'npm install typescript@latest -g'.**

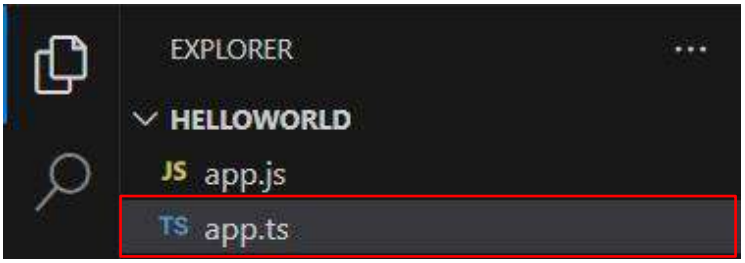


Step 7: Then enter **'npm install -g live-server'** to install the Live Server.

Step 8: Enter **'live-server'** to run the Live Server.



Step 9: Create a new empty Folder in the Desktop & open that folder in the VS Code. Then create a new file named **'app.ts'** & enter the below codes.



Step 10: Go back to the Terminal and enter '**tsc app.js**'.

```
C:\Users\hk923\OneDrive\Desktop\HELLOWORLD>tsc app.ts
```

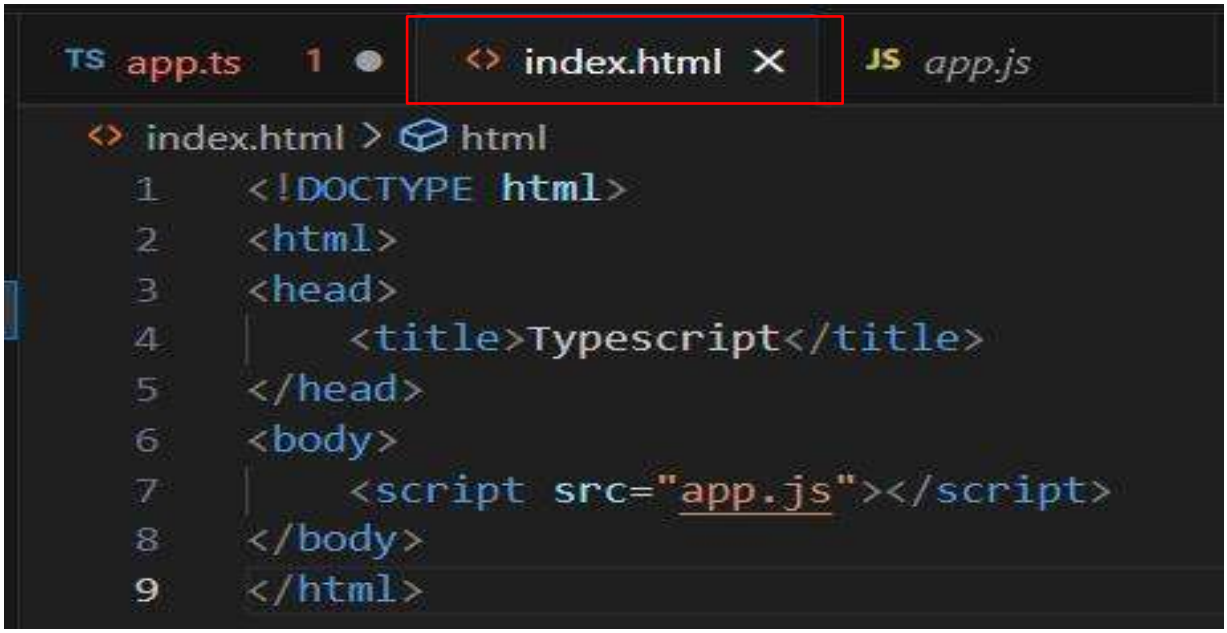
Step 11: Now we can see a new file called **'app.js'** has been generated by the TypeScript Compiler.

Step 12: Use the command **'node app.js'** to run that file.

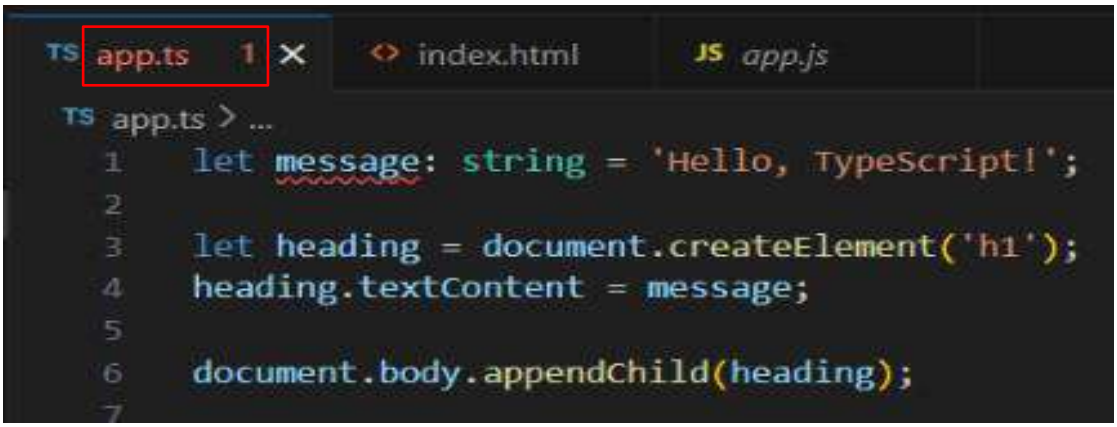
```
C:\Users\hk923\OneDrive\Desktop\HELLOWORLD>node app.js
Hello, World!
```

- **<u>Usage of TypeScript & Live Server to run the 'Hello World' program.</u>**

Step 13: Now create a new file named **'index.html'** & add the **'app.js'** as shown below.



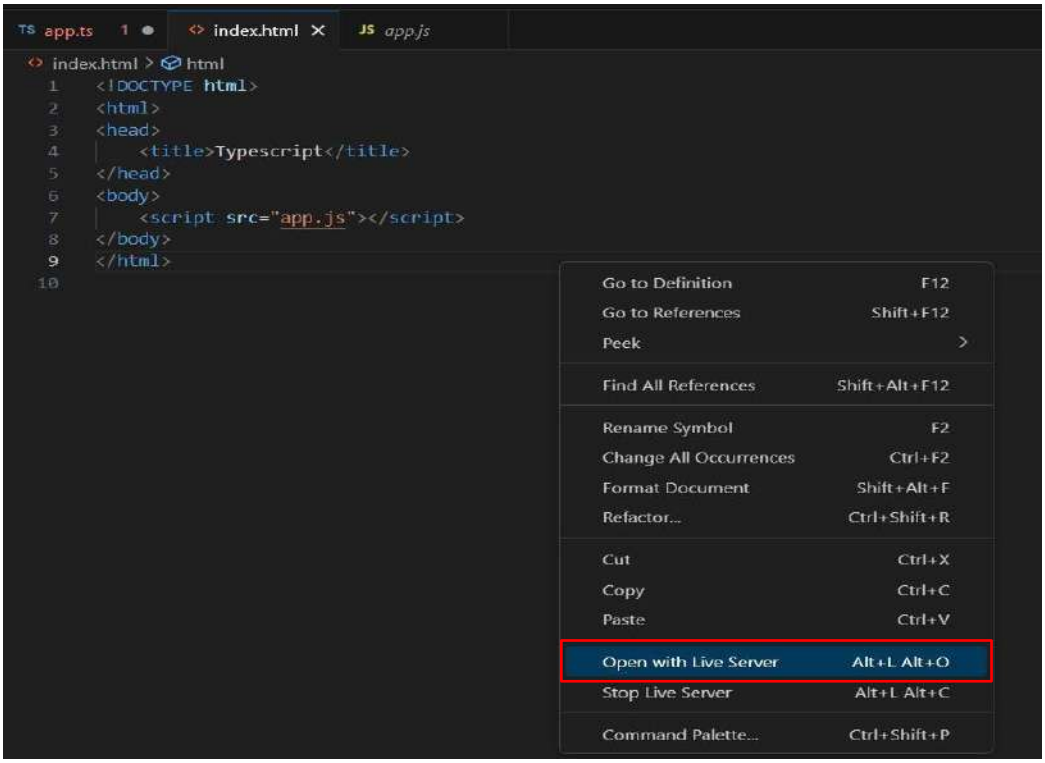
Step 14: Then make changes to the **'app.ts'** code as shown below.



Step 15: Enter 'tsc app.ts' to run the compiler.

```
C:\Users\hk923\OneDrive\Desktop\HELLOWORLD>tsc app.ts
```

Step 16: Now go to 'index.html' file □ Right click on it □ Select **Open with Live Server**.



Output:

3. Suitable cases to be used to learn and implement program constructs.

A. Login.html

app1 > public > login.html

```html
<!DOCTYPE html>
<html>
<head>
  <title>Login Page</title>
</head>
<body>
  <h1>Login</h1>
  <form id="login-form">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required><br>
    <input type="submit" value="Login">
  </form>
  <p id="message"></p>
  <script src="login.js"></script>
</body>
</html>
```

B. Login.js

app1 > src > login.js

```js
document.getElementById("login-form").addEventListener("submit", function (event) {
    event.preventDefault();

    // Get user input
    var username = document.getElementById("username").value;
    var password = document.getElementById("password").value;

    // Load JSON data
    fetch('login.json')
      .then(response => response.json())
      .then(data => {
        const users = data.users;
        const user = users.find(u => u.username === username && u.password === password);

        if (user) {
          document.getElementById("message").textContent = "Login successful!";
        } else {
          document.getElementById("message").textContent = "Invalid username or password.";
        }
      })
      .catch(error => console.error('Error loading JSON: ', error));
});
```

C. Login.json

app1 > src > {} login.json > ...

```
{
    "users": [
        {
            "username": "user1",
            "password": "password1"
        },
        {
            "username": "user2",
            "password": "password2"
        }
    ]
}
```

Output:

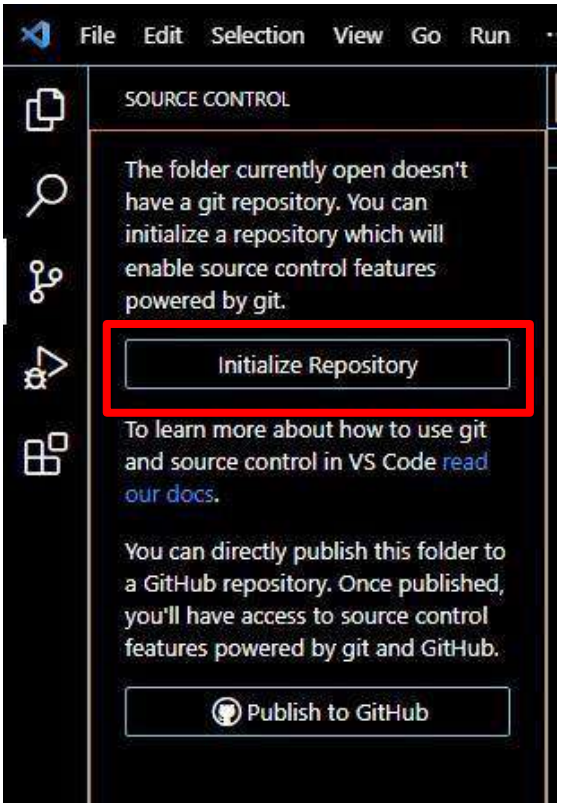
Please leave some space here while writing in record.

And continue from here.

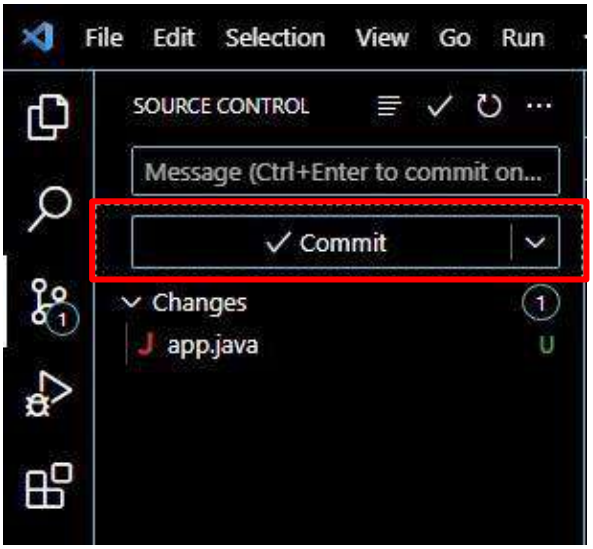
4. Creating a repository in Github and cloning the repository in Vs code.

Step 1: Create an empty folder in the Desktop and open that folder via VS Code. Then create a new file named **'app.java'** & add some code to it.

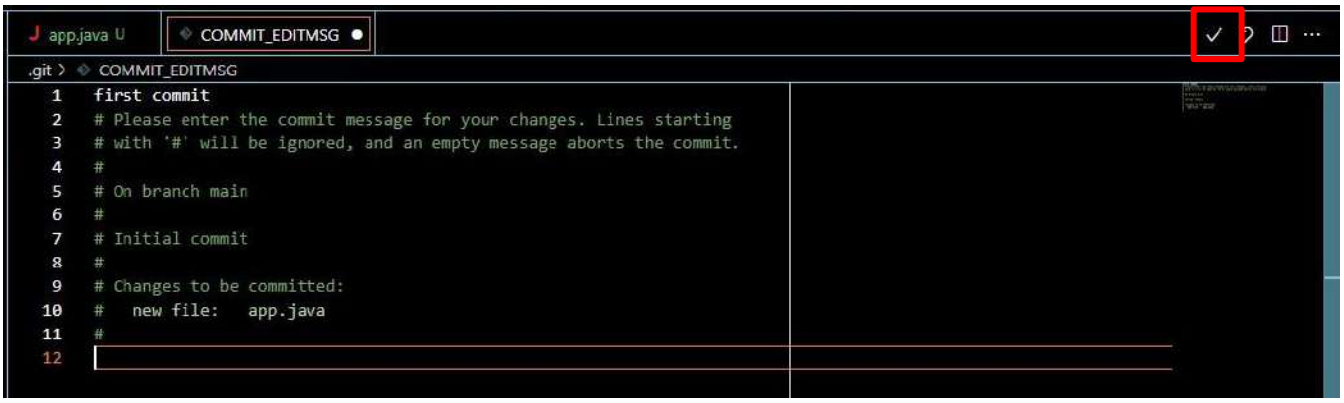
Step 2: Go to Extensions Initialize Repository.



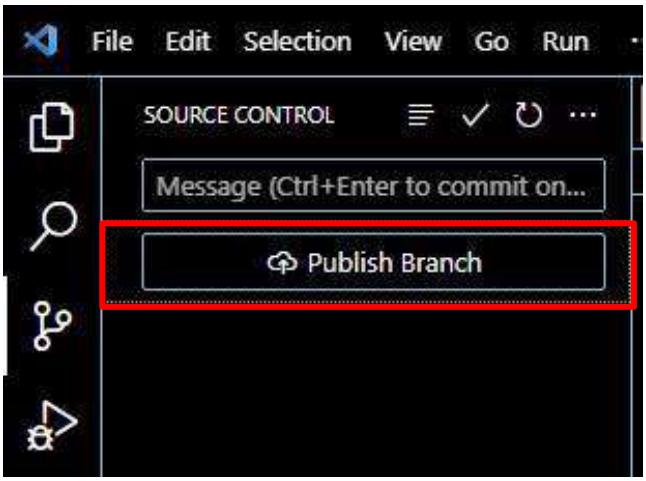
Step 3: Now click on **'Commit'**.



Step 4: Enter a command as **'first commit'** & Save it.

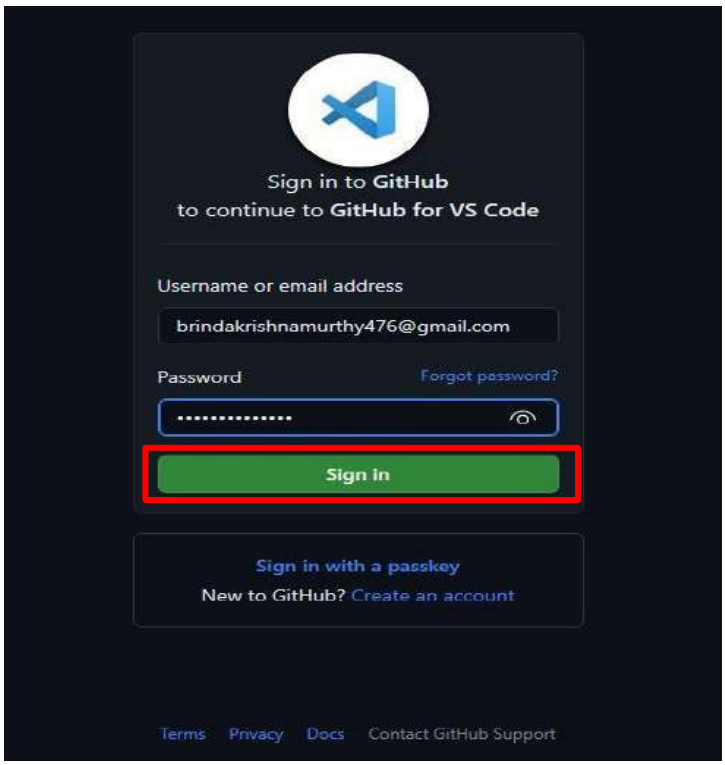


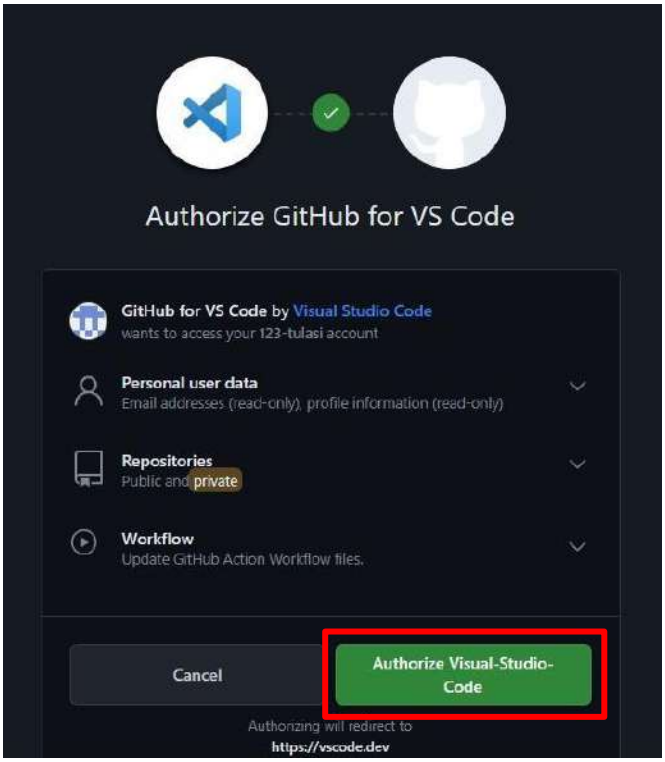
Step 5: Then click on **'Publish Branch'**.



Step 6: Now sign-in into your GitHub account and click on **'Sign in'**.

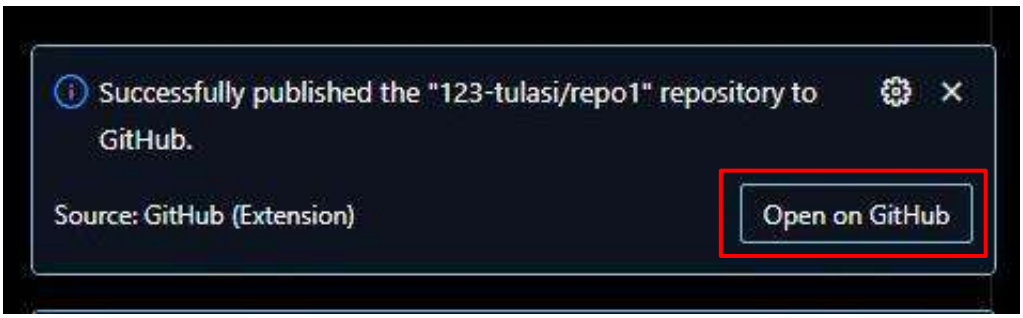
Step 7: then click on **'Authorize Visual-Studio-Code'**.



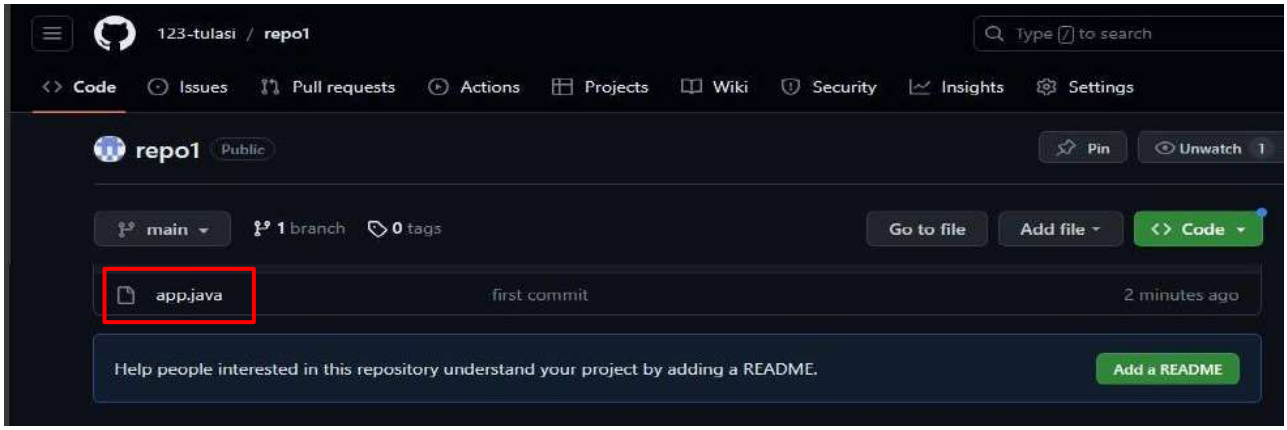


Step 8: Select **'Publish to GiHub public repository'**.

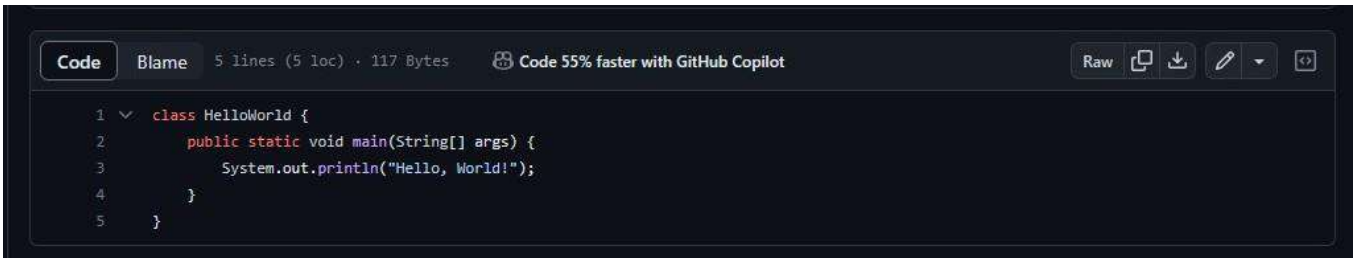
Step 9: Click on **'Open on GitHub'**.



Step 10: Now you can see the newly file named 'app.java' here.

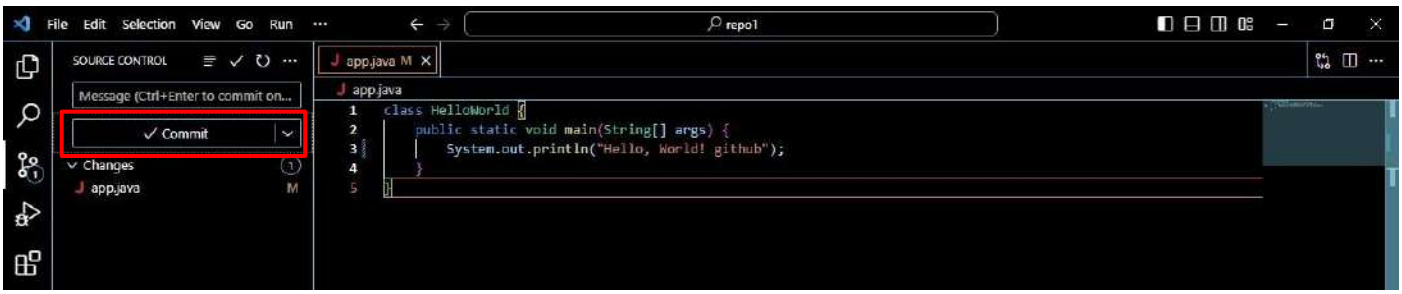


Step 11: The code is made visible here.

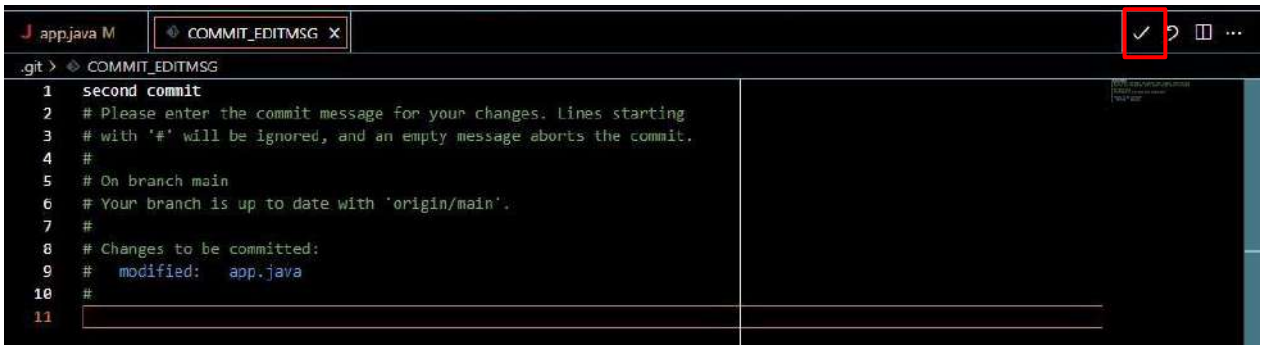


- **<u>Synchronising Changes to the code.</u>**

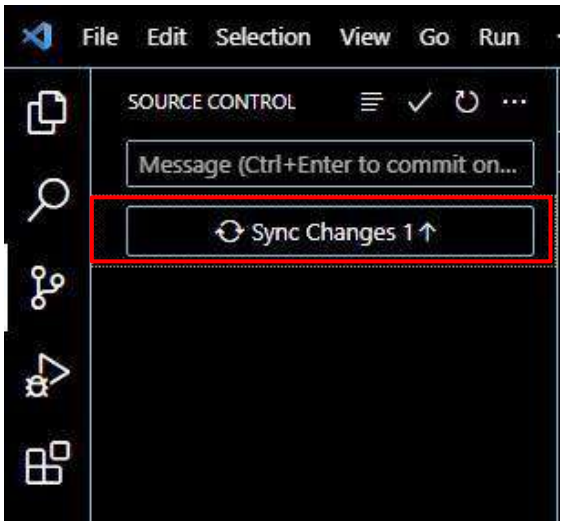
Step 12: Do some changes to your code & click on **'Commit'**.



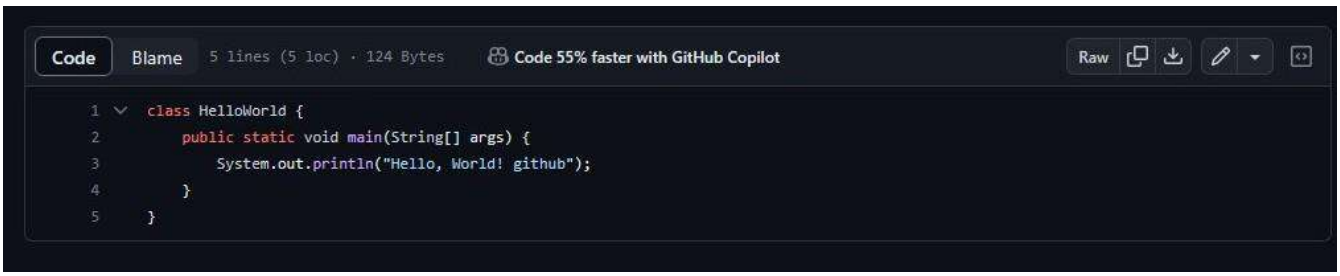
Step 13: Type a command as **'second commit'** & Save it.



Step 14: Click on **'Sync Changes'**.

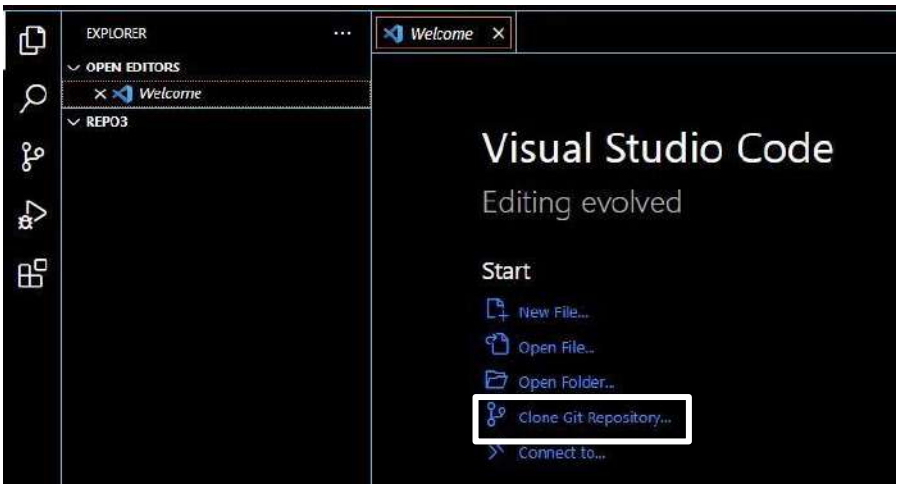


Step 15: The changes made are being displayed here.

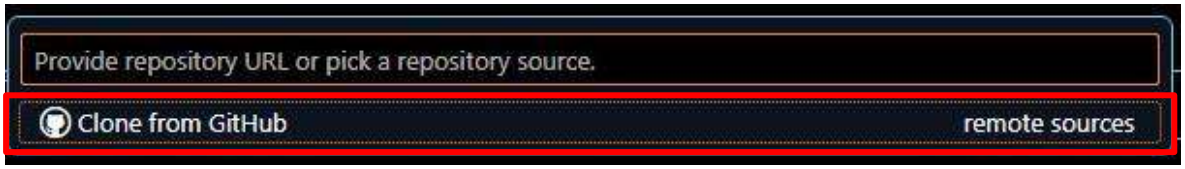


- **<u>Cloning an empty repository from VS Code to GitHub.</u>**

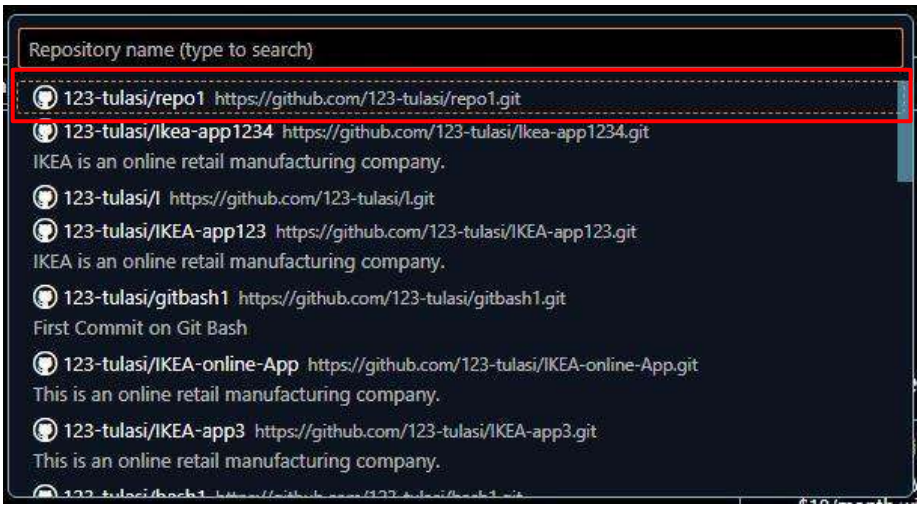
Step 16: Click on **'Clone Git Repository'**.



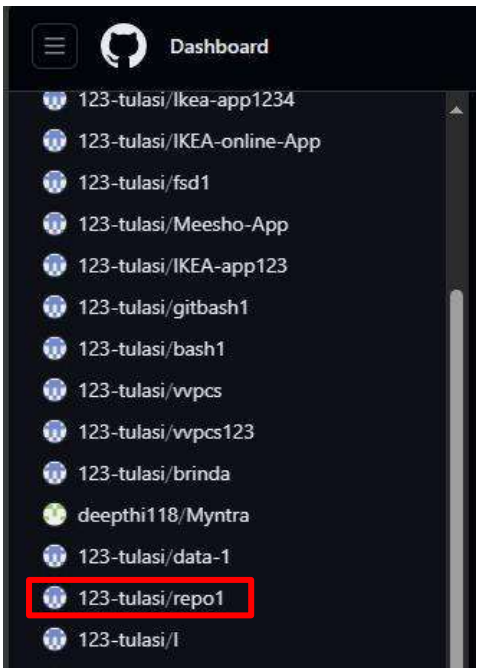
Step 17: Now click on **'Clone from GitHub'**.



Step 18: Select a repository you want to Clone.



Step 19: Now open the selected repository in your GitHub Repository.

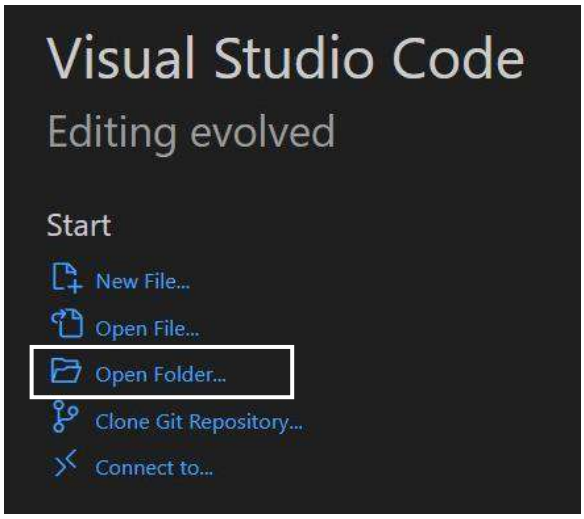


Step 20: Here, I've selected the already created repository & displaying the simple HelloWorld java program as shown below.

WEEK – 5

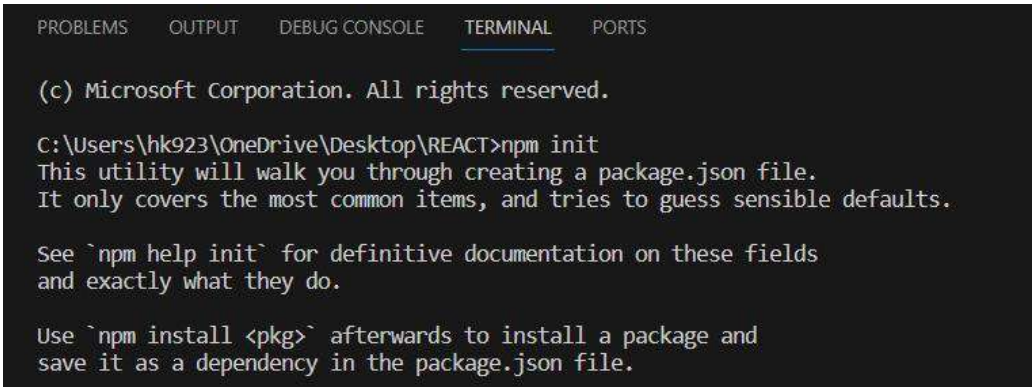
1. Creating and running a React.js app using VS Code.

Step 1: Create a new folder in the Desktop & open the folder in Visual Studio Code.

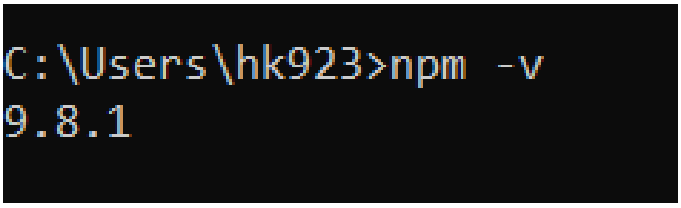


Step 2: Now go to Ternimal □ New Terminal □ Select Command Prompt.

Step 3: Enter the command **'npm init'** to install a new package.



Step 4: Enter **'npm -v'** to check the installation of Node Package Manager current version on your Windows **cmd**.



Step 5: For launching, make sure to check the installation of Node.js by entering the command **'node -v'** to check the current version installed.

```
C:\Users\hk923\OneDrive\Desktop\REACT>node -v
v20.6.1
```

- **Folder structure for React App:**

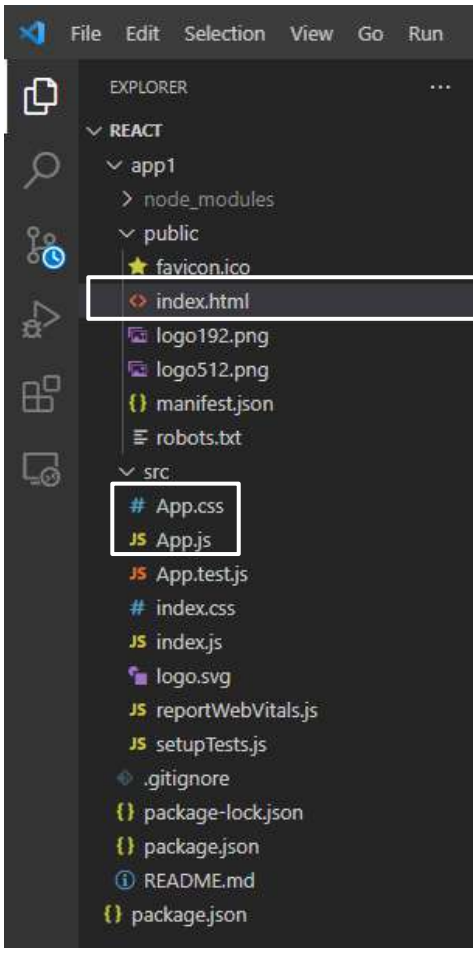
Step 6: Enter the command **'npx create-react-app'** to create a React Project & give the folder name as **app1**.

```
C:\Users\hk923\OneDrive\Desktop\REACT>npx create-react-app app1

Creating a new React app in C:\Users\hk923\OneDrive\Desktop\REACT\app1.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

[########.........] | idealTree:@babel/core: sill placeDep ROOT semver@7.5.4 OK for: react-scripts@5.0.1 want: ^7.3.5
```

Step 7: Enter the command **'cd app1'** to change the working directory.

```
C:\Users\hk923\OneDrive\Desktop\REACT>cd app1
```

Step 8: Now select the **'index.html'** from **public** & Select the **'app.js'** and **'app.css'** file from the **src** folder on the right side of your VS Code Window.

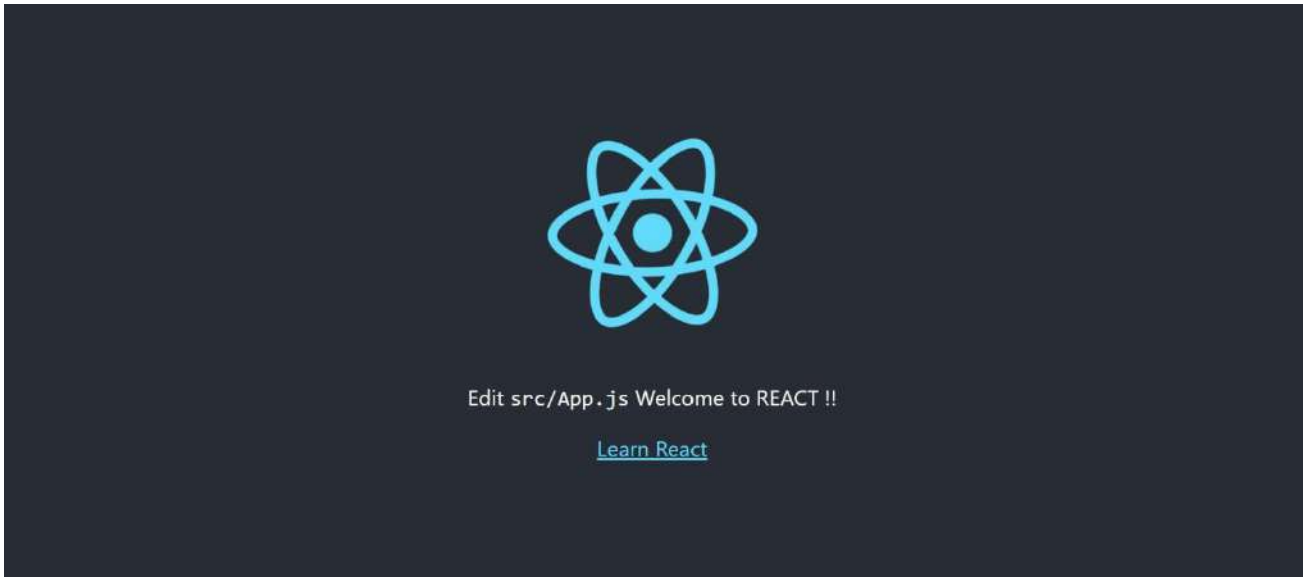


- **Launching of React.js app:**

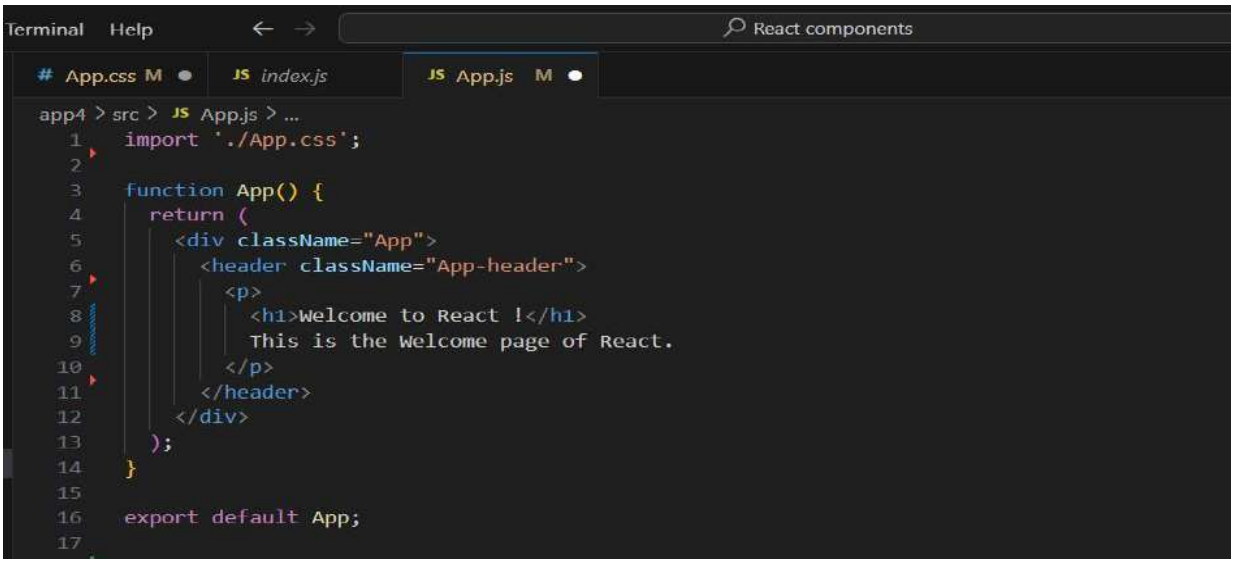
Step 9: Enter the command **'npm start app1'** to run the react app.

```
C:\Users\hk923\OneDrive\Desktop\REACT\app1>npm start app1

> app1@0.1.0 start
> react-scripts start app1
```

Output:



- **Creating your own changes to the code.**

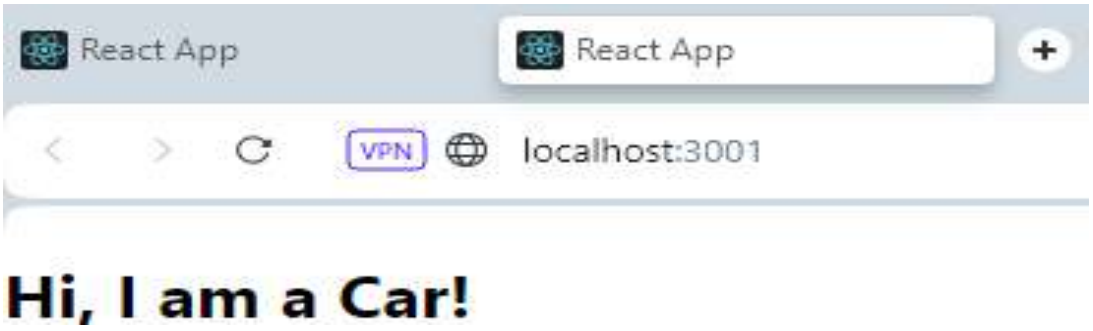


```
Terminal  Help                                   React components
# App.css M ●   JS index.js        JS App.js  M ●
app4 > src > # App.css > ...
1   .App {
2     text-align: center;
3     font-style: italic;
4     font-family: 'Times New Roman', Times, serif;
5   }
6
7   .App-header {
8     background-color: #08090b;
9     min-height: 100vh;
10    display: flex;
11    flex-direction: column;
12    align-items: center;
13    justify-content: center;
14    font-size: calc(10px + 2vmin);
15    color: white;
16  }
17
```

2. React Components

- **Functional Components:** Functional components are simply javascript functions. We can create a functional component in React by writing a javascript function. These functions may or may not receive data as parameters. The functional component is also known as a stateless component because they do not hold or manage state.

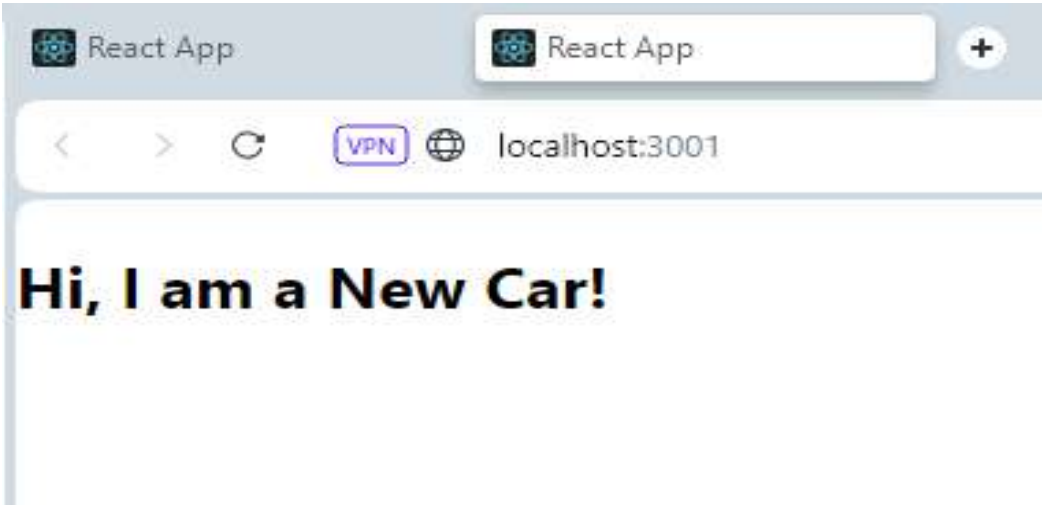
```
app3 > src > JS App.js > ...
1   import './App.css';
2
3   import React from 'react';
4   import ReactDOM from 'react-dom/client';
5
6   function App() {
7     return <h2>Hi, I am a Car!</h2>;
8   }
9
10  const root = ReactDOM.createRoot(document.getElementById('root'));
11  root.render(<App />);
12
13  export default App;
14
15
16
17
```



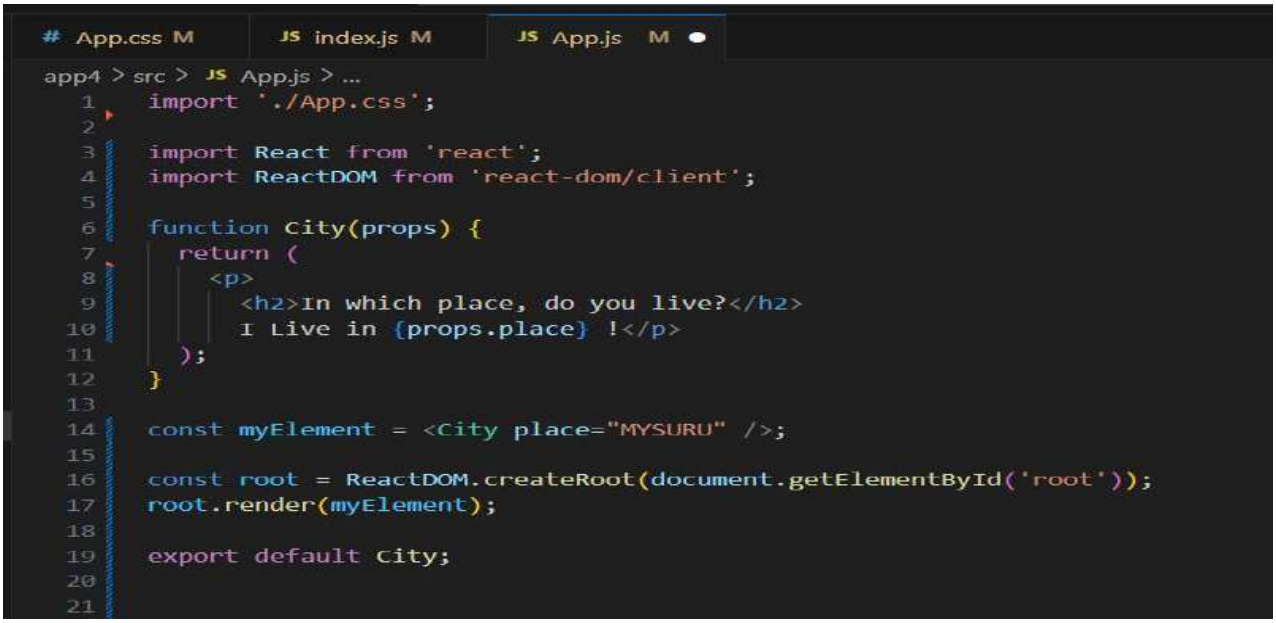
- **Class Component:** A class component are little more complex then functional component. A class component must include the extends React. Component statement. The class component is also known as a stateful component because they can hold or manage local state.

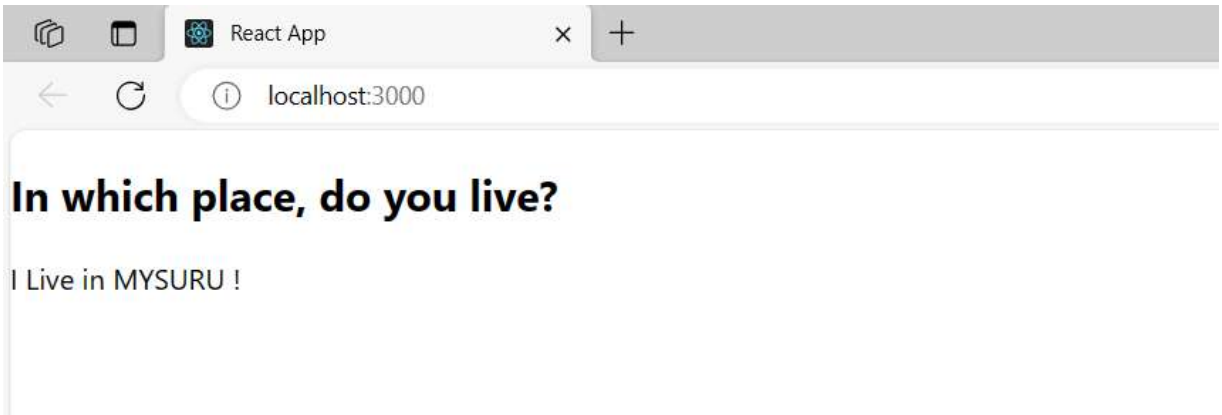
app3 > src > JS App.js > ...

```
1   import './App.css';
2
3   import React from 'react';
4   import ReactDOM from 'react-dom/client';
5
6   class App extends React.Component {
7     render() {
8       return <h2>Hi, I am a New Car!</h2>;
9     }
10  }
11
12  const container = document.getElementById('root');
13  const root = ReactDOM.createRoot(container);
14  root.render(<App />);
15
16  export default App;
17
18
19
```

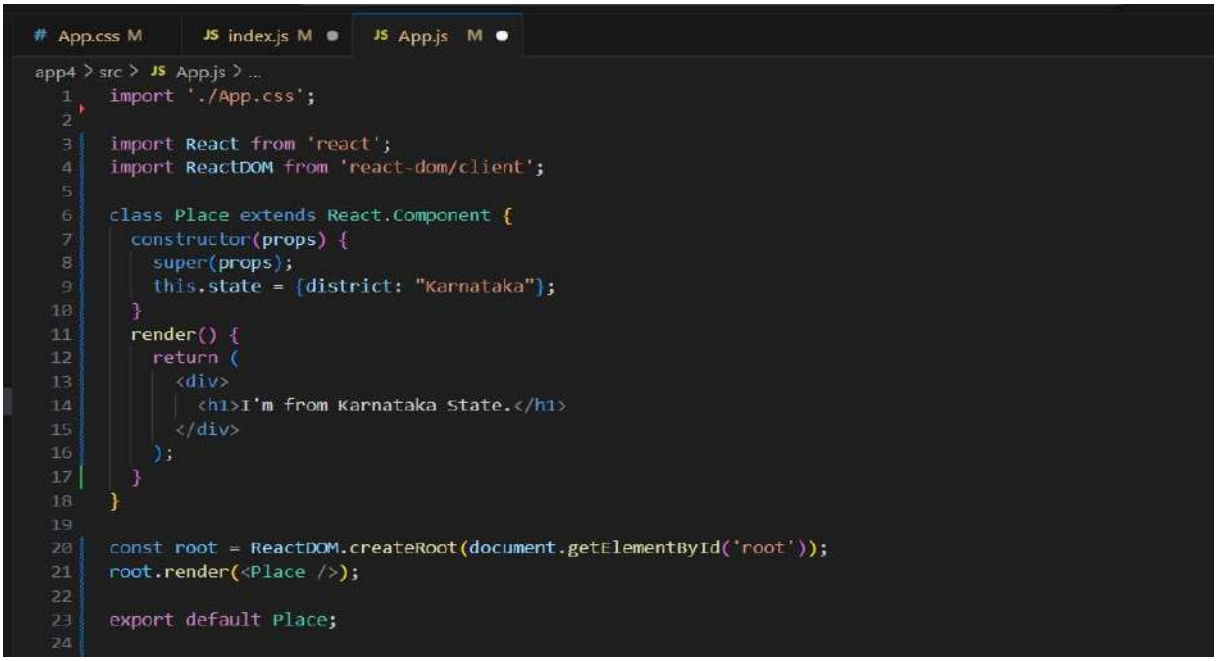


- **React Props:** Props is short for properties and they are used to pass data between React components. React's data flow between components is uni-directional (from parent to child only).



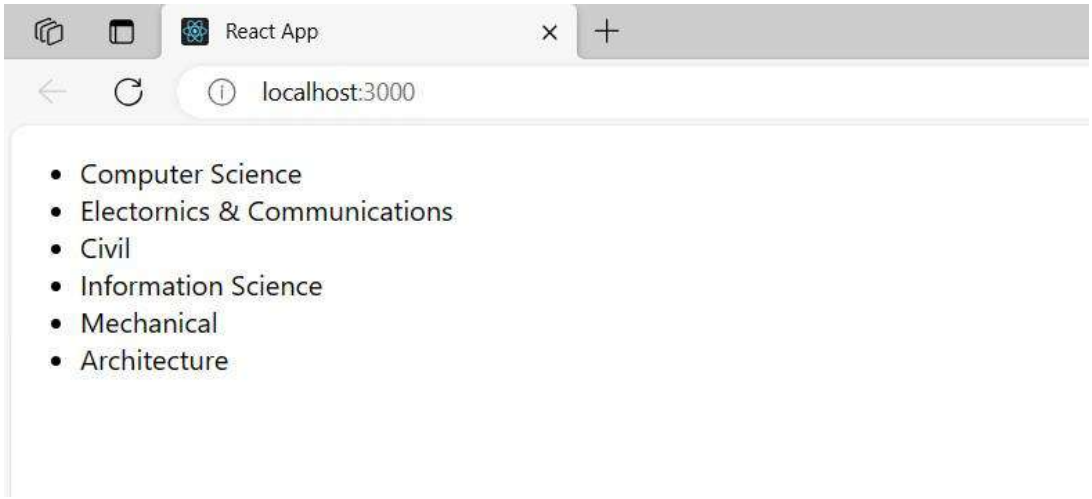


- **React States:** React has another special built-in object called state, which allows components to create and manage their own data. So unlike props, components cannot pass data with state, but they can create and manage it internally.



- **Render a List in React with Keys:** In the following React List example, we render a list of items that contain Course names and their respective id. We are using the **.map()** method to fetch the items from the course array, and every item has a unique key property. Keys are used in Reactto figure out how to update a list, be it adding, updating, or deleting an item in a list.

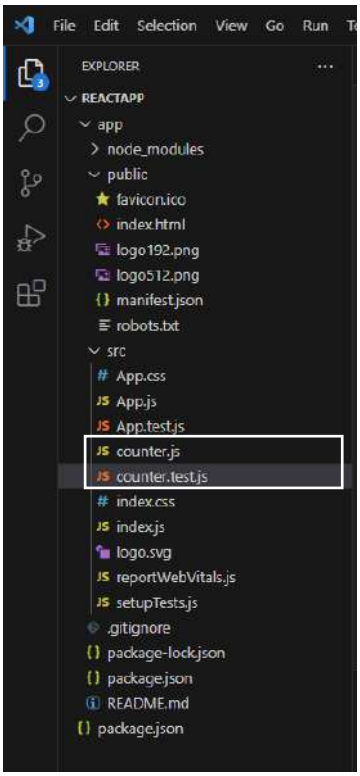
```
# App.css M    JS index.js    JS App.js M ×
app4 > src > JS App.js > default
1  import './App.css';
2
3  import React from 'react';
4  import ReactDOM from 'react-dom/client';
5
6  function App() {
7    const Movies = [
8      { id: 1, name: 'Computer Science' },
9      { id: 2, name: 'Electornics & Communications' },
10     { id: 3, name: 'Civil' },
11     { id: 4, name: 'Information Science' },
12     { id: 5, name: 'Mechanical' },
13     { id: 6, name: 'Architecture' },
14   ];
15   return (
16     <ul>
17       {Movies.map(data => (
18         <li key={data.id}> {data.name}</li>
19       ))}
20     </ul>
21   );
22 }
23 const root = ReactDOM.createRoot(document.getElementById('root'));
24 root.render(<App />);
25
26 export default App;
27
```



3. Testing React App using Jest

Step 1: Create a new react app **'npx create-react-app react-testing'**.

Step 2: Create a component called **'Counter.js'** in the **'src'** folder as shown below.



counter.js index.js counter.test.js

app > src > counter.js > ...

```
import React, { useState } from "react";
const Counter = () => {
const [counter, setCounter] = useState(0);
const incrementCounter = () => {
setCounter((prevCounter) => prevCounter + 1);
};
const decrementCounter = () => {
setCounter((prevCounter) => prevCounter - 1);
};
return (
<>
<button data-testid="increment" onClick={incrementCounter}>
+
</button>
<p data-testid="counter">{counter}</p>
<button data-testid="decrement" onClick={decrementCounter}>
-
</button>
</>
);
};
export default Counter;
```

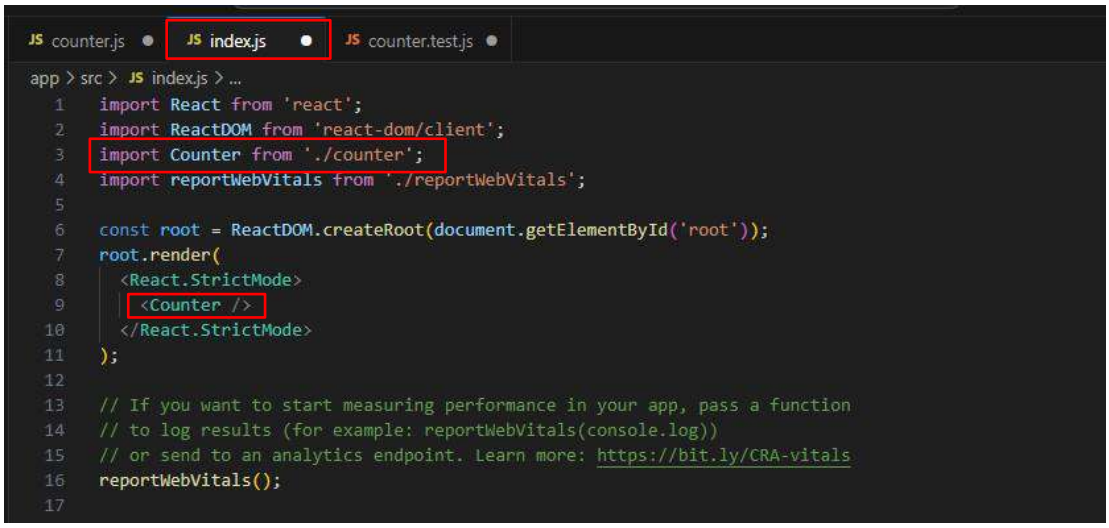
Step 3: Now open the **'counter.test.js'** file in the **'src'** folder as shown below.

counter.js index.js counter.test.js

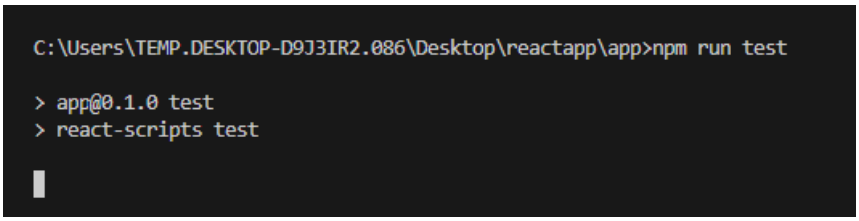
app > src > counter.test.js > ...

```
import { render, fireEvent, screen } from "@testing-library/react";
import Counter from "./counter";
//test block
test("increments counter", () => {
// render the component on virtual dom
render(<Counter />);
//select the elements you want to interact with
const counter = screen.getByTestId("counter");
const incrementBtn = screen.getByTestId("increment");
//interact with those elements
fireEvent.click(incrementBtn);
//assert the expected result
expect(counter).toHaveTextContent("1");
});
```

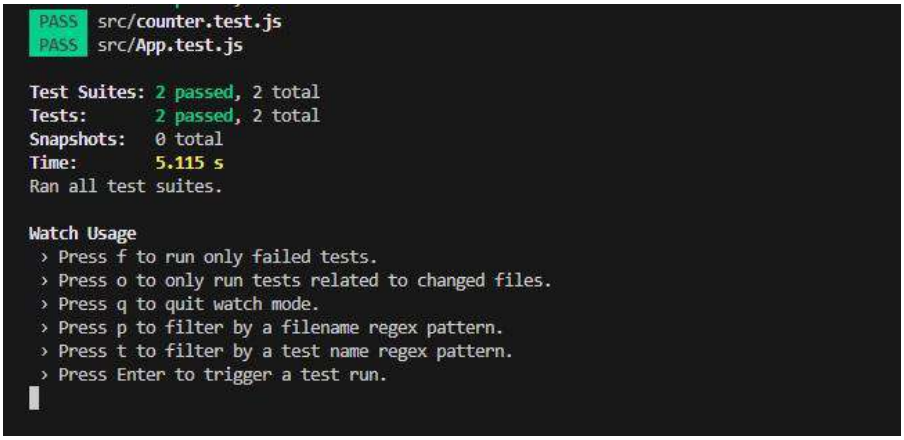
Step 4: Now open the ‘index.js’ file in the src folder and do the below changes as shown below.



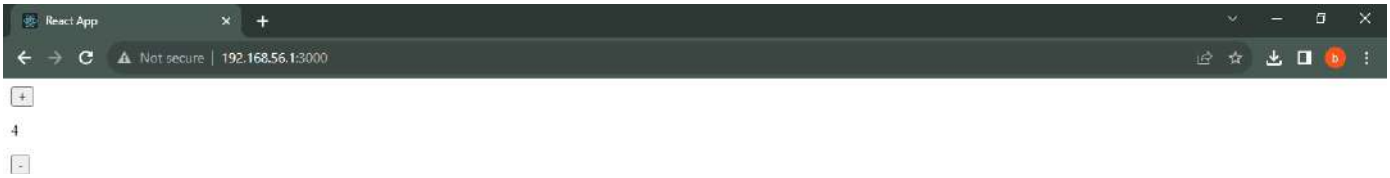
Step 5: Then go to Terminal □ New Terminal □ Select Command Prompt & give the command **‘npm run test’** to run the unit test file.



Output:

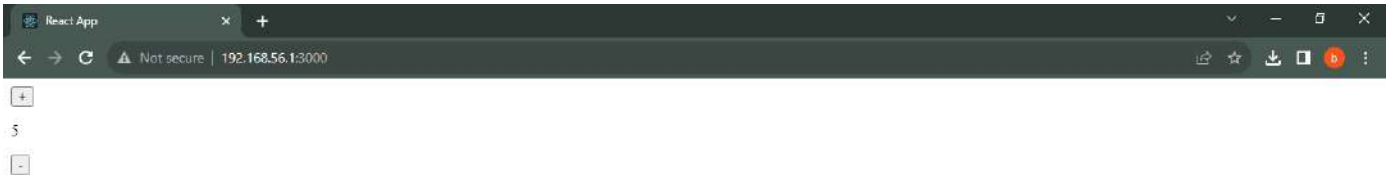


Note: Here is your simple React application



Decrement Value:

Increment Value:

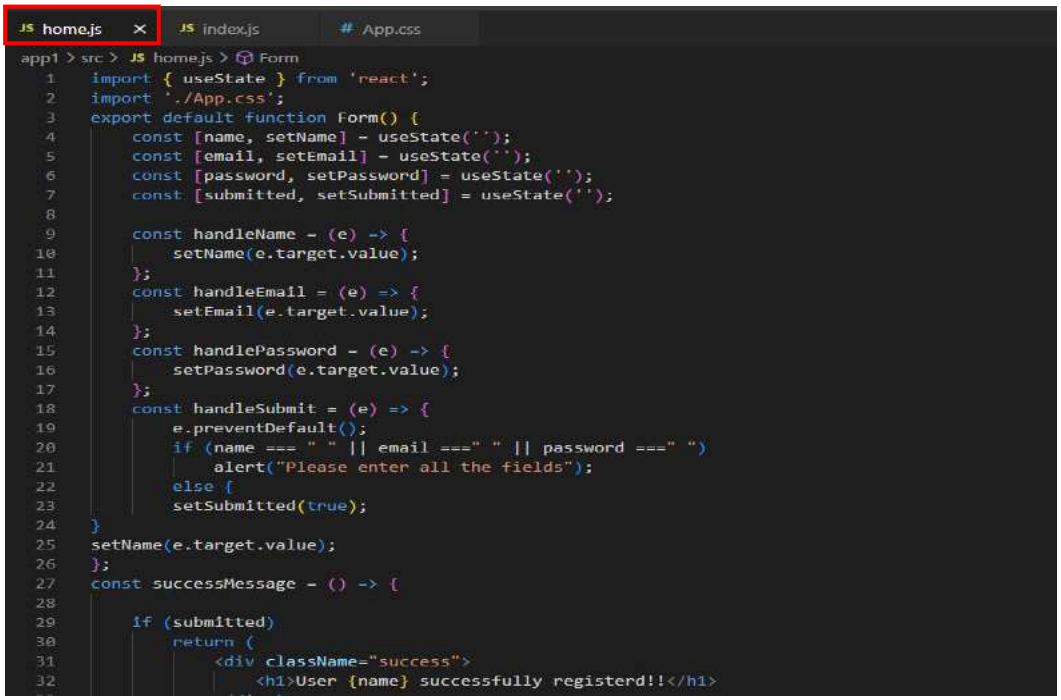


4. Testing single page application (Registration form) using React.

Note: Add **Home.js** file in **index.js** file inside the **'src'** folder.

Step 1: Create a new react app **'npx create-react-app app1'**.

Step 2: Now create a new file named **'home.js'** in the **'src'** folder and write the below code.



```
if (submitted)
    return (
        <div className="success">
            <h1>User {name} successfully registerd!!</h1>
        </div>);
};
return (
    <div className="form">
        <div>
            <h1>User Registration</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></
                <button onClick={handleSubmit} className="btn" type="submit">Submit
                </button>
            </fieldset>
        </form>
    </div>
);
}
```

Step 3: Now create a new file named **'App.css'** in the **'src'** folder to style your application & write the below code.

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

Step 4: Now create a new file named **'index.js'** in the **'src'** folder & do the changes in the code as shown below.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import Home from './home';
import reportWebVitals from './reportWebVitals';

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

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Output:


React App
localhost:3000
User Registration
Name
Brinda
Email
Brinda@gmail.com
Password
Submit


React App
localhost:3000
User Registration
User Brinda successfully registerd!!
Name
Brinda
Email
Brinda@gmail.com
Password
Submit

WEEK - 6

1. Implement navigation using react router.

Components in React Router:

There are two types of router components:

- **<BrowserRouter>:** It is used for handling the dynamic URL.
- **<HashRouter>:** It is used for handling the static request.

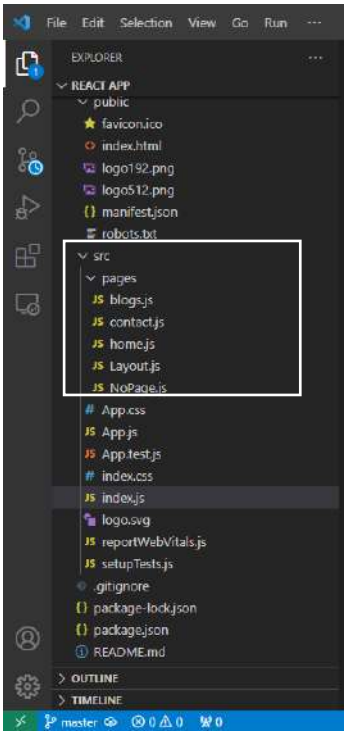
Step 1: Open the VS Code and launch the React on the local server.

- ## Add React Router:

To add React Router in your application, run this command **'npm i -D react-router-dom'** in the terminal from the root directory of the application. (Go to Terminal □ New Terminal □ Command Prompt).

Step 2: Open the folder in the VS Code and in the **'src'** folder, create a new folder named **'Pages'**.

Step 3: Now within the **'Pages'** folder, create the new files named **'blogs.js'**, **'contact.js'**, **'home.js'**, **'Layout.js'** & **'NoPage.js'** as shown below.



Step 4: Then add code inside the **'index.js'** file as shown here.

```
index.js
router > src > 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";
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 />);
```

Step 5: Now add code to **'App.css'** file to style your application.

```
index.js    App.css
router > src > App.css > li a:hover:not(.active)
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;
}
```

Step 6: Write the below code to **'blogs.js'** file inside the **'Pages'** folder.

```
index.js    blogs.js
router > src > pages > blogs.js > default
const Blogs = () => {
    return <h1>Blog Articles</h1>
};
export default Blogs;
```

Step 7: Write the below code to **'contact.js'** file inside the **'Pages'** folder.

```
index.js    contact.js
router > src > pages > contact.js > default
const contact =() => {
    return <h1>contact me</h1>
};
export default contact;
```

Step 8: Write the below code to **'home.js'** file inside the **'Pages'** folder.

index.js | home.js

router > src > pages > home.js > default

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

Step 9: Write the below code to **'Layout.js'** file inside the **'Pages'** folder.

index.js | Layout.js

router > src > pages > Layout.js > default

```
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;

```

Step 10: Write the below code to **'NoPages.js'** file inside the **'Pages'** folder.

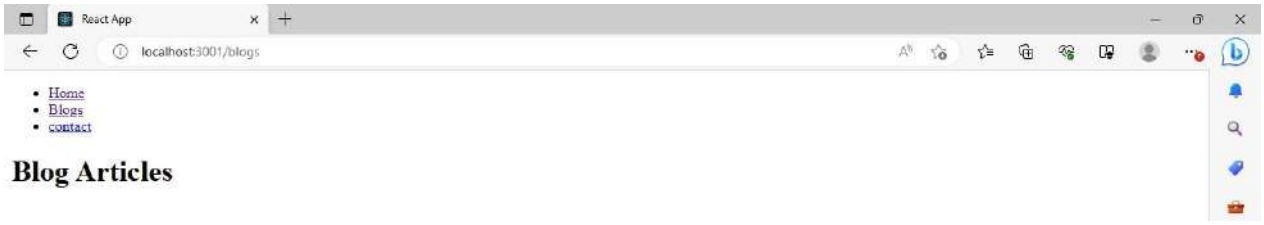
index.js | NoPage.js

router > src > pages > NoPage.js > default

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

Now run the **'index.js'** file to view the final output.

Output:



2. React Hooks-demonstrating useState, useEffect, useContext, useReducer.

React Hooks: Hooks (Receive or Steal) are the new feature introduced in the React 16.8 version. It allows you to use state and other React features without writing a class.

- **Types of Hooks in React:**

1) **UseState** - The React useState Hook allows us to track state in a function component. State generally refers to data or properties that need to be tracking in an application. To use the useState Hook, we first need to import it into our component.

   Syntax: import { useState } from "react";

2) **UseEffect** - The useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.

   Syntax: useEffect(<function>, <dependency>)

3) **UseContext** - React Context is a way to manage state globally. It can be used together with the useState Hook to share state between deeply nested components more easily than with useState alone. To create context, you must Import createContext and initialize it.

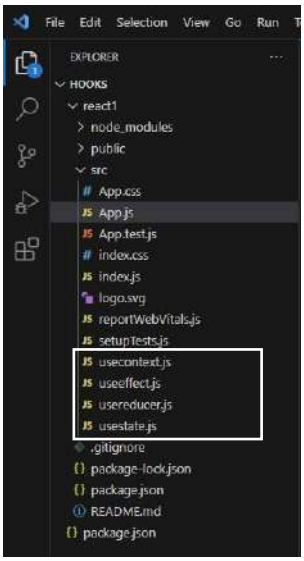
   Syntax: const myValue = useContext(MyContext);

4) **UseReducer** - The useReducer Hook is similar to the useState Hook. It allows for custom state logic.The useReducer Hook accepts two arguments.

   Syntax: useReducer(<reducer>, <initialState>)

Step 1: Open the VS Code and launch the React on the local server.

Step 2: To run the React Hooks, create the files (**usestate.js, useeffect.js, usecontext.js, usereducer.js**) as shown below in the **'src'** folder.



- **Example of UseState:**

Step 3: Write the below code in the **'App.js'** file inside the **'src'** folder.

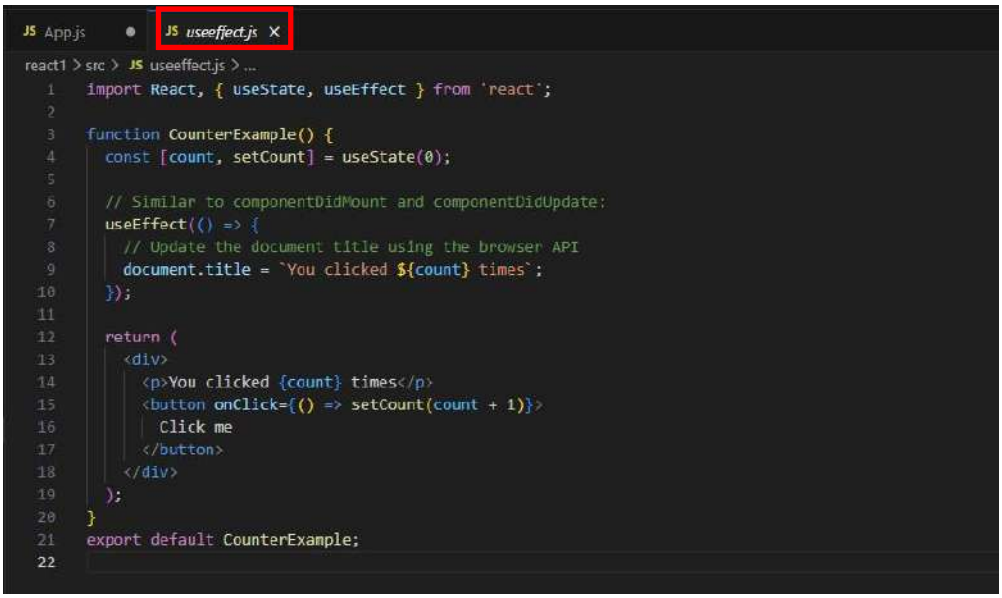
Step 4: Then add code in **'usestate.js'** file within the **'src'** folder.

Output:

- **Example of UseEffect:**

Step 5: Write the below code in the **'App.js'** file inside the **'src'** folder.

Step 6: Then add code in **'useeffect.js'** file within the **'src'** folder.

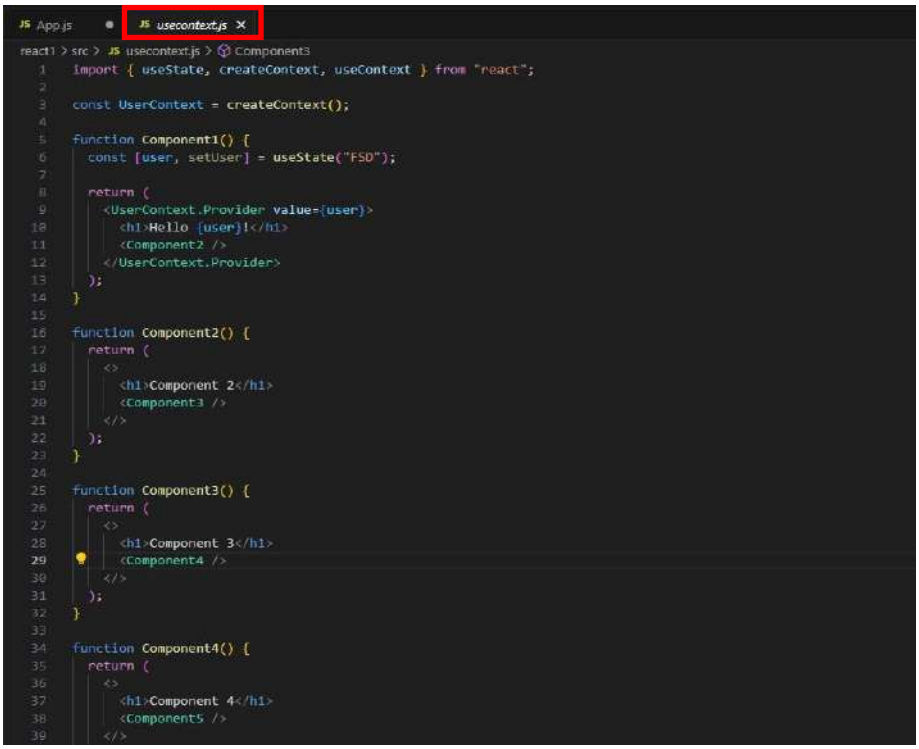


Output:

- **Example of UseContext:**

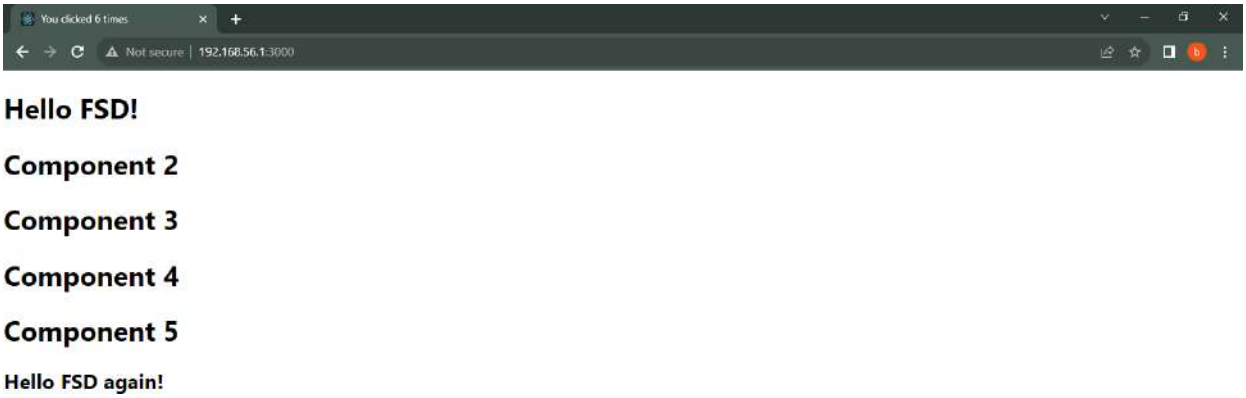
Step 7: Write the below code in the **'App.js'** file inside the **'src'** folder.

Step 8: Then add code in the **'usecontext.js'** file within the **'src'** folder.



```
App.js    usecontext.js
react1 > src > usecontext.js > Component3
38          <Component5 />
39        </>
40      );
41    }
42
43    function Component5() {
44      const user = useContext(UserContext);
45
46      return (
47        <>
48          <h1>Component 5</h1>
49          <h2>Hello {user} again!</h2>
50        </>
51      );
52    }
53
54    /*const root = ReactDOM.createRoot(document.getElementById('root'));
55    root.render(<Component1 />);*/
56    export default Component1;
57
58
```

Output:



- **Example of UseReducer:**

Step 9: Write the below code in the **'App.js'** file inside the **'src'** folder.

```
App.js    usereducer.js
react1 > src > App.js > ...
1   import AppReducer from './usereducer';
2
3   function App() {
4     return (
5       <div>
6
7       <AppReducer/>
8       </div>
9     );
10  }
11
12  export default App;
13
```

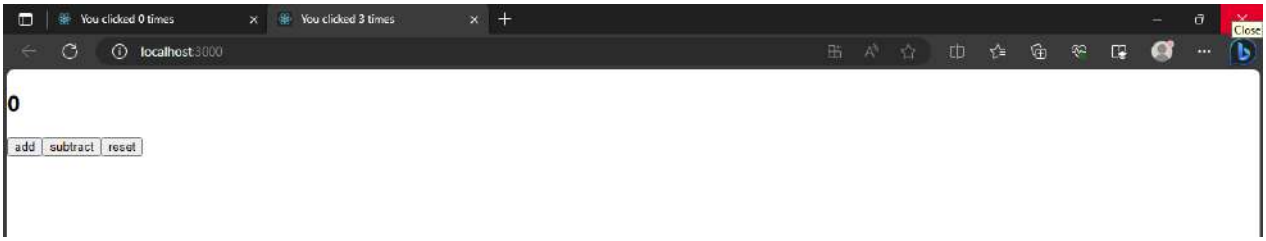
Step 10: Then add code in the **'usereducer.js'** file within the **'src'** folder.

App.js usereducer.js

react1 > src > usereducer.js > ...

```
import React, { useReducer } from "react";

// Defining the initial state and the reducer
const initialState = 0;
const reducer = (state, action) => {
  switch (action) {
    case "add":
      return state + 1;
    case "subtract":
      return state - 1;
    case "reset":
      return 0;
    default:
      throw new Error("Unexpected action");
  }
};

const AppReducer = () => {
    // Initialising useReducer hook
  const [count, dispatch] = useReducer(reducer, initialState);
  return (
    <div>
      <h2>{count}</h2>
      <button onClick={() => dispatch("add")}>
        add
      </button>
      <button onClick={() => dispatch("subtract")}>
        subtract
      </button>
      <button onClick={() => dispatch("reset")}>
        reset
      </button>
    </div>
  );
};
export default AppReducer;

```

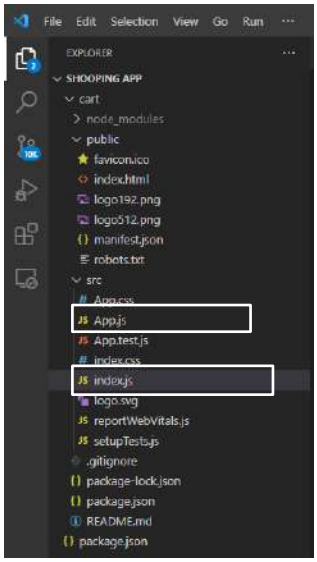
Output:



3. Build single page application - Shopping cart.

Step 1: Open the VS Code and launch the React on the local server.

- **React folder structure:**



Step 2: Add code to **'index.js'** file in the **'src'** folder.

App.js index.js

cart > src > index.js > ...

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

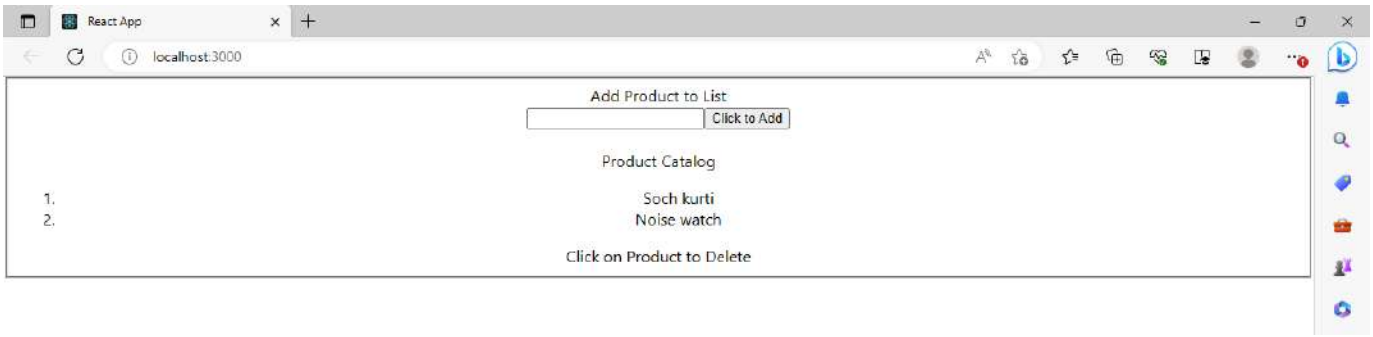
Step 3: Then write the below code in the **'App.js'** within the **'src'** folder.

App.js index.js

cart > src > App.js > ...

```
import { useState } from "react";
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;
```

Output:

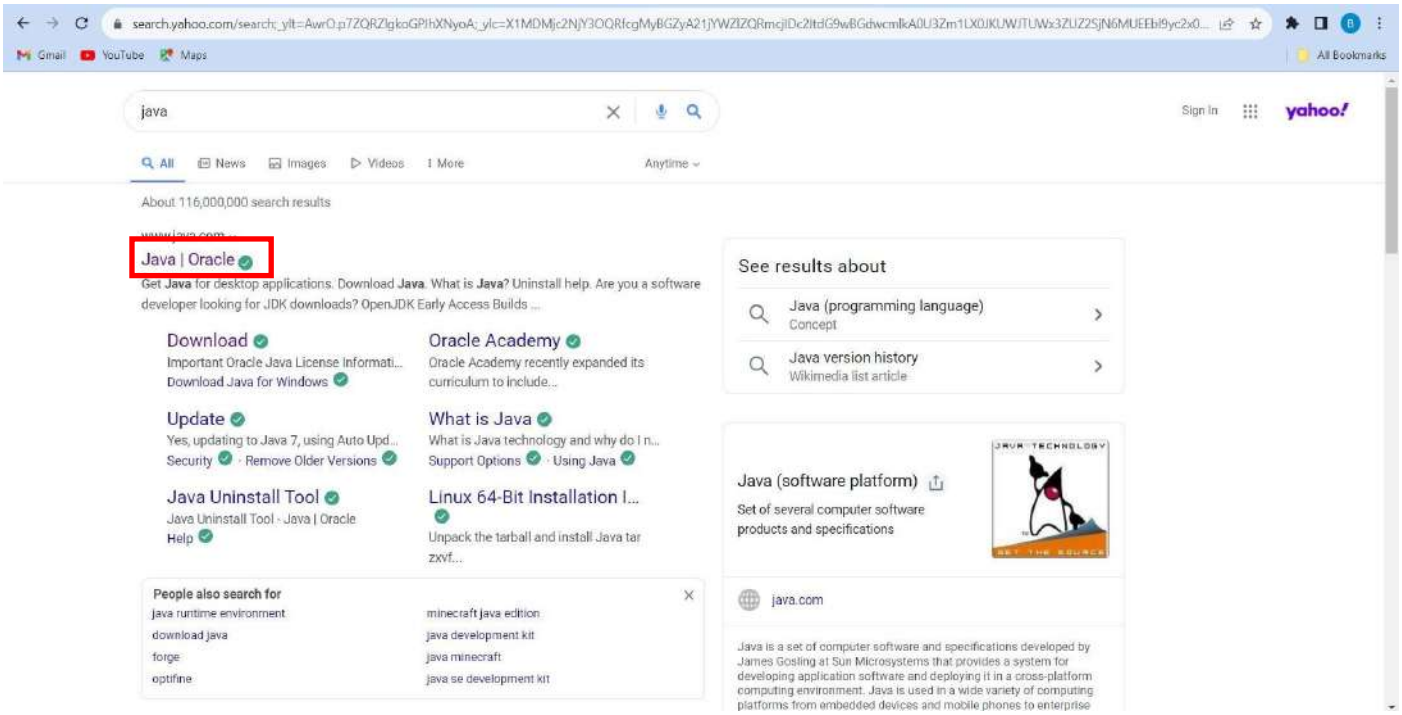


4. Setting up the environment and tools to install Java (latest stable version) and add environment variable to it.

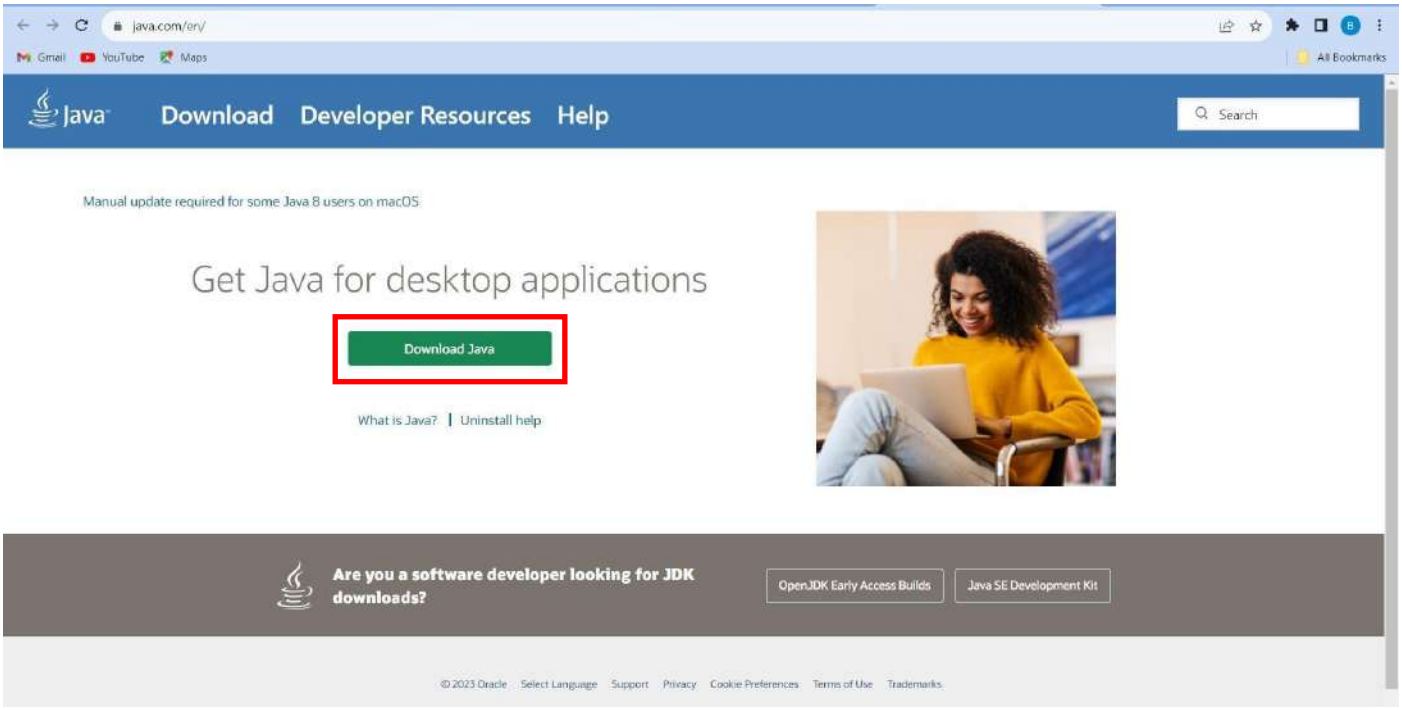
JAVA: Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers *write once, run anywhere* (WORA), meaning that compiled Java code can run on all platforms that support Java without the need to recompile.

Installation of Java on Windows:

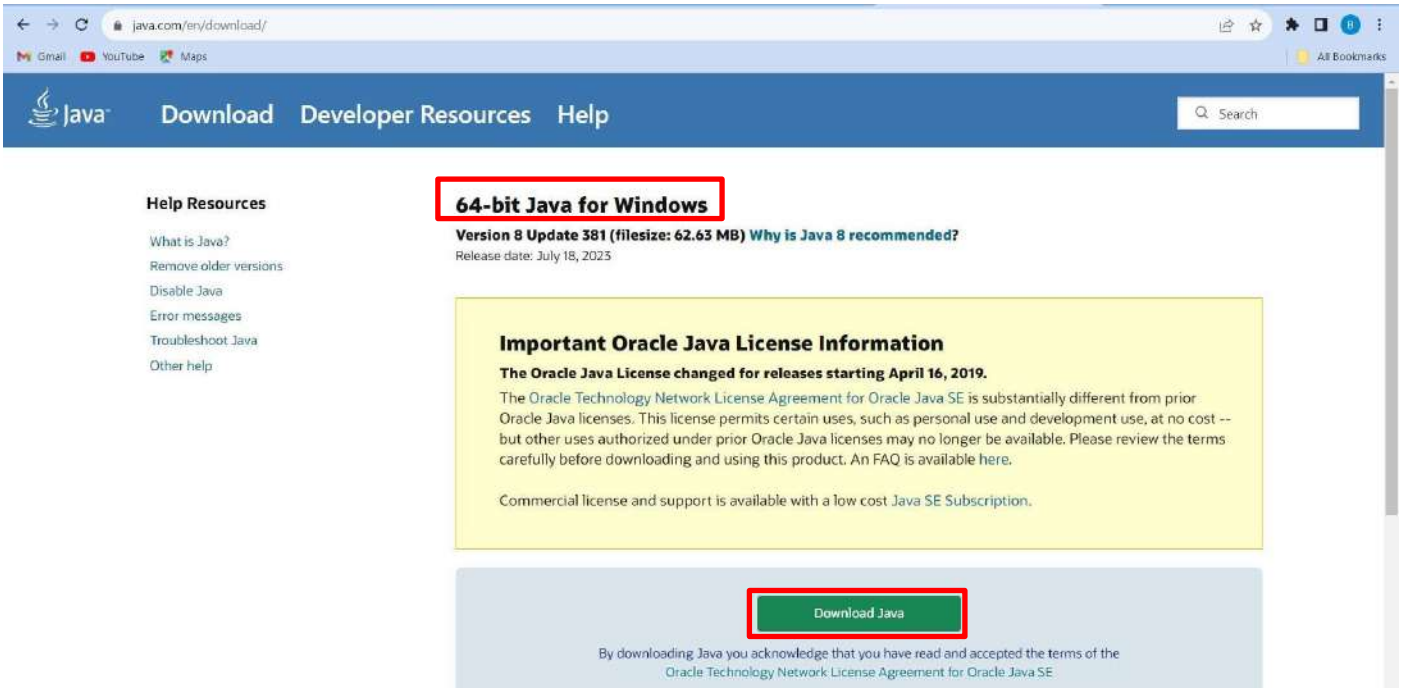
Step 1: Go to Web Browser → Search for Java → Click on the first link here.



Step 2: Now click on **'Download Java'**.



Step 3: Make sure to download the stable version for Windows. Here **64-bit** system type is recommended for download. Now click on **'Download Java'**.



Step 4: Open the downloaded file & click on **'Install'**.

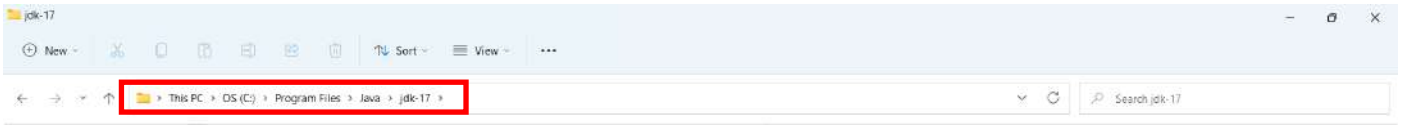
Step 5: Wait until the installation is completed

Step 6: Click on **'close'**.

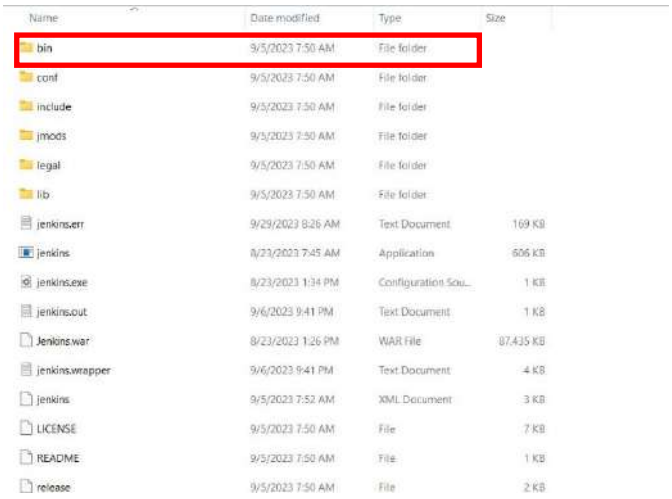
- **<u>Setting up of Environment Variables for Java.</u>**

Note: To run Java program, make sure to setup the Environment variables for Java on your windows system.

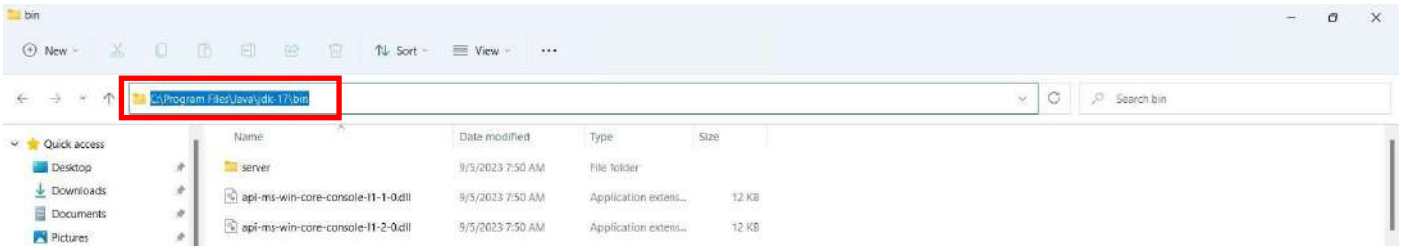
Step 1: After the installation of Java, go to This PC → C drive → Program files → Java → jdk17 as shown below.



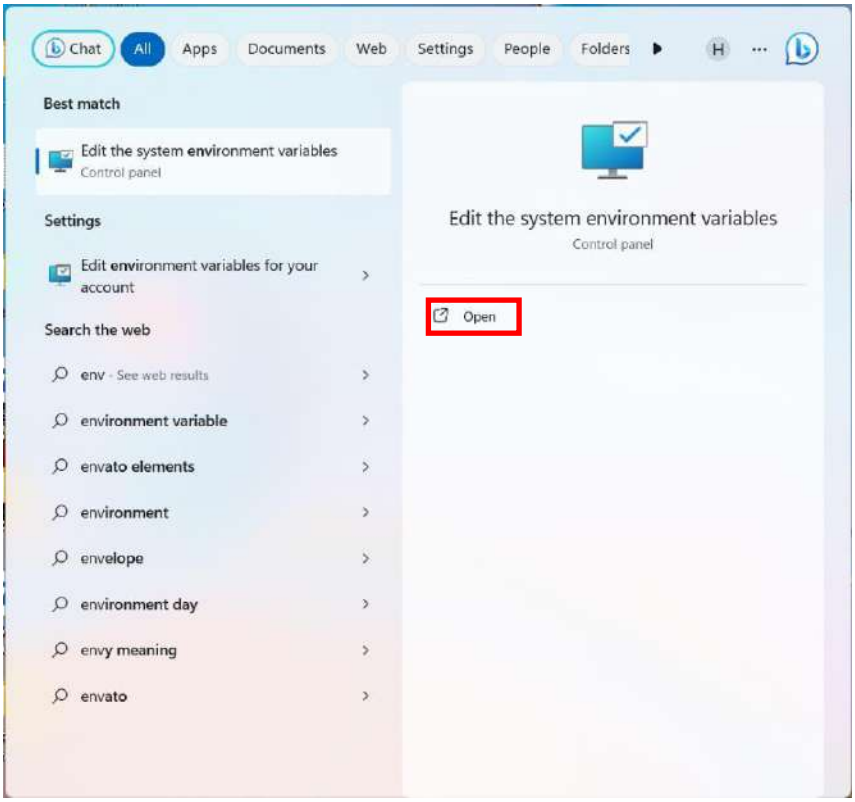
Step 2: Now click on **'bin'** folder.



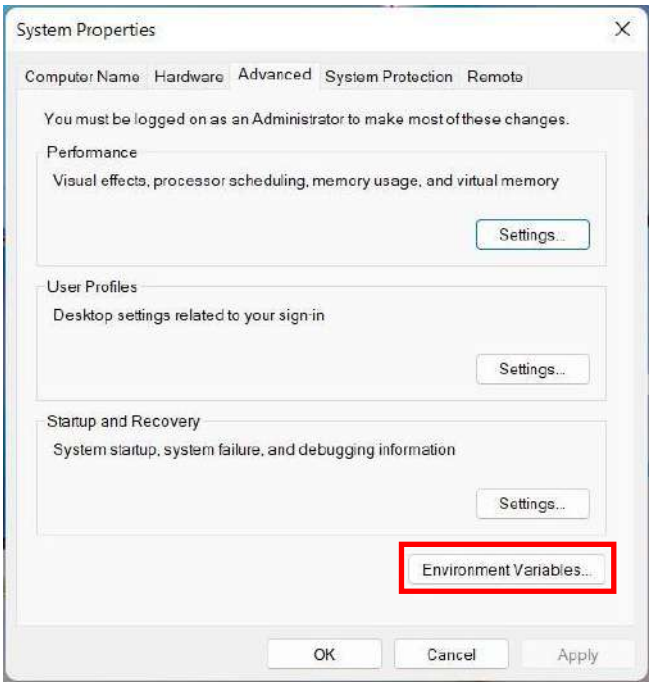
Step 3: Then copy the copy the bin location here.



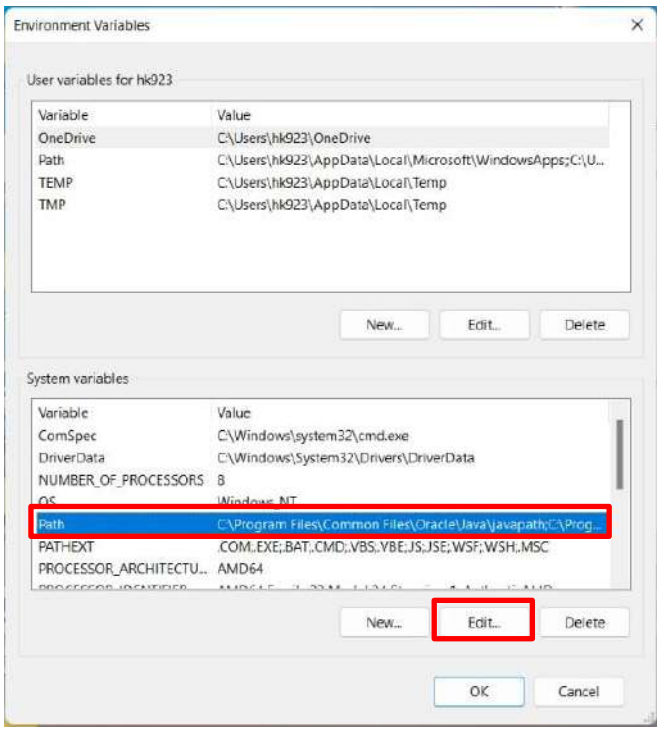
Step 4: Go to windows search bar Search for **'Environment Variables'** & click on **Open**



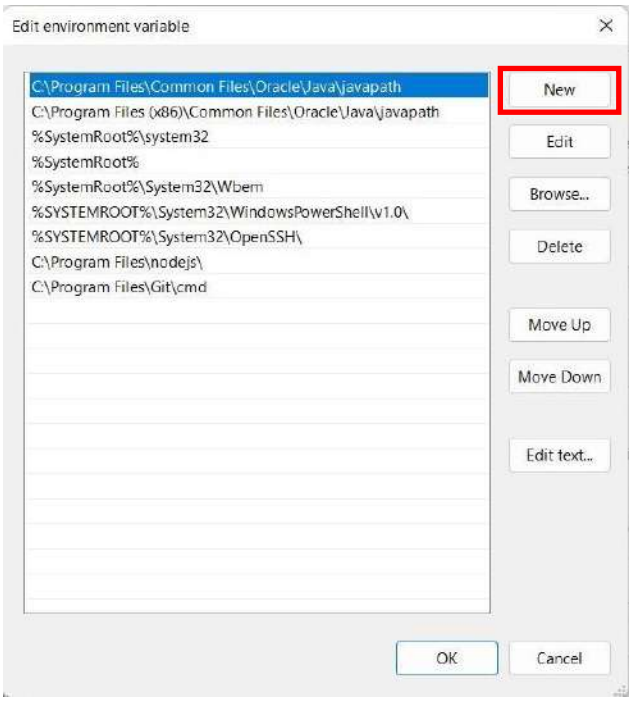
Step 5: Now click on **'Environment Variables'** here.



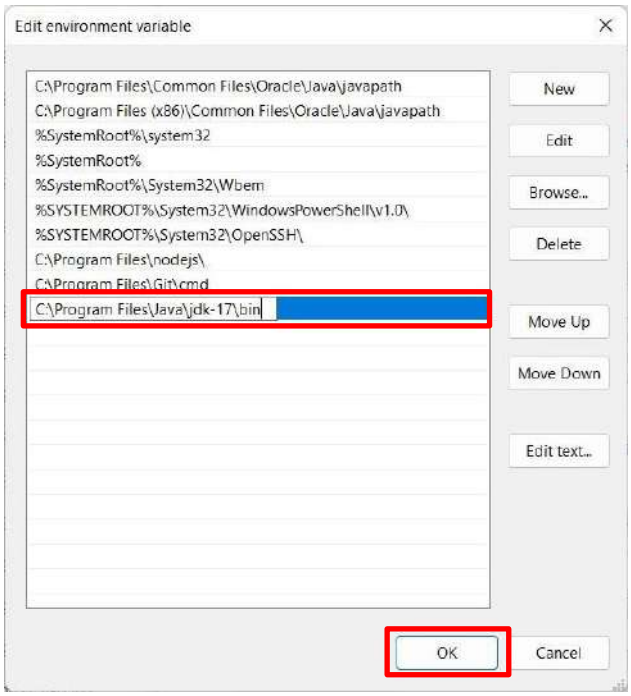
Step 6: Under System Variables, select Path location & click on **'Edit'**.



Step 7: Click on **'New'**.



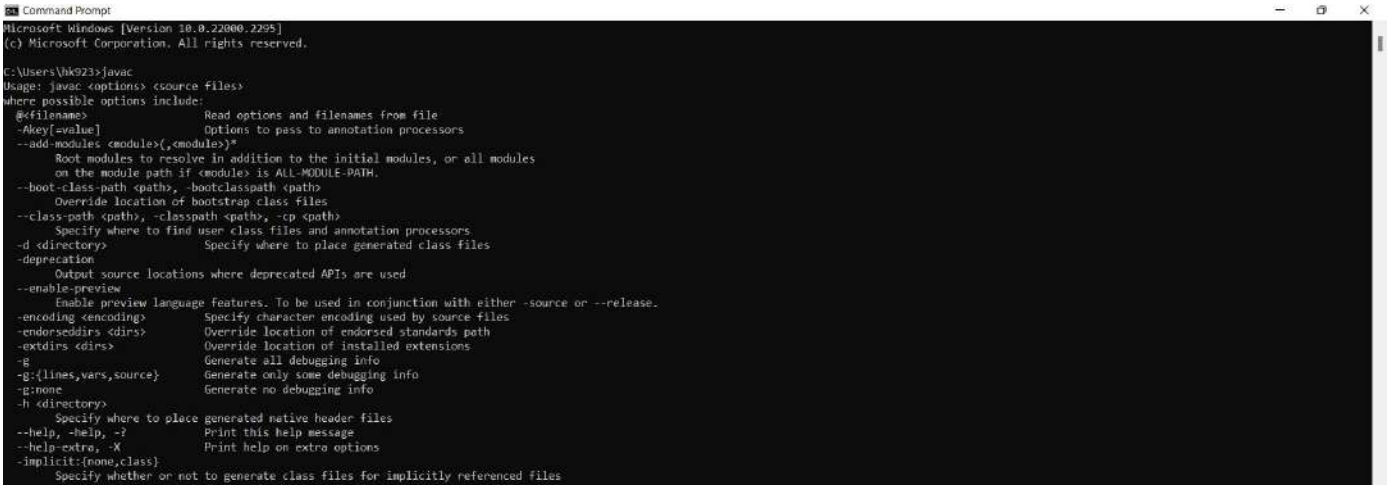
Step 8: Now paste the bin location which was copied in step 3.



Step 9: Use Command Prompt to check whether the Java Program is working or not.

Give the command as **'java -version'** to check the Version and other details as shown below.

Step 10: Now again enter the command as **'javac'** to check the working of Java compiler on your Windows.

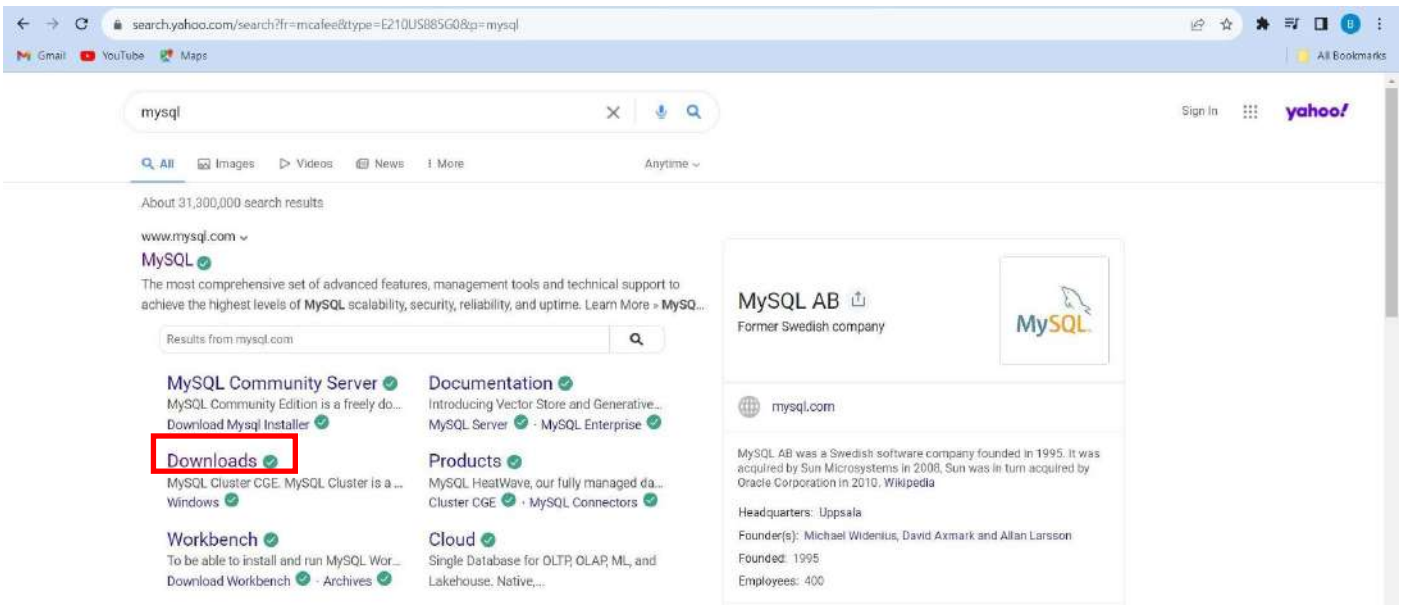


5. Install DBMS (MySQL, PostgreSQL or any other) tools.

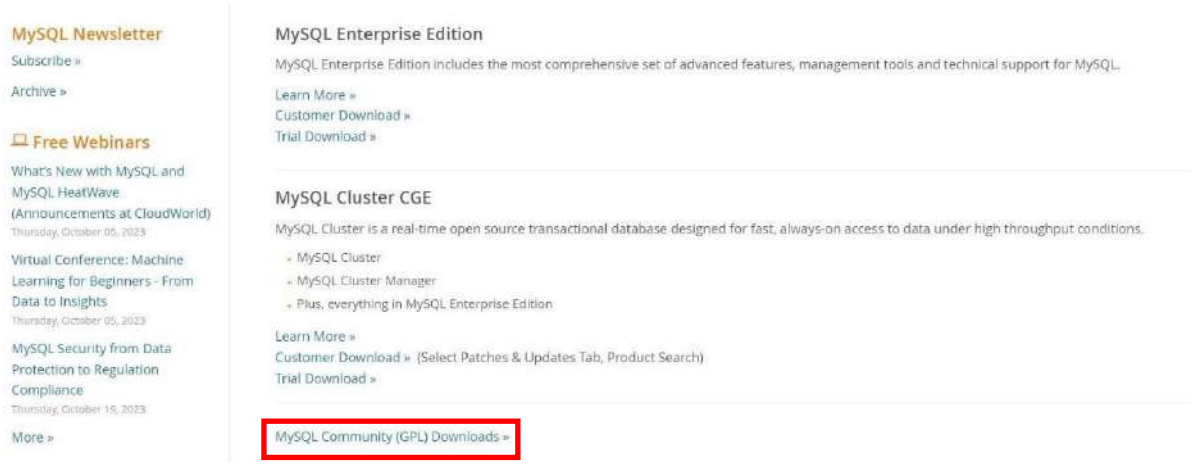
MySQL: **MySQL** is an open-source relational database management system (RDBMS). SQL is a language that programmers use to create, modify and extract data from the relational database, as well as control user access to the database. It manages the users, allows for network access and facilitates testing database integrity and creation of backups.

- **Installation procedures for MySQL:**

Step 1: Go to Web browser Search for MySQL Click on **'Downloads'**.

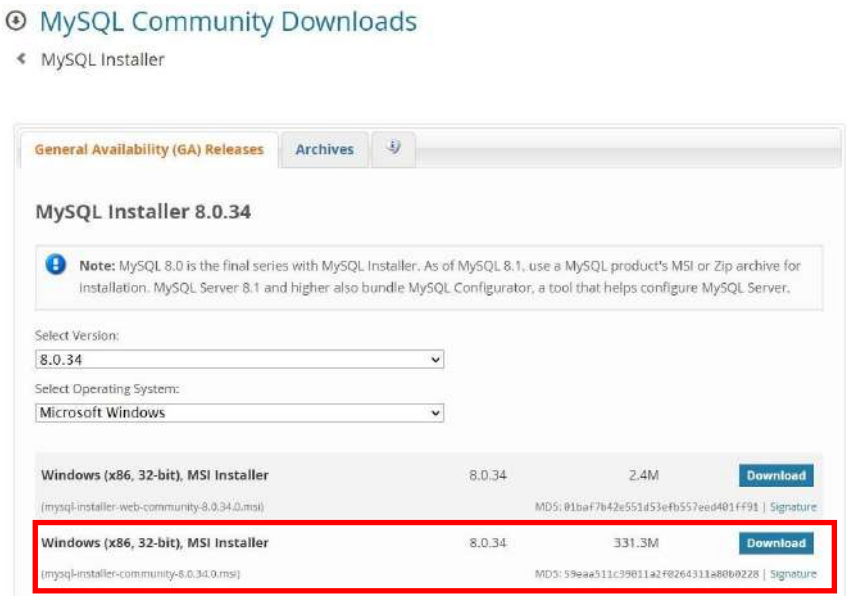


Step 2: Click on **'MySQL Community (GPL) Downloads'**.



Step 3: Now click on **'MySQL Installer for Windows'**.

Step 4: Then select the second option & click on **'Download'**.

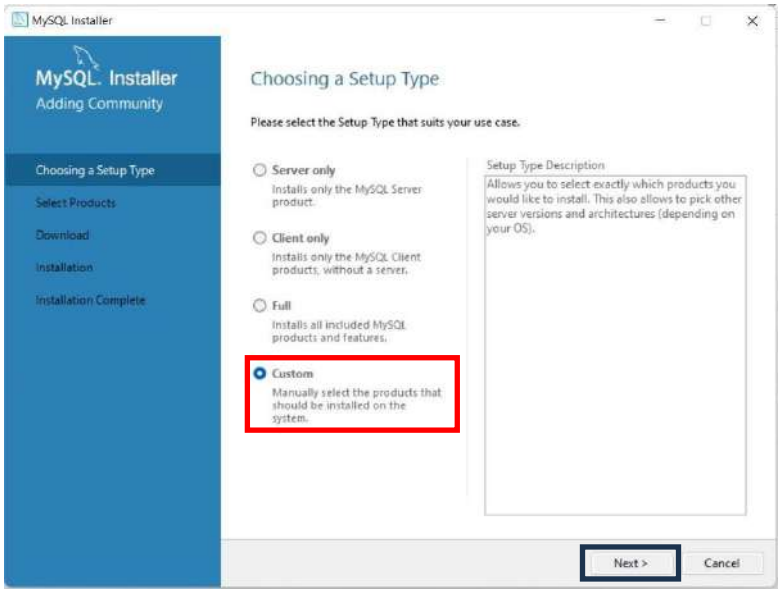


Step 5: Now you can Login or Sign up with your Oracle account or simply click on **'No thanks, just start my download'**.

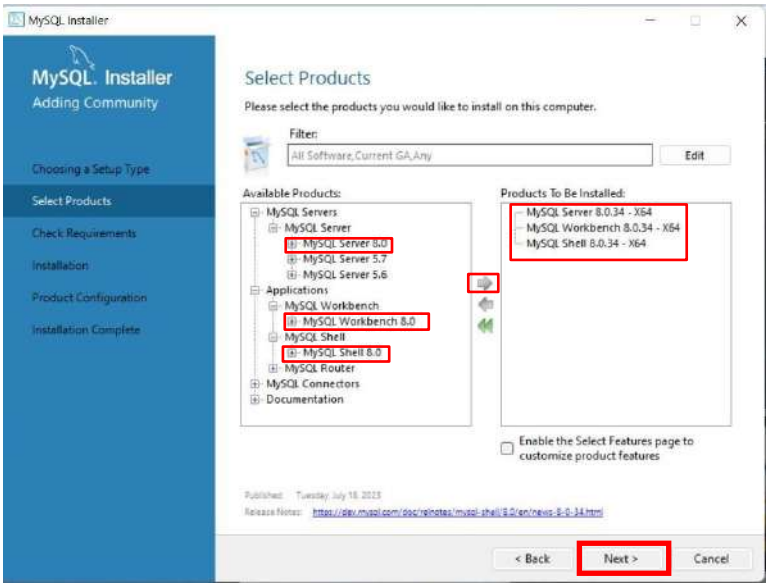
Step 6: Validating the internet connections.

Installing the MySQL on both client & server side interfaces.

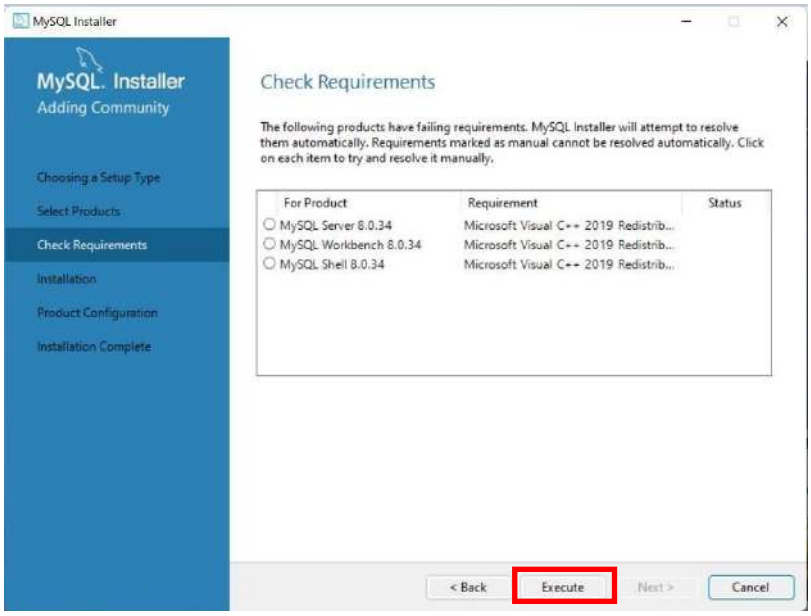
Step 7: Select **'Custom'** & click on **'Next'**.



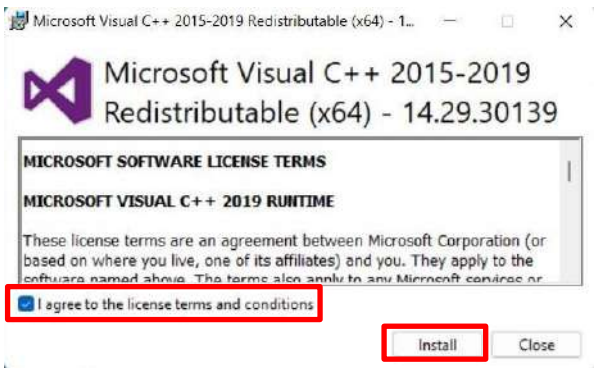
Step 8: Select the MySQL products from **Available products** to **Products to be installed** stage & click on **'Next'**.



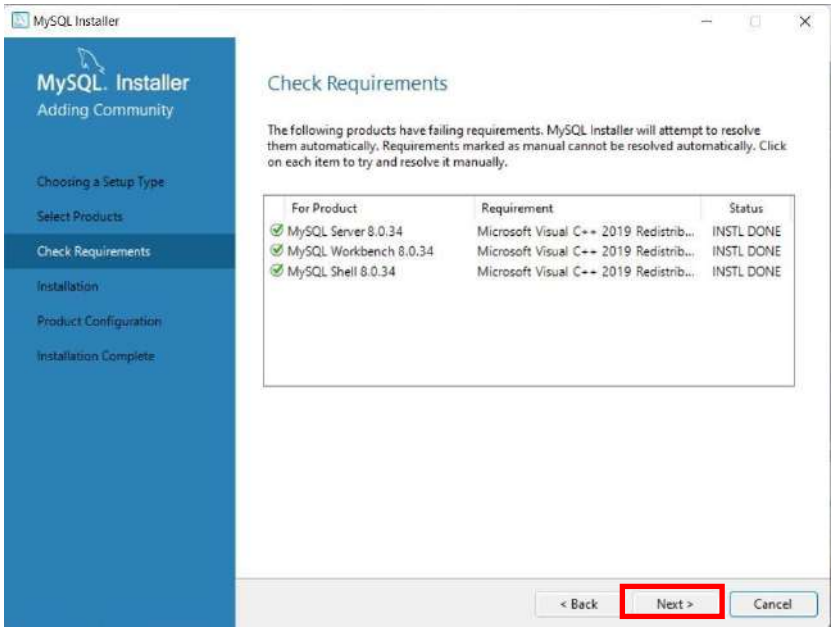
Step 9: Click on **'Execute'**.



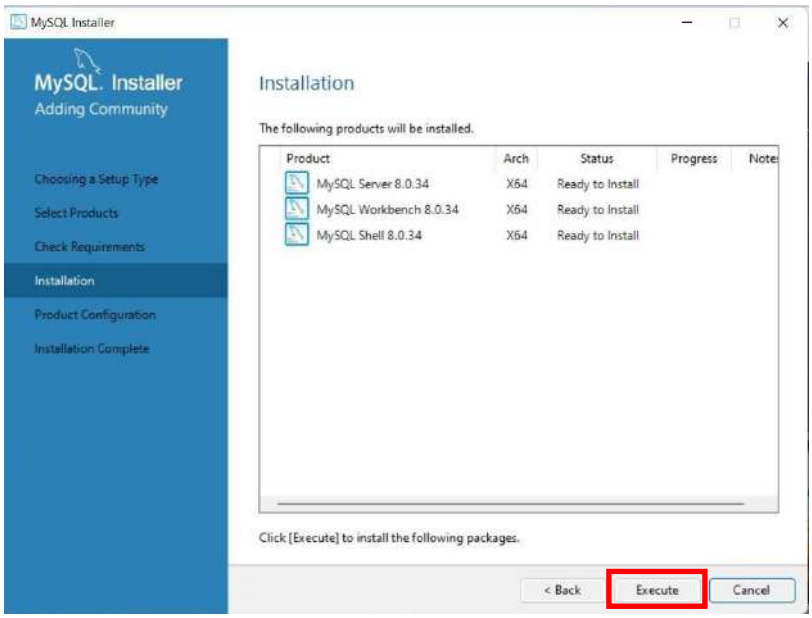
Step 10: Enable the option below & click on **'Install'**.



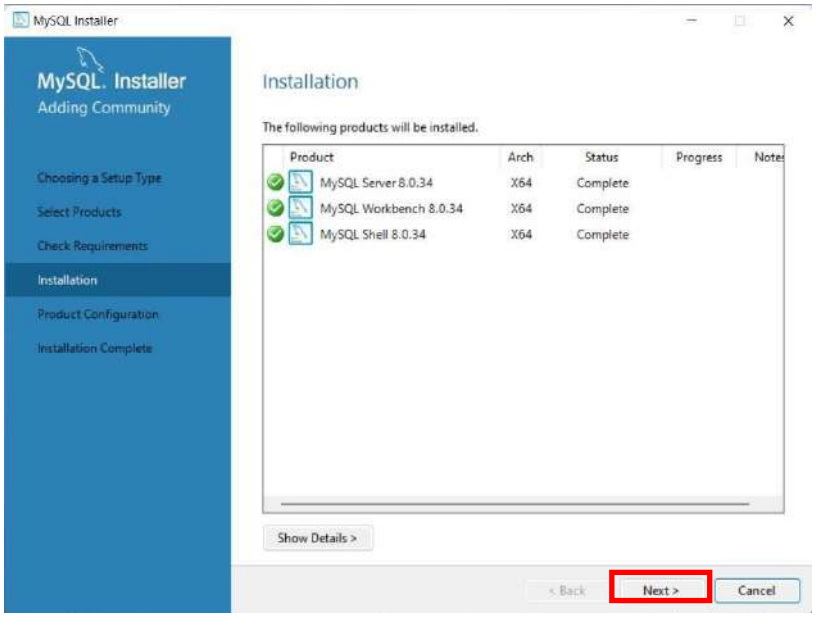
Step 11: After the installation of the products are complete, click on **'Next'**.



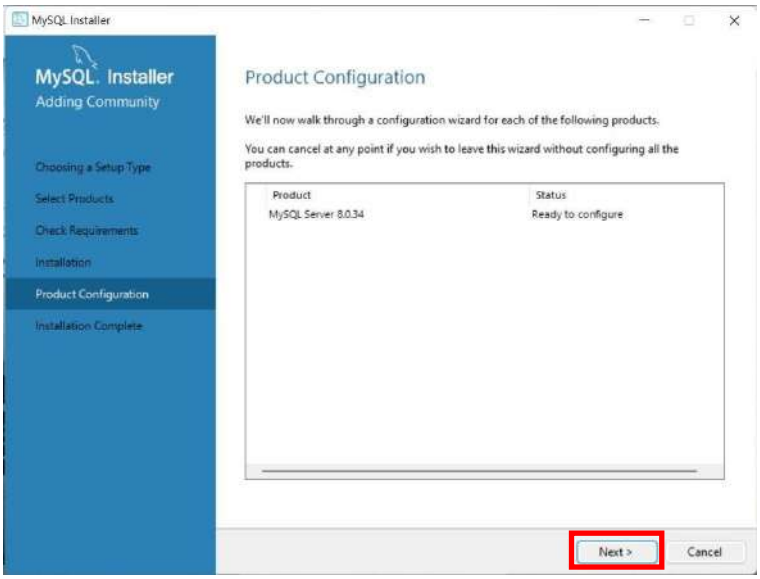
Step 12: Click on **'Execute'**.



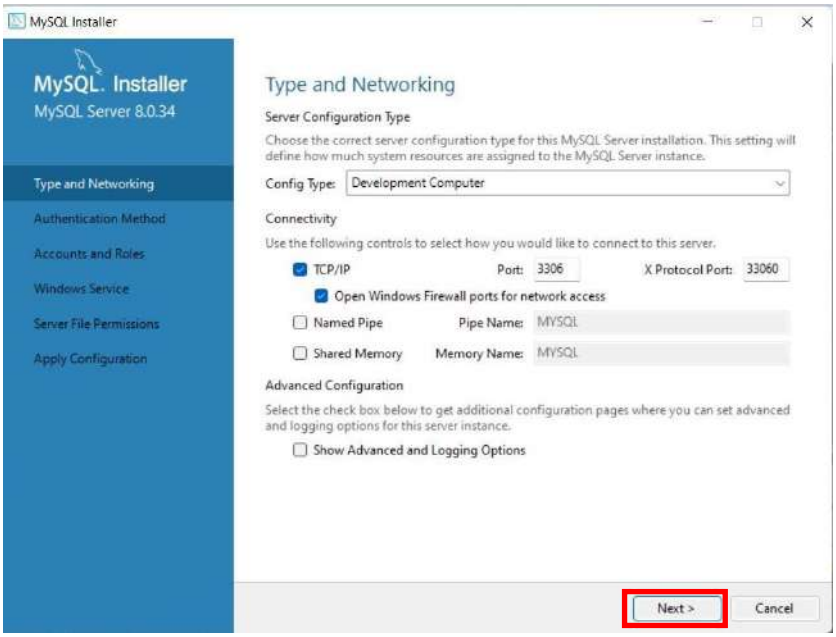
Step 13: After the complete execution, click on **'Next'**.



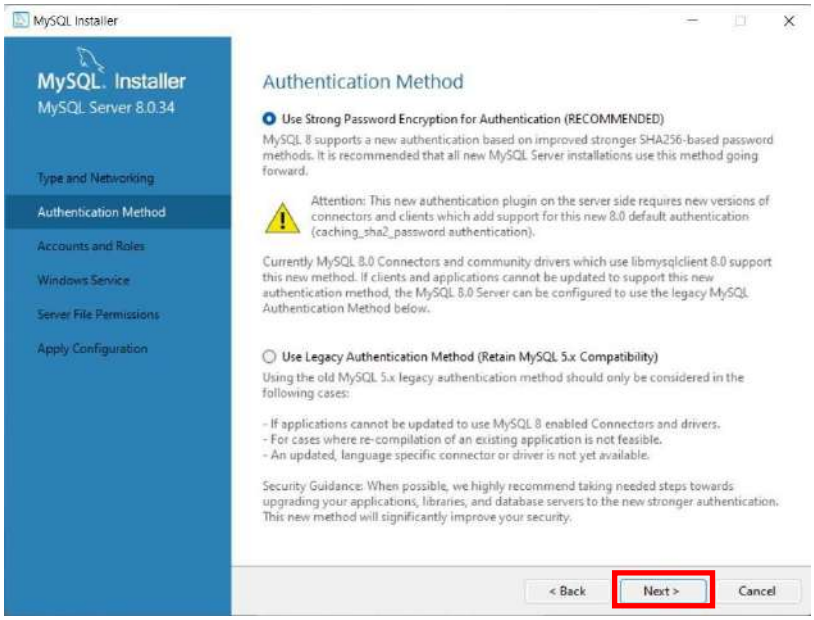
Step 14: Click on **'Next'**.



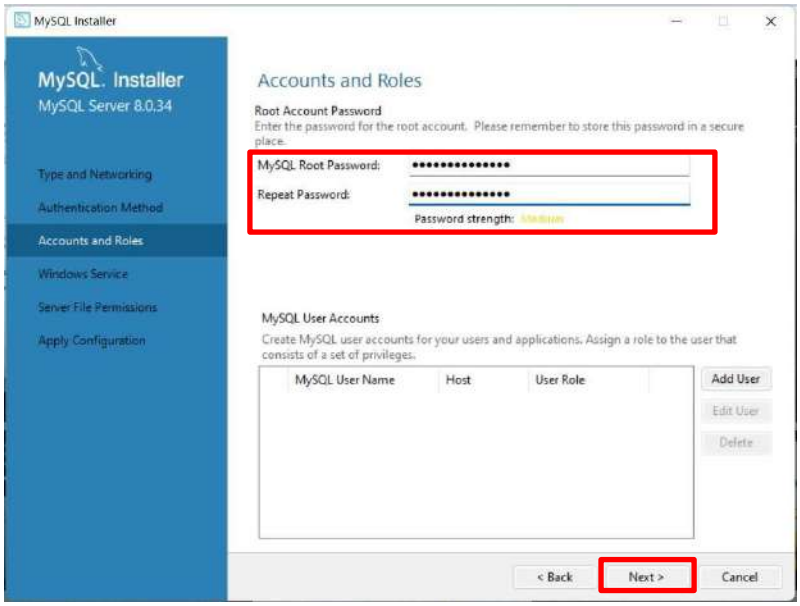
Step 15: Don't change the default selections, simply click on **'Next'**.



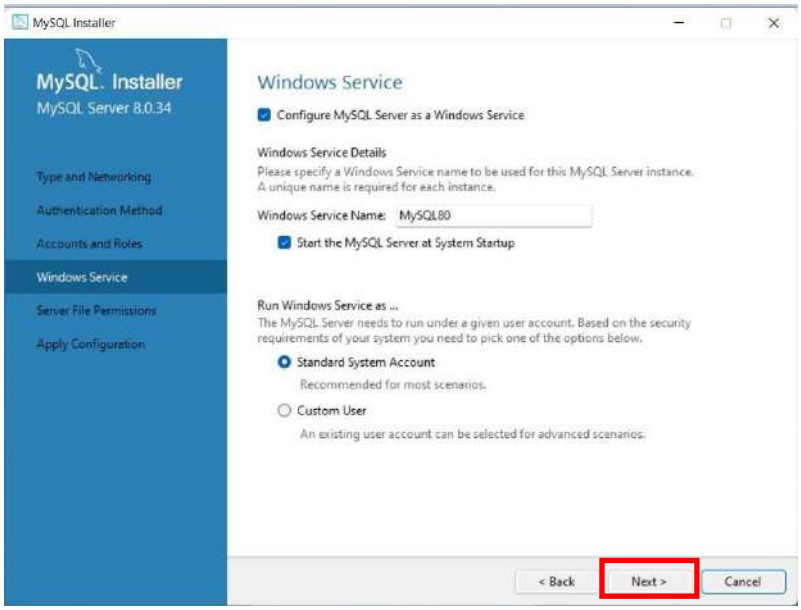
Step 16: Click on **'Next'**.



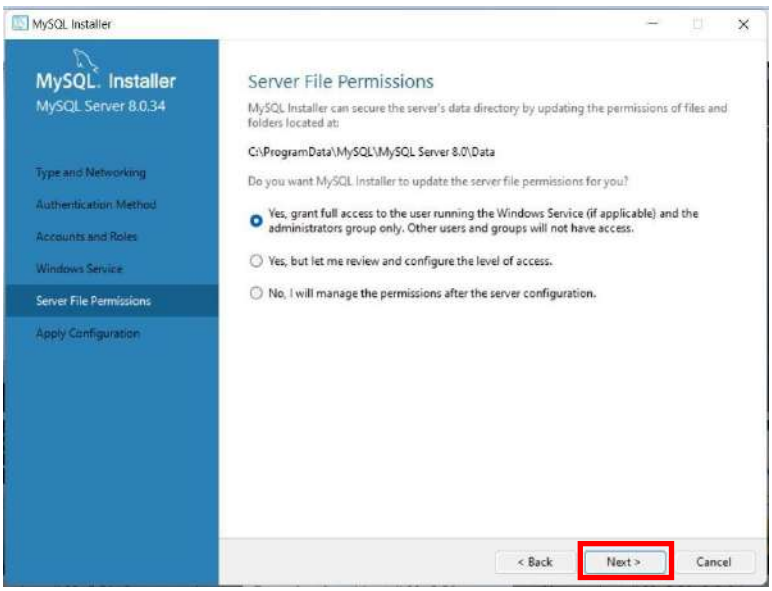
Step 17: Create a strong password for MySQL server & click on **'Next'.**



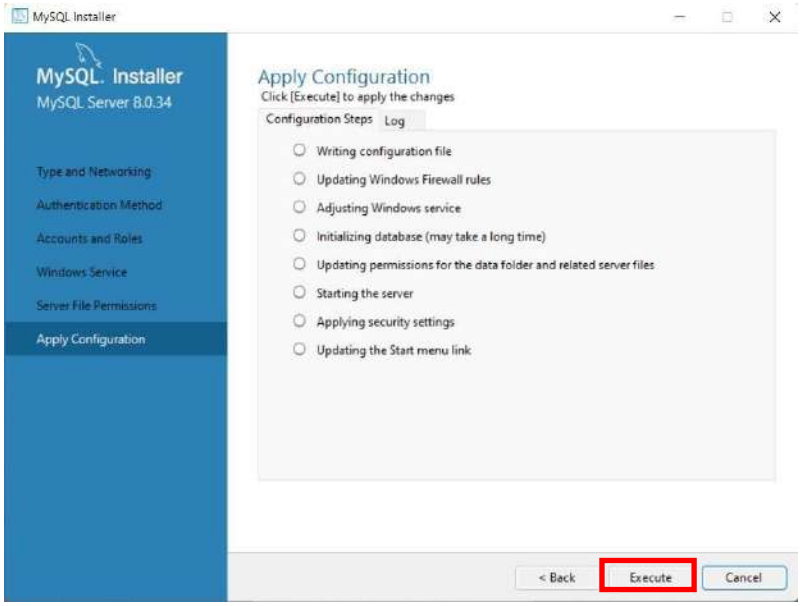
Step 18: Then click on **'Next'**.



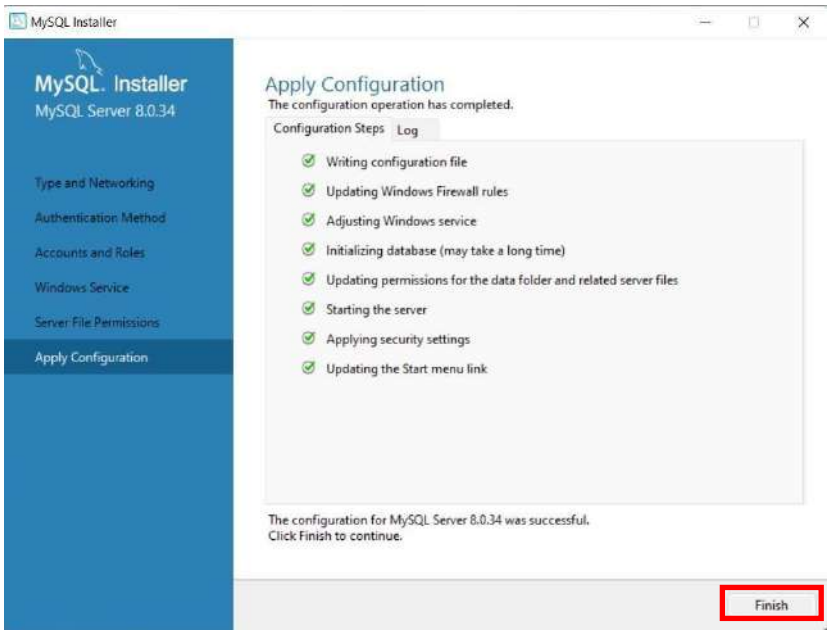
Step 19: Now click on **'Next'**.



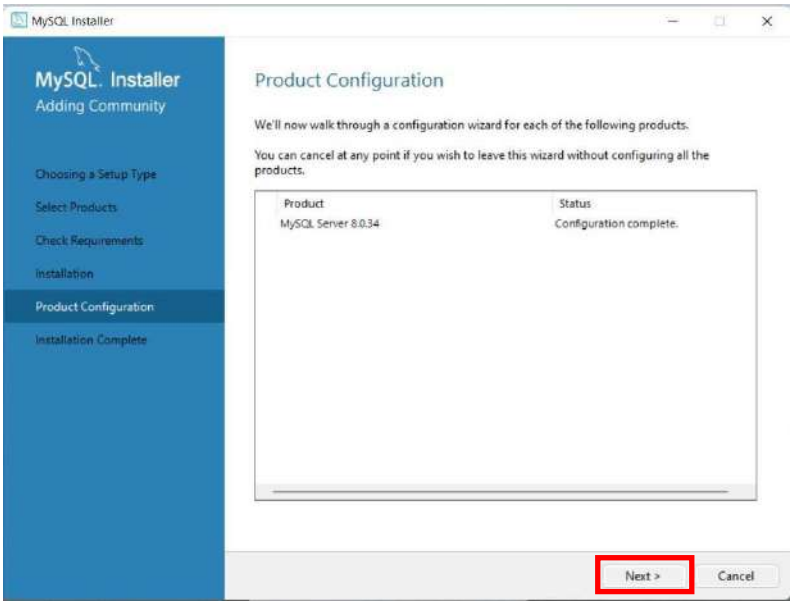
Step 20: Click on **'Execute'**.



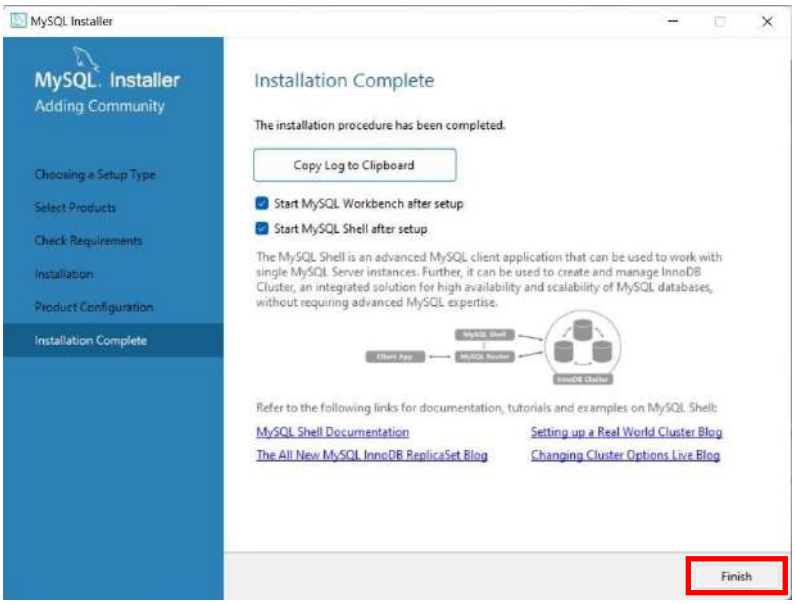
Step 21: Simply click on **'Finish'**.



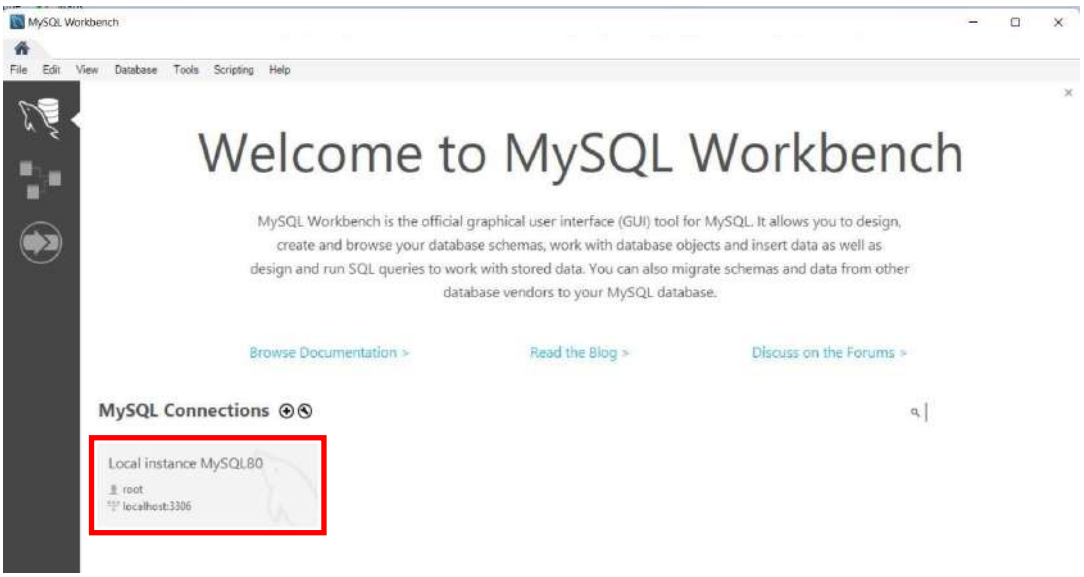
Step 22: Click on **'Next'**.



Step 23: Click on **'Finish'**.

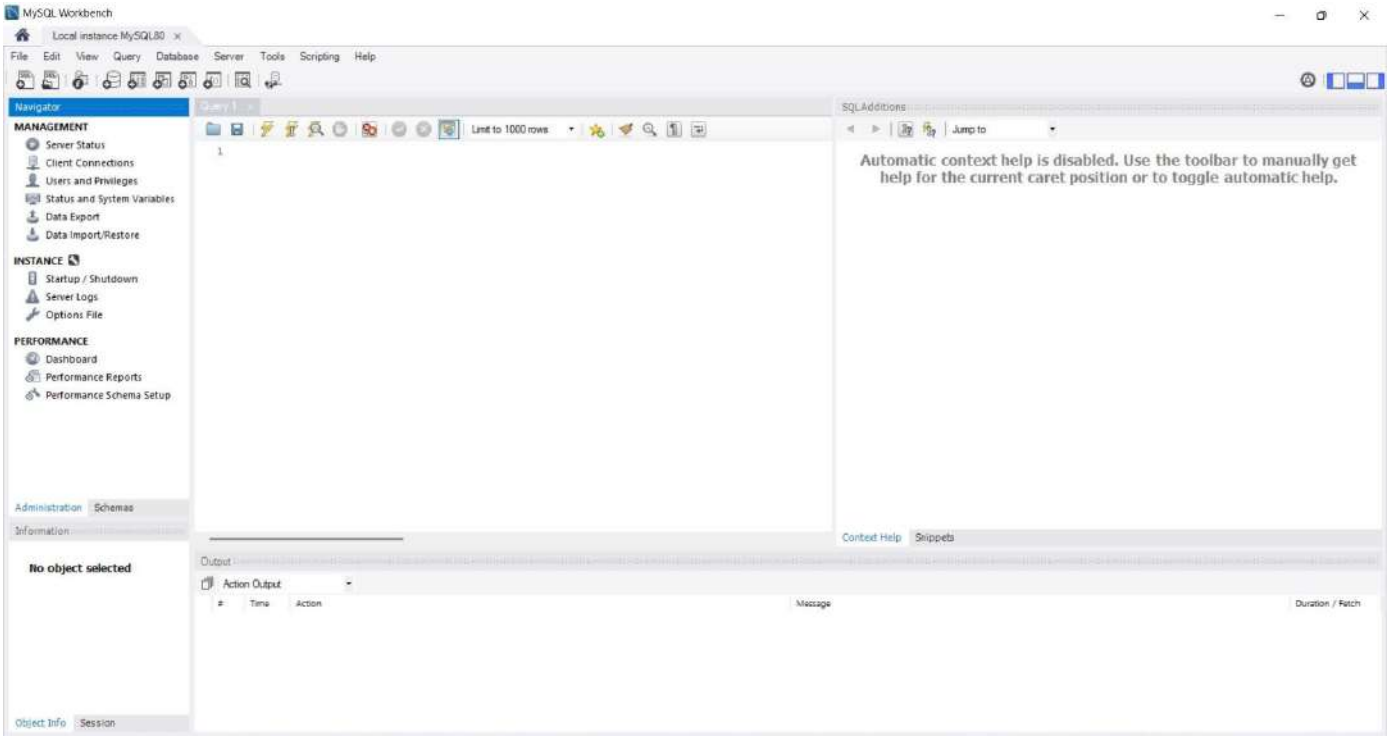


Step 24: Now click on **'Local instance MySQLBO'** here.



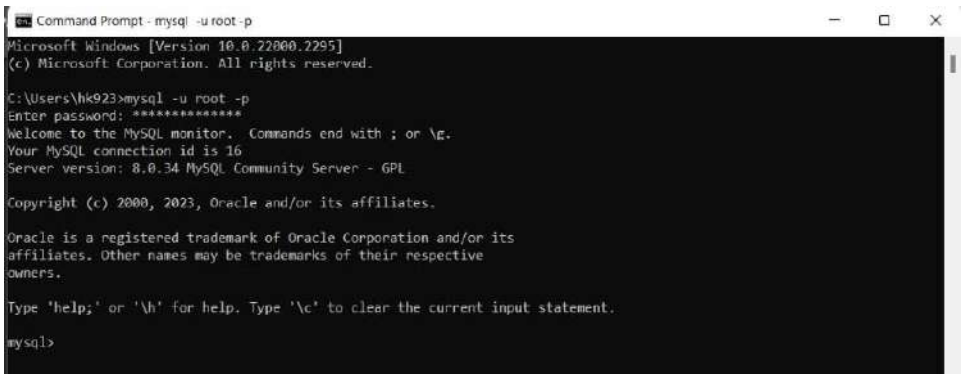
Step 25: Now enter your MySQL Server password here & click on **'OK'**.

Step 26: Here is your MySQL Workbench is ready to use.

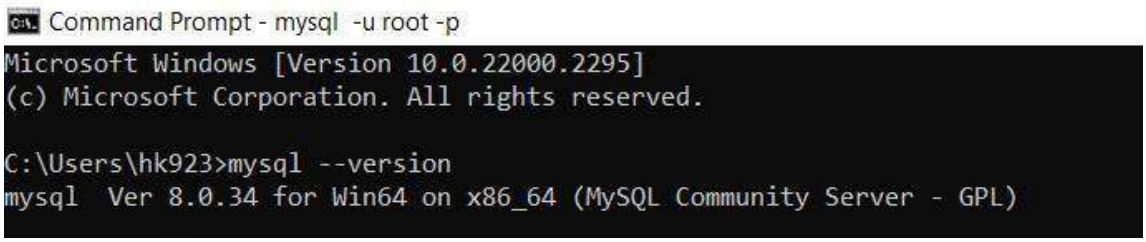


To run MySQL server:

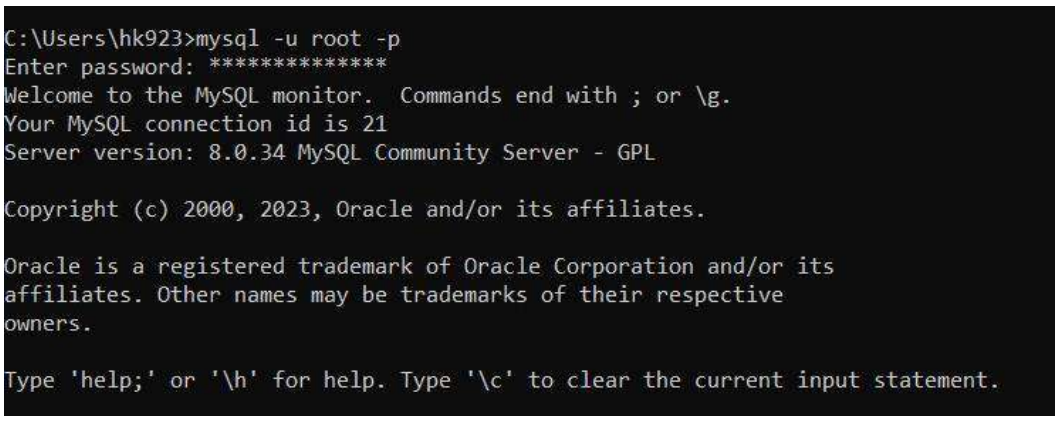
Setup the Environment Variables for MySQL server & enter the command as **'mysql -u root -p'**. Also enter your password to use the MySQL Monitor.



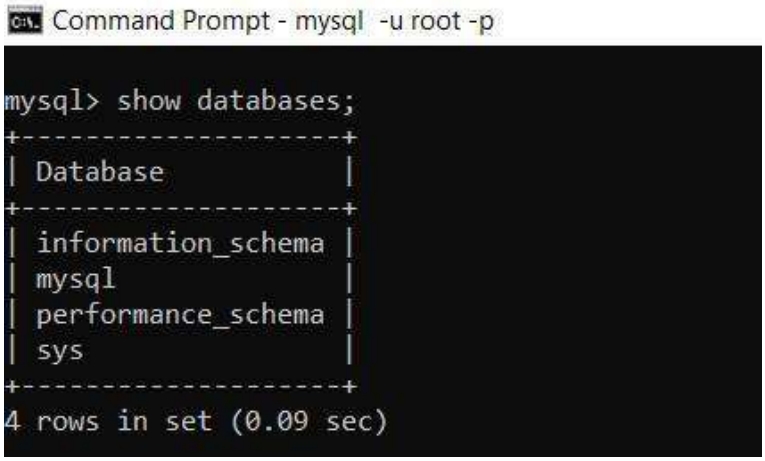
Step 1: Go to command Prompt → give the command as **'mysql –version'** → to check the current version of MySQL.



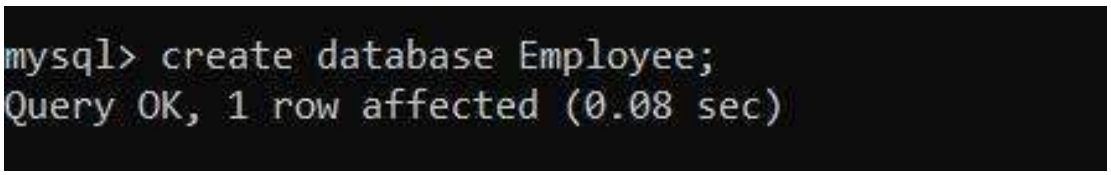
Step 2: Now give the command as **'mysql -u root -p'** & enter the password to enter into the MySQL server.



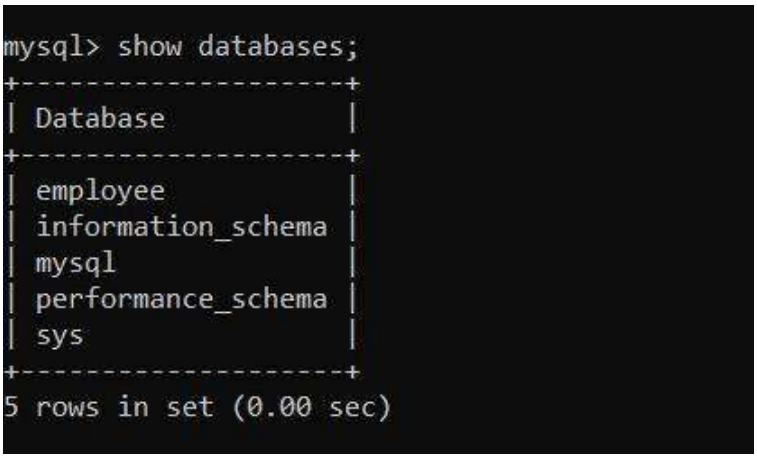
Step 3: Then enter **'show databases;'** to view the embedded databases.



Step 4: Enter the command **'create database Employee'** to create a new database.



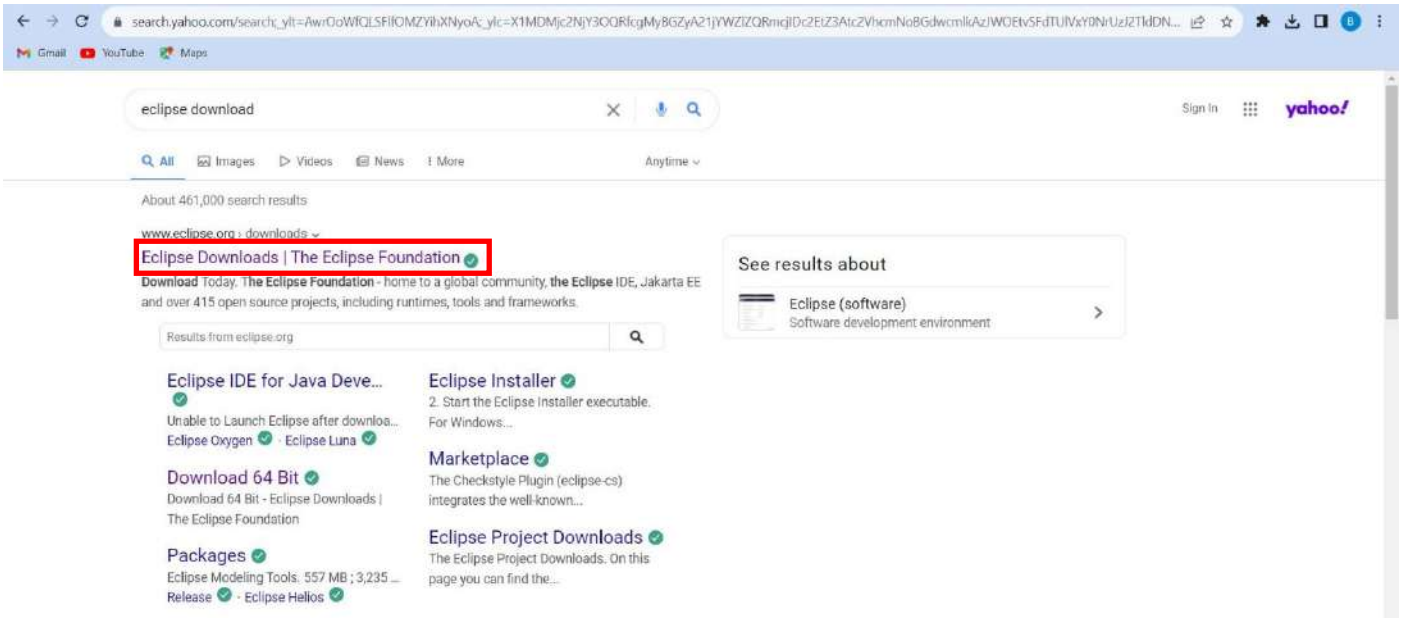
Step 5: Again enter the command 'show databases' to view the created databases as shown below.


mysql> show databases;
+--------------------+
| Database |
+--------------------+
| employee |
| information_schema |
| mysql |
| performance_schema |
| sys |
+--------------------+
5 rows in set (0.00 sec)

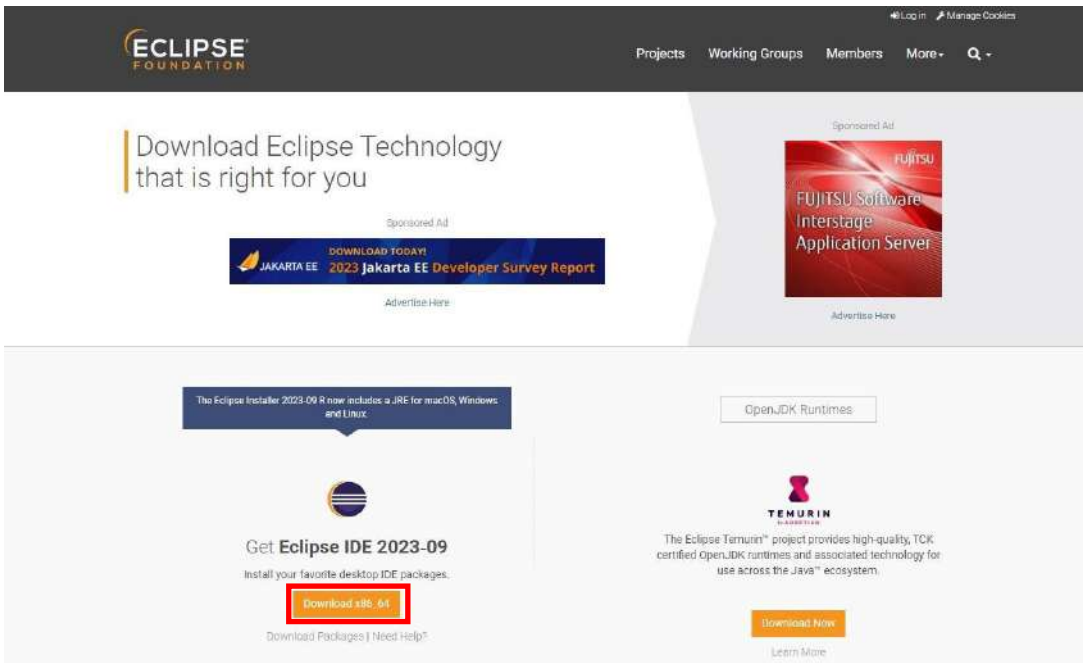
WEEK – 7

1. Installation of Eclipse IDE for Enterprise Java and Web Developers latest version on Windows.

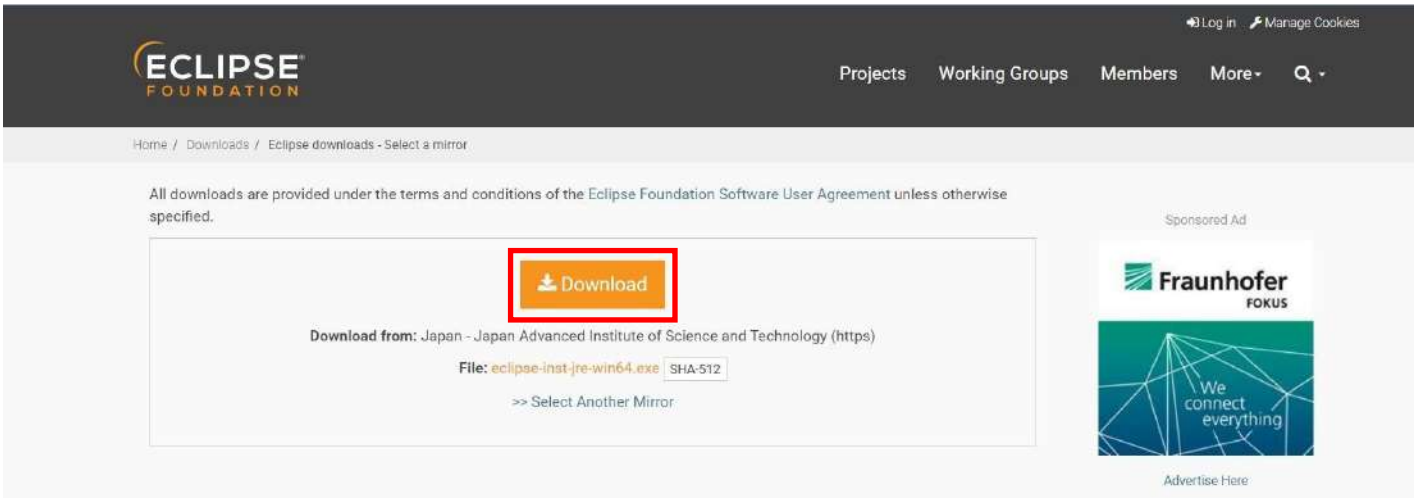
Step 1: Go to Web Browser Search for **'Eclipse download'** & click on the first link here.



Step 2: Now click on **'Download x86_64'** option.

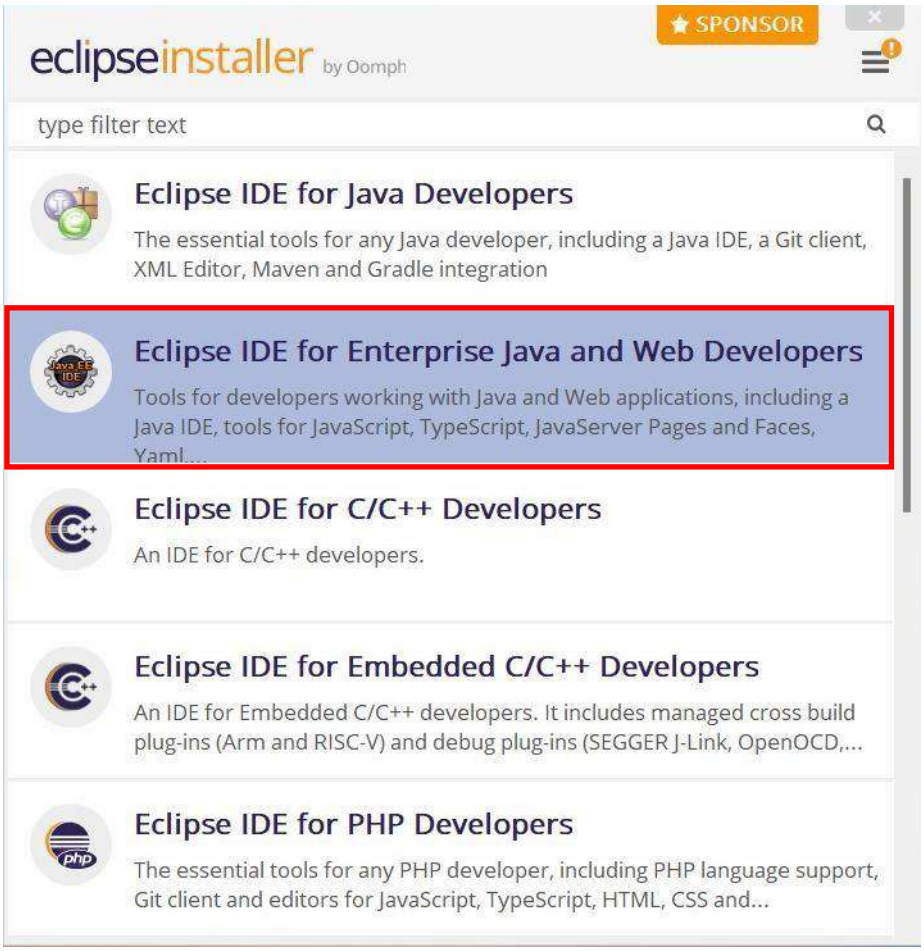


Step 3: Then click on **'Download'**.

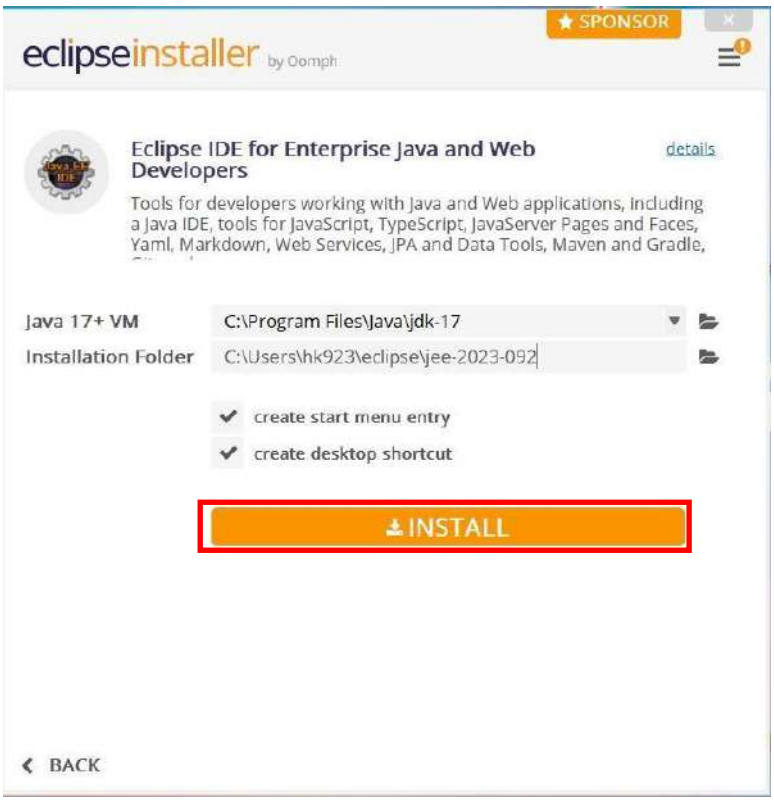


Step 4: Open the Downloaded file.

Step 5: Now select **'Eclipse IDE for Enterprise Java & Web Developers'.**

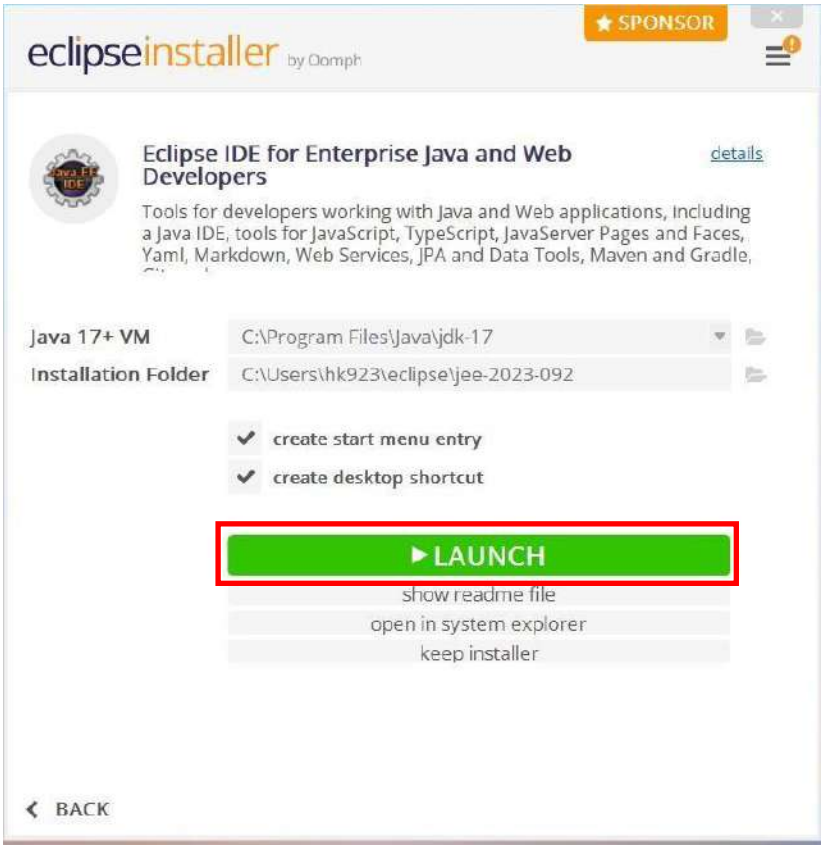


Step 6: Now click on **'Install'**.



Step 7: Wait until the installation is complete.

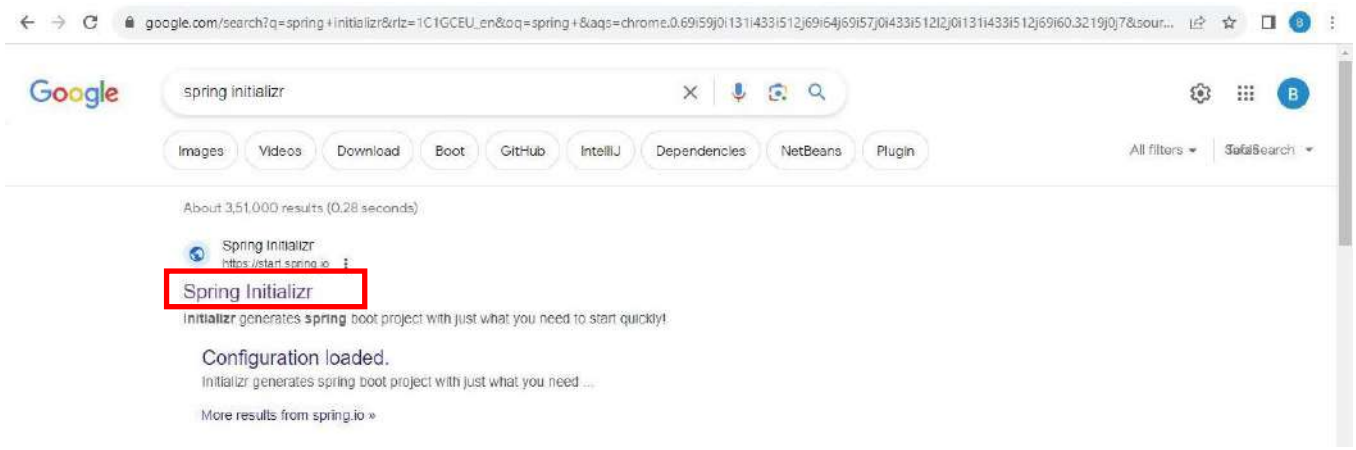
Step 8: Now click on **'Launch'**.



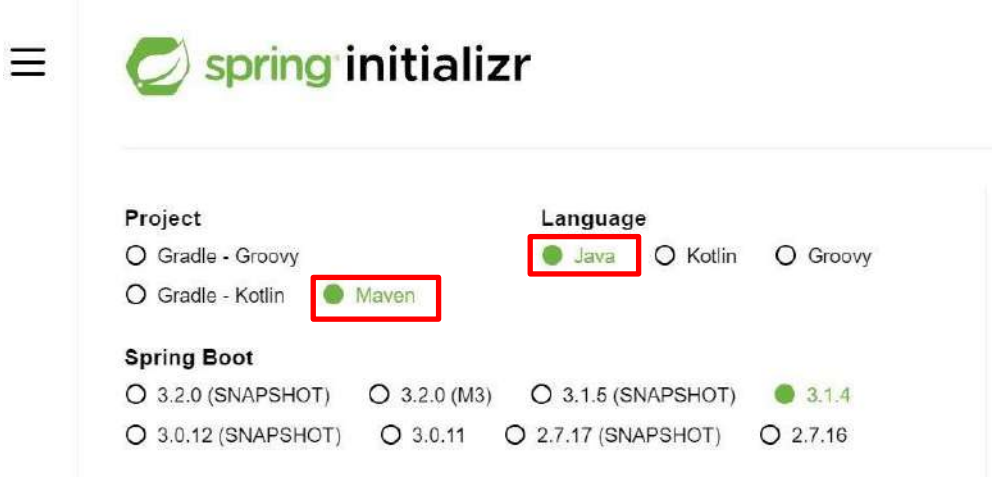
Step 9: Now the Eclipse IDE is ready to use.

2. Create Spring application with Spring Initializer using dependencies like Spring Web, Spring Data JPA.

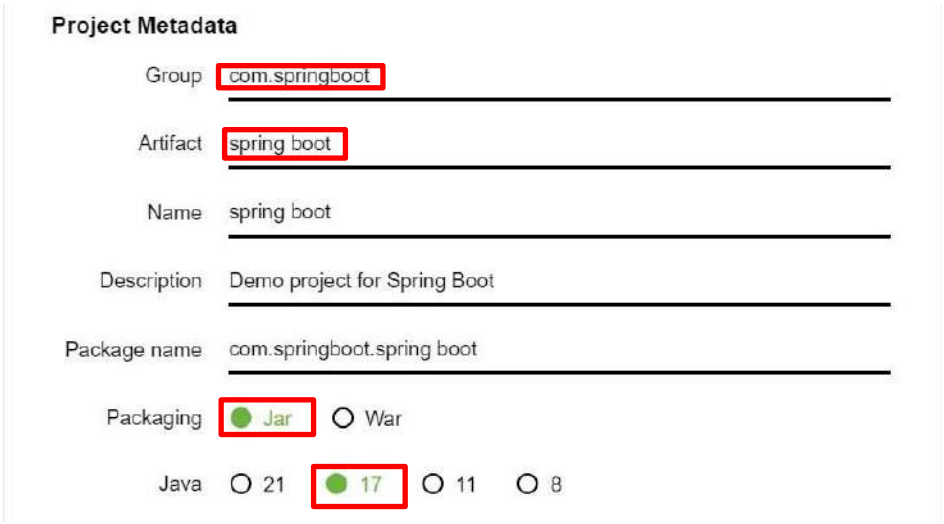
Step 1: Go to web browser search for **'Spring Initializr'** & click on the first link.



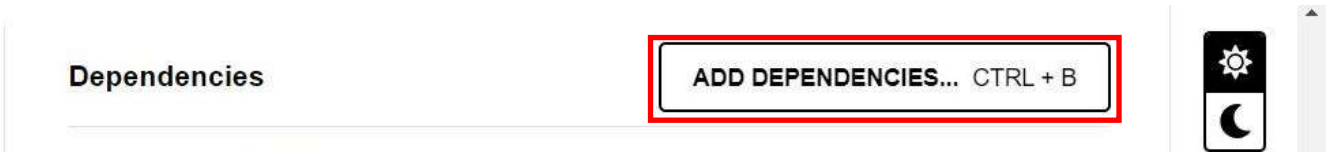
Step 2: Now select the below options.



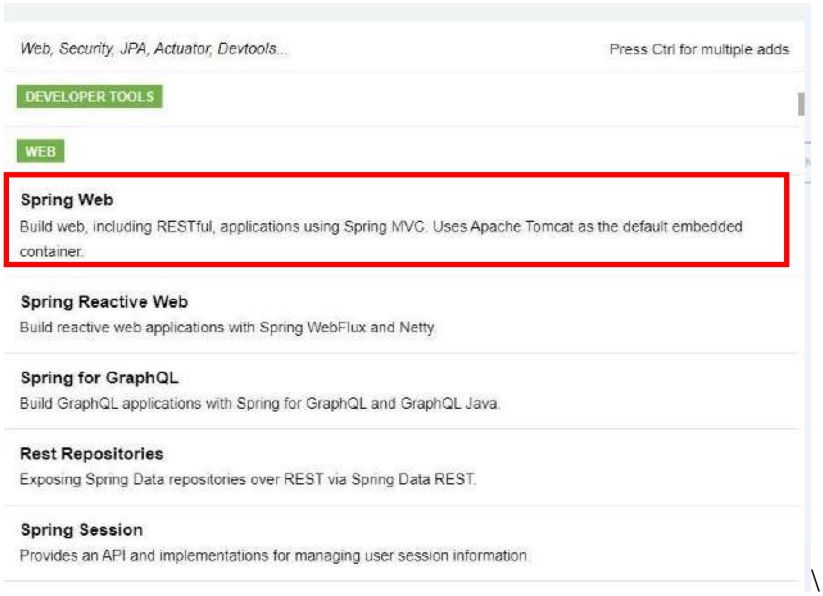
Step 3: In the Project Metadata section, give the name as **'springboot'**.



Step 4: Now click on **'ADD DEPENDENCIES'**.



Step 5: Under the **'Web'** section, select the **'Spring Web'**.

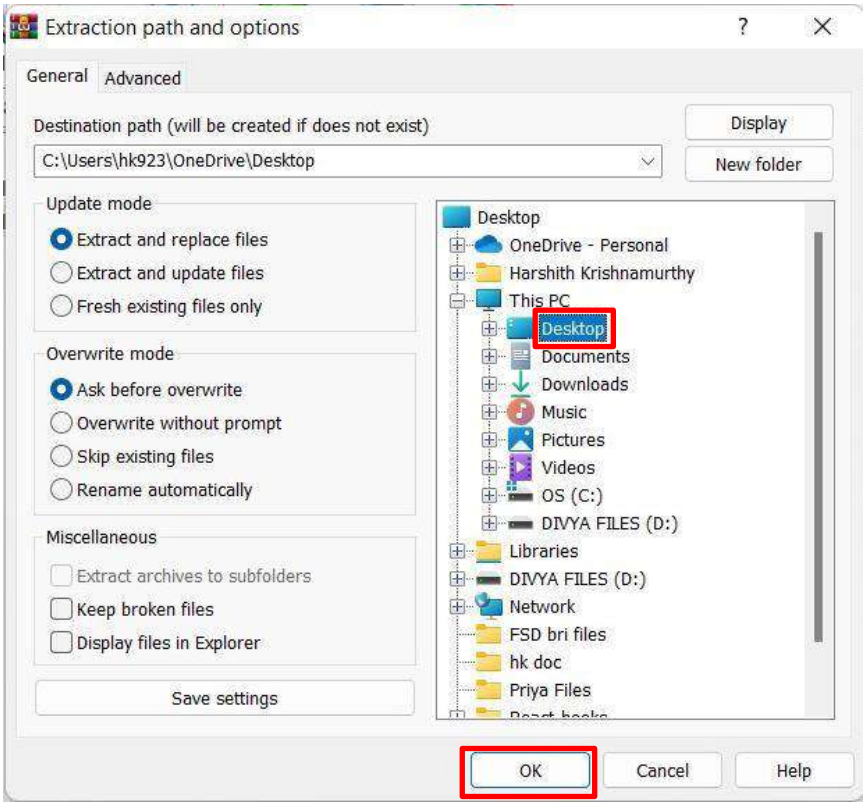


Step 7: Now click on **'GENERATE'** option.

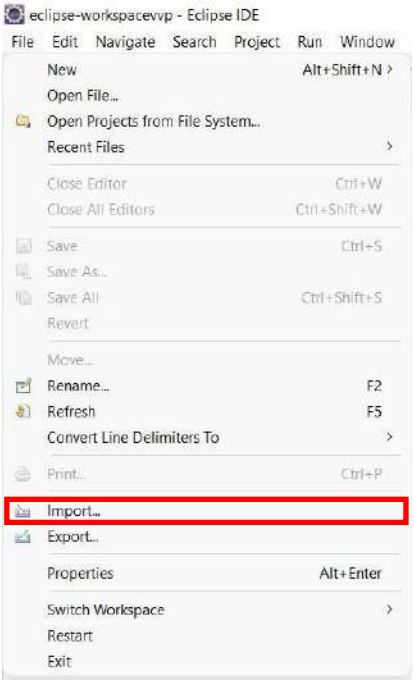
Step 8: Click on the zip file downloaded.

Step 9: Now extract the zip file which was downloaded.

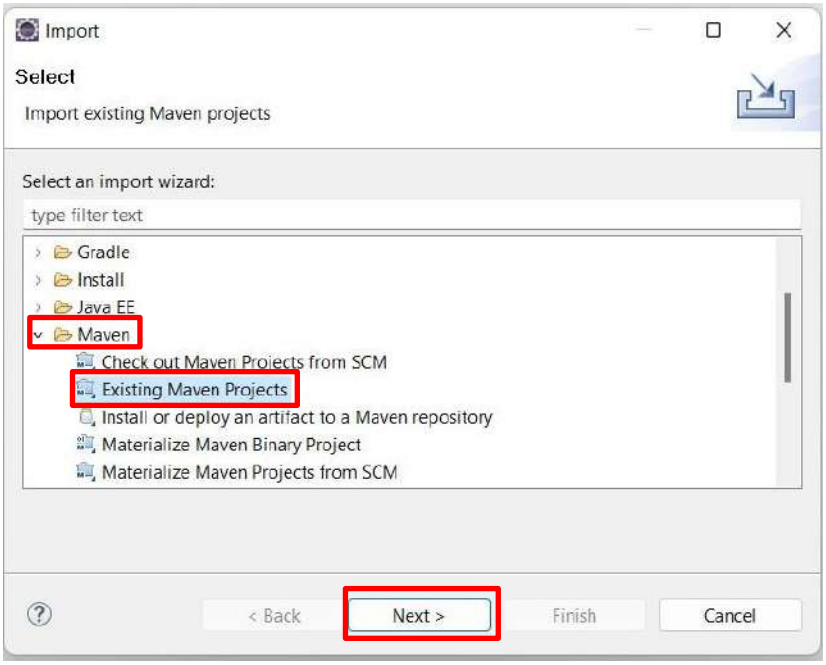
Step 10: Specify the location to save the file & click on 'OK'.



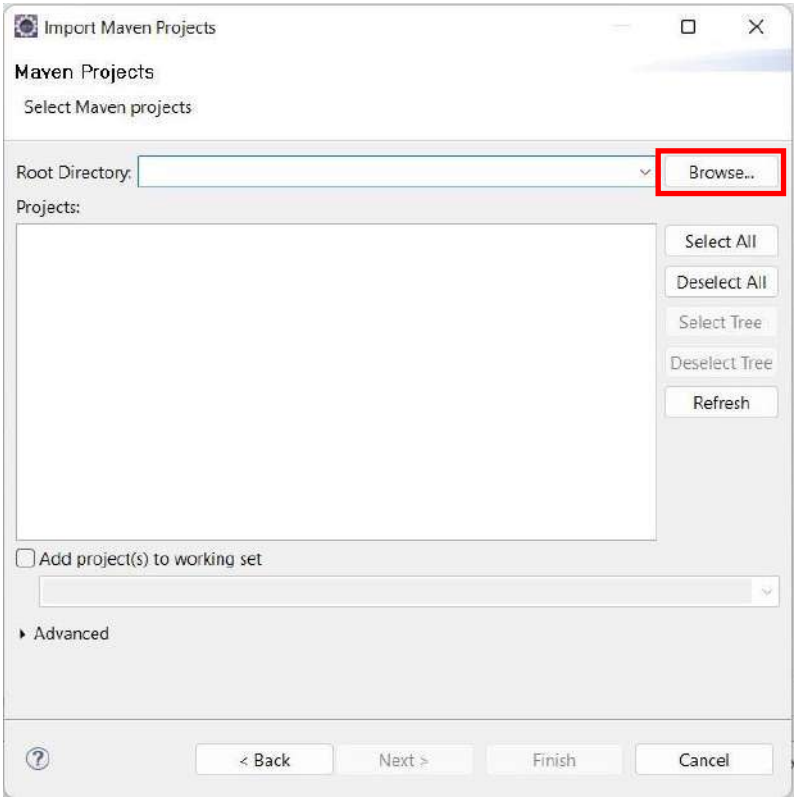
Step 11: Now launch the Eclipse IDE, go to **File** -→ click on **'Import'** option.



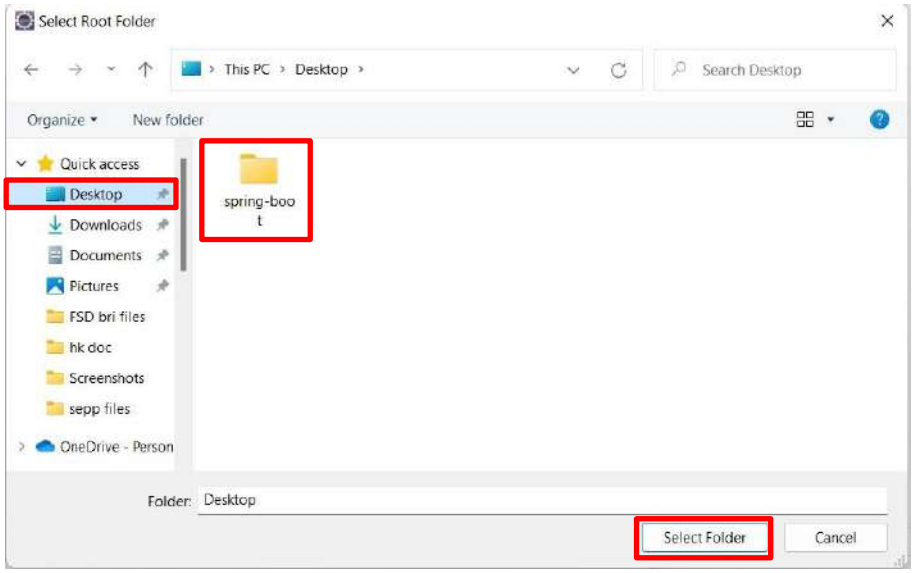
Step 12: Then select the **'Existing Maven Projects'** under the **'Maven'** folder & click on **'Next'**.



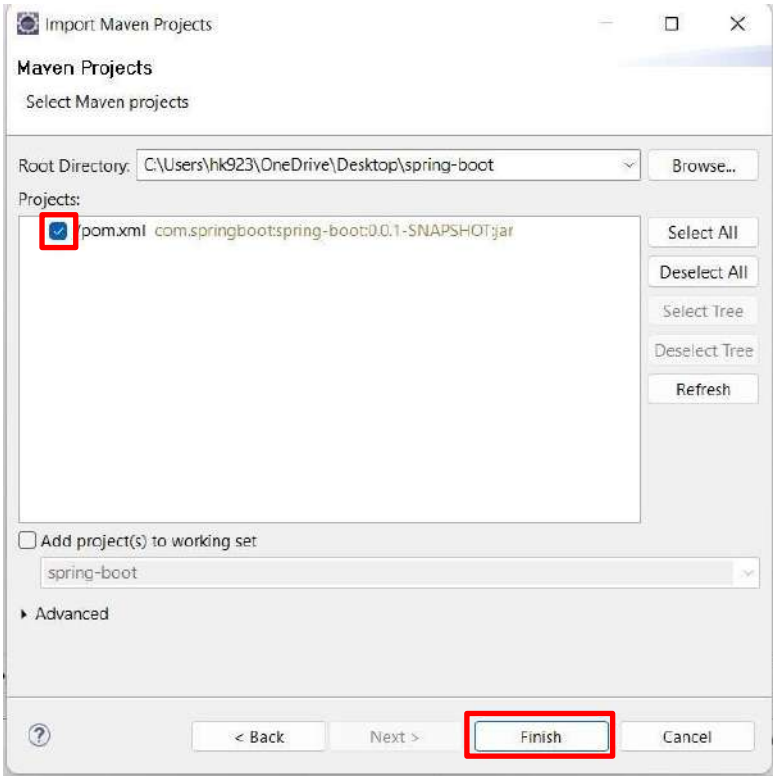
Step 13: Click on **'Browse'** to specify the Root Directory.



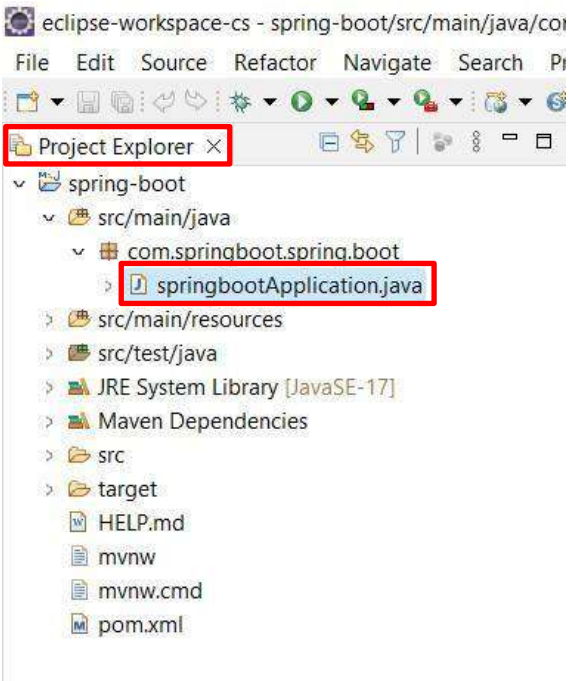
Step 14: Now select the location of the file where the zip file was extracted.



Step 15: Now click on **'Finish'**.



Step 16: Now go to → **'springboot'** folder → **'src/main/java'** → **'springbootApplication.java'** created in the **'Package Explorer'** section.



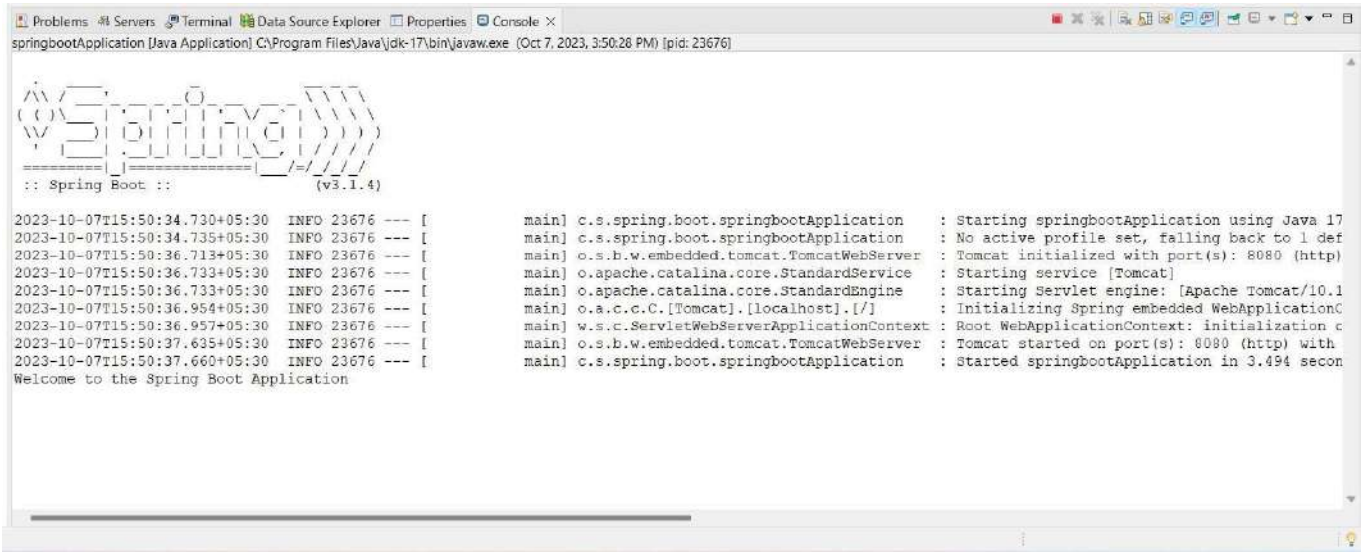
Step 17: Then add the below code to this file & Save it.

springbootApplication.java

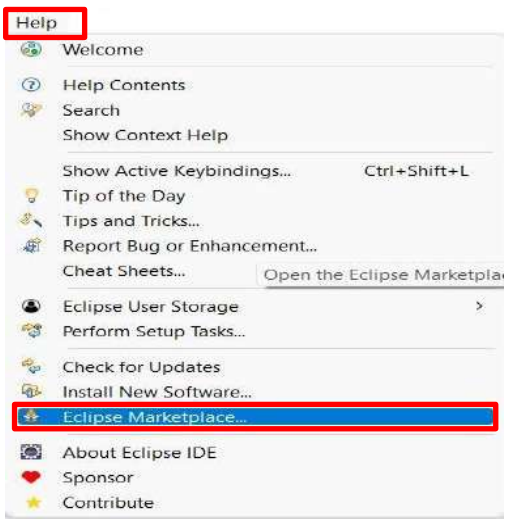
```java
package com.springboot.spring.boot;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class springbootApplication {

    public static void main(String[] args) {
        SpringApplication.run(springbootApplication.class, args);
        System.out.println("Welcome to the Spring Boot Application");
    }
}
```

Step 18: Now right click on the **'springbootApplication.java'** file □ **'Run As'** □ **'1Java Application'**. If the option is not made visible, click on **'7Maven install'** & install the **'1Java Application'**.

Output:

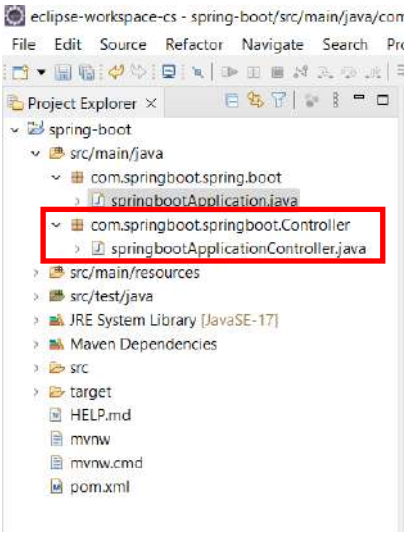


Step 19: Go to **'Help'** □ **'Eclipse Market place'** for downloading the required spring tools.



Step 20: Now search for **'spring tool suite'** & install the **'Spring Tool 4 (aka Spring Tool Suite 4) 4.20.0 RELEASE'** latest version.

Step 21: Now create a new package named **'com.springboot.springboot.Controller'** & a class named **'springbootApplicationController.java'** file in the **'Project Explorer'** section.



Step 22: Then write the below code in the **'springbootController.java'** file as shown below & save it. Now run the **'springbootAppliction.java'** file to view the output.

springbootApplication.java | springbootController.java

```
1  package springboot.Controller;
2
3  import org.springframework.stereotype.Controller;
4  import org.springframework.web.bind.annotation.RequestMapping;
5  import org.springframework.web.bind.annotation.ResponseBody;
6
7  @Controller
8  public class springbootController {
9      @RequestMapping("/hello")
10     @ResponseBody
11     public String HelloWorld () {
12         return "Welcome to Spring Boot";
13     }
14 }
15
16
17
18
19
```

Outline: springboot.Controller; springbootController; HelloWorld() : String

Problems | Servers | Terminal | Data Source Explorer | Properties | Console

springbootApplication [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Oct 7, 2023, 4:30:29 PM) [pid: 13128]

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.1.4)

2023-10-07T16:30:32.659+05:30  INFO 13128 --- [           main] c.s.spring.boot.springbootApplication    : Starting springbootApplication using Java 17
2023-10-07T16:30:32.665+05:30  INFO 13128 --- [           main] c.s.spring.boot.springbootApplication    : No active profile set, falling back to 1 def
2023-10-07T16:30:35.029+05:30  INFO 13128 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2023-10-07T16:30:35.054+05:30  INFO 13128 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-10-07T16:30:35.054+05:30  INFO 13128 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1
2023-10-07T16:30:35.306+05:30  INFO 13128 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationC
2023-10-07T16:30:35.310+05:30  INFO 13128 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization c
2023-10-07T16:30:36.221+05:30  INFO 13128 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with
2023-10-07T16:30:36.249+05:30  INFO 13128 --- [           main] c.s.spring.boot.springbootApplication    : Started springbootApplication in 4.397 secon
Welcome to the Spring Boot Application
2023-10-07T16:30:45.224+05:30  INFO 13128 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispa
2023-10-07T16:30:45.224+05:30  INFO 13128 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2023-10-07T16:30:45.226+05:30  INFO 13128 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 2 ms
```

Writable | Smart Insert | 26 : 1 : 404

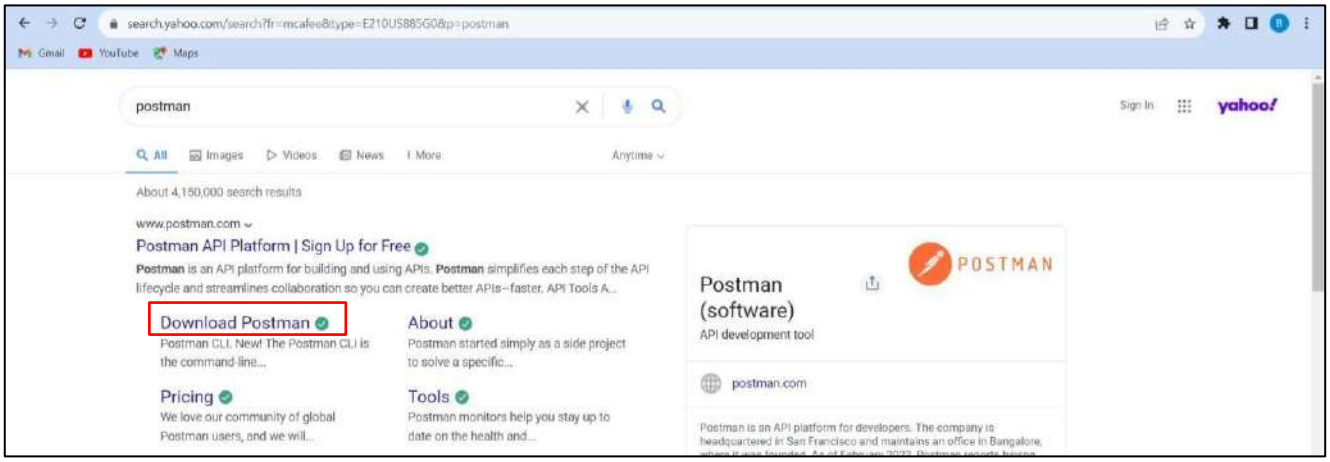
Step 23: Then go to the Web browser type **'localhost:8080/hello'** & press **Enter**.

Output:

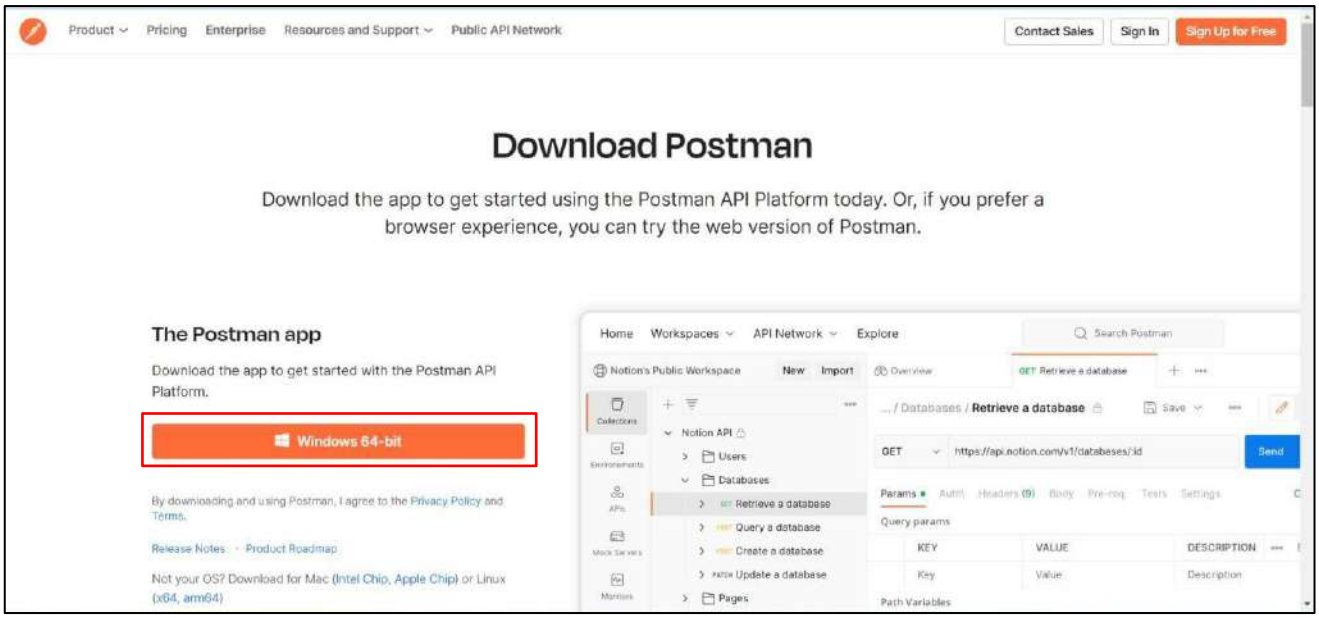
WEEK – 8

1. How to create RESTful service & Install Postman

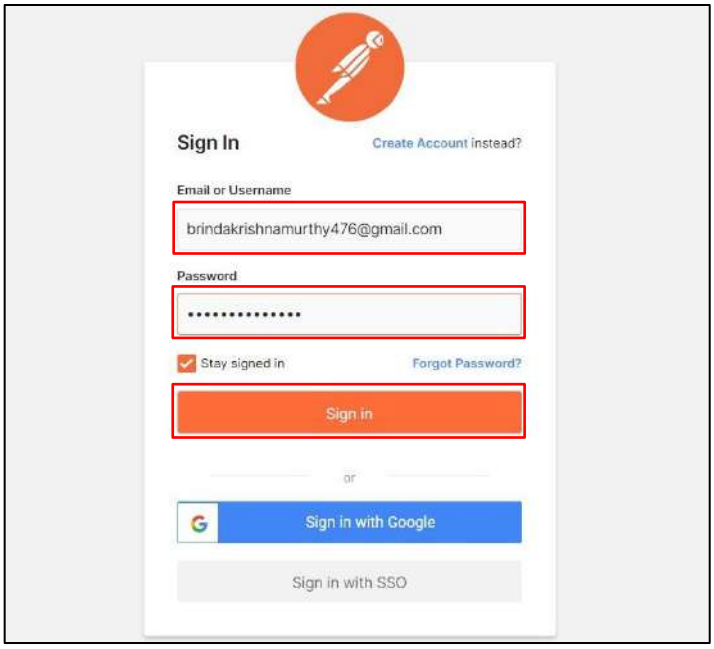
Step 1: Go to Web Browser □ search for 'Postman' □ select the second option here.



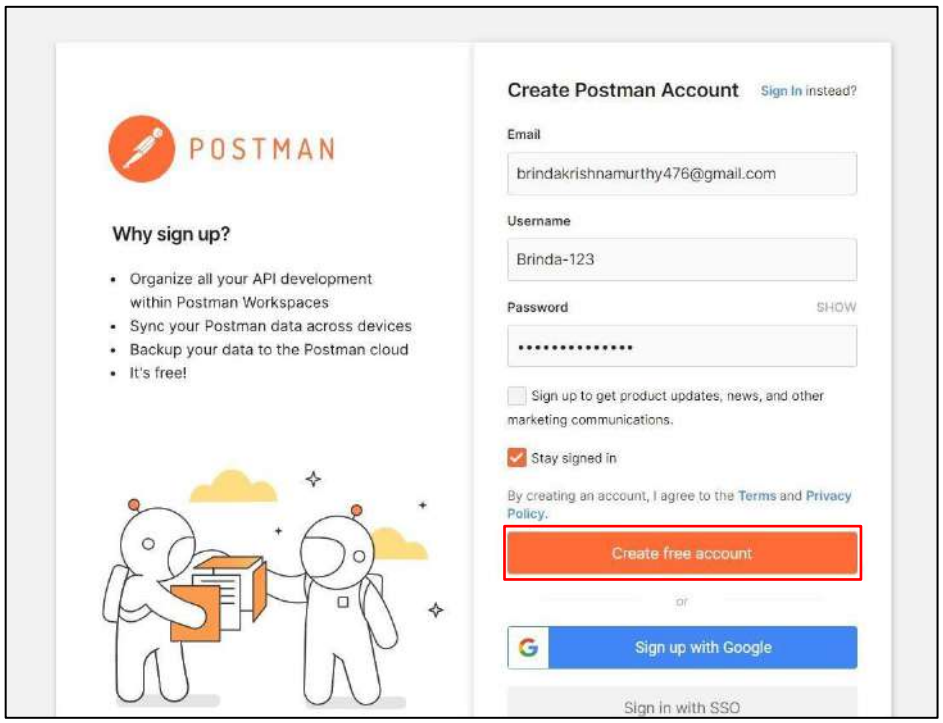
Step 2: Now click on **'Windows 64-bit'** option to download for windows OS.



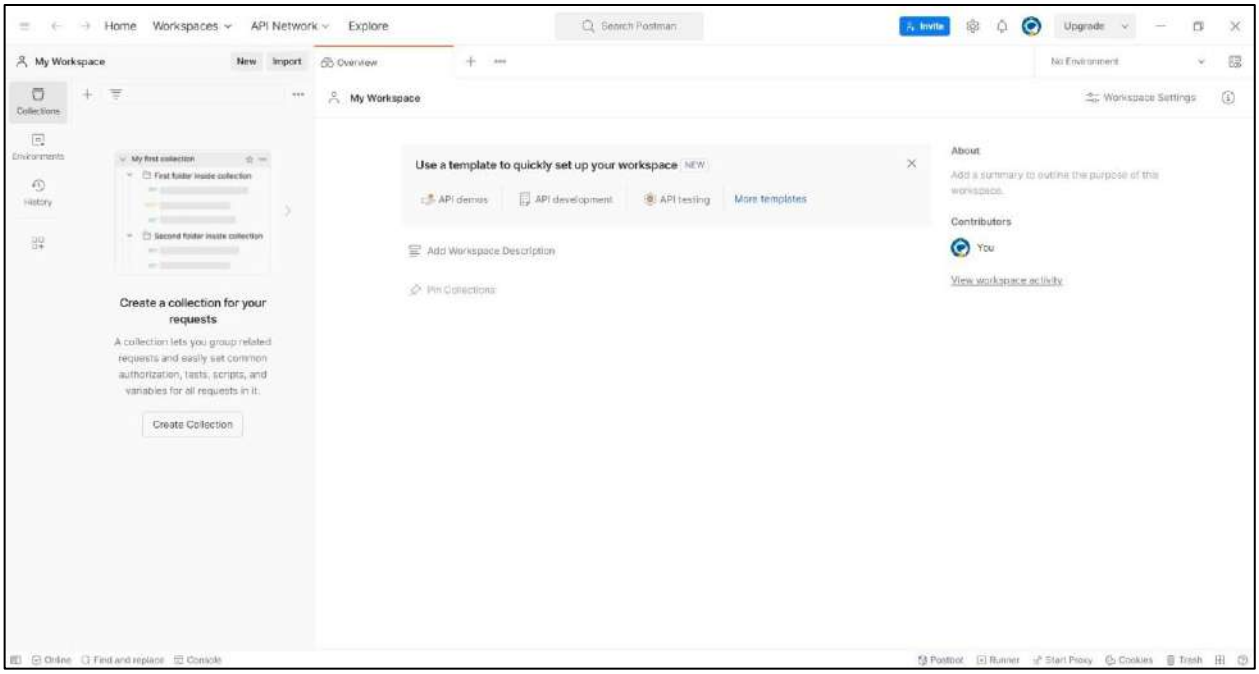
Step 3: You can simply sign to your account by giving your Email-id and Password here.



Step 4: If you don't have an account, click on **'Create Postman Account'** by giving your Email-id, Username & Password. Then click on **'Create free account'** here.



Step 5: This is your POSTMAN workspace.

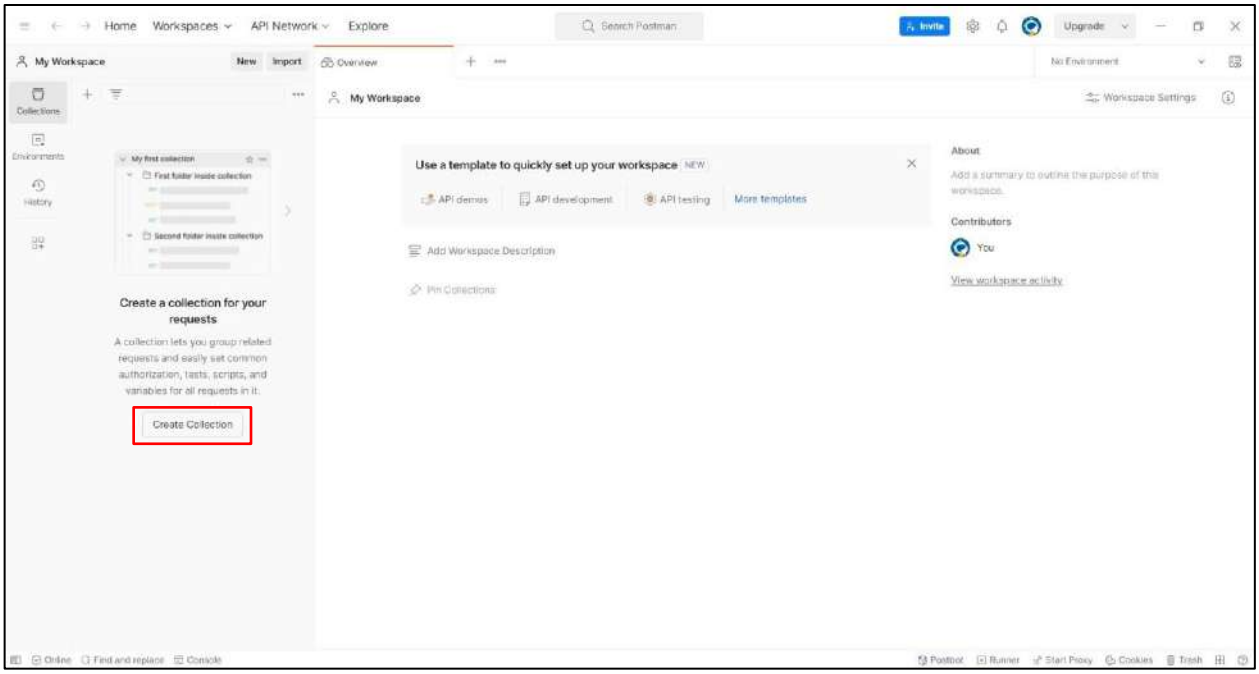


- **Test created APIs with the help of Postman**

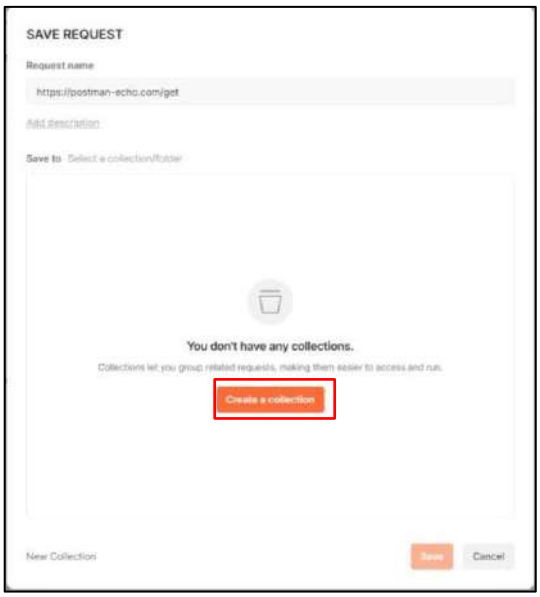
HTTP Methods:

- **HTTP GET** - Use GET requests to retrieve resource representation/information only – and not modify it in any way. As GET requests do not change the resource's state, these are said to be safe methods.
- **HTTP POST** - Use POST APIs **to create new subordinate resources**, e.g., a file is subordinate to a directory containing it or a row is subordinate to a database table.
- **HTTP PUT** - Use PUT APIs primarily to update an existing resource (if the resource does not exist, then API may decide to create a new resource or not).
- **HTTP DELETE** - As the name applies, DELETE APIs **delete the resources** (identified by the Request-URI). DELETE operations are **idempotent**. If you DELETE a resource, it's removed from the collection of resources.

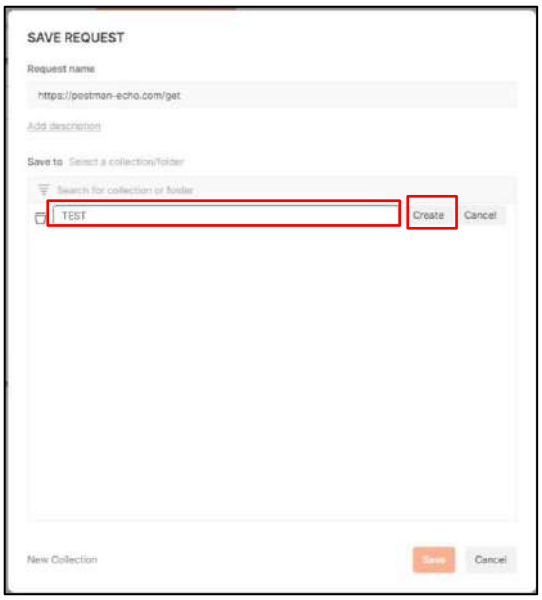
Step 1: Now click on **'Create Collection'** here.



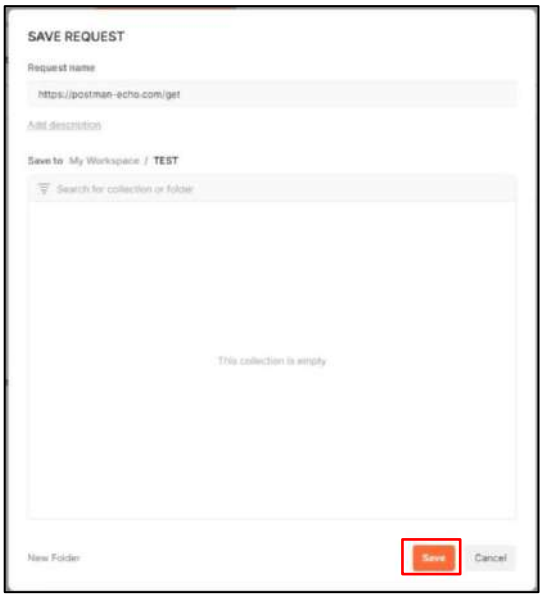
Step 2: Again click on **'Create a Collection'**.



Step 3: Give a name to your collection & click on **'Create'** option as shown below.

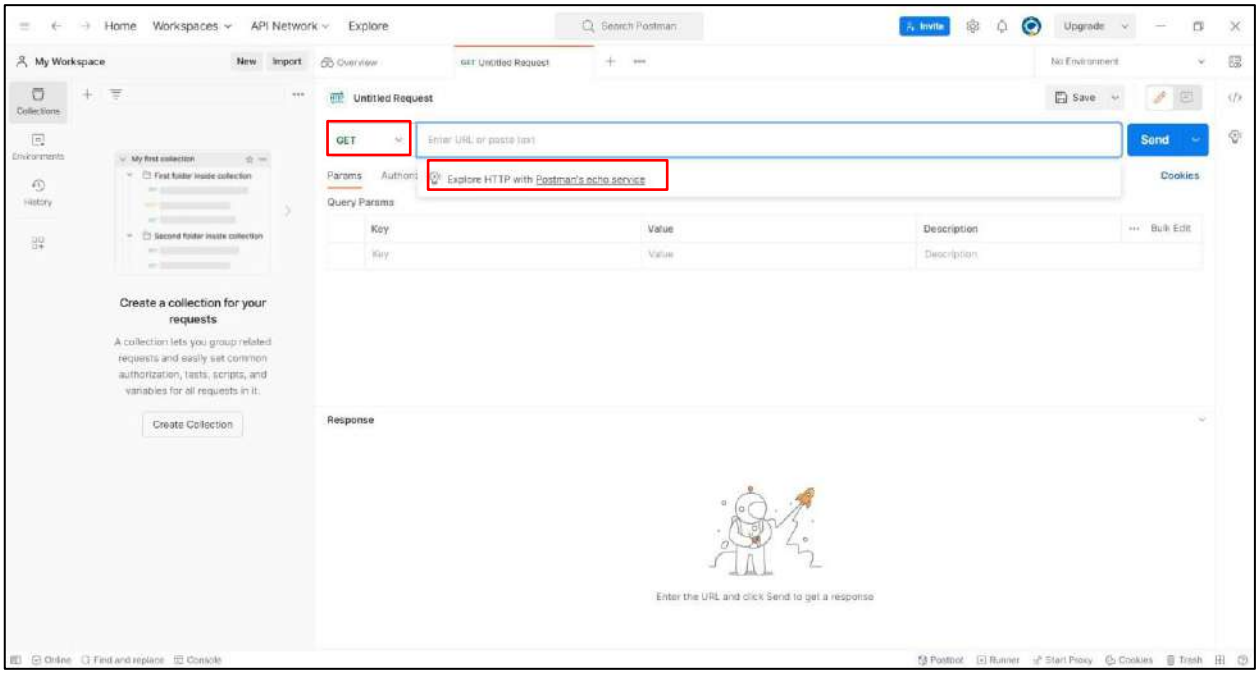


Step 4: Now click on **'Save'**.

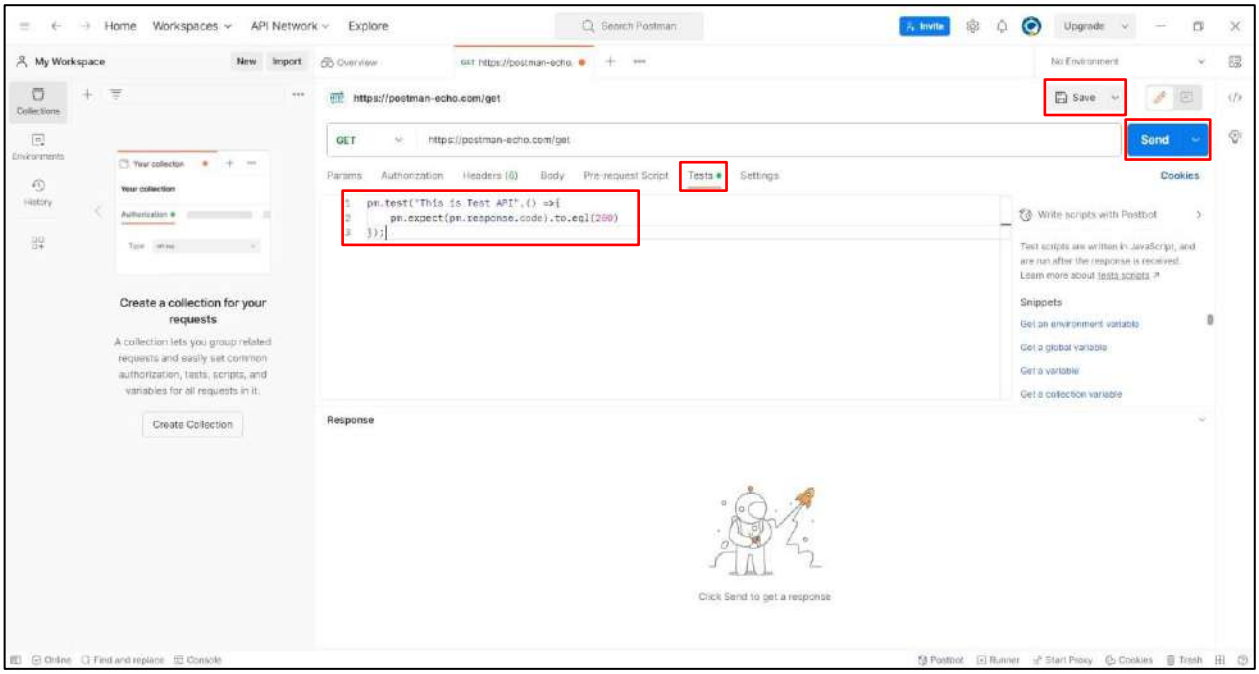


Step 5: Click on '+' option here.

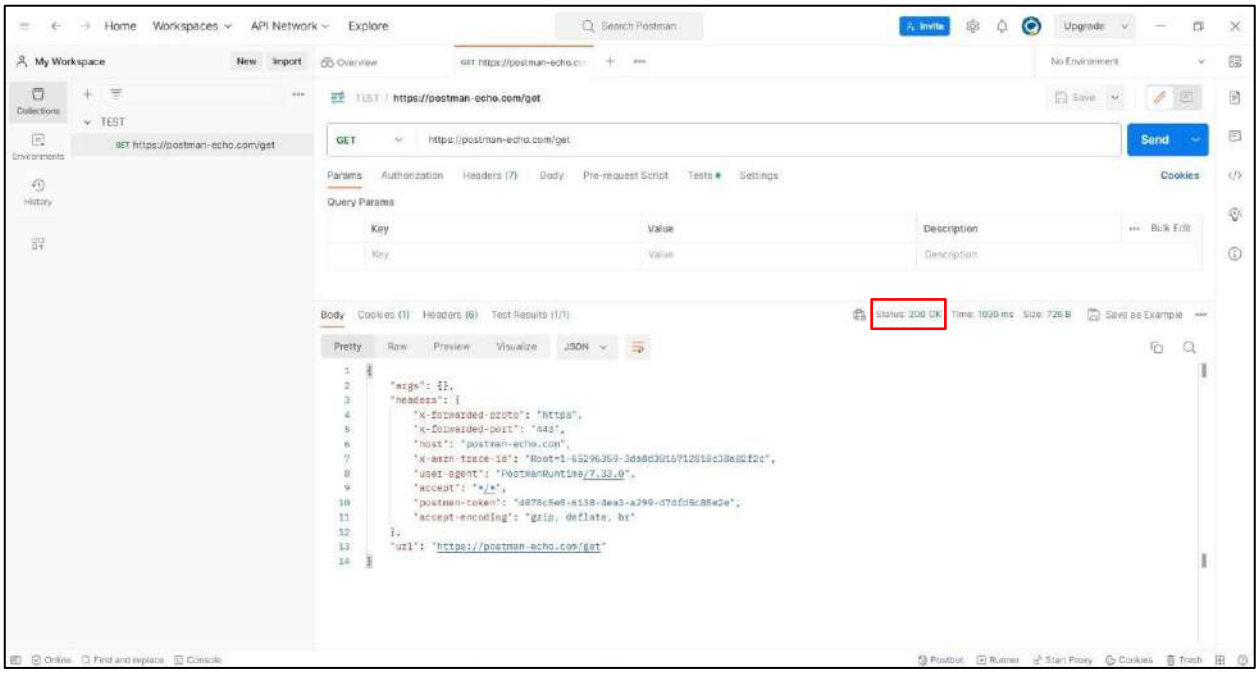
Step 6: Now select **'GET'** method & click on the HTTP link as shown below.



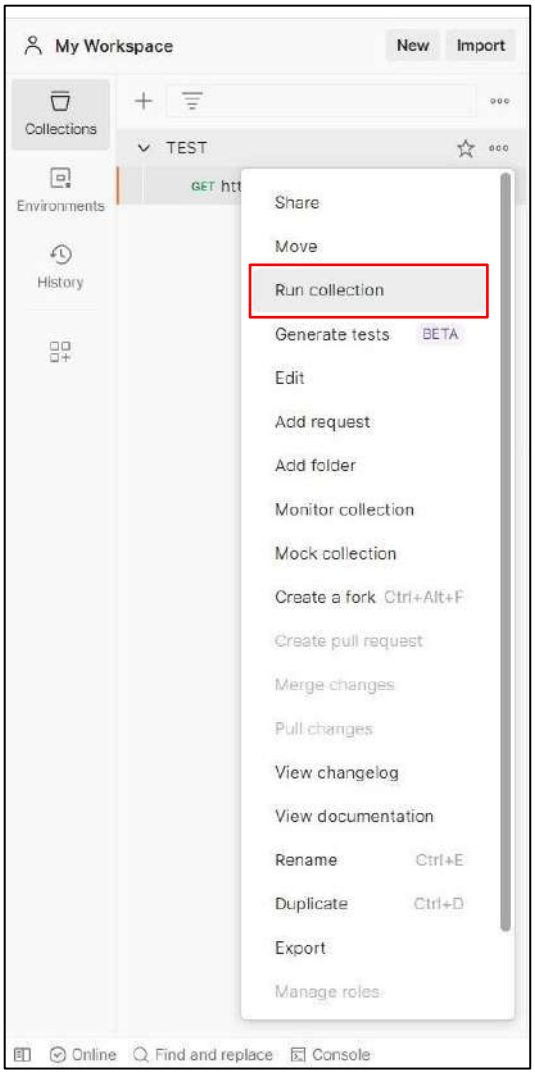
Step 7: Now go to → Tests → Enter the below code → Save it & Send.



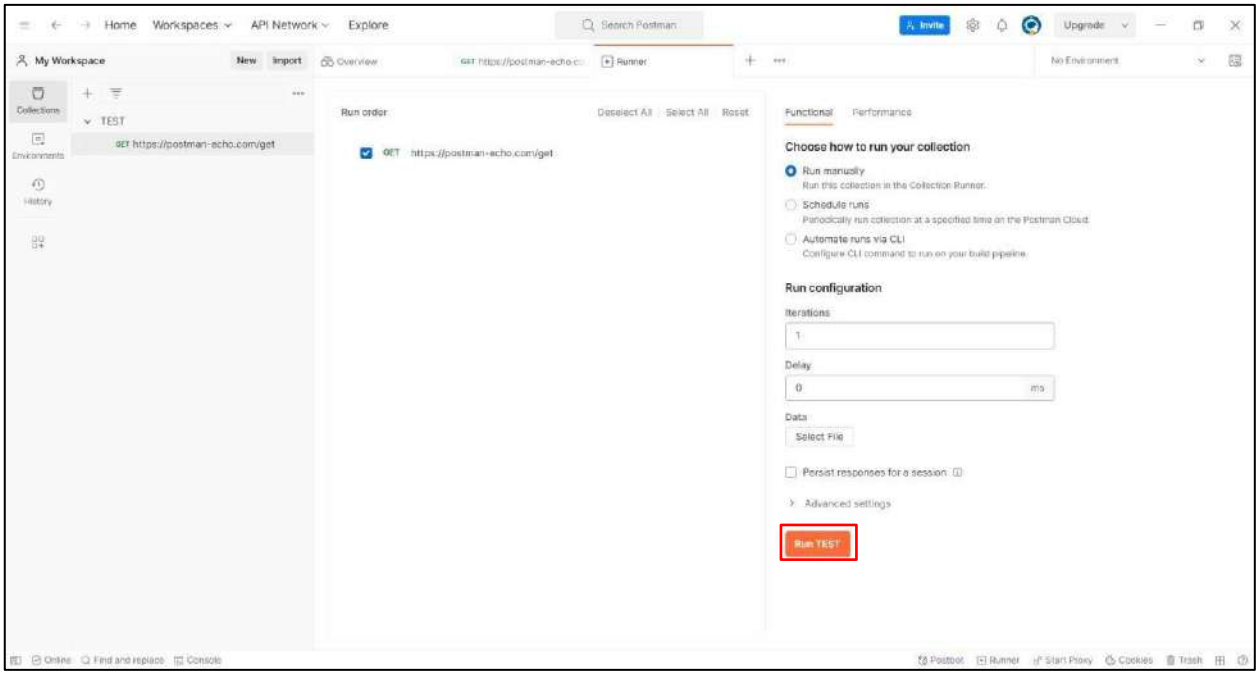
Step 8: Then check the response status here. If the status is 200 OK means the response has been accepted by the POSTMAN.



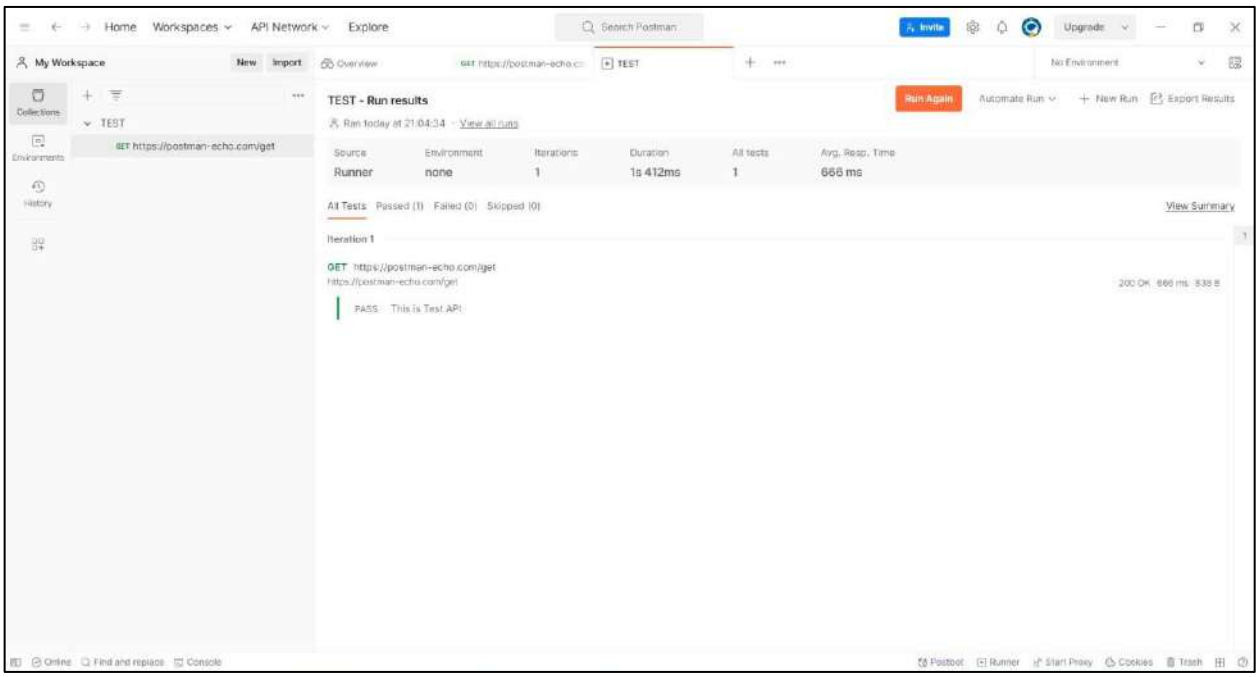
Step 9: Now right click on the new collection created & select **'Run Collection'**.



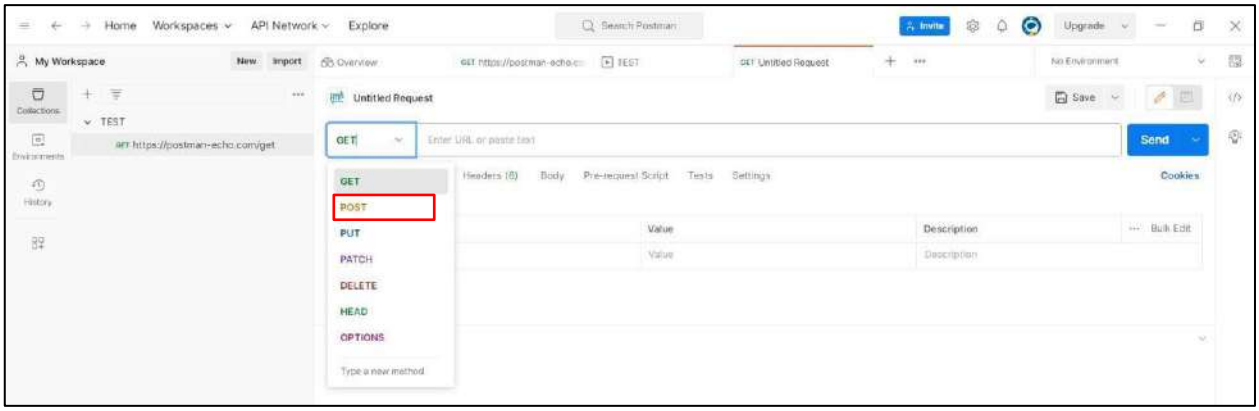
Step 10: Then click on **'Run Test'** here.



Step 11: Now the **TEST – Run results** output is displayed as shown below.

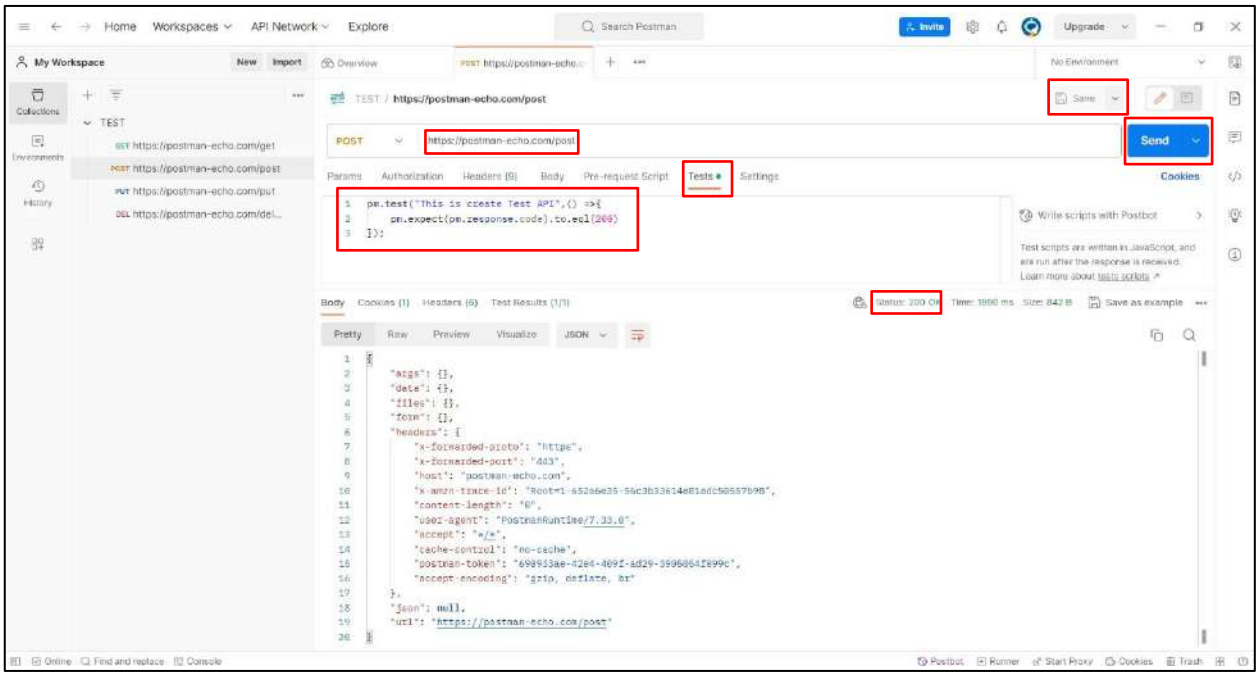


Step 12: Now select **'+'** option & select **'POST'** method here.



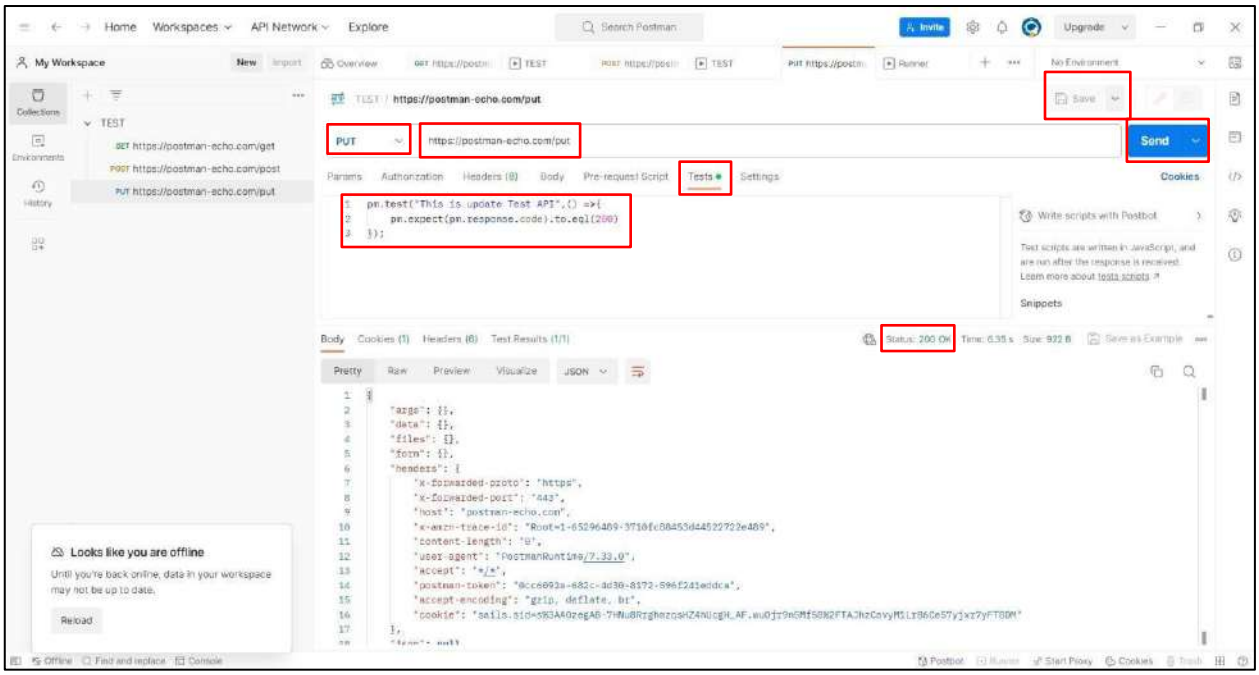
Step 13: Now select **'POST'** method & click on the HTTP link as shown below & then go to → Tests →

Enter the below code → Save it & Send.

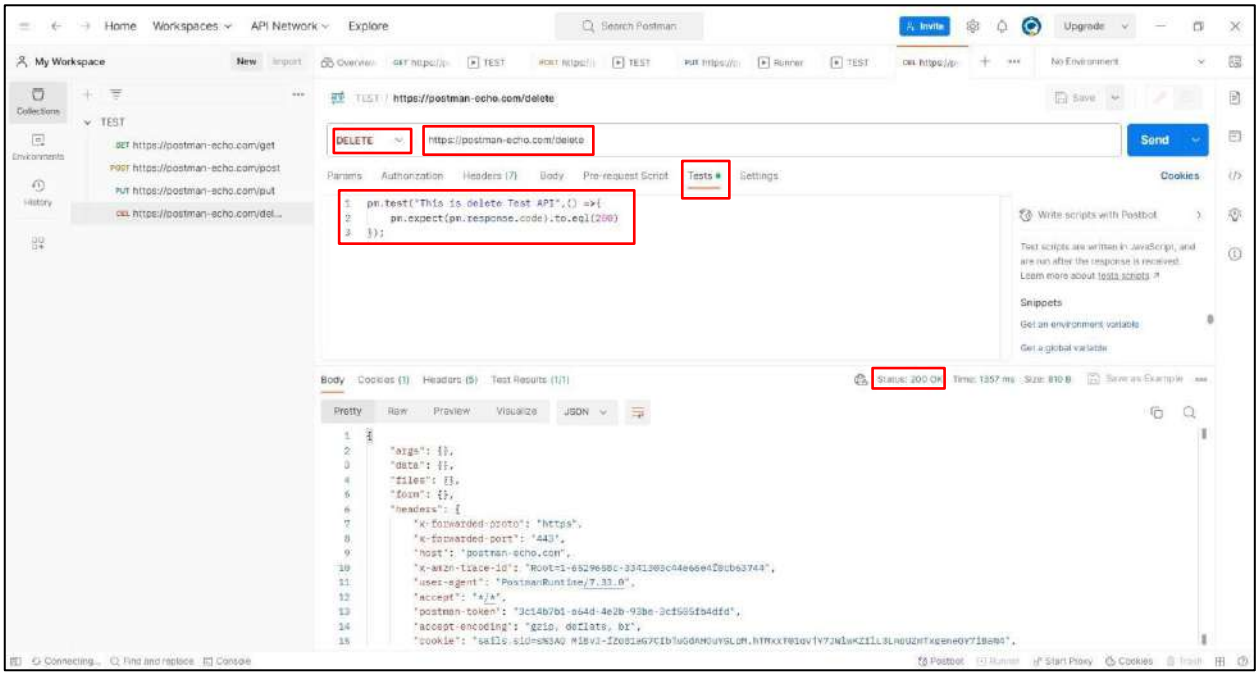


Now perform the same procedures for other two HTTP methods as shown below.

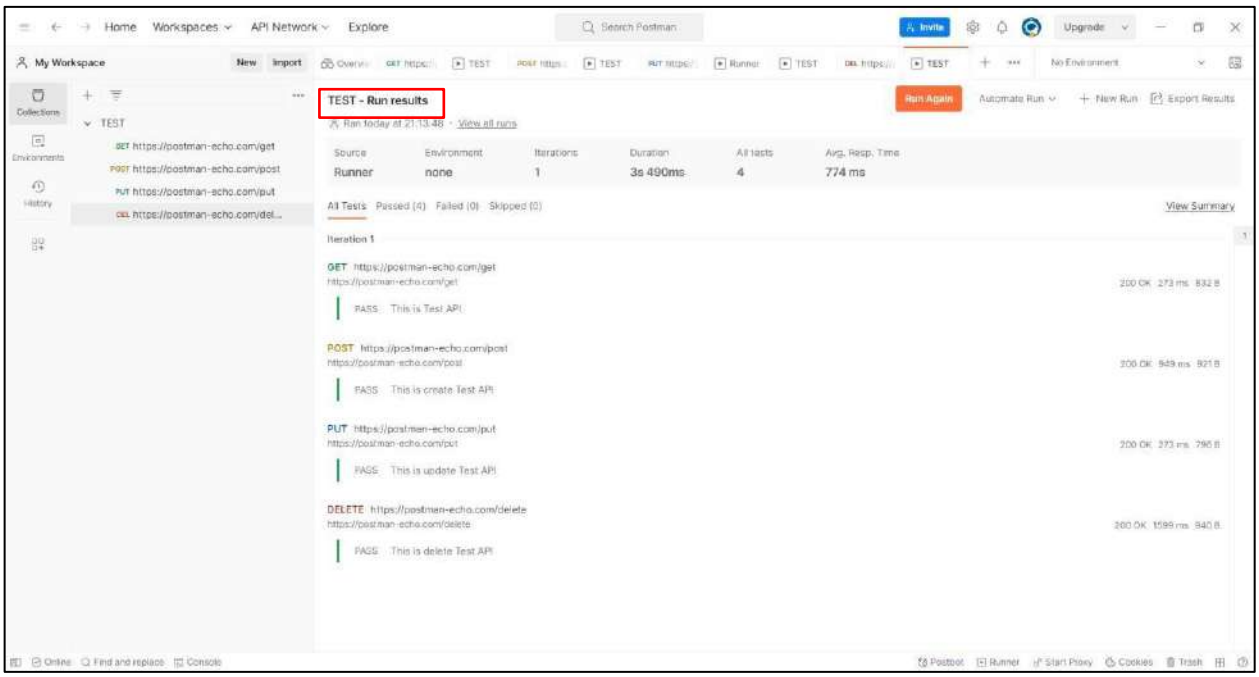
Step 14: Now select **'PUT'** method & click on the HTTP link as shown below & then go to → Tests → Enter the below code → Save it & Send.



Step 15: Now select **'DELETE'** method & click on the HTTP link as shown below & then go to → Tests → Enter the below code → Save it & Send.

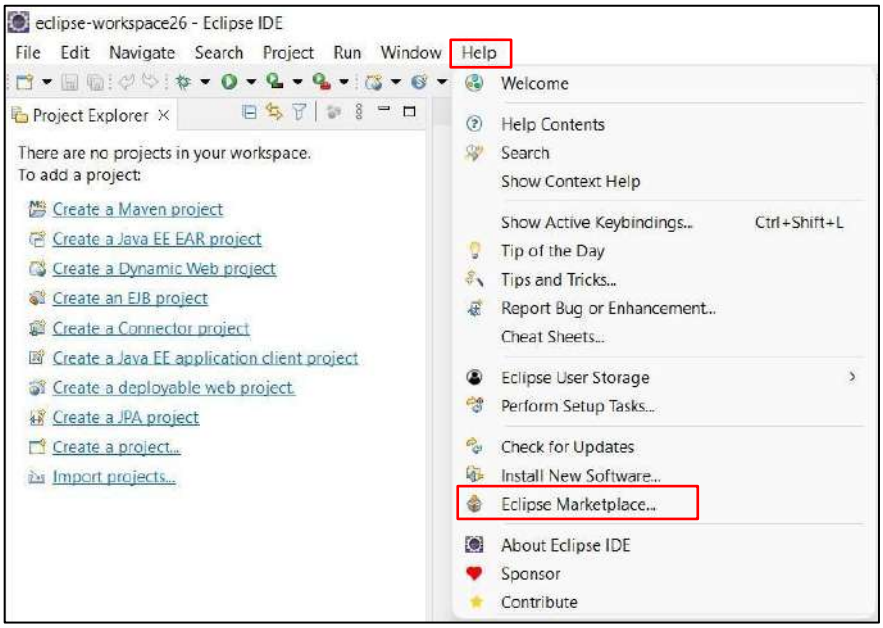


Step 16: Now the **TEST – Run results** outputs are displayed as shown below.

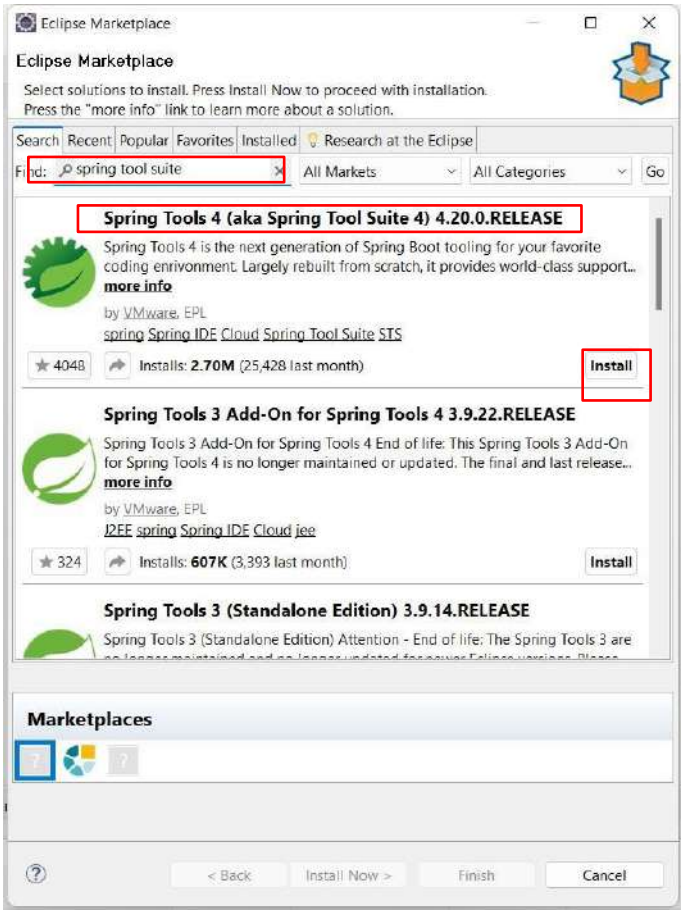


2. <u>Create REST controller for CRUD operations Versioning Spring REST APIs</u>

Step 1: Open **'Eclipse IDE for Enterprise edition'** & go to -→ Help → Eclipse Marketplace as shown below.

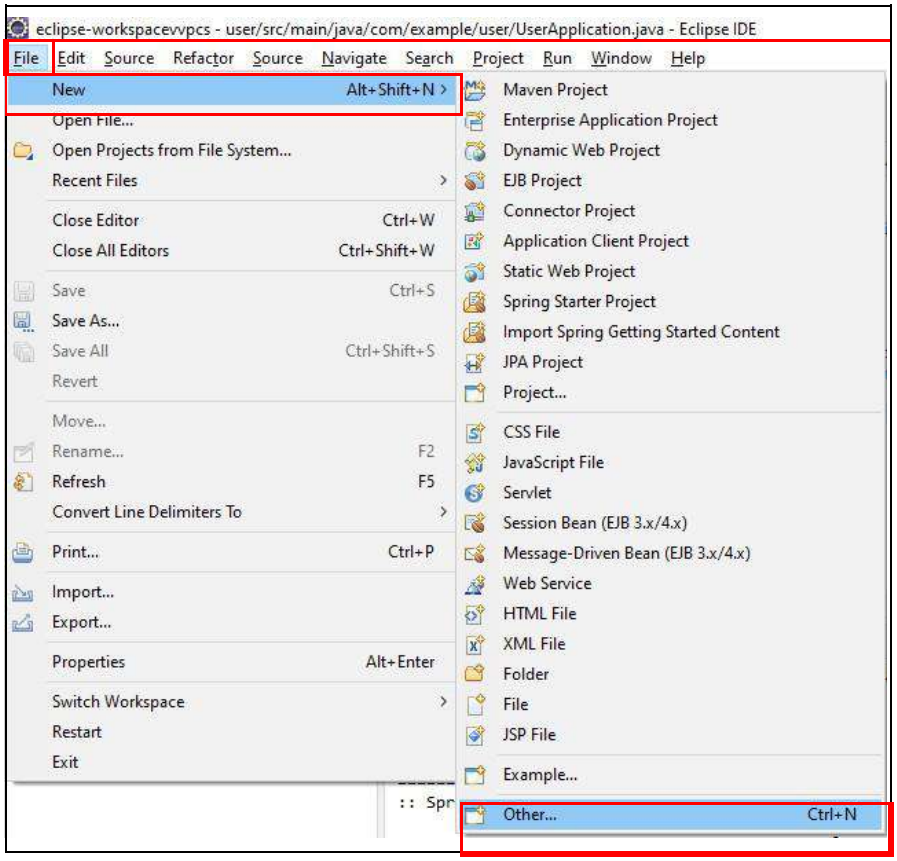


Step 2: Now search for **'Spring tool suite'** & click on **'Install'** to install the latest version.

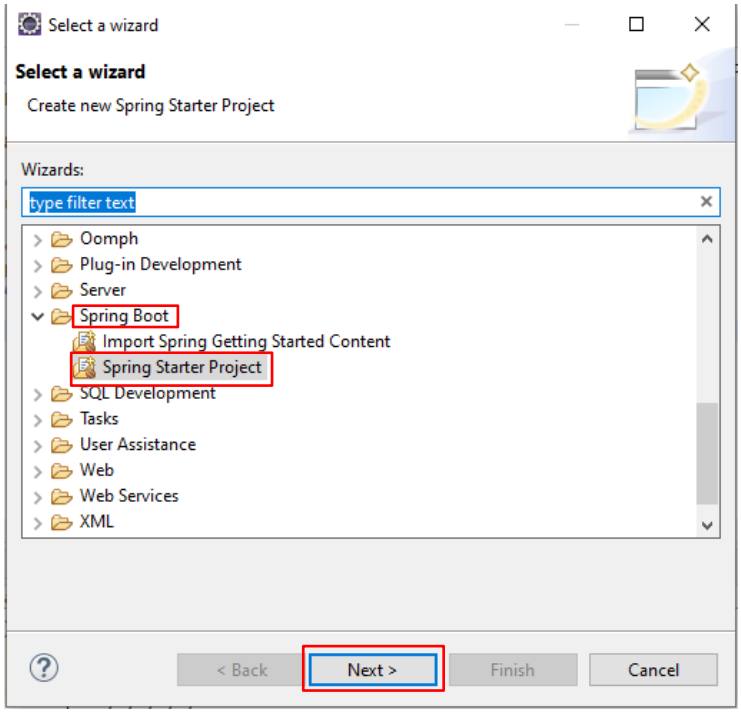


Configure using SpringBoot application property file

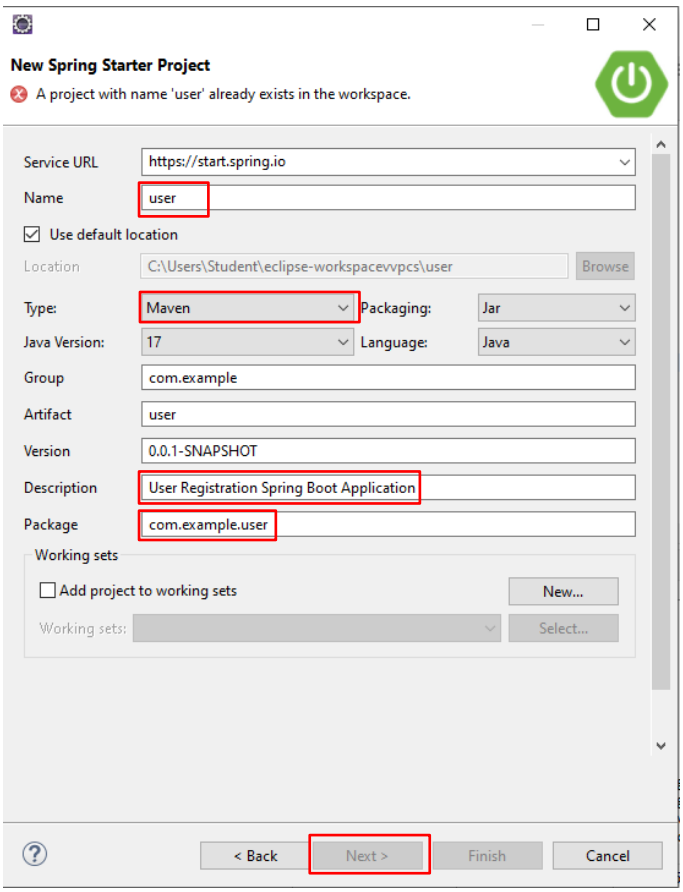
Step 3: After the installation is complete, go to → File → New → Other option as shown below.



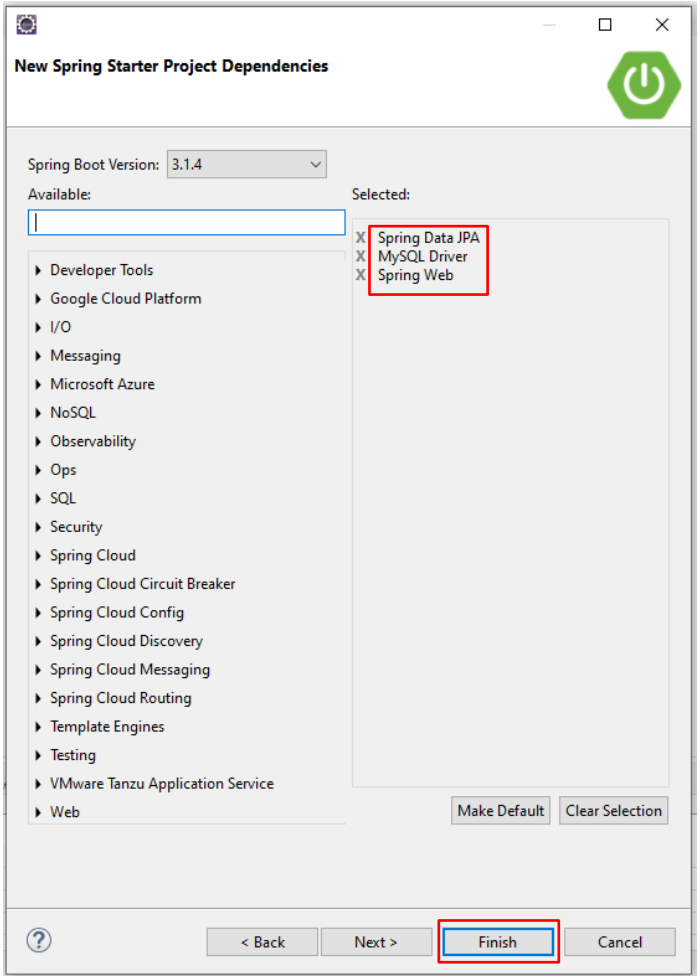
Step 4: Now select **'Spring Starter Project'** & click on **'Next'** option.



Step 5: Give the Name as **'User'** → select **'Maven'** → Description as **User Registration Spring Boot Application** → Package as **'com.example.user'** & then click on **'Next'**.



Step 6: Now add all the required dependencies as shown below & click on **'Finish'**.



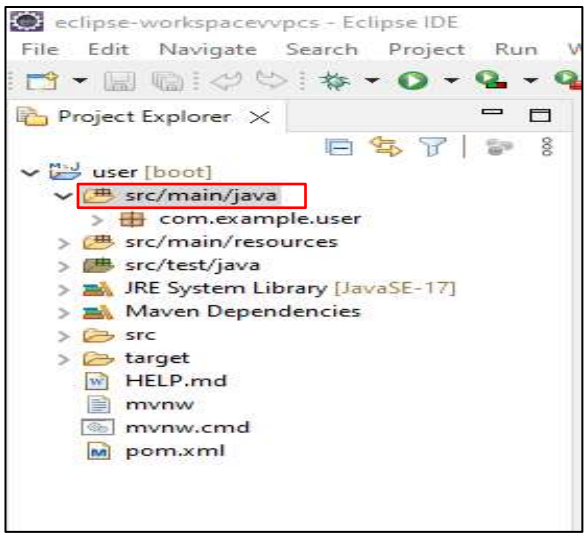
Spring REST – creating Spring REST controller

- Controller Layer (handling request and responses)

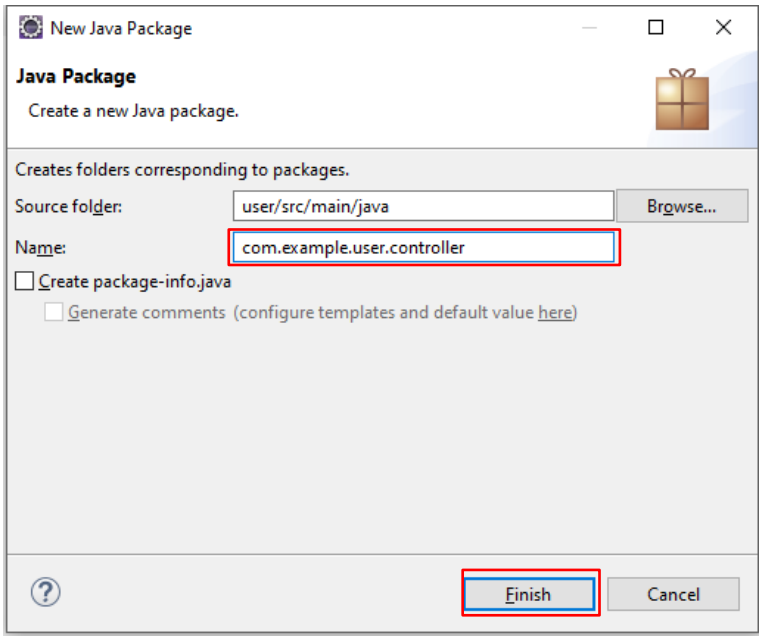
- Service Layer (Application business logic)

- Repository layer (Communicate with DB)

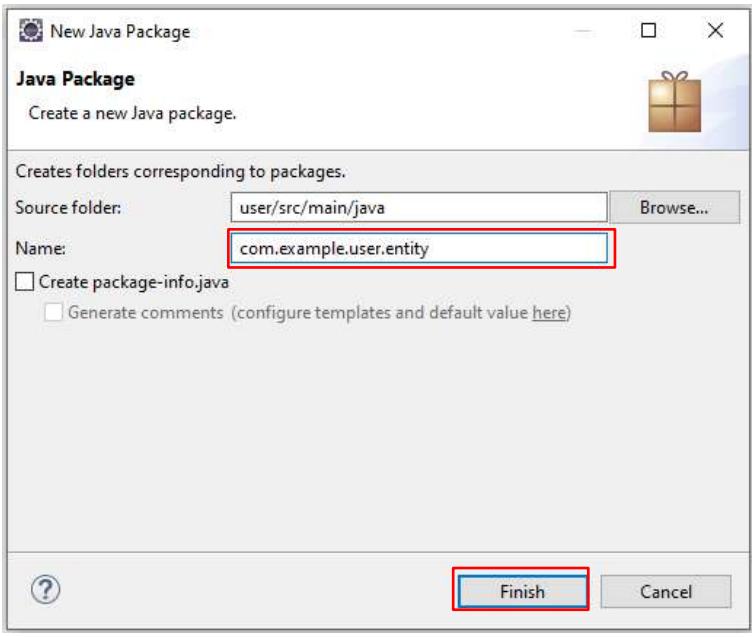
Step 7: Then select the **'src'** folder □ New □ Package.



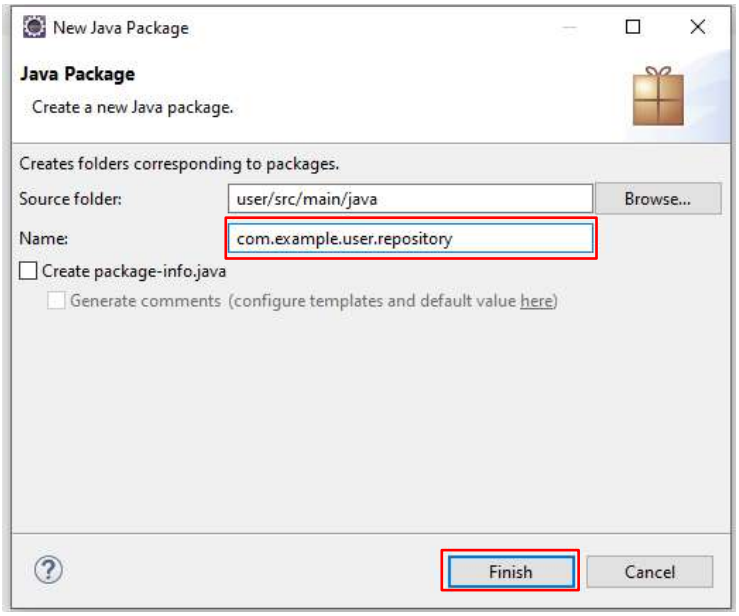
Step 8: Create a new package named **'com.example.user.controller'** & click on **'Finish'**.



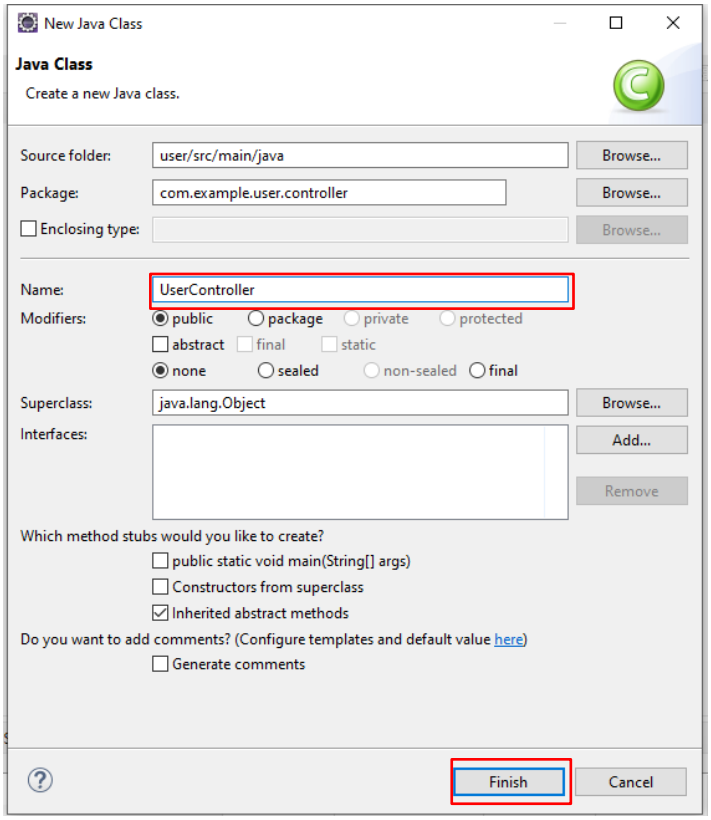
Step 9: Create a new package named **'com.example.user.entity'** & click on **'Finish'**.



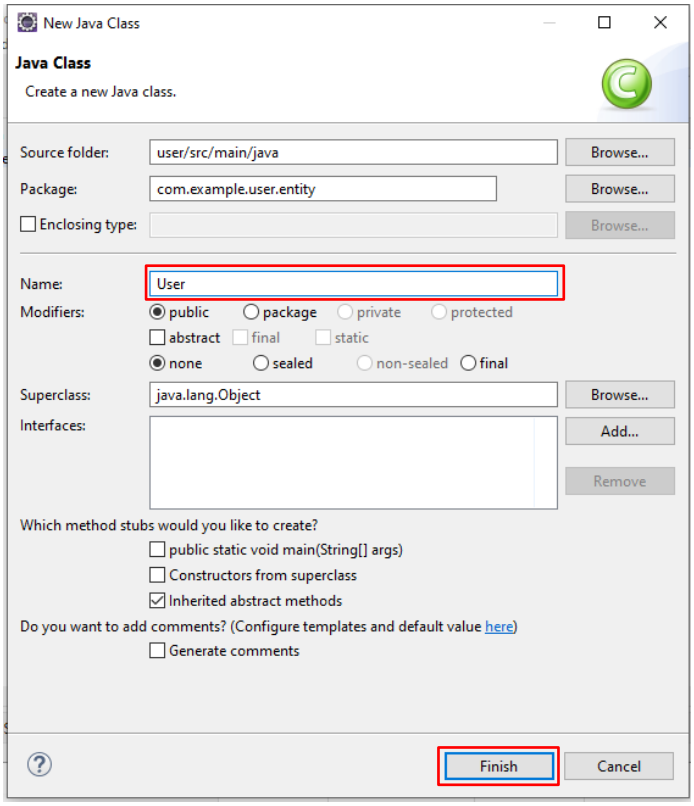
Step 10: Create a new package named **'com.example.user.repository'** & click on **'Finish'**.



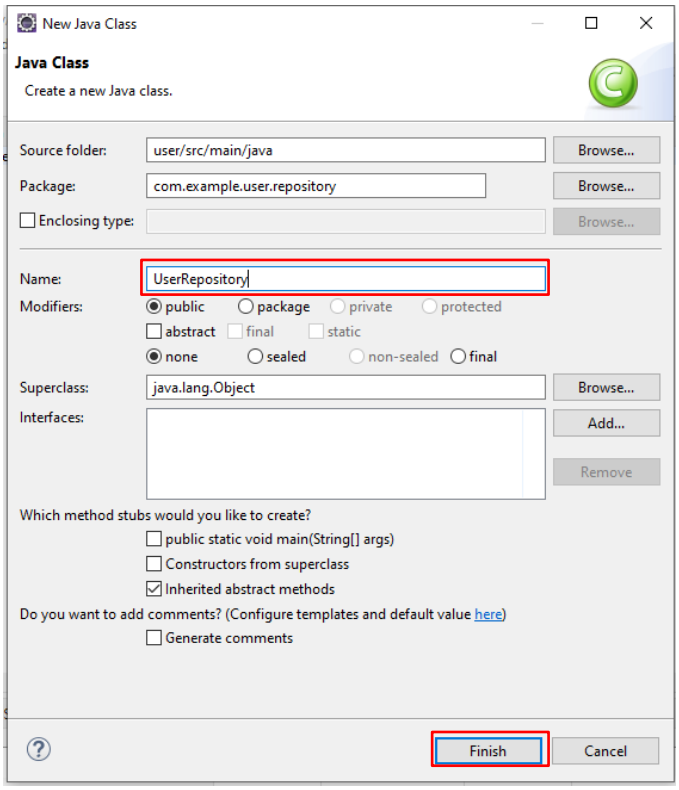
Step 11: Now right click on the **'com.example.user.controller'** package New Class & create a new class named **'UserController'** & then click on **'Finish'**.



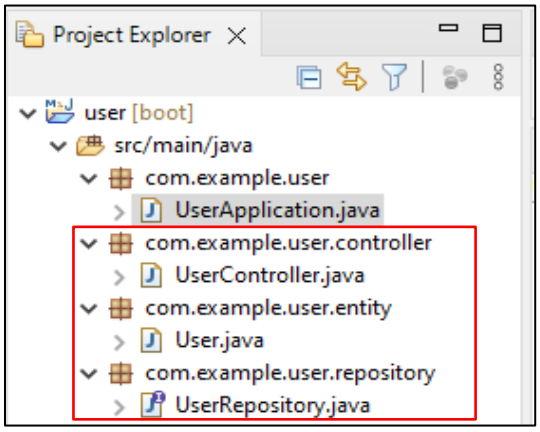
Step 12: Now right click on the **'com.example.user.entity'** package New Class & create a new class named **'User'** & then click on **'Finish'**.



Step 13: Now right click on the **'com.example.user.repository'** package □ New □ Interface & create a new class named **'UserRepository'** & then click on **'Finish'**.



Step 14: All the created packages and classes are displayed here.



Step 15: Now add code to **UserController.java** file as shown below.

UserController.java | User.java | UserRepository.java | application.properties | UserApplication.java

```java
package com.example.user.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.user.entity.User;
import com.example.user.repository.UserRepository;

@RestController
@RequestMapping("/users")
public class UserController {
@Autowired
private UserRepository userRepository;

@GetMapping
public List<User> getAllUser()
{
    return this.userRepository.findAll();
}

@GetMapping("/{id}")
public User getUserById(@PathVariable(value="id") long userId) {
    return this.userRepository.findById(userId).orElseThrow();
}
```

```java
@PostMapping
public User createUser(@RequestBody User user)
{
    return this.userRepository.save(user);
}
@PutMapping("/{id}")
public User updateUser(@RequestBody User user,@PathVariable("id") long userId)
{
    User ex=this.userRepository.findById(userId).orElseThrow();
    ex.setFirstname(user.getFirstname());
    ex.setlastname(user.getlastname());
    return this.userRepository.save(ex);
}
@DeleteMapping("/{id}")
public ResponseEntity<User> deleteUser(@PathVariable("id") long userId)
{
    User ex=this.userRepository.findById(userId).orElseThrow();
    this.userRepository.delete(ex);
    return ResponseEntity.ok().build();
}
}
```

Step 16: Now add code to **User.java** file as shown below.

UserController.java | *User.java | UserRepository.java | application.properties | UserApplication.java

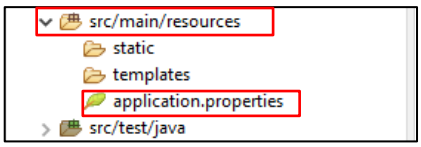
```java
package com.example.user.entity;

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

@Entity
@Table(name="user")
public class User {
    public User () {
    }
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;
    private String firstname;
    private String lastname;
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getFirstname() {
        return firstname;
    }
    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
    public User(Long id, String firstname, String lastname) {
        super();
        this.id = id;
        this.firstname = firstname;
        this.lastname = lastname;
    }
    public String getlastname() {
        return lastname;
    }
    public void setlastname(String lastname) {
        this.lastname = lastname;
    }

}

```

Step 17: Now add code to **UserRepository.java** file as shown below.

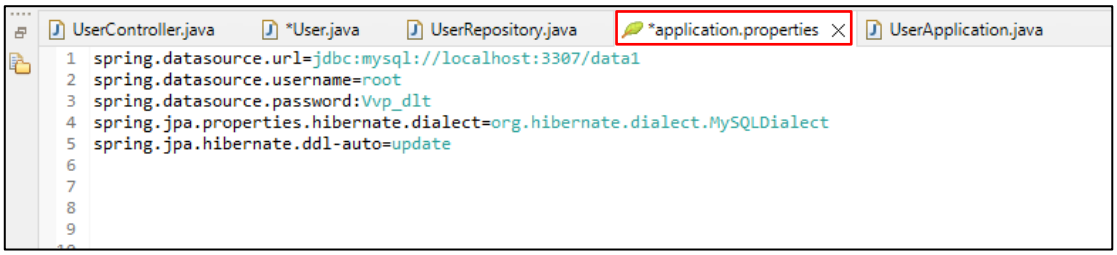
UserController.java | *User.java | UserRepository.java | application.properties | UserApplication.java

```java
package com.example.user.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.user.entity.User;

@Repository
public interface UserRepository extends JpaRepository<User,Long>
{
}

```

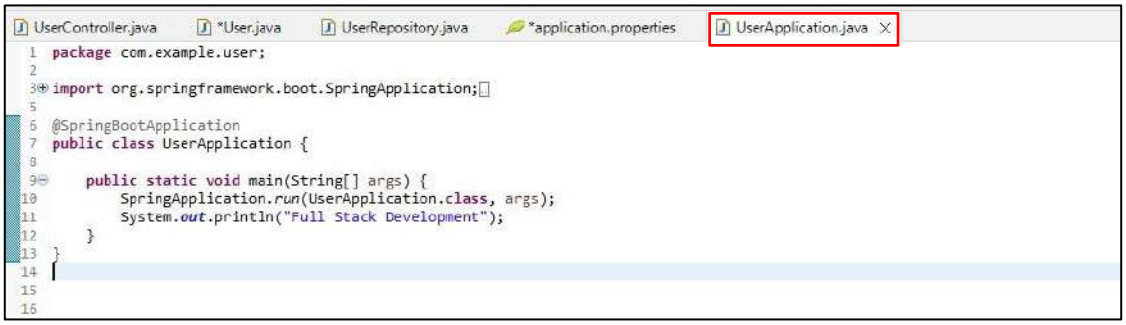
Step 18: Now go to **'src/main/resources'** folder □ select **'application.properties'** file here.



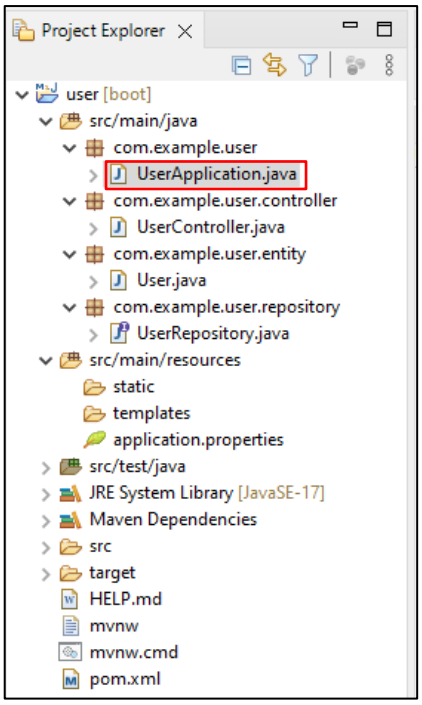
Step 19: Now add code to **application.properties** file as shown below.



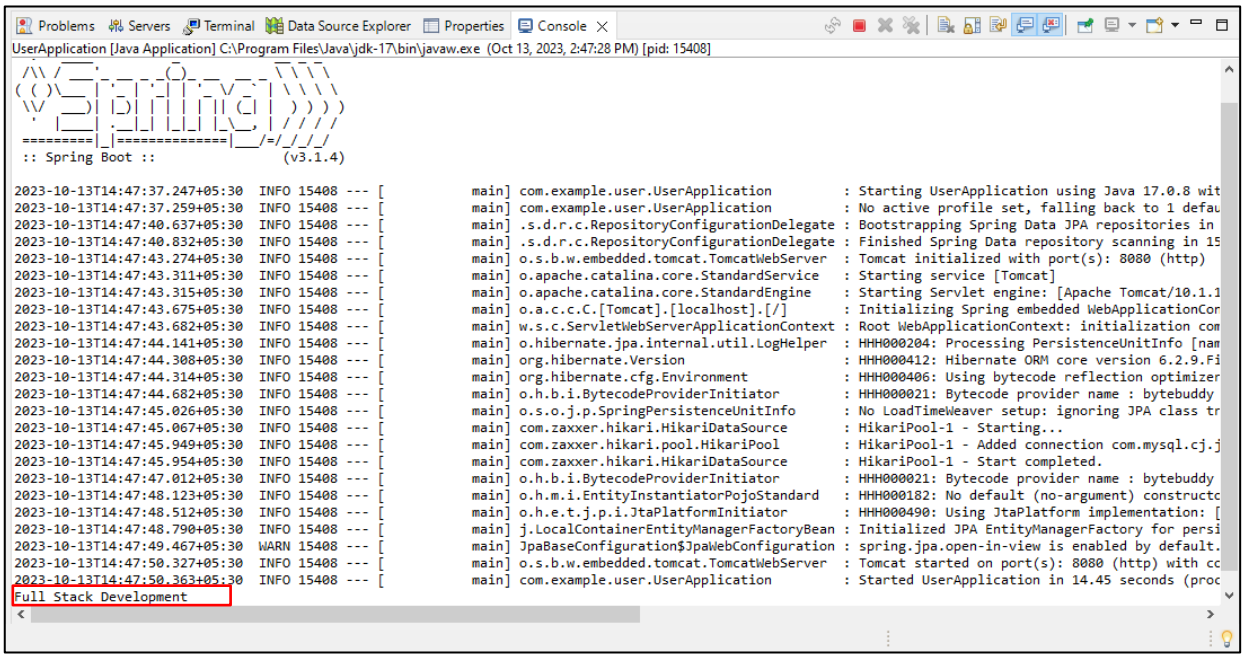
Step 20: Now add code to **UserApplication.java** file which includes the main method as shown below.



Step 21: Go to **src/main/java** □right click on the **UserApplication.java** □ **Run as** □ **1java application.**

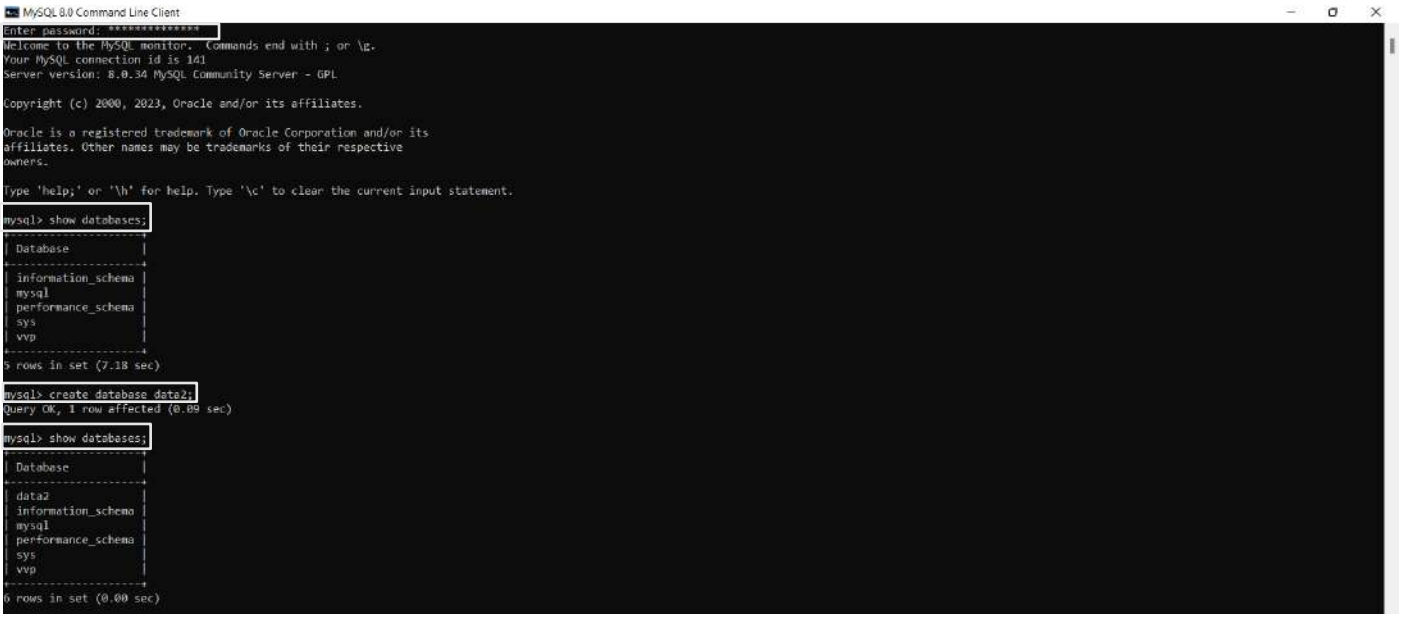


Step 22: Now the server is running successfully.



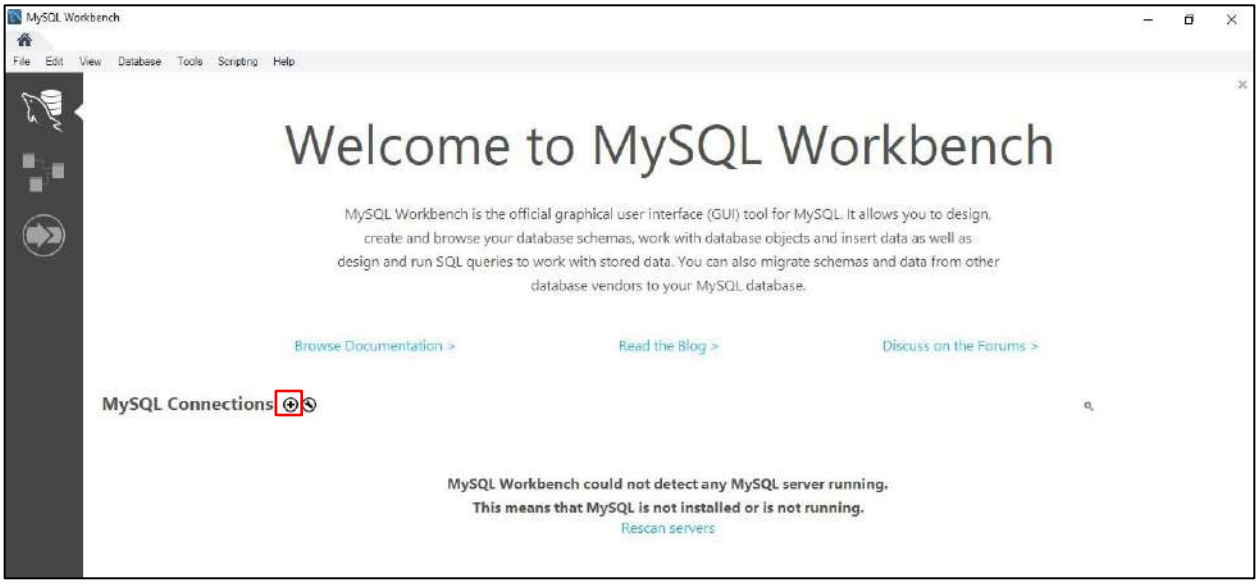
- **Creation of database using MySQL Command Line Client**

Step 23: Enter the password created→ give the command as **show databases;** to view all the inbuilt databases →now enter the command as **create database data1;** to create a new database with any name→ again enter **show databases;** to view the newly created database as shown below.

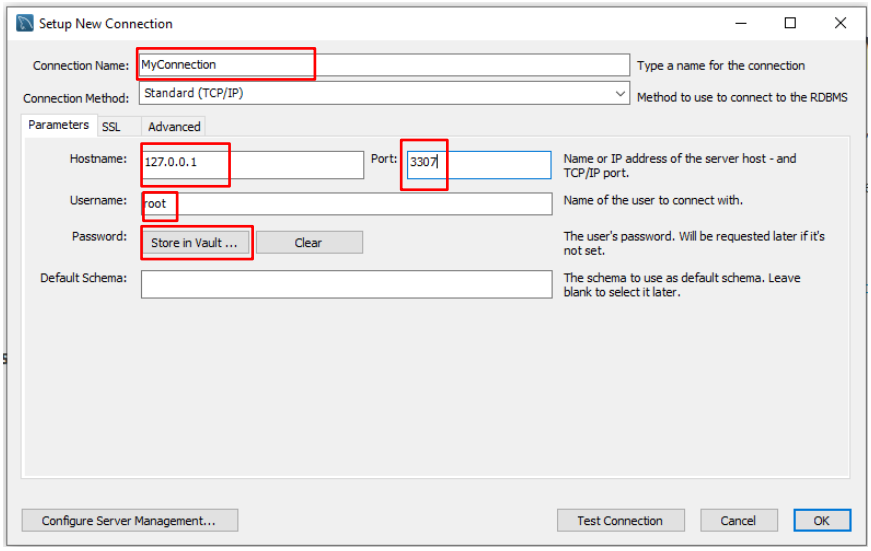


- **Connection of newly created database to the MySQL Workbench.**

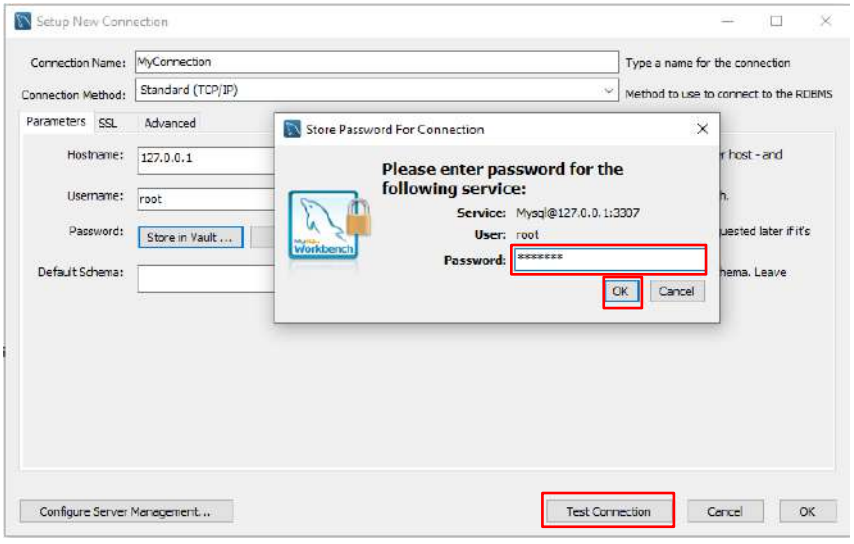
Step 24: Go to '+' option to add new connection.



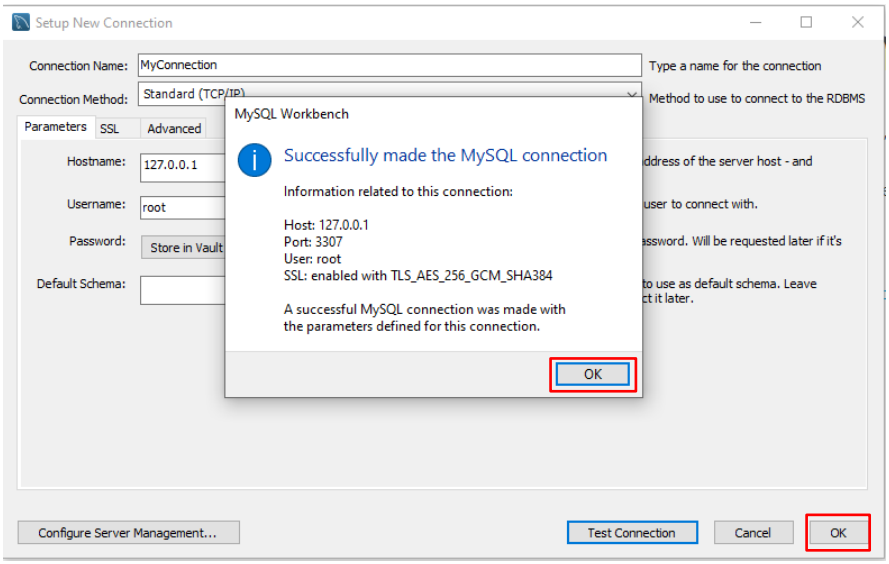
Step 25: Give a name to your connection as 'MyConnection' □ select the Hostname as 'localhost' or select the default ip address of your local computer □ give the port no. which you've entered while installation of MySQL Workbench (3307) □ Username as 'root'.



Step 26: Then enter the Password – which was created while installing the MySQL Workbench & click on **'OK'** & then click on **'Test connection'**.



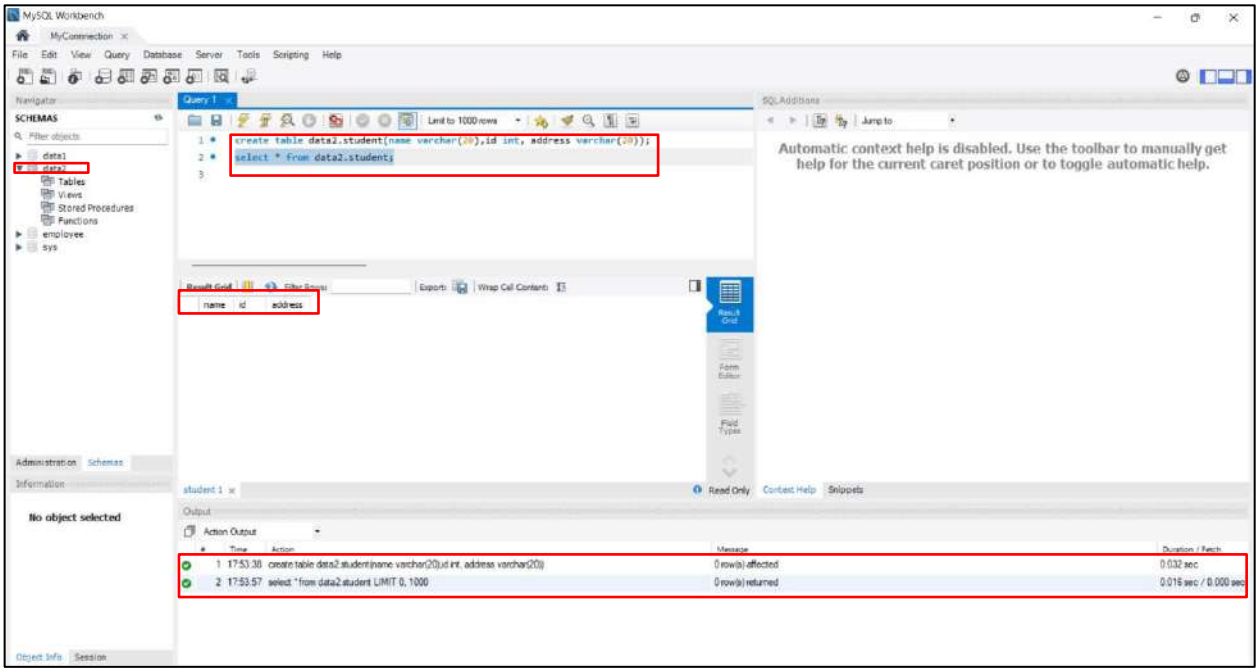
Step 27: Now click on **'OK'** option as shown below.



- Creation of simple table.

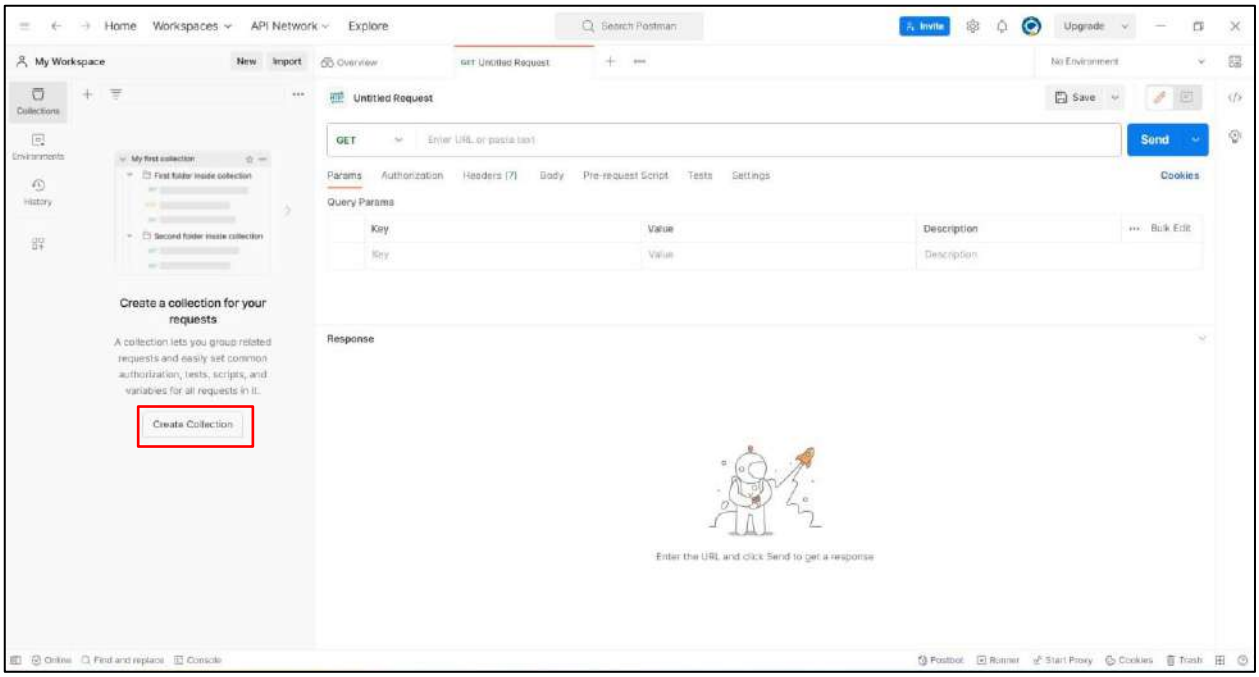
Step 28: Now enter the below code to create a simple table as shown below.

Note: To run the code, select each code line separately & press **ctrl+shift+enter** key to view the output.

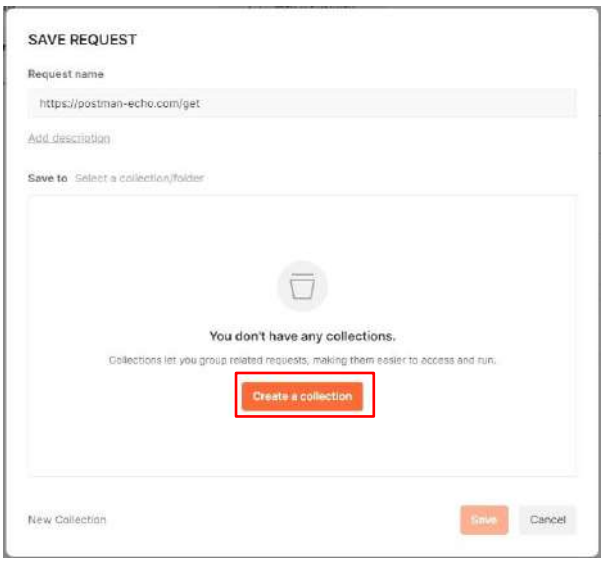


- **<u>Performing CRUD operations using Postman RESTful Service.</u>**

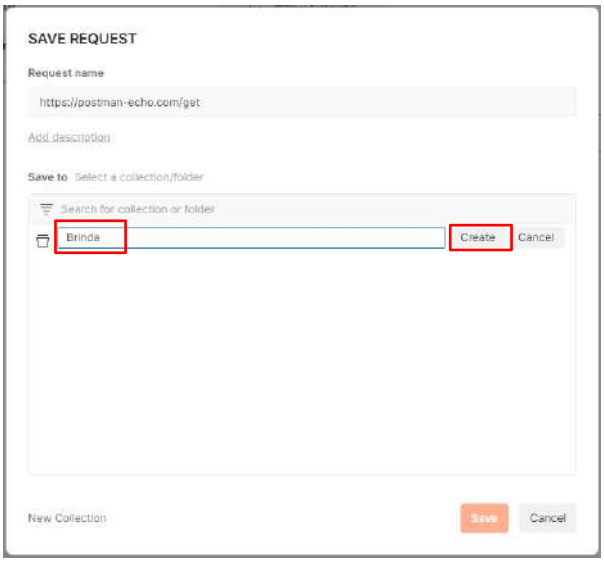
Step 29: Login to the Postman & click on **'Create collection'** here.



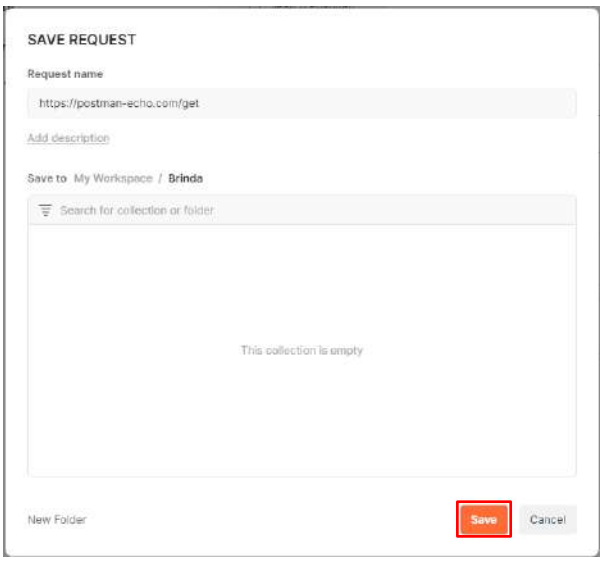
Step 30: Again click on **'Create a collection'** here.



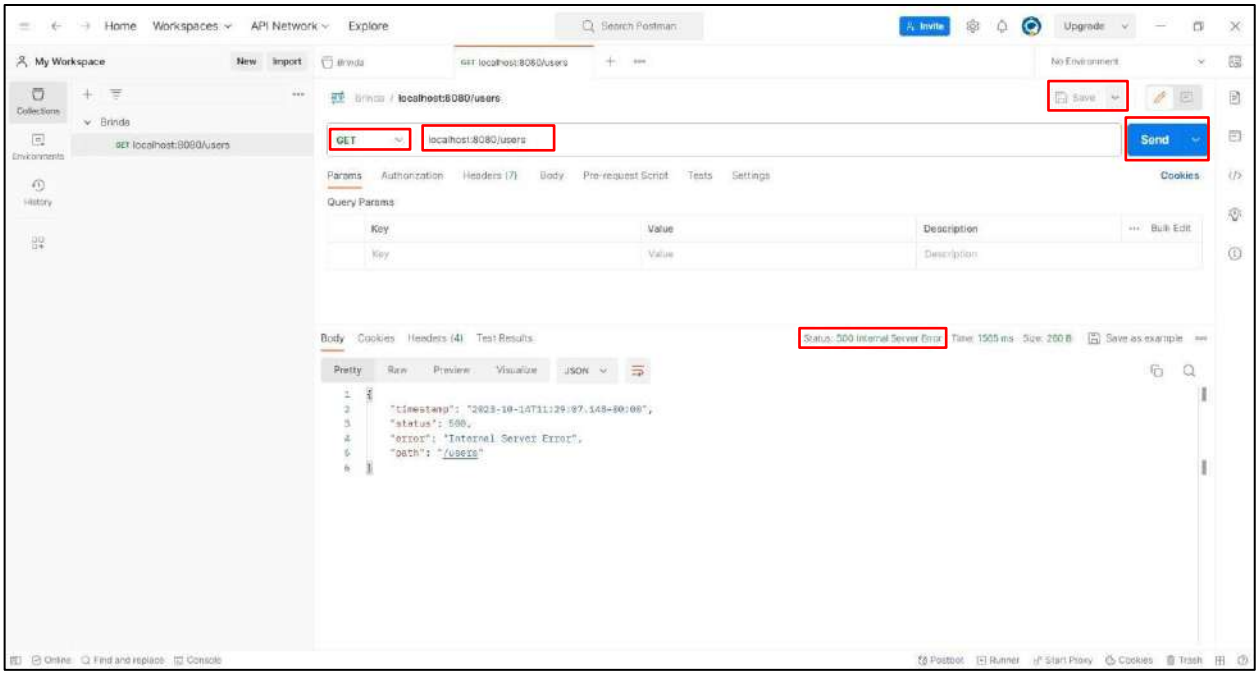
Step 31: Give a name to your collection & click on **'Create'** option here.



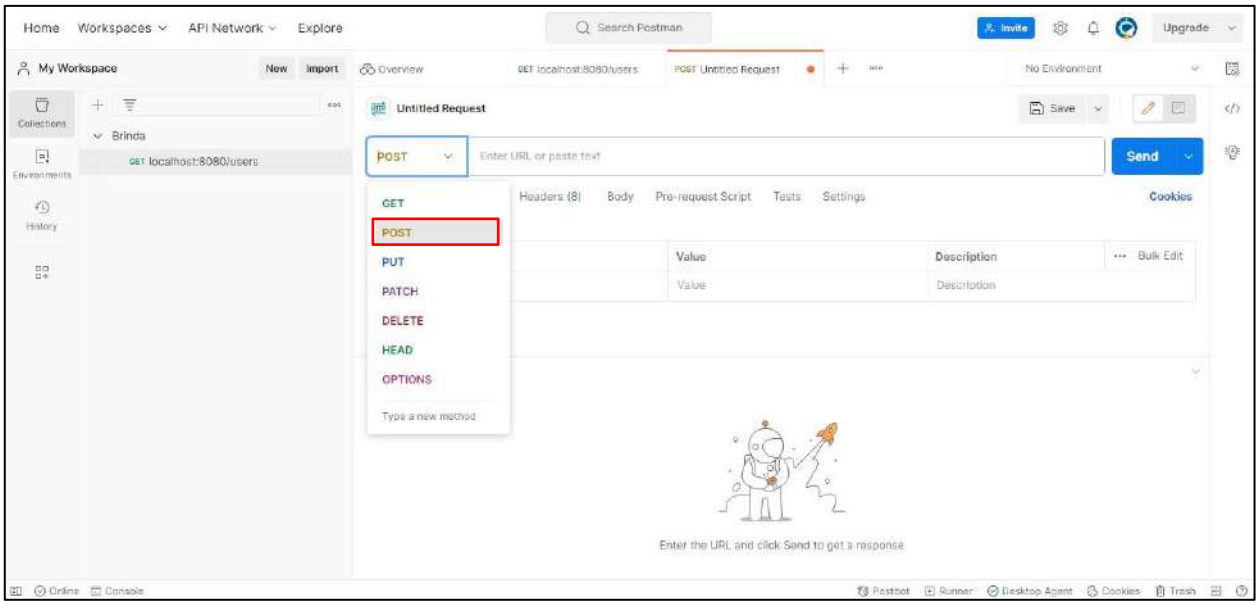
Step 32: Then click on **'Save'**.



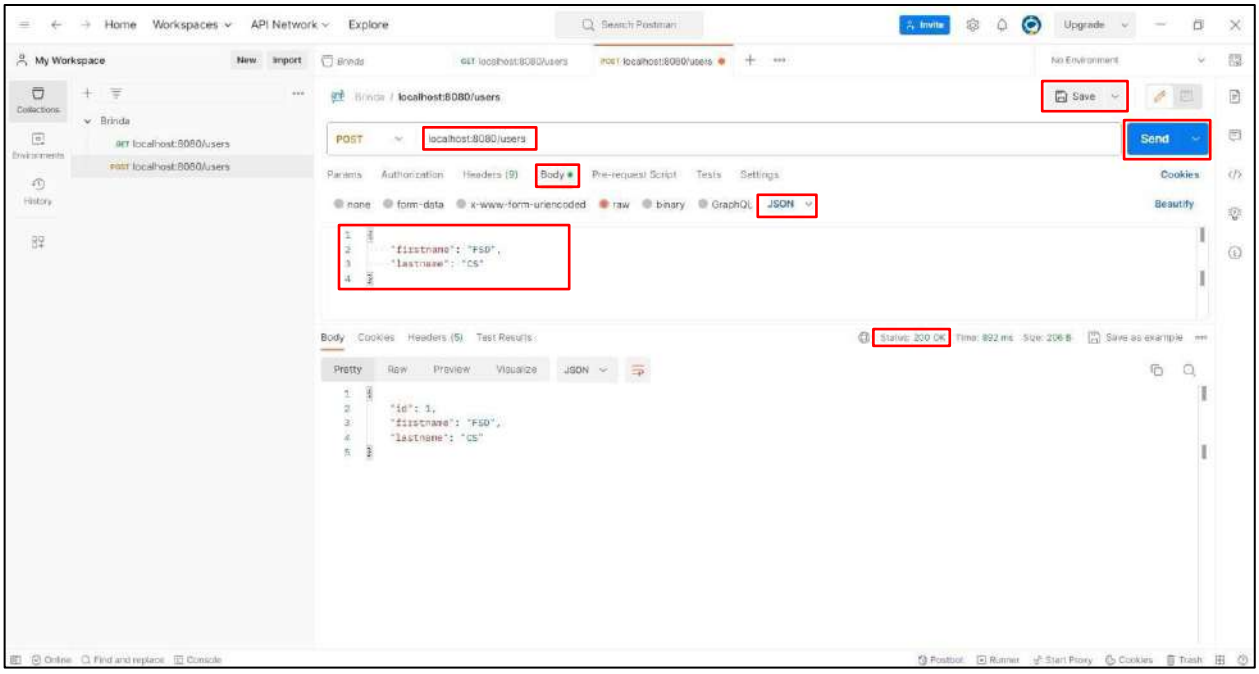
Step 33: Select **'GET'** method & enter the command as **'localhost:8080/users'** □ Save it & Send.



Step 34: Select **'POST'** method.

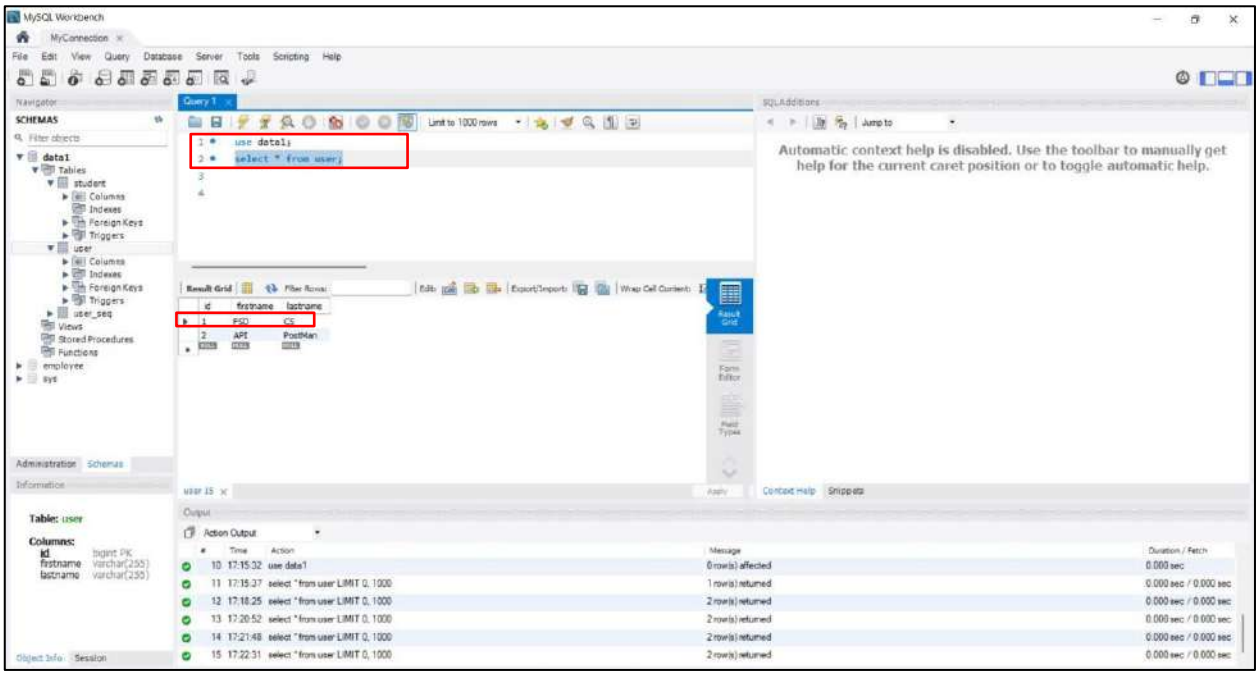


Step 35: Then enter the command as **'localhost:8080/users'** □ select **Body** □ Text type - **JSON** □ enter the below code as shown below □ Save it & Send.

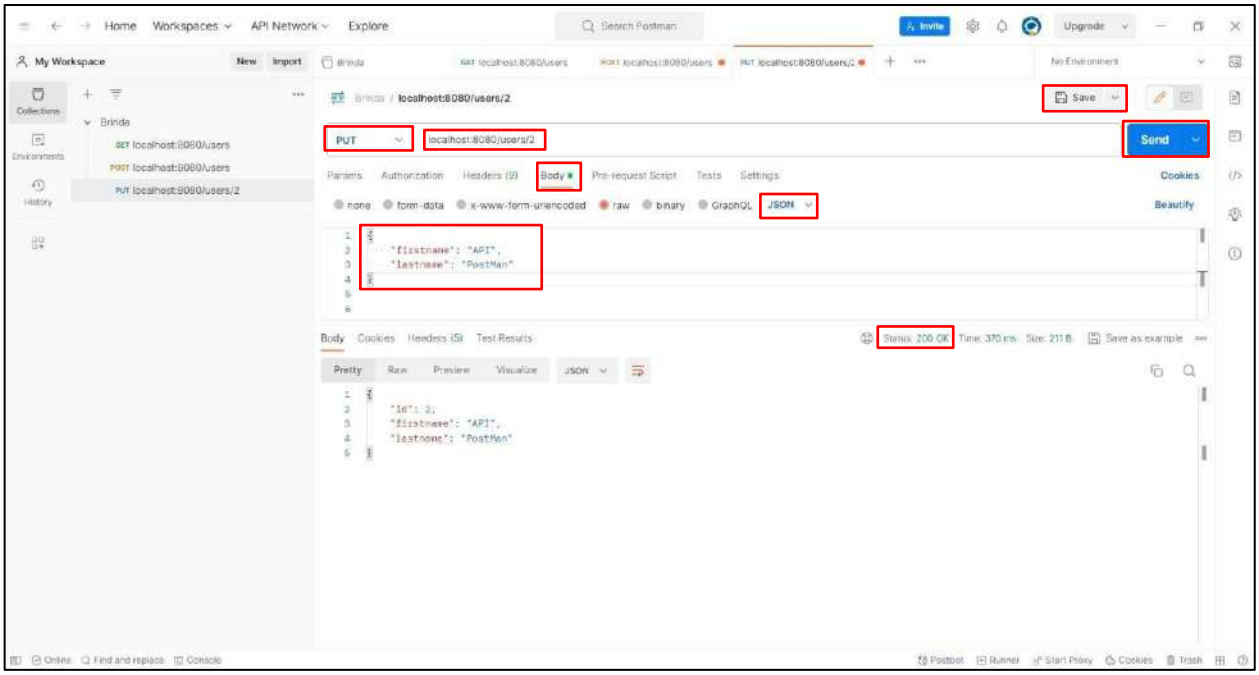


Step 36: Now go to MySQL Workbench & enter the below command as shown below. To run the code, select each code line separately & press **ctrl+shift+enter** key to view the output.

Note: Here the POST method is created successfully.

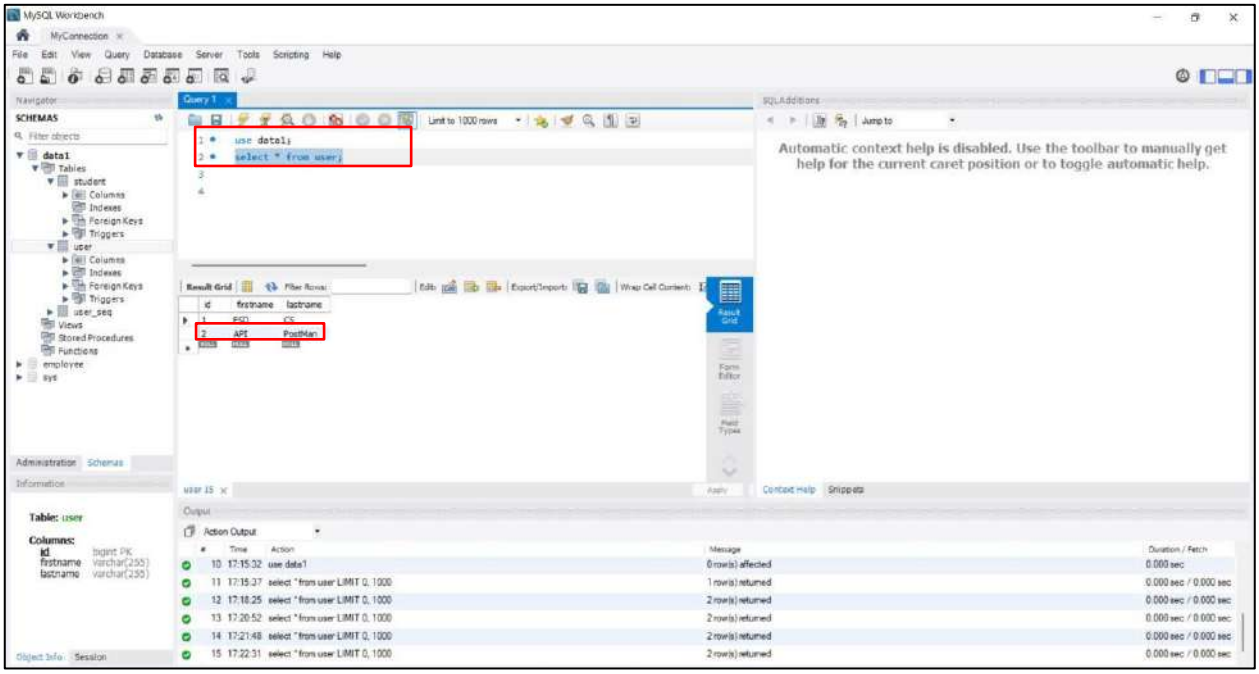


Step 37: Select **'PUT'** method & then enter the command as **'localhost:8080/users/2'** □ select **Body** □ Text type - **JSON** □ enter the below code as shown below □ Save it & Send.

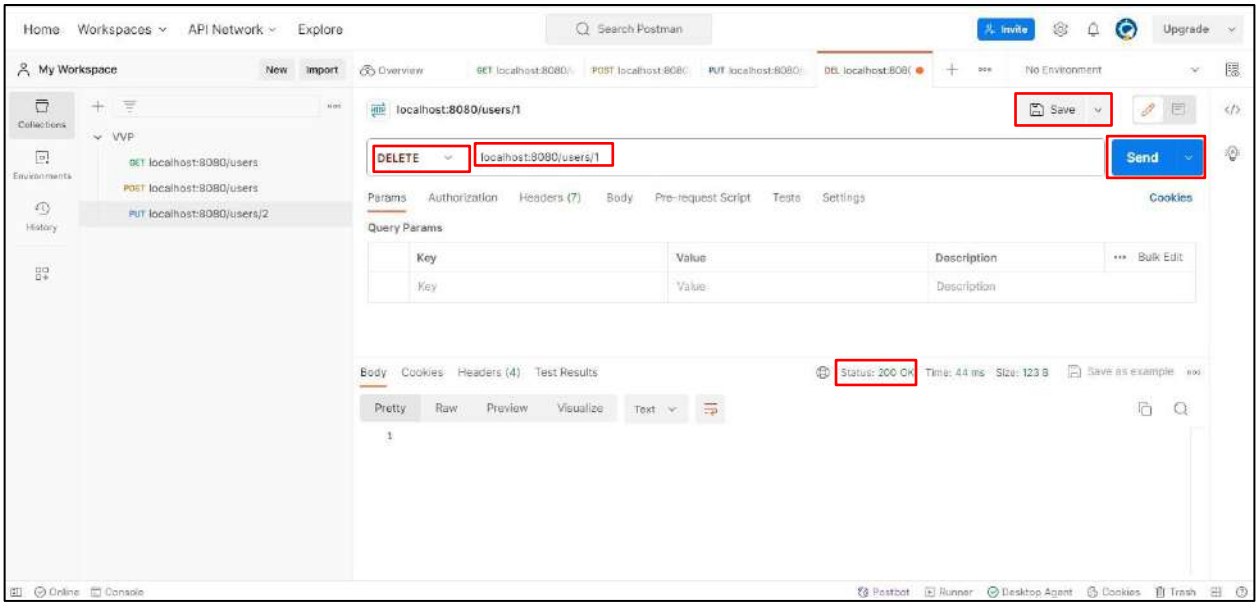


Step 38: Now go to MySQL Workbench & enter the below command as shown below. To run the code, select each code line separately & press **ctrl+shift+enter** key to view the output.

Note: Here the PUT method is updated successfully.

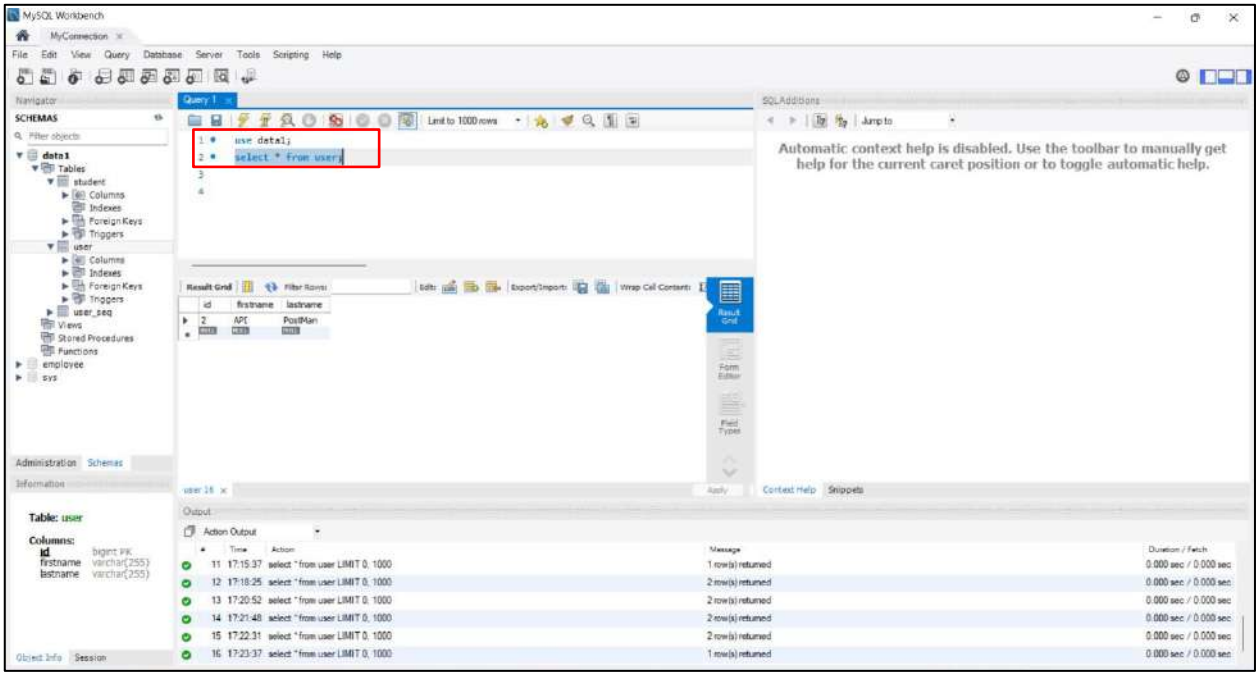


Step 39: Select **'DELETE'** method & then enter the command with the id no. specified as **'localhost:8080/users/1'** → select **Body** → Text type - **JSON** → enter the below code as shown below → Save it & Send.



Step 40: Now go to MySQL Workbench & enter the below command as shown below. To run the code, select each code line separately & press **ctrl+shift+enter** key to view the output.

Note: Here the DELETE method is deleted successfully.



3. Create user registration form.

Step 1: Now go to 'Eclipse IDE for enterprise edition' → File → New → Spring starter project → Add all the dependencies → Create three packages named as **(com.example.user.controller, com.example.user.entity, com.example.user.repository)** & create two classes named as **UserController.java, User.java** files & then create interface named as **UserRepository.java** file as shown below.

Step 2: Now add code to **UserController.java** file as shown below.

UserController.java | User.java | UserRepository.java | application.properties | UserApplication.java

```
package com.example.user.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.user.entity.User;
import com.example.user.repository.UserRepository;

@RestController
@RequestMapping("/users")
public class UserController {
@Autowired
private UserRepository userRepository;

@GetMapping
public List<User> getAllUser()
{
    return this.userRepository.findAll();
}

@GetMapping("/{id}")
public User getUserById(@PathVariable(value="id") long userId) {
    return this.userRepository.findById(userId).orElseThrow();
}

@PostMapping
public User createUser(@RequestBody User user)
{
    return this.userRepository.save(user);
}
```

```
@PutMapping("/{id}")
public User updateUser(@RequestBody User user,@PathVariable("id") long userId)
{
    User ex=this.userRepository.findById(userId).orElseThrow();
    ex.setFirstname(user.getFirstname());
    ex.setLastname(user.getLastname());
    ex.setRegisternumber(user.getRegisternumber());
    ex.setDOB(user.getDOB());
    ex.setAddress(user.getAddress());
    ex.setDepartment(user.getDepartment());
    return this.userRepository.save(ex);
}
@DeleteMapping("/{id}")
public ResponseEntity<User> deleteUser(@PathVariable("id") long userId)
{
    User ex=this.userRepository.findById(userId).orElseThrow();
    this.userRepository.delete(ex);
    return ResponseEntity.ok().build();
}
}
```

Step 3: Now add code to **User.java** file as shown below.

```java
package com.example.user.entity;

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

@Entity
@Table(name="user")
public class User {
public User() {
    }
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;
    private String firstname;
    private String lastname;
    private String Registernumber;
    private String DOB;
    private String Address;
    private String Department;
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getFirstname() {
        return firstname;
    }
    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
    public String getLastname() {
        return lastname;
    }
    public void setLastname(String lastname) {
        this.lastname = lastname;
    }
```

```java
    public String getRegisternumber() {
        return Registernumber;
    }
    public void setRegisternumber(String registernumber) {
        Registernumber = registernumber;
    }
    public String getDOB() {
        return DOB;
    }
    public void setDOB(String dOB) {
        DOB = dOB;
    }
    public String getAddress() {
        return Address;
    }
    public void setAddress(String address) {
        Address = address;
    }
    public String getDepartment() {
        return Department;
    }
    public void setDepartment(String department) {
        Department = department;
    }
    public User(Long id, String firstname, String lastname, String registernumber, String dOB, String address,
            String department) {
        super();
        this.id = id;
        this.firstname = firstname;
        this.lastname = lastname;
        Registernumber = registernumber;
        DOB = dOB;
        Address = address;
        Department = department;
    }
}
```

Step 4: Now add code to **UserRepository.java** file as shown below.

```java
package com.example.user.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.user.entity.User;

@Repository
public interface UserRepository extends JpaRepository<User,Long>
{

}
```

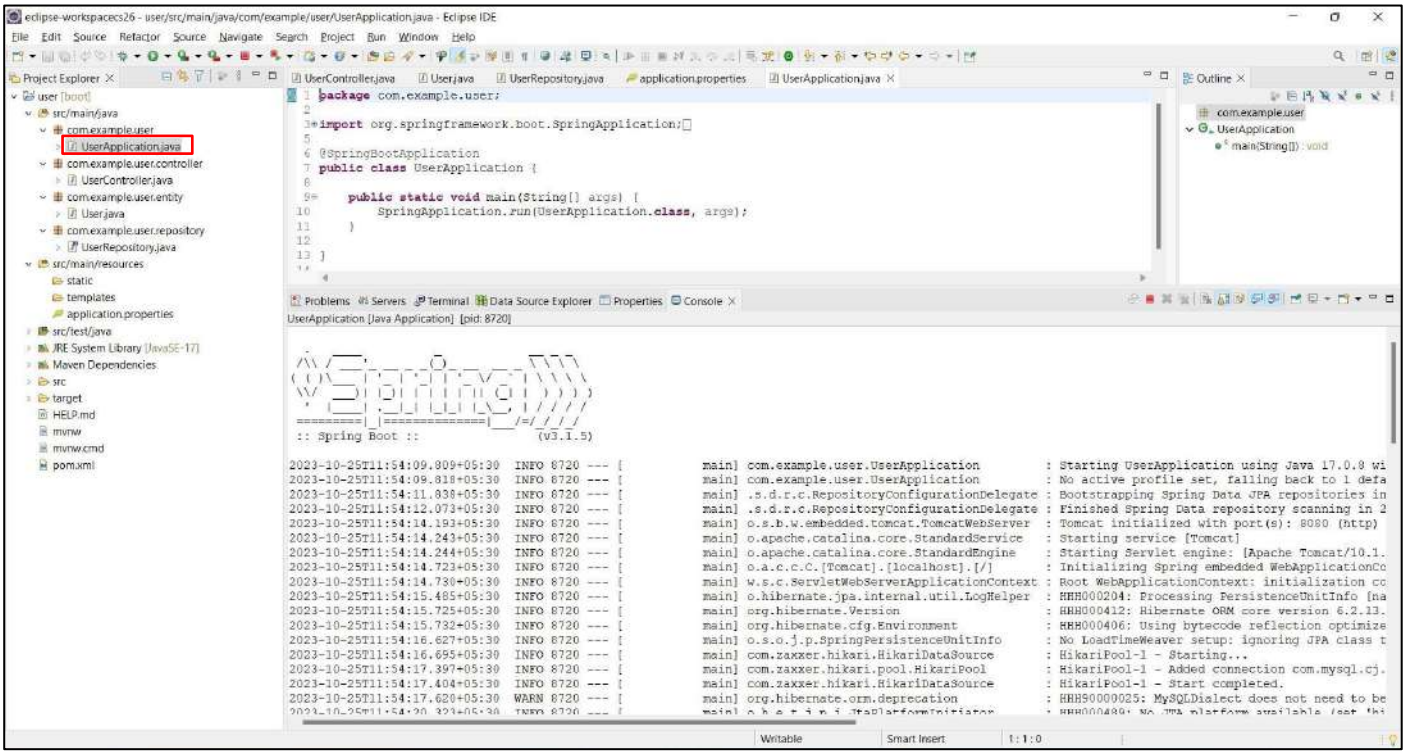
Step 5: Now go to **'src/main/resources'** folder select **'application.properties'** file here.

Step 6: Now add code to **application.properties** file as shown below.

```
spring.datasource.url=jdbc:mysql://localhost:3306/register
spring.datasource.username=root
spring.datasource.password=Vvp_dit@123
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update
server.port=8080
```

Step 7: Go to **src/main/java** right click on the **UserApplication.java** **Run as** **1java application.**

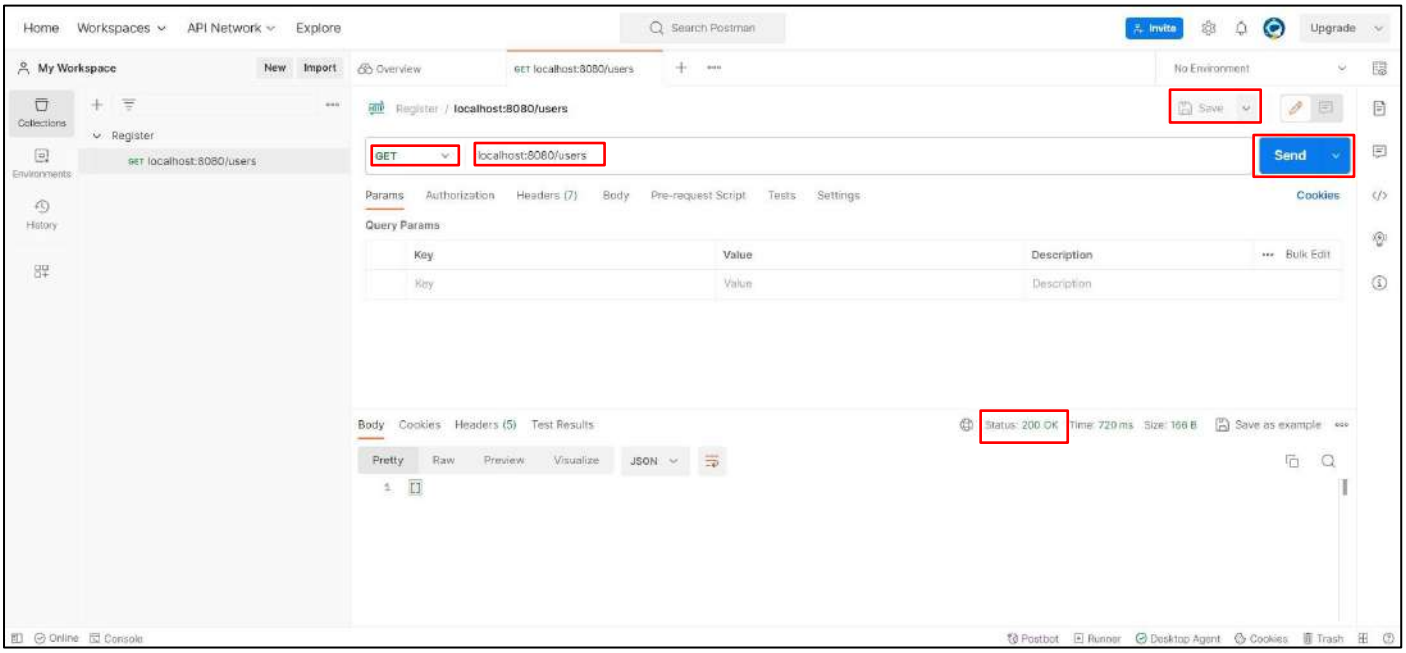
Now the server is running successfully.



Step 8: Create a new database by entering the command as **'create database register;'** as shown below.

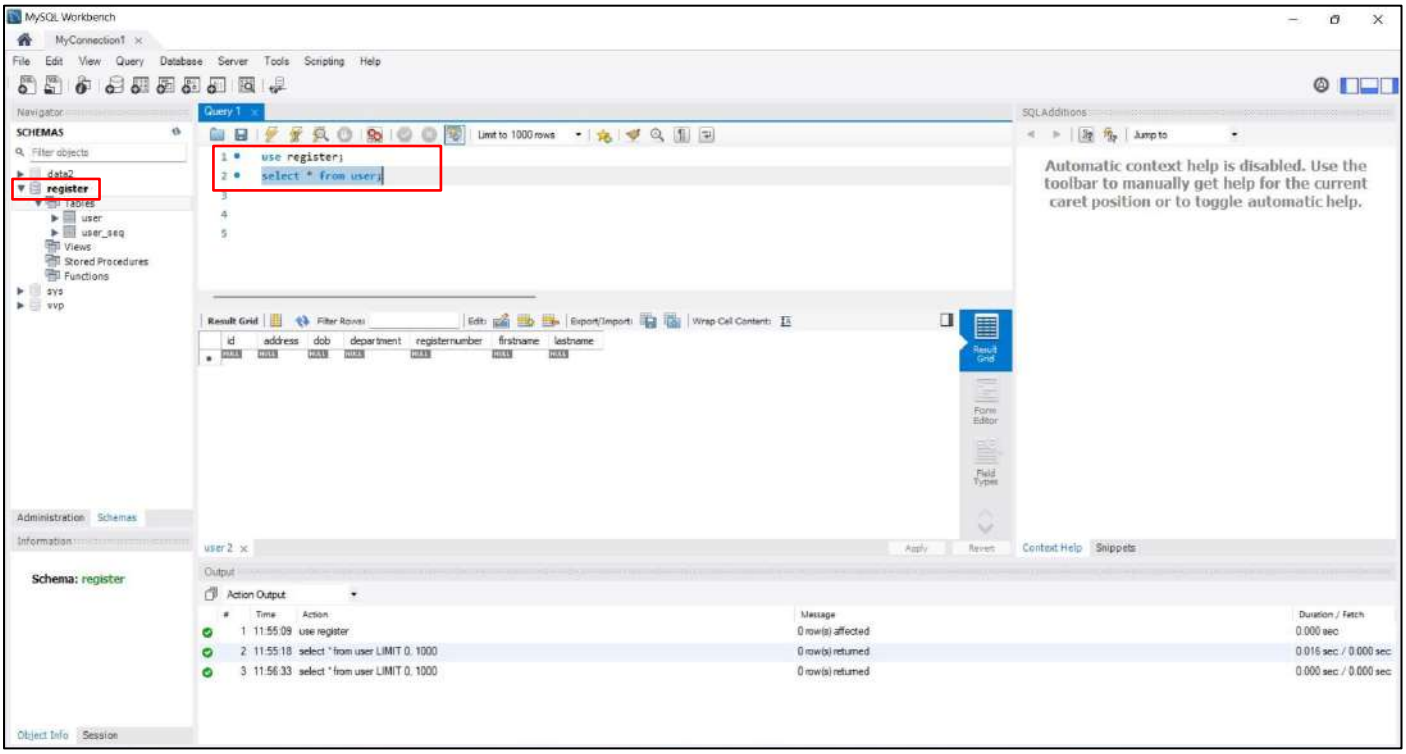
Step 9: Login to the Postman & select **'GET'** method □ enter the command as **'localhost:8080/users'** as shown below □ Save it □ Send.

Note: Then check the response status here. If the status is 200 OK means the response has been accepted by the POSTMAN.

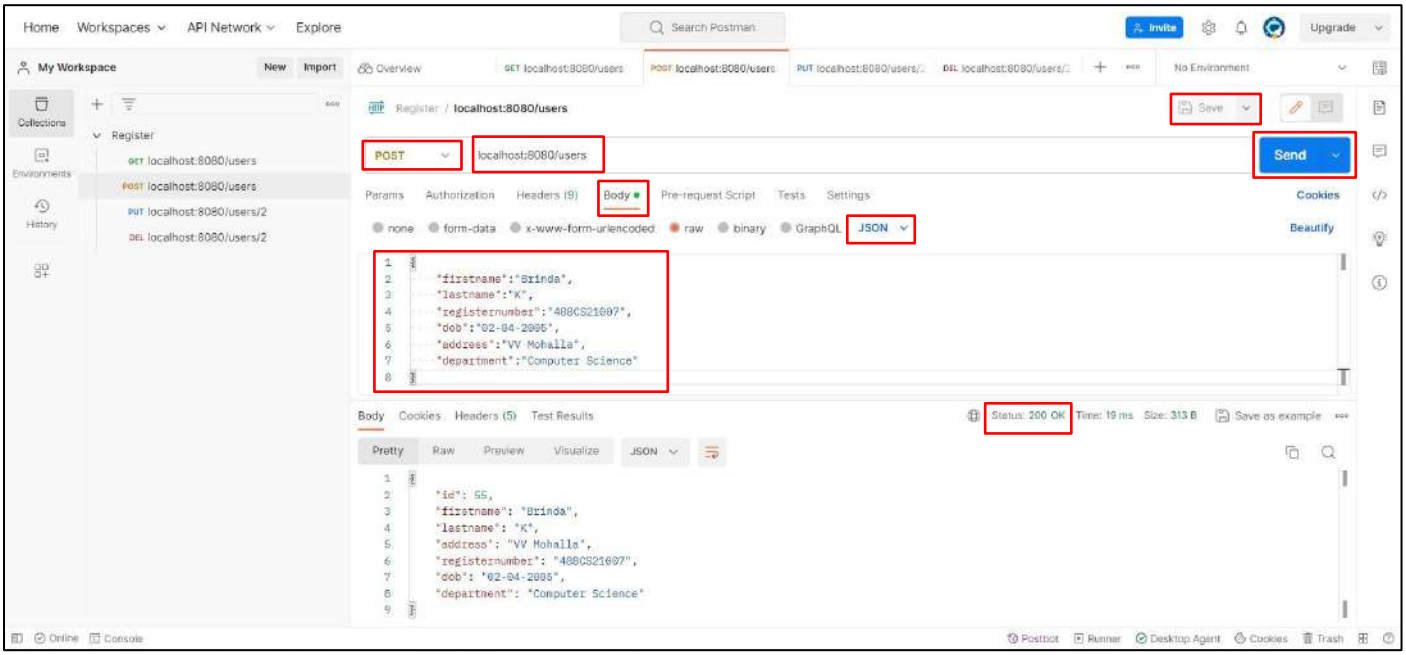


Step 10: Now enter the code as shown below.

Note: To run the code, select each code line separately & press **ctrl+shift+enter** key to view the output.

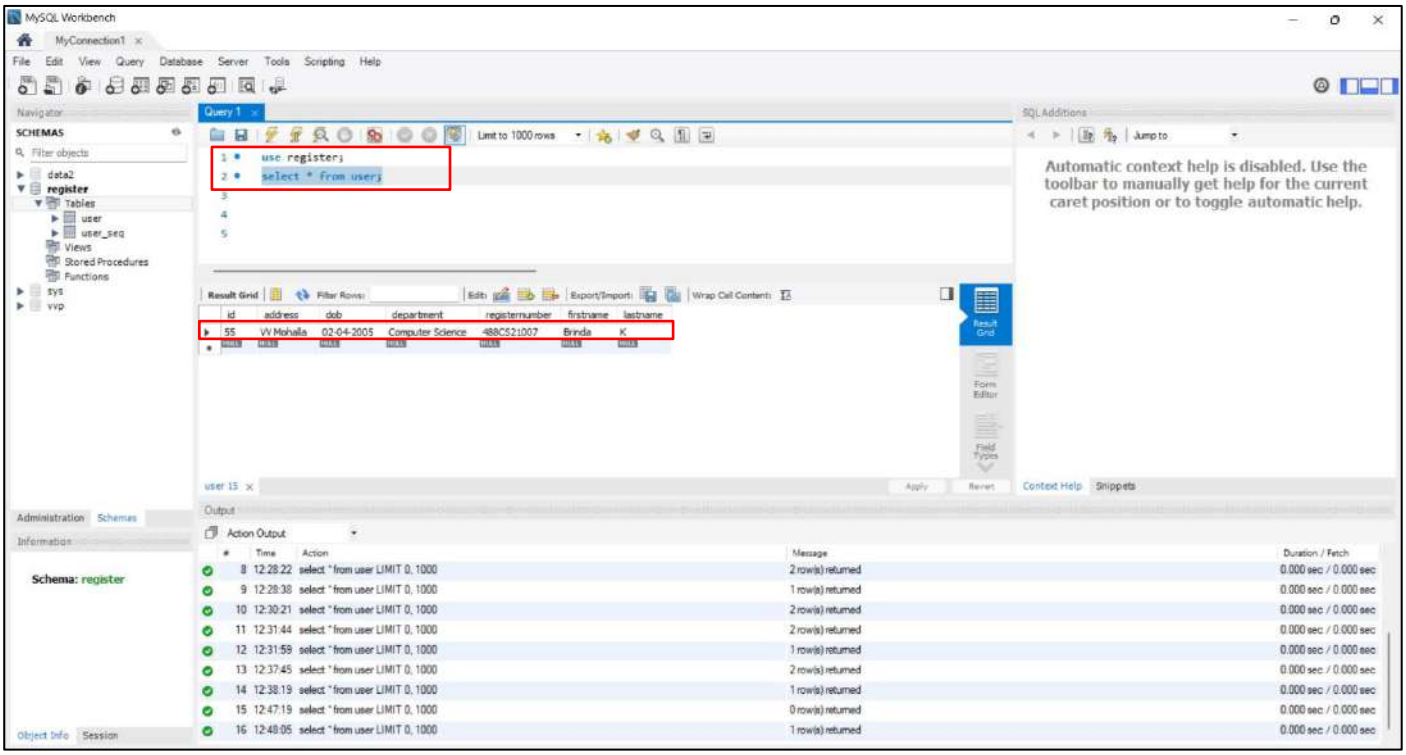


Step 11: Select **'POST'** method →enter the command as **'localhost:8080/users'** as shown below → select **Body** → text type – **JSON**→ enter the below code → Save it → Send.

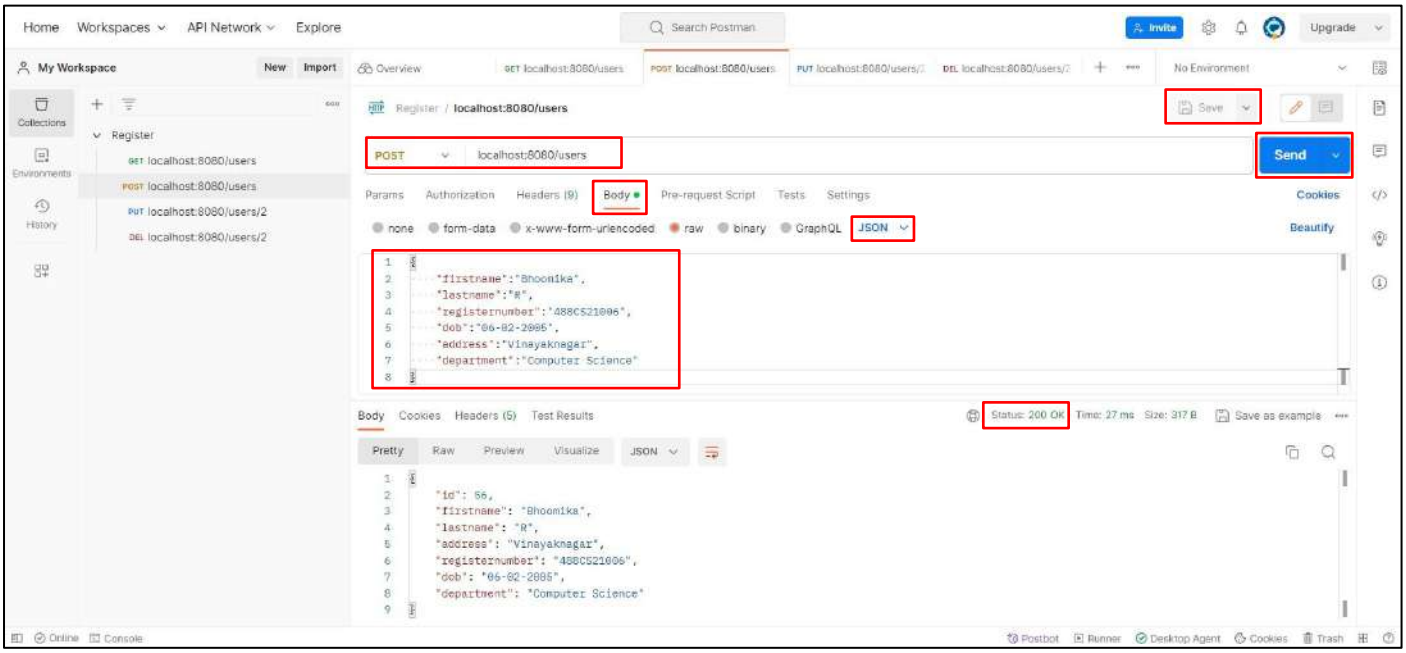


Step 12: Now go to MySQL Workbench & enter the below command as shown below. To run the code, select each code line separately & press **ctrl+shift+enter** key to view the output.

Note: Here the POST method is created successfully.



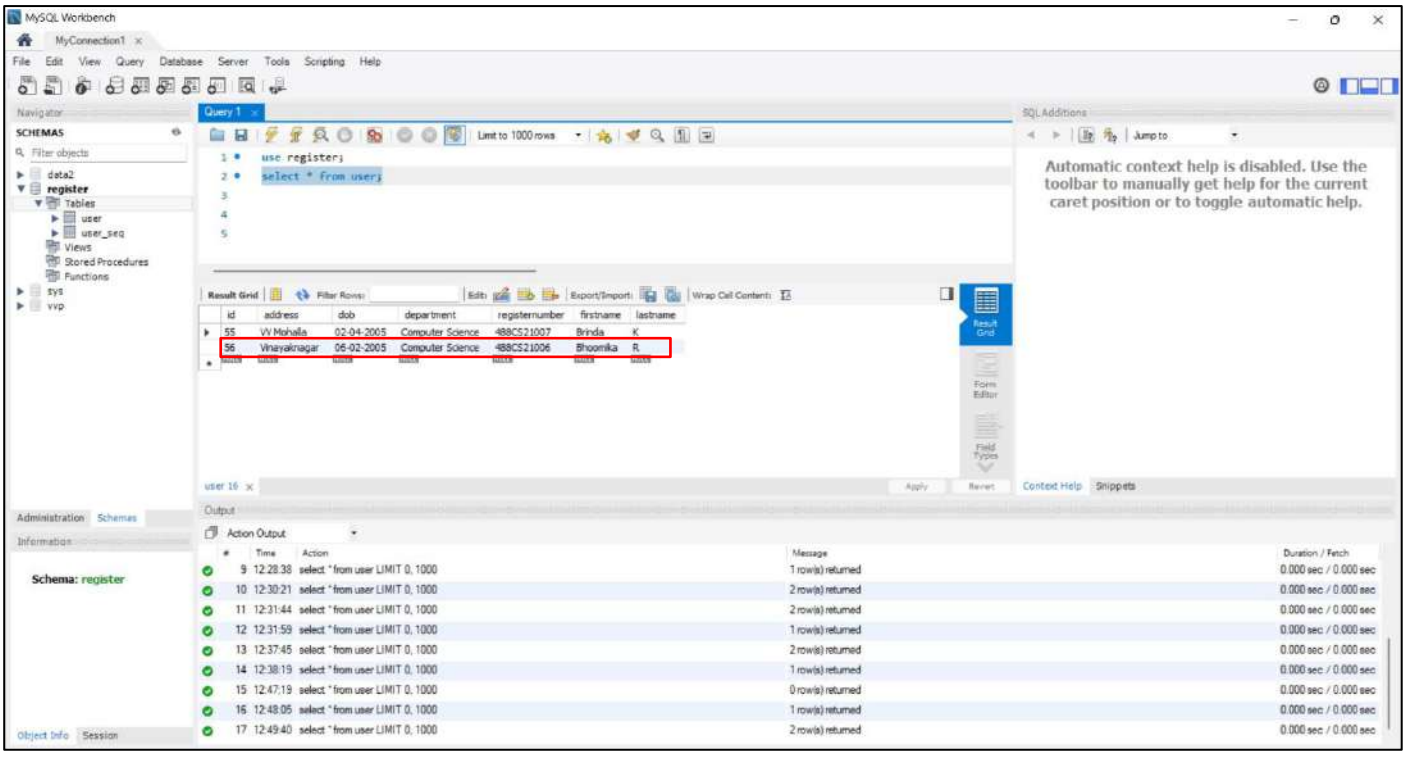
Step 13: Select **'POST'** method → enter the command as **'localhost:8080/users'** as shown below → select **Body** → text type – **JSON**→ enter the below code → Save it → Send.



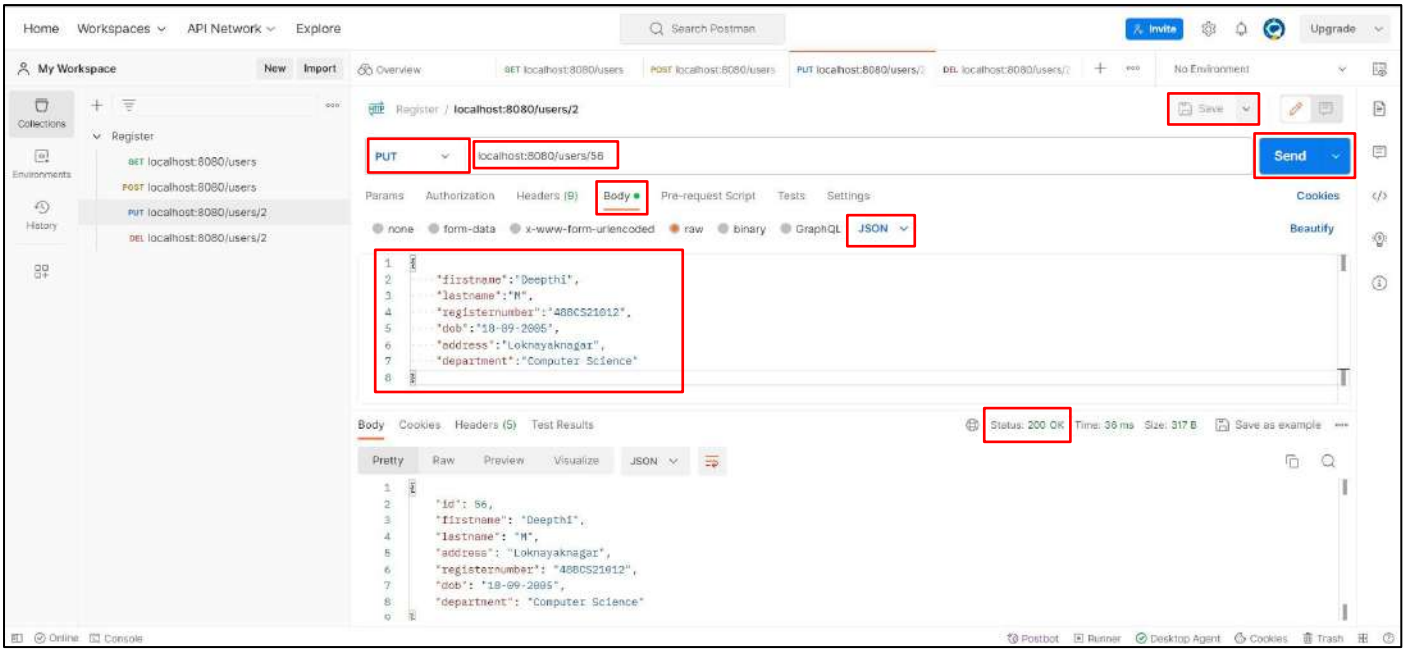
Step 14: Now go to MySQL Workbench & enter the below command as shown below. To run the code, select each code line separately & press **ctrl+shift+enter** key to view the output.

Note: Here the POST method is created successfully.

Note: Here I've created two POST methods with different ids to view the PUT method clearly.

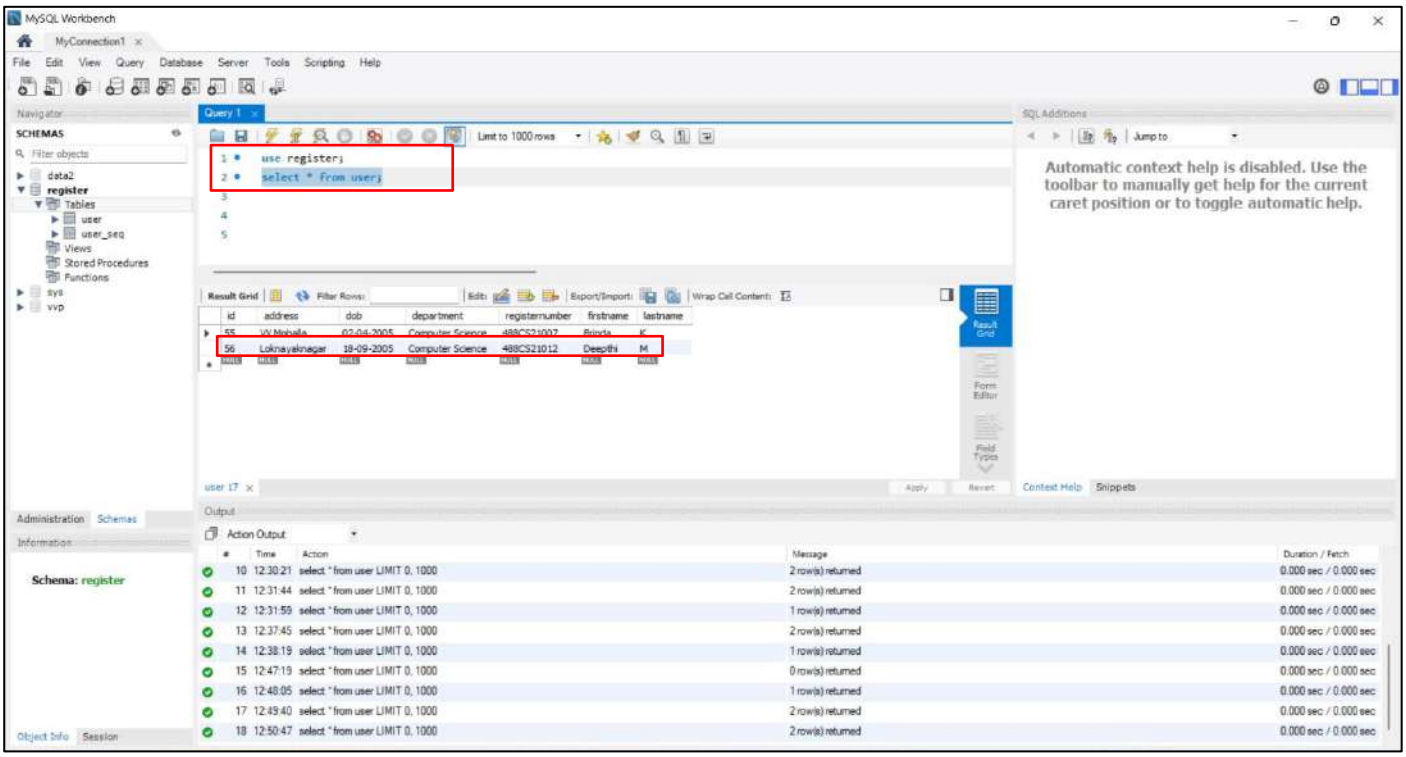


Step 15: Select **'PUT'** method → enter the command with id specified as **'localhost:8080/users/56'** as shown below → select **Body** → text type – **JSON**→ enter the below code→ Save it → Send.

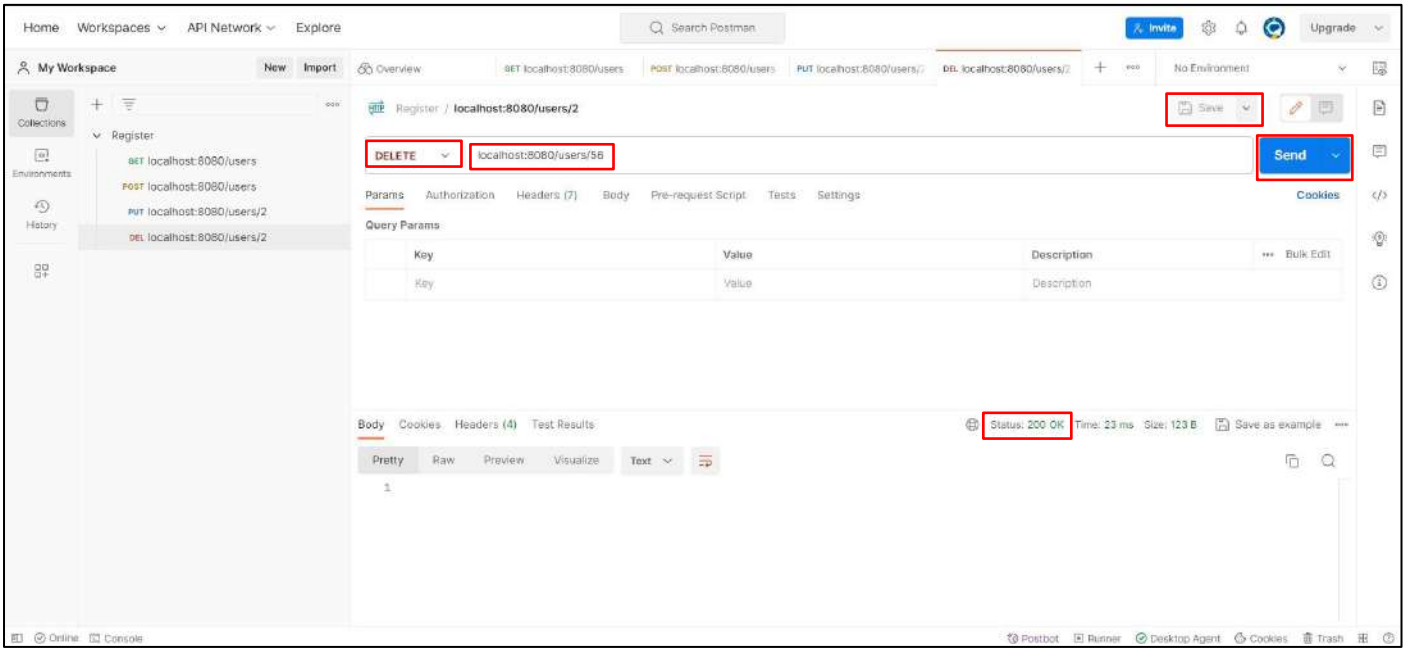


Step 16: Now go to MySQL Workbench & enter the below command as shown below. To run the code, select each code line separately & press **ctrl+shift+enter** key to view the output.

Note: Here the PUT method is updated successfully.

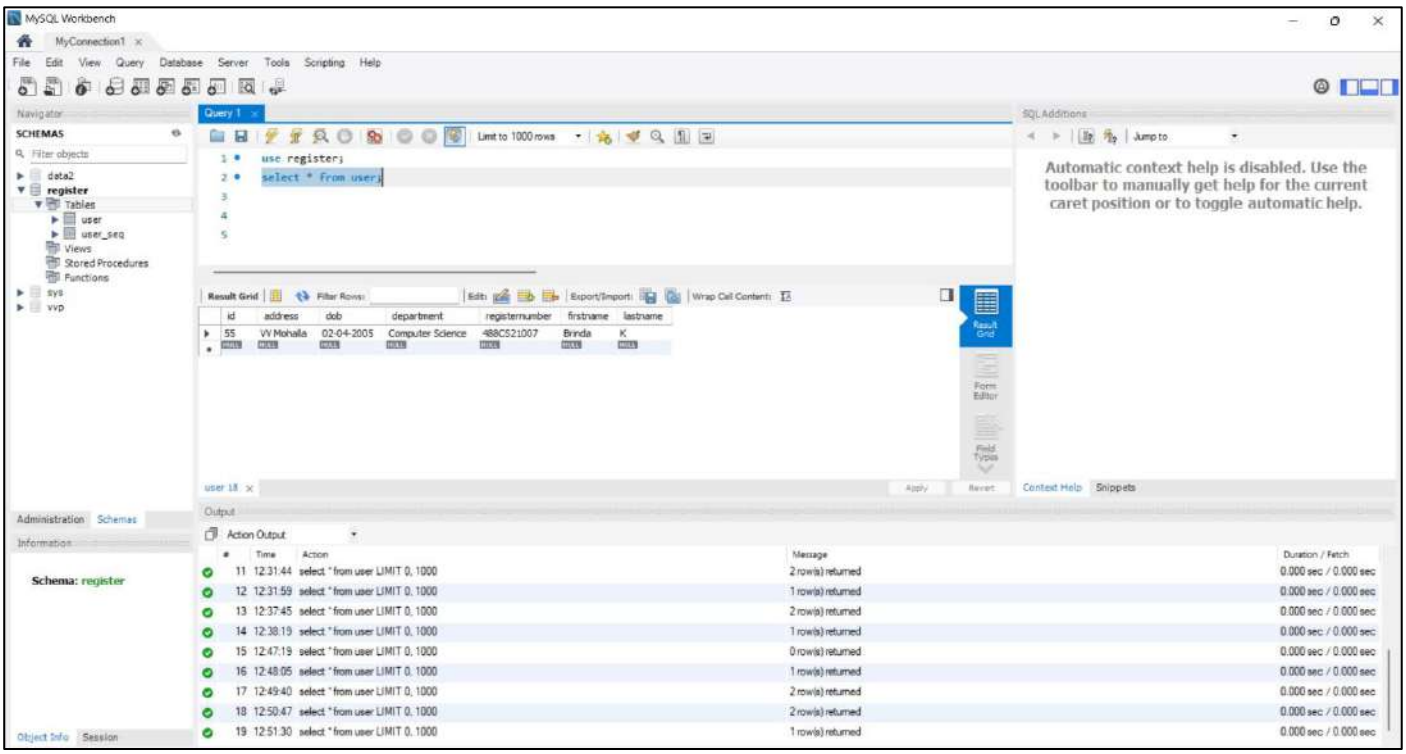


Step 17: Select **'DELETE'** method → enter the command with id specified as **'localhost:8080/users/56'** as shown below → select **Body** → text type – **JSON** → enter the below code → Save it → Send.



Step 18: Now go to MySQL Workbench & enter the below command as shown below. To run the code, select each code line separately & press **ctrl+shift+enter** key to view the output.

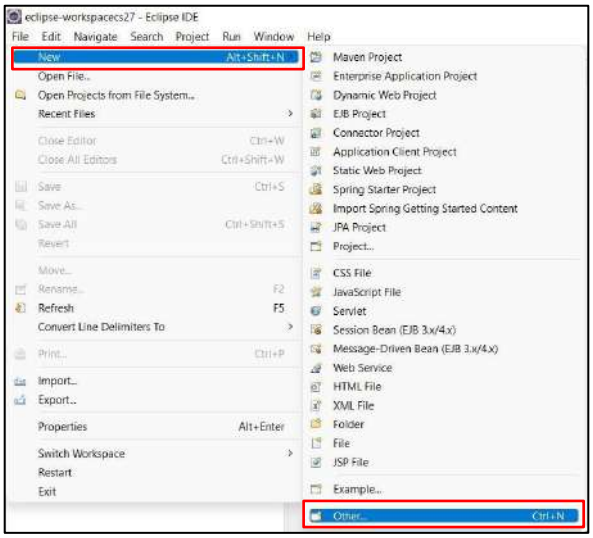
Note: Here the DELETE method is deleted successfully.



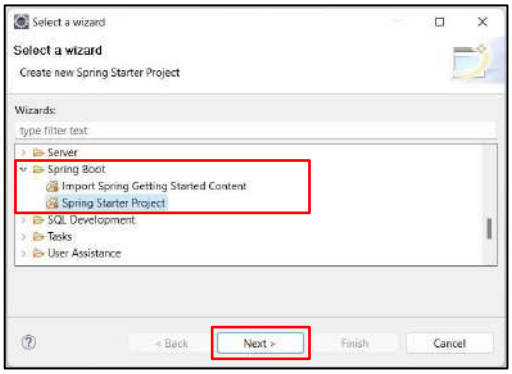
WEEK – 9

1. Build user authentication flow and authorization using SpringSecurity.

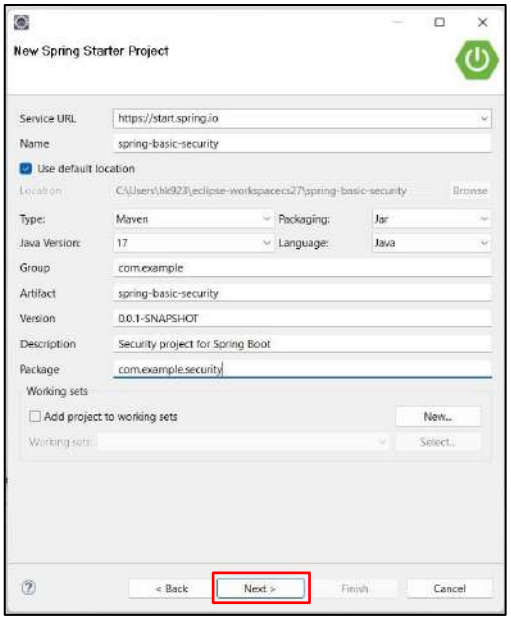
Step 1: Open ‘Eclipse IDE for Enterprise edition’ & go to → Help →Eclipse Marketplace as shown below. Now search for ‘Spring tool suite’ & click on ‘Install’ to install the latest version. After the installation is complete, go to → File → New → Other option as shown below.



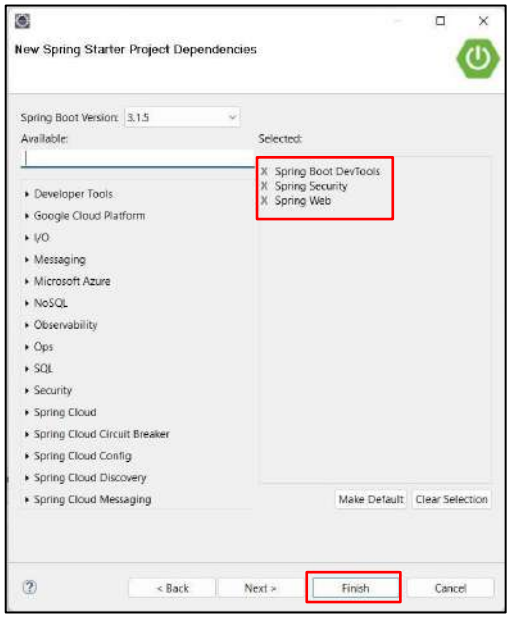
Step 2: Now select ‘Spring Starter Project’ & click on ‘Next’ option.



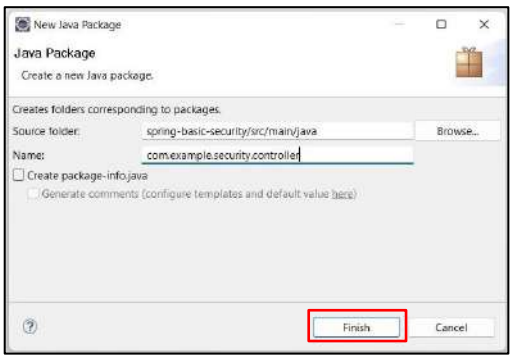
Step 3: Give the Name as ‘User’ → select ‘Maven’ → Description as User Registration Spring Boot Application → Package as ‘spring-basic-security’ & then click on ‘Next’.



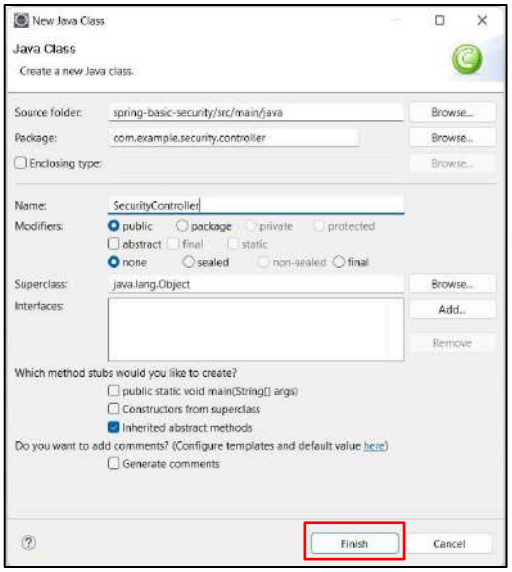
Step 4: Now add all the required dependencies as shown below & click on ‘Finish’.



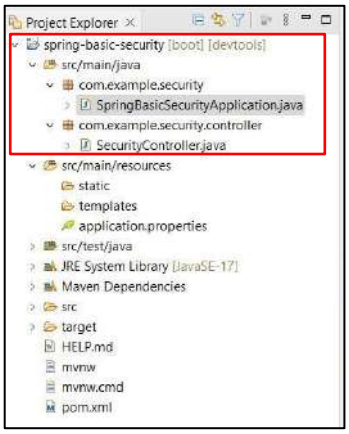
Step 5: Create a new package named ‘com.example.security.controller’ & click on ‘Finish’.



Step 6: Now create a new class named ‘SecurityController’ & click on ‘Finish’.



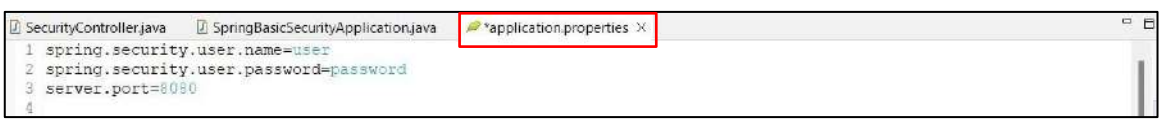
Step 7: All the created packages and classes are displayed here.



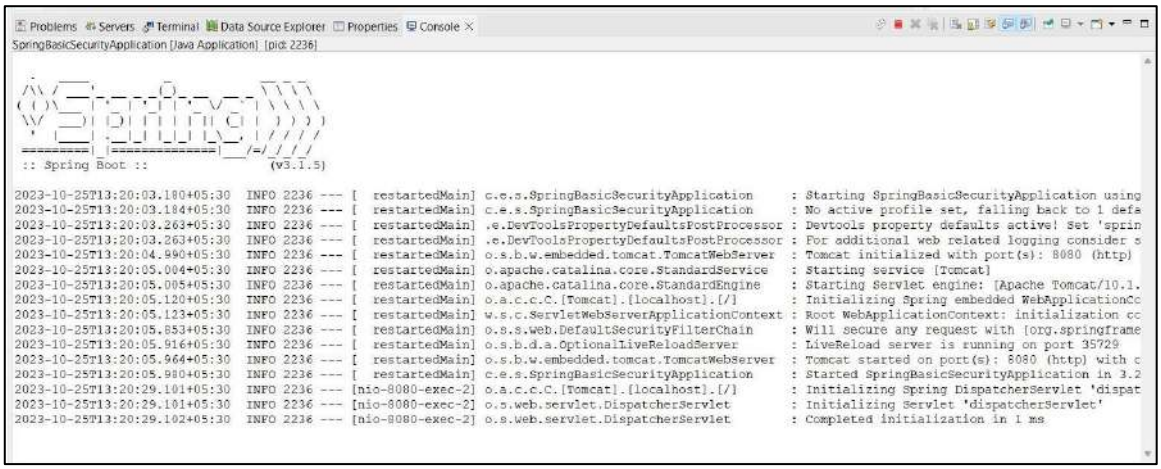
Step 8: Now add code to SecurityController.java file as shown below.

Step 9: Now add code to SpringBasicSecurityApplication.java file as shown below.

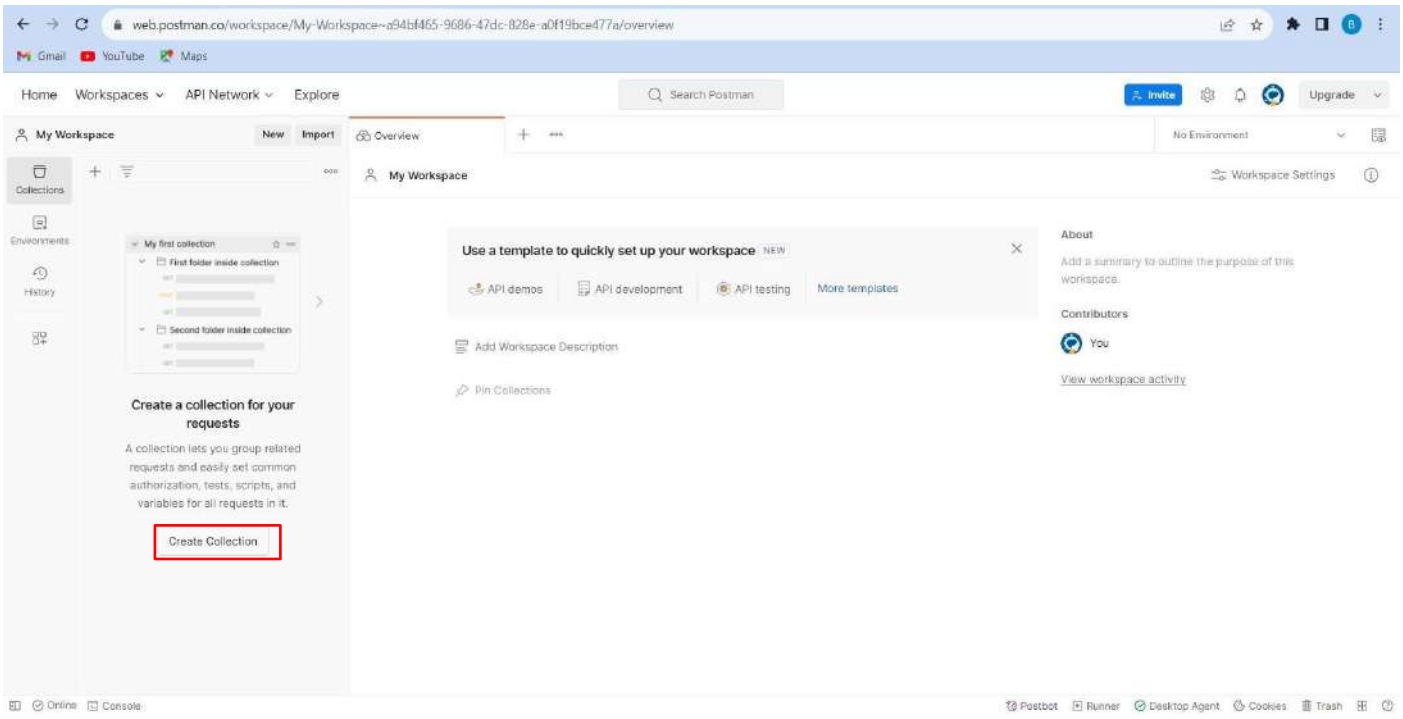
Step 10: Now add code to ‘application.properties’ file as shown below.



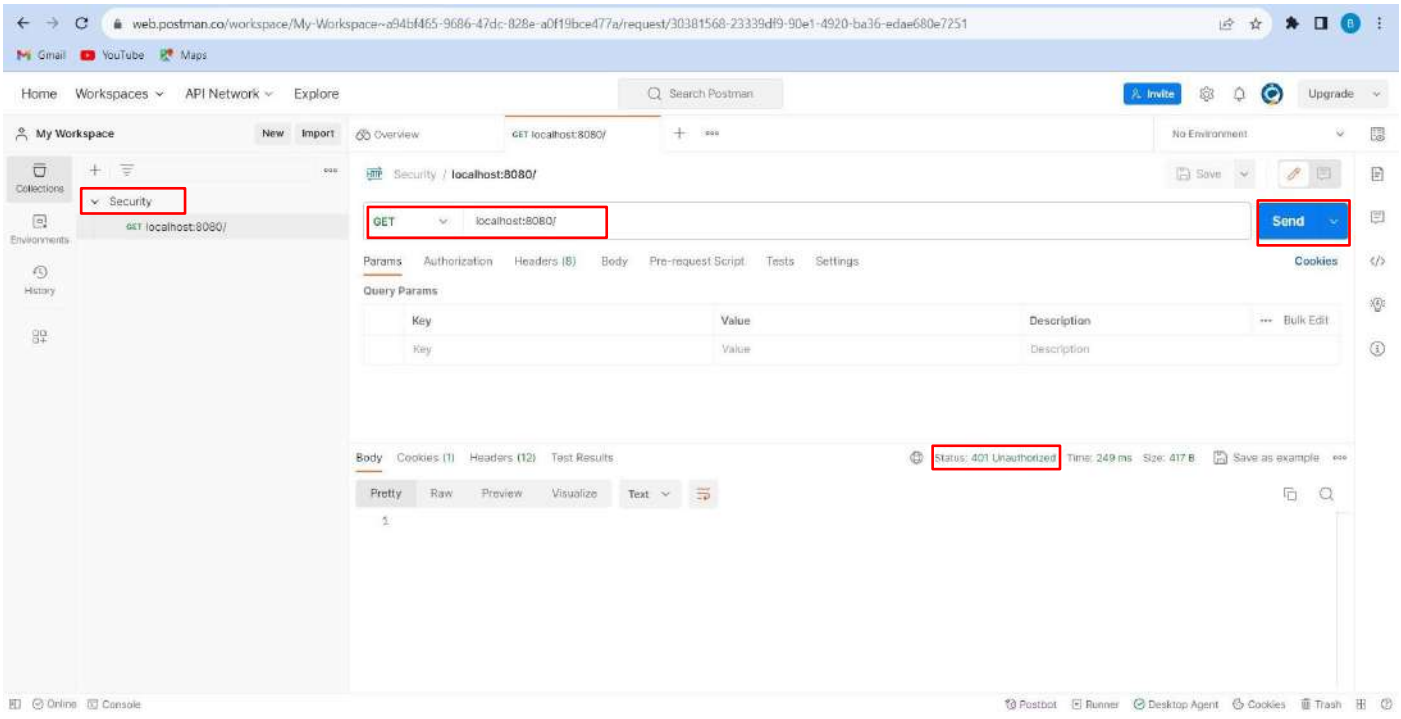
Step 11: Start the server as shown below.



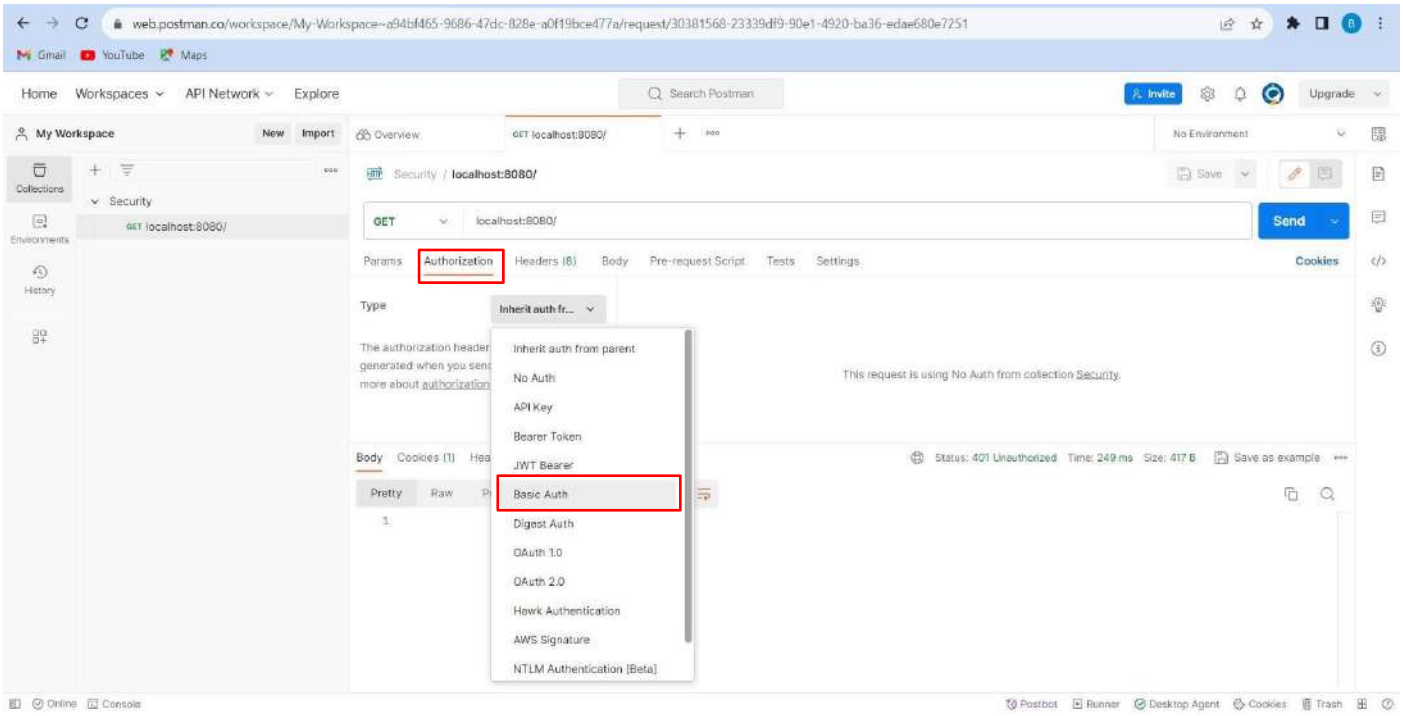
Step 12: Now login to the Postman & create a new collection as shown below.



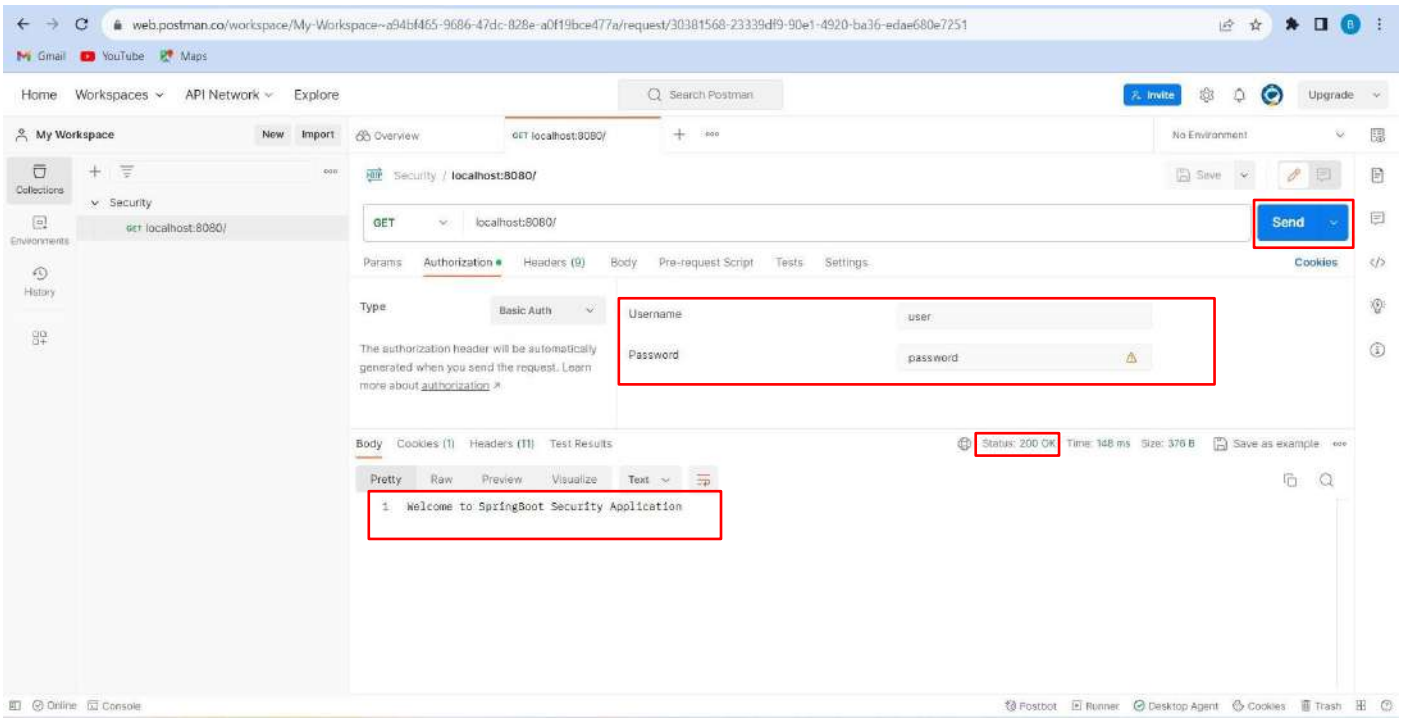
Step 13: Select GET method & enter ‘localhost:8080/’. Now Save it, then click on ‘Send’. The status displayed here is unauthorized so make sure to add the authorization to your code.



Step 14: Now go to ‘Authorization’ tab select the type as ‘Basic Auth’.



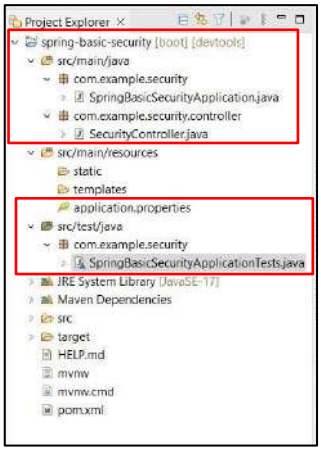
Step 15: Now enter Username & Password as mentioned below. Then Save it & click on Send to view the output.



- **<u>Implementing Junit for the above Security API</u>**

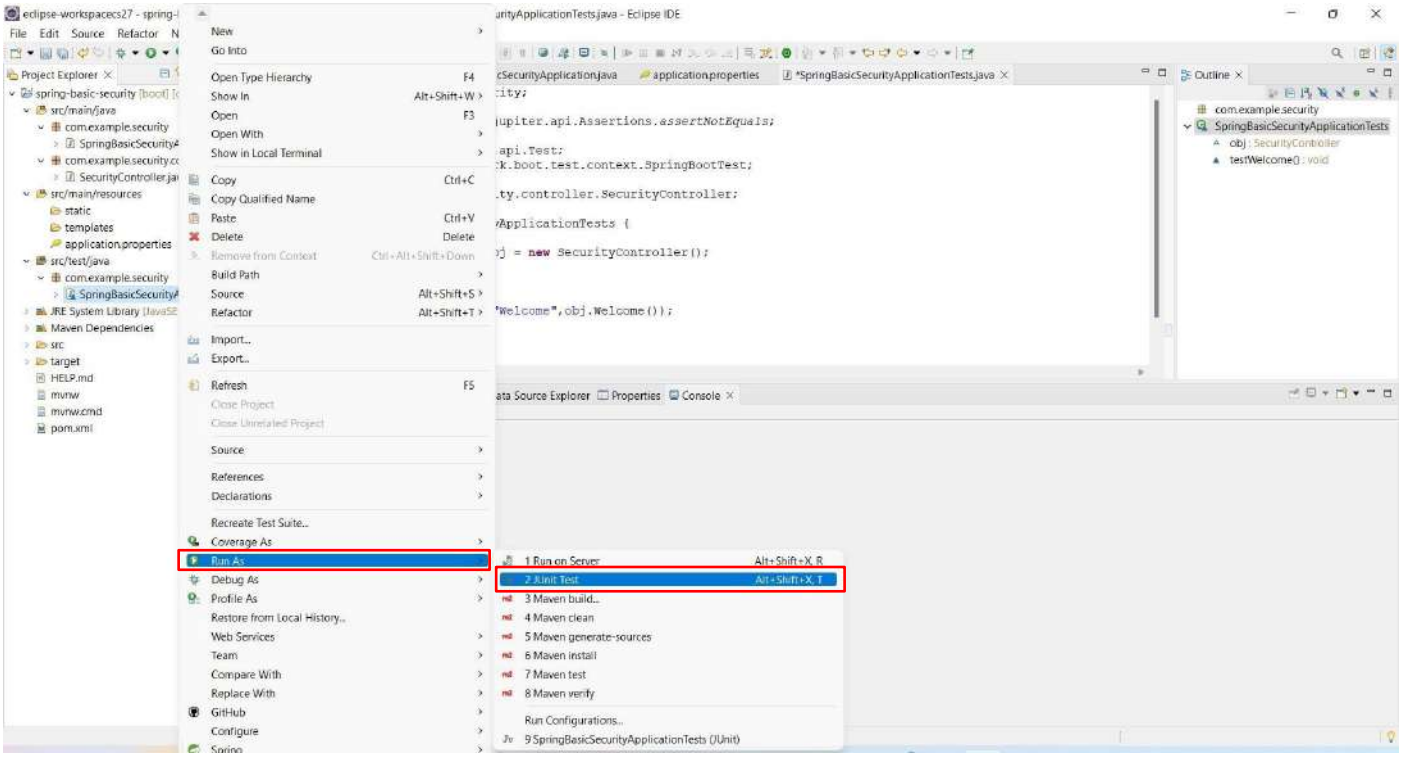
Step 16: Follow the same procedures as above from ‘step 1’.

Note: Create the packages & classes as shown below, then add code as mentioned from ‘step 8’.



Step 17: Now add code to ‘SpringBasicSecurityApplicationTests.java’ as shown below.

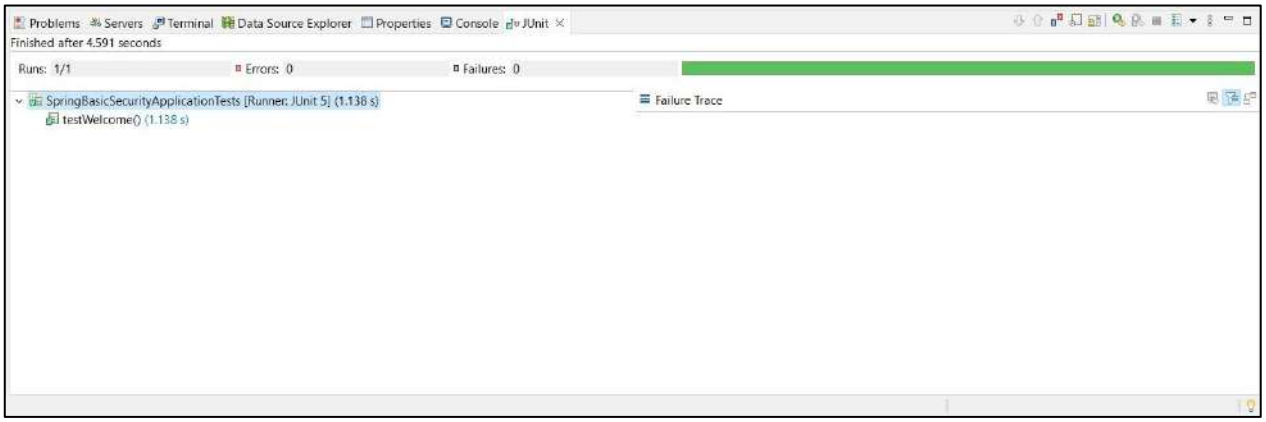
Step 18: Right click on the ‘SpringBasicSecurityApplicationTests.java’ file Run As 2 Junit Test.



Step 19: View the output as shown below.

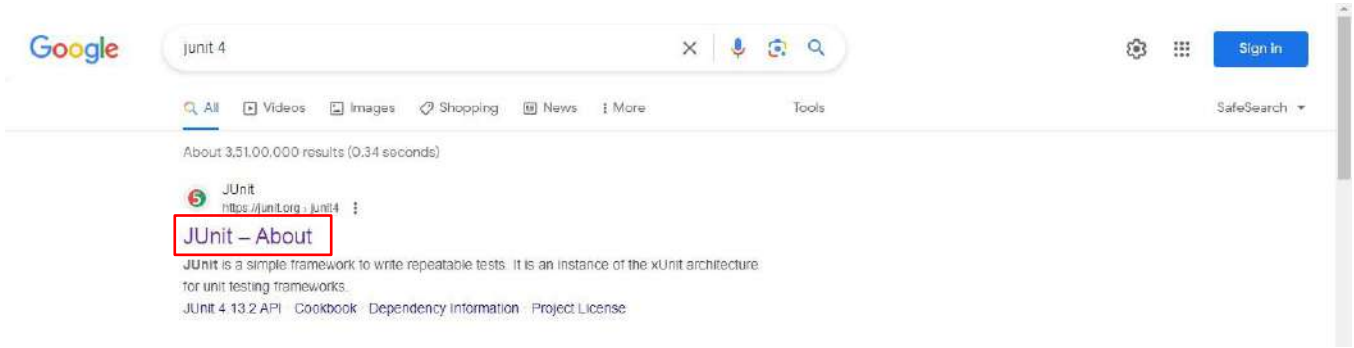
Step 20: The Junit has executed successfully.

Note: The green colour indicates the given test condition is true & passed. As well as red colour indicates the given test condition is false & failed.

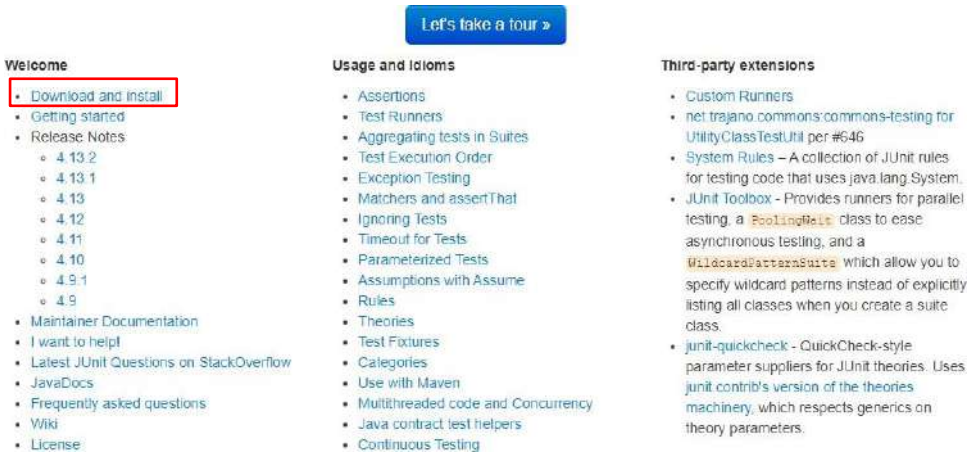


2. Writing Junit test cases for CRUD operations

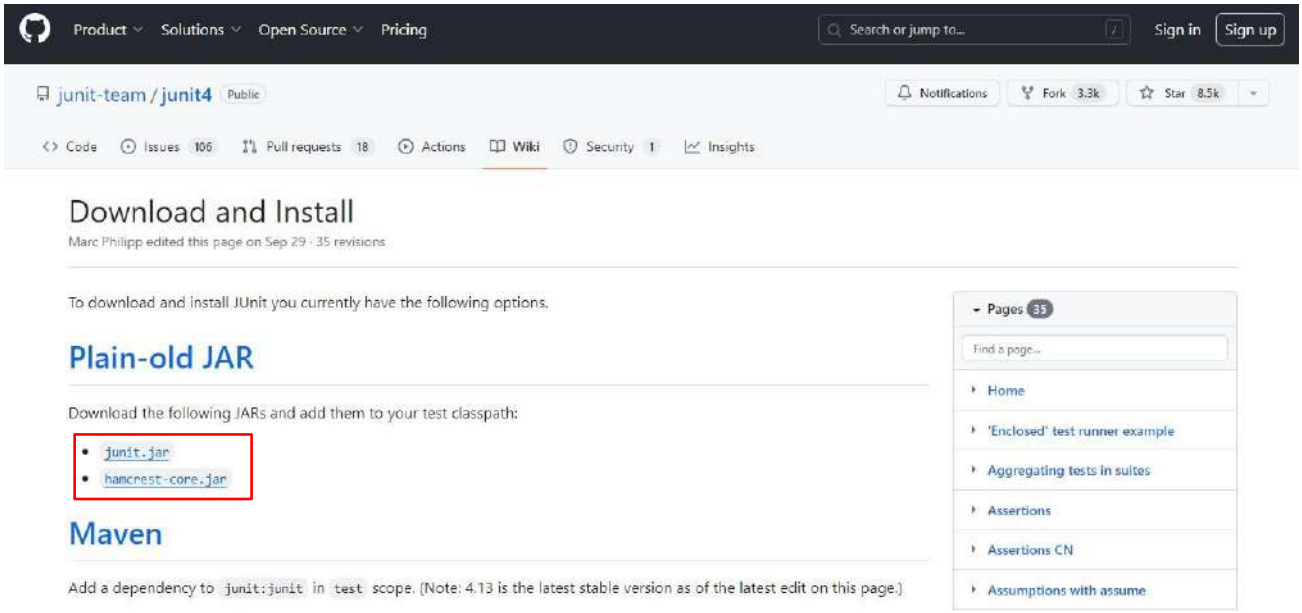
Step 1: Go web browser □ search for 'Junit 4' □ select the first link shown here.



Step 2: Click on 'Download & Install' option.



Step 3: Now click on 'junit.jar' file here.



Step 4: Then select the 'junit-4.13.2.jar' version & download it.

junit/junit/4.13.2

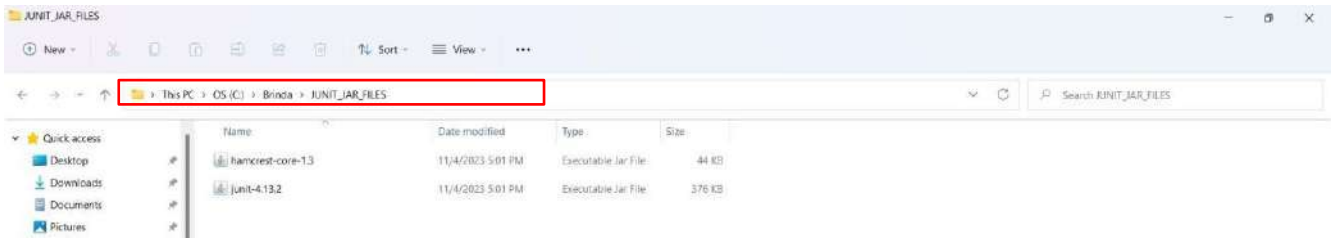
../		
junit-4.13.2-javadoc.jar	2021-02-13 16:31	1674580
junit-4.13.2-javadoc.jar.asc	2021-02-13 16:31	833
junit-4.13.2-javadoc.jar.asc.md5	2021-02-13 16:31	32
junit-4.13.2-javadoc.jar.asc.sha1	2021-02-13 16:31	40
junit-4.13.2-javadoc.jar.md5	2021-02-13 16:31	32
junit-4.13.2-javadoc.jar.sha1	2021-02-13 16:31	40
junit-4.13.2-sources.jar	2021-02-13 16:31	234540
junit-4.13.2-sources.jar.asc	2021-02-13 16:31	833
junit-4.13.2-sources.jar.asc.md5	2021-02-13 16:31	32
junit-4.13.2-sources.jar.asc.sha1	2021-02-13 16:31	40
junit-4.13.2-sources.jar.md5	2021-02-13 16:31	32
junit-4.13.2-sources.jar.sha1	2021-02-13 16:31	40
junit-4.13.2.jar	2021-02-13 16:31	384581
junit-4.13.2.jar.asc	2021-02-13 16:31	833
junit-4.13.2.jar.asc.md5	2021-02-13 16:31	32
junit-4.13.2.jar.asc.sha1	2021-02-13 16:31	40
junit-4.13.2.jar.md5	2021-02-13 16:31	32
junit-4.13.2.jar.sha1	2021-02-13 16:31	40
junit-4.13.2.pom	2021-02-13 16:31	27018
junit-4.13.2.pom.asc	2021-02-13 16:31	833
junit-4.13.2.pom.asc.md5	2021-02-13 16:31	32
junit-4.13.2.pom.asc.sha1	2021-02-13 16:31	40
junit-4.13.2.pom.md5	2021-02-13 16:31	32
junit-4.13.2.pom.sha1	2021-02-13 16:31	40

Step 5: Also select ‘hamcrest-core.jar’ file and download the ‘hamcrest-core-1.3.jar’ version here.

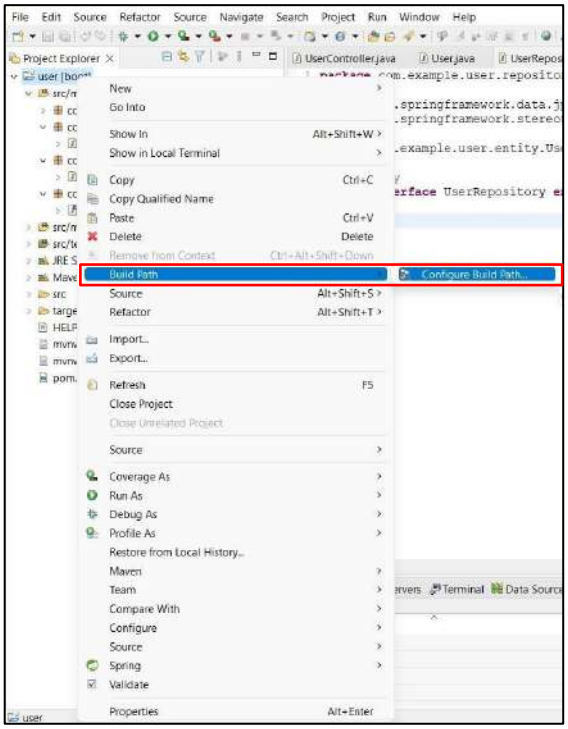
org/hamcrest/hamcrest-core/1.3

../		
hamcrest-core-1.3-javadoc.jar	2012-07-09 21:08	242519
hamcrest-core-1.3-javadoc.jar.asc	2012-07-09 21:08	499
hamcrest-core-1.3-javadoc.jar.asc.md5	2012-07-09 21:08	32
hamcrest-core-1.3-javadoc.jar.asc.sha1	2012-07-09 21:08	40
hamcrest-core-1.3-javadoc.jar.md5	2012-07-09 21:08	32
hamcrest-core-1.3-javadoc.jar.sha1	2012-07-09 21:08	40
hamcrest-core-1.3-sources.jar	2012-07-09 21:08	32624
hamcrest-core-1.3-sources.jar.asc	2012-07-09 21:08	499
hamcrest-core-1.3-sources.jar.asc.md5	2012-07-09 21:08	32
hamcrest-core-1.3-sources.jar.asc.sha1	2012-07-09 21:08	40
hamcrest-core-1.3-sources.jar.md5	2012-07-09 21:08	32
hamcrest-core-1.3-sources.jar.sha1	2012-07-09 21:08	40
hamcrest-core-1.3.jar	2012-07-09 21:08	45024
hamcrest-core-1.3.jar.asc	2012-07-09 21:08	499
hamcrest-core-1.3.jar.asc.md5	2012-07-09 21:08	32
hamcrest-core-1.3.jar.asc.sha1	2012-07-09 21:08	40
hamcrest-core-1.3.jar.md5	2012-07-09 21:08	32
hamcrest-core-1.3.jar.sha1	2012-07-09 21:08	40
hamcrest-core-1.3.pom	2012-07-09 21:08	766
hamcrest-core-1.3.pom.asc	2012-07-09 21:08	499
hamcrest-core-1.3.pom.asc.md5	2012-07-09 21:08	32
hamcrest-core-1.3.pom.asc.sha1	2012-07-09 21:08	40
hamcrest-core-1.3.pom.md5	2012-07-09 21:08	32
hamcrest-core-1.3.pom.sha1	2012-07-09 21:08	40

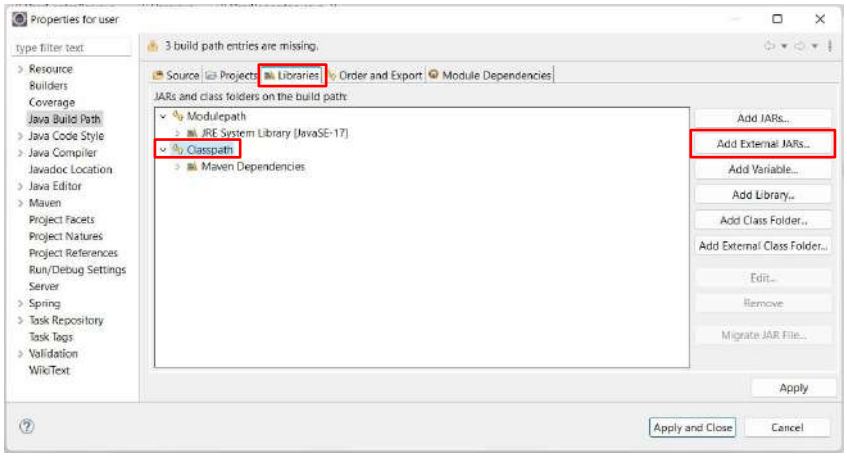
Step 6: Now select any drive & create a new folder within that folder, create a new folder named ‘JUNIT_JAR_FILES’ & copy the downloaded jar files & paste it here.



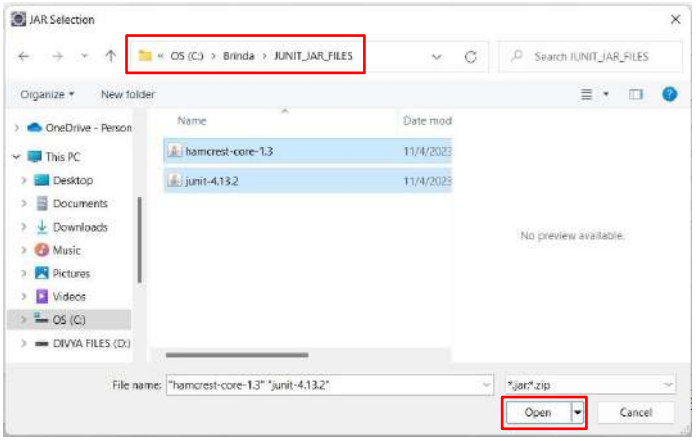
Step 7: Go to Eclipse □ user □ Build Path □ Configure Build Path here.



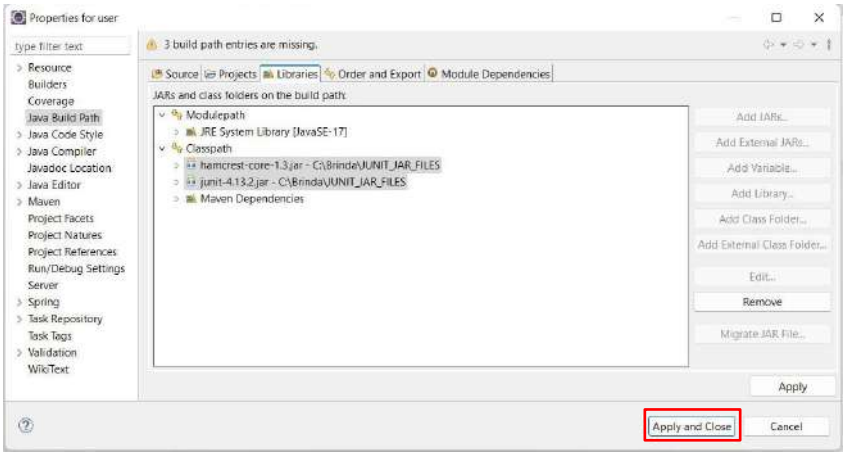
Step 8: Go to Libraries Classpath Add External JARs.. here.



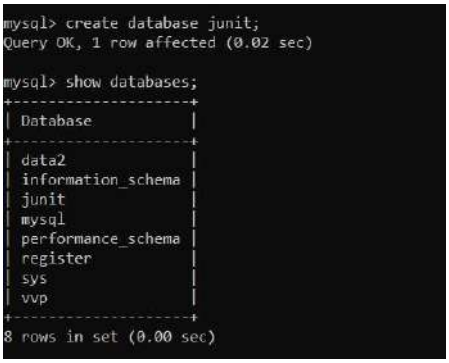
Step 9: Select the two jar files downloaded here & click on ‘Open’.



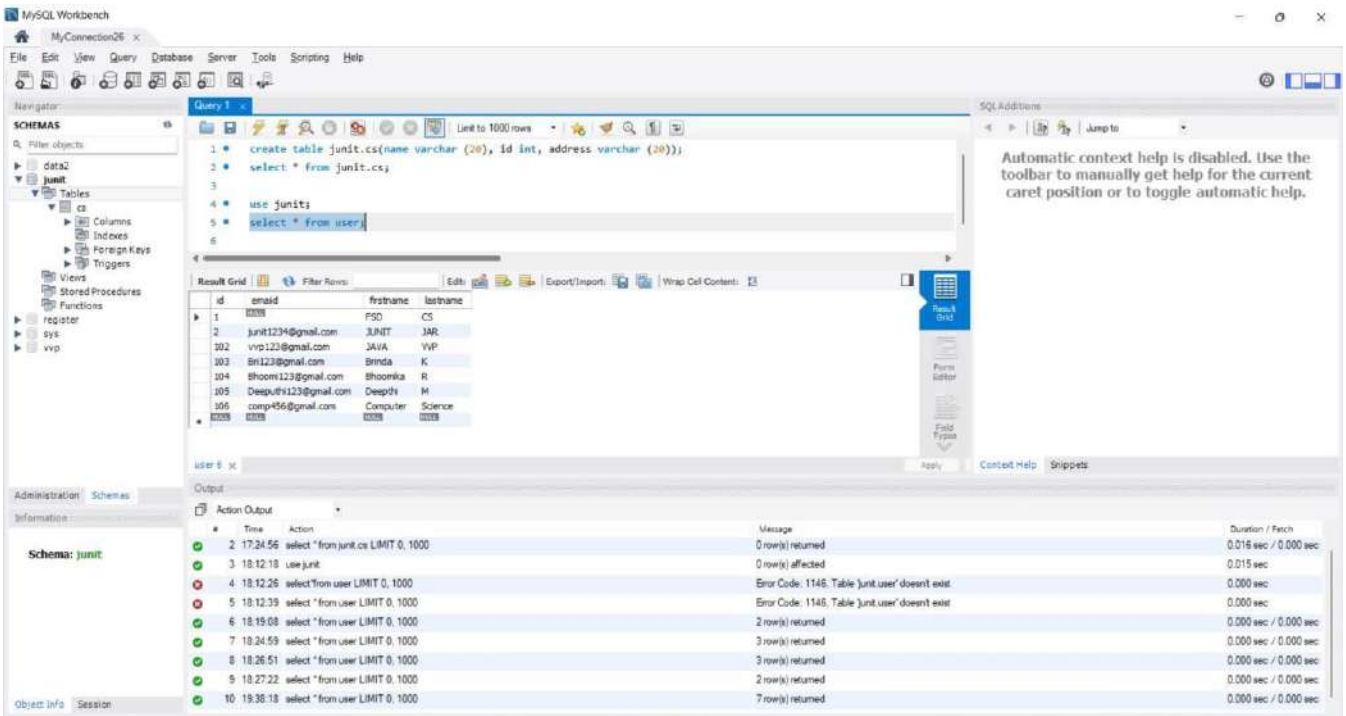
Step 10: Now click on ‘Apply and Close’ here.



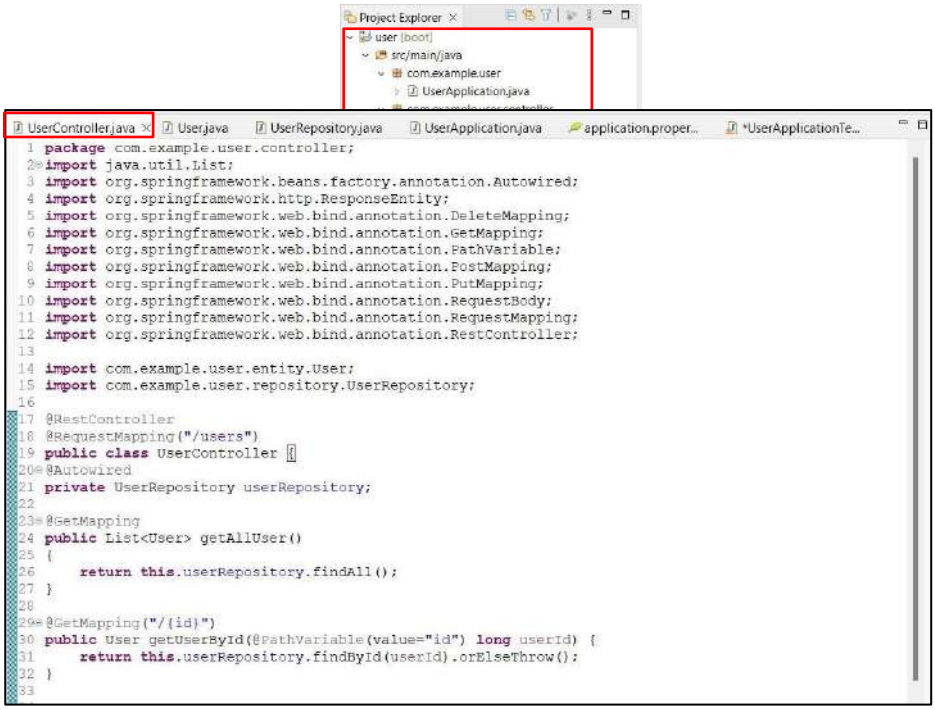
Step 11: Now create a new database in MySQL Command Line Client.



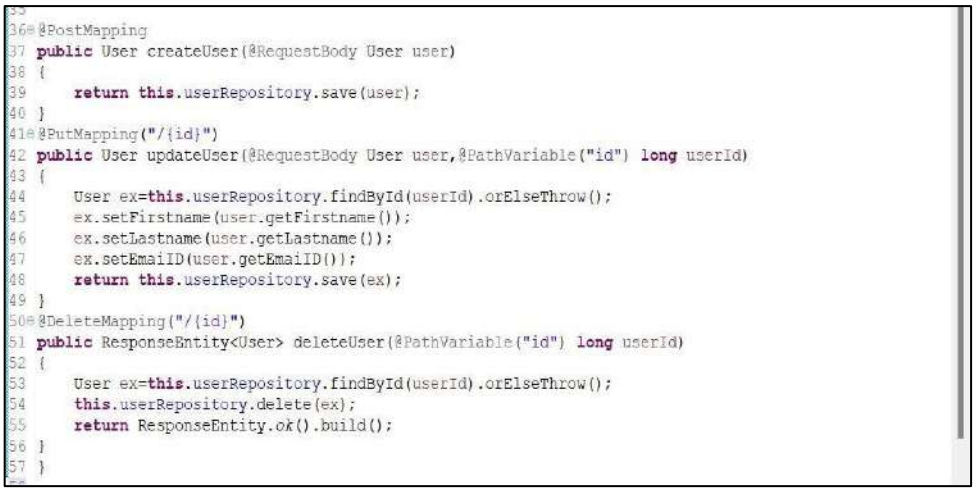
Step 12: Go to MySQL Workbench and create few more tables using Postman as well as with CRUD options.



Step 13: Follow the same procedures followed in CRUD operations and add code to the classes created.



Step 14: Now add code to ‘UseController.java’ file here.



Step 15: Now add code to ‘User.java’ file here.

```java
package com.example.user.entity;

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

@Entity
@Table(name="user")
public class User {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;
    private String firstname;
    private String lastname;
    private String emailID;

    public User() {

    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getFirstname() {
        return firstname;
    }
    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
    public String getLastname() {
        return lastname;
    }
    public void setLastname(String lastname) {
        this.lastname = lastname;
    }
    public String getEmailID() {
        return emailID;
    }
    public void setEmailID(String emailID) {
        this.emailID = emailID;
    }
    public User(Long id, String firstname, String lasttname, String emailID) {
        super();
        this.id = id;
        this.firstname = firstname;
        this.lastname = lasttname;
        this.emailID = emailID;
    }

}
```

Step 16: Now add code to ‘UseRepository.java’ file here.

```java
package com.example.user.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.user.entity.User;

@Repository
public interface UserRepository extends JpaRepository<User,Long>
{
}
```

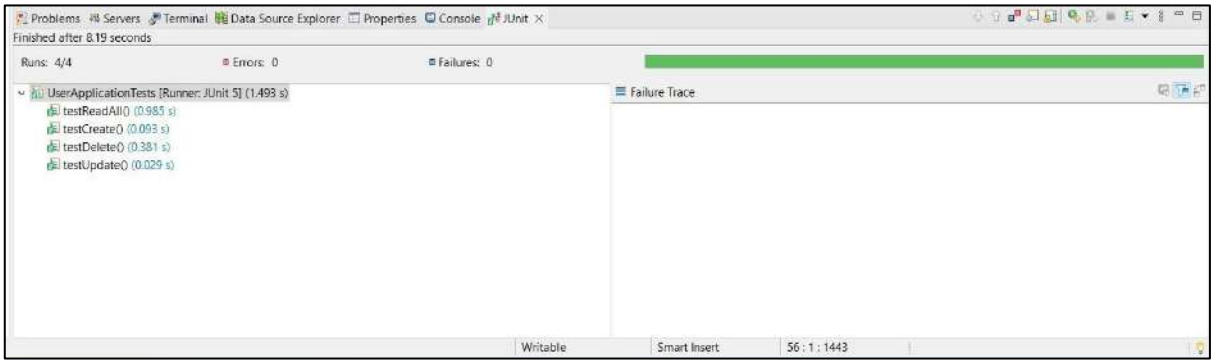
Step 17: Now add code to ‘UseApplication.java’ file where main method class present here.

```java
package com.example.user;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class UserApplication {

    public static void main(String[] args) {
        SpringApplication.run(UserApplication.class, args);
    }

}
```

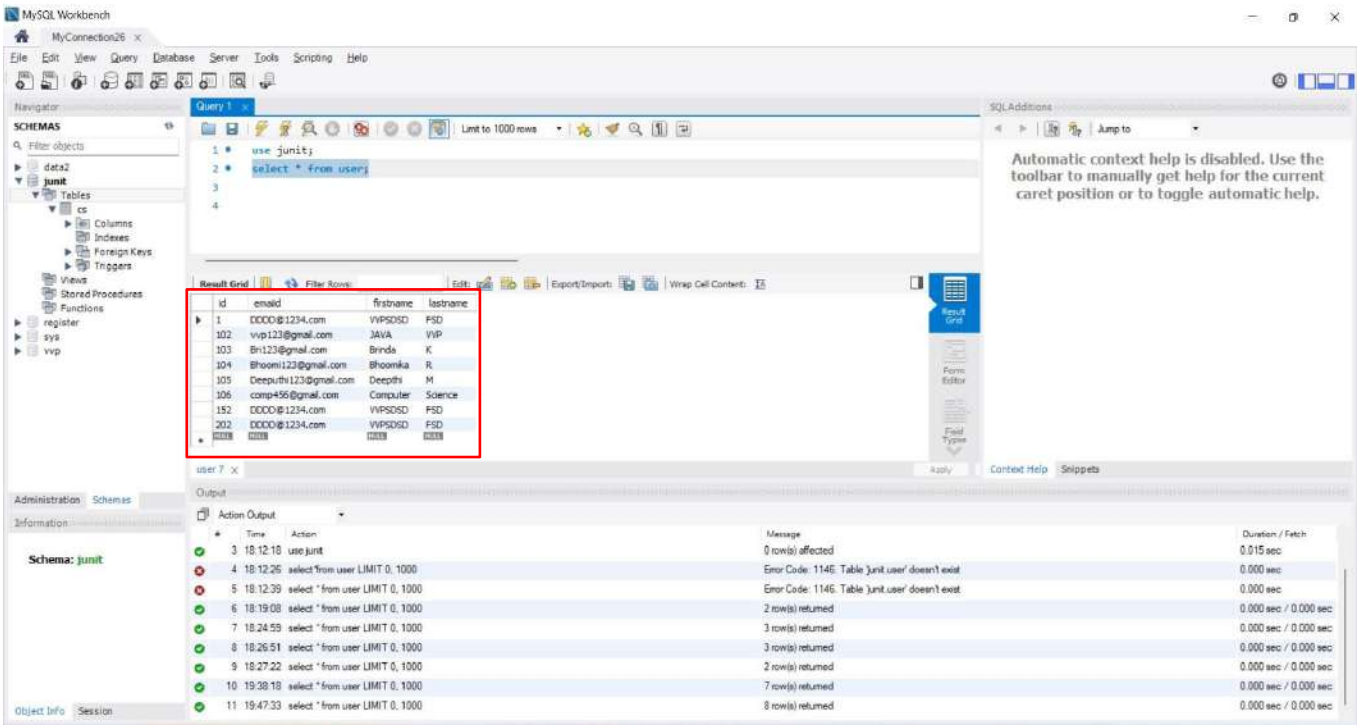
Step 18: Now add code to ‘application.properties’ file here.

Step 19: Go to ‘src/test/java’ folder in the Project explorer section & add code junit test code to ‘UserApplicationTest.java’ class file as shown here.

Step 20: Now right click on the ‘UserApplicationTest.java’ file □ Run As □ 2Junit Test.

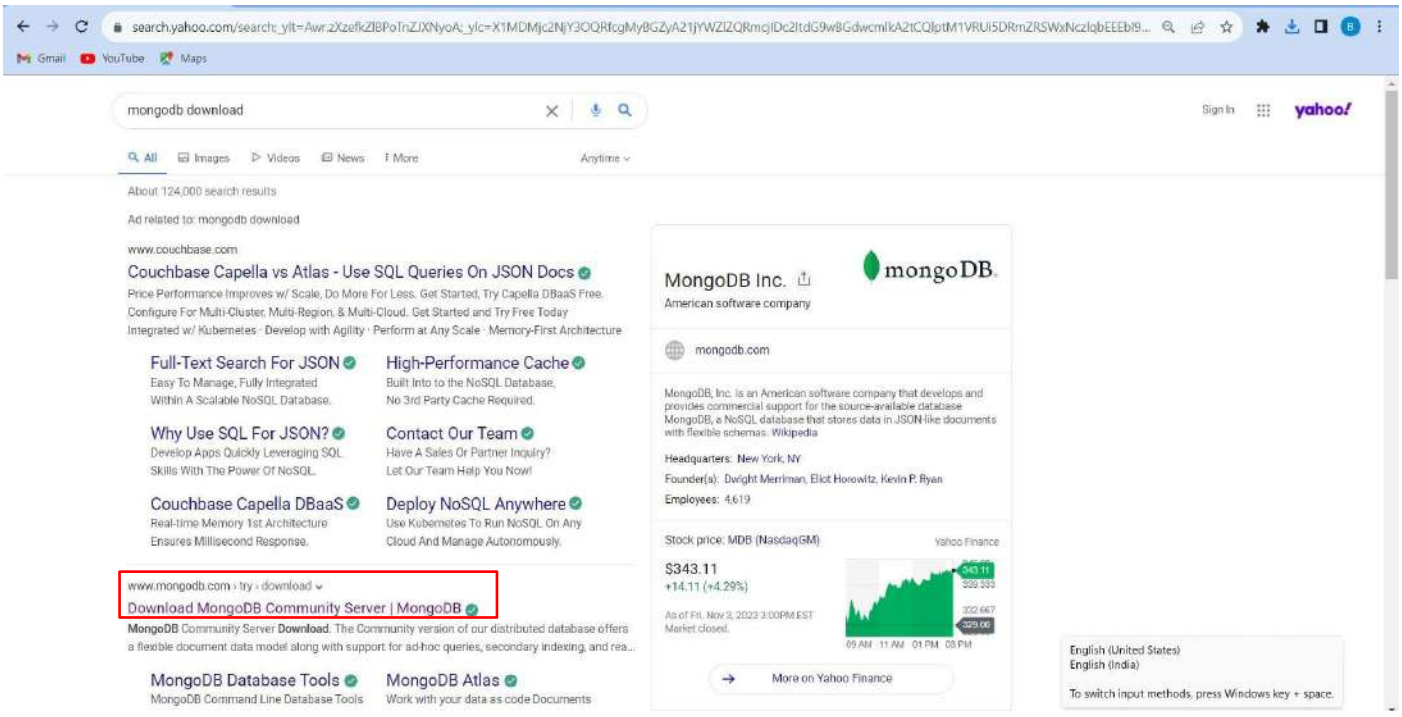


Step 21: The above test condition is successfully working in MySQL Workbench as shown below.

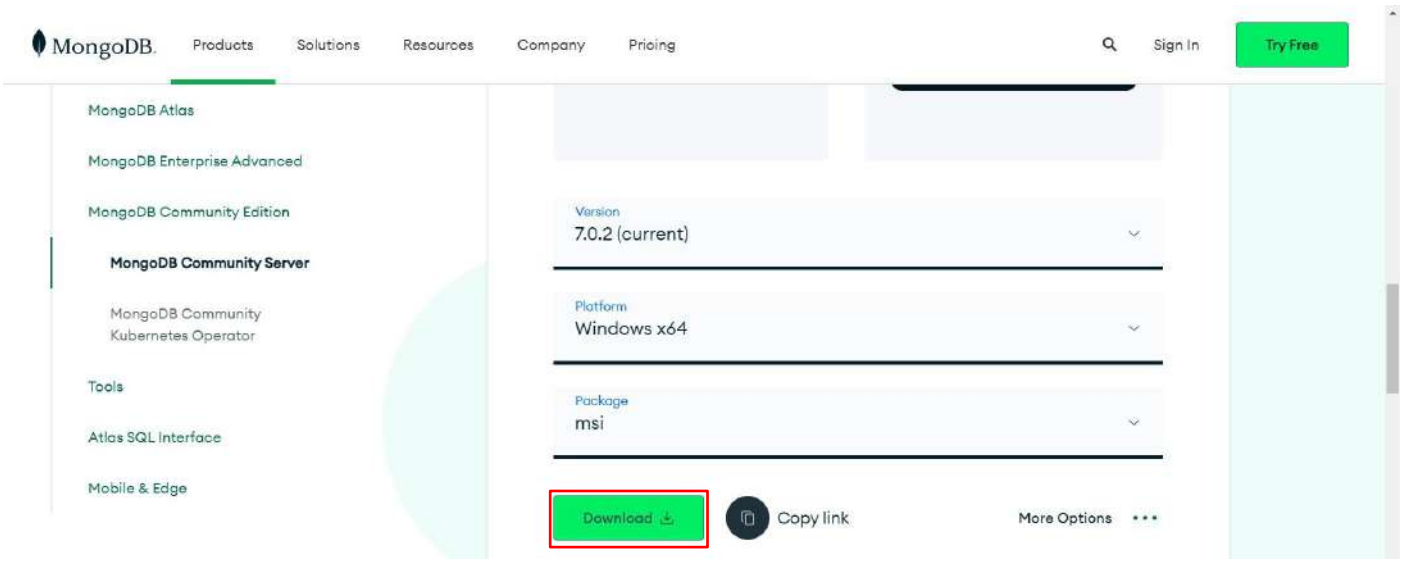


3. **Installation of Mongodb, mongosh, compass.**

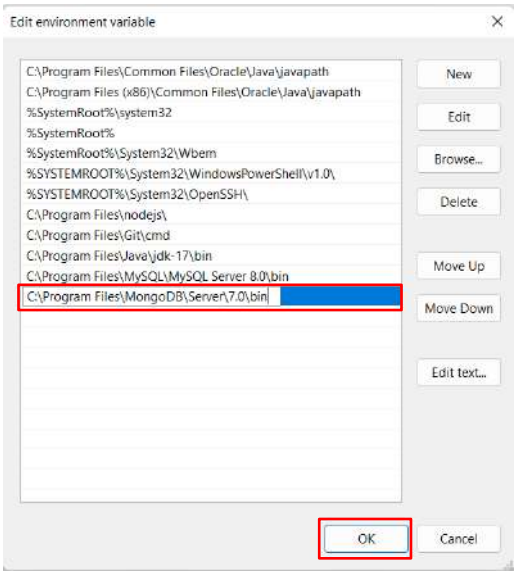
Step 1: Go to web browser Search for 'mongodb download' click on the link as shown below.



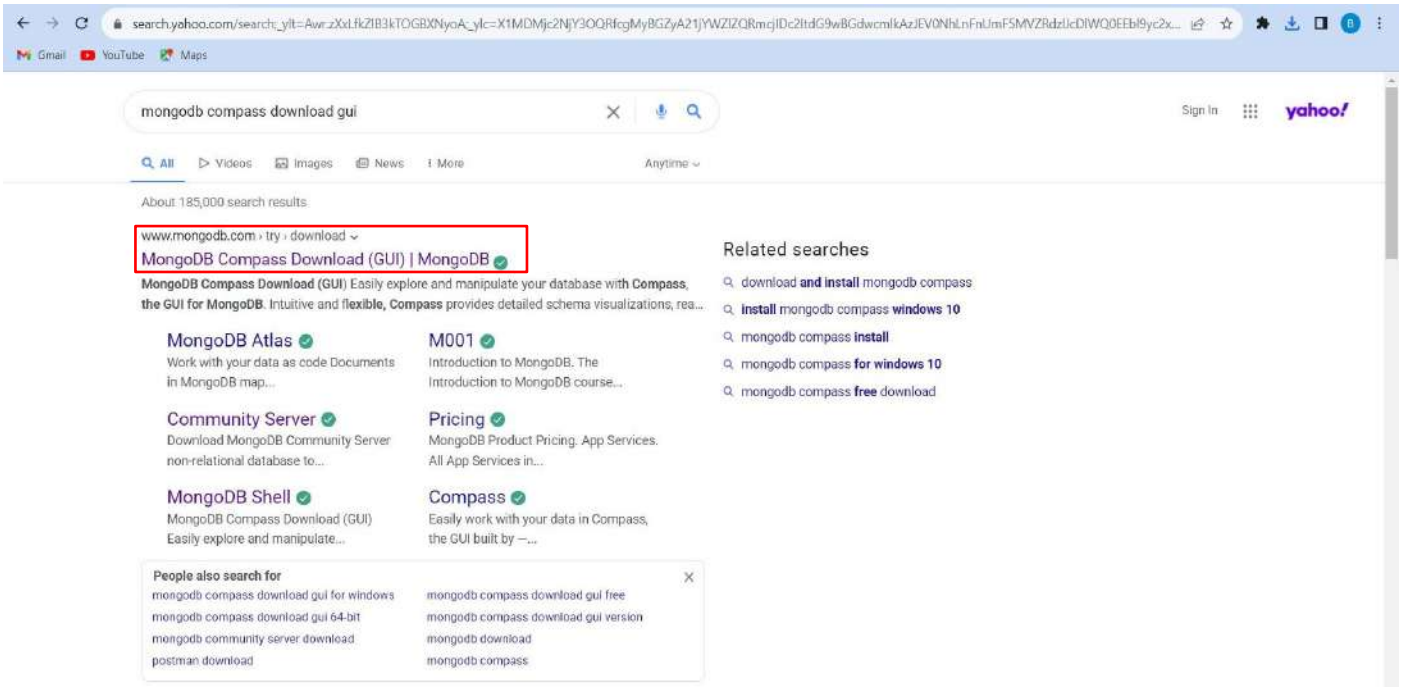
Step 2: Click on 'Download' here.



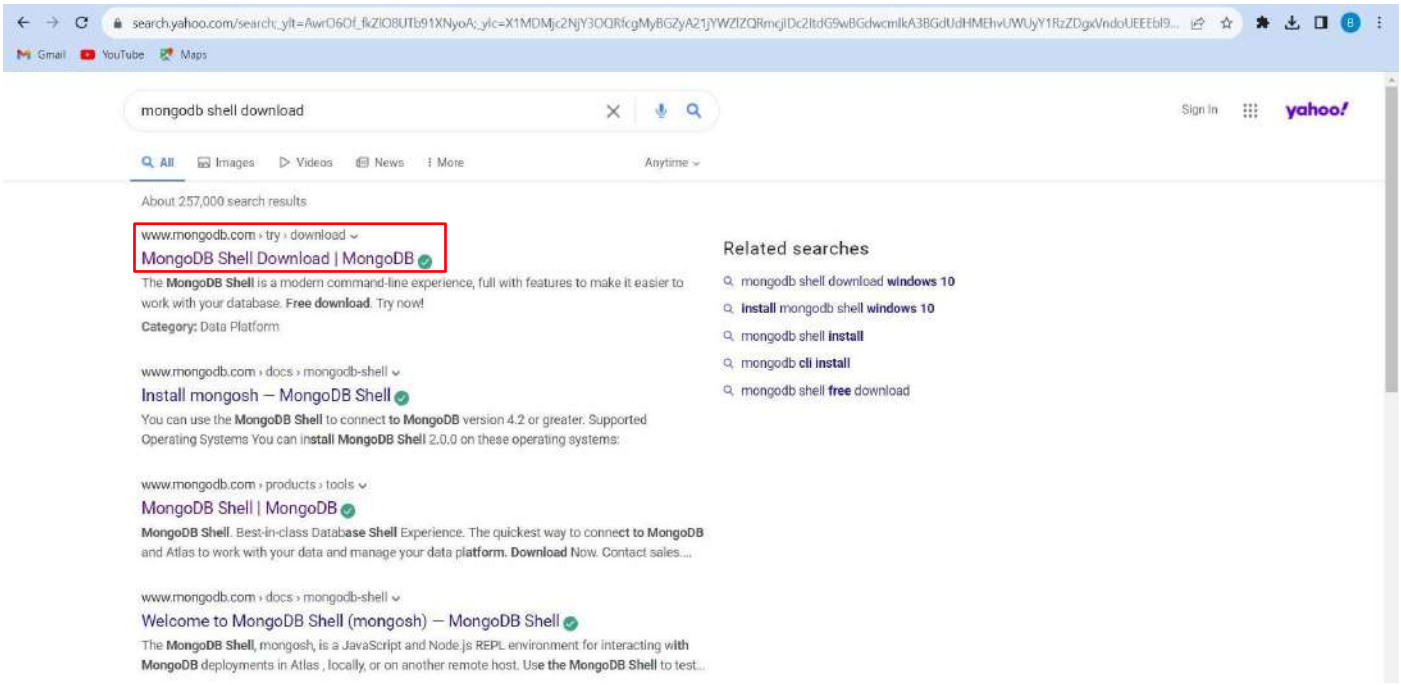
Step 2: Now add ‘Environment variables’ for the downloaded file.



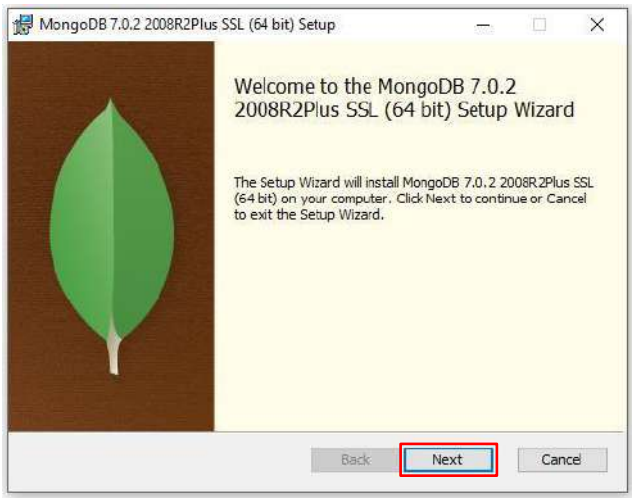
Step 3: Go to web browser search for ‘mongodb compass download (GUI)’ click on the first link here & download the MongoDB Compass.



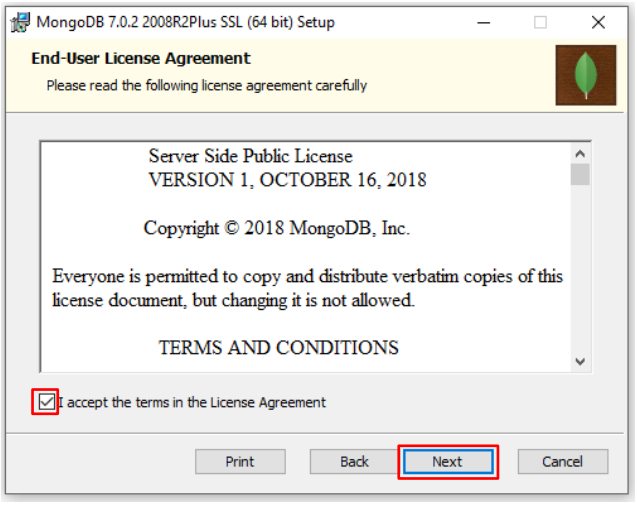
Step 4: Go to web browser □ search for ‘mongodb shell download’ □ click on the first link here & download the MongoDB Shell.



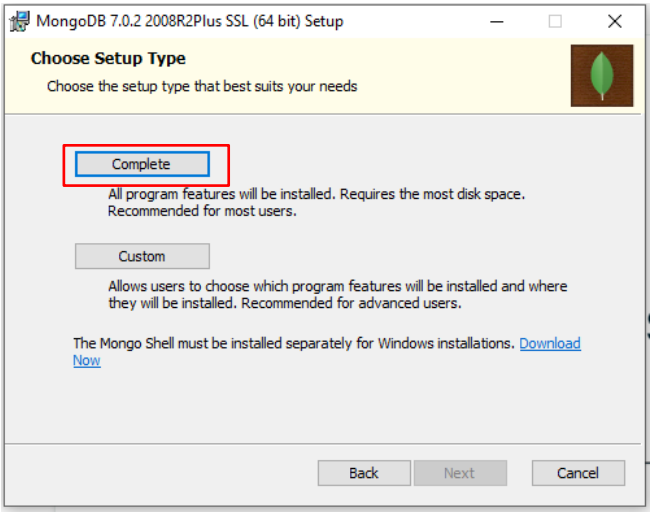
Step 5: Follow the ongoing steps to install mongodb compass, click on ‘Next’.



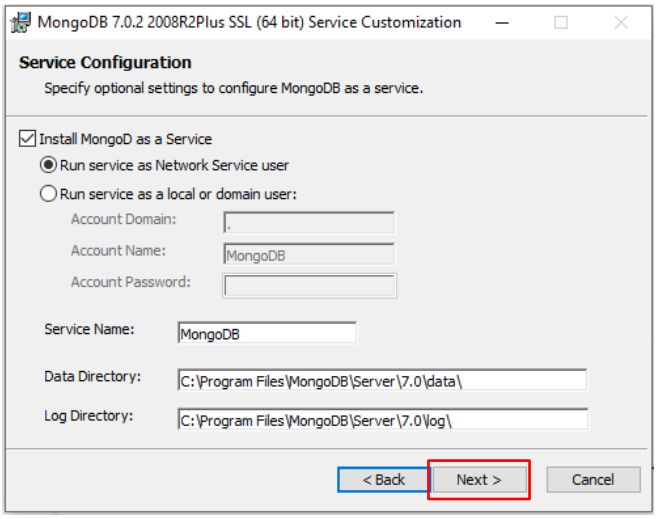
Step 6: Enable the checkbox & click on ‘Next’.



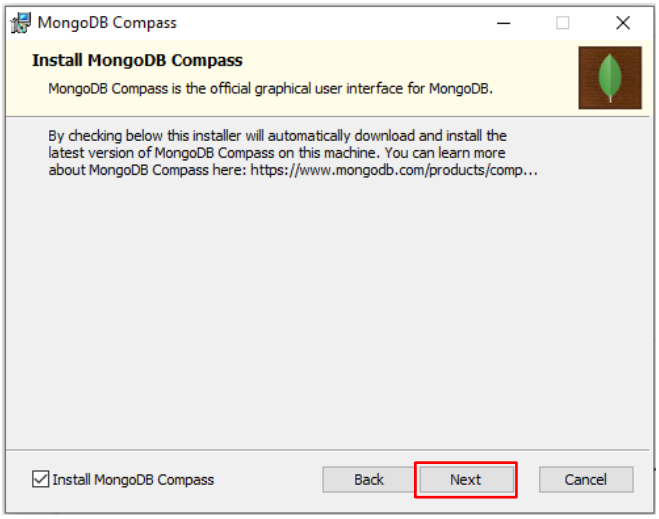
Step 7: Click on ‘complete’ here.



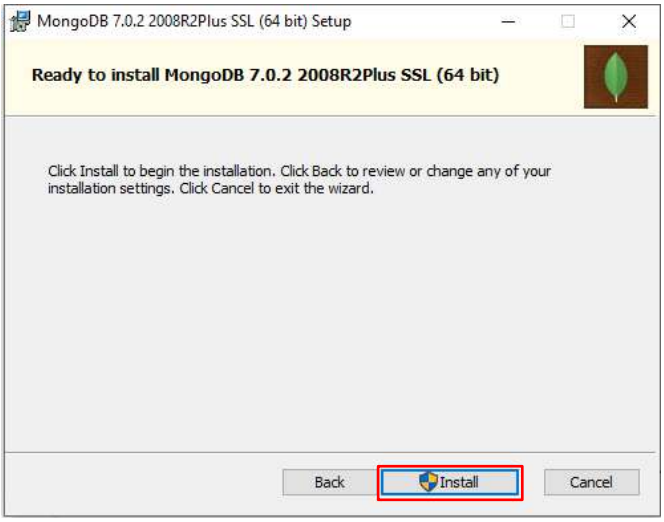
Step 8: Just click on ‘Next’ here.



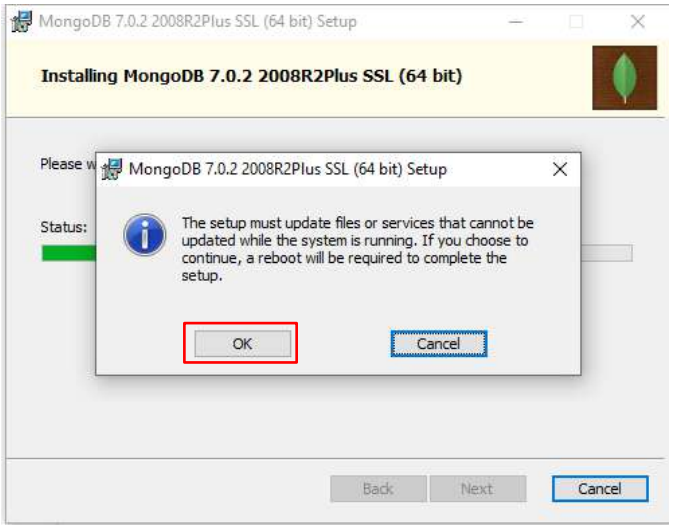
Step 9: Click on ‘Next’.



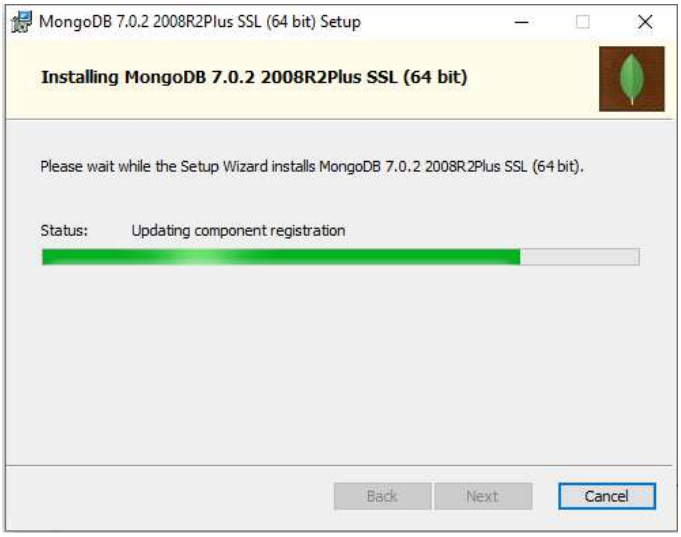
Step 10: Now click on ‘Install’.



Step 11: Click on ‘OK’ here.

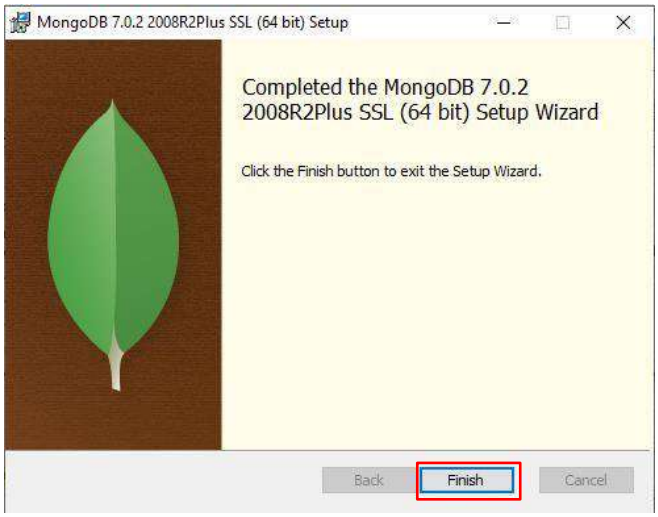


Step 12: Wait until the installation is complete.



Step 13: MongoDB Compass is ready to launch.

Step 14: Then click on ‘Finish’ here.



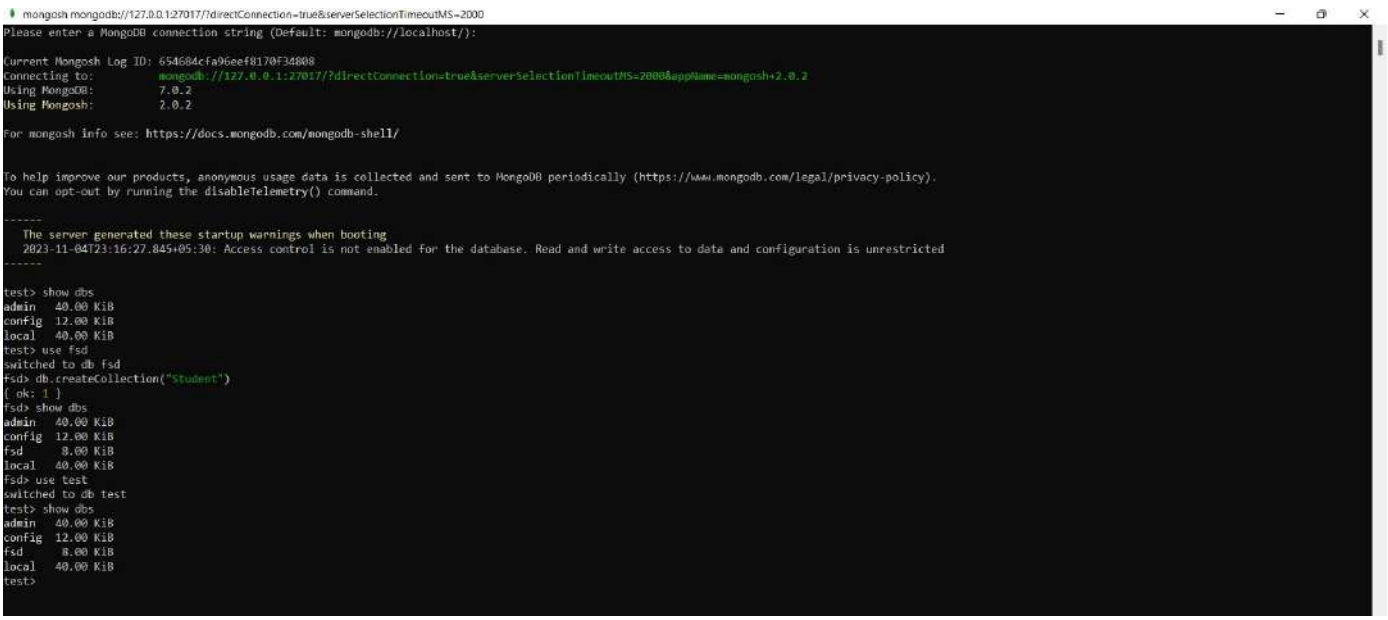
Step 15: Once all installation done, To verify mongod ,Open cmd prompt –execute (mongod --version and mongosh --version)

- For mongodb server –msi installer version 7 and above
- For mongosh-extract zip file in a secure folder
- For Mongodb compass run the application(.exe) file.

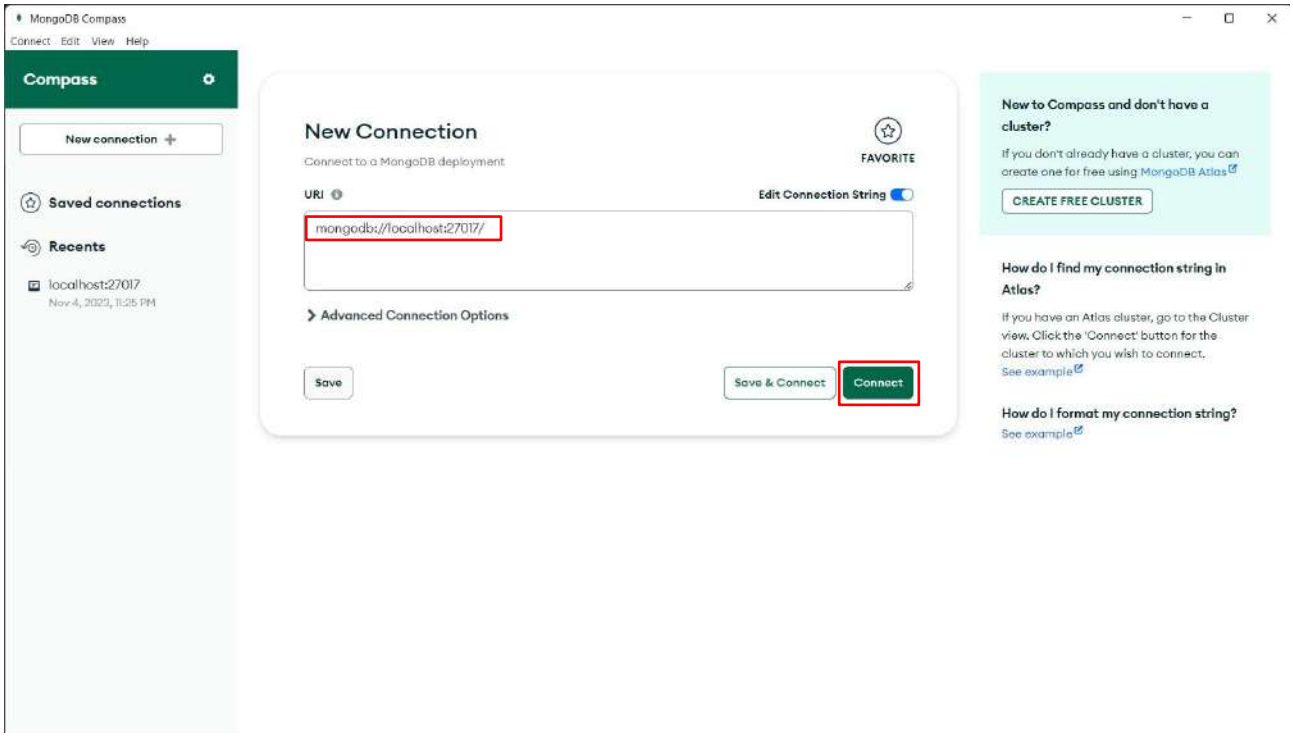
Step 16: Next in search bar type ‘mongoshell’ ->click and open and press enter

Next to create new database ‘use fsd’ as shown below and ‘show dbs’ shows all the databases created.

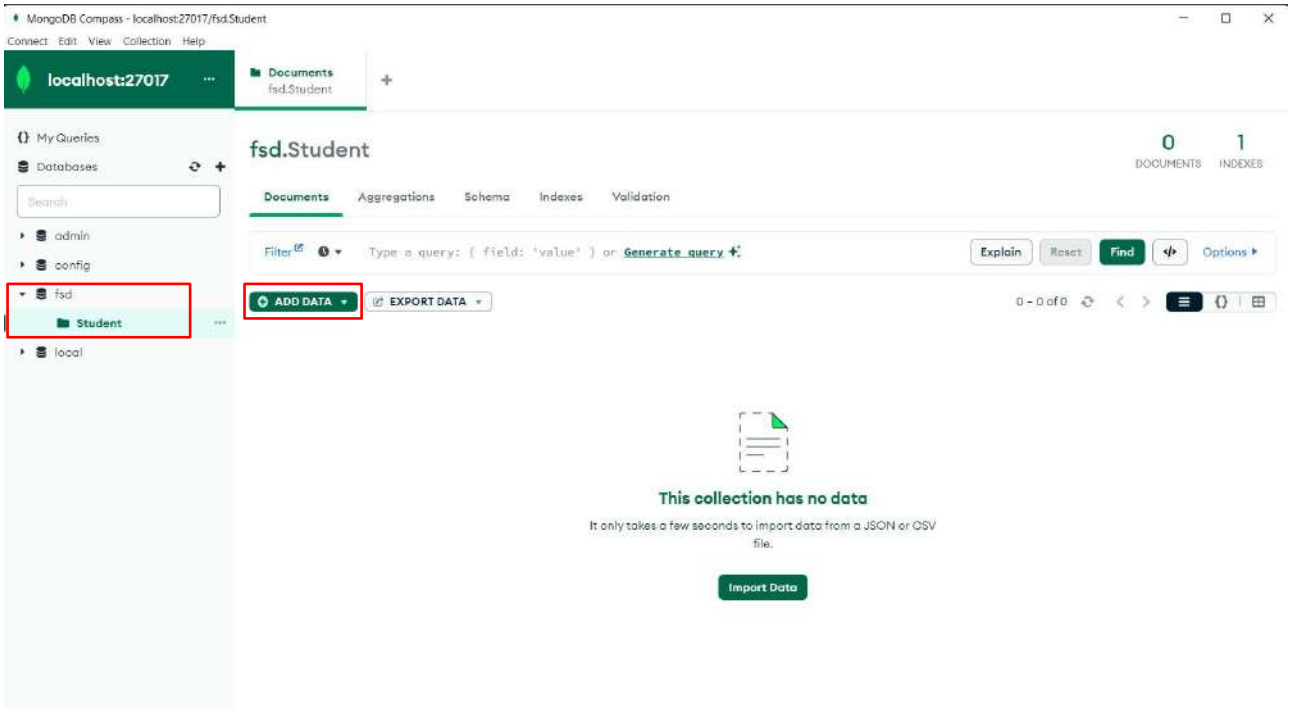
Remember: In MongoDB, a database is not actually created until it gets content!



Step 17: Now search for ‘mongoDB’ compass in start menu and open it. Now click on ‘Connect’ to connect with the localhost here.



Step 18: Select the database created & click on ‘ADD DATA’ to insert data.



Week-10

1. Create and Drop database

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

Syntax:

'use Database_Name'

Example: If you want to use a database with name <mydb>,

'use mydb'

- To check your currently selected database, use the command db

'db'

- Your created database (mydb) is not present in list. To display database, you need to insert at least one document into it.

'db.data.insert({"name":"MongoDB"})' 'show

dbs'

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Please enter a MongoDB connection string (Default: mongodb://localhost/):

Current Mongosh Log ID: 6558d1500def099b126fba7f
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.0.2
Using MongoDB:          7.0.2
Using Mongosh:          2.0.2

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

------
   The server generated these startup warnings when booting
   2023-11-18T20:15:01.463+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use mydb
switched to db mydb
mydb> db
mydb
mydb> db.data.insert({"name":"MongoDB"})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6558d1740def099b126fba80") }
}
mydb> show dbs
admin    40.00 KiB
config  108.00 KiB
local    72.00 KiB
mydb      8.00 KiB
```

- The **dropDatabase** command is used to drop a database. It also deletes the associated data files. It operates on the current database.

Syntax:

'db.dropDatabase()'

```
mydb> db.dropDatabase()
{ ok: 1, dropped: 'mydb' }
mydb> show dbs
admin    40.00 KiB
config  108.00 KiB
local    72.00 KiB
mydb>
```

2. Create and Drop Collections

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

Syntax:

'db.createCollection(name, options)'

In MongoDB, you don't need to create collection. MongoDB creates collection automatically, when you insert some document.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Please enter a MongoDB connection string (Default: mongodb://localhost/):

Current Mongosh Log ID: 6558d4c17cba999bca49f5d1
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.0.2
Using MongoDB:          7.0.2
Using Mongosh:          2.0.2

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

------
   The server generated these startup warnings when booting
   2023-11-18T20:15:01.463+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> db.createCollection("mycollection")
{ ok: 1 }
test> db.mycollection.insert({"course":"Java"})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6558d53a7cba999bca49f5d2") }
}
test> show collections
mycollection
test>
```

MongoDB's **db.collection.drop()** is used to drop a collection from the database.

Syntax:

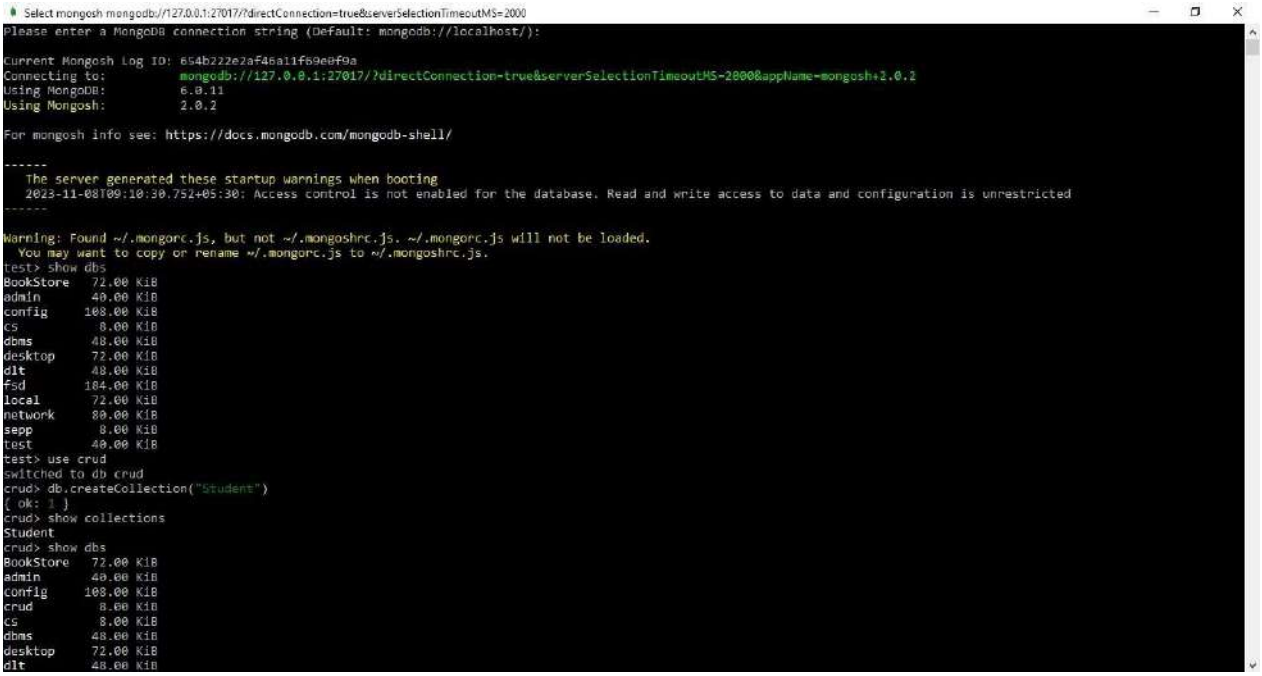
'db.Collection_Name.drop()'

drop() method will return true, if the selected collection is dropped successfully, otherwise it will return false.

```
test> db.mycollection.drop()
true
test> show collections

test>
```

3. CRUD Operations in MongoDB.

Step 1: Open Mongosh which will directly connect to your default localhost port address →Type **show dbs** → Create a new database with **name 'crud'** & create a new collection with name **'Student'** → Now give the name as **'show dbs'** to check the created database as shown below.



- **MongoDB insert documents**

In MongoDB , the **db.collection.insert()** method is used to add or insert new documents into a collection in your database.

Note: Here I've inserted two insert documents as **MongoDB & FSD.**

```
crud> db.Student.insert({"name":"MongoDB"})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("654b228c2af46a11f69e0f9b") }
}
crud> db.Student.insert({"course":"FSD"})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("654b22a42af46a11f69e0f9c") }
}
```

To read the document enter the command as **'db.Student.find( )'** & to read the document in a formatted way, enter the command as **'db.Student.find( ).pretty( )'**.

```
crud> db.Student.find()
[
  { _id: ObjectId("654b228c2af46a11f69e0f9b"), name: 'MongoDB' },
  { _id: ObjectId("654b22a42af46a11f69e0f9c"), course: 'FSD' }
]
crud> db.Student.find().pretty()
[
  { _id: ObjectId("654b228c2af46a11f69e0f9b"), name: 'MongoDB' },
  { _id: ObjectId("654b22a42af46a11f69e0f9c"), course: 'FSD' }
]
```

- MongoDB insert multiple documents: If you want to insert multiple documents in a collection, you have to pass an array of documents to thedb.collection.insert() method.

- **Create an array of documents:**

Define a variable named **data** that hold an array of documents to insert.

```
crud> var data=
... [
... {
... course:"Java",
... name:"Computer",
... ussn:"488CS123",
... category:"VVP"
... },
... {
... course:"Python",
... name:"Science",
... ussn:"123VVP488",
... category:"CS"
... }
... ]
```

- **Inserts the documents:**

Pass this **data** array to the db.collection.insert() method as,

db.Student.insert(data)

```
crud> db.Student.insert(data)
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("654b23772af46a11f69e0f9d"),
    '1': ObjectId("654b23772af46a11f69e0f9e")
  }
}
```

To check whether the document is successfully inserted or not, use the command as **db.Student.find( )**

```
crud> db.Student.find()
[
  { _id: ObjectId("654b228c2af46a11f69e0f9b"), name: 'MongoDB' },
  { _id: ObjectId("654b22a42af46a11f69e0f9c"), course: 'FSD' },
  {
    _id: ObjectId("654b23772af46a11f69e0f9d"),
    course: 'Java',
    name: 'Computer',
    ussn: '488CS123',
    category: 'VVP'
  },
  {
    _id: ObjectId("654b23772af46a11f69e0f9e"),
    course: 'Python',
    name: 'Science',
    ussn: '123VVP488',
    category: 'CS'
  }
]
```

- **MongoDB update documents**

In MongoDB, update() method is used to update or modify the existing documents of a collection.

Syntax:

db.Collection_Name.**update**(SELECTIOIN_CRITERIA, UPDATED_DATA)

To check whether the document is successfully updated or not, use the command as '**db.Student.find( )'.**

```
crud> db.Student.update({'course':'Python'},{$set:{'course':'C Bash'}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
crud> db.Student.find()
[
  { _id: ObjectId("654b228c2af46a11f69e0f9b"), name: 'MongoDB' },
  { _id: ObjectId("654b22a42af46a11f69e0f9c"), course: 'FSD' },
  {
    _id: ObjectId("654b23772af46a11f69e0f9d"),
    course: 'C++',
    name: 'Computer',
    ussn: '488CS123',
    category: 'VVP'
  },
  {
    _id: ObjectId("654b23772af46a11f69e0f9e"),
    course: 'C Bash',
    name: 'Science',
    ussn: '123VVP488',
    category: 'CS'
  }
]
```

- **MongoDB Delete documents**

In MongoDB, the **db.collection.remove()** method is used to delete documents from a collection. The remove() method works on two parameters.

- JustOne: It removes only one document when set to true or 1.

Syntax:

db.collection_name.remove (DELETION_CRITERIA)

- Remove all documents: If you want to remove all documents from a collection, pass an empty query document {} to the remove()method.

Syntax:

db.fullstackdev.remove({})

```
mydata> db.Employee.remove({})
{ acknowledged: true, deletedCount: 3 }
mydata> db.Employee.find()

mydata>
```

The findOne() method: Apart from the find() method, there is **findOne()** method, that returns only one document.

Syntax:

db.Collection_Name.findOne()

```
crud> db.Student.findOne()
{ _id: ObjectId("654b228c2af46a11f69e0f9b"), name: 'MongoDB' }
crud> db.Student.find({}, {"name":1, _id:0}).sort({"name":1})
[
  {},
  { name: 'Computer' },
  { name: 'Database' },
  { name: 'FSD' },
  { name: 'MongoDB' },
  { name: 'Science' }
]
```

MongoDB - Sort Records:

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

Syntax: The basic syntax of sort() method is as follows

db.Collection_Name.find( ).sort({KEY:1})

```
crud> db.Student.find({}, {"course":-1, _id:0}).sort({"course":-1})
[
  { course: 'Python' },
  { course: 'Java' },
  { course: 'FSD' },
  { course: 'C++' },
  { course: 'C Bash' },
  {}
]
```

MongoDB - Limit Records:

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

Syntax: The basic syntax of limit() method is as follows −

db.Collection_Name.find().limit(NUMBER)

```
crud> db.Student.find({}, {"ussn":0, _id:0}).limit(2)
[ { name: 'MongoDB' }, { course: 'FSD' } ]
crud> db.Student.find({}, {"ussn":1, _id:1}).limit(2)
[
  { _id: ObjectId("654b228c2af46a11f69e0f9b") },
  { _id: ObjectId("654b22a42af46a11f69e0f9c") }
]
```

MongoDB Skip() Method:

Apart from limit() method, there is one more method **skip()** which also accepts number type argument and is used to skip the number of documents.

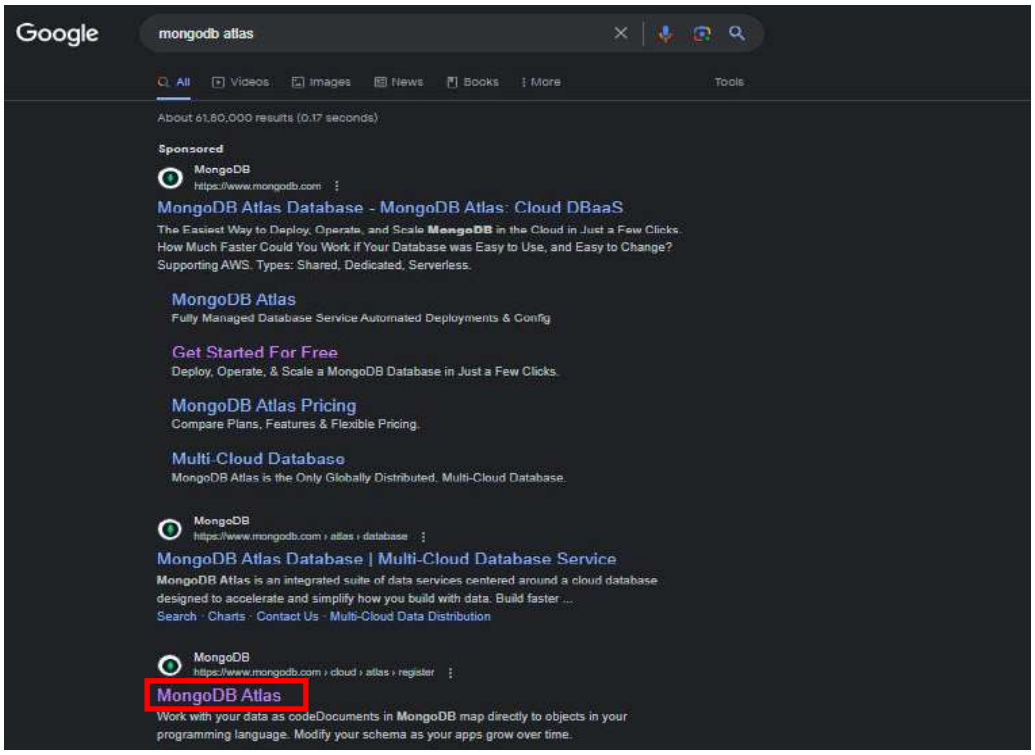
Syntax: The basic syntax of skip() method is as follows −

db.Collection_Name.find().limit(NUMBER).skip(NUMBER)

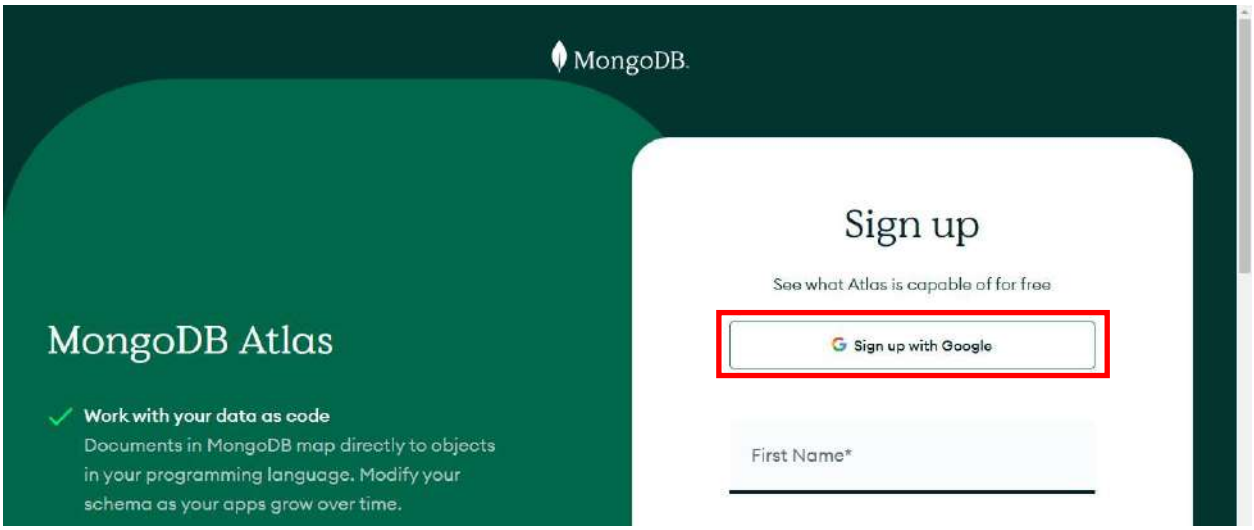
```
crud> db.Student.find({}, {"ussn":1, _id:0}).limit(2).skip(2)
[ { ussn: '488CS123' }, { ussn: '123VVP488' } ]
crud> db.Student.find({}, {"ussn":1, _id:0}).limit(2).skip(1)
[ {}, { ussn: '488CS123' } ]
```

4. How to Run Mongo DB on Cloud

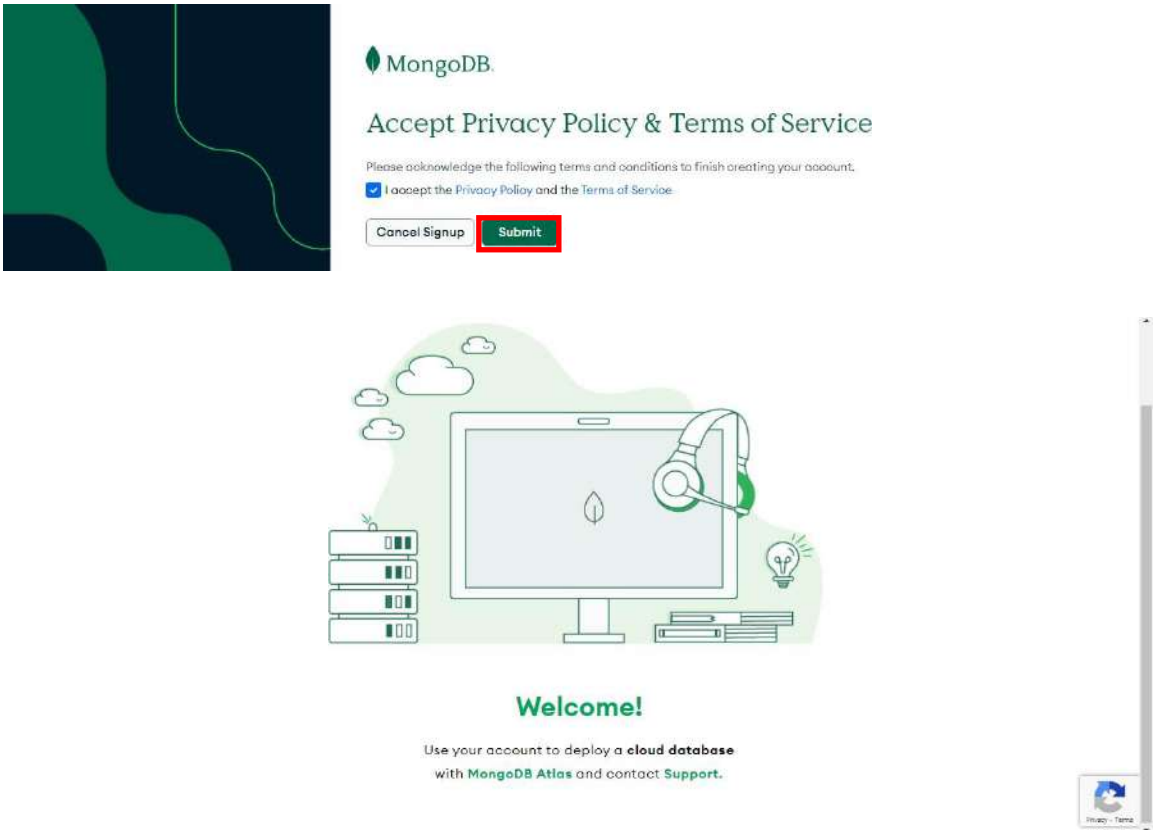
Step 1: Go to Web browser and search for '**MongoDB Atlas' &** click on the link as shown below.



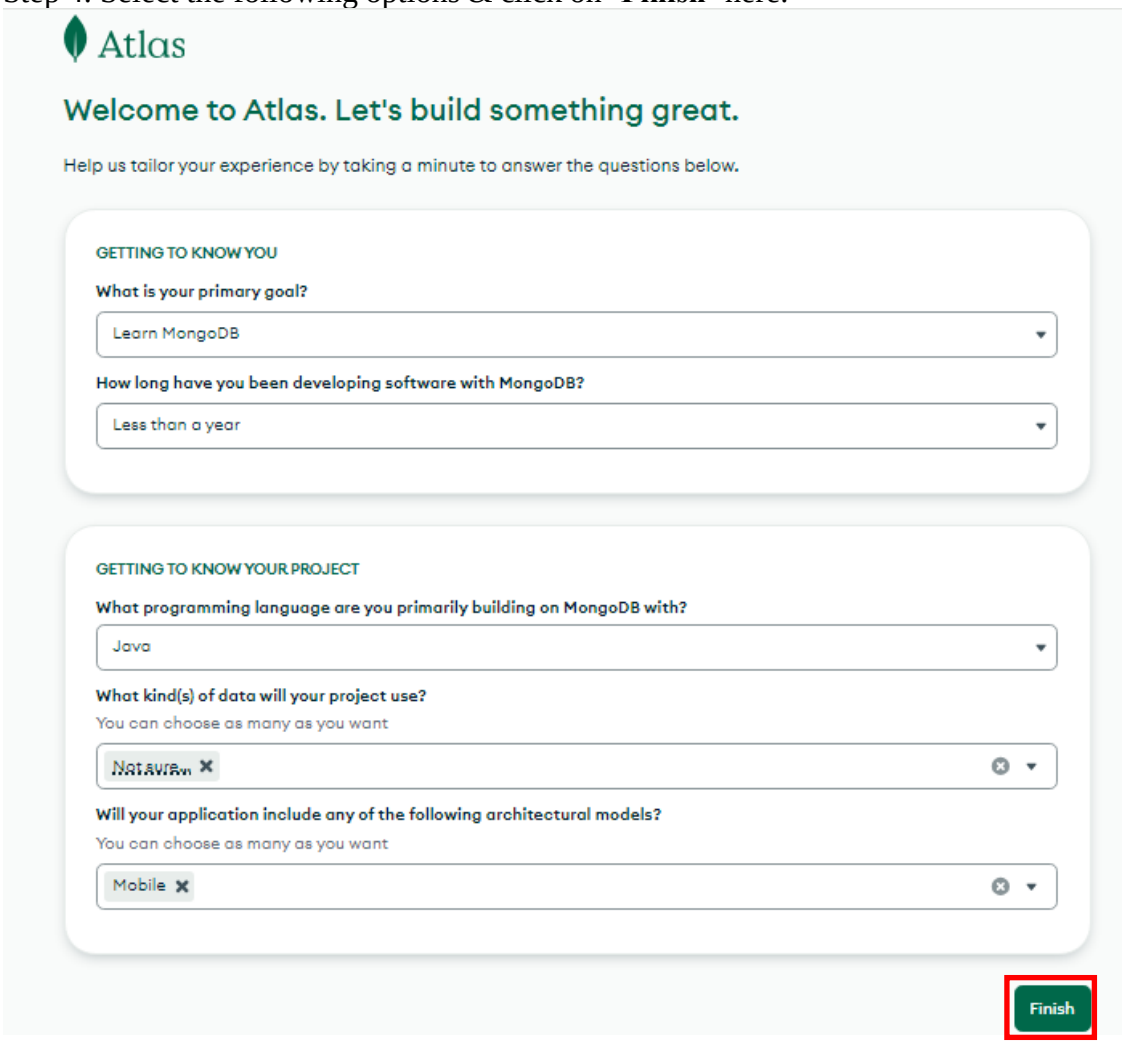
Step 2: Now click on **'Sign up with Google'** here.



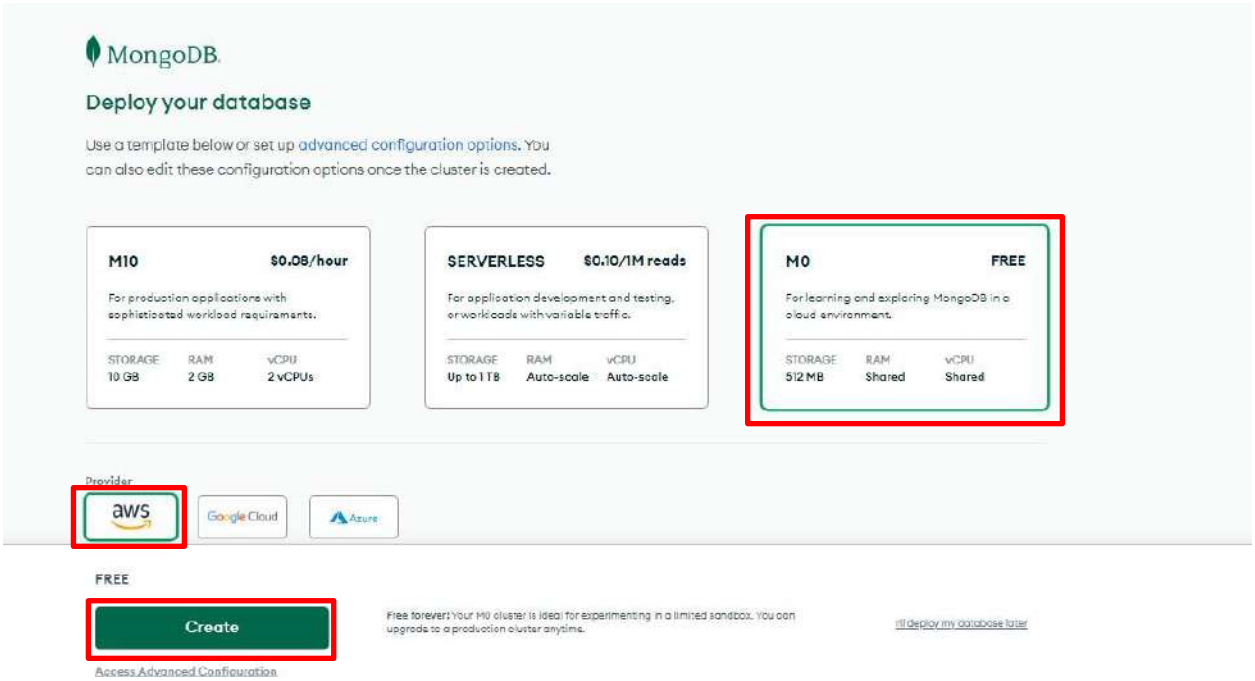
Step 3: Enable the checkbox & click on **'Submit'** option here.



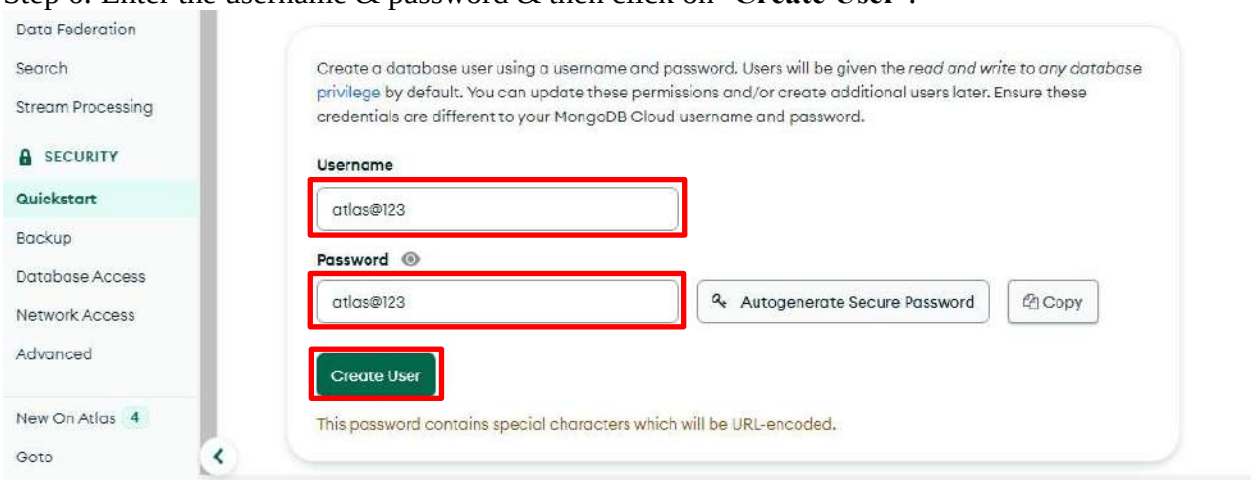
Step 4: Select the following options & click on **'Finish'** here.



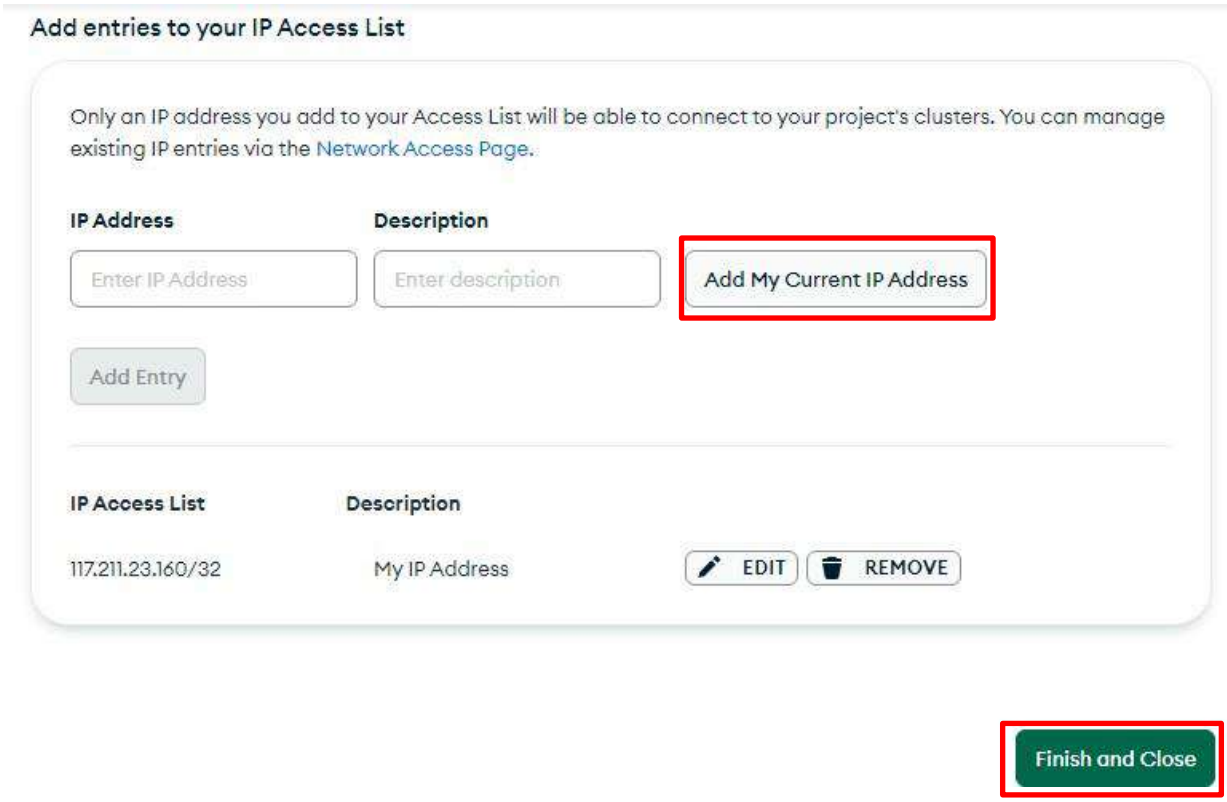
Step 5: Now select **'M0'** free version & also select **'aws'** her. And then click on **'Create'**.



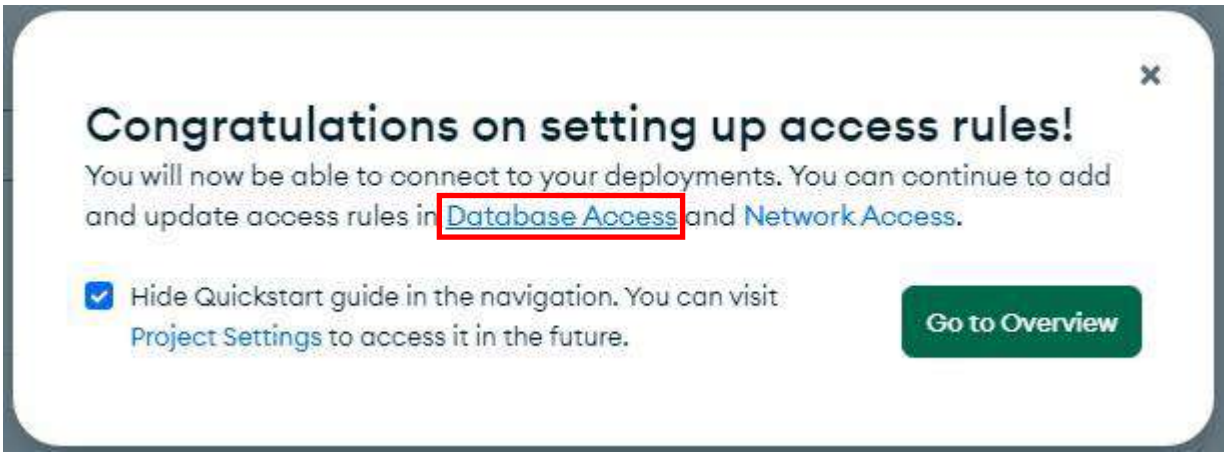
Step 6: Enter the username & password & then click on **'Create User'**.



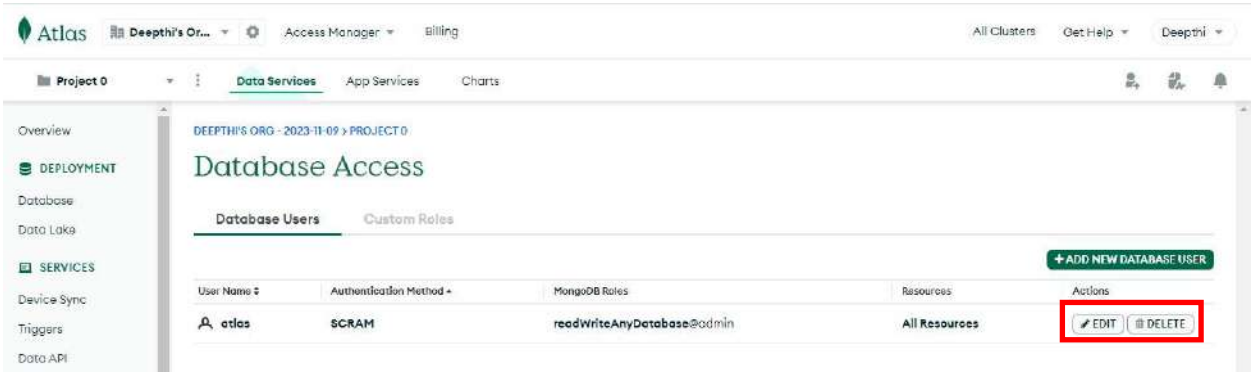
Step 7: Now click on **'Add My Current IP Address'** & click on **'Finish and Close'.**



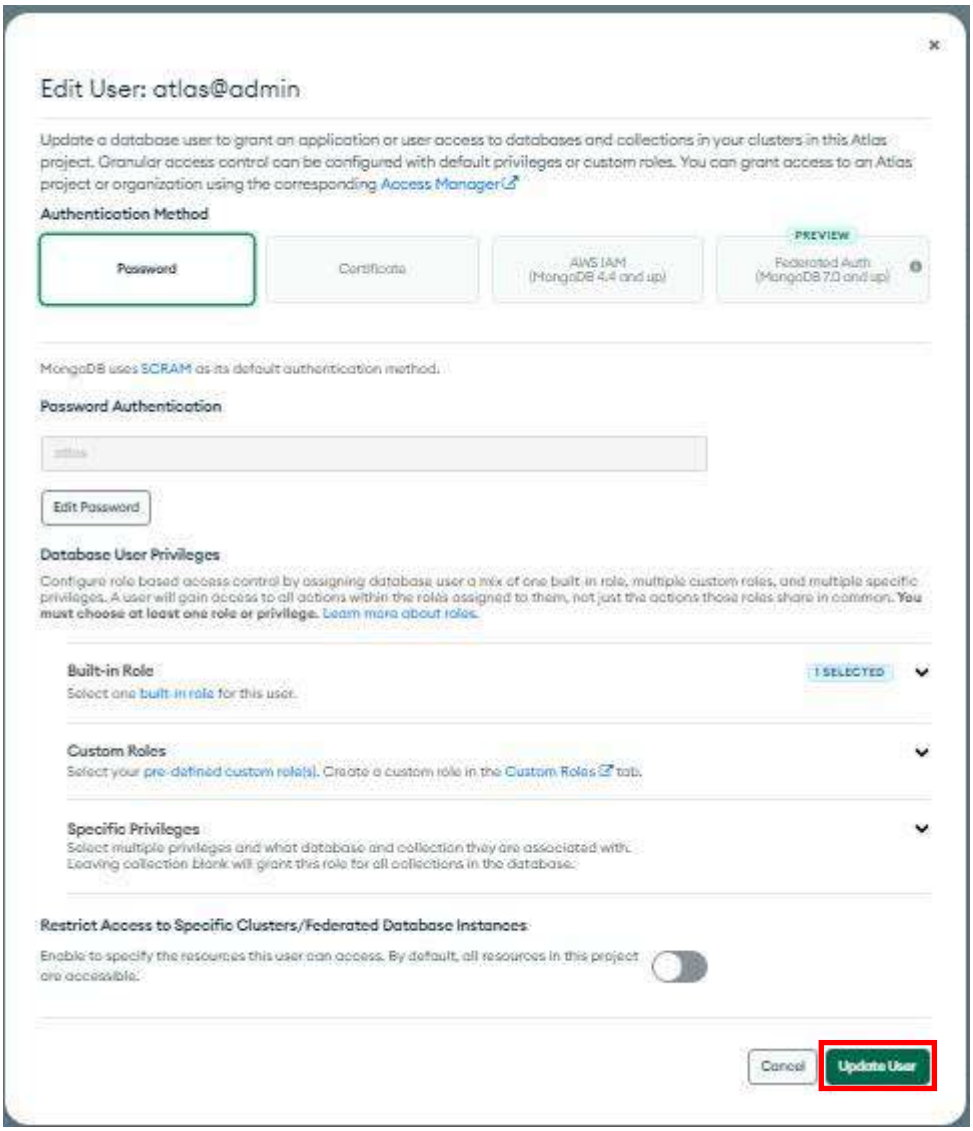
Step 8: Then click on **'Database Access'** here.



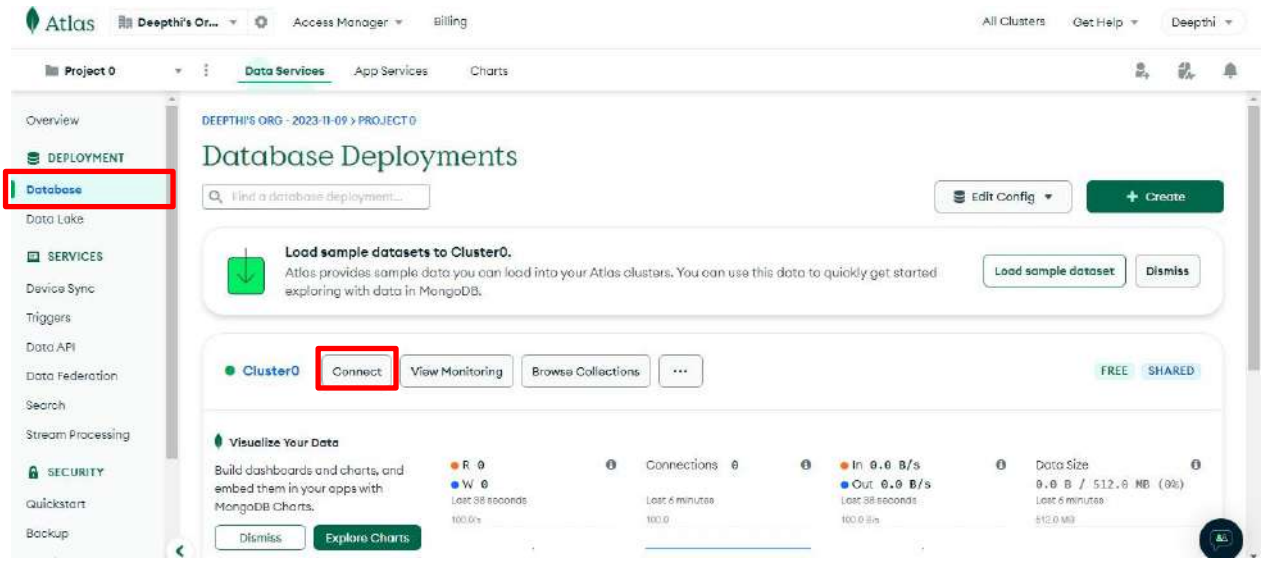
Step 9: Now you can edit or delete your created username and password here.



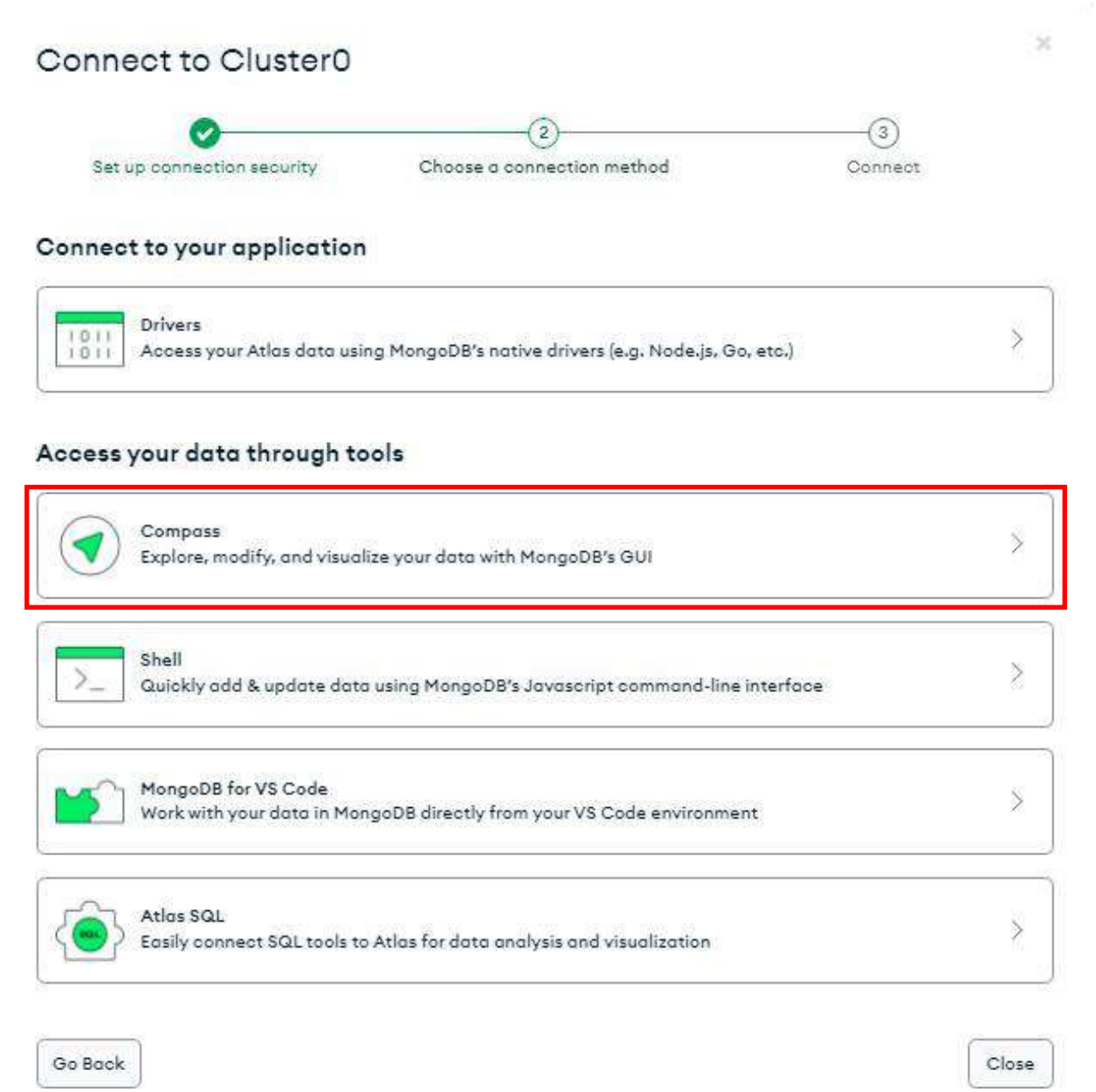
Step 10: After editing your username and password, click on **'Update User'** here.



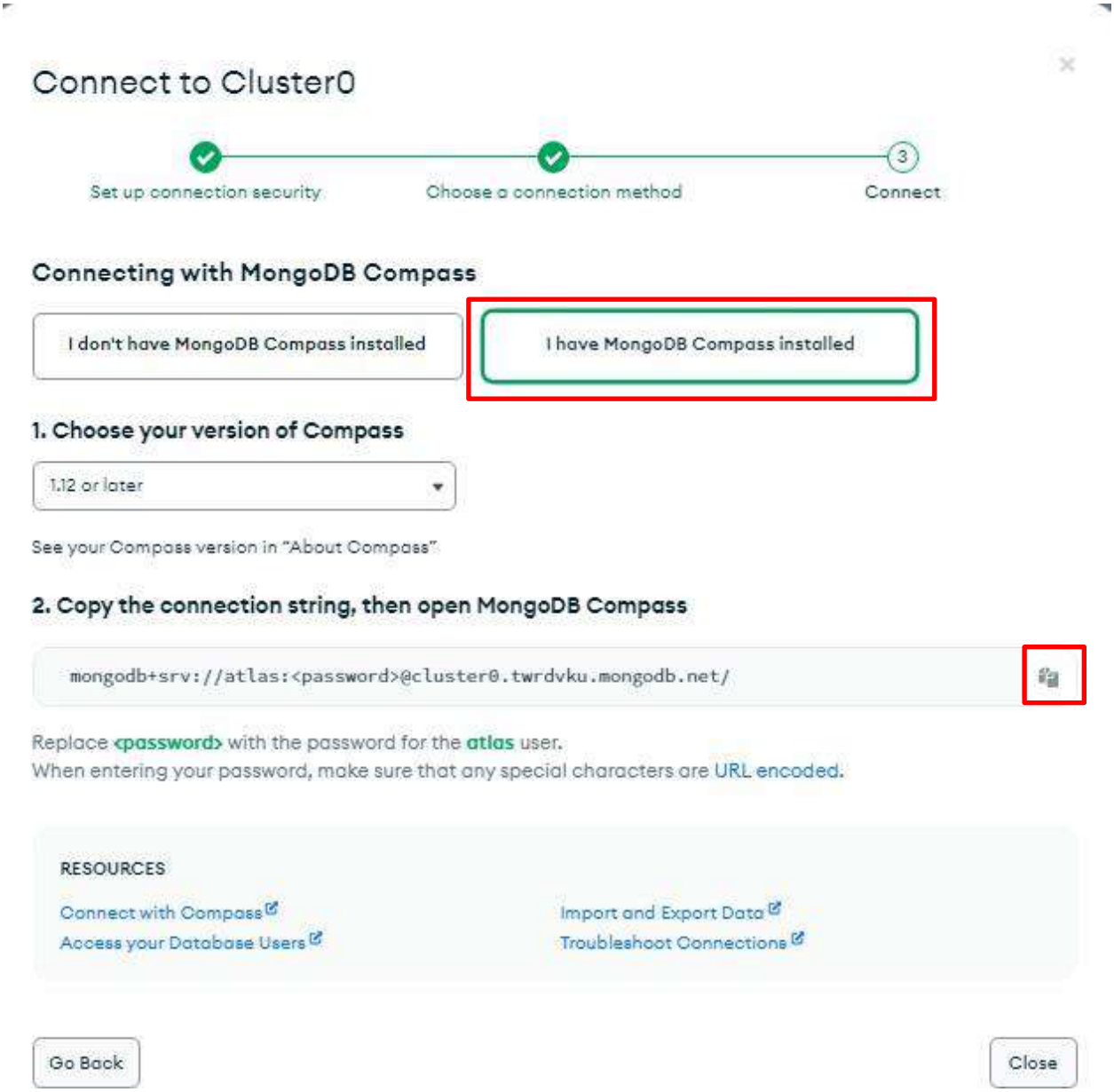
Step 11: Select **'Database'** & click on **'Connect'** option here.



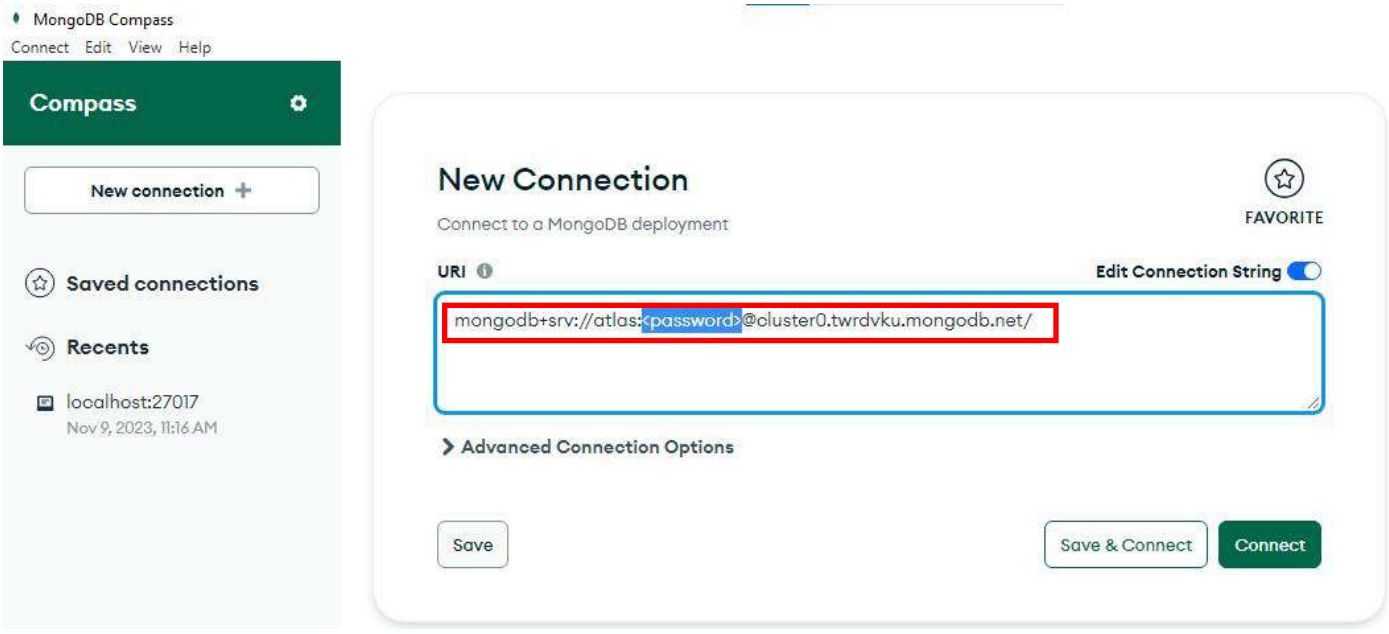
Step 12: Now select **'Compass'** here.



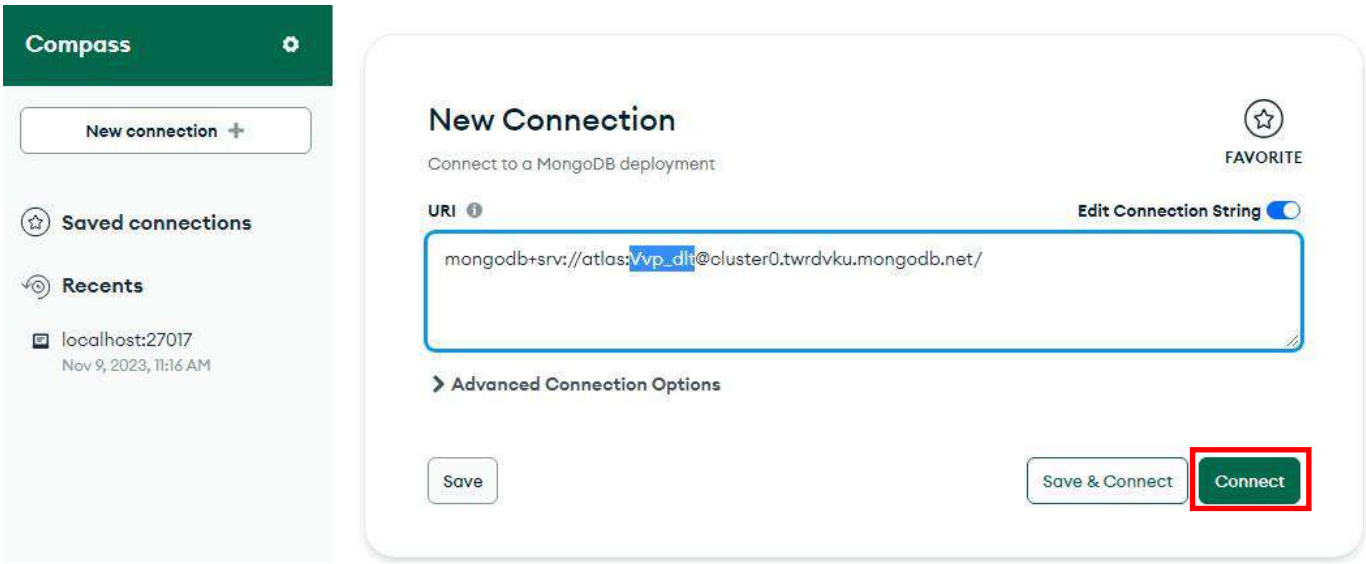
Step 13: Select **'I have MongoDB Compass installed'** option & copy the connection path here.



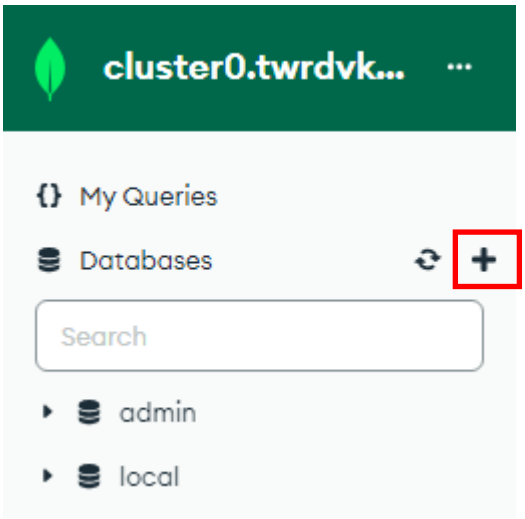
Step 13: Go to MongoDB Compass & paste the path here.



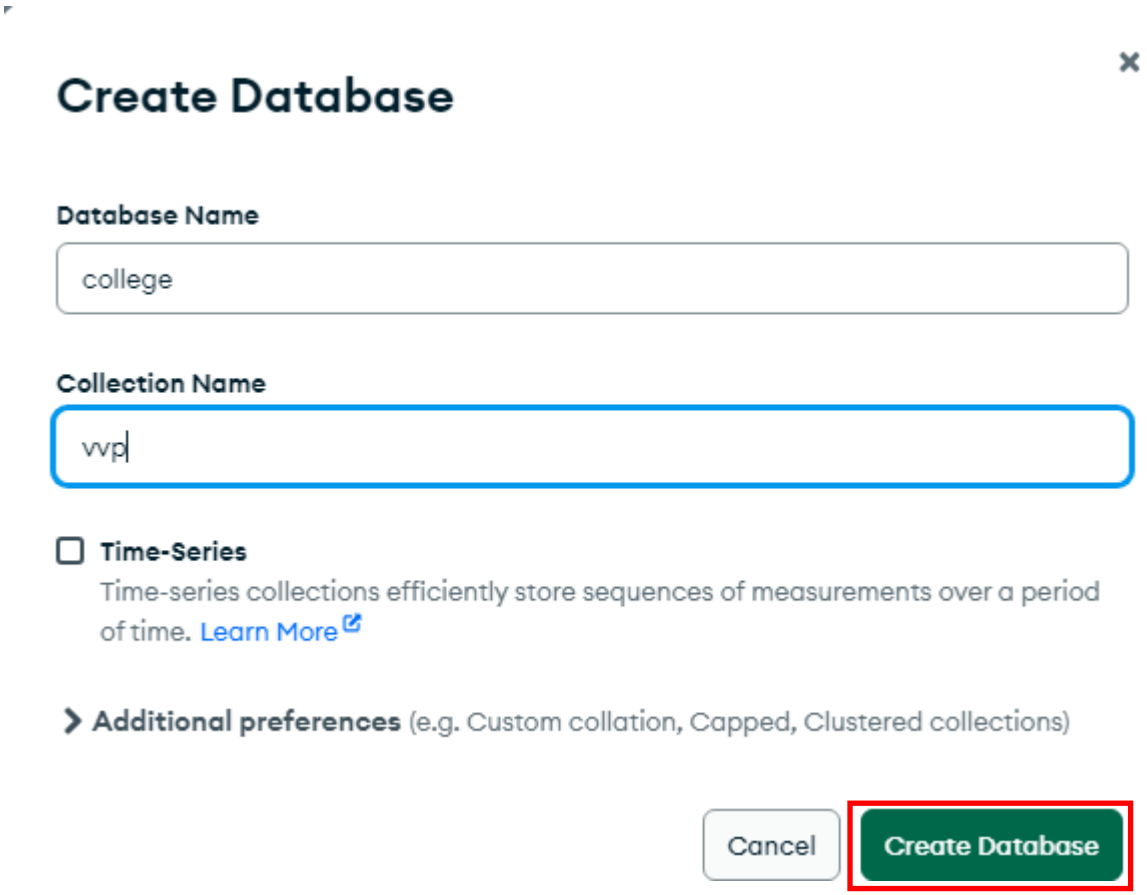
Step 14: Then give the password & click on **'Connect'.**



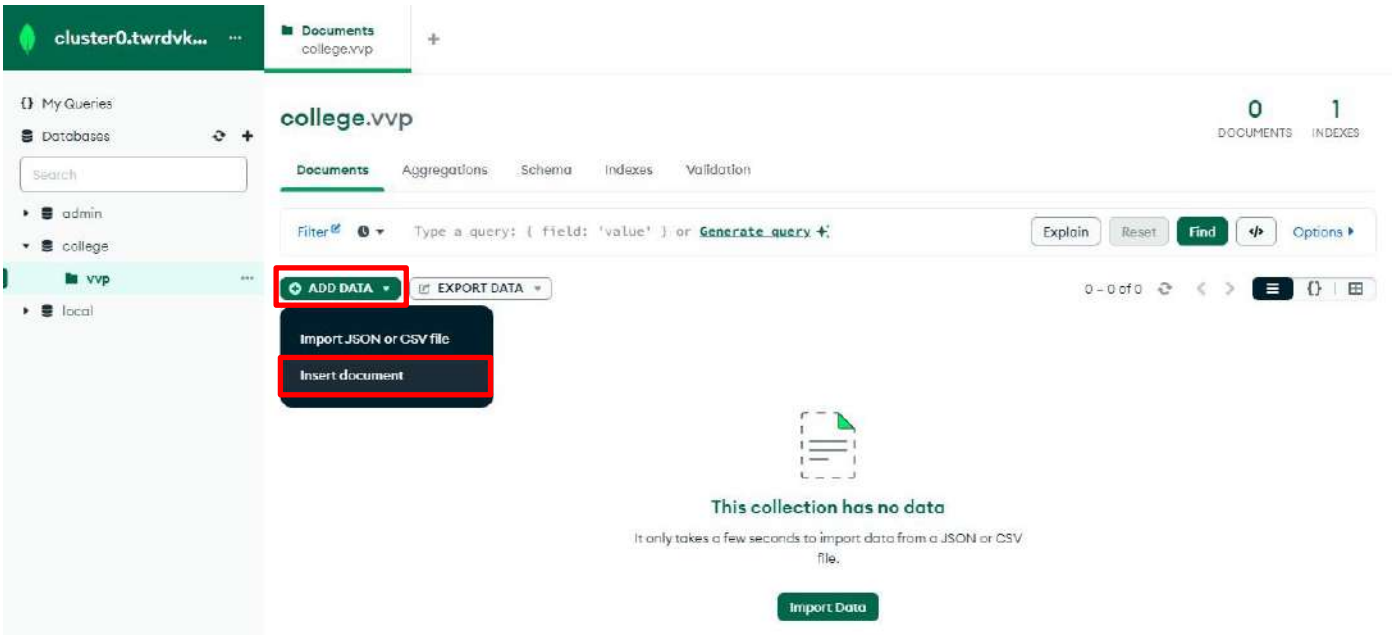
Step 15: Click on '+' icon here.



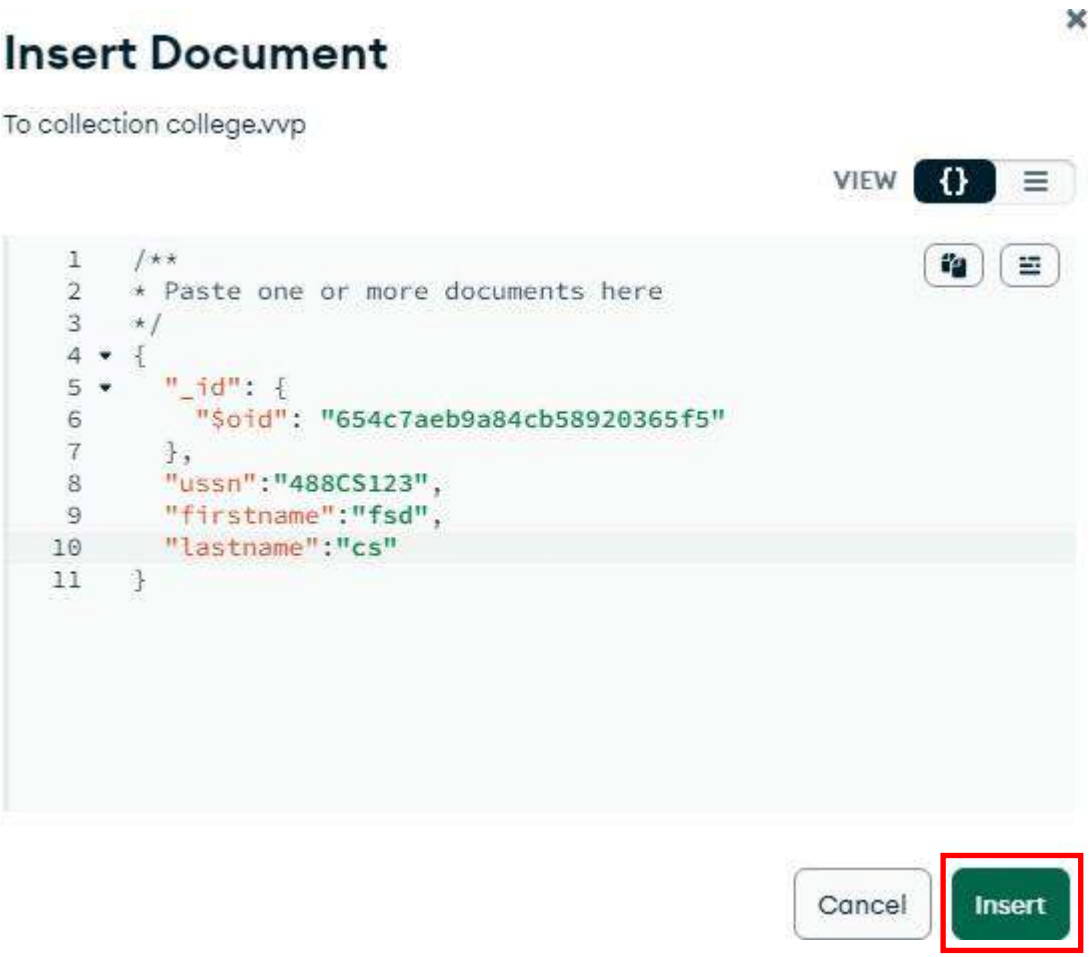
Step 16: Create a new database with the name **'college'** & collection with the name **'vvp'**. And click on **'Create Database'** here.



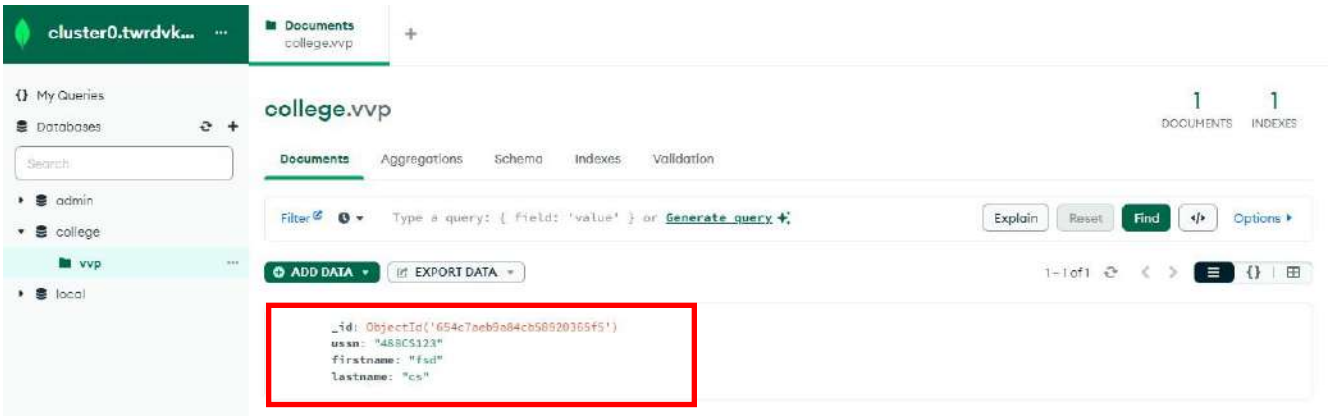
Step 17: Now click on **'ADD DATA'** & select **'insert document'** here.



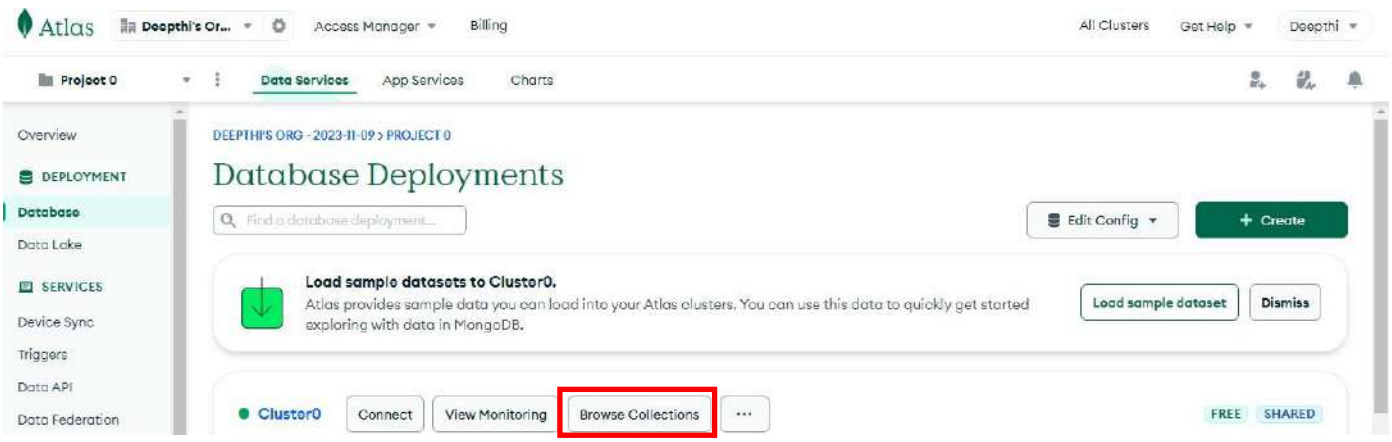
Step 18: Insert dome document & click on **'insert'** option.



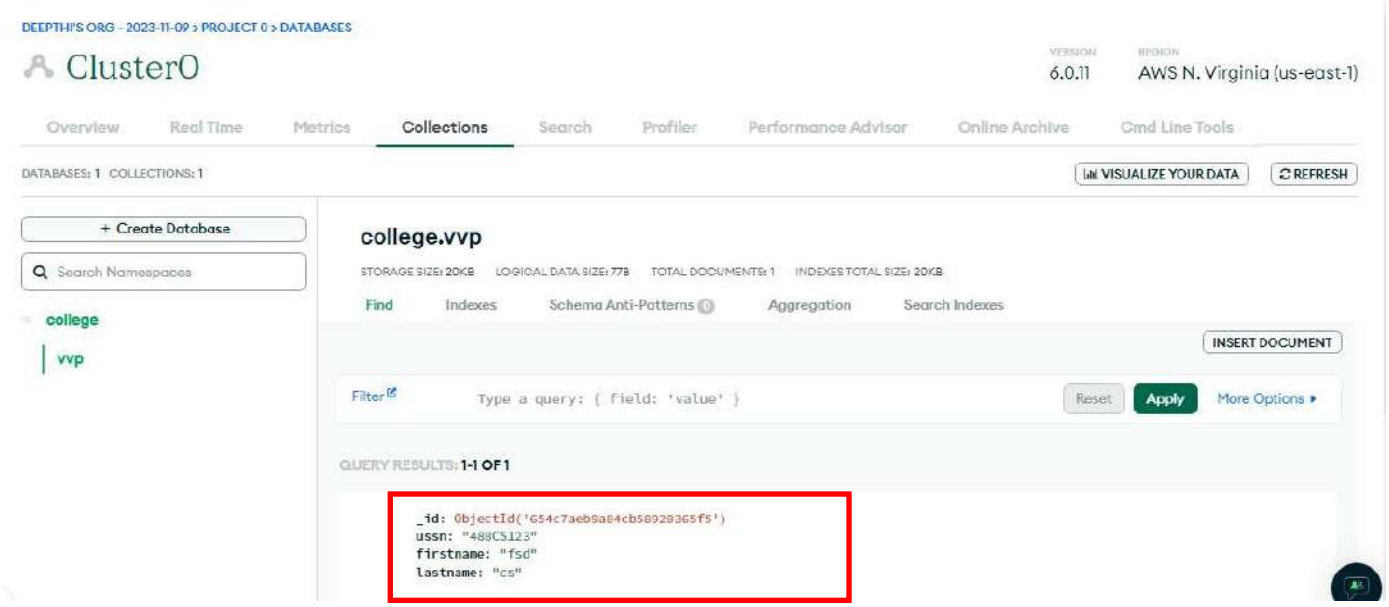
Step 19: Your inserted document is being displayed here.



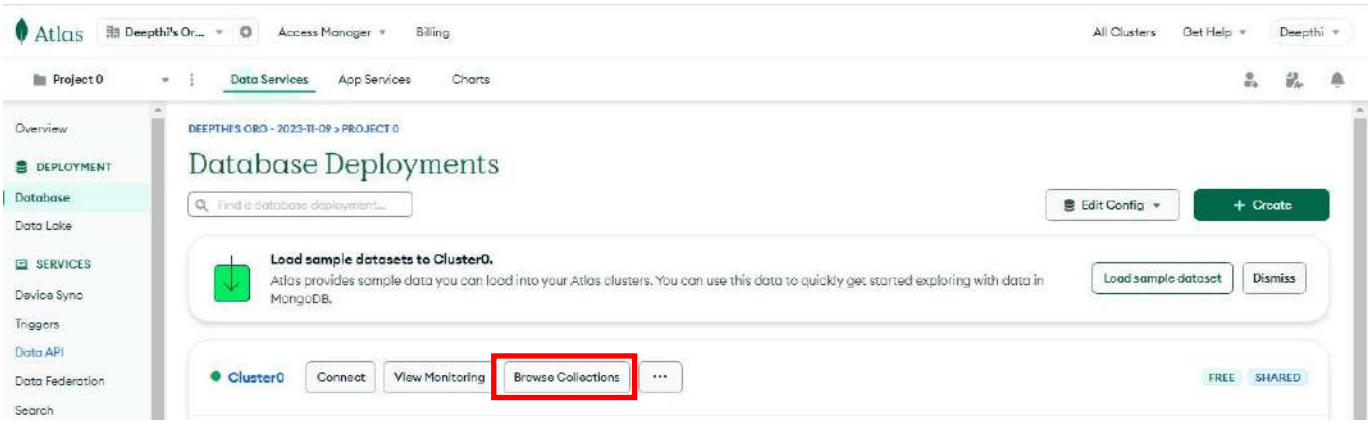
Step 20: Click on **'Browse Collections'** here.



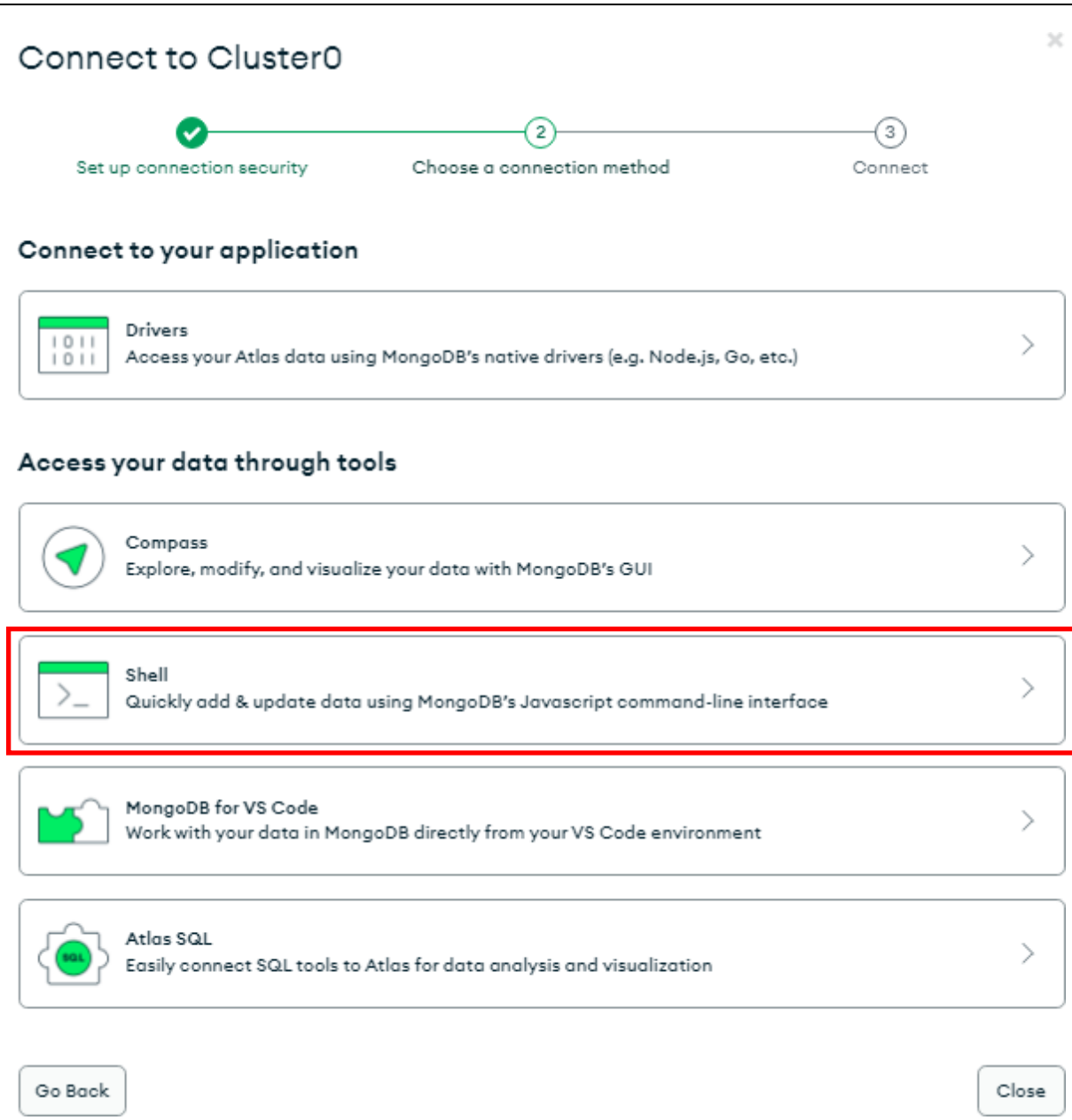
Step 21: Observe the output as shown below.



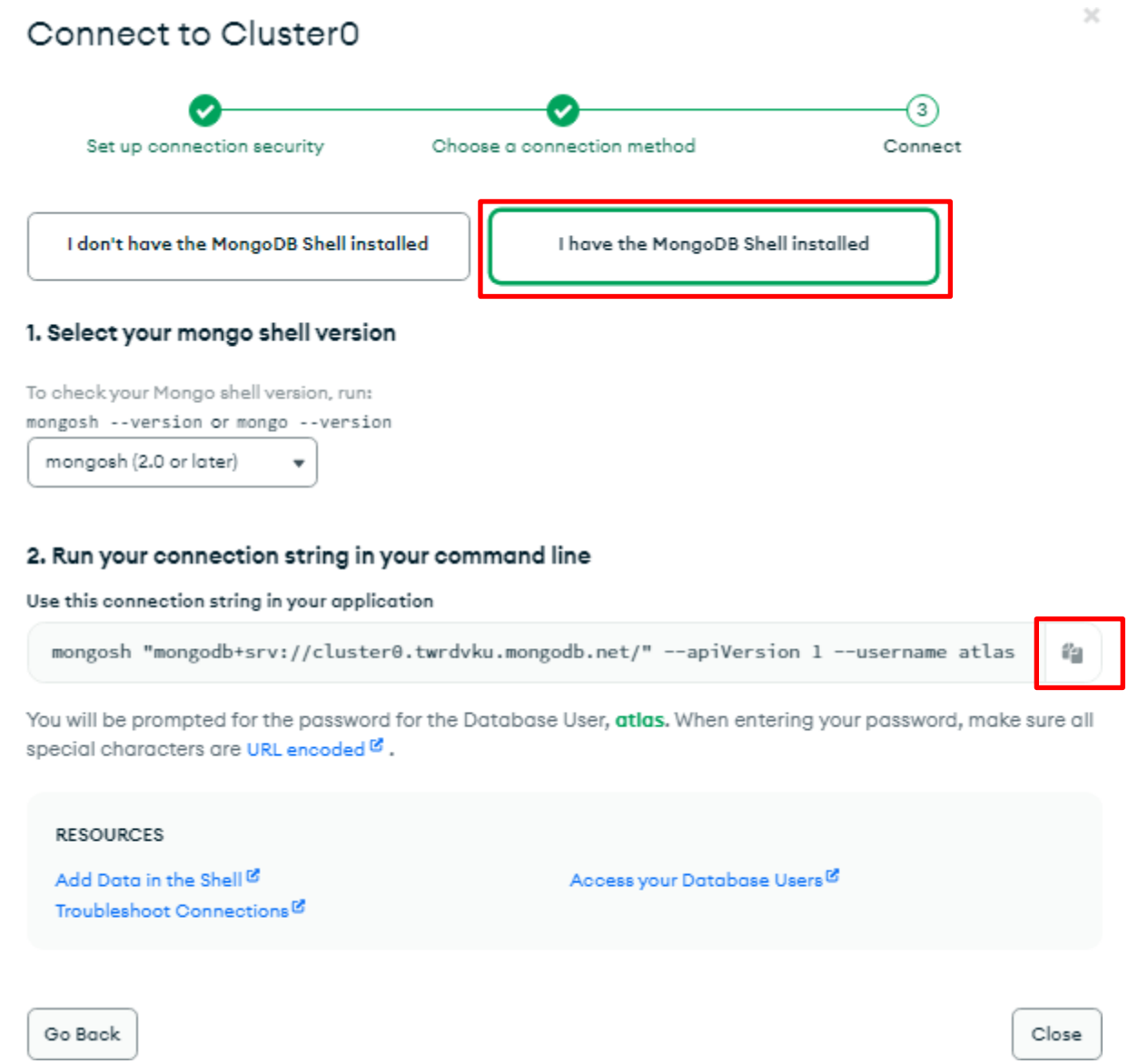
Step 22: Now click on **'connect'** option here.



Step 23: Select **'Shell'** option here.



Step 24: Select **'I have MongoDB Shell installed'** option & copy the connection path here.



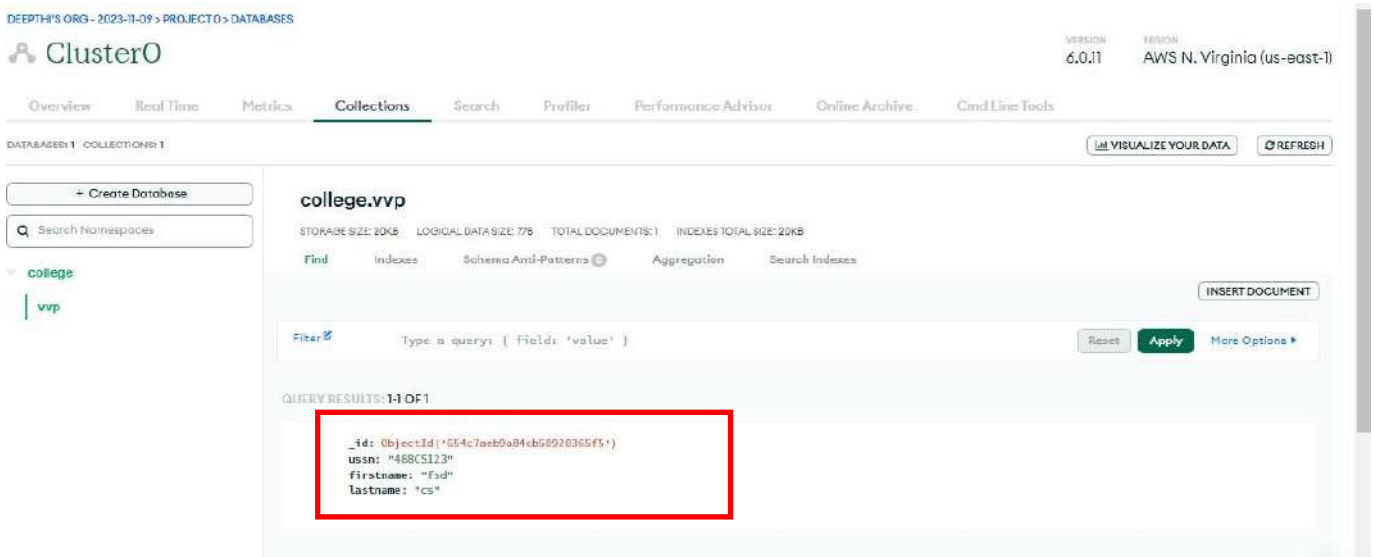
Step 25: Now copy the bin location of MongoDB Shell downloaded path as shown here & press **'cmd'** .

Step 26: Now paste the path which was copied in Step 25 as **cd …….path…….** & press enter key. And also paste the path which was copied in Step 24 & press enter key. Then enter the password here.

Step 27: Use the existing database as shown below.

```
Atlas atlas-n83st3-shard-0 [primary] test> show dbs
college   40.00 KiB
admin    280.00 KiB
local     40.51 GiB
Atlas atlas-n83st3-shard-0 [primary] test> use college
switched to db college
Atlas atlas-n83st3-shard-0 [primary] college> show collections
vvp
Atlas atlas-n83st3-shard-0 [primary] college> db.vvp.find()
[
  {
    _id: ObjectId("654c7aeb9a84cb58920365f5"),
    ussn: '488CS123',
    firstname: 'fsd',
    lastname: 'cs'
  }
]
Atlas atlas-n83st3-shard-0 [primary] college>
```

Step 28: Now go to **'Browse collections'** to view the below output.

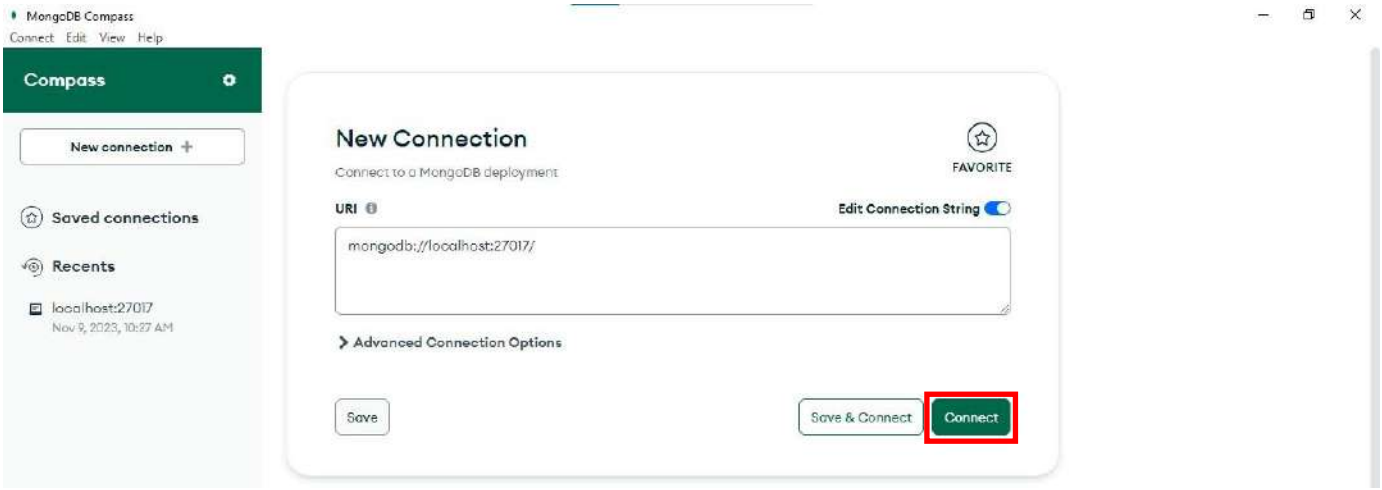


5. <u>Aggregation concept in MongoDB.</u>

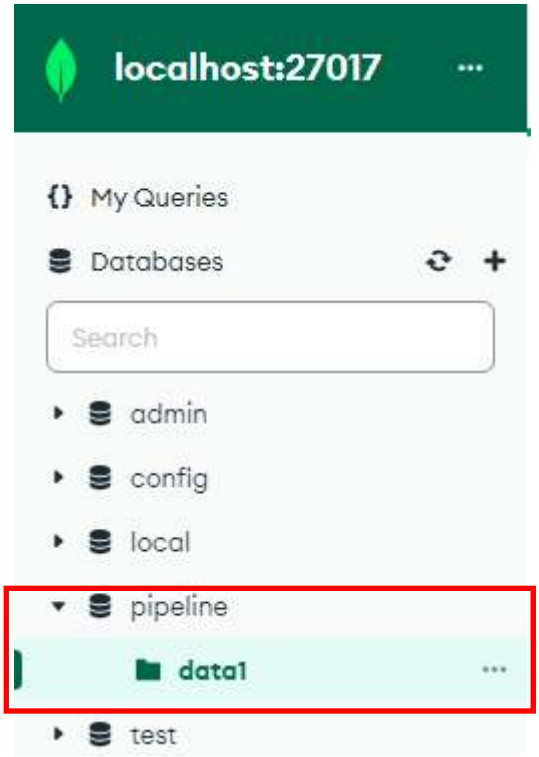
Step 1: Go to Mongosh and create a new database and collection as shown below.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Please enter a MongoDB connection string (Default: mongodb://localhost/):

Current Mongosh Log ID: 654c71b14ba60b6bc808dfd3
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.0.2
Using MongoDB:          6.0.11
Using Mongosh:          2.0.2

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

------
   The server generated these startup warnings when booting
   2023-11-08T14:00:07.743+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use pipeline
switched to db pipeline
pipeline> db.createCollection("data1")
{ ok: 1 }
pipeline> show collections
data1
pipeline> show dbs
BookStore   48.00 KiB
CS          72.00 KiB
admin       40.00 KiB
config     116.00 KiB
fsd         72.00 KiB
local       72.00 KiB
pipeline     8.00 KiB
test        40.00 KiB
pipeline>
```

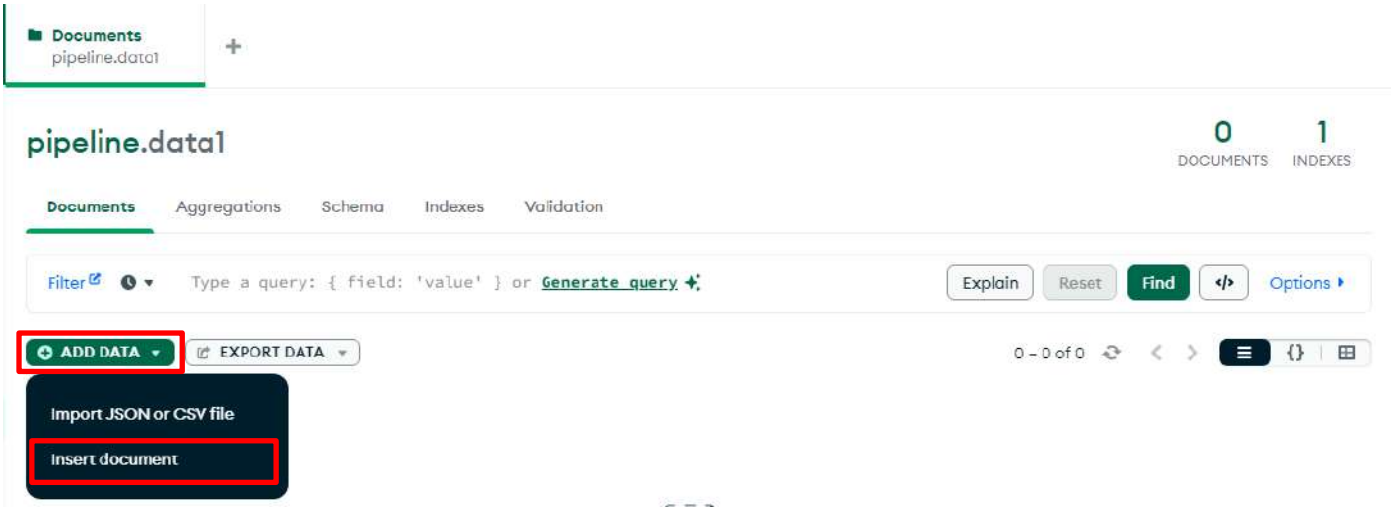
Step 2: Now go to MongoDB Compass & click on **'Connect'** option here.



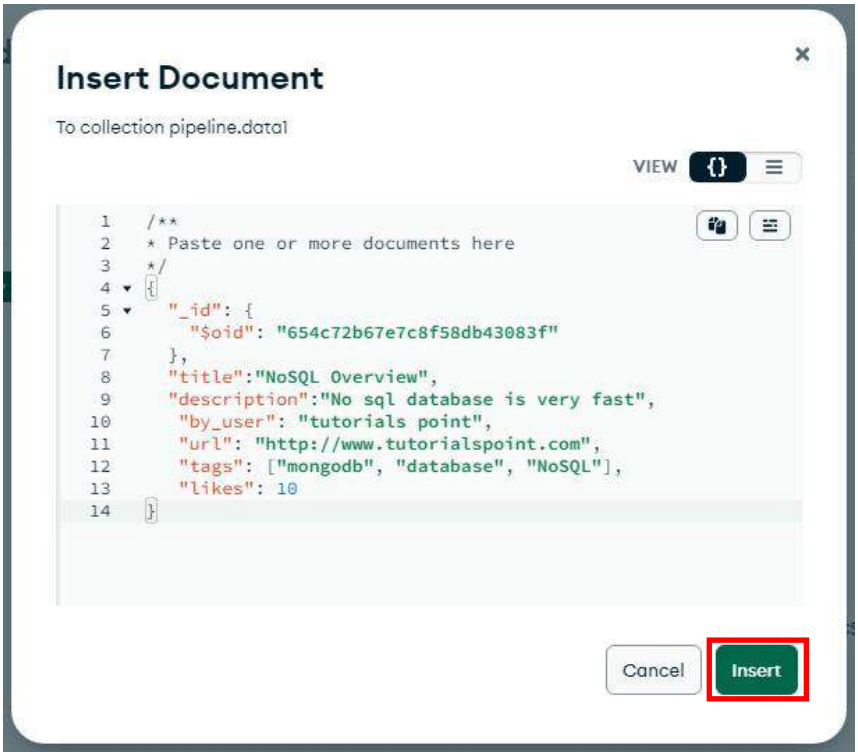
Step 3: Select the created database and collection here.



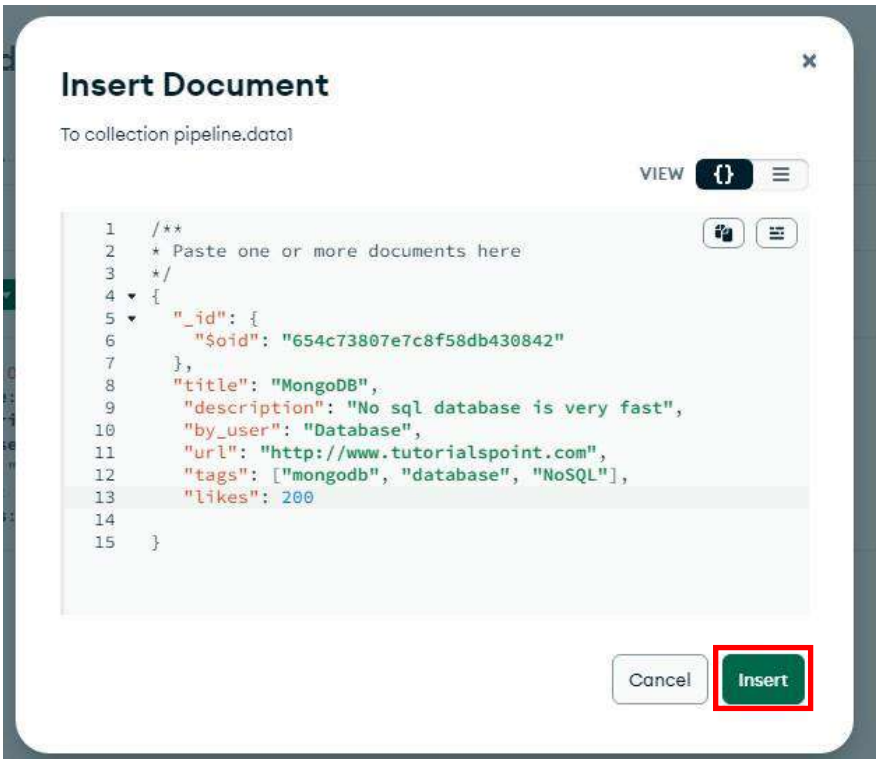
Step 4: Now click on **'ADD DATA'** & select **'insert document'** option here.



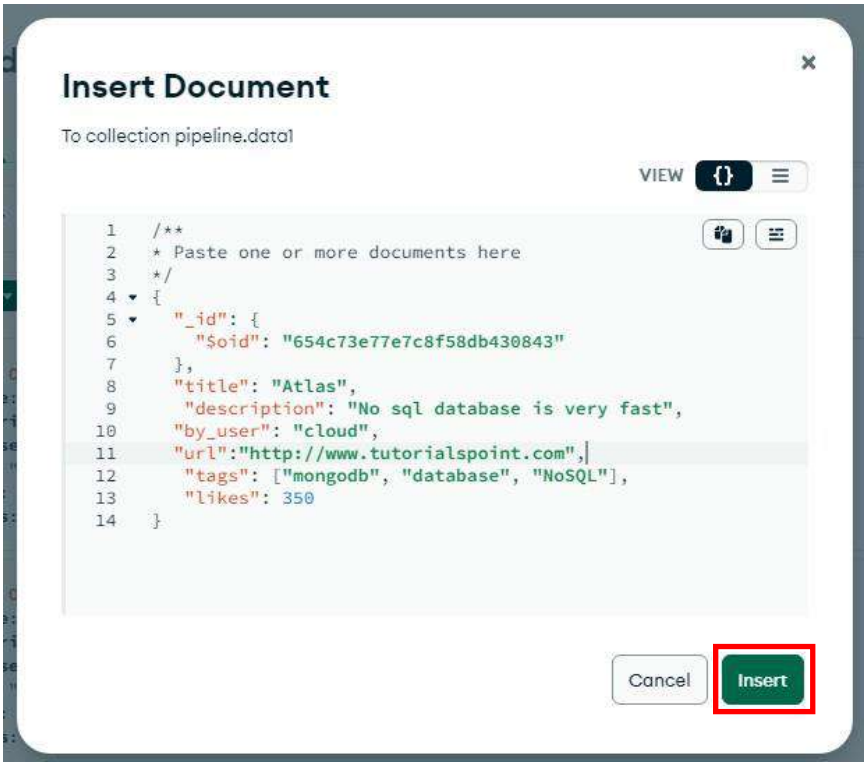
Step 5: Now insert first document & click on **'Insert'** here.



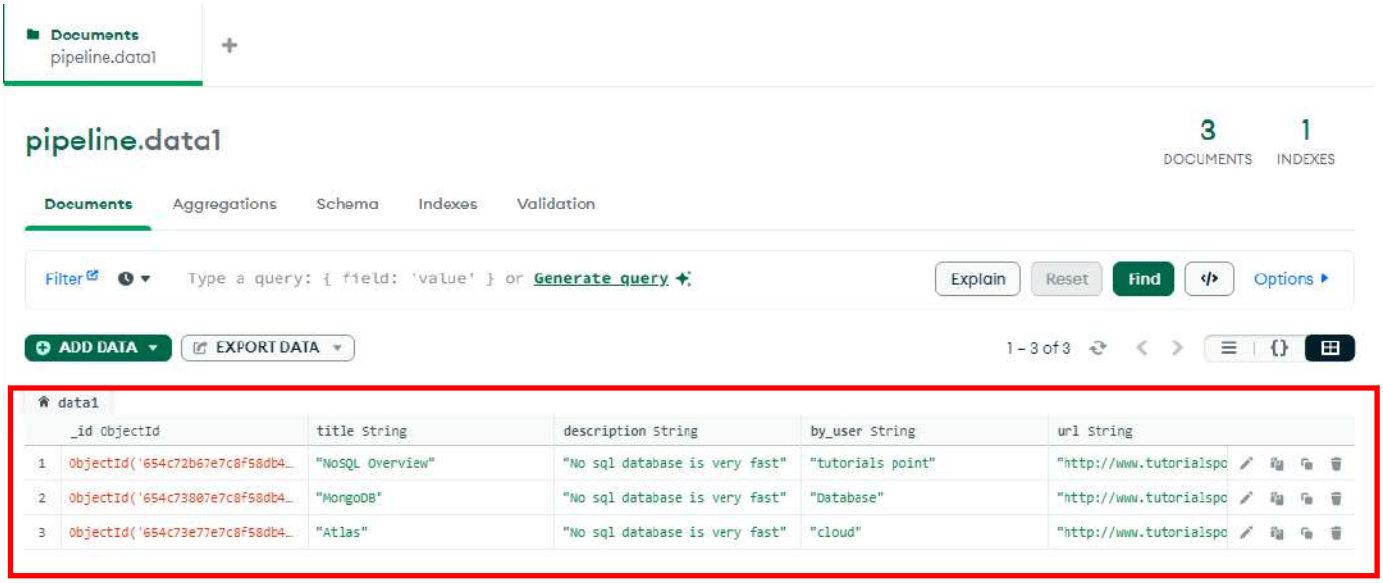
Step 6: Now insert second document & click on **'Insert'** here.



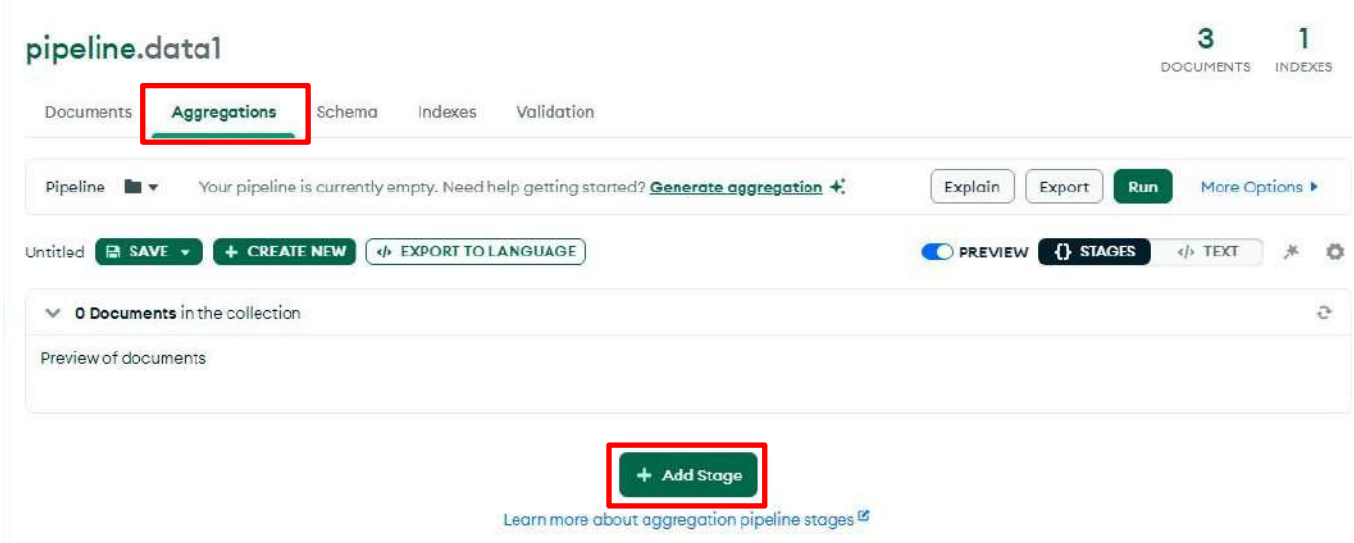
Step 7: Now insert third document & click on **'Insert'** here.



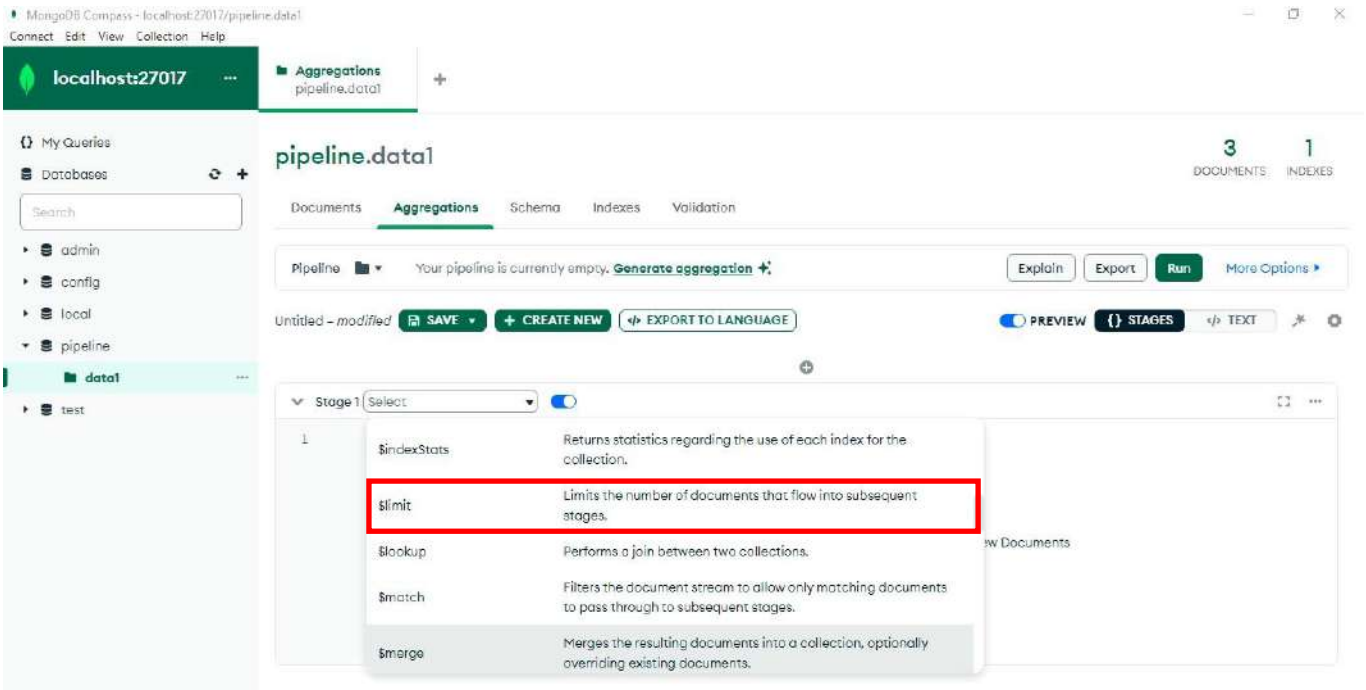
Step 8: Now you can view the inserted documents in the table format as shown below.



Step 9: Go to **'Aggregation'** tab & click on **'Add Stage'** here.

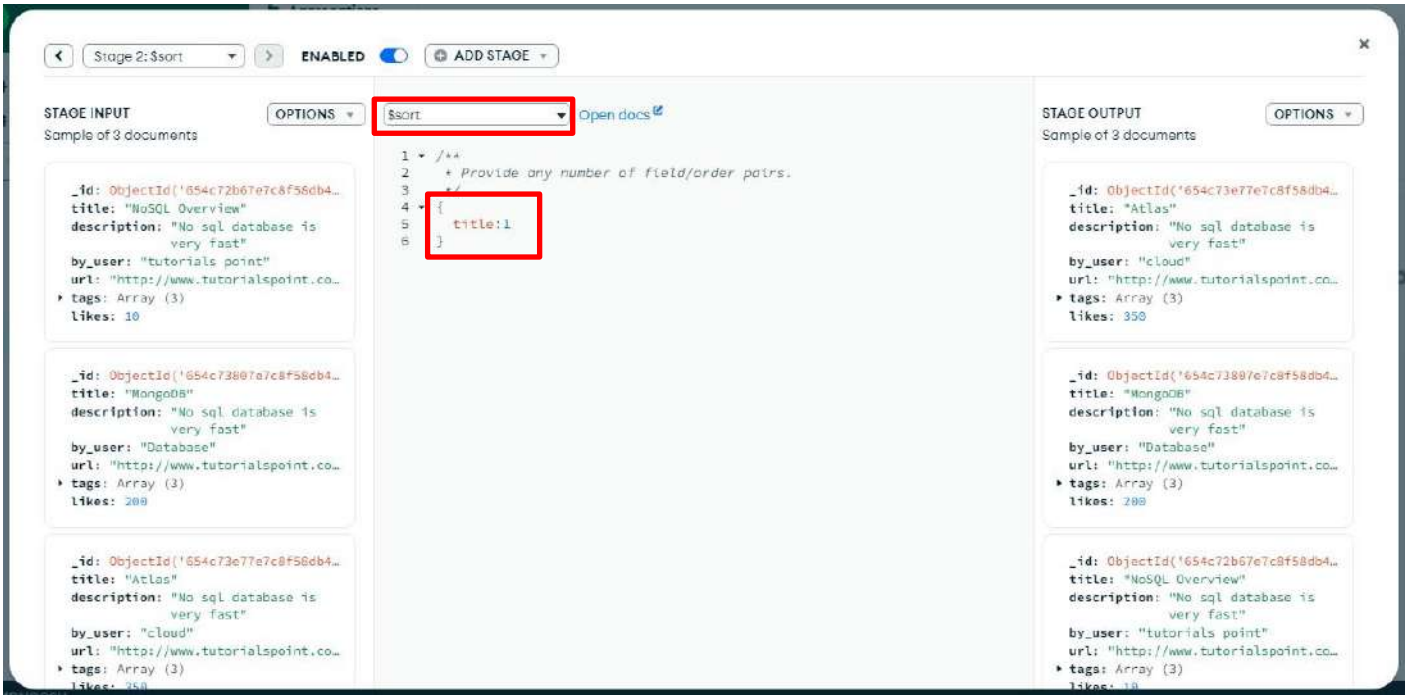


Step 10: Now select **'limit'** aggregate operation here.

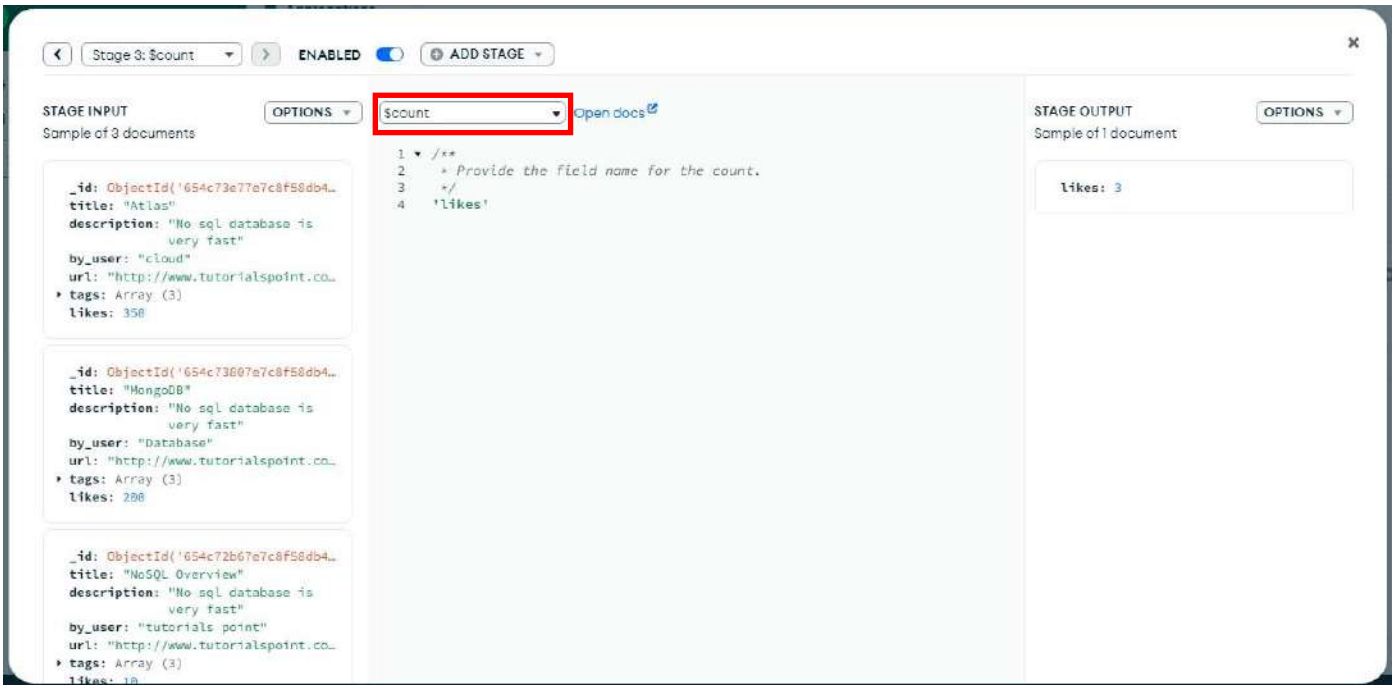


Step 11: Now give a number below. The given result will be the input for next stages.

Step 12: Select **'Sort'** aggregate operation & enter the field for sorting here.

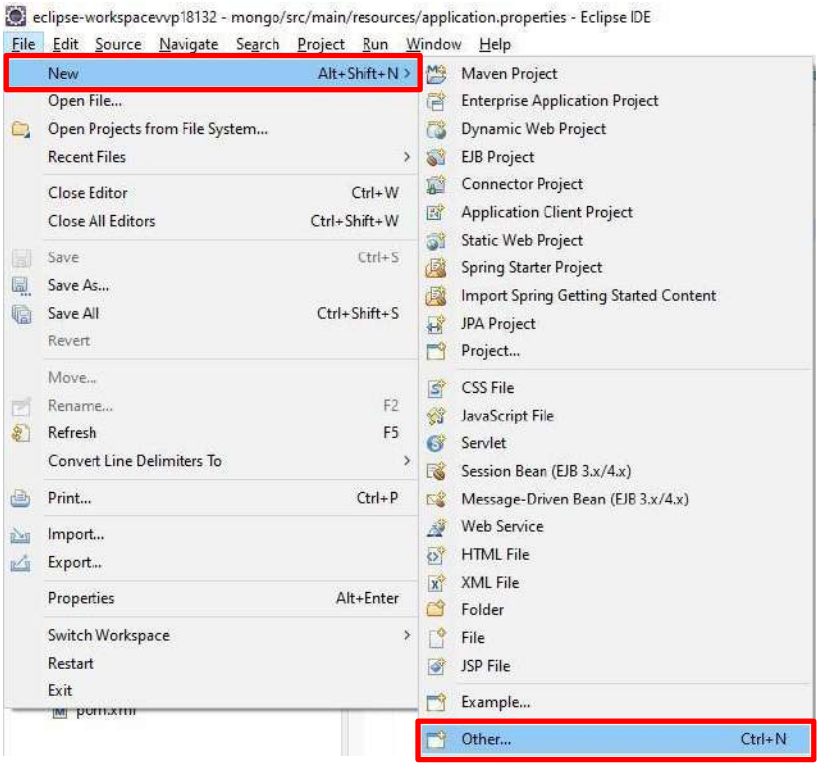


Step 13: Select **'Count'** aggregate operation & enter the field for counting here.

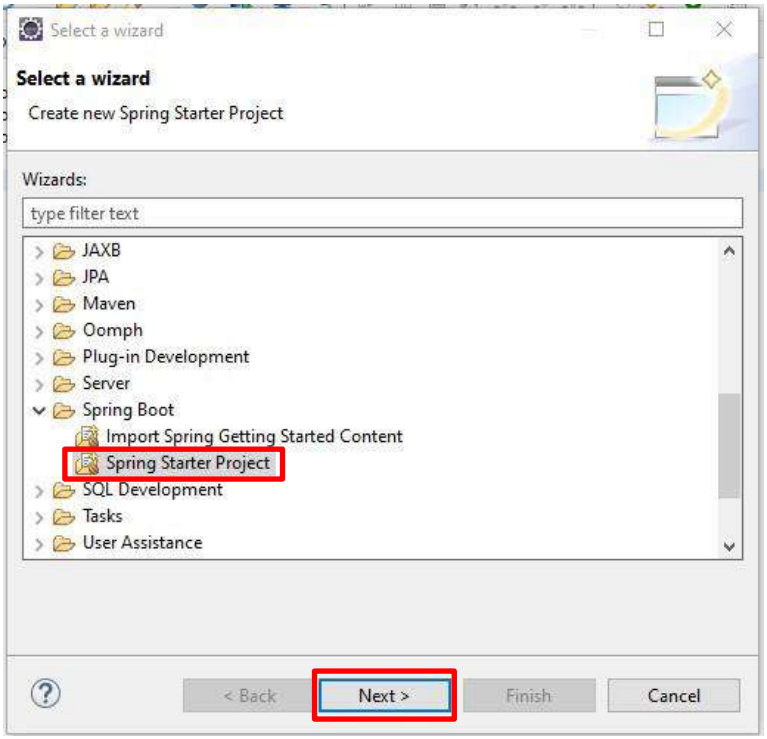


6. Perform CRUD Operations on MongoDB through REST API using Spring Boot Starter Data MongoDB.

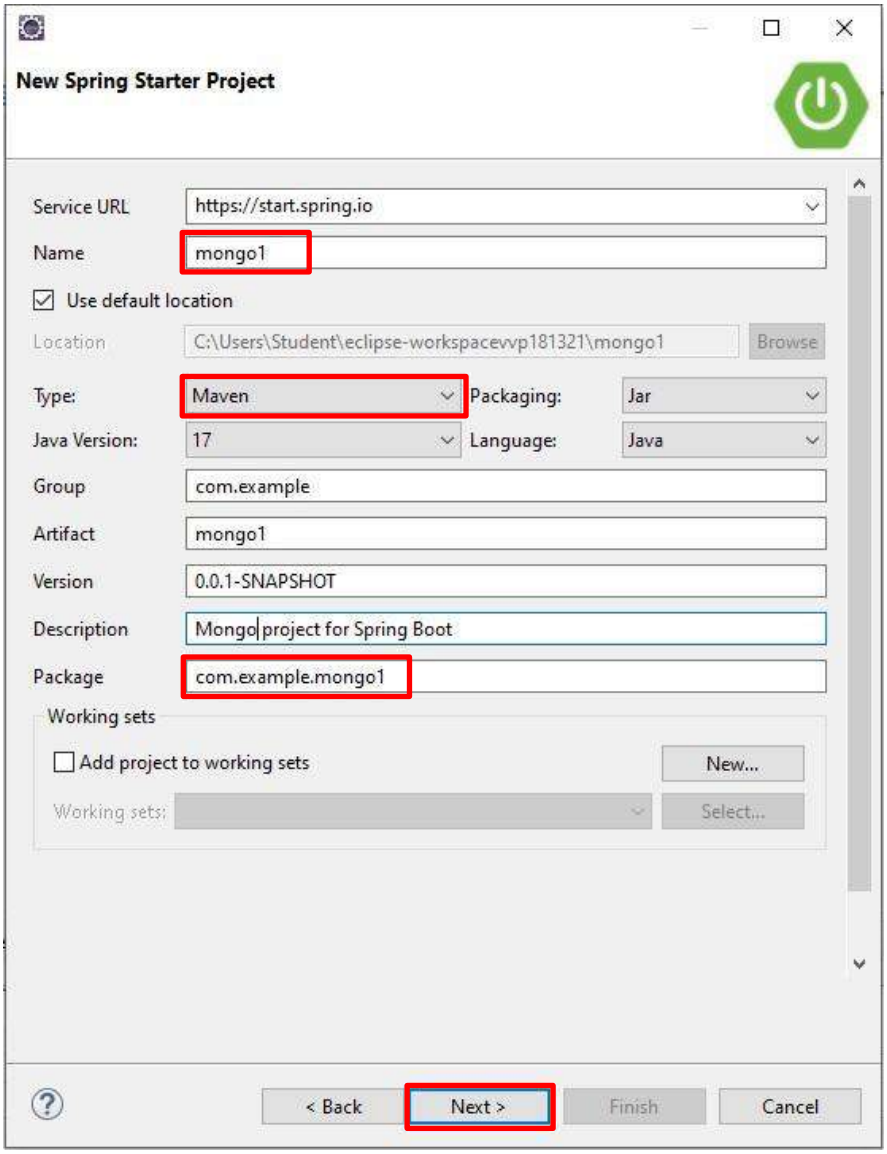
Step 1: Open the Eclipse IDE for Enterprise Edition New Other.



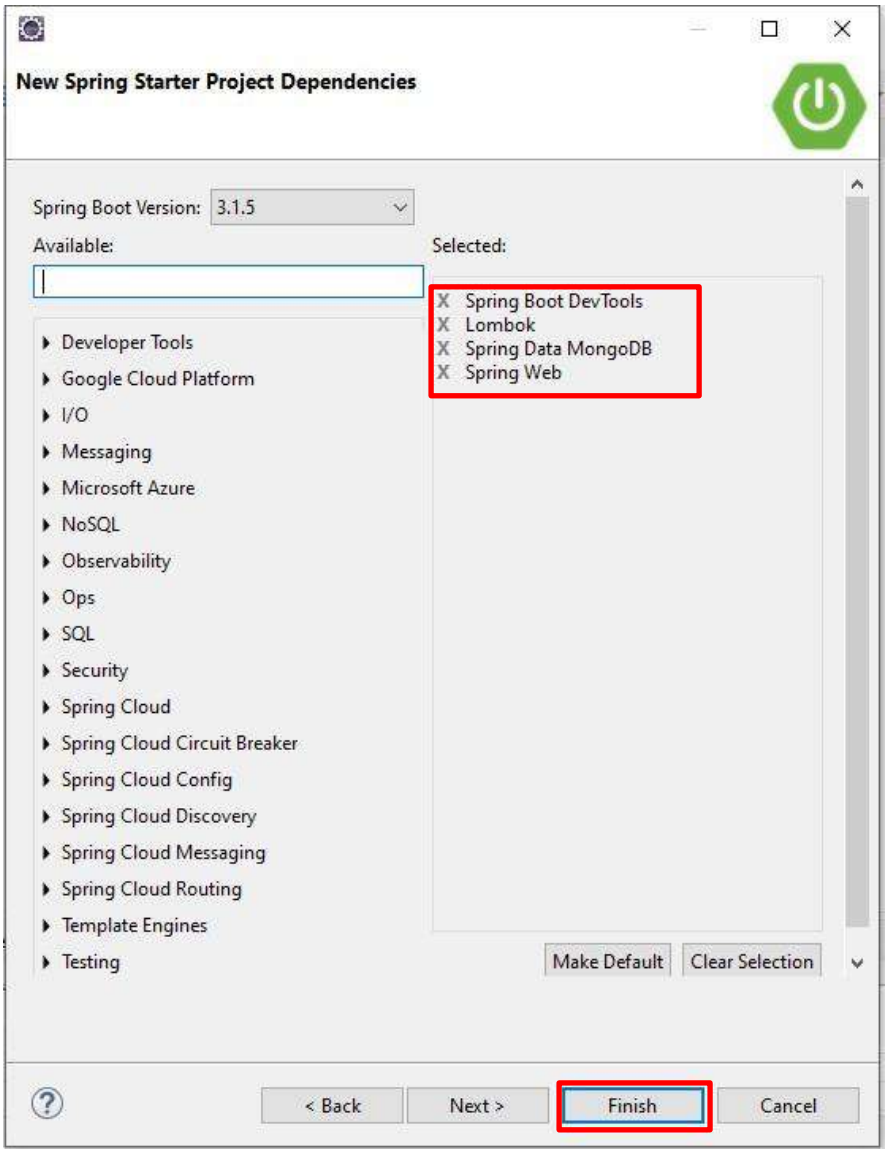
Step 2: Select **'Spring starter project'** & click on **'Next'**.



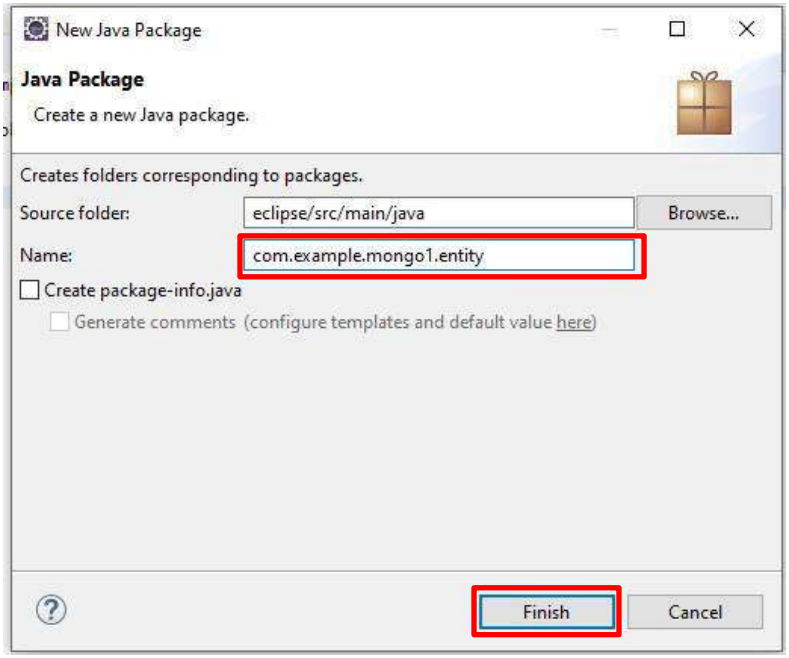
Step 3: Select the options as shown below & click on **'Next'**.



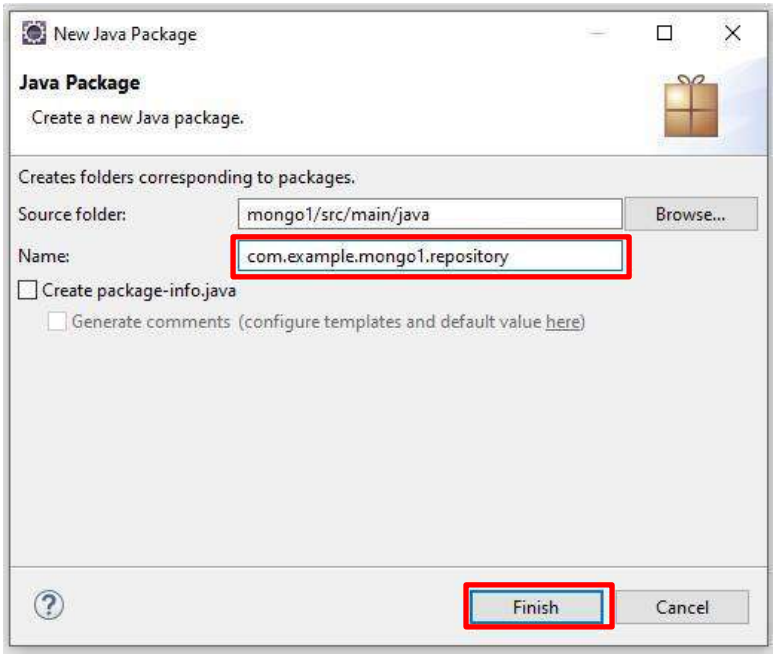
Step 4: Add 4 below mentioned dependencies & click on **'Finish'**.



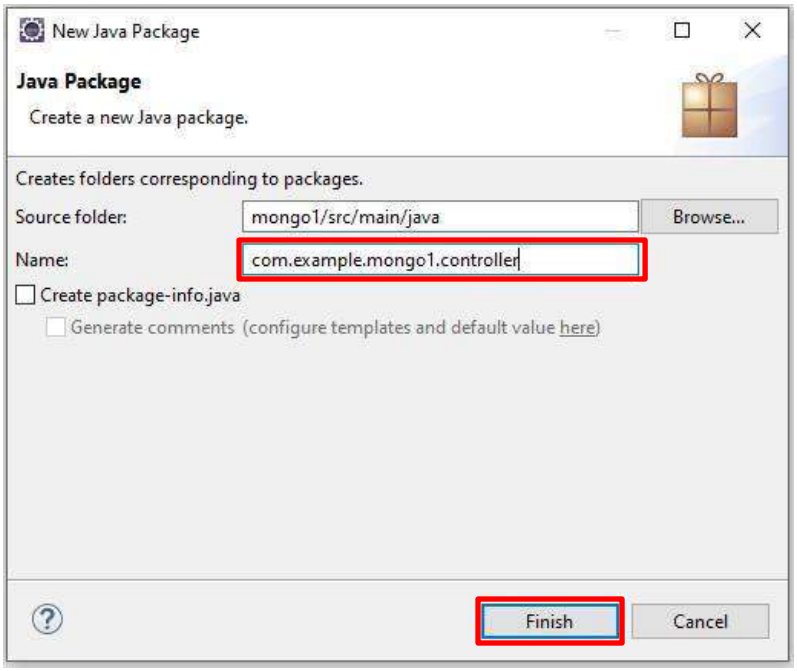
Step 5: Now create a new package named **'com.example.mongo1.entity'** & click on **'Finish'**.



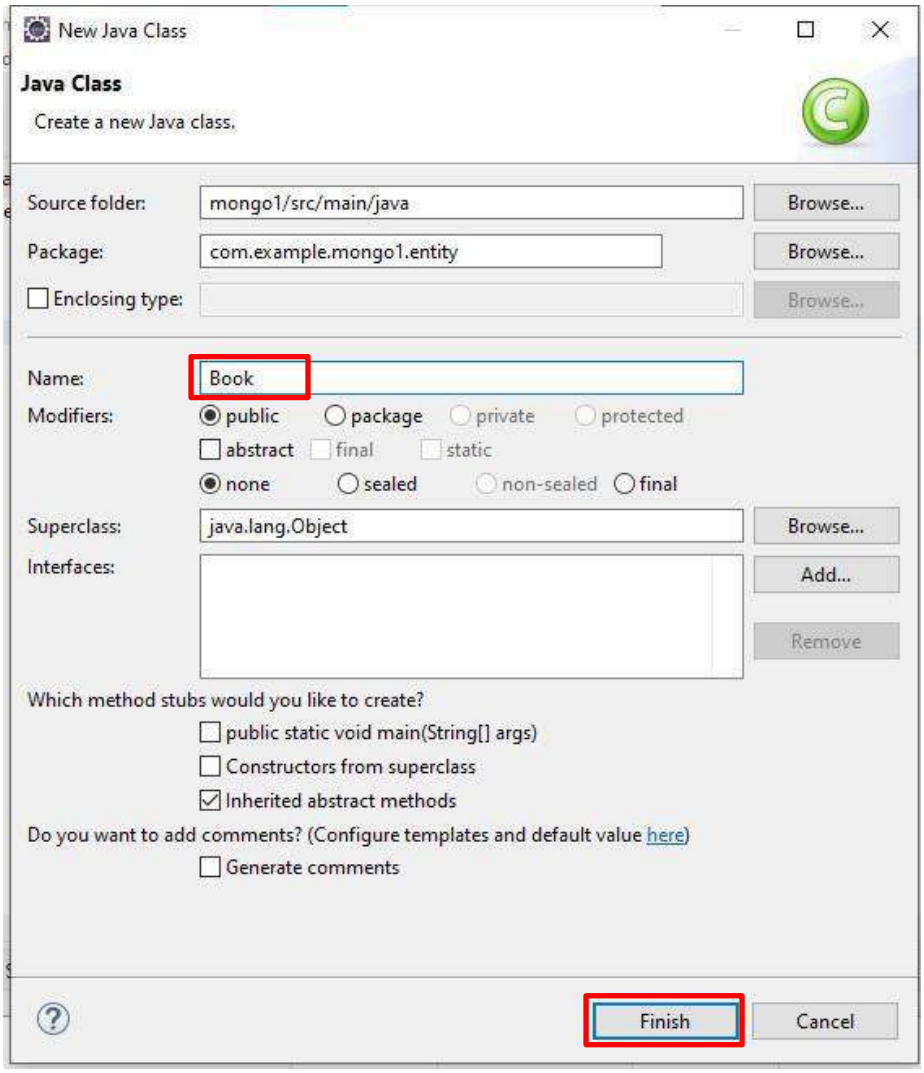
Step 6: Now create a new package named **'com.example.mongo1.repository'** & click on **'Finish'**.



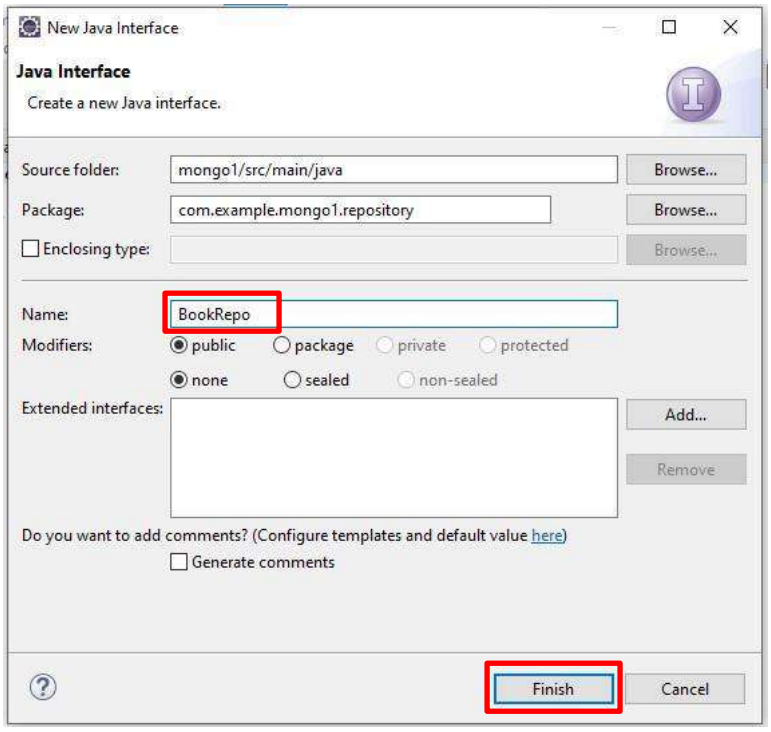
Step 7: Now create a new package named **'com.example.mongo1.controller'** & click on **'Finish'**.



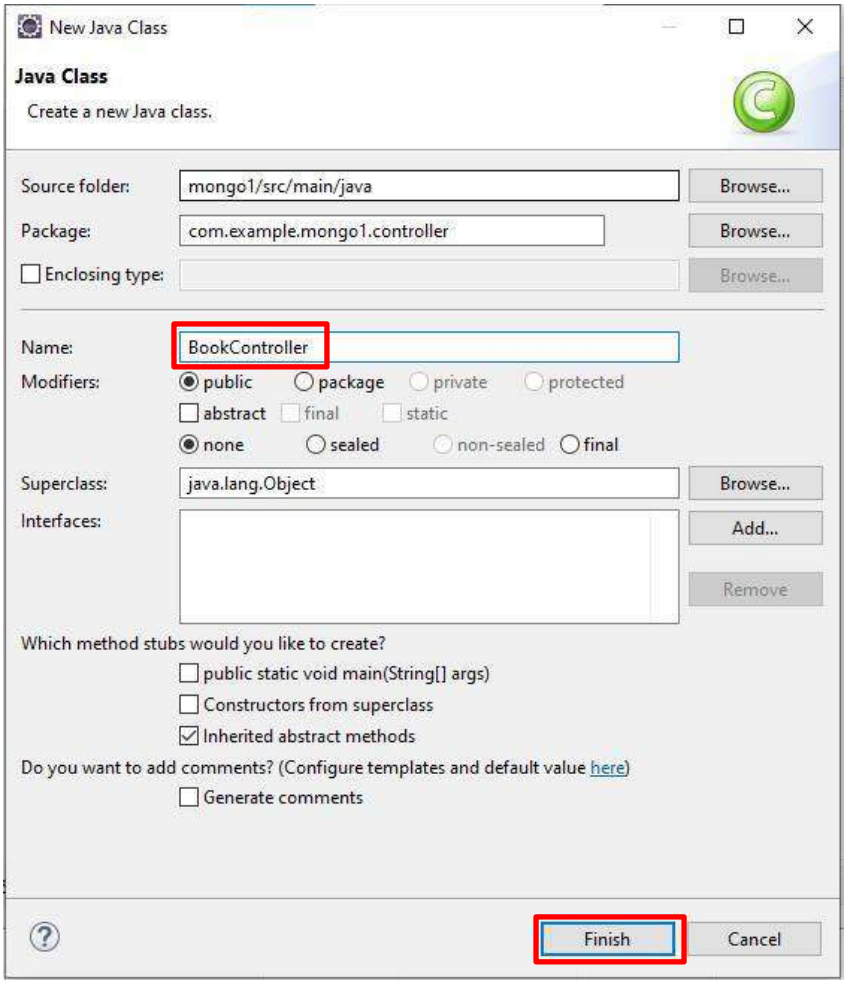
Step 8: Now create a new class named **'Book'** for the package named **'com.example.mongo1.entity'** & click on **'Finish'**.



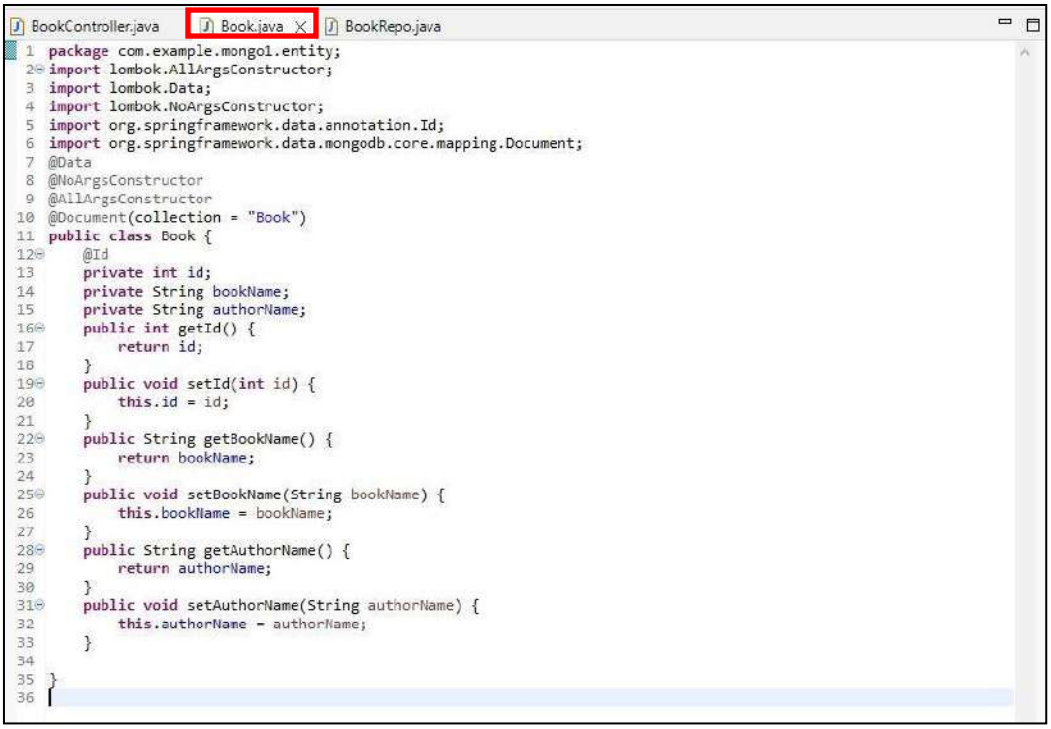
Step 9: Now create a new interface named **'BookRepo'** for the package named **'com.example.mongo1.repository'** & click on **'Finish'**.



Step 10: Now create a new class named **'BookController'** for the package named **'com.example.mongo1.controller'** & click on **'Finish'**.



Step 11: Now add code to **'Book.java'** file.



Step 12: Now add code to **'BookController.java'** file.

*BookController.java | Book.java | BookRepo.java

```java
package com.example.mongo1.controller;

import java.util.List;

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

import com.example.mongo1.entity.Book;
import com.example.mongo1.repository.BookRepo;

@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();
    }
@PutMapping("/update/{id}")
public String updatebook(@RequestBody Book book,@PathVariable("id") Integer bookId)
{
    Book ex=this.repo.findById(bookId).orElseThrow();
    ex.setBookName(book.getBookName());
    ex.setAuthorName(book.getAuthorName());
    this.repo.save(ex);
    return "Updated Successfully";
}
@DeleteMapping("/delete/{id}")
    public String deleteBook(@PathVariable int id){
        repo.deleteById(id);
        return "Deleted Successfully";
    }
}
```

Step 13: Now add code to **'BookRepo.java'** file.

*BookController.java | Book.java | BookRepo.java

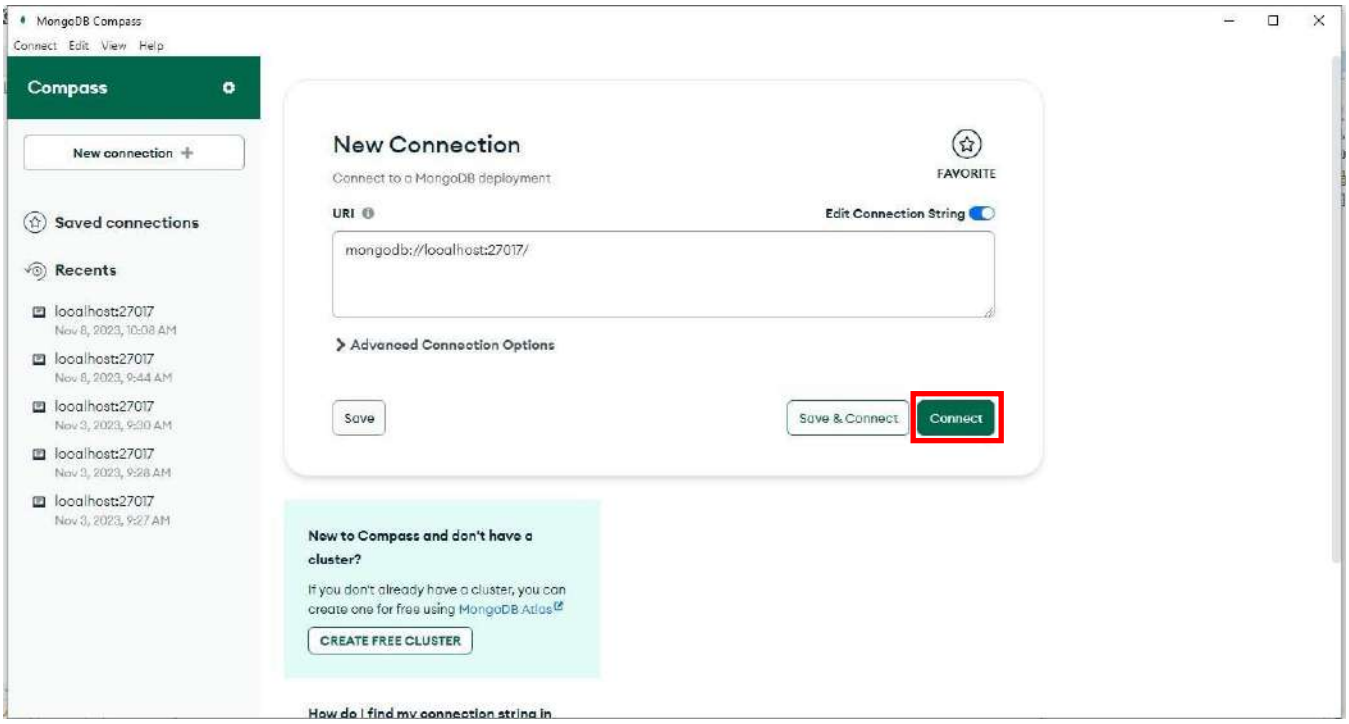
```java
package com.example.mongo1.repository;
import org.springframework.data.mongodb.repository.MongoRepository;
import com.example.mongo1.entity.Book;
public interface BookRepo
    extends MongoRepository<Book, Integer> {
}
```

Step 14: Now add code to **'application.properties'** file.

*BookController.java | Book.java | BookRepo.java | application.properties

```
server.port:8080
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=BookStore
```

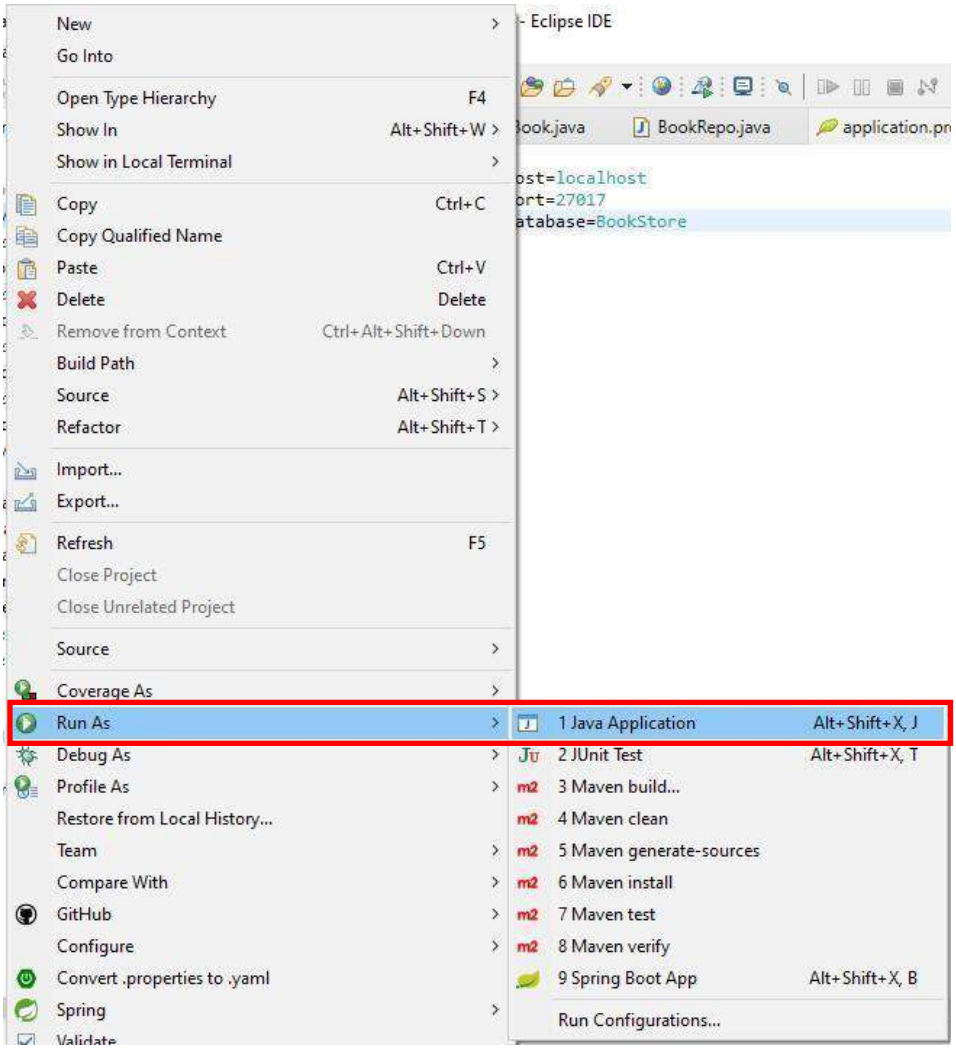
Step 15: Go to MongoDB Compass & connect to the existing URI path & then click on **'Connect'** here.



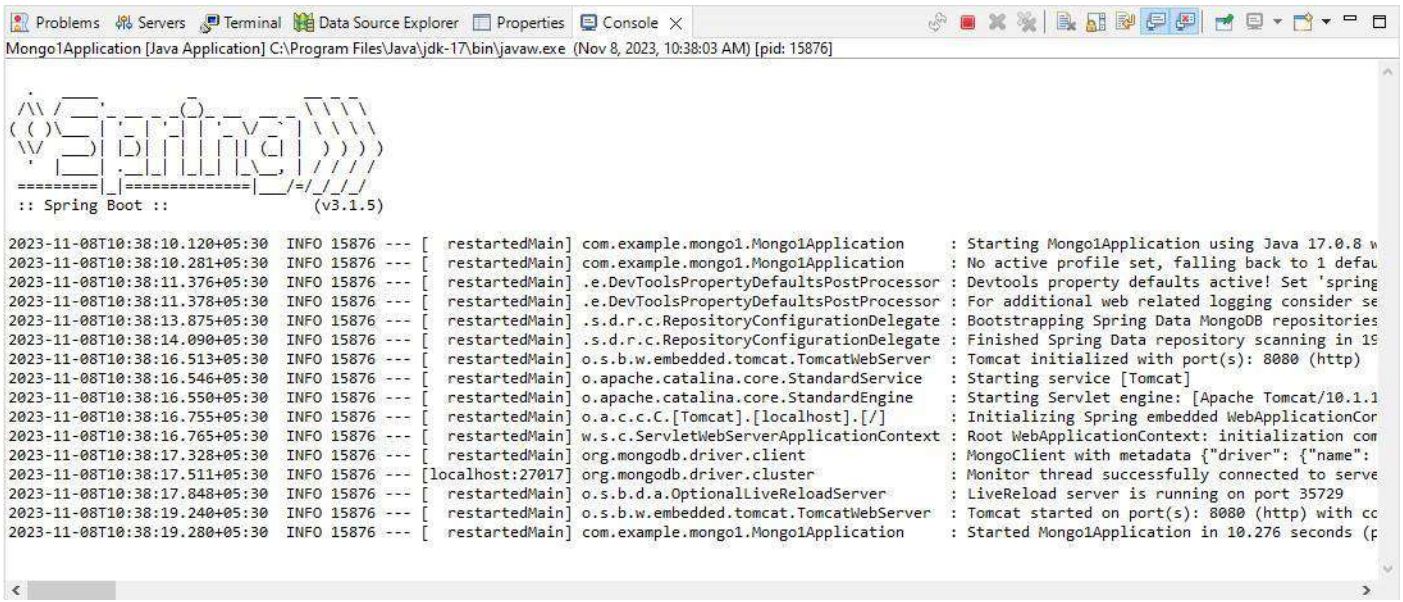
Step 16: Now create a new database named **'BookStore'** and collection named **'Book'** in Mongosh as shown below.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Using MongoDB:          6.0.11
Using Mongosh:          2.0.2

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

------
   The server generated these startup warnings when booting
   2023-11-08T09:10:30.752+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
  You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
test> use BookStore
switched to db BookStore
BookStore> show collections
Book
BookStore> show dbs
BookStore   72.00 KiB
admin       40.00 KiB
config     108.00 KiB
cs           8.00 KiB
dbms        48.00 KiB
desktop     72.00 KiB
dlt         48.00 KiB
fsd        184.00 KiB
local       72.00 KiB
network     80.00 KiB
sepp         8.00 KiB
test        40.00 KiB
```

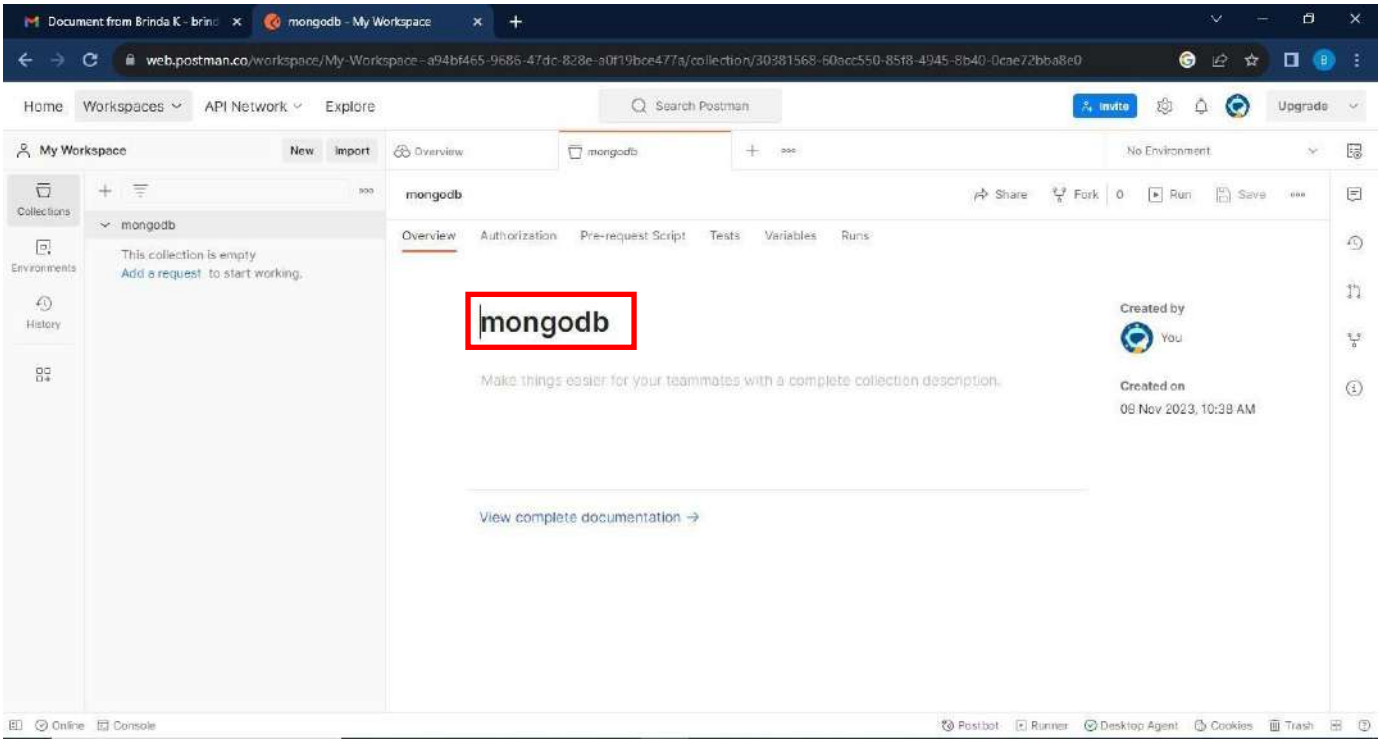
Step 17: Now run the main method file in Eclipse as shown below.



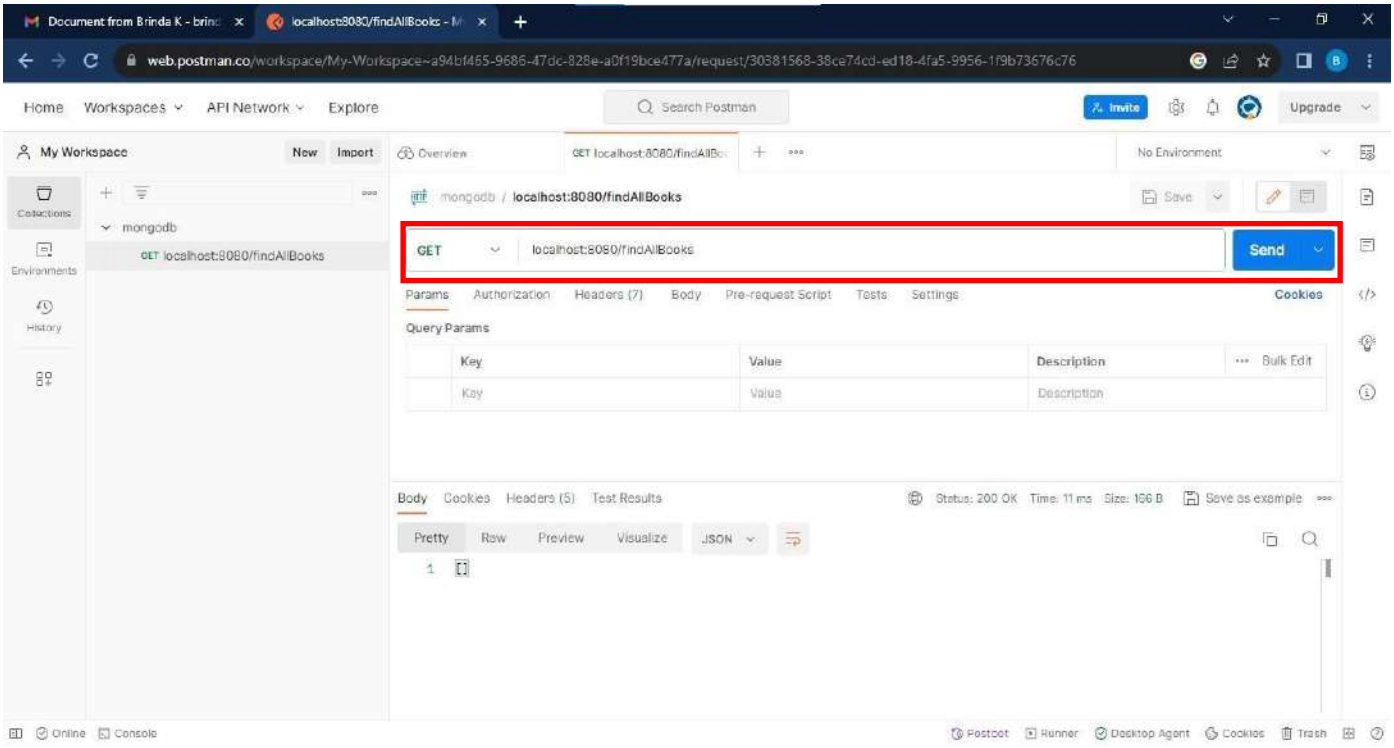
Step 18: Observe that the server is up.



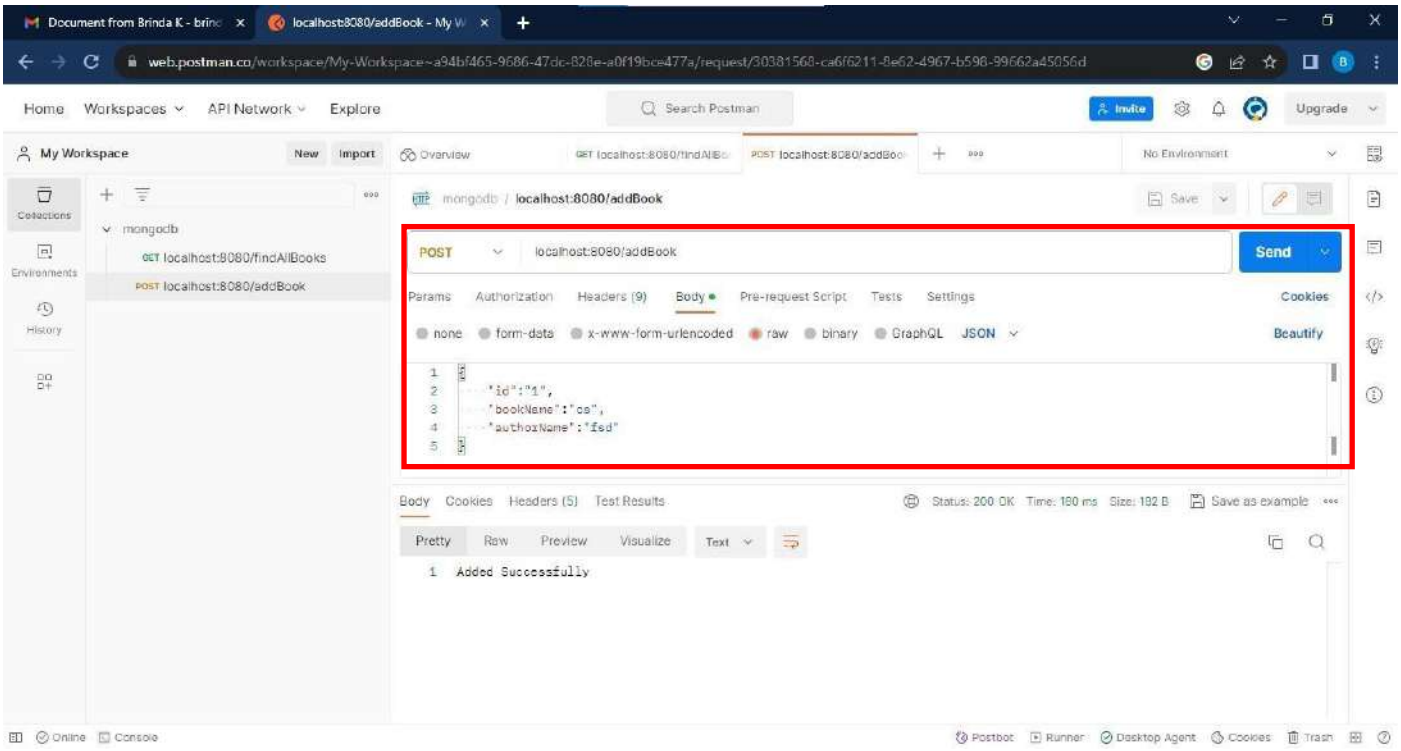
Step 19: Now login to your PostMan workspace and create a new collection named **‘mongodb’** as shown below.



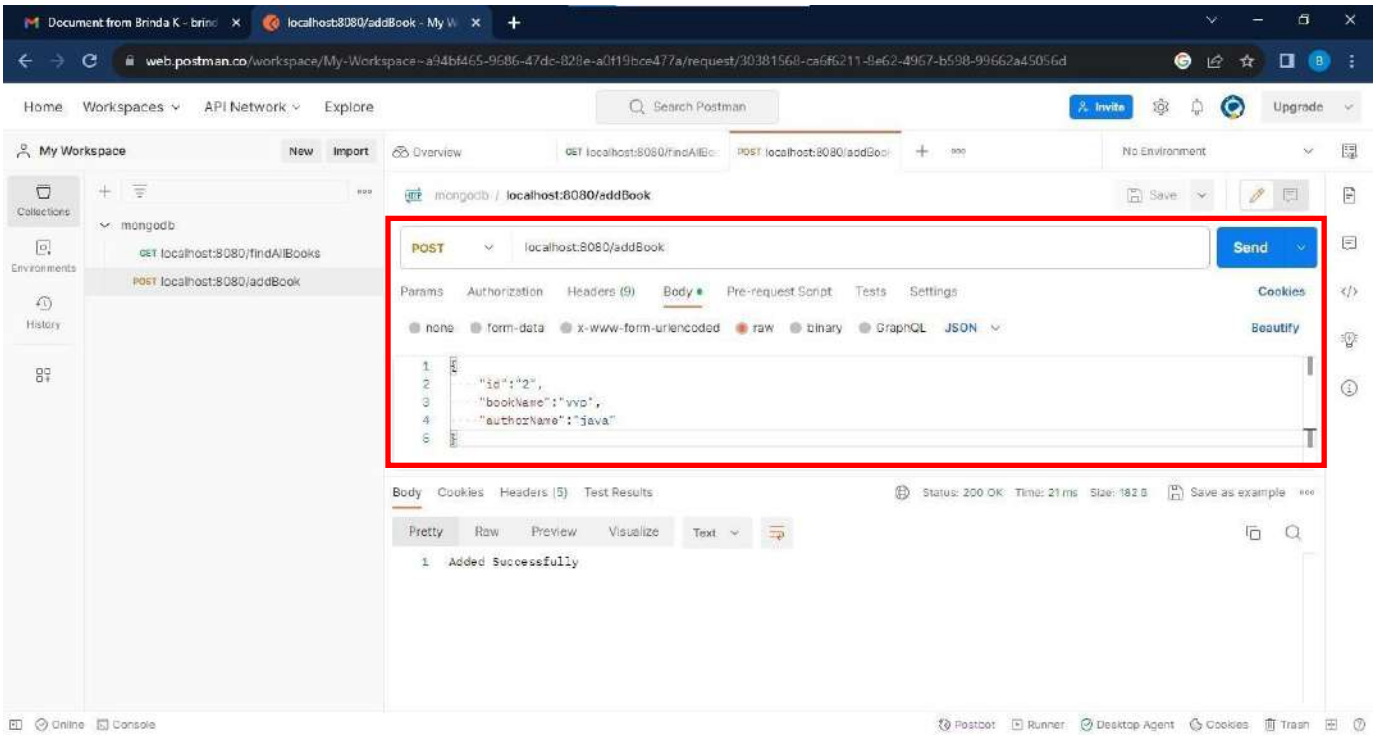
Step 20: Perform GET method by entering the command as **localhost:8080/findAllBooks** here.



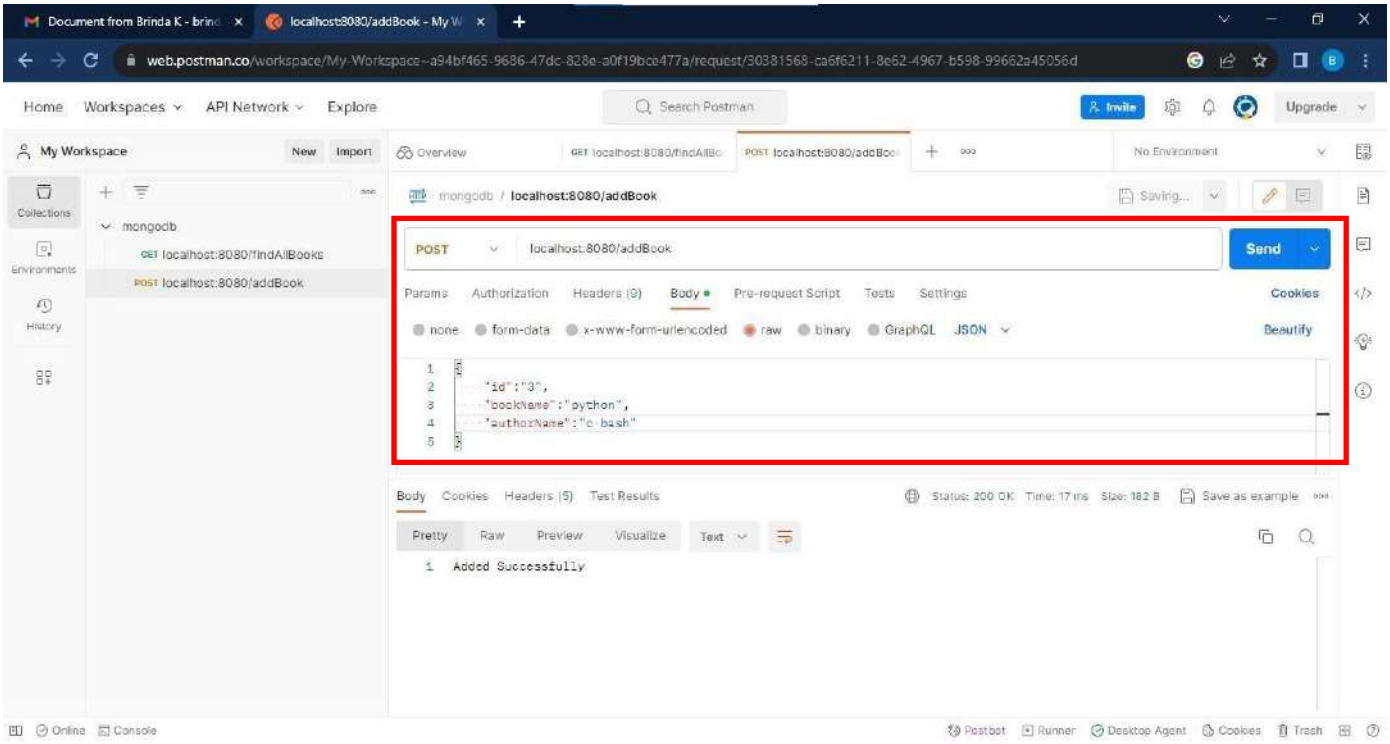
Step 21: Perform POST method by entering the command as **localhost:8080/addBook** here.



Step 22: Perform POST method to one more document as shown below.



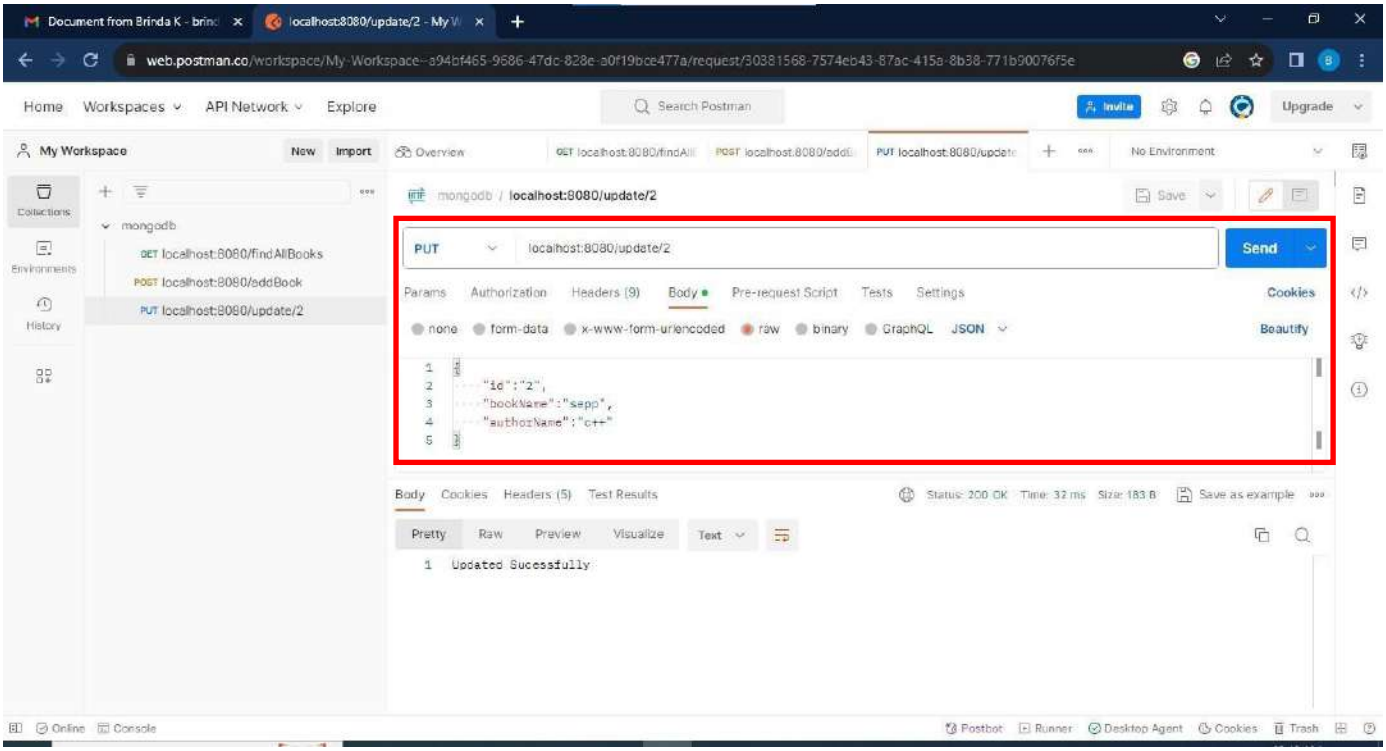
Step 23: Perform POST method to another document as shown below.



Step 24: Go to Mongosh to read the inserted documents as shown below.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
   2023-11-08T09:10:30.752+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
  You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
test> use BookStore
switched to db BookStore
BookStore> show collections
Book
BookStore> show dbs
BookStore   72.00 KiB
admin       40.00 KiB
config     108.00 KiB
cs           8.00 KiB
dbms        48.00 KiB
desktop     72.00 KiB
dlt         48.00 KiB
fsd        184.00 KiB
local       72.00 KiB
network     80.00 KiB
sepp         8.00 KiB
test        40.00 KiB
BookStore> db.Book.find()
[
  {
    _id: 1,
    bookName: 'cs',
    authorName: 'fsd',
    _class: 'com.example.mongo1.entity.Book'
  },
  {
    _id: 2,
    bookName: 'vvp',
    authorName: 'java',
    _class: 'com.example.mongo1.entity.Book'
  },
  {
    _id: 3,
    bookName: 'python',
    authorName: 'c bash',
    _class: 'com.example.mongo1.entity.Book'
  }
]
```

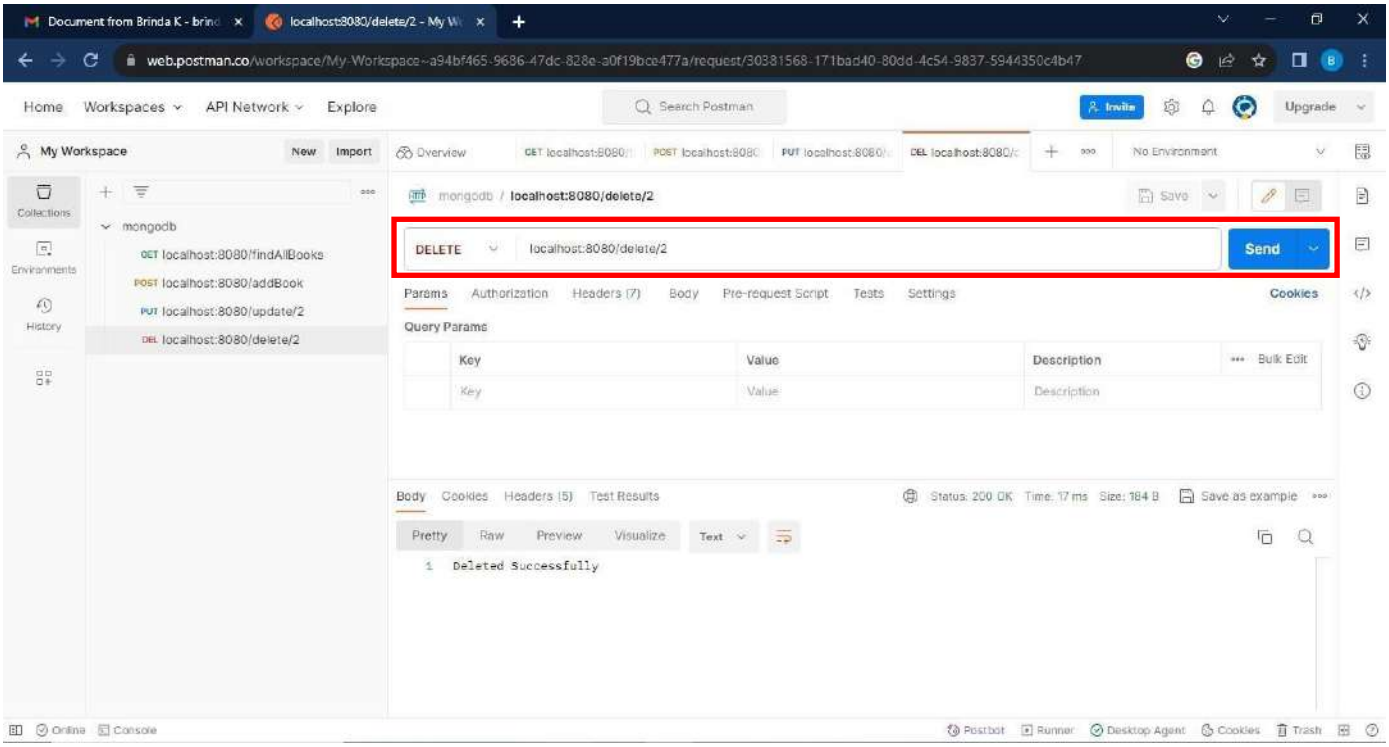
Step 25: Perform PUT method by entering the command as **localhost:8080/update/2** here.



Step 26: Use the below command to check whether update operation is reflecting in Mongosh.

```
BookStore> db.Book.find()
[
  {
    _id: 1,
    bookName: 'cs',
    authorName: 'fsd',
    _class: 'com.example.mongo1.entity.Book'
  },
  {
    _id: 2,
    bookName: 'sepp',
    authorName: 'c++',
    _class: 'com.example.mongo1.entity.Book'
  },
  {
    _id: 3,
    bookName: 'python',
    authorName: 'c bash',
    _class: 'com.example.mongo1.entity.Book'
  }
]
```

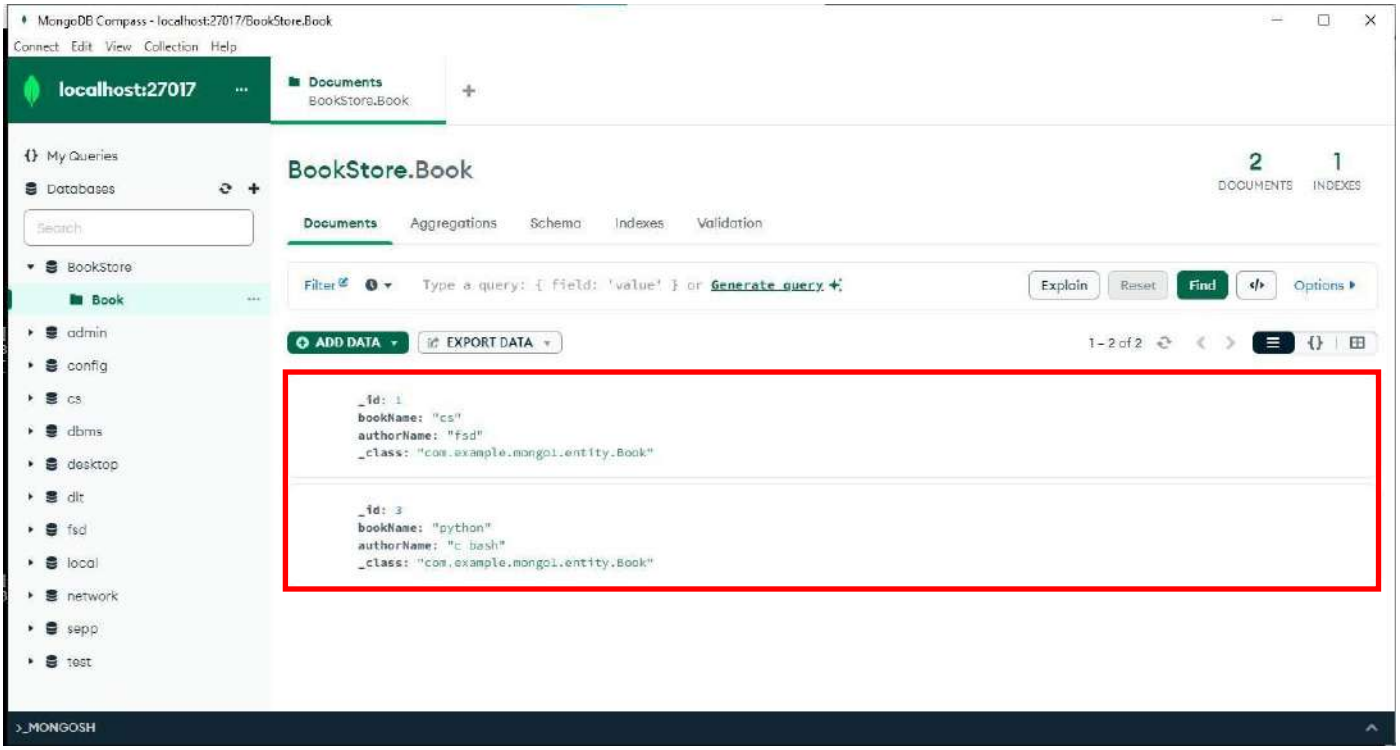
Step 27: Perform DELETE method by entering the command as **localhost:8080/delete/2** here.



Step 28: Use the below command to check whether delete operation is reflecting in Mongosh.

```
BookStore> db.Book.find()
[
  {
    _id: 1,
    bookName: 'cs',
    authorName: 'fsd',
    _class: 'com.example.mongo1.entity.Book'
  },
  {
    _id: 3,
    bookName: 'python',
    authorName: 'c bash',
    _class: 'com.example.mongo1.entity.Book'
  }
]
```

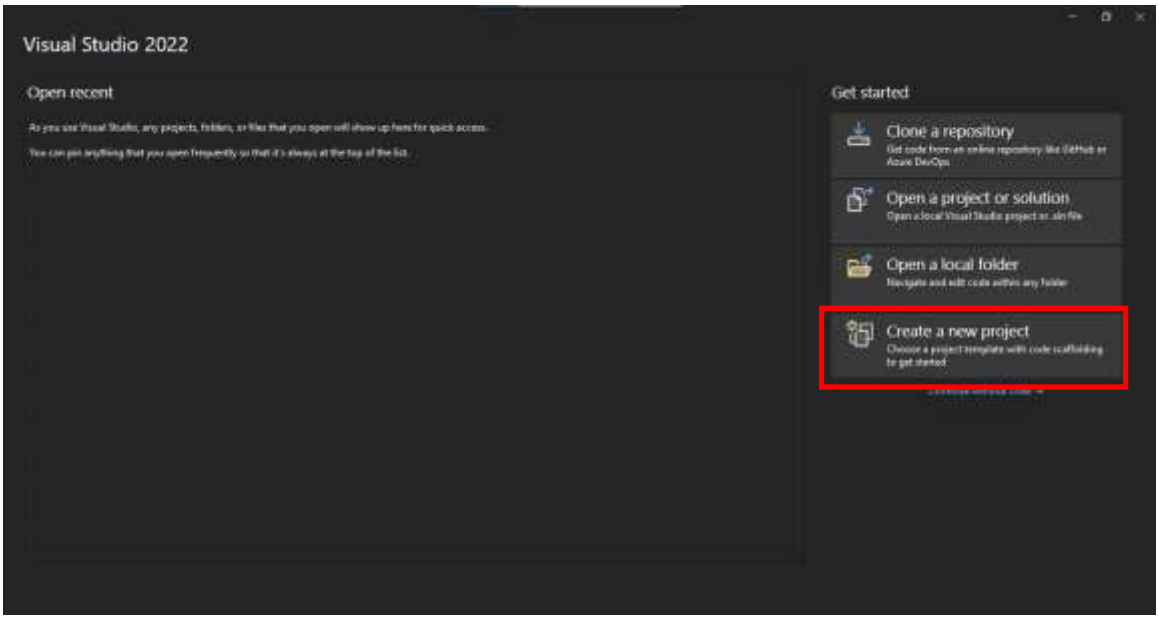
Step 29: Follow the same procedures to perform CRUD operations in MongoDB Compass through POSTMAN as well.


MongoDB Compass - localhost:27017/BookStore.Book
Connect Edit View Collection Help
localhost:27017
Documents
BookStore.Book
My Queries
Databases
Search
BookStore
Book
admin
config
cs
dbms
desktop
dlt
fsd
local
network
sepp
test
BookStore.Book
2
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 2 of 2
_id: 1
bookName: "cs"
authorName: "fsd"
_class: "com.example.mongo1.entity.Book"
_id: 3
bookName: "python"
authorName: "c bash"
_class: "com.example.mongo1.entity.Book"
>_MONGOSH

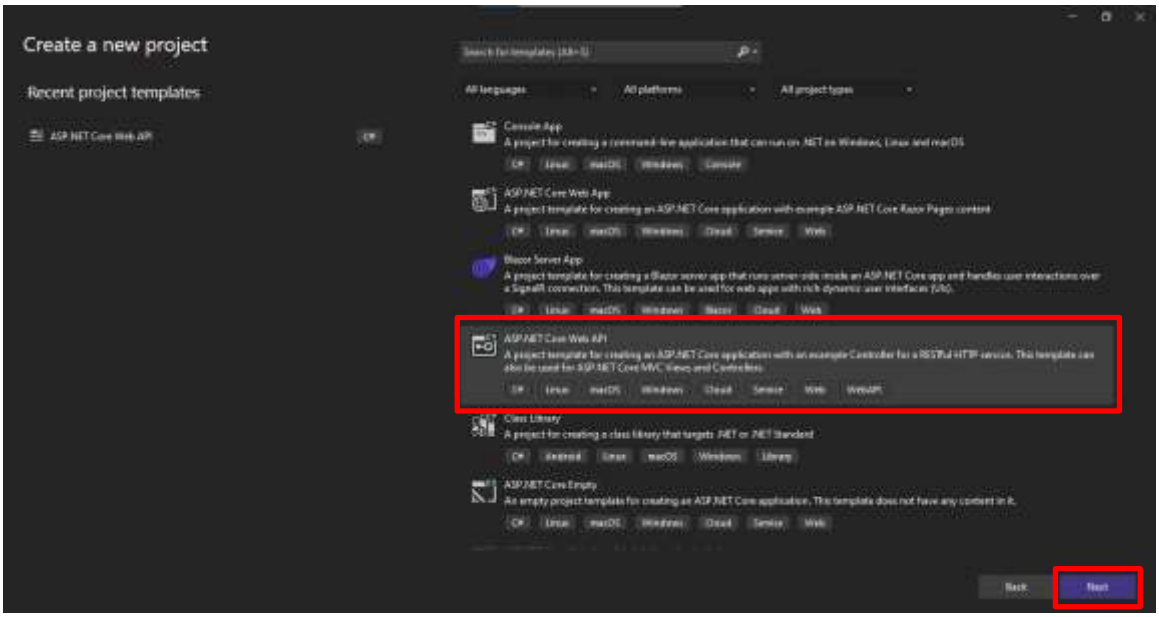
Week-11

1. Integrate the work of each group and carry out integration testing.

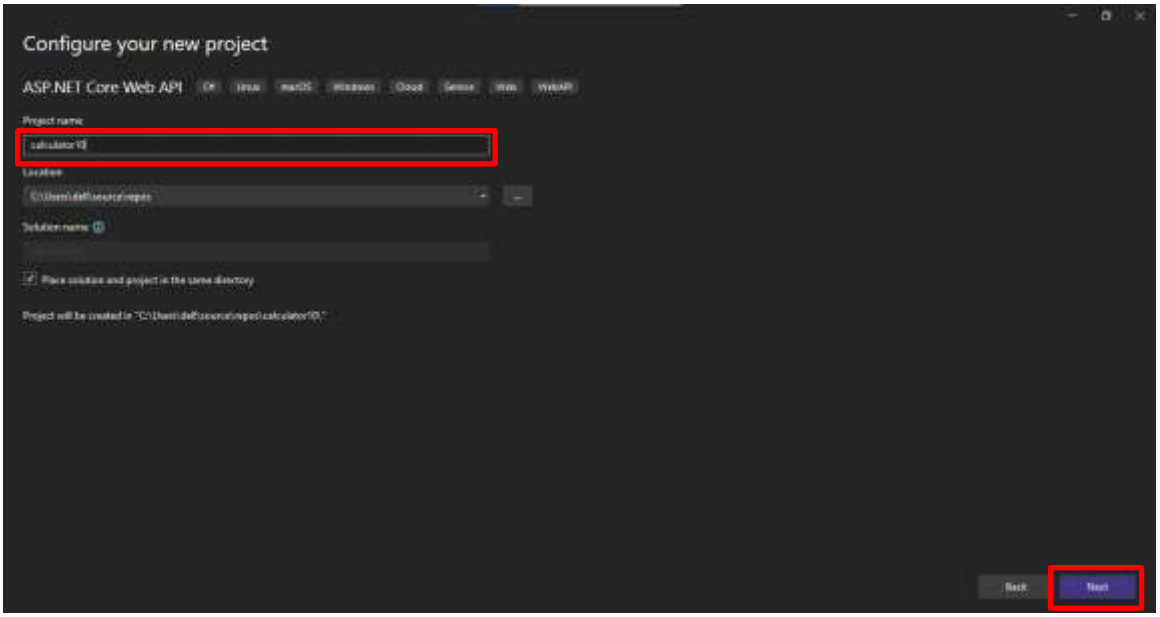
Step 1: Open Microsoft Visual Studios And click on **'Create a new Project'**.



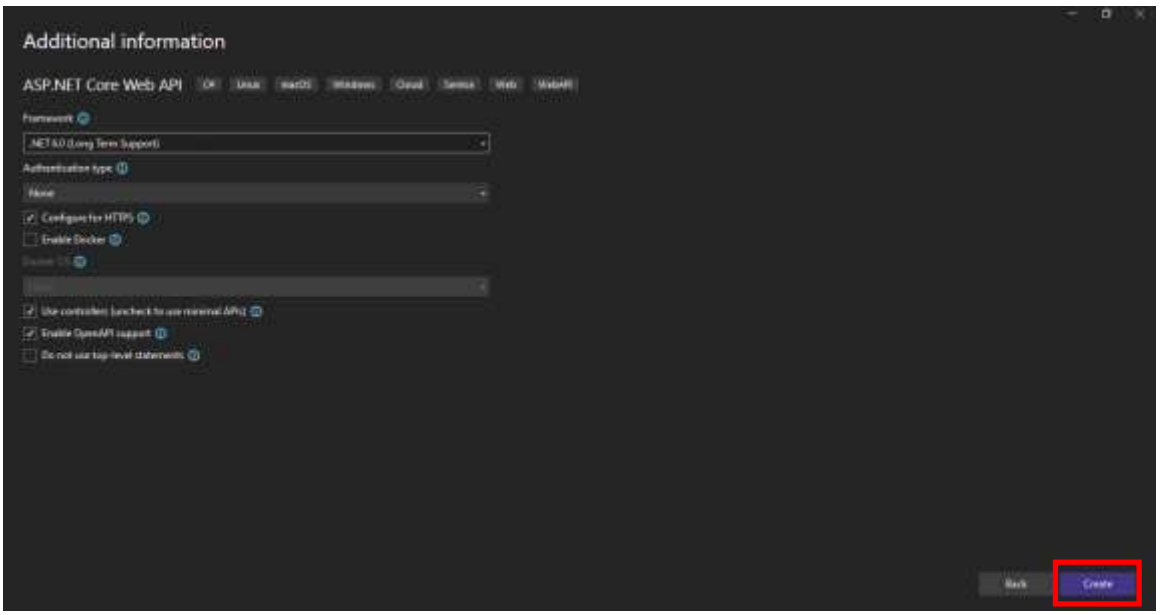
Step 2: Select **'ASP.NETCORE Web API'** & click on **'Next'.**



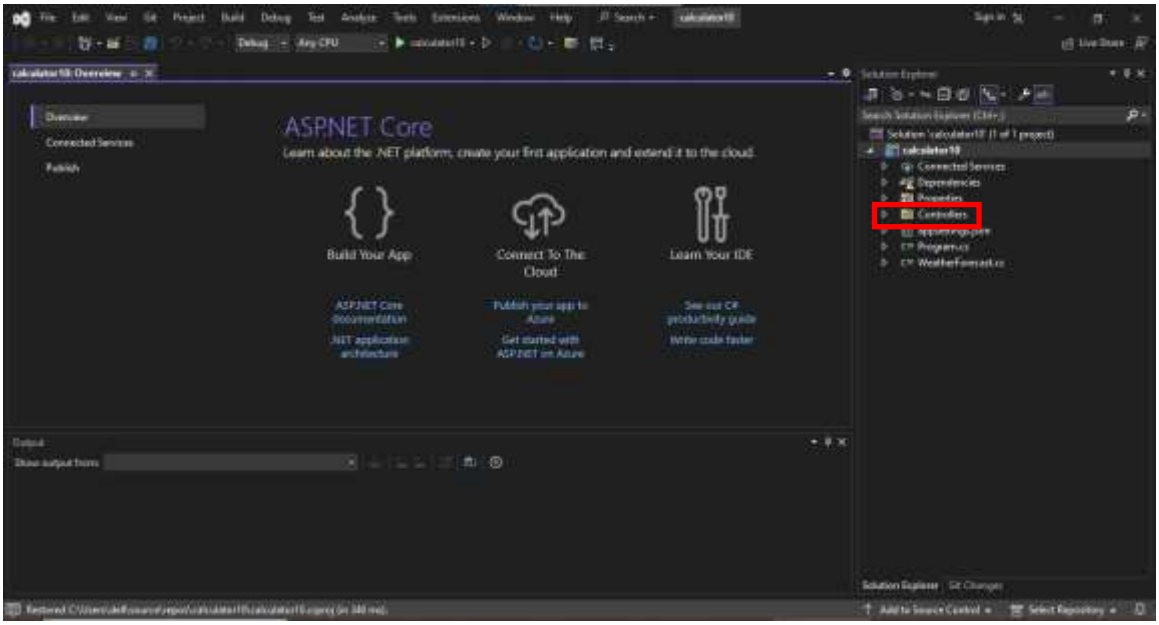
Step 3: Give a Project name & click on **'Next'.**



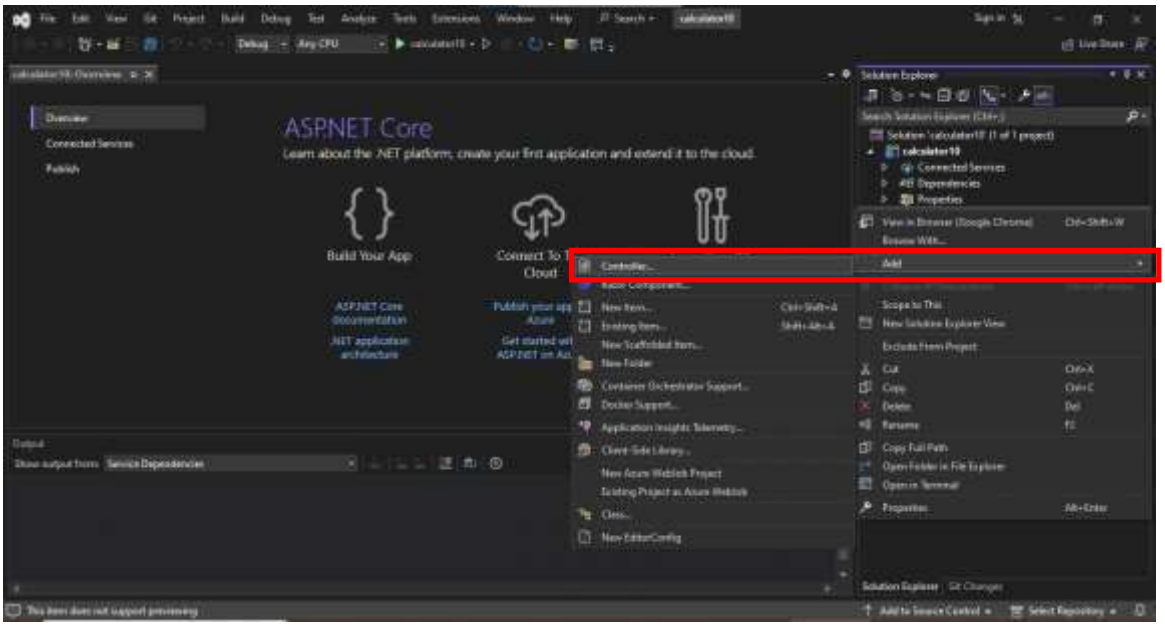
Step 4: Disable the Docker if you don’t Docker Desktop installed & Click on **‘Create’**.



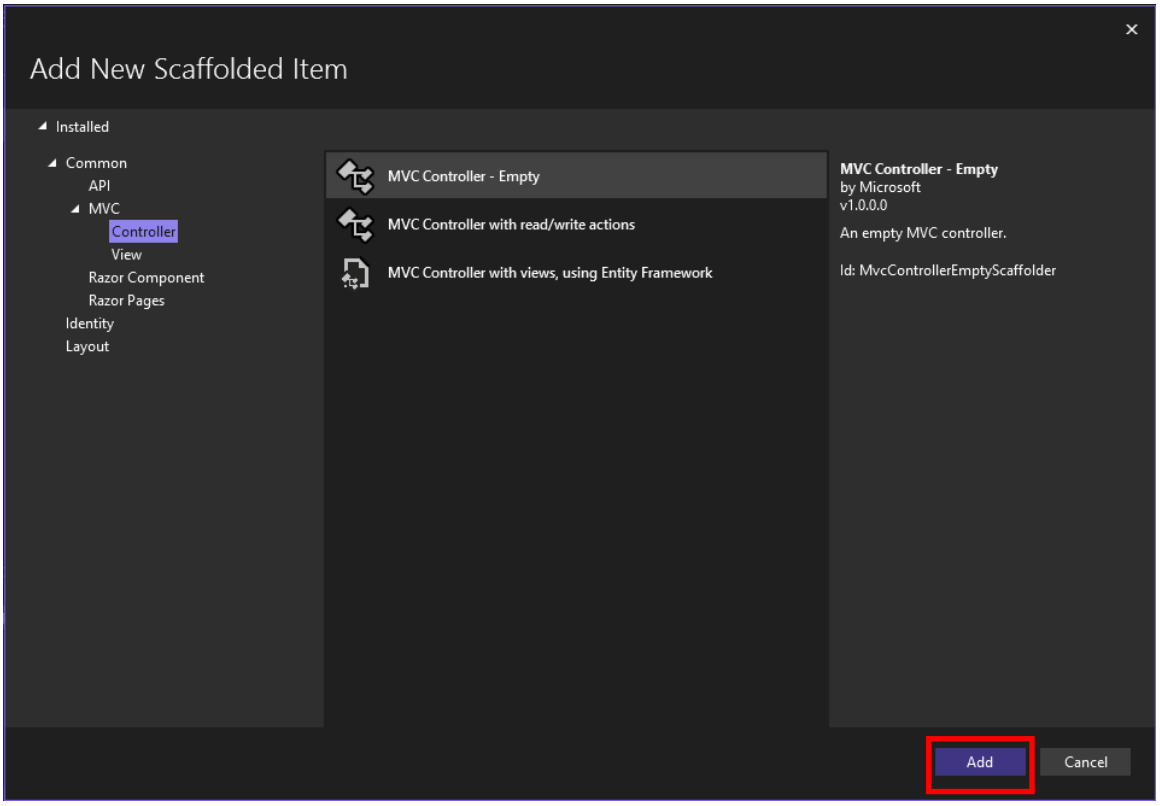
Step 5: Now right click on **‘controllers’** here.



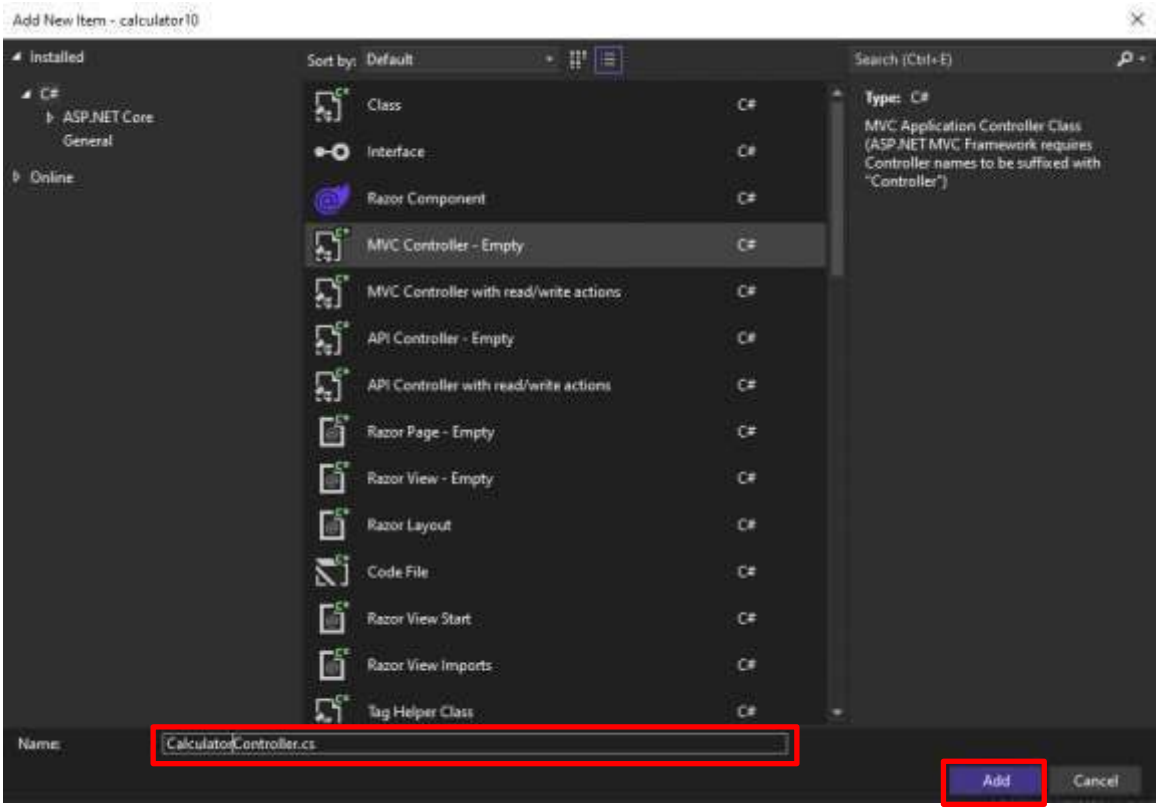
Step 6: Select **Add** & click on **'Controllers'**.



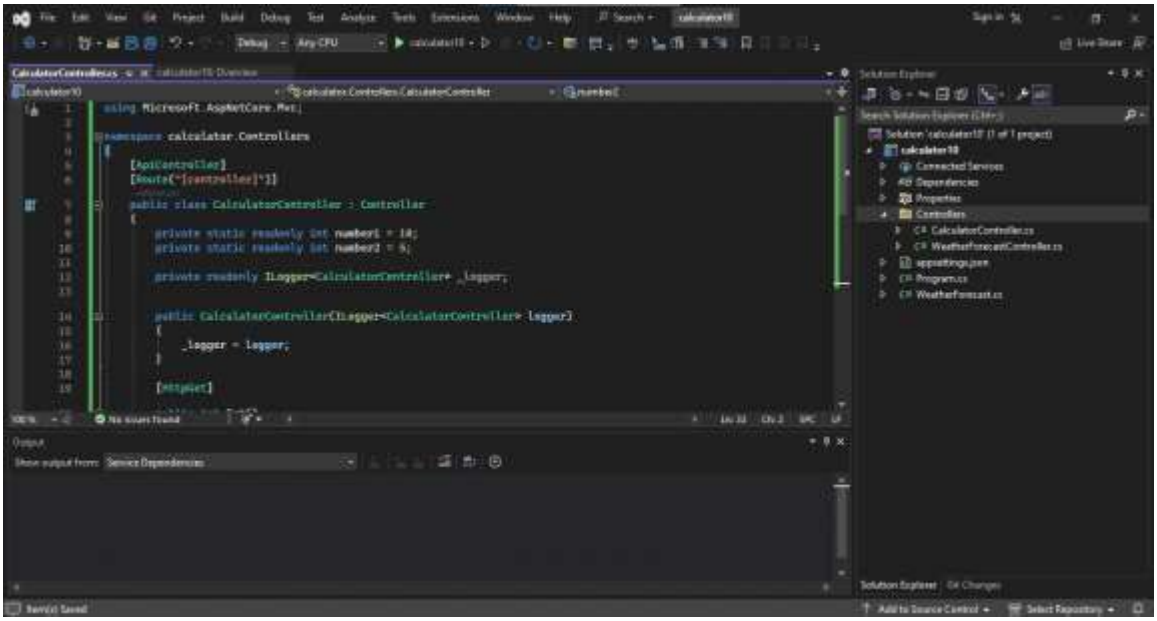
Step 7: Click on **Add**.



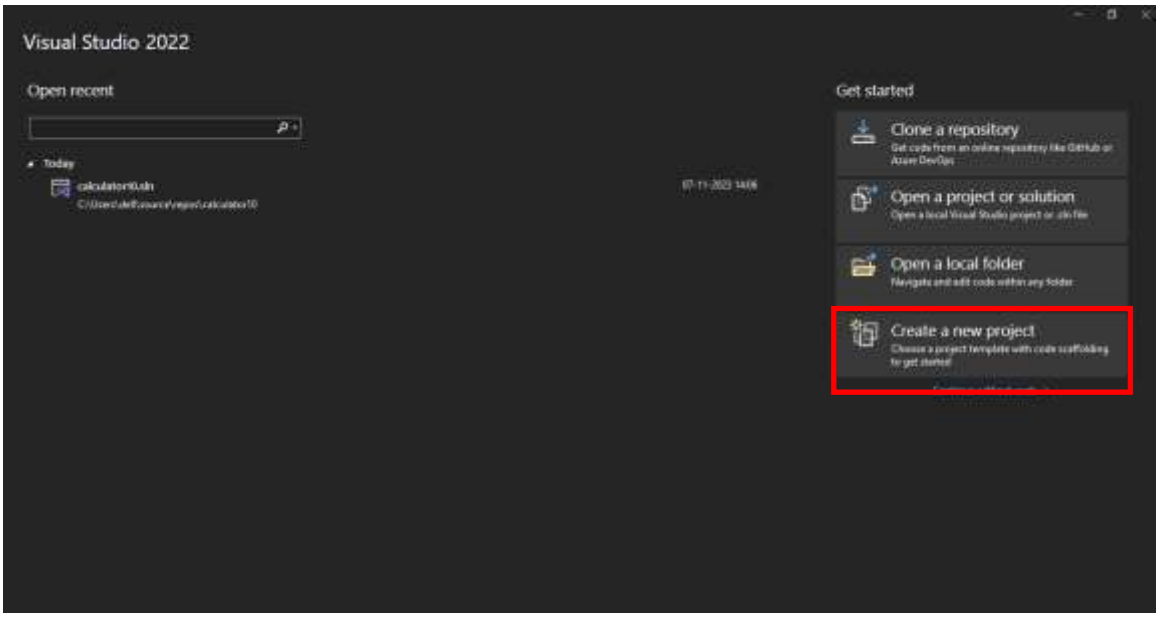
Step 8: Give a name & click on **Add**.



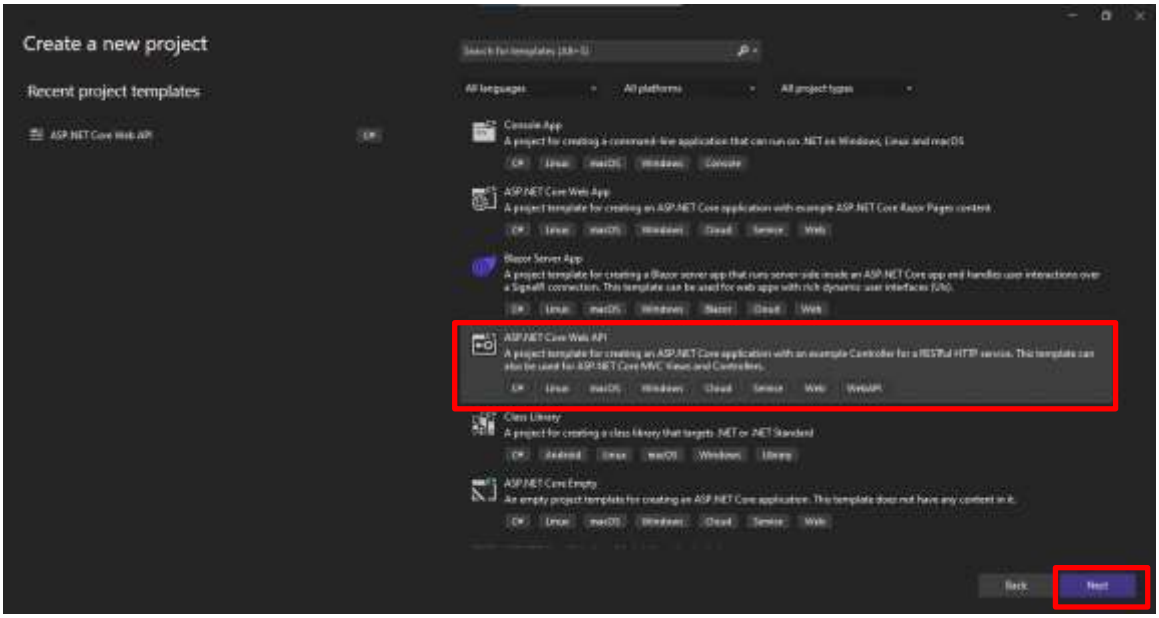
Step 9: Add code to **Calculator.Controller.cs** file.



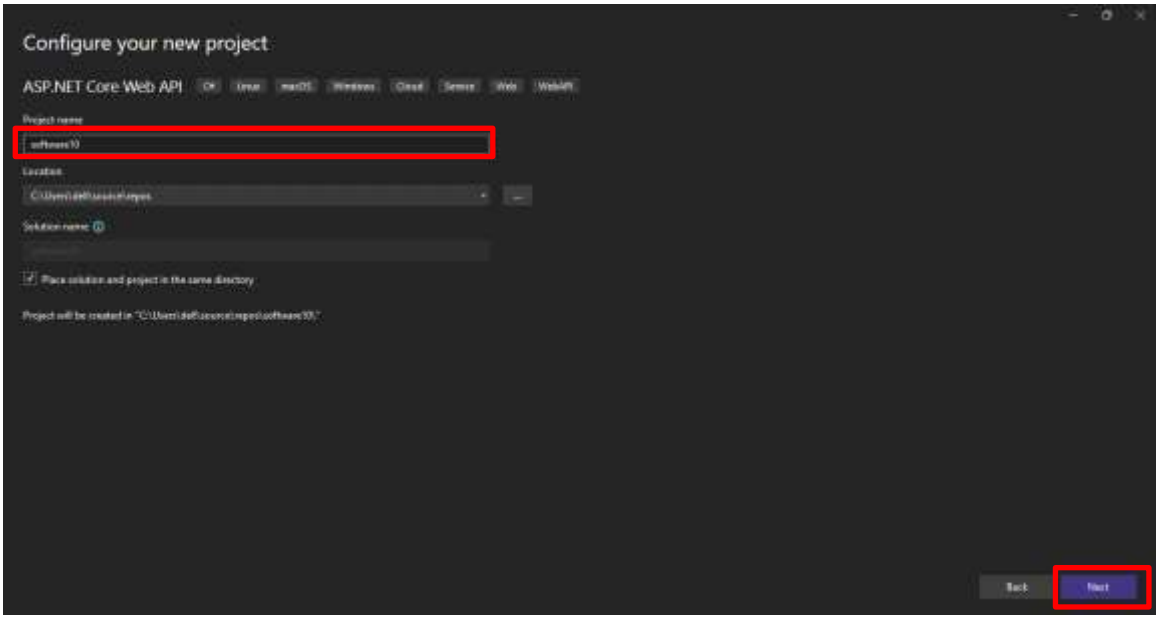
Step 10: Now close the other windows and open the Microsoft Visual Studios one more time & click on ‘Create a new Project’.



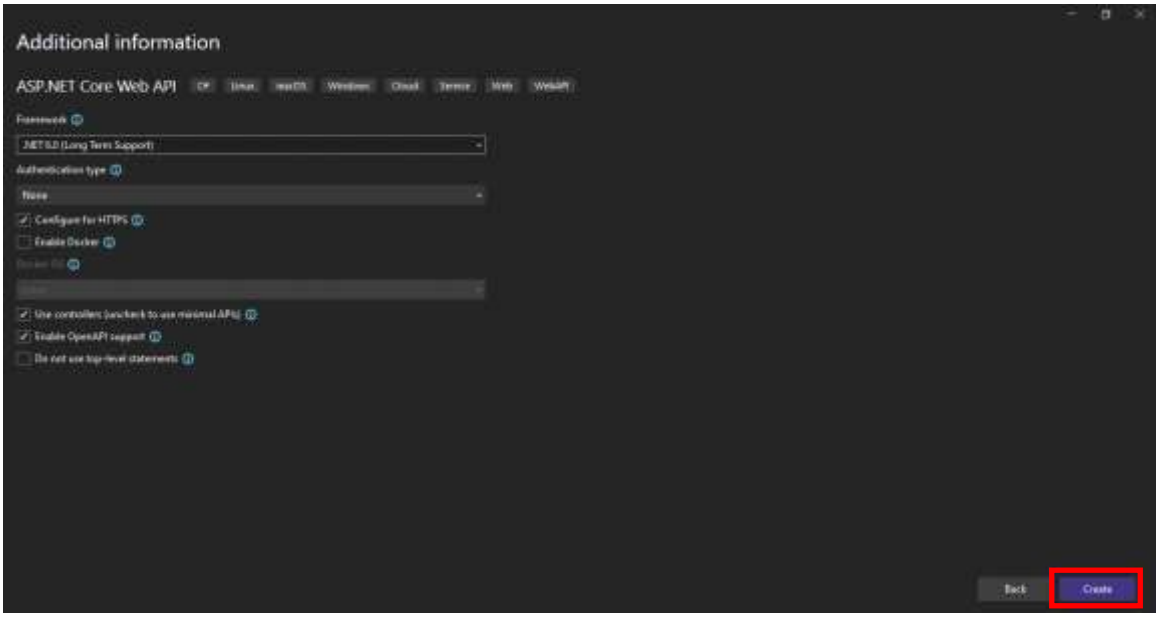
Step 11: Select **‘ASP.NETCORE Web API’** & click on **‘Next’.**



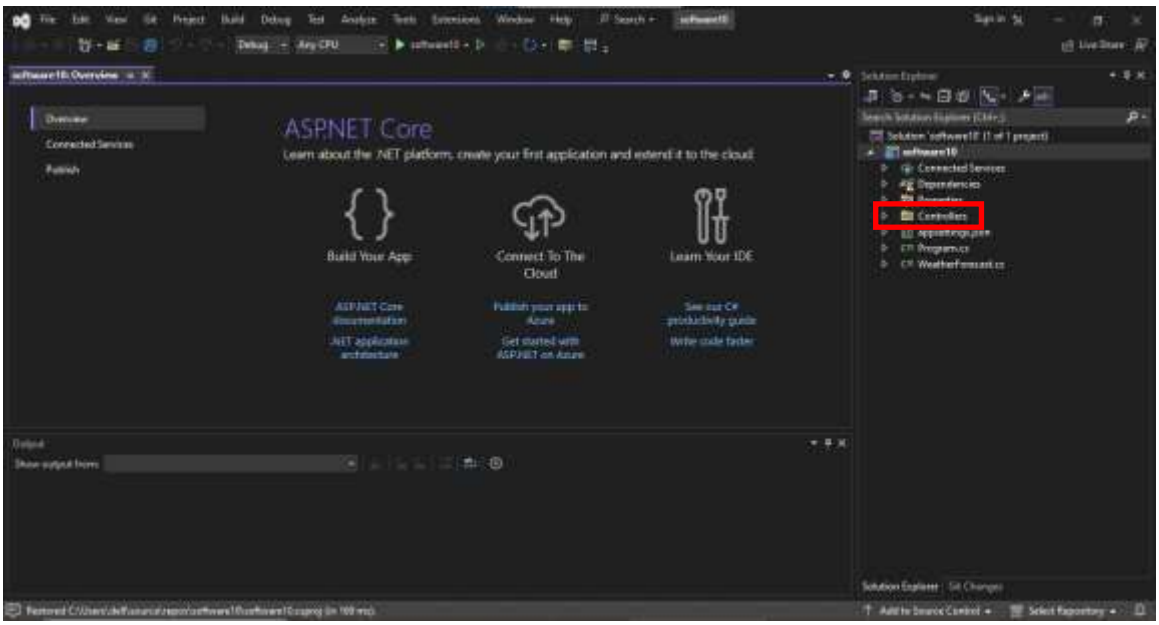
Step 12: Give a Project name & click on **'Next'.**



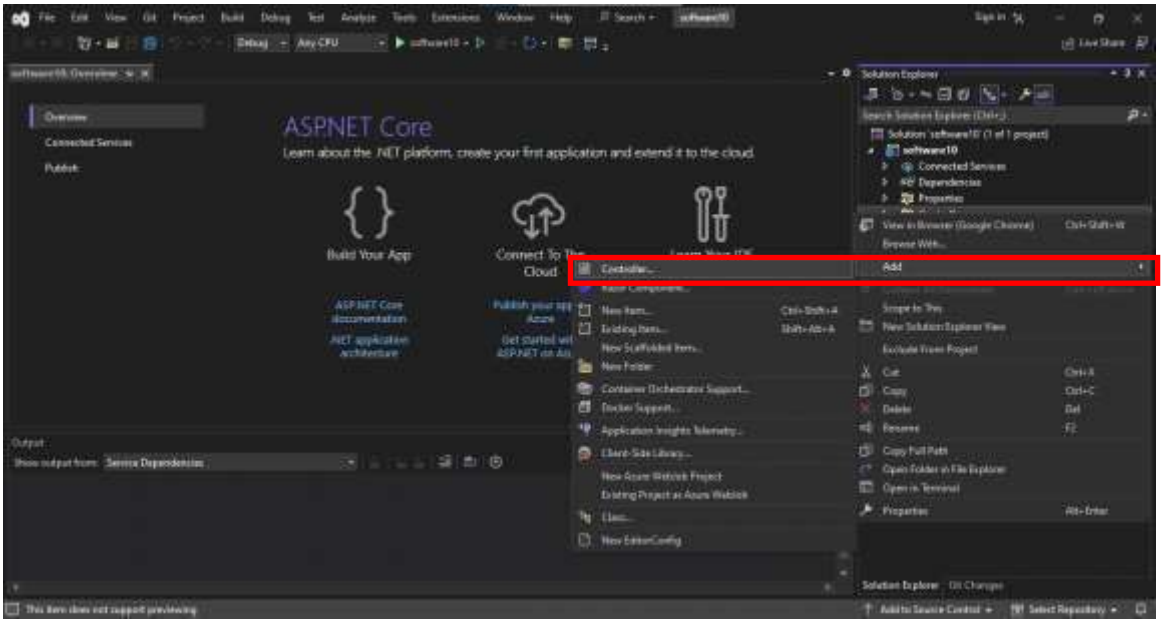
Step 13: Disable the Docker if you don't Docker Desktop installed & Click on **'Create'**.



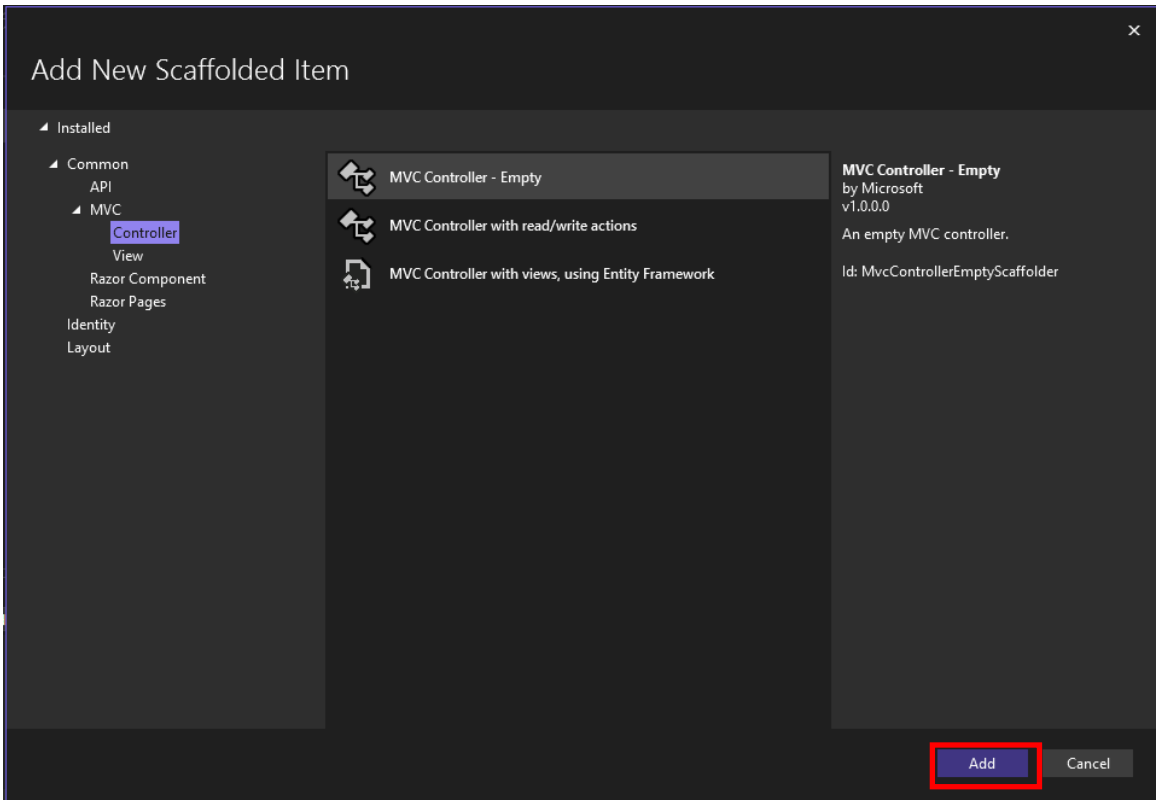
Step 14: Now right click on **‘controllers’** here.



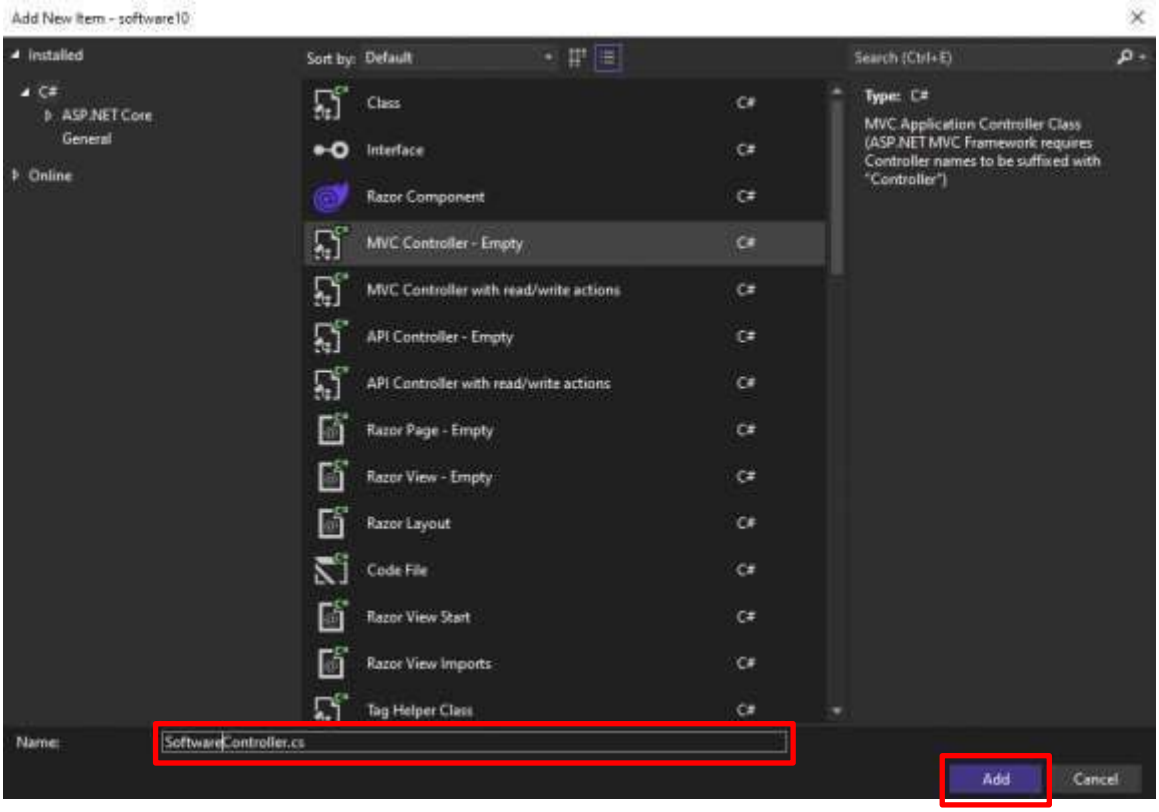
Step 15: Select **Add** & click on **‘Controllers’**.



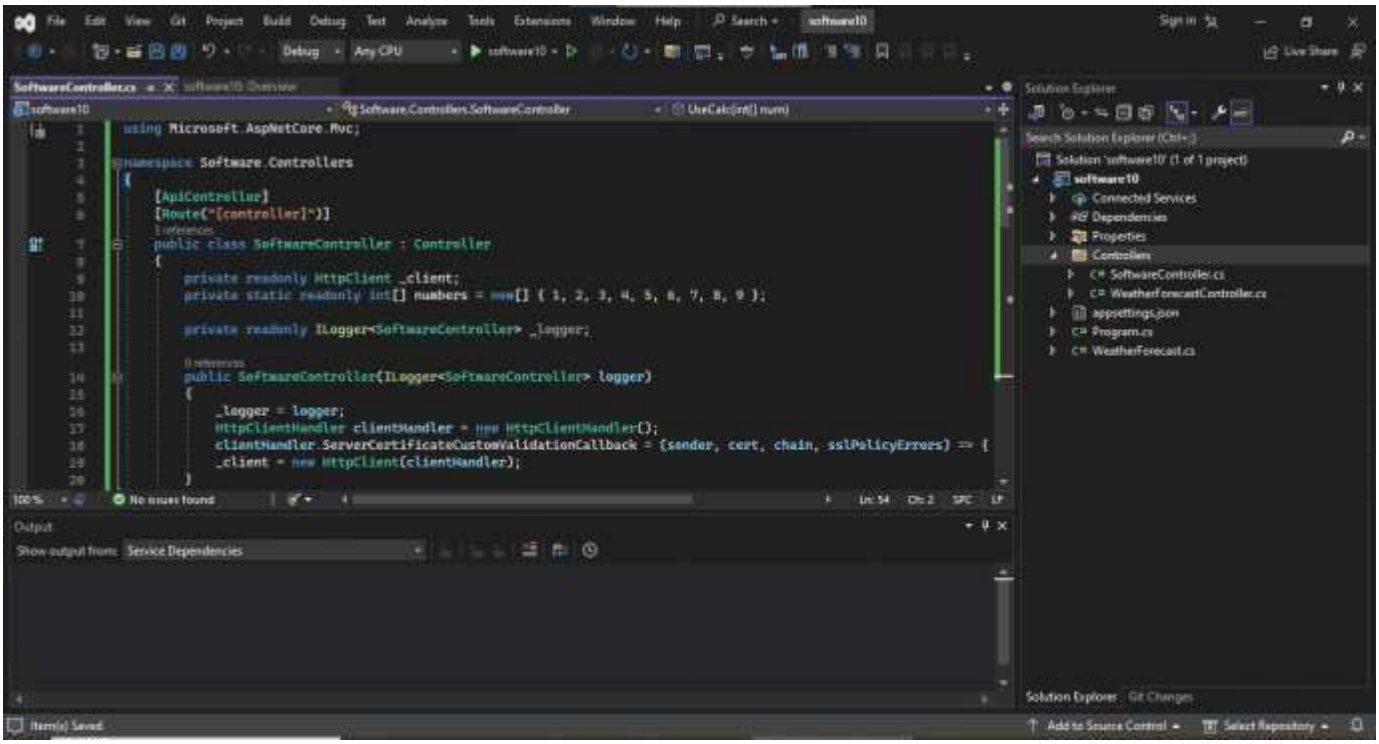
Step 16: Click on **Add**.



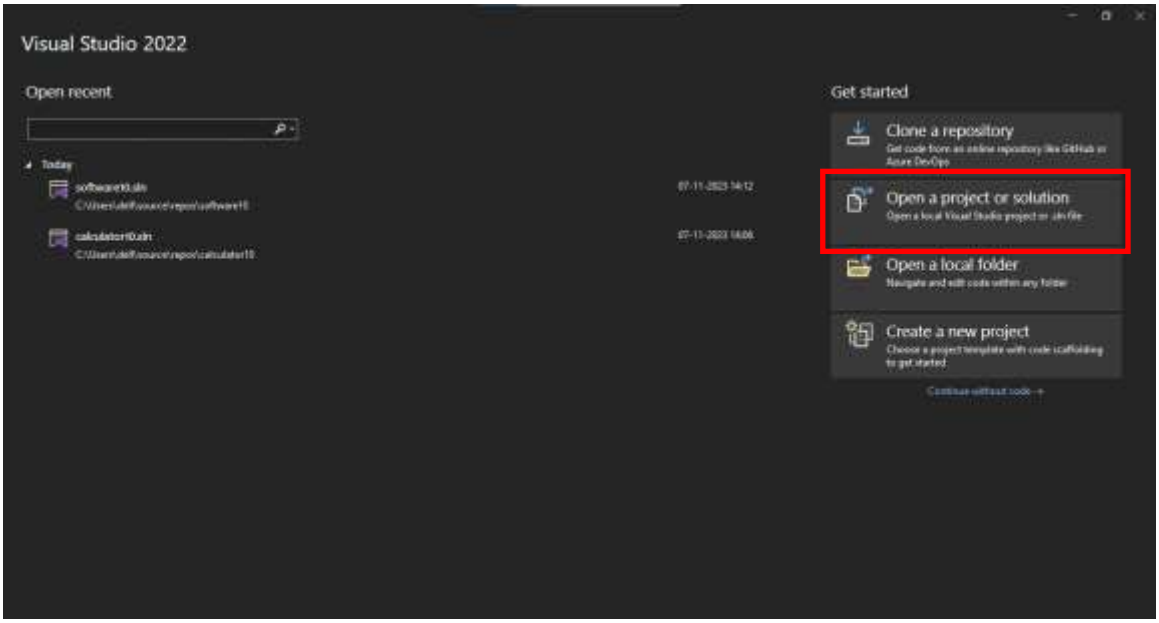
Step 17: Give a name & click on **Add**.



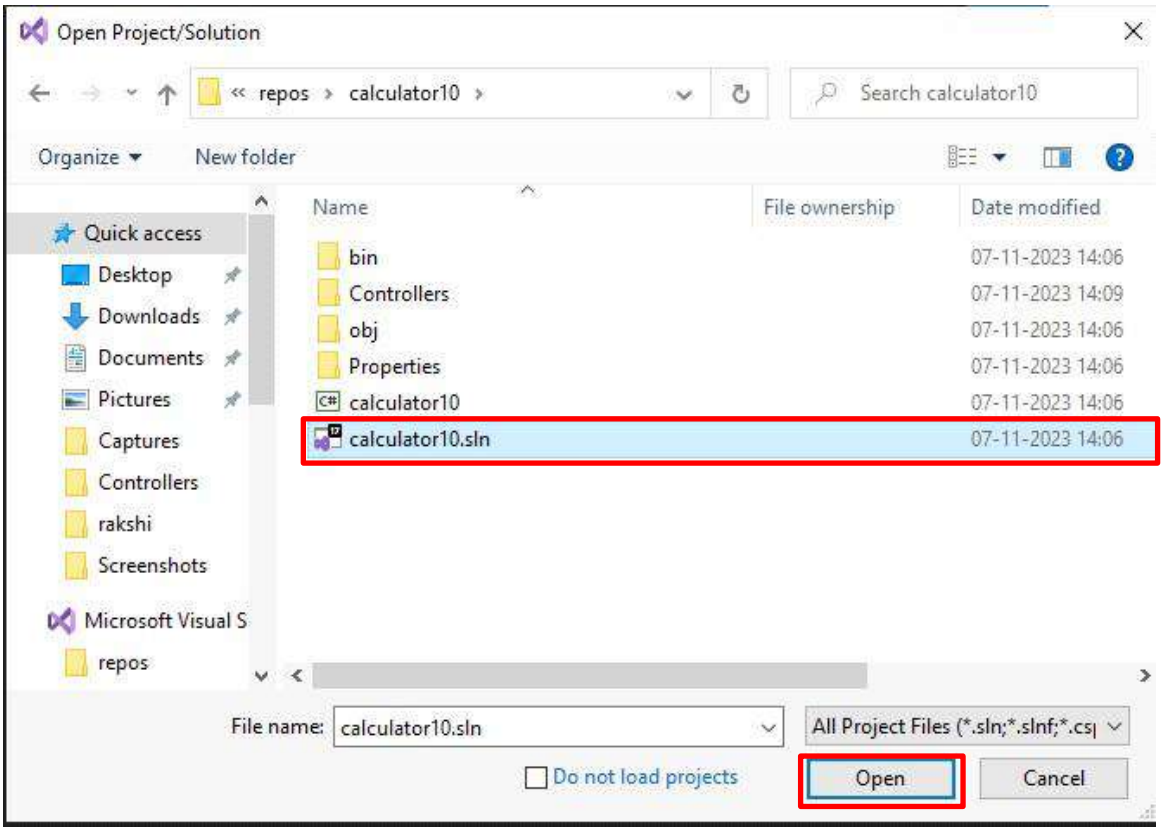
Step 18: Add code to **Software.Controller.cs** file.



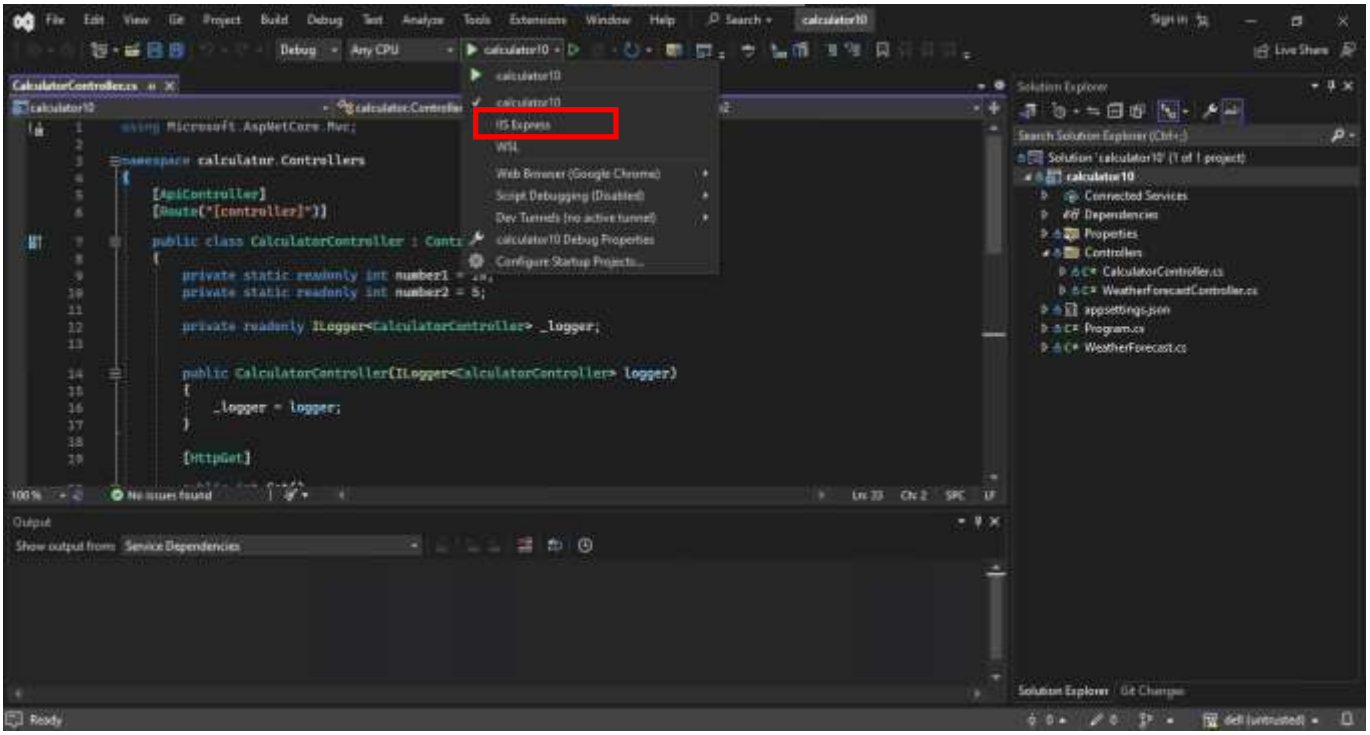
Step 19: Now once again open the Microsoft Visual Studios & click on **'Open a Project Solution'**.



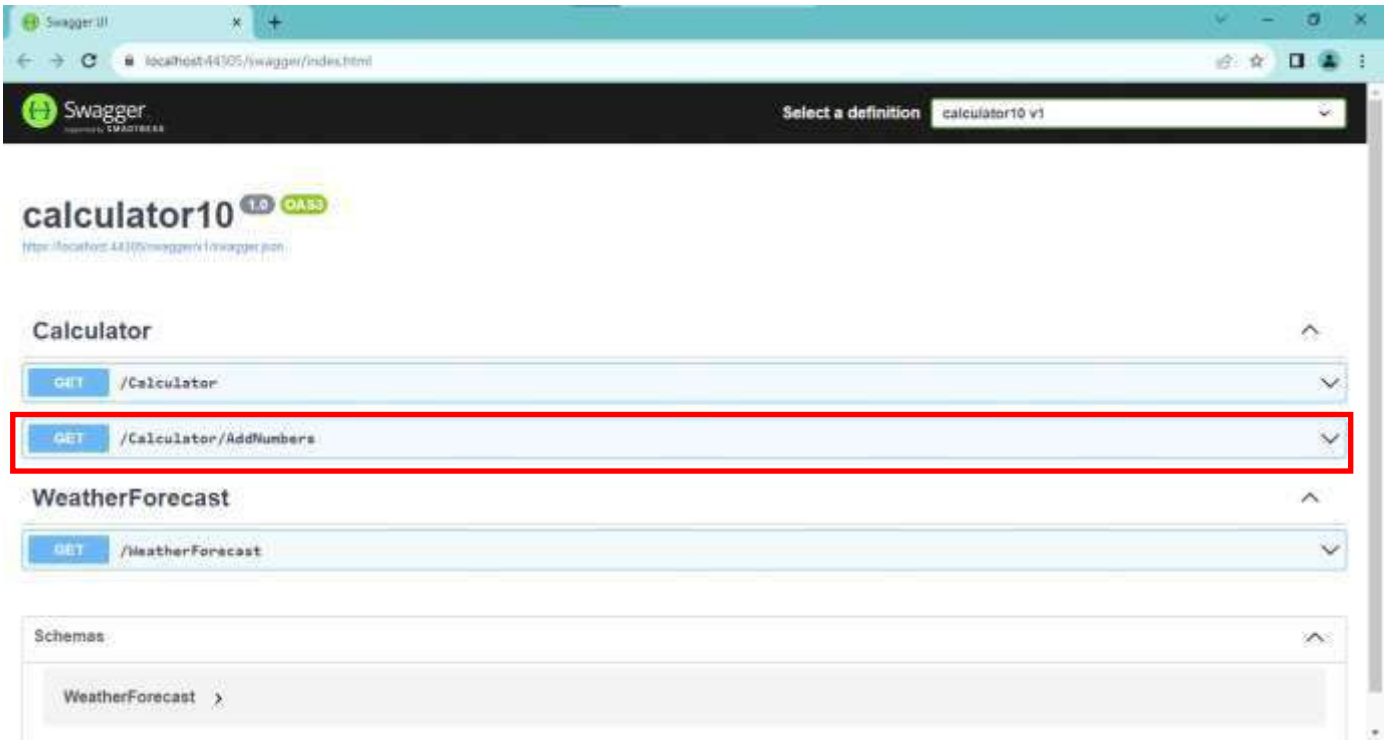
Step 20: Now open the **calculator10.sln** file.



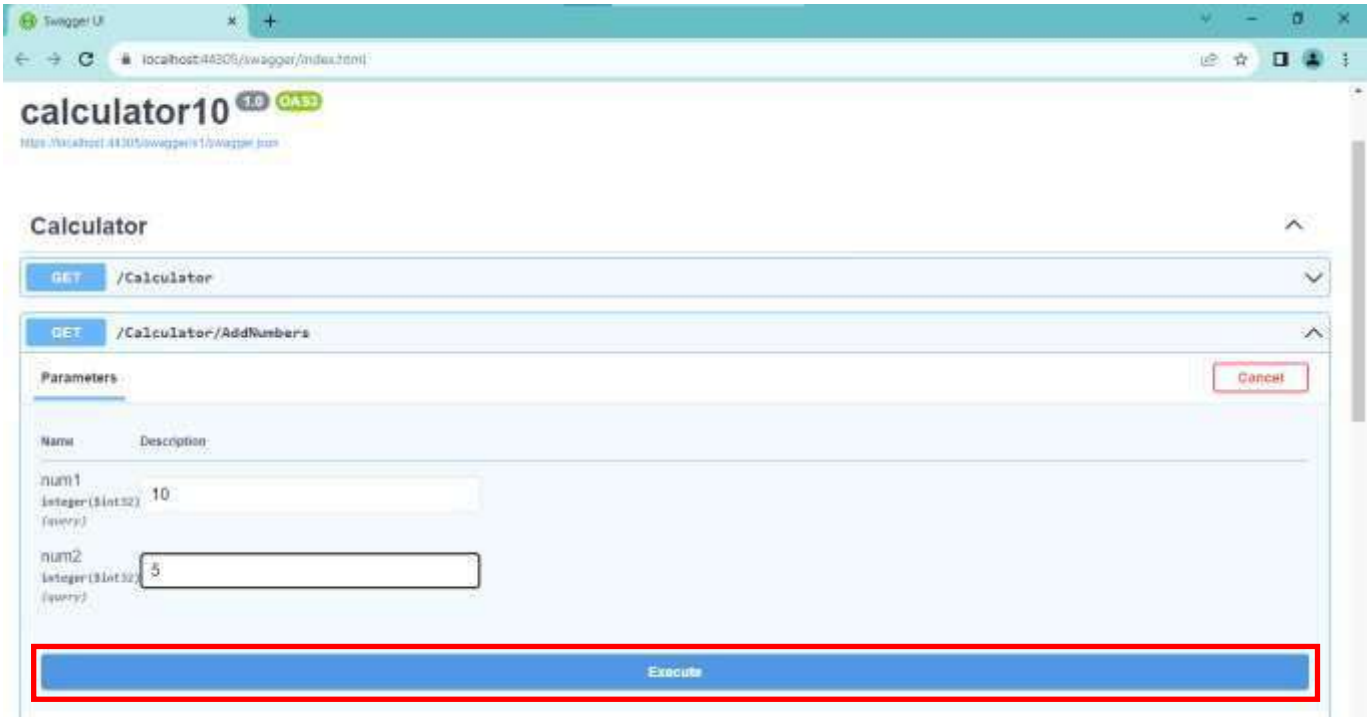
Step 21: Select **IIS Express** for running the code.



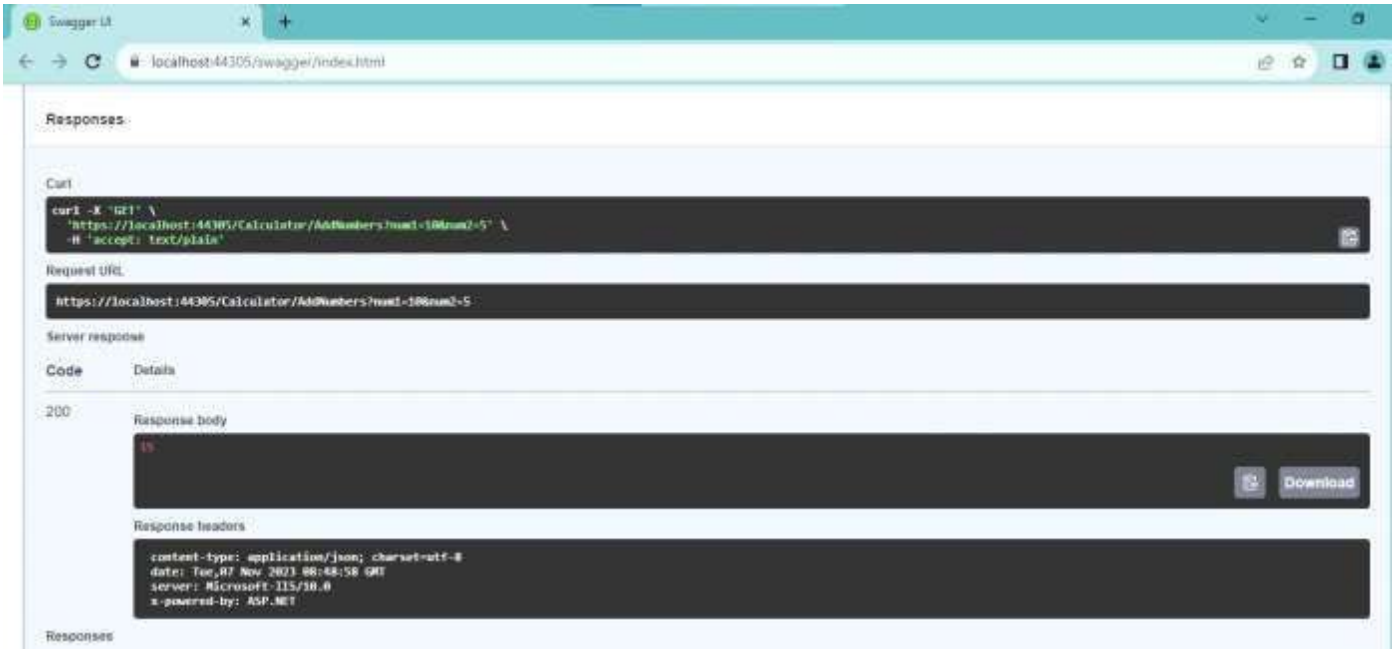
Step 22: Now click on **Add Numbers**.



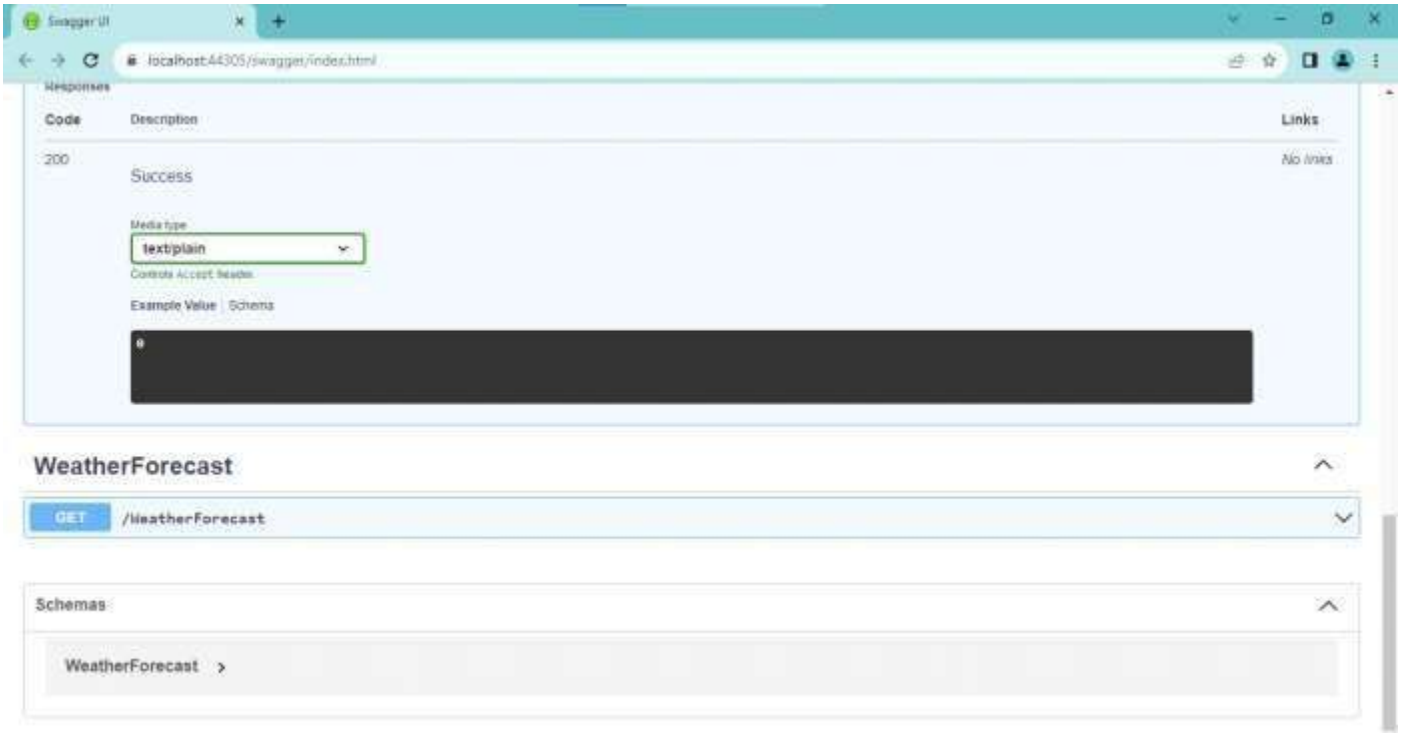
Step 23: Select **Try it out** option & enter any 2 numbers. Then click on **Execute**.



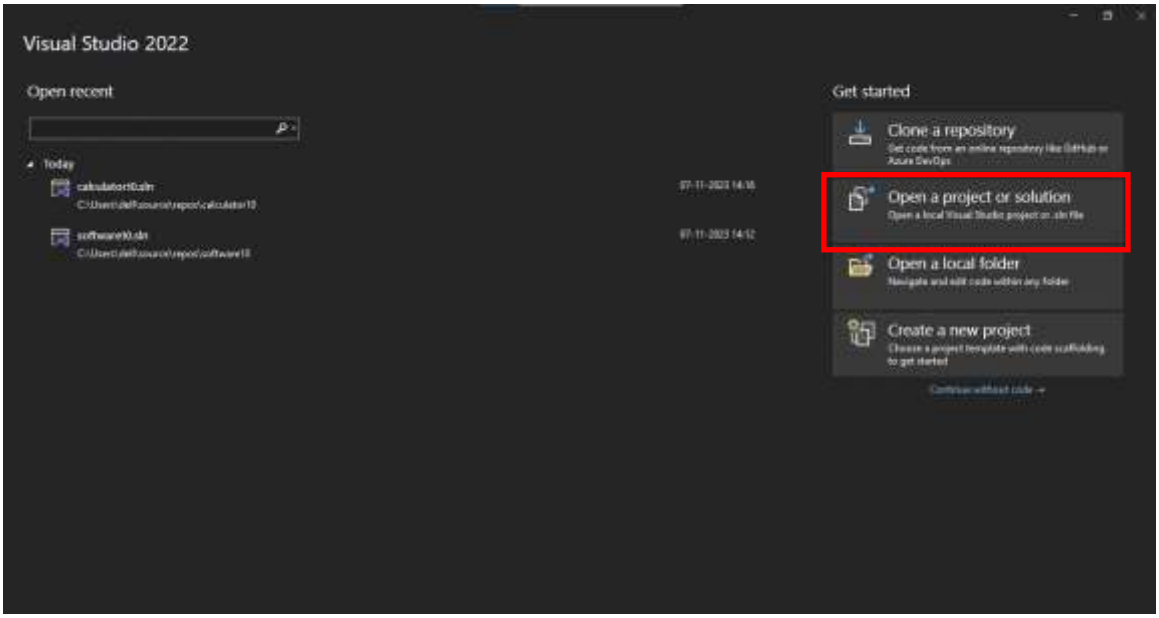
Step 24: Observe the result.



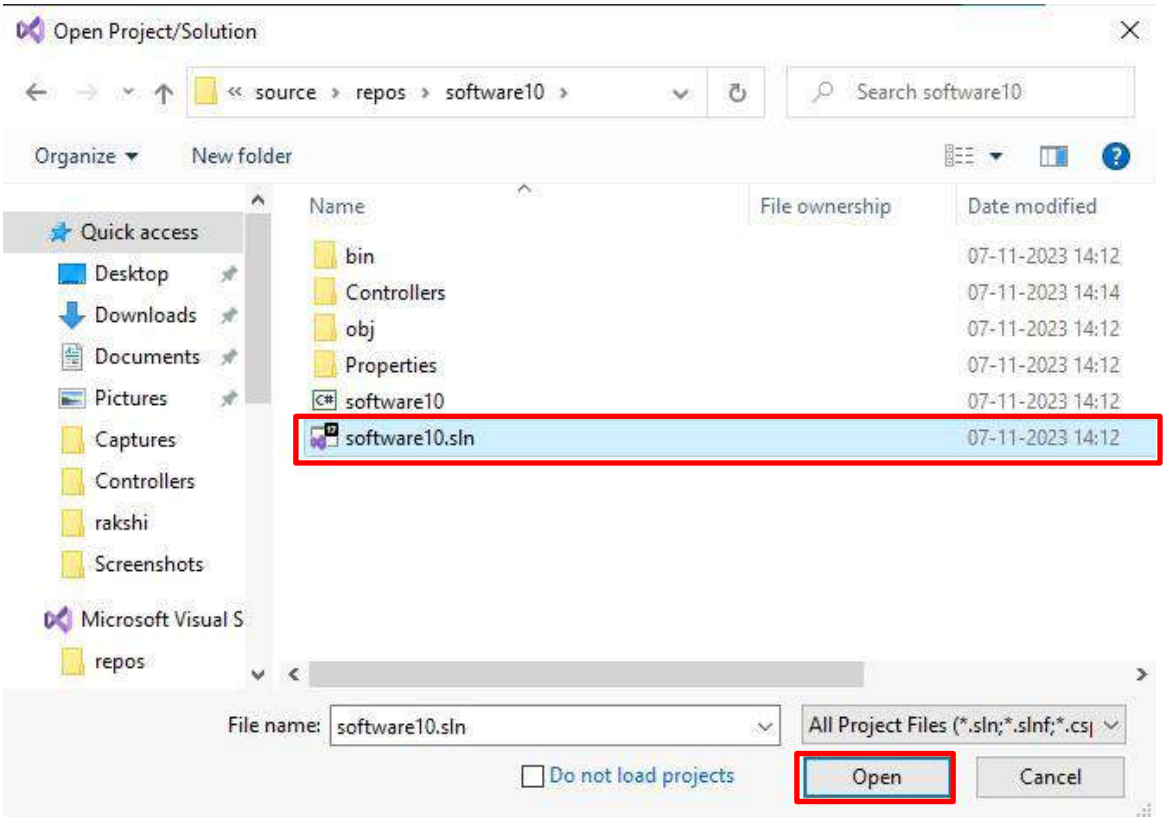
Step 25: Code execution is successful.



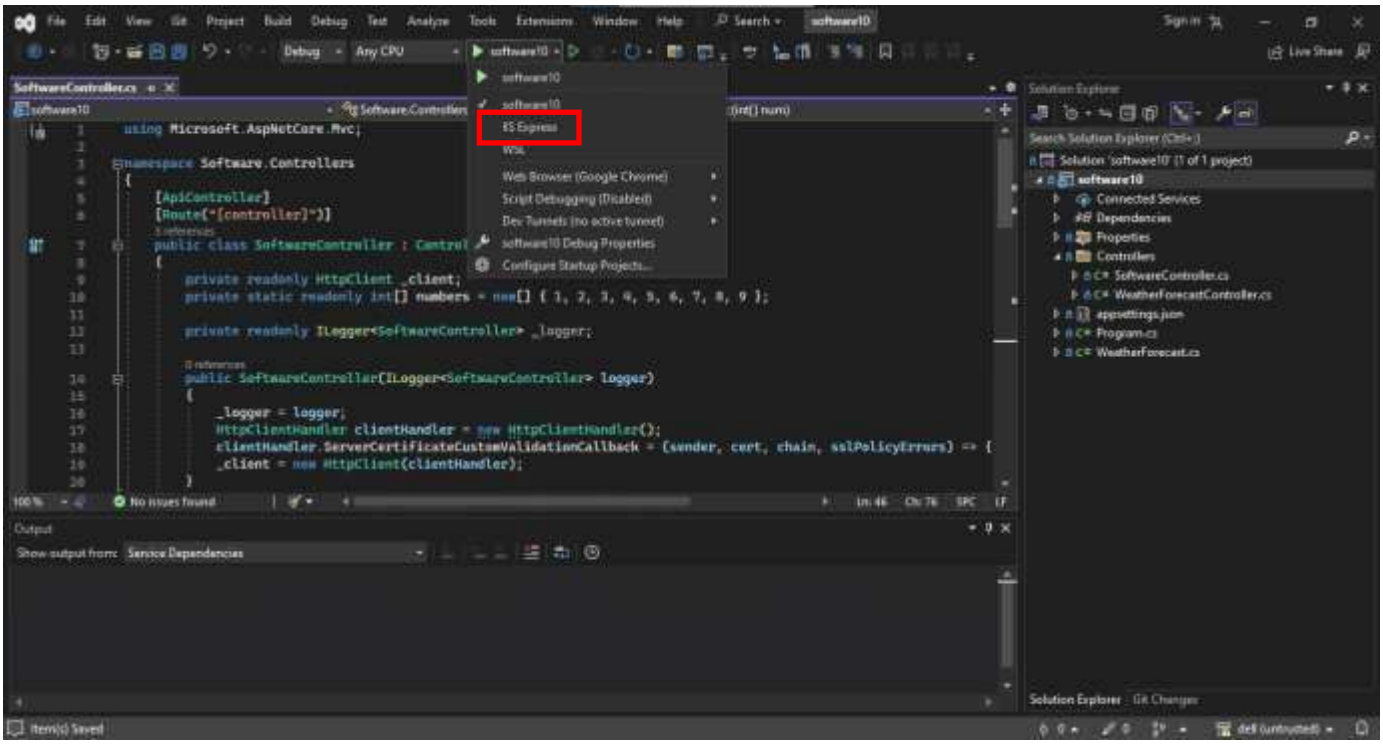
Step 26: Now one more time open the Microsoft Visual Studios & click on **'Open a Project Solution'**.



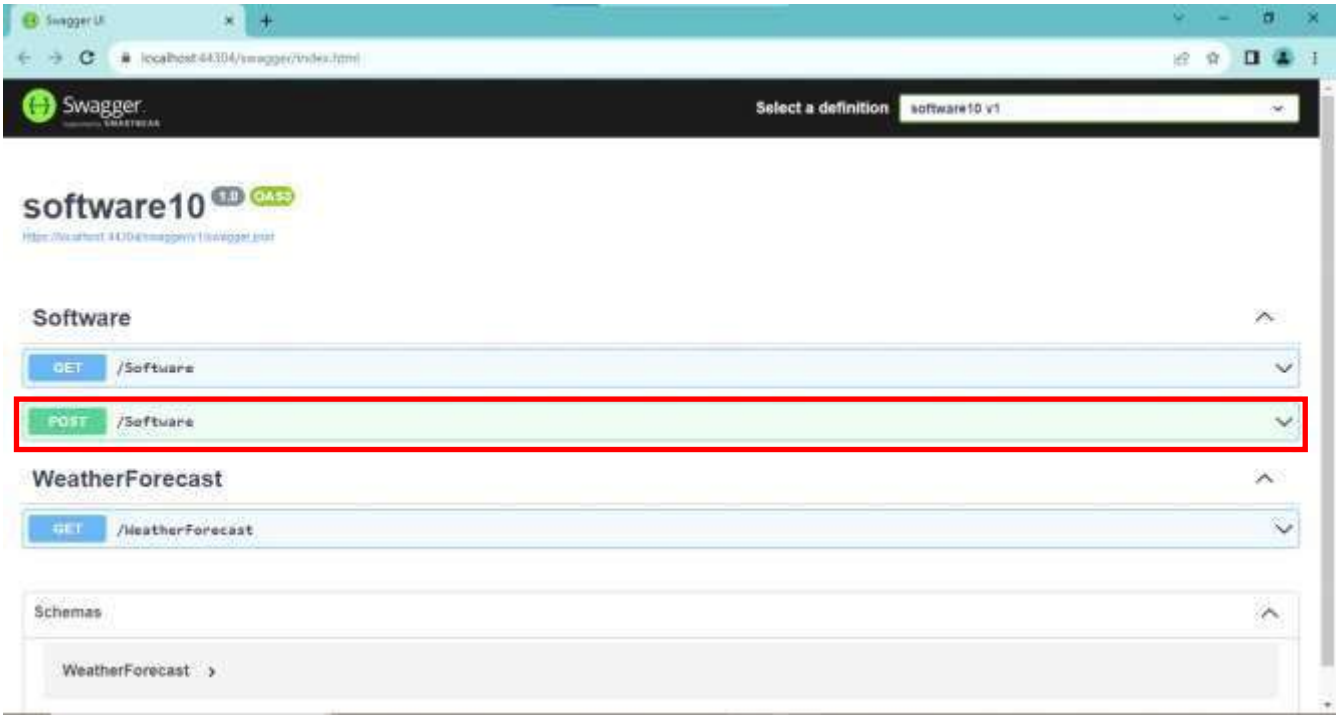
Step 27: Now open the **software10.sln** file.



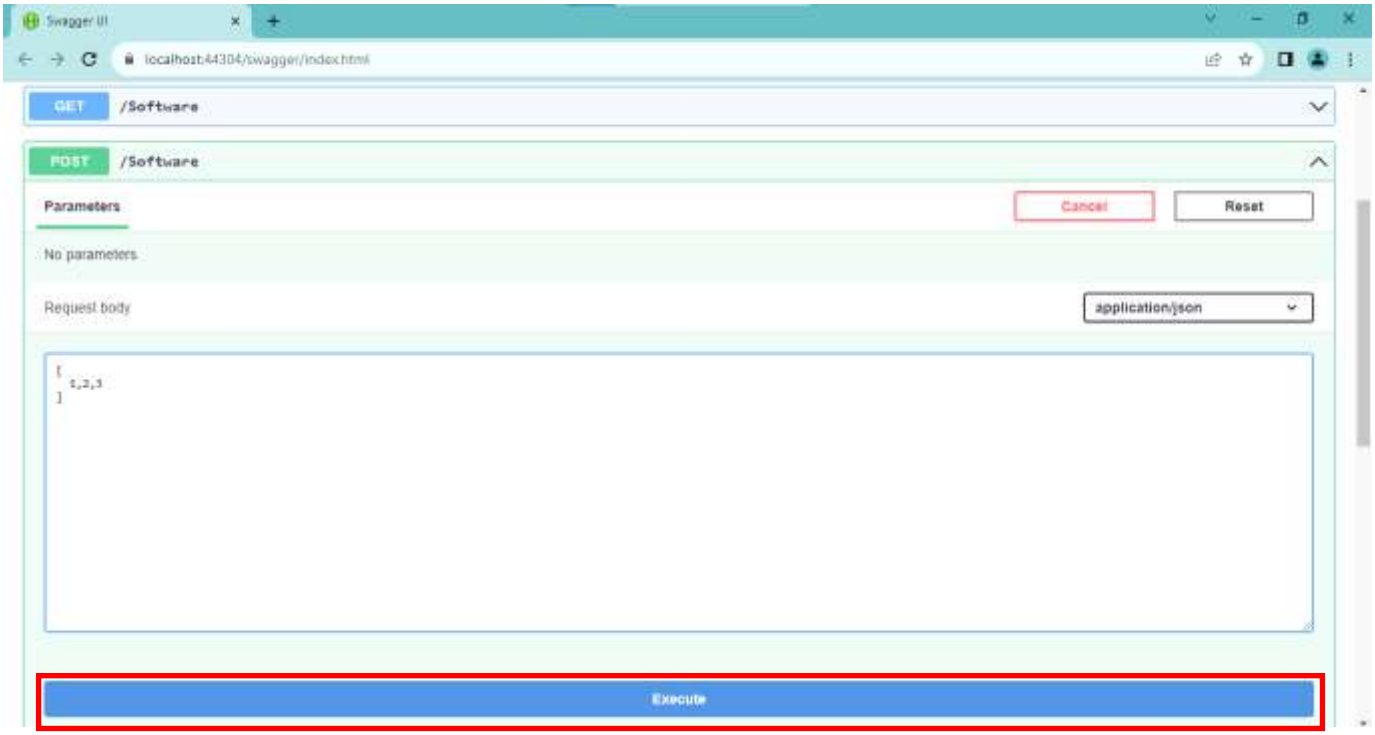
Step 28: Change the port number from the existing localhost port number, then select **IIS Express** for running the code.



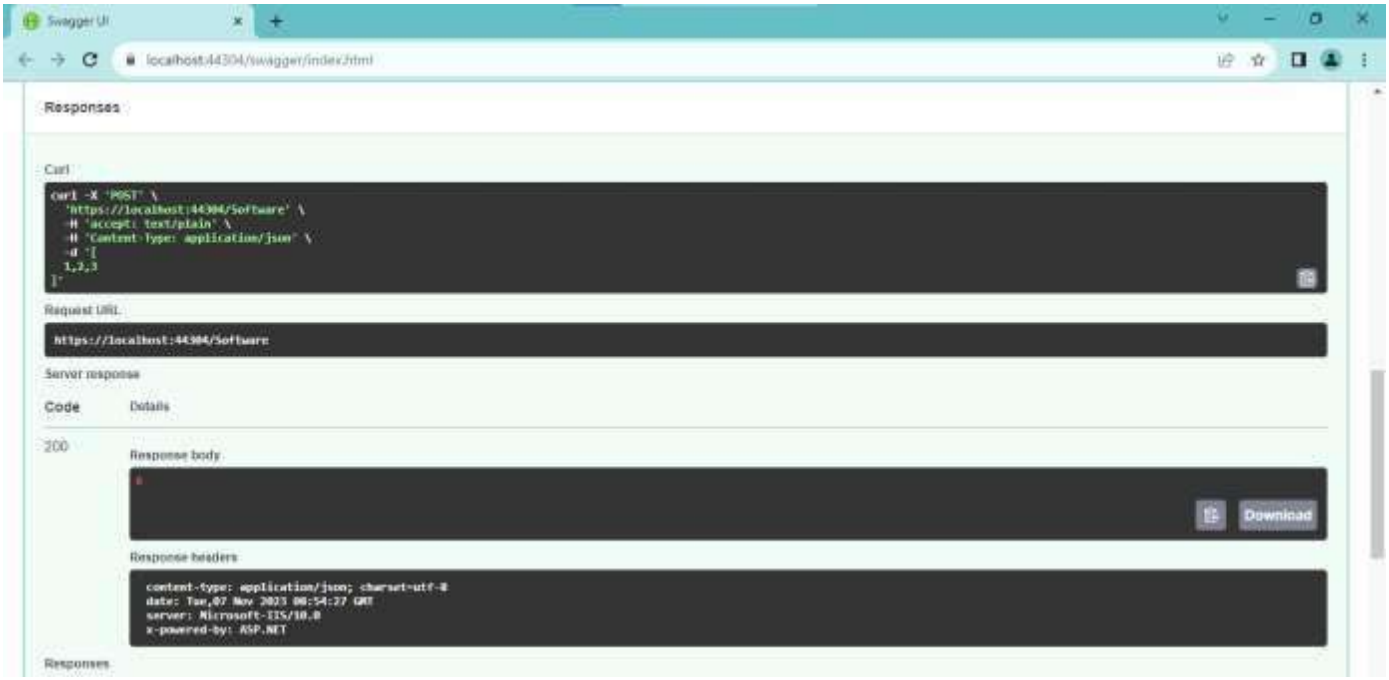
Step 29: Now click on **Software** here.



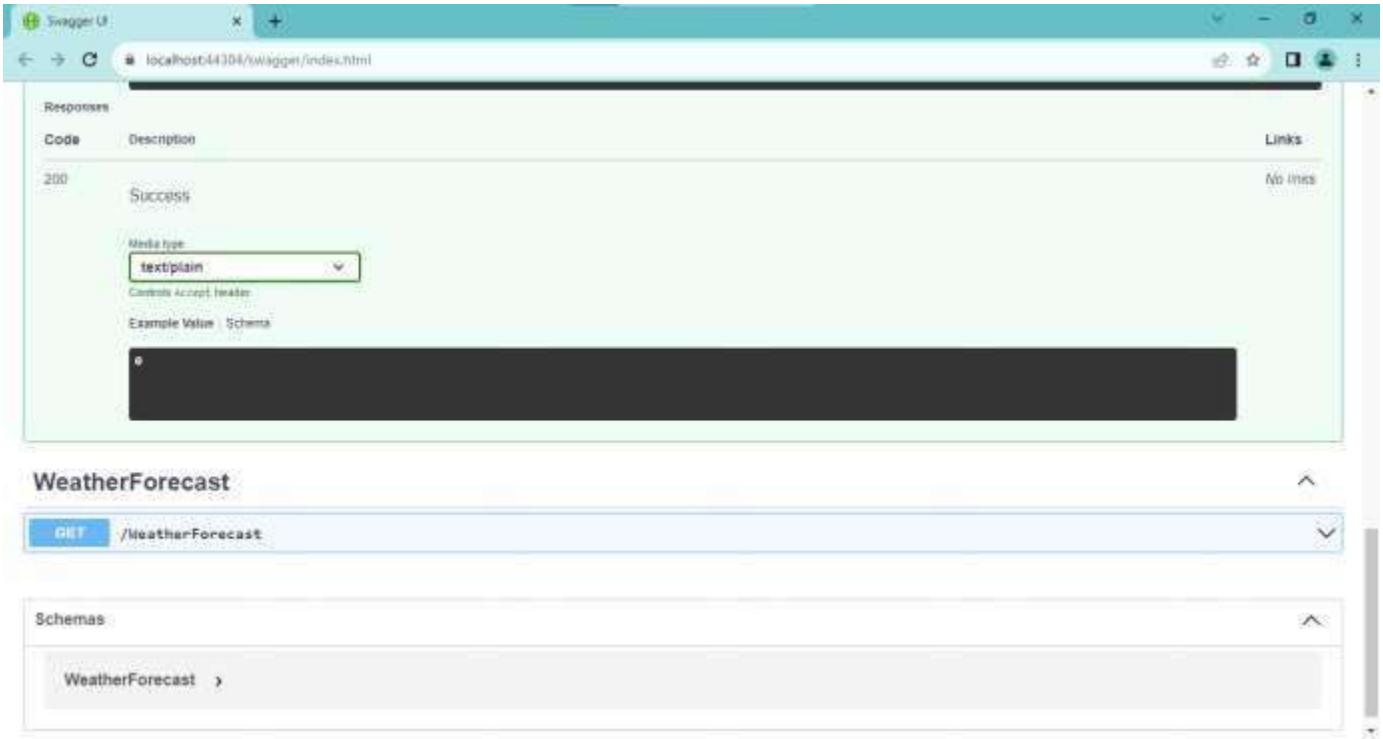
Step 30: Select **Try it out** option & enter any 3 numbers. Then click on **Execute**.



Step 31: Observe the result.

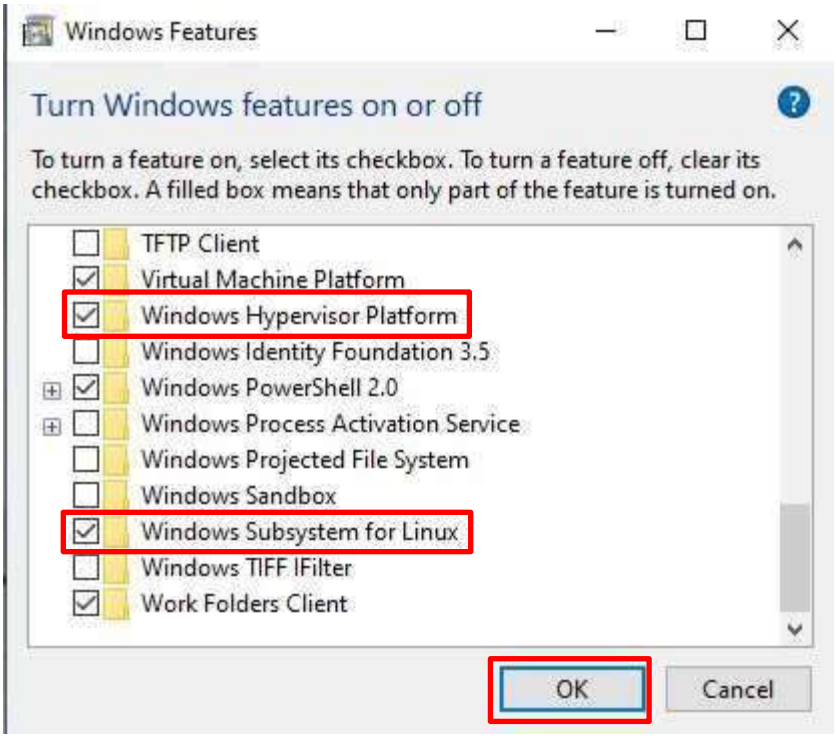


Step 32: Code execution is successful.

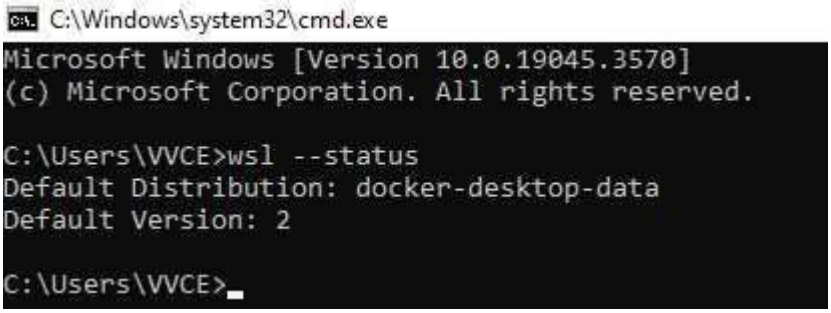


2. Build docker image with docker file.

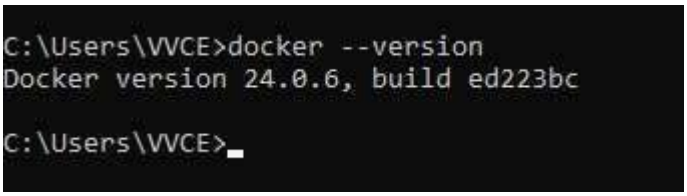
Step 1: Go to Taskbar search for **Turn Windows features on or off** & enable the 2 checkboxes here (**Windows Hypervisor Platform, Windows Subsystem for Linux)**. Then click on **OK.**



Step 2: Open Command Prompt and enter the command as **wsl –status** to check the status.

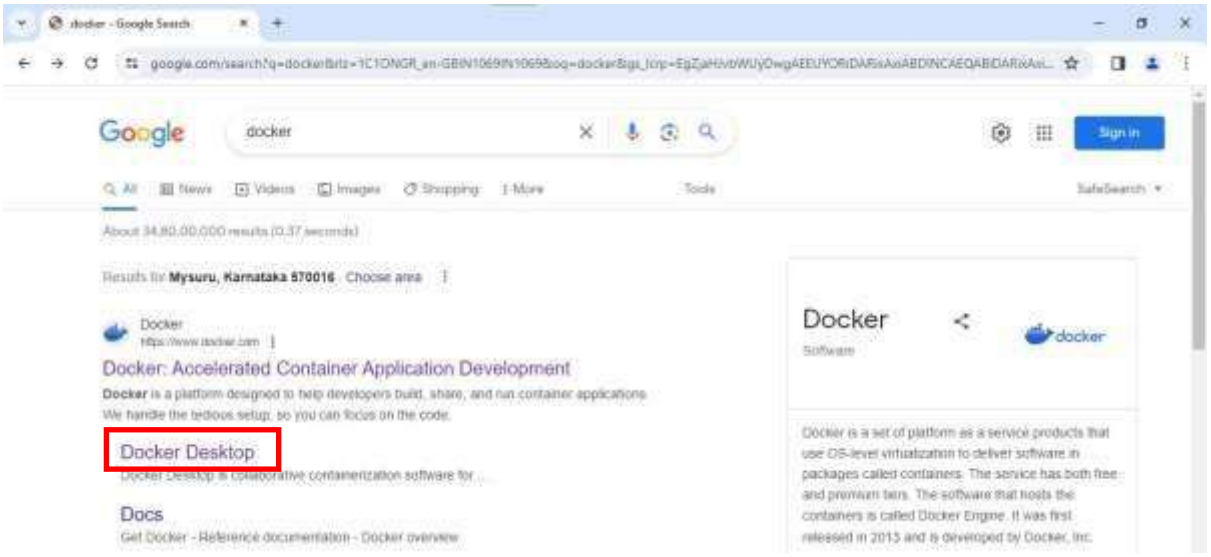


Step 3: Now enter the command as **docker –version** to check the version.

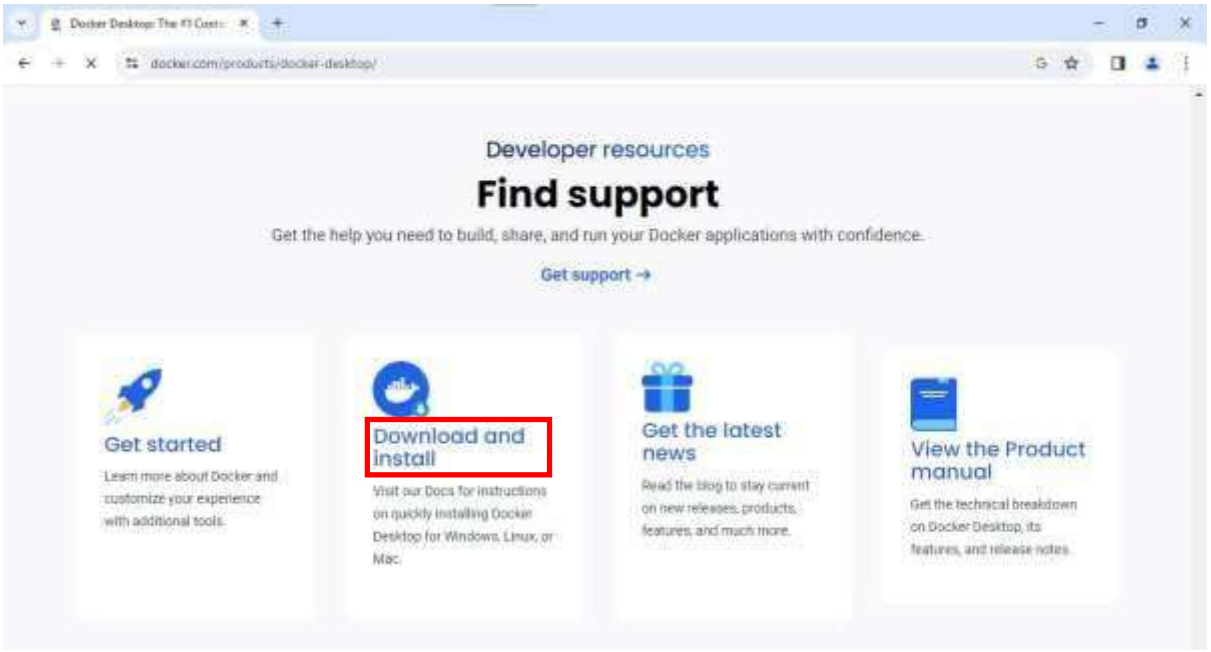


Installation of Docker Desktop

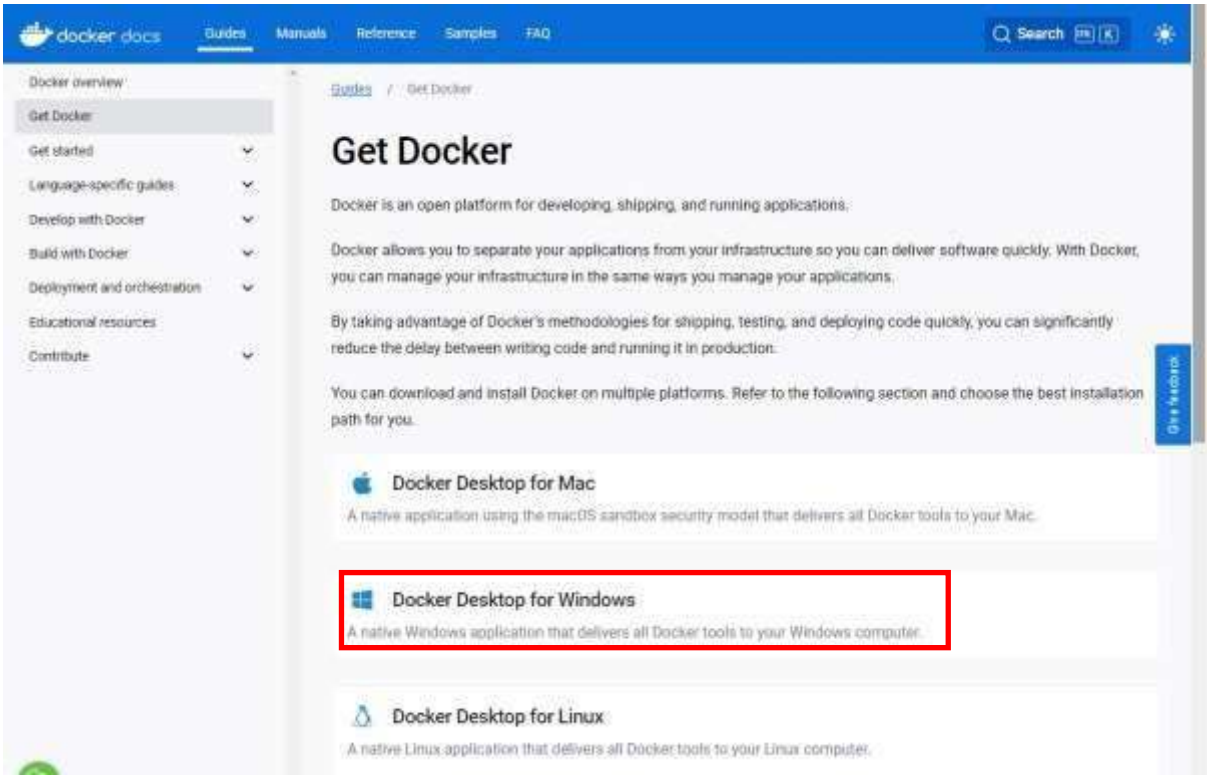
Go to Web Browser Search for Docker Click on the second link below.



Scroll down & select **Download and Install** option here.



Now click on **Docker Desktop for Windows** here.



Then click on **Docker Desktop for Windows** for download.

Step 4: Now create a new folder named **Docker_files** in the Desktop.

Step 5: Go to the folder path and enter cmd as shown below.

Step 6: Now enter the command as **docker –version** to check the version & enter the command as **docker images** to view the already created docker images.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\VVCE\Desktop\Docker_files>docker --version
Docker version 24.0.6, build ed223bc

C:\Users\VVCE\Desktop\Docker_files>docker images
REPOSITORY                          TAG       IMAGE ID       CREATED         SIZE
thirdapp                            latest    7a0ee2c58104   29 hours ago    1.02GB
secondapp                           latest    6c007e3cebab   4 days ago      1.02GB
calculator30                        dev       3598ea73cf35   4 days ago      208MB
software14                          dev       5662c71cee81   4 days ago      208MB
calculator18                        dev       7775fff32e5d   4 days ago      208MB
calculator90                        dev       b09e7c328d91   4 days ago      208MB
software92                          dev       fc92c31f7741   4 days ago      208MB
sofware                             dev       f73caddf2a7c   4 days ago      208MB
calculator34                        dev       d3115d556df8   4 days ago      208MB
calculator14                        dev       12d7495fa256   4 days ago      208MB
software30                          dev       8f26f67498b0   4 days ago      208MB
software18                          dev       9729a0be0b0c   4 days ago      208MB
calculator16                        dev       bc0d63a6c8c1   4 days ago      208MB
software16                          dev       f13f74852de4   4 days ago      208MB
mcr.microsoft.com/dotnet/aspnet     6.0       63bd24373d54   6 days ago      208MB
software                            dev       e9e34ceef255   11 days ago     208MB
calculator                          dev       1350b3abbbeb   11 days ago     208MB
calculator15                        dev       3025c4813c1b   11 days ago     208MB
software12                          dev       034e18603adc   11 days ago     208MB
calculator13                        dev       f733c25cf05f   11 days ago     208MB
software122                         dev       4a8d4d31892d   11 days ago     208MB
calculator1                         dev       b7dfd64155fb   11 days ago     208MB
mcr.microsoft.com/dotnet/aspnet     <none>    2335ab8c4f9c   13 days ago     208MB

C:\Users\VVCE\Desktop\Docker_files>
```

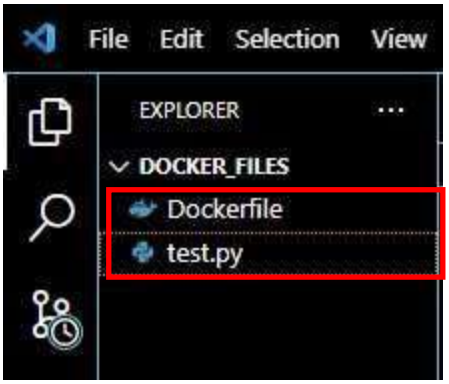
Step 7: Now type **python** to check the installation version of python.

```
Command Prompt - python
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\VVCE>python
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct  2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("welcome")
welcome
>>>
```

Step 8: Now enter the command as **code .** to directly enter into the VS Code.

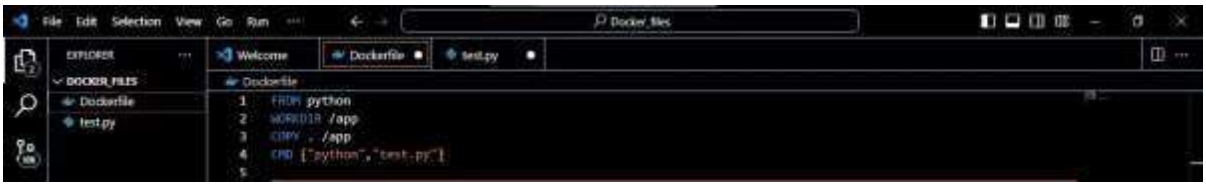
```
C:\Users\VVCE\Desktop\Docker_files>code .

C:\Users\VVCE\Desktop\Docker_files>
```

Step 9: Create 2 files named as **Dockerfile & test.py** as shown below.

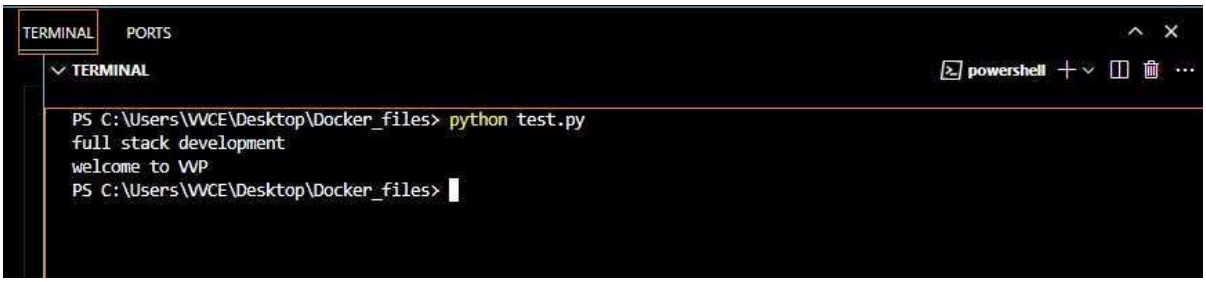


Step 10: Enter the below code to **test.py** file.

Step 11: Enter the below code to **Dockerfile** as well.



Step 12: Go to Terminal New Terminal enter the command as **python test.py** here.



Step 13: Now enter the command as **docker build -t firstapp .** here.

```
TERMINAL  PORTS
TERMINAL                                                    powershell + ⋯
PS C:\Users\VVCE\Desktop\Docker_files> python test.py
full stack development
welcome to VVP
PS C:\Users\VVCE\Desktop\Docker_files> docker build -t firstapp .
[+] Building 4.0s (8/8) FINISHED                                    docker:default
 => [internal] load build definition from Dockerfile                        0.1s
 => => transferring dockerfile: 105B                                        0.0s
 => [internal] load .dockerignore                                           0.1s
 => => transferring context: 2B                                             0.0s
 => [internal] load metadata for docker.io/library/python:latest            3.5s
 => [1/3] FROM docker.io/library/python@sha256:7b8d65a924f596eb65306214f559253c468336bcae09fd57542977456346  0.0s
 => [internal] load build context                                           0.1s
 => => transferring context: 197B                                           0.0s
 => CACHED [2/3] WORKDIR /app                                               0.0s
 => [3/3] COPY . /app                                                       0.1s
 => exporting to image                                                      0.1s
 => => exporting layers                                                     0.1s
 => => writing image sha256:0a789c6c4588e6cbc392d2003d9394c8830469b854fdd1a08afee47dbb84d68a  0.0s
 => => naming to docker.io/library/firstapp                                 0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\VVCE\Desktop\Docker_files>
```

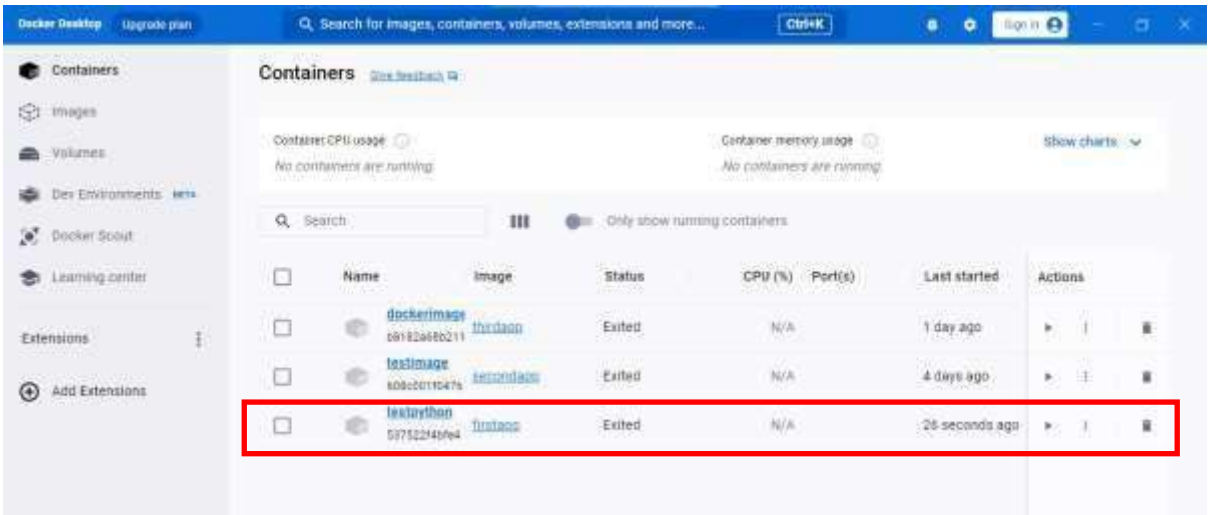
Step 14: In the folder path cmd, enter the command as **docker images** to view created image.

```
C:\Users\VVCE\Desktop\Docker_files>docker images
REPOSITORY                      TAG       IMAGE ID       CREATED          SIZE
firstapp                        latest    0a789c6c4588   46 seconds ago   1.02GB
thirdapp                        latest    7a0ee2c58104   29 hours ago     1.02GB
secondapp                       latest    6c007e3cebab   4 days ago       1.02GB
calculator18                    dev       7775fff32e5d   4 days ago       208MB
software16                      dev       f13f74852de4   4 days ago       208MB
software18                      dev       9729a0be0b0c   4 days ago       208MB
software30                      dev       8f26f67498b0   4 days ago       208MB
software14                      dev       5662c71cee81   4 days ago       208MB
calculator16                    dev       bc0d63a6c8c1   4 days ago       208MB
```

Step 15: Enter the command as **docker run –name testpython firstapp** to run the docker image.

```
C:\Users\VVCE\Desktop\Docker_files>docker run --name testpython firstapp
full stack development
welcome to VVP
```

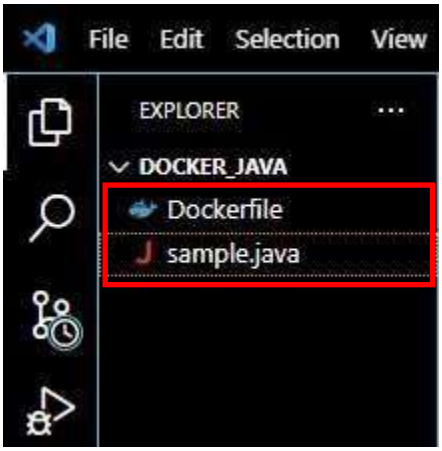
Step 16: Now open the Docker Desktop, your created docker image is displayed here.



Follow the same procedures for running a docker image through java file.

Step 17: Now create a new folder named **Docker_java** in the Desktop.

Step 18: Go to the folder path and enter cmd. And enter the command as **code .** to directly enter into the VS Code. Now create 2 files named as **Dockerfile & sample.java** as shown below.



Step 19: Enter the below code to **sample.java** file.

Step 20: Also enter the below code to **Dockerfile** here.

Step 21: Go to Terminal → New Terminal → enter the command as **javac sample.java** & **java sample** here.

```
TERMINAL  PORTS
TERMINAL                                                    powershell
PS C:\Users\VVCE\Desktop\Docker_java> javac sample.java
PS C:\Users\VVCE\Desktop\Docker_java> java sample
Welcome to FSD
VVCE
PS C:\Users\VVCE\Desktop\Docker_java>
```

Step 22: Enter the command as **docker build -t secondjavaapp .** to build a docker image.

```
PS C:\Users\VVCE\Desktop\Docker_java> docker build -t secondjavaapp .
[+] Building 73.6s (9/9) FINISHED                                          docker:default
 => [internal] load .dockerignore                                                   0.1s
 => => transferring context: 2B                                                     0.0s
 => [internal] load build definition from Dockerfile                                0.1s
 => => transferring dockerfile: 122B                                                0.0s
 => [internal] load metadata for docker.io/library/openjdk:latest                   3.7s
 => [1/4] FROM docker.io/library/openjdk@sha256:9b448de897d211c9e0ec635a485650aed6e28d4eca1efbc34940560a48  61.8s
 => => resolve docker.io/library/openjdk@sha256:9b448de897d211c9e0ec635a485650aed6e28d4eca1efbc34940560a480  0.0s
 => => sha256:9b448de897d211c9e0ec635a485650aed6e28d4eca1efbc34940560a480b3f1f 1.04kB / 1.04kB  0.0s
 => => sha256:fe05457a5e9b9403f8e72eeba507ae80a4237d2d2d3f219fa62ceb128482a9ee 954B / 954B      0.0s
 => => sha256:71260f256d19f4ae5c762601e5301418d2516ca591103b1376f063be0b7ba056 4.46kB / 4.46kB  0.0s
 => => sha256:197c1adcd755131915cd019bdd58658d44445b3638f65449932c18ee39b6047c 44.56MB / 44.56MB  22.6s
 => => sha256:95a27dbe0150755fca4304b4afd0d7d6dd6a40ede6fdb30da8568e9e8cdf23a9 188.74MB / 188.74MB  53.9s
 => => sha256:57b698b7af4b18900b53c768746b1dfb603dfb9aec1eea328fdac86d37001e2a 12.26MB / 12.26MB  8.9s
 => => extracting sha256:197c1adcd755131915cd019bdd58658d44445b3638f65449932c18ee39b6047c  13.9s
 => => extracting sha256:57b698b7af4b18900b53c768746b1dfb603dfb9aec1eea328fdac86d37001e2a   3.7s
 => => extracting sha256:95a27dbe0150755fca4304b4afd0d7d6dd6a40ede6fdb30da8568e9e8cdf23a9   7.5s
 => [internal] load build context                                                   0.1s
 => => transferring context: 800B                                                   0.0s
 => [2/4] WORKDIR /app                                                              5.0s
 => [3/4] COPY . /app                                                               0.1s
 => [4/4] RUN javac sample.java                                                     1.8s
 => exporting to image                                                              0.7s
 => => exporting layers                                                             0.7s
 => => writing image sha256:3736e2c68a1e8e83a2ea7e8da2f637497dc8096f1aedb7f6e1a54acc4ae4ad3f  0.0s
 => => naming to docker.io/library/secondjavaapp                                    0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\VVCE\Desktop\Docker_java>
```

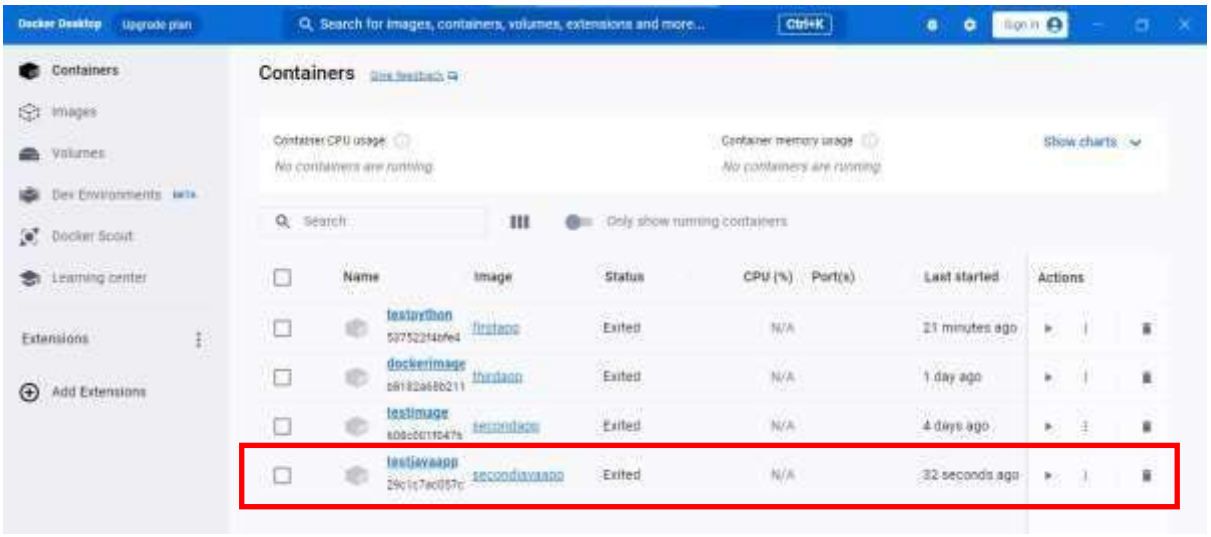
Step 23: In the folder path cmd, enter the command as **docker images** to view the list of created docker images.

```
C:\Users\VVCE\Desktop\Docker_java>docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
secondjavaapp                 latest    3736e2c68a1e   4 minutes ago    470MB
firstapp                      latest    0a789c6c4588   22 minutes ago   1.02GB
thirdapp                      latest    7a0ee2c58104   30 hours ago     1.02GB
secondapp                     latest    6c007e3cebab   4 days ago       1.02GB
software14                    dev       5662c71cee81   4 days ago       208MB
sofware                       dev       f73caddf2a7c   4 days ago       208MB
software92                    dev       fc92c31f7741   4 days ago       208MB
software16                    dev       f13f74852de4   4 days ago       208MB
software30                    dev       8f26f67498b0   4 days ago       208MB
calculator90                  dev       b09e7c328d91   4 days ago       208MB
```

Step 24: Now enter the command as **docker run –name testjavaapp secondjavaapp** to the image.

```
C:\Users\VVCE\Desktop\Docker_java>docker run --name testjavaapp secondjavaapp
Welcome to FSD
VVCE

C:\Users\VVCE\Desktop\Docker_java>
```

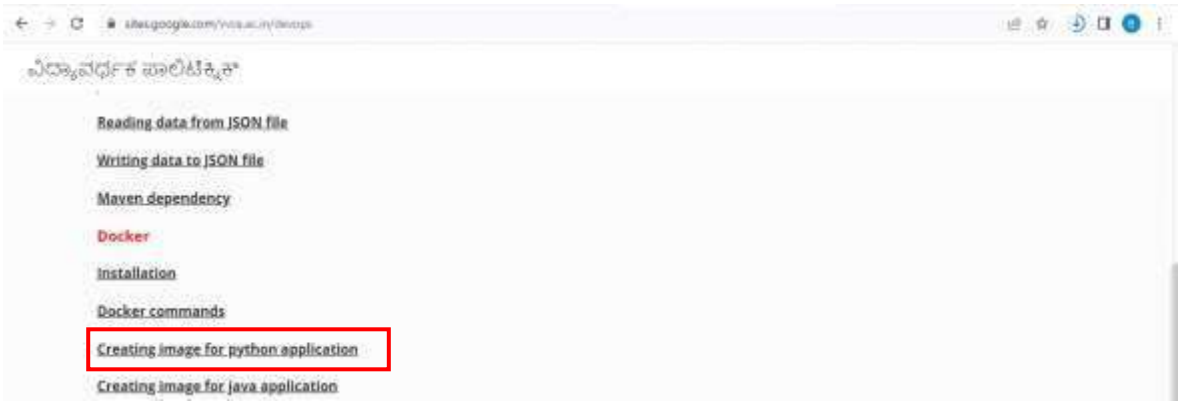
Step 25: You can see the image has been created in Docker Desktop.



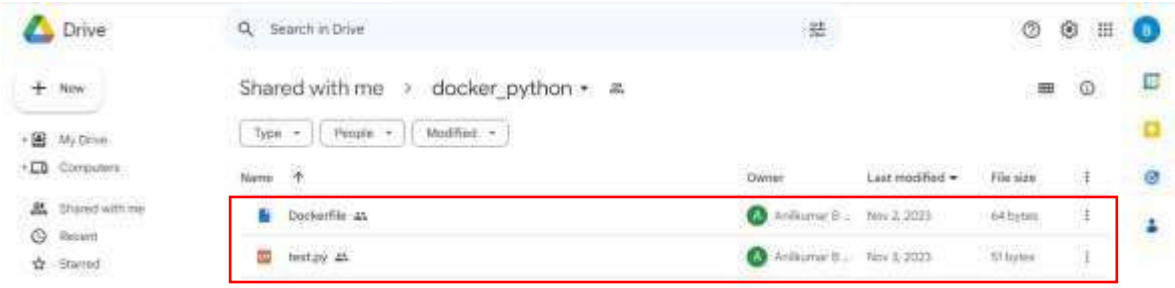
3. Create docker container from docker image & Run the docker container

Step 1: Create a new folder in desktop named as '**docker**' as shown below.

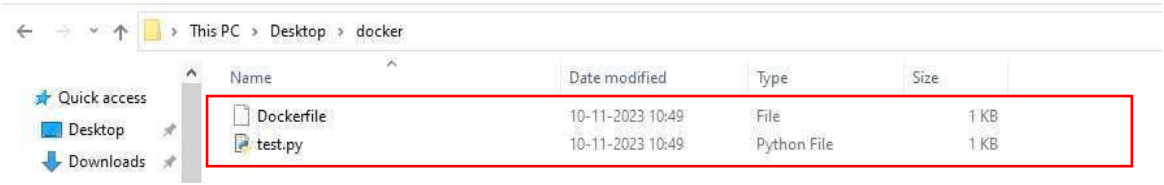
Step 2: Open the official website https://sites.google.com/vvce.ac.in/devops on web browser and click on 'creating image for python application'.



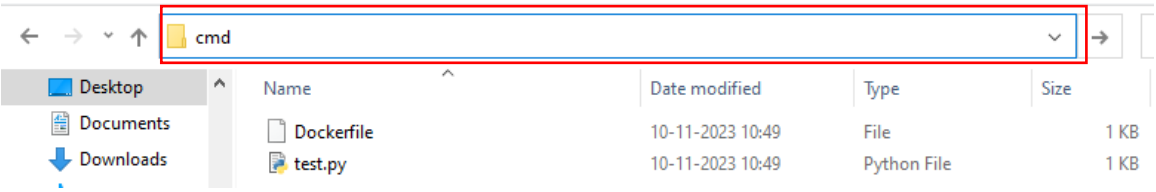
Step 3: Now download the 2 files '**Dockerfile**' & '**test.py**' which are displayed here.



Step 4: Then copy & paste the 2 downloaded files in the new folder created as '**docker**' as shown below.

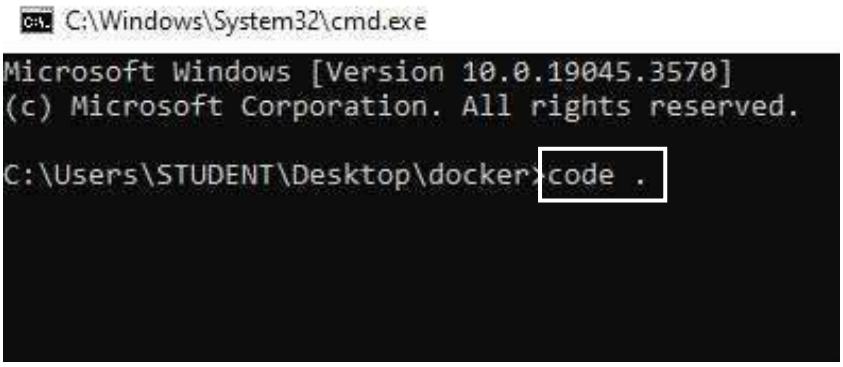


Step 5: Go to Desktop docker cmd as shown below.

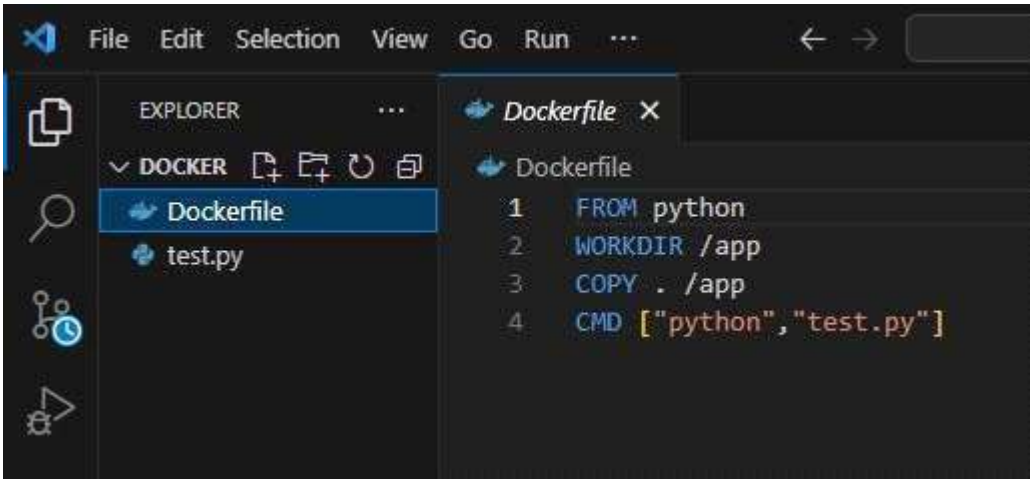


Step 6: Now enter '**code .**' as show below.

Note: This command will directly enter to the VS Code.

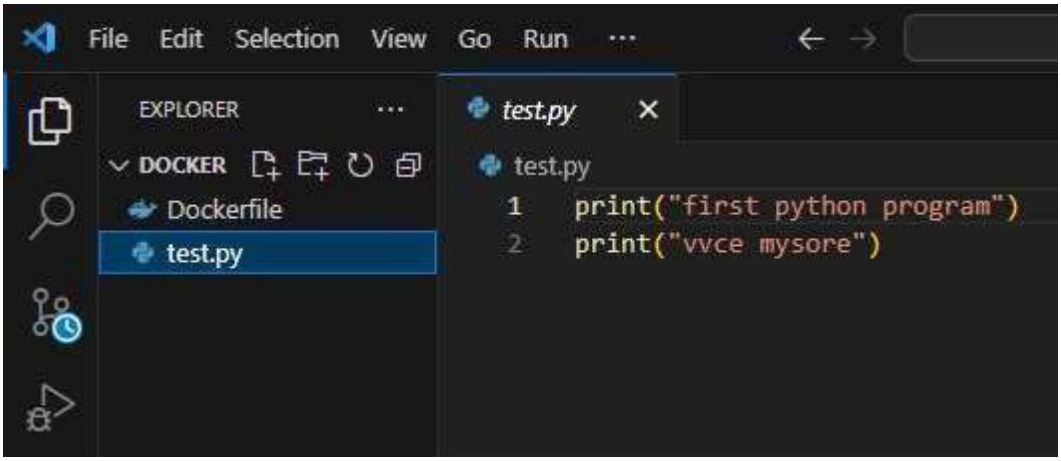


Step 7: Now save the default code in '**Dockerfile**' as shown here.



Step 8: Now save the default code in '**test.py**' as shown here.

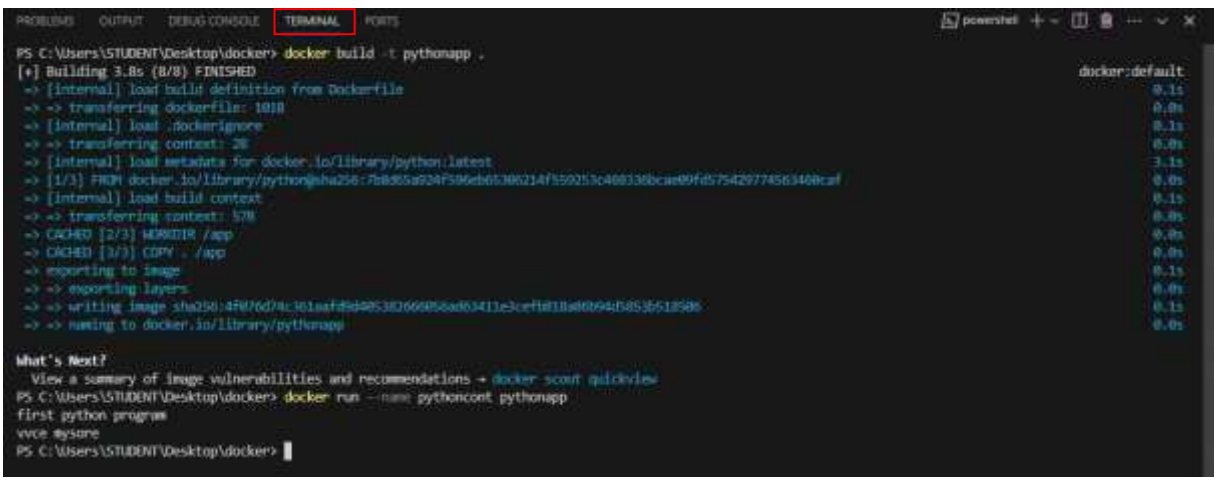
Note: You can change the code if you want.



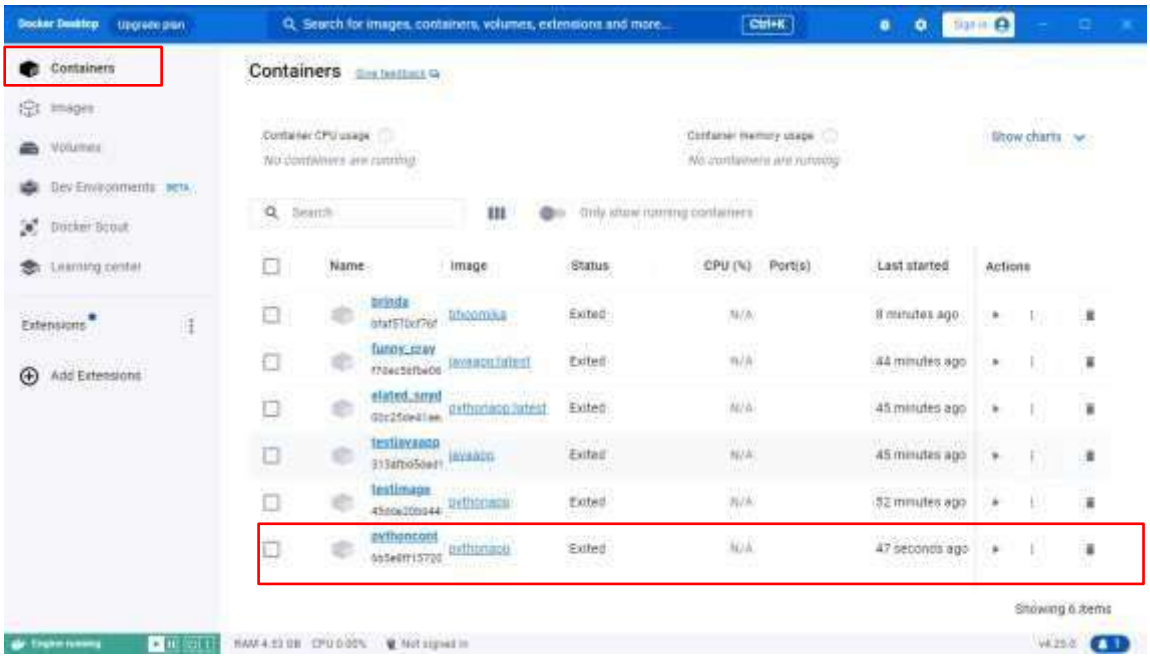
Step 9: Now go to Terminal□ New Terminal □ Enter the command as '**docker build -t pythonapp .**' to run the docker image & '**docker run --name pythoncont pythonapp**' to run the docker container with image.

Note: pythonapp - name of docker image

pythoncont - name of docker container

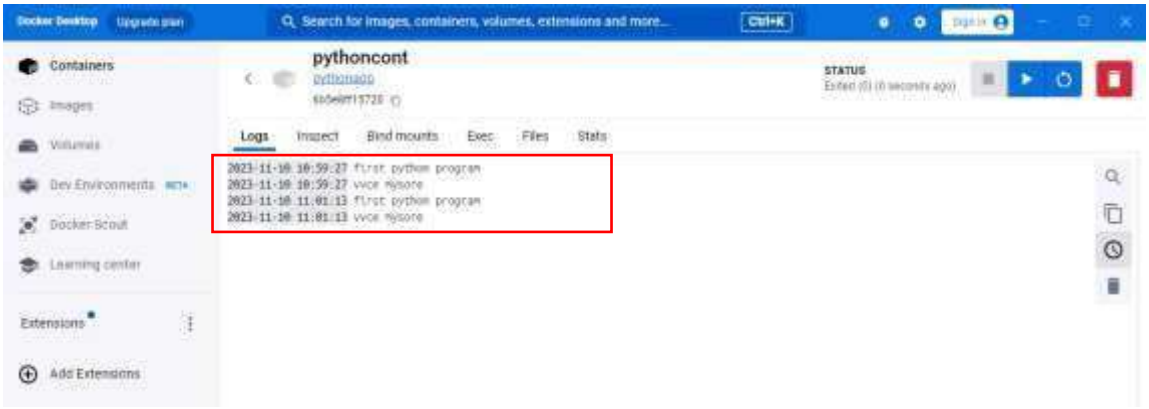


Step 10: Go to Docker Desktop --> Containers --> you can see the created container with image here.



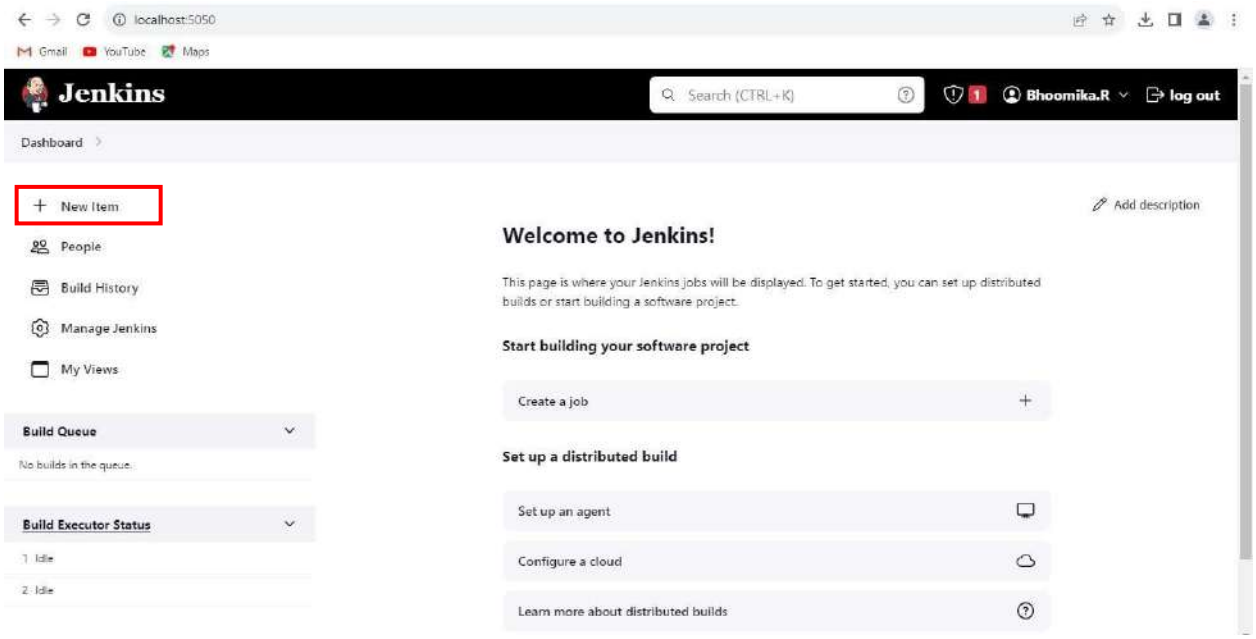
Step 11: Now click on '**Run**' option here.

Step 12: You can see the output here.

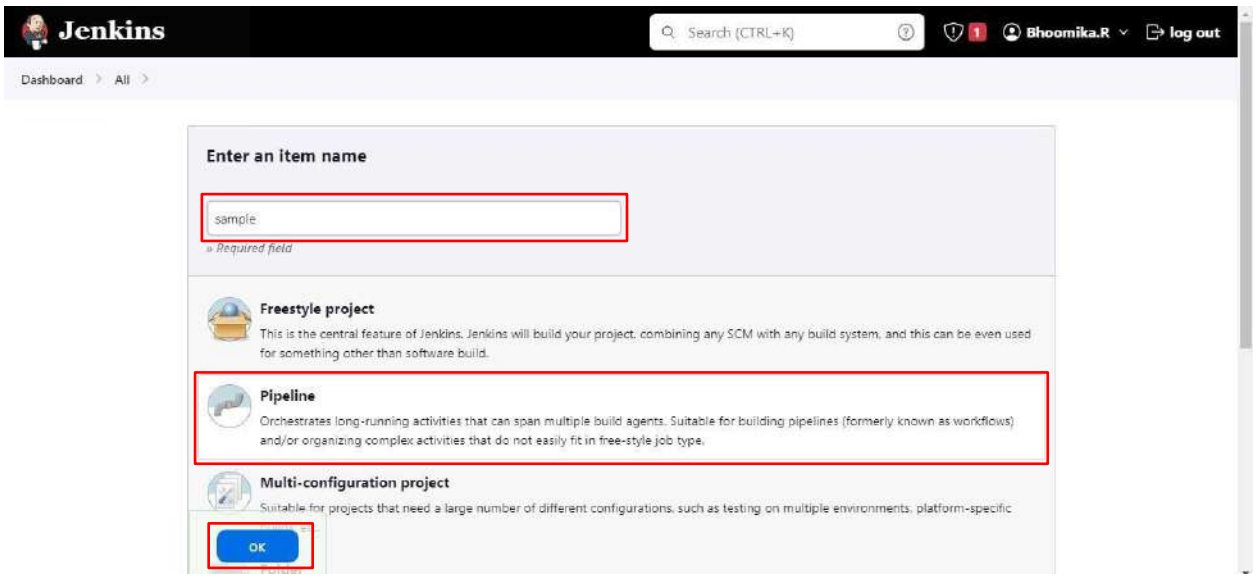


4. Jenkins Pipeline

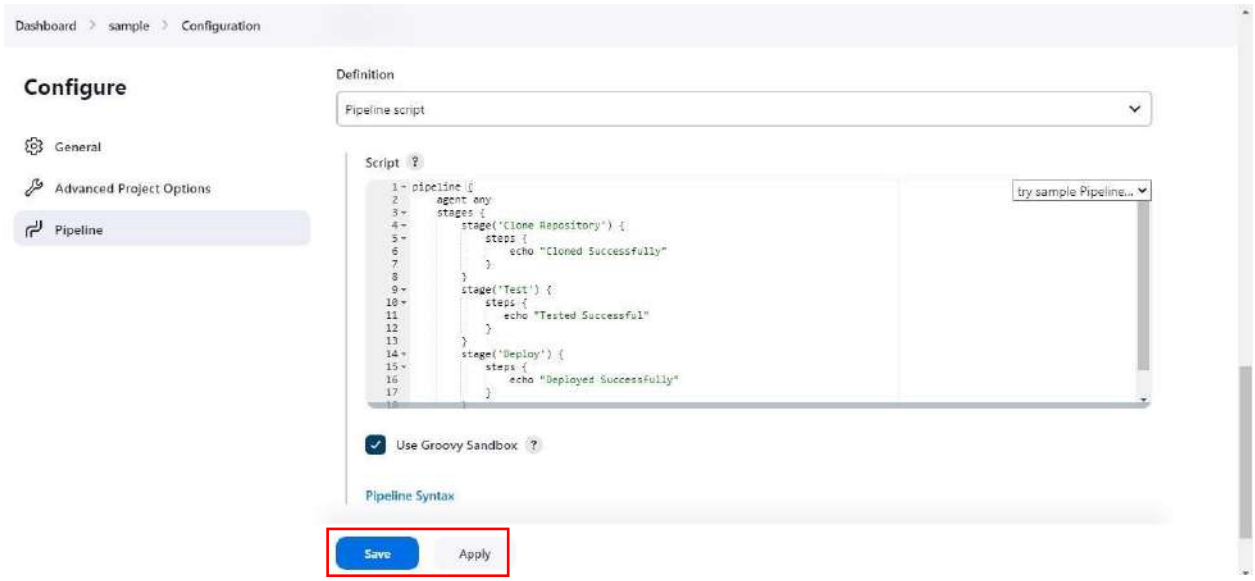
Step 1: Login to your Jenkins Account. Then click on **'New Item'** here.



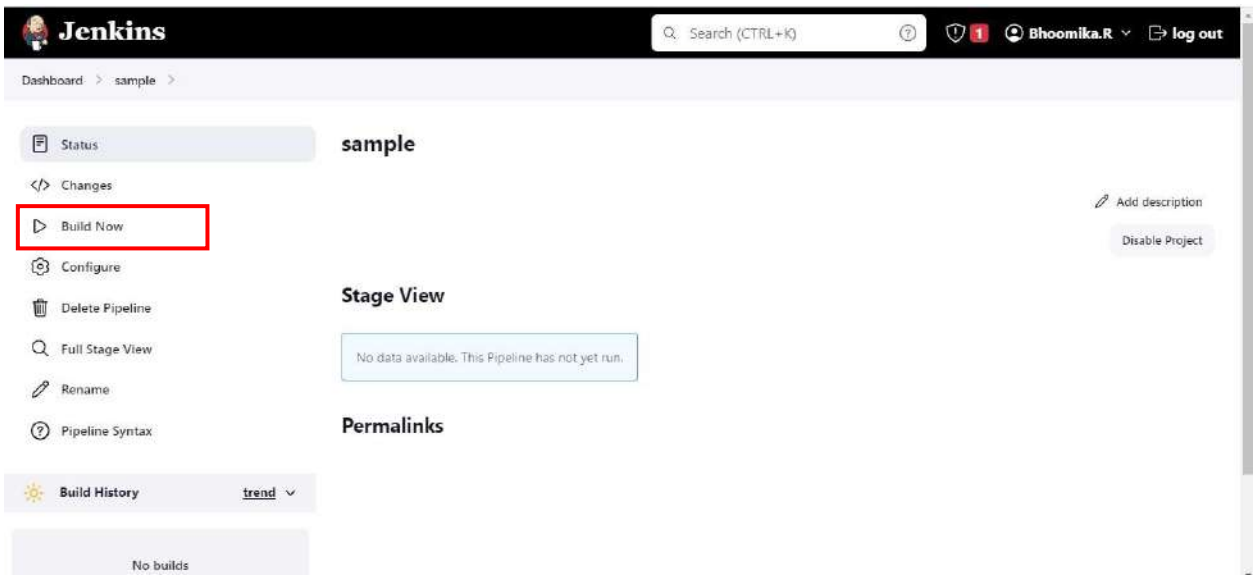
Step 2: Enter an item name here & select **'Pipeline'** field. Then click on **ok**.



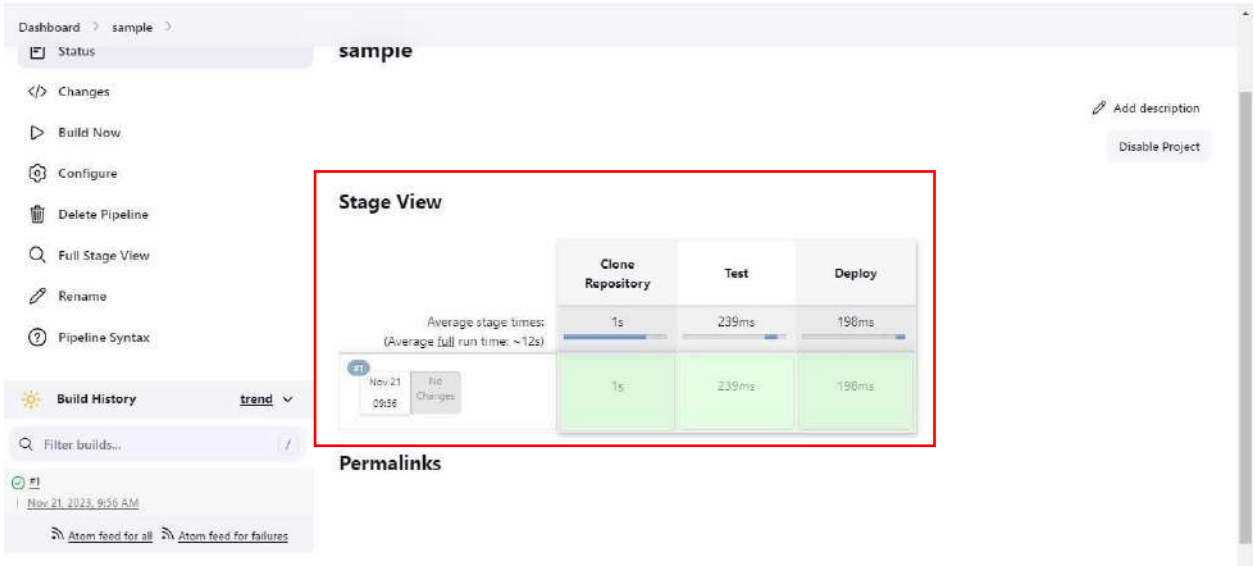
Step 3: Scroll down and add a simple code under Script section. Then click on **Apply & Save**.



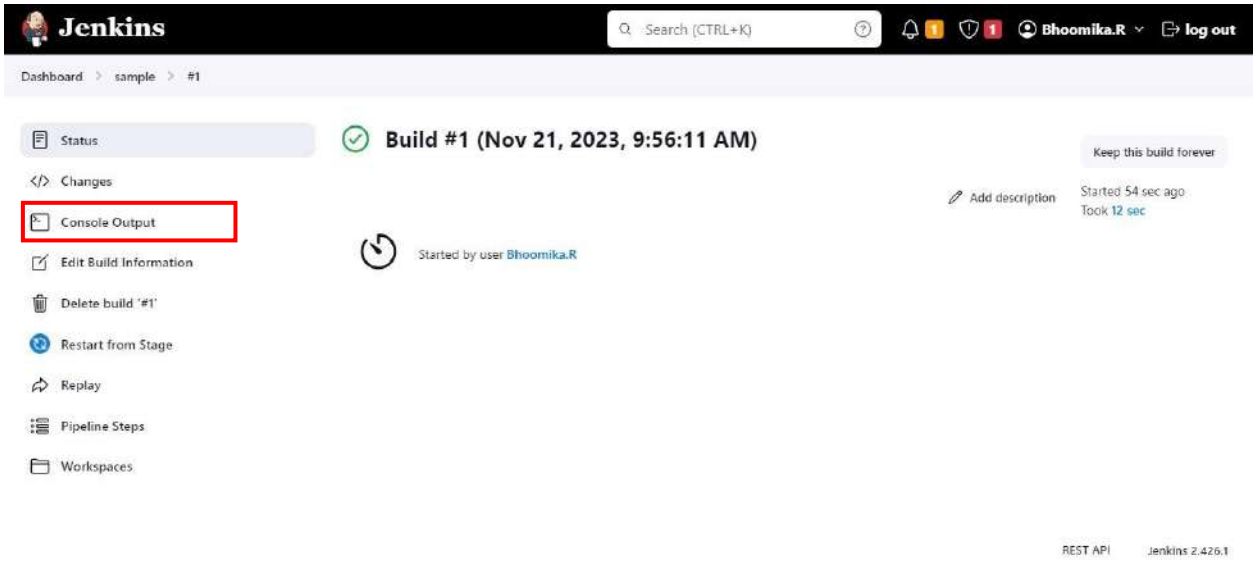
Step 4: Now click on **Build Now** option here.



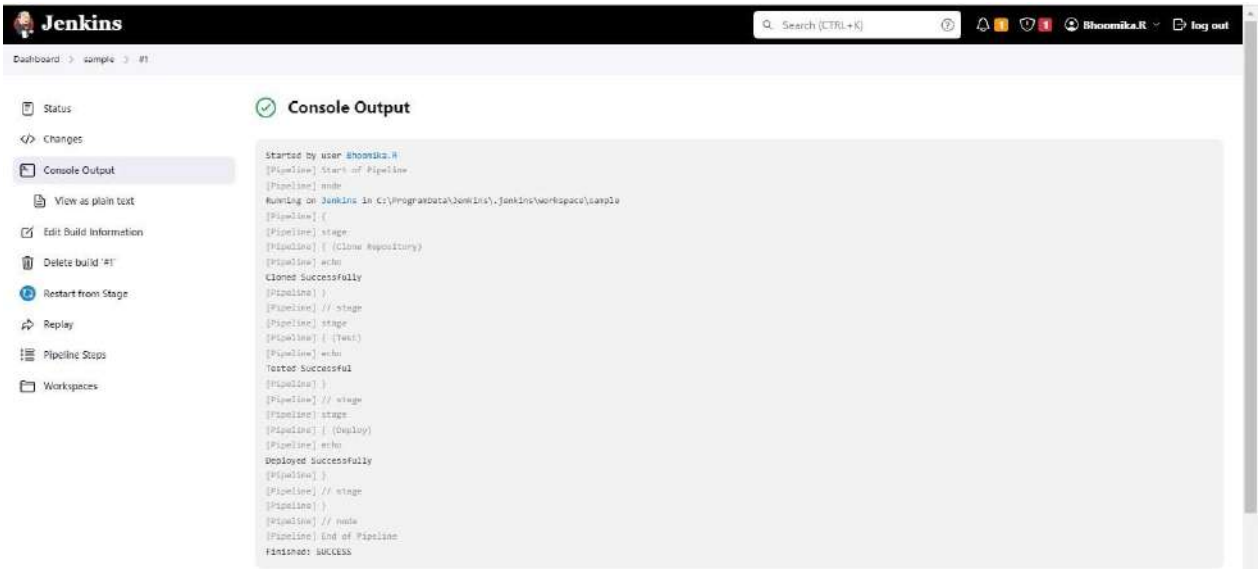
Step 5: Now observe the three stage views (**Clone Repository, Test, Deploy**) as shown below.



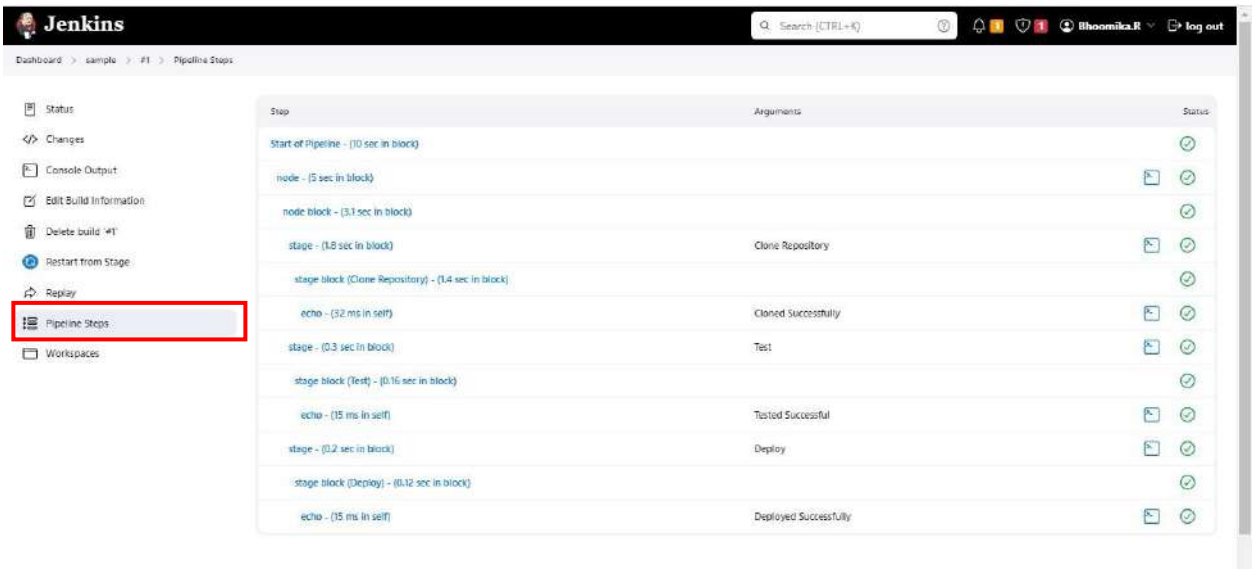
Step 6: Then click on **Console Output** here.



Step 7: Now observe the three stages running successfully.



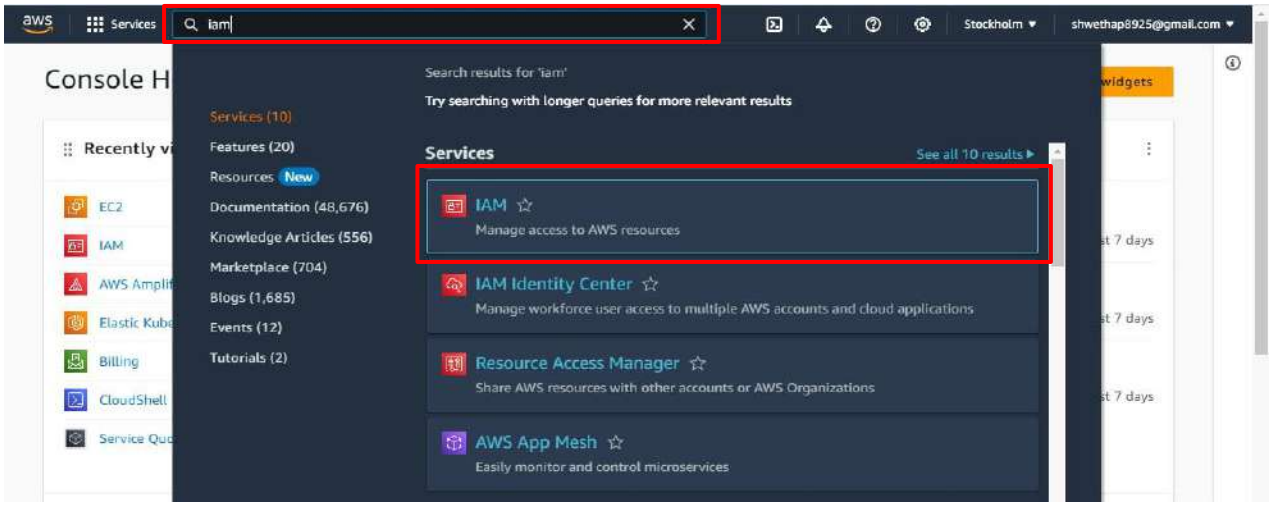
Step 8: Then click on **Pipeline Steps** to view the pipeline steps as shown below.



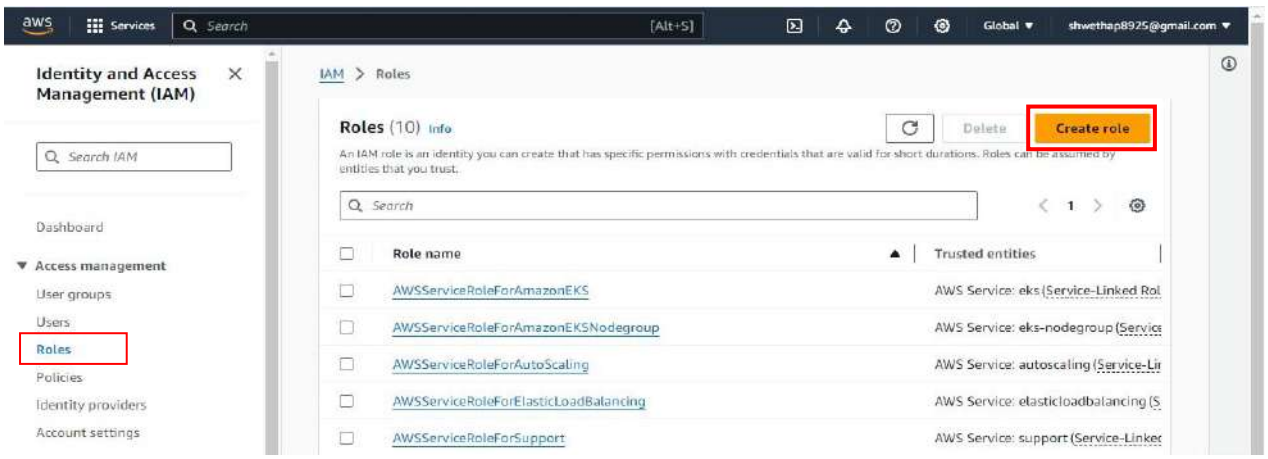
WEEK – 12

1. Kubernetes Configuration on Deployment, Cluster creation, Services & Load Balancer.

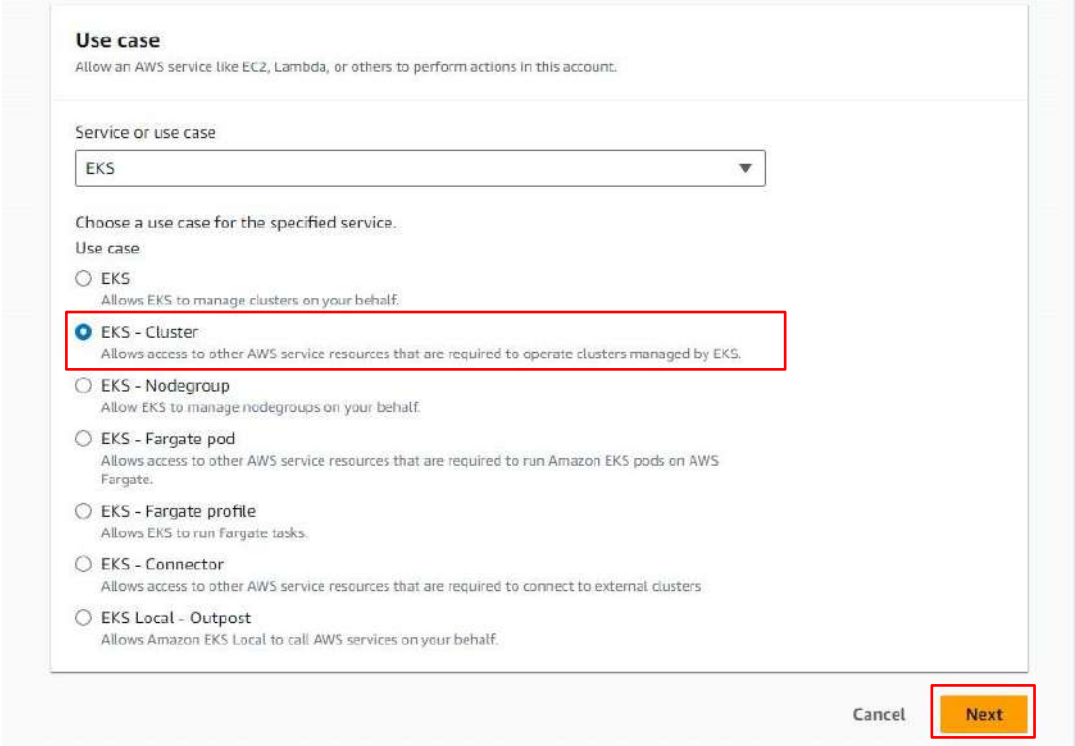
Step 1: Login to your AWS account & search for **'IAM'** Services here.



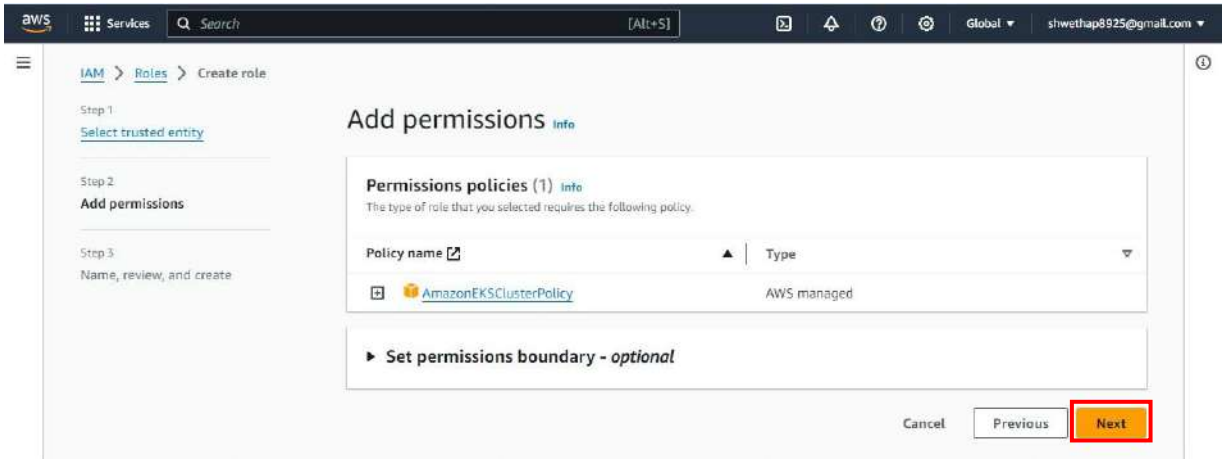
Step 2: Now go to **Roles** □ click on **Create Role** here.



Step 3: Select **EKS Cluster** here & click on **Next**.



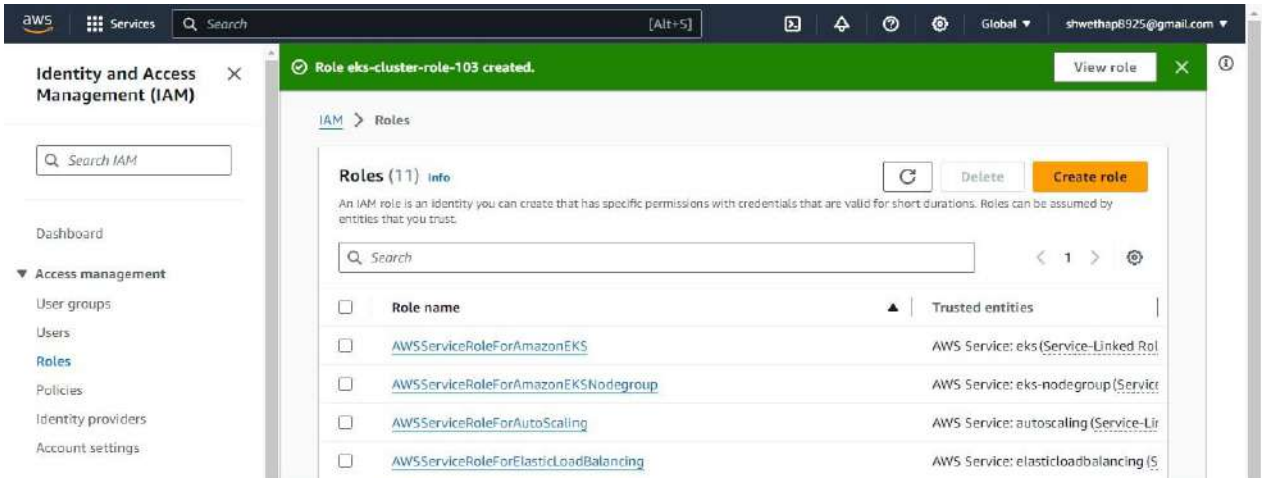
Step 4: Click on **Next.**



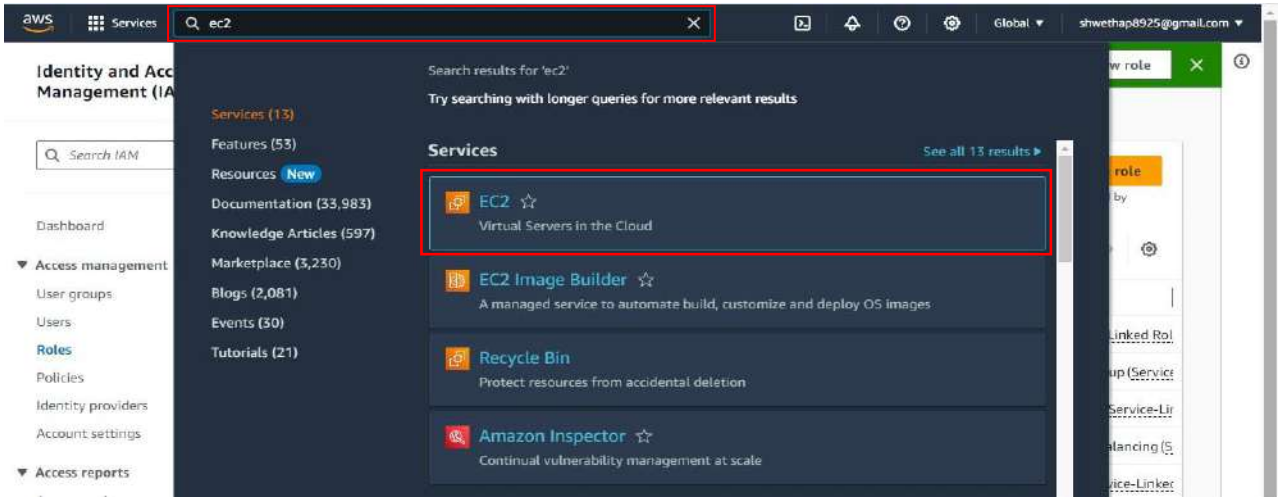
Step 5: Give the role name as **eks-cluster-role-103** as shown here.

Step 6: Now click on **Create Role**.

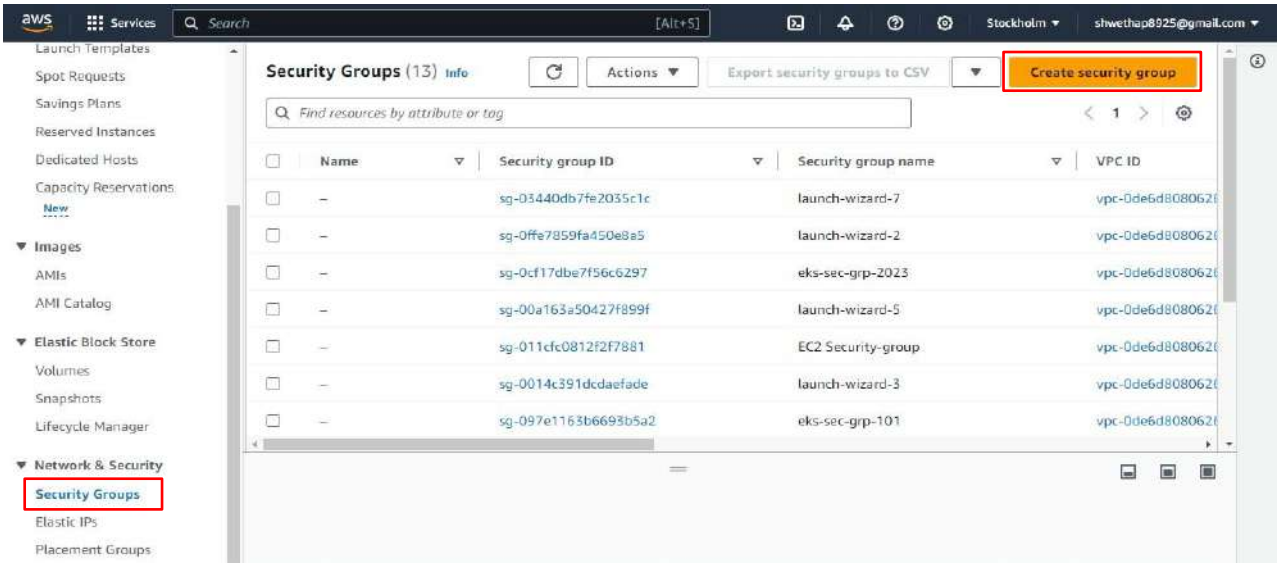
Step 7: Now you can see that your role is successfully created here.



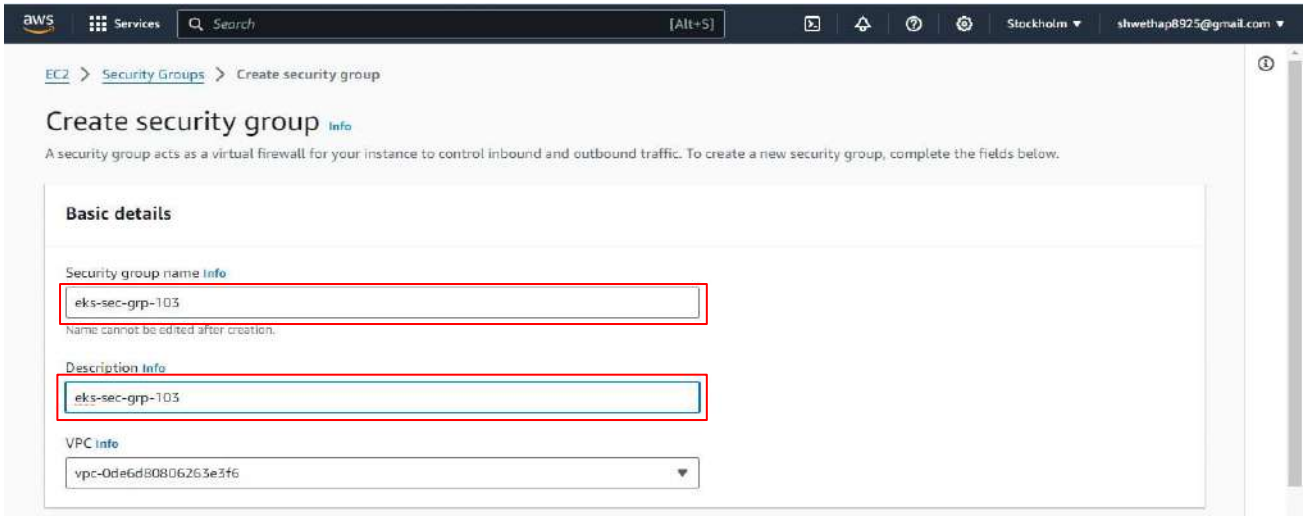
Step 8: Search for **EC 2** and click on first link here.



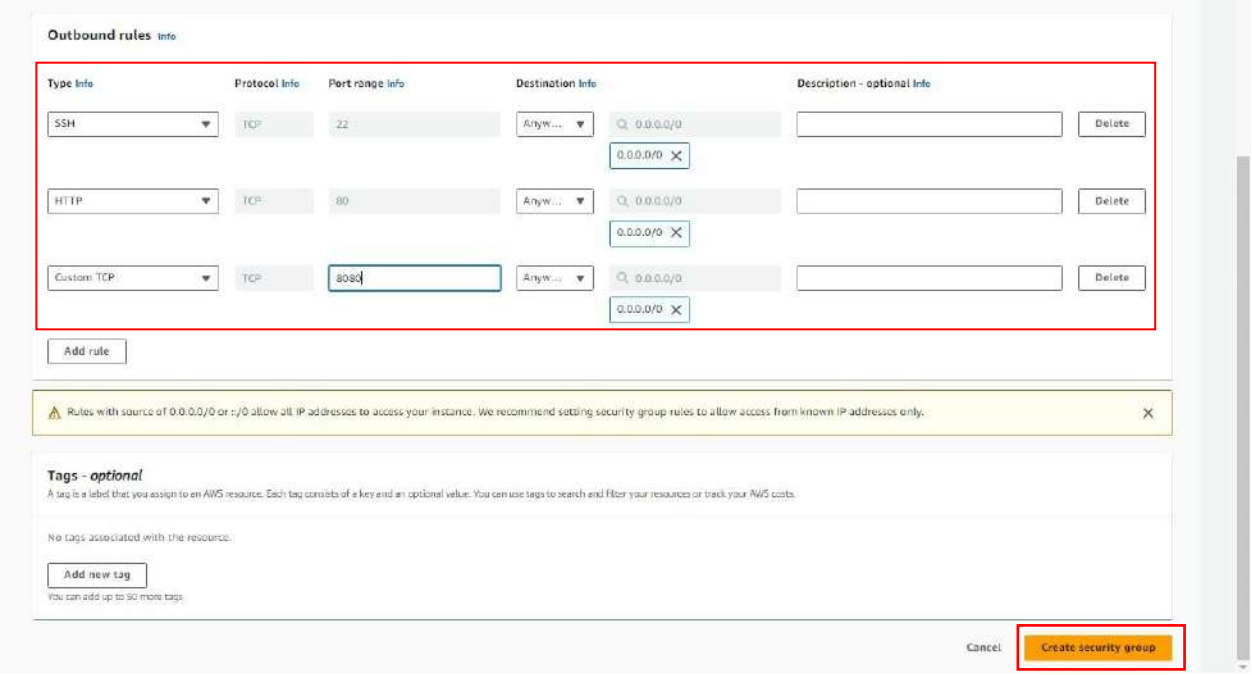
Step 9: Go to **Security Groups** & click on **Create security group** here.



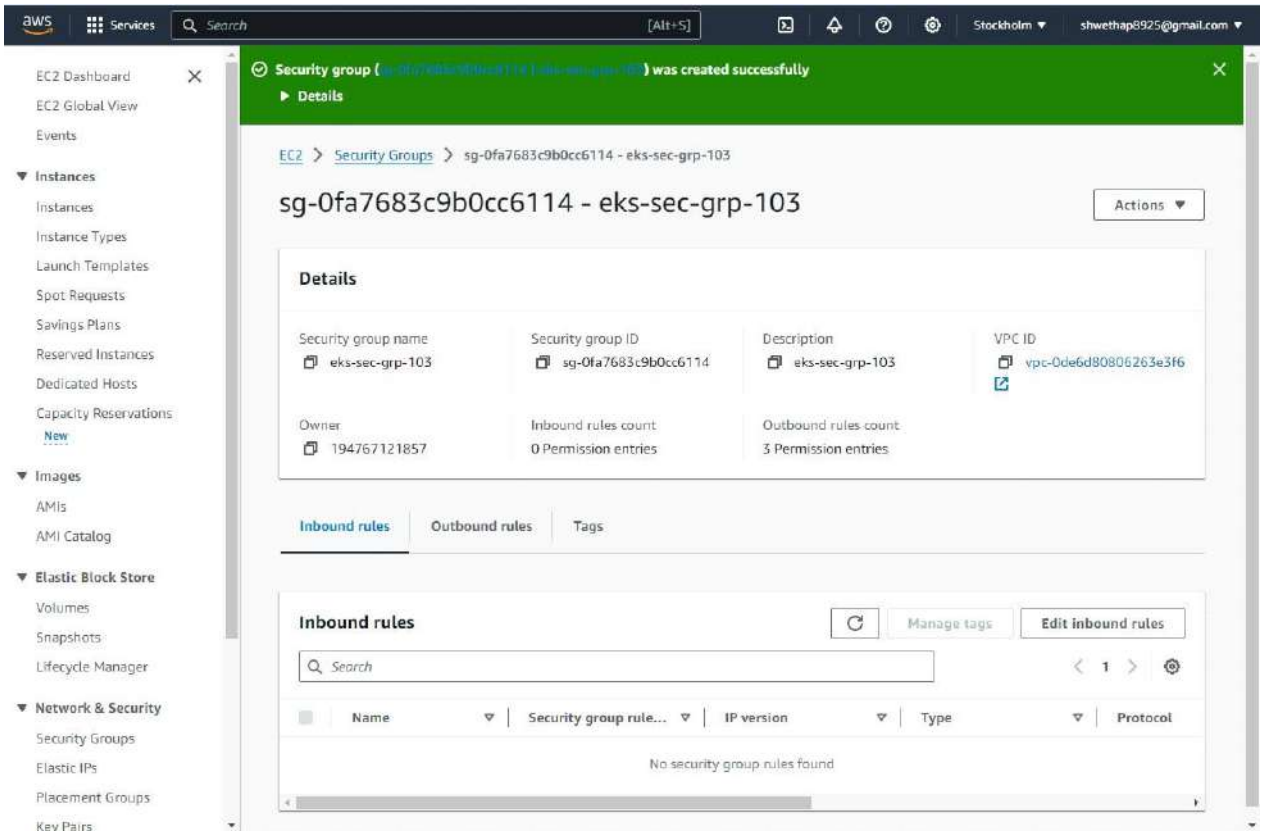
Step 10: Give the name for Security group here. Now copy the same name & paste to the Description as shown here.



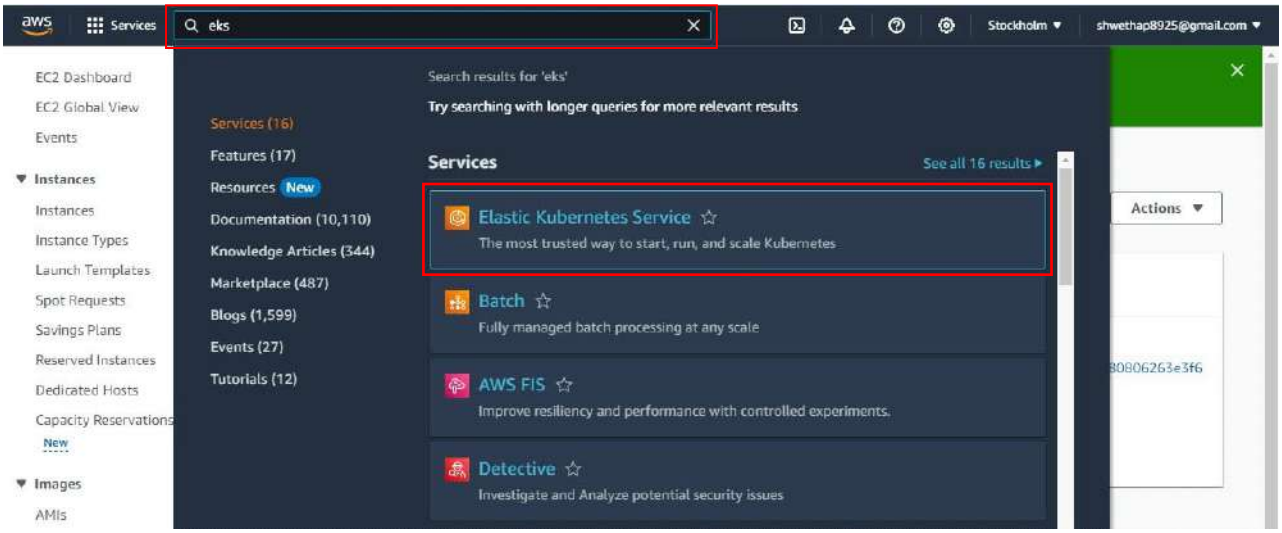
Step 11: Now add rules as shown below & click on **Create Security Group** here.



Step 12: Now the Security group is successfully created here.

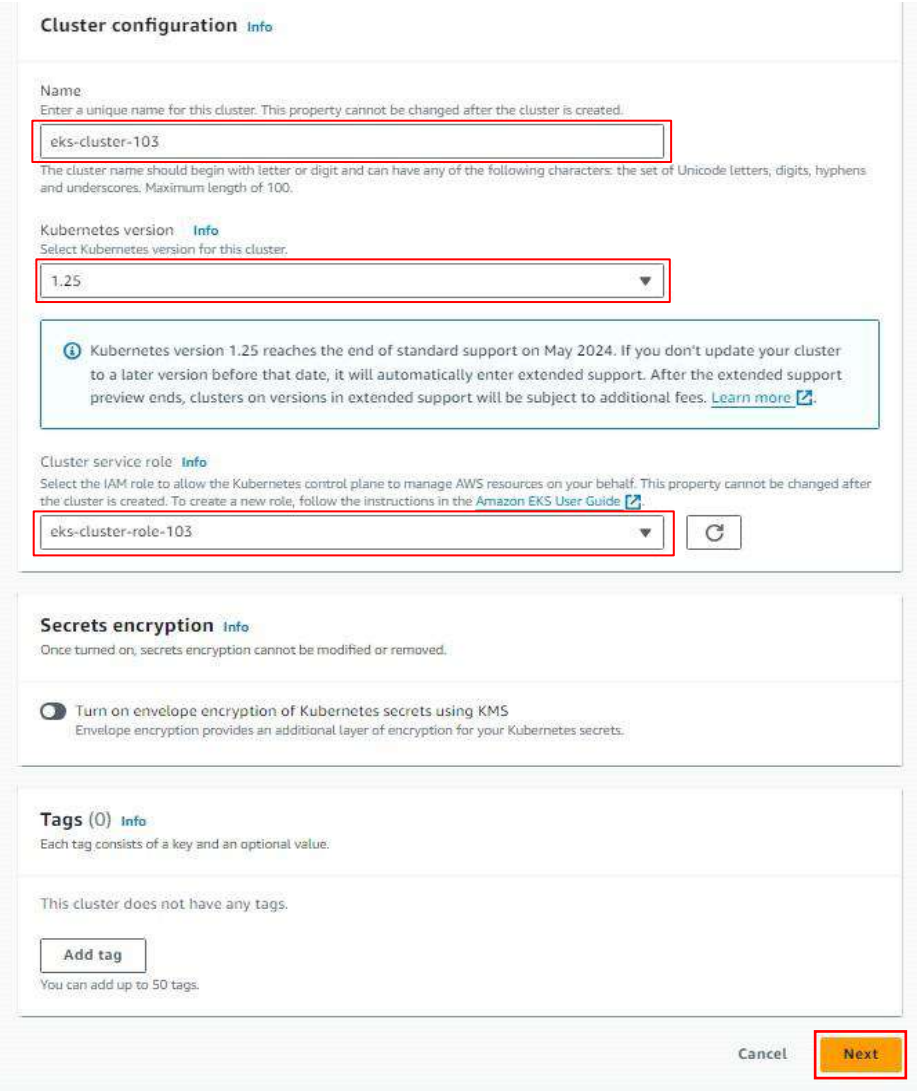


Step 13: Search for **eks** & click on the first link here.

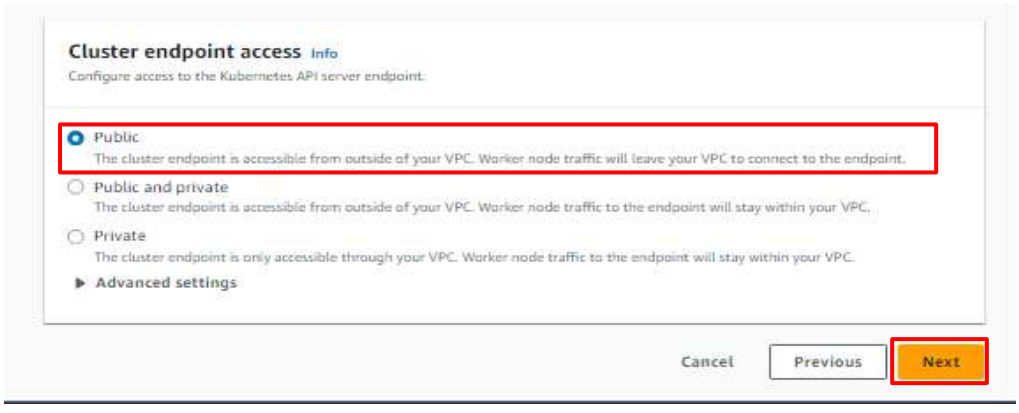


Step 14: Now select **Create** option here.

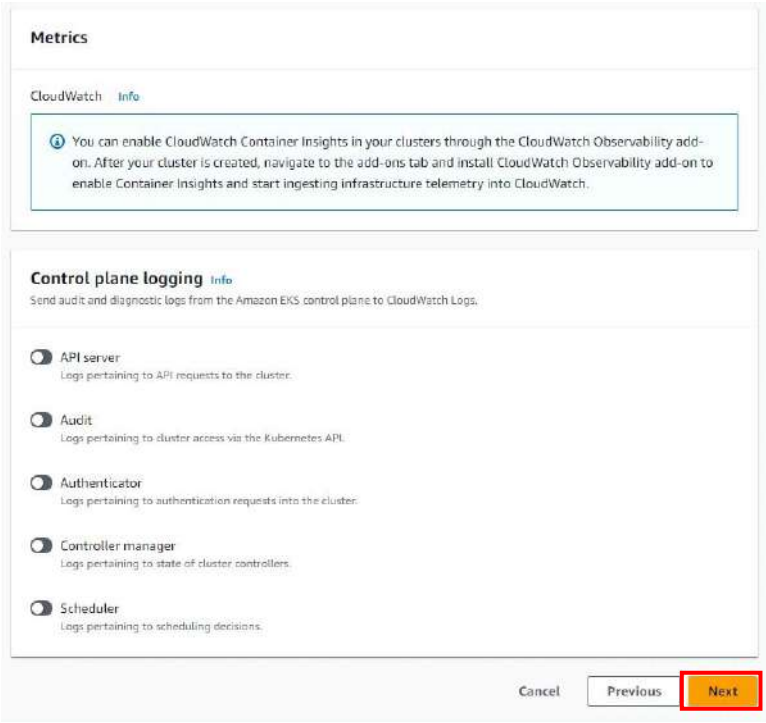
Step 15: Give the name as **eks-cluster-103** □ select the Kubernetes version as **1.25** □ select thecreated cluster service role here & click on **Next** here.



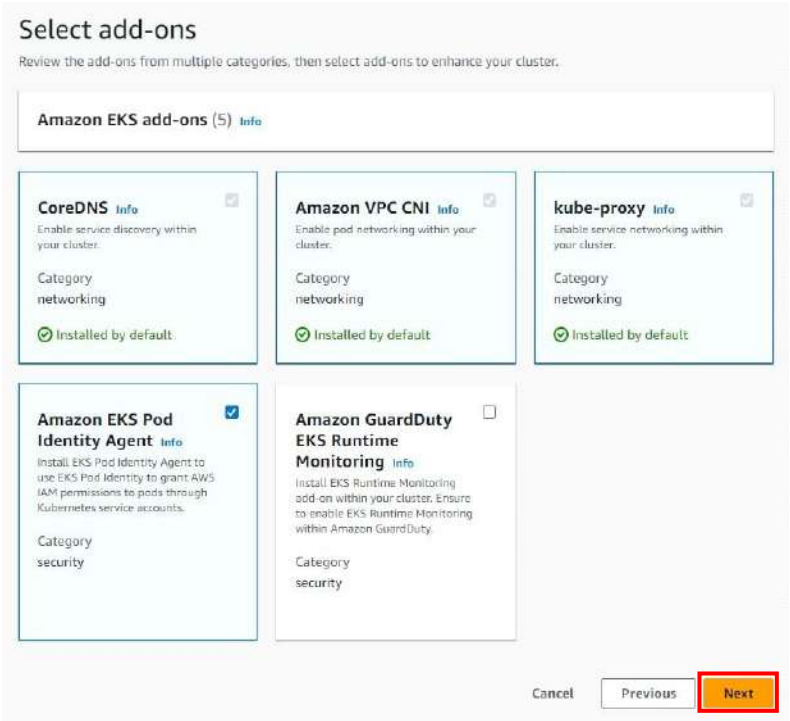
Step 16: Select **Public** endpoint access & click on **Next** here.



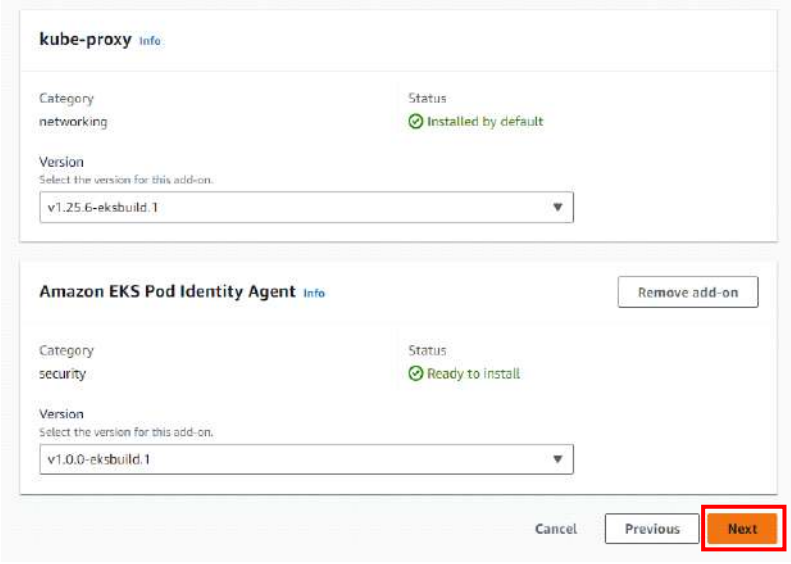
Step 17: Now click on **Next** here.



Step 18: Then click on **Next** here.

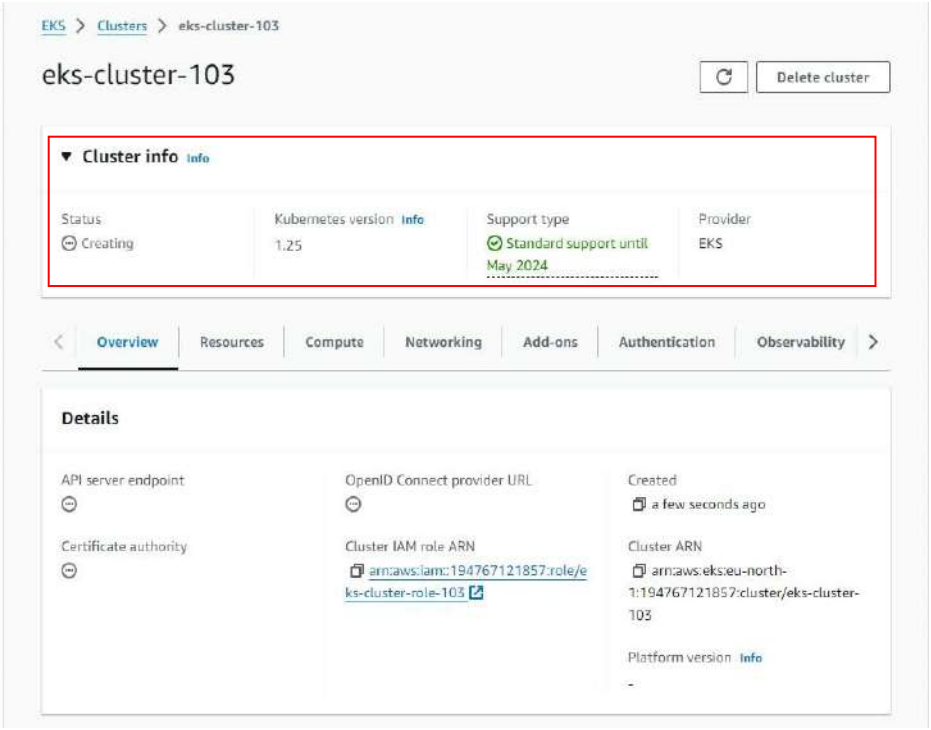


Step 19: Again, click on **Next** here.

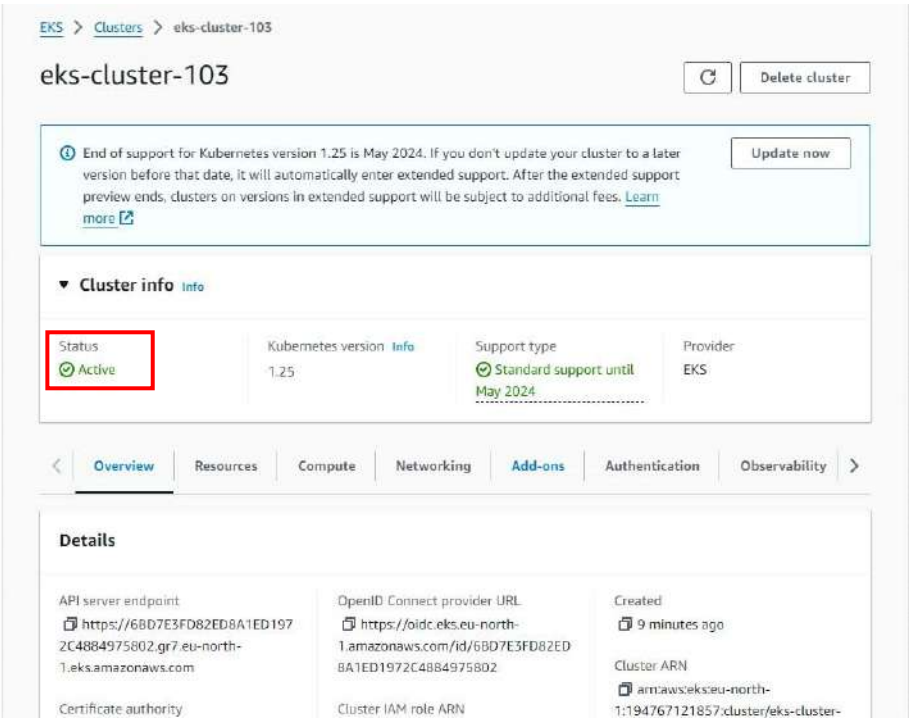


Step 20: Now click on **Create** option here.

Step 21: Now the cluster has started creating as shown here.

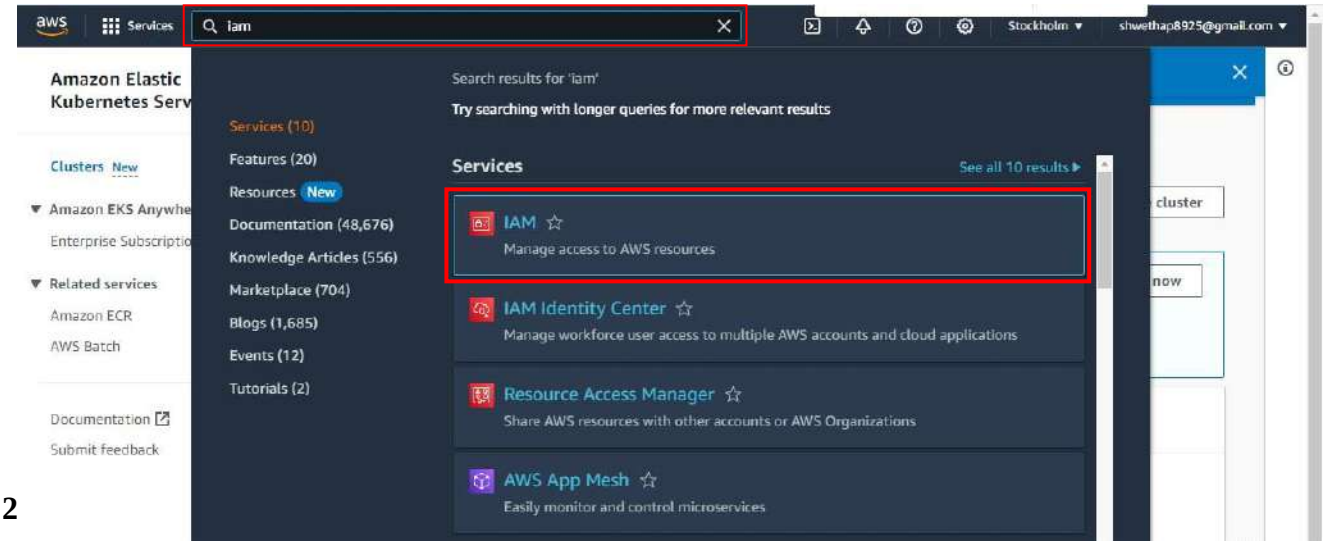


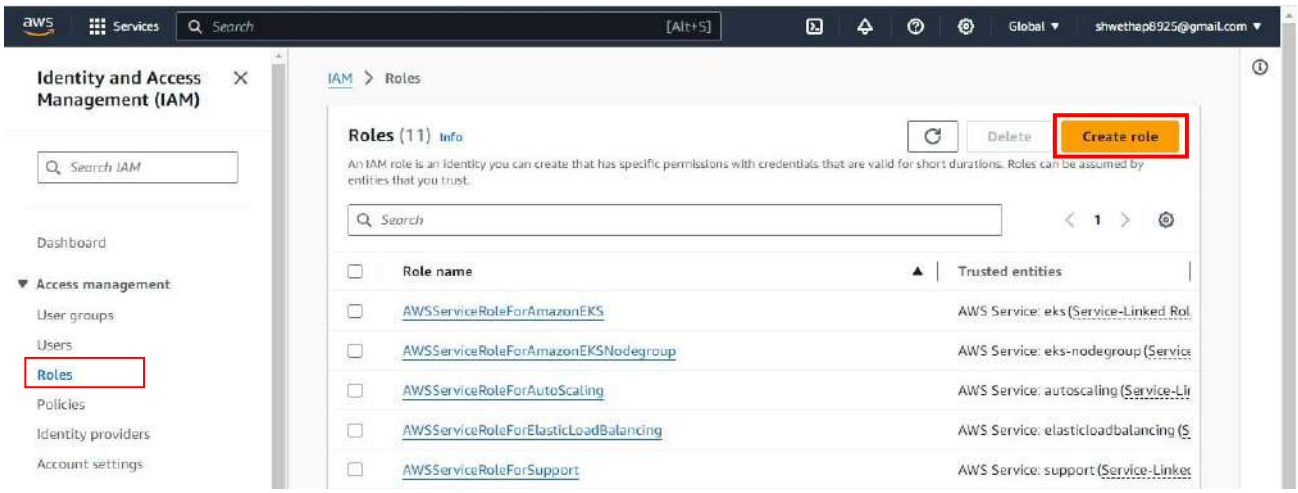
Step 22: Now the cluster creation is successfully done here.



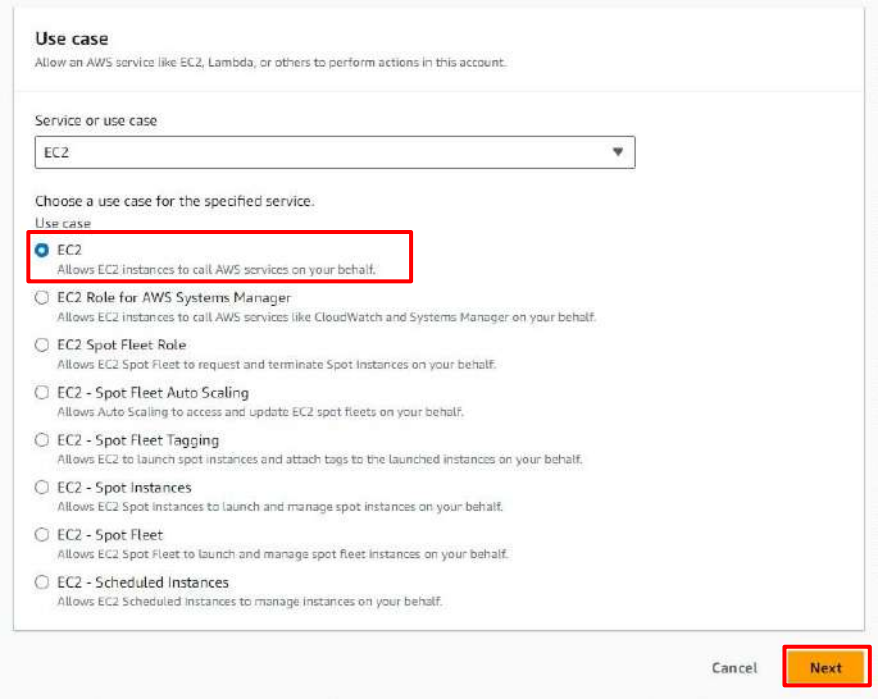
Step 23: Search for **IAM** services & click on the first link here.

Step 2

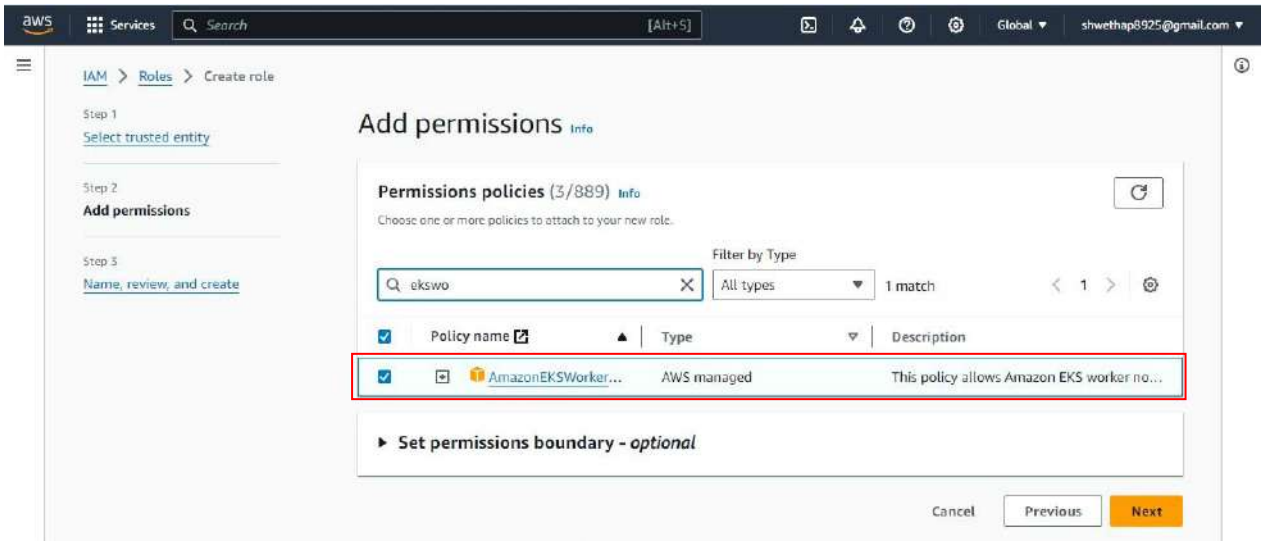




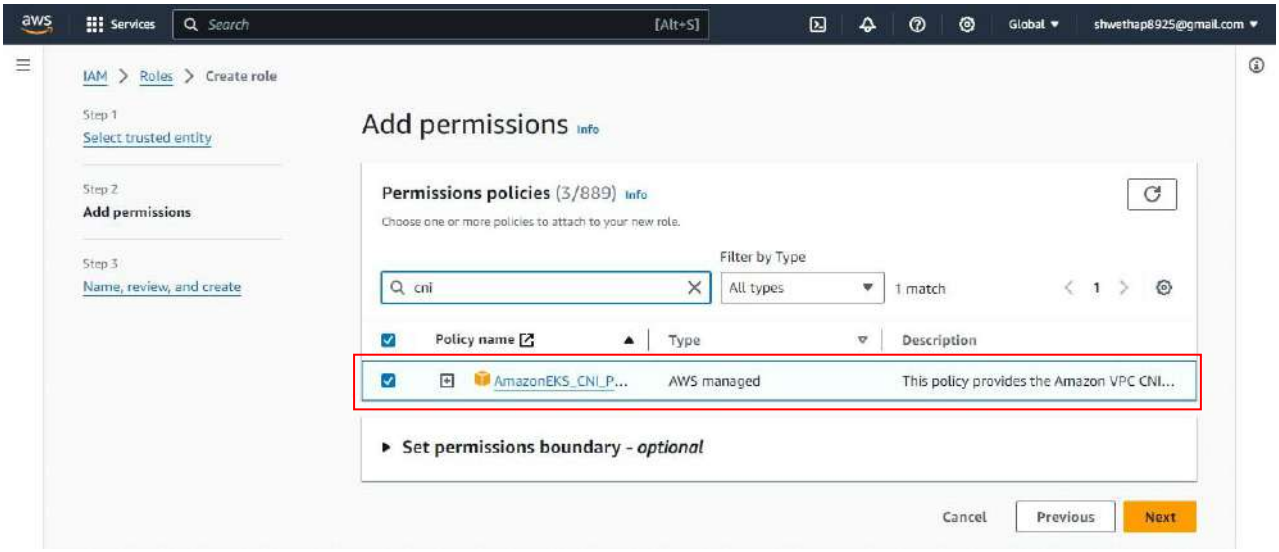
Step 25: Select **EC2** & click on **Next** here.



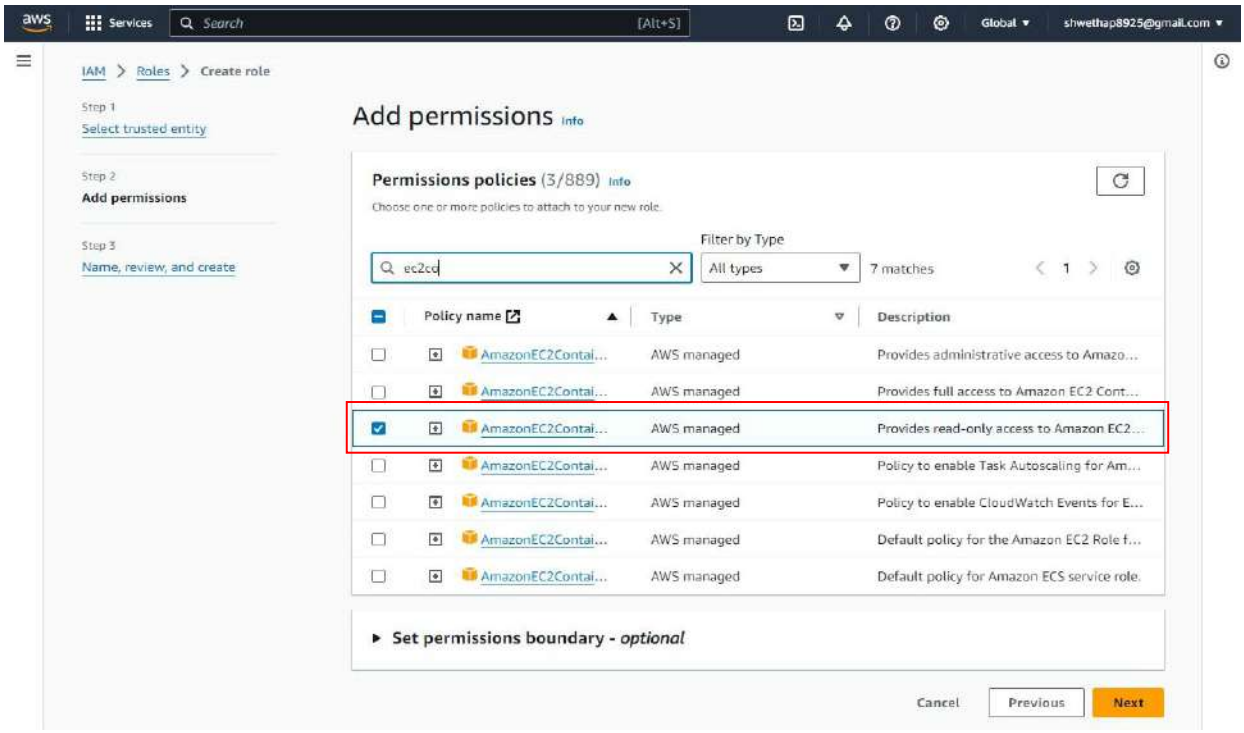
Step 26: Select the policy name as **AmazonEKSWorkerNodePolicy** here.



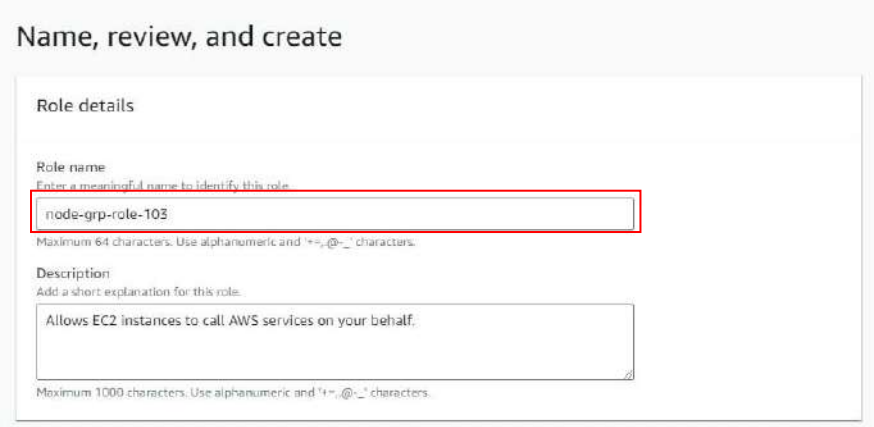
Step 27: Select the policy name as **AmazonEKS_CNI_Poilicy** here.



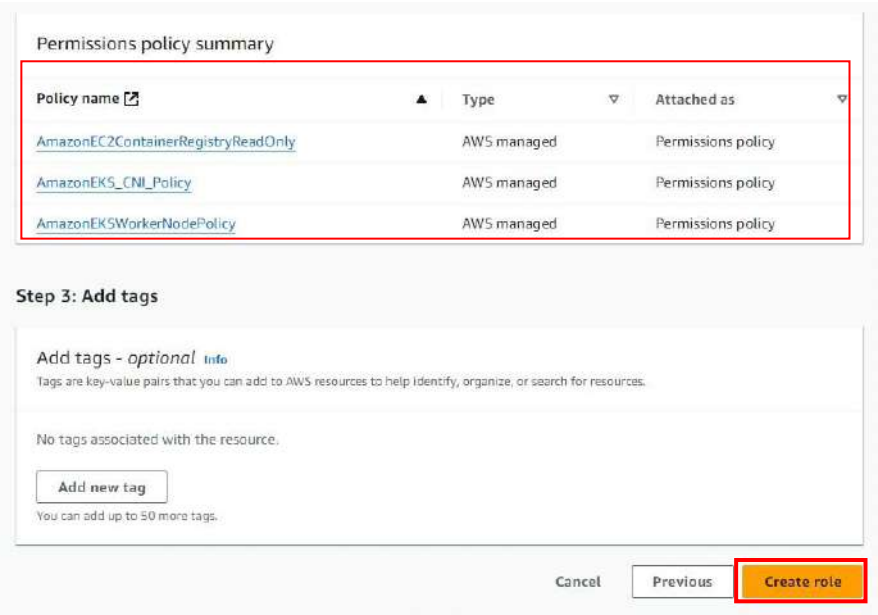
Step 28: Select the policy name as **AmazonEC2ContainerRegistryReadOnly** here & click on **Next**.



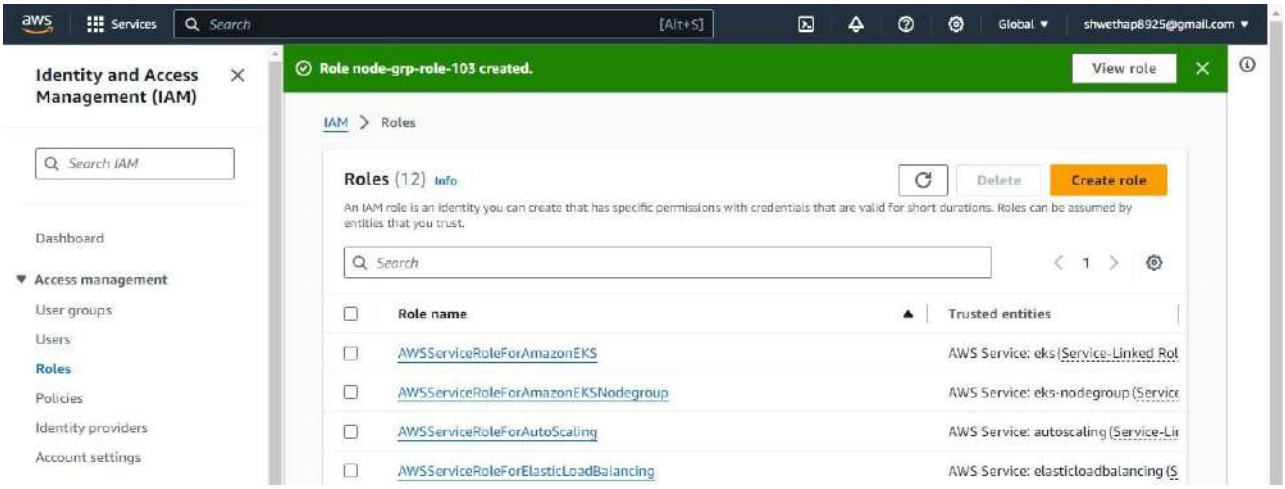
Step 29: Now give the Role name as **node-grp-role-103** here.



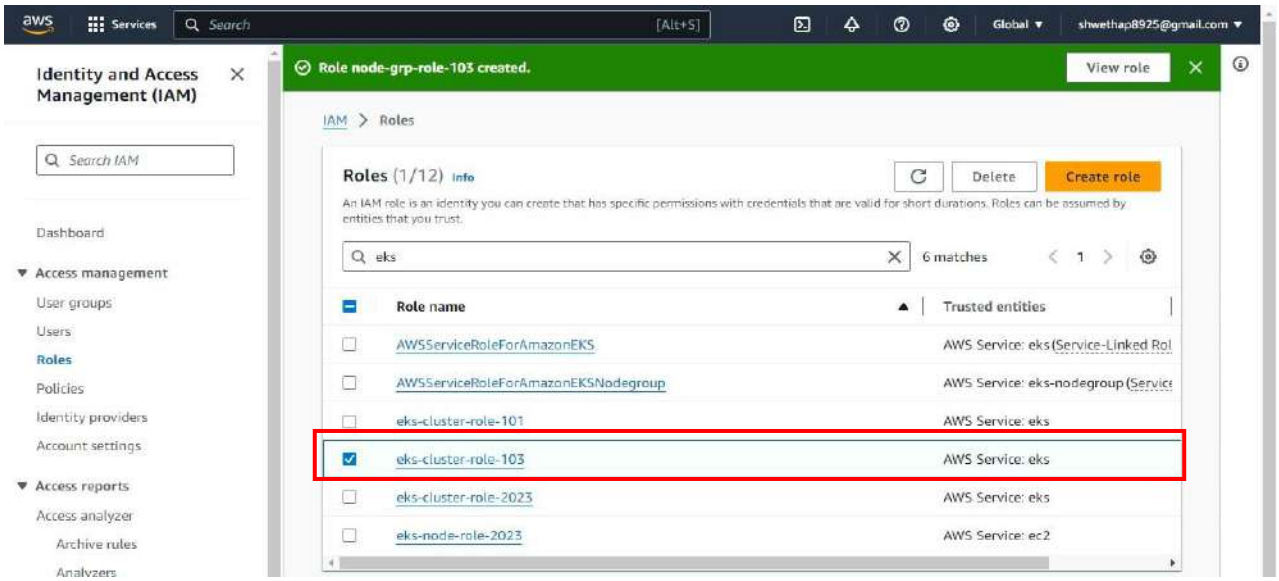
Step 30: Check that all the permissions of the policies are attached as needed. Click on **CreateRole**.



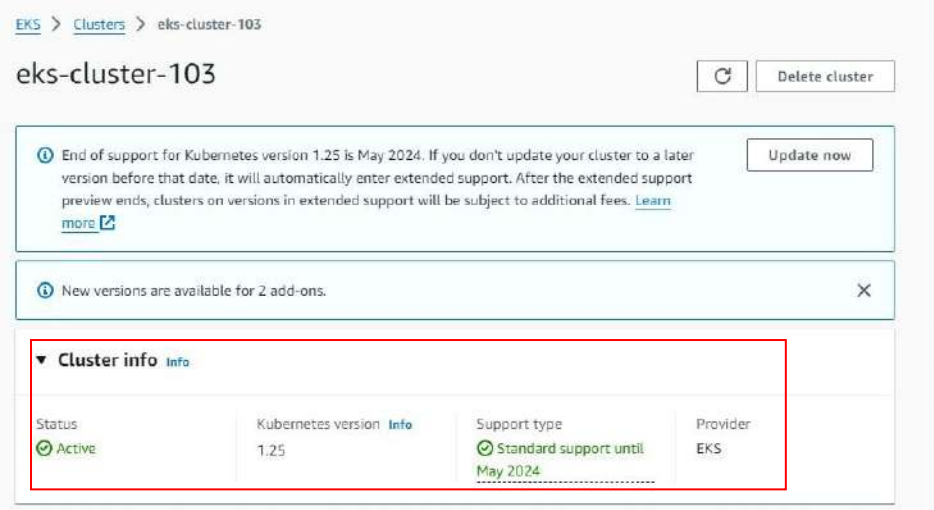
Step 31: Now the Role is successfully created.



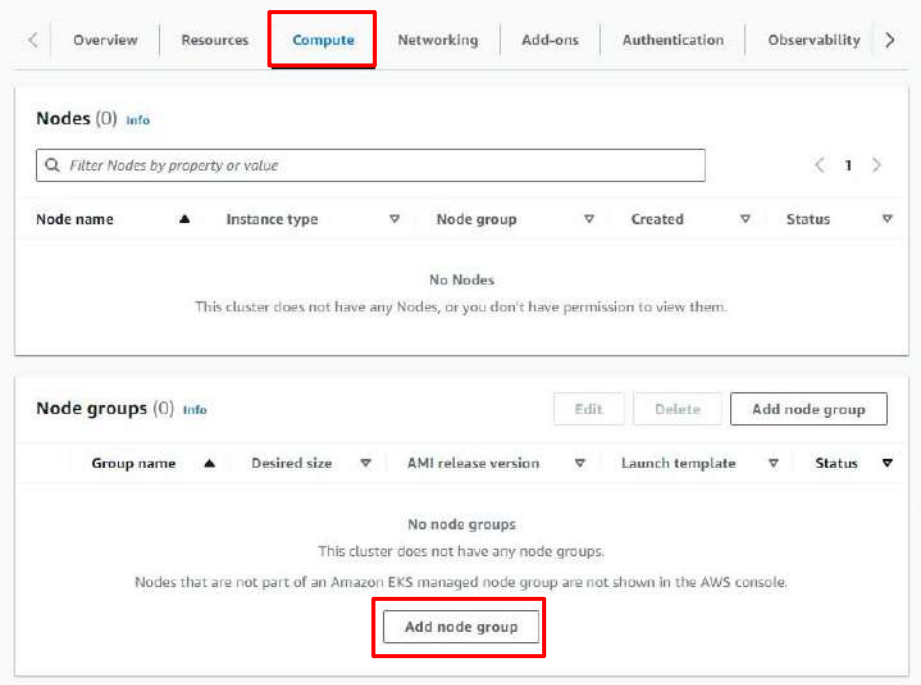
Step 32: Search for **eks** & click on the created EKS cluster here.



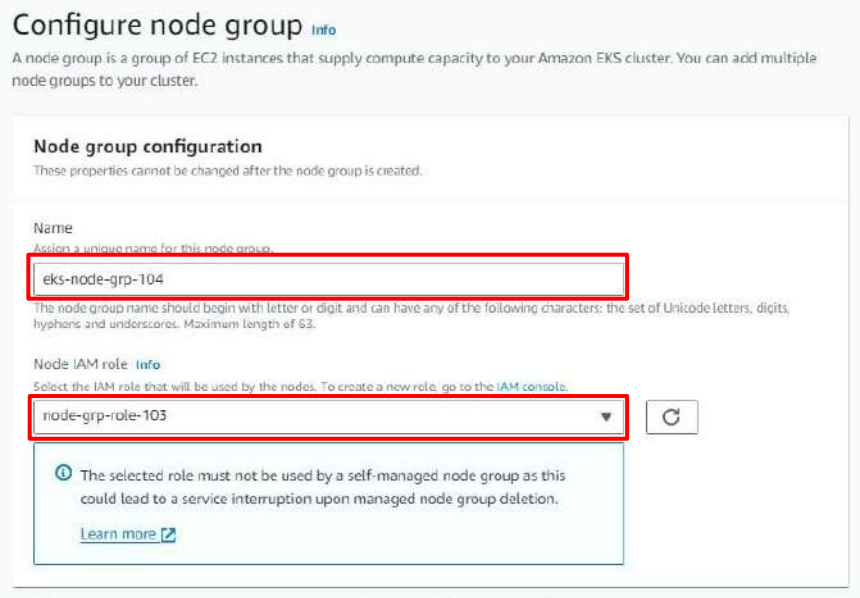
Step 33: View the Cluster info here.



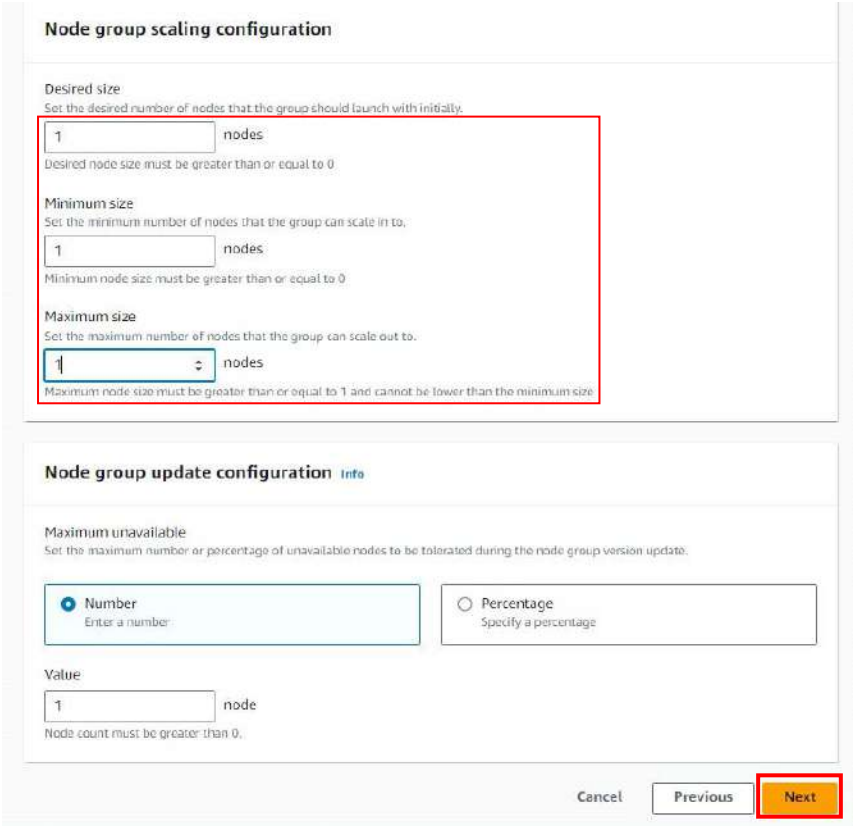
Step 34: Select **Compute** & click on **Add node group** option here.



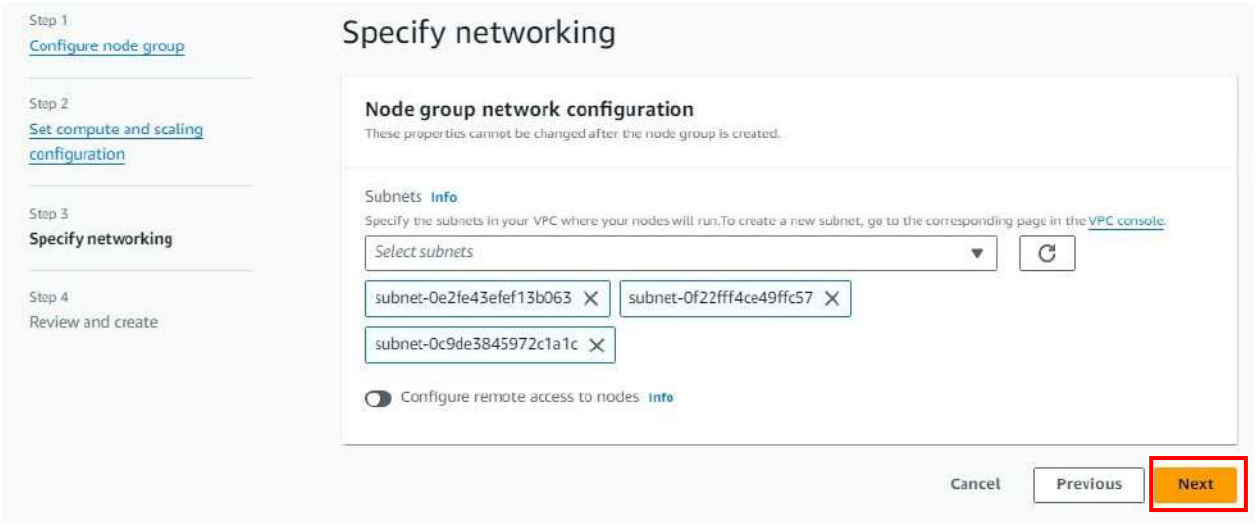
Step 35: Now give the name as **eks-node-grp-104** & select the Node IAM role for that as shownbelow.



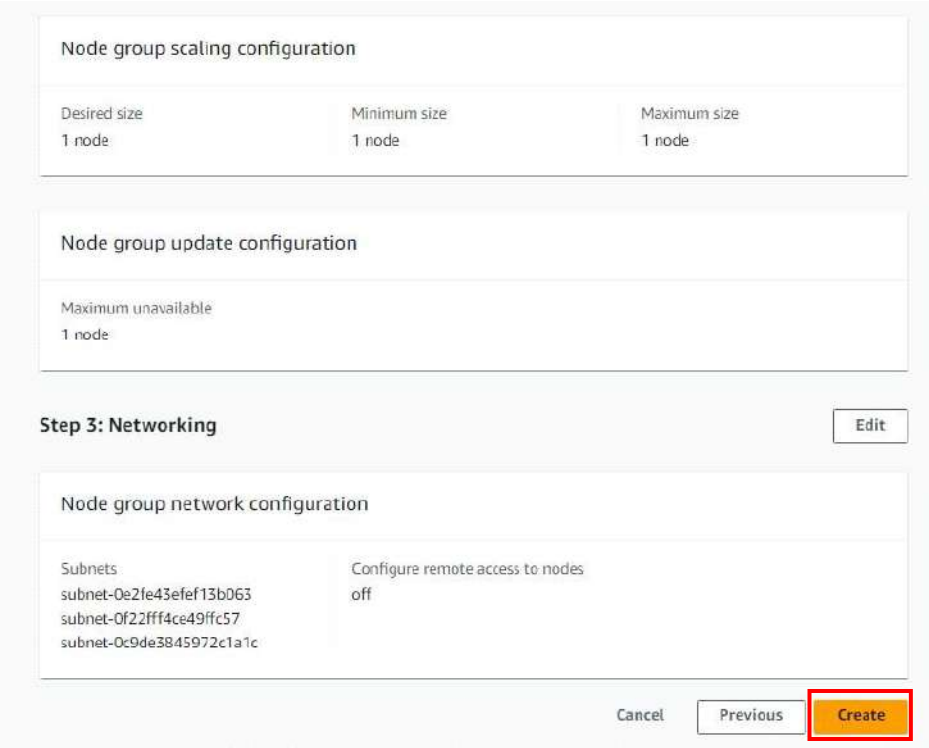
Step 36: Give the Node group scaling configuration to **1** here & click on **Next** here.



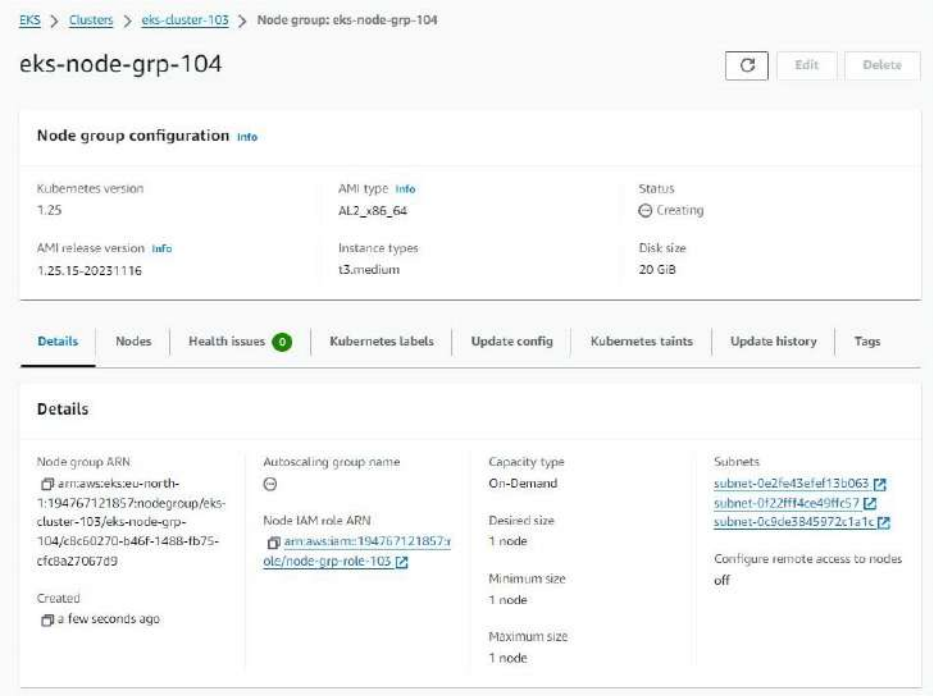
Step 37: Click on **Next** here.



Step 38: Now click on **Next** here.



Step 39: Now the node group is successfully created.



Step 40: Enter the command as **$ aws sts get-caller-identity** as shown below.Enter

the below commands in the AWS CloudShell

$ aws sts get-caller-identity

$ aws update-kubeconfig --region eu-north-1 --name eks-cluster-103

$ cat . kube/config

$ sudo yum installl nano -y

$ nano mygame-svc.yaml

Enter key □ ctrl+o □ Enter key □ ctrl x

$ kubectl apply -f mygame-svc.yaml

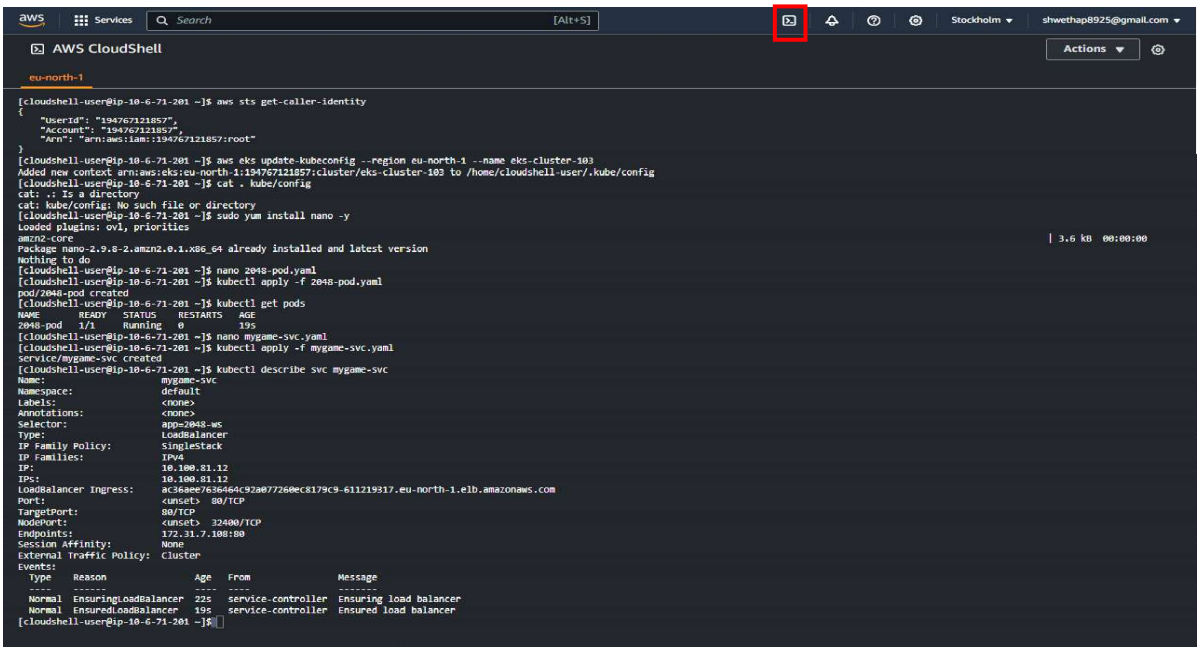
$ kubectl get pods

$ nano mygame-svc.yaml

Enter key □ ctrl+o □ Enter key □ ctrl x

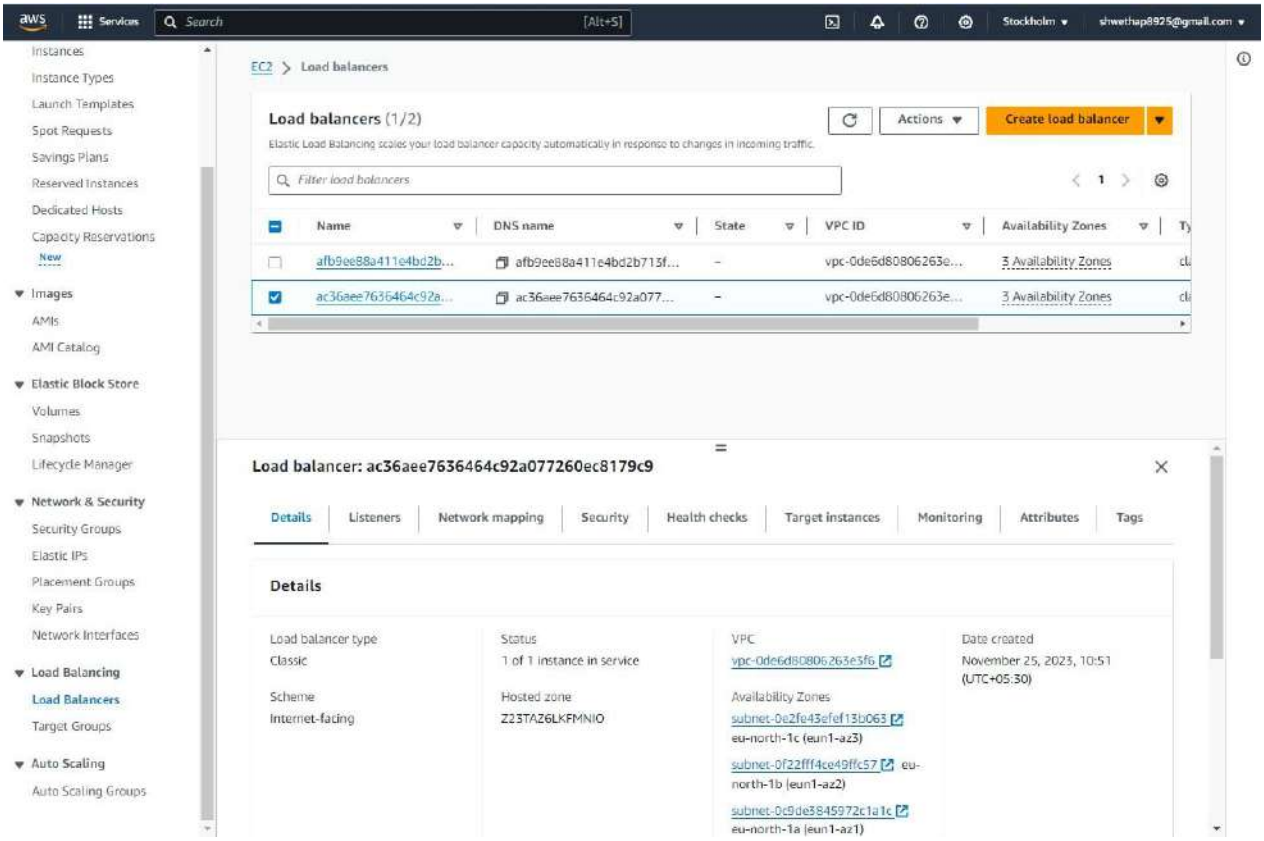
$ kubectl apply -f mygame-svc.yaml

$ kubectl describe svc mygame-svc

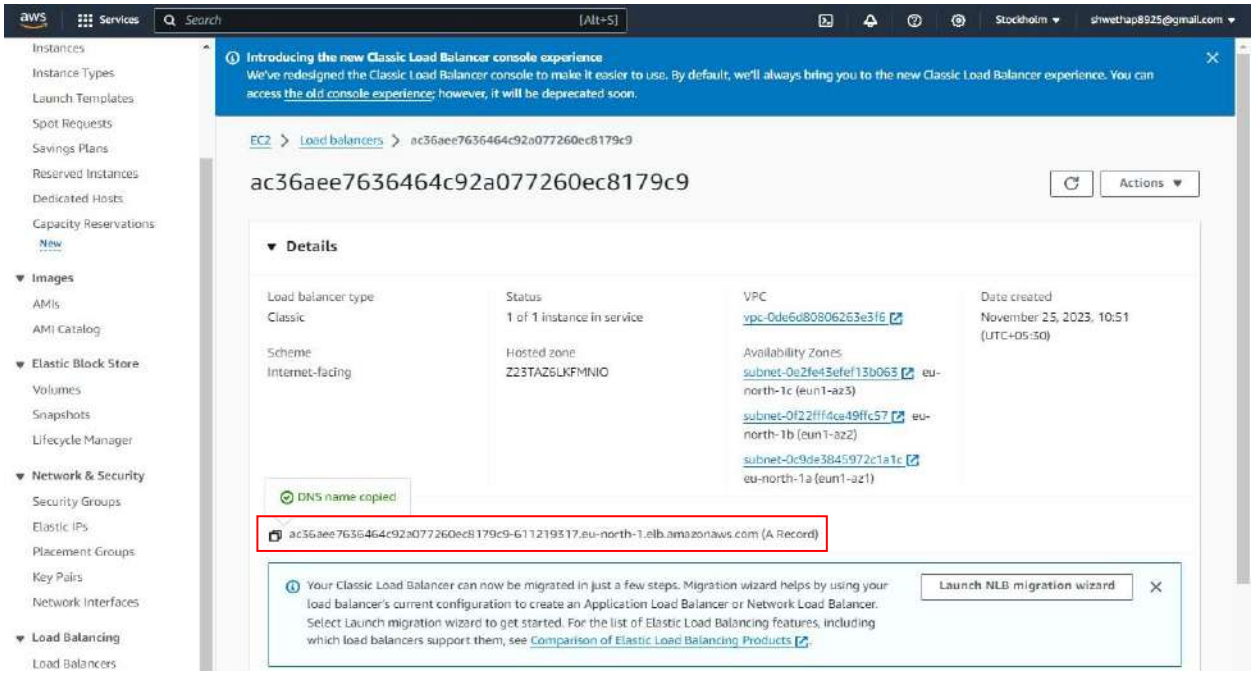


Step 41: Go back to AWS Search for **EC2** click on the first link below.

Step 42: Select **Load Balancers** now select the recently created record name as shownbelow.

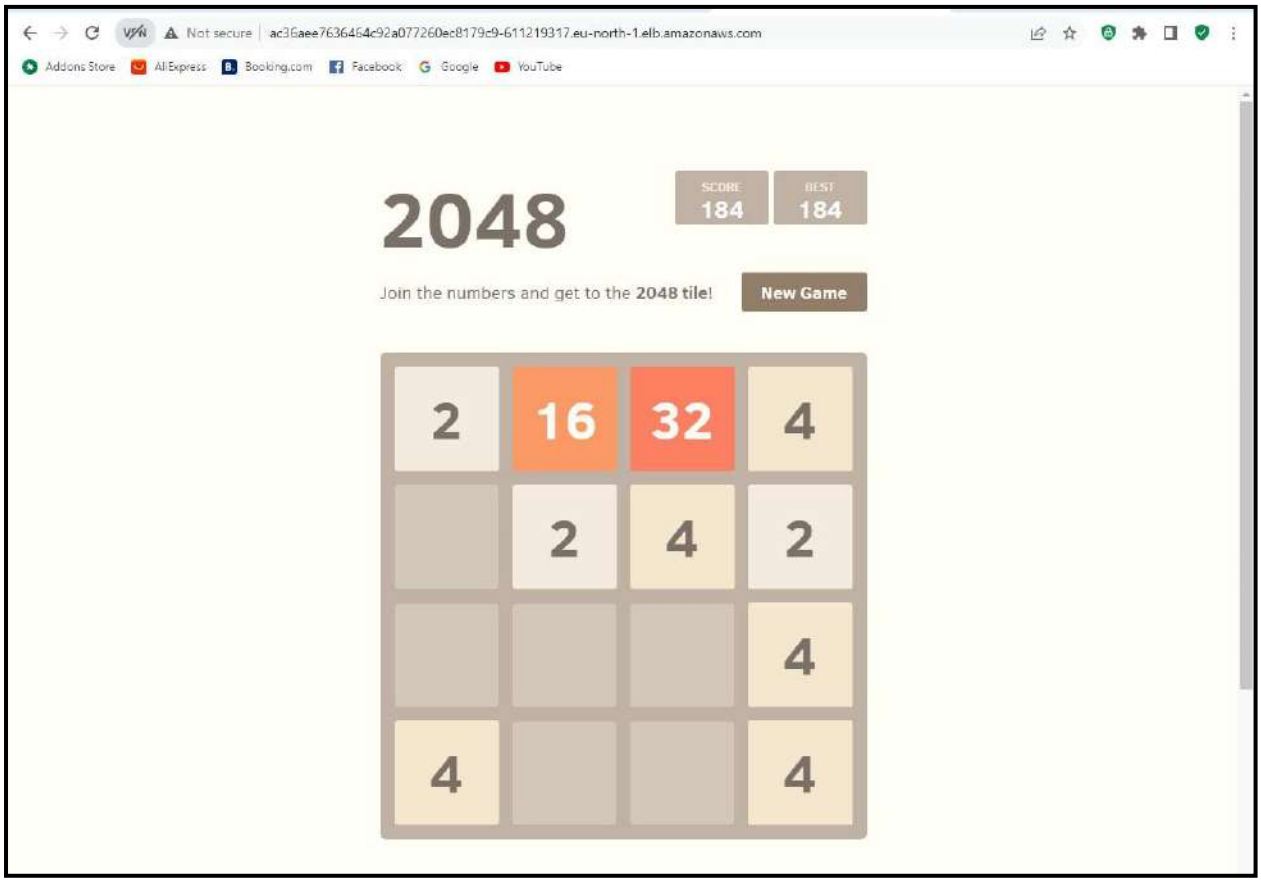


Step 43: Then copy the **DNS name** as shown below.



Step 44: Now paste the **DNS name** copied from step 43 in Chrome & play the game displayed

below.


Not secure
ac36aee7636464c92a077260ec8179c9-611219317.eu-north-1.elb.amazonaws.com
Addons Store
AliExpress
Booking.com
Facebook
Google
YouTube
2048
SCORE
184
BEST
184
Join the numbers and get to the 2048 tile!
New Game
2
16
32
4
2
4
2
4
4
4