

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 the plan they want and can experience the service provided.
- Its headquarters are situated in Los Gatos, California, U.S. registration is required to experience this 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 transformed into a leading global streaming platform.
- By leveraging (maximum use) digital technology and data analytics, Netflix personalized its content recommendations and optimized its 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 telecommunications.

Operational Technology: It is the practice of using hardware and software to control industrial equipments and it primarily interacts 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 come up 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 is required 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 inflationary periods. If increased costs are to be passed on to the buyer, then demand typically falls, therefore fewer goods 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 trends and 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 led to 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 has tested VR in stores to help customers visualize how furniture fits in their homes and has acquired California-based Geomagical 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 and response 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 to adopt new technology and systems.

Conclusion

In conclusion, IKEA's digital transformation journey serves as an inspiring example of how a legacy 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, IKEA has 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 proper measures in adopting new technologies and recruiting more manpower for the required workforce, proper quality control for controlling regarding production issues etc. this leads to 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.

The screenshot shows the Jira Software interface in a web browser. The top navigation bar includes links for 'Your work', 'Projects', 'Filters', 'Dashboards', 'Teams', 'Apps', and a 'Create' button. A search bar and various filter options are also present. On the left, a sidebar menu lists 'PLANNING' (selected), 'DEVELOPMENT', and 'REPORTS' sections, with 'IA board' under PLANNING. The main area is titled 'Timeline' for the 'IA board' project. It displays a grid with columns for 'Sprints' (one row for 'IA-1 IKEA'), 'Releases' (one row for 'IA-1 IKEA'), and dates ('AUG', 'SEP'). Below this, a detailed backlog is shown for 'IA-1 IKEA' with tasks like 'IA-2 Splash screen', 'IA-3 Displays the logo of the...', 'IA-4 Login or Sign up', 'IA-5 Login with using your g...', 'IA-6 Homepage', 'IA-7 Displays all the product...', 'IA-8 Search', and 'IA-9 Custom...'. Each task has a progress bar and a 'TODAY' button. At the bottom, there are buttons for 'Today', 'Weeks', 'Months' (selected), 'Quarters', and a 'More' icon.

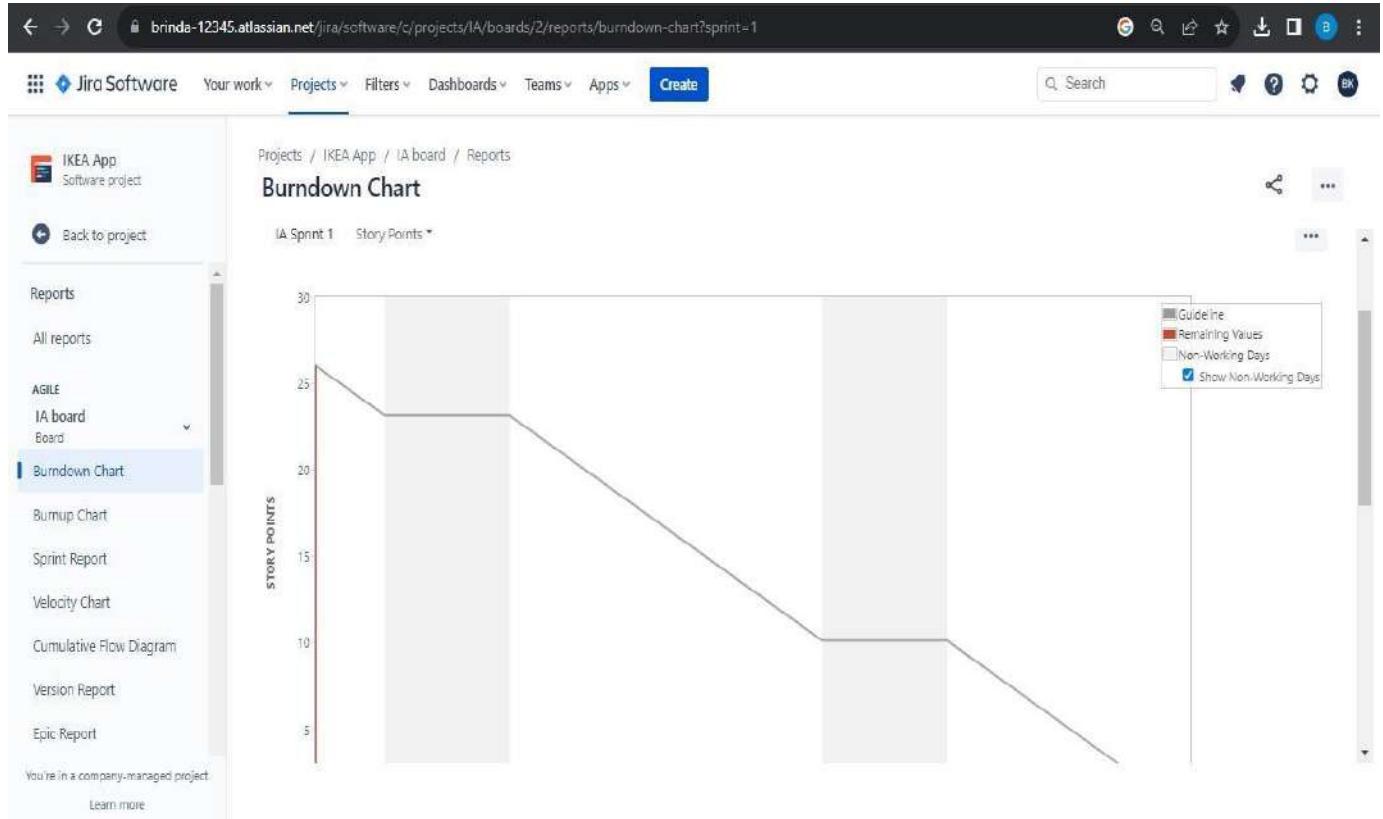
The screenshot shows the Jira Software interface for the 'IKEA App' project. The left sidebar has 'IA board' selected under 'PLANNING'. The main area is titled 'Timeline' and displays a grid for the month of August. The grid rows are labeled 'Sprints' and 'Releases'. Under 'Releases', there is a list of user stories: IA-8, IA-9, IA-10, IA-11, IA-12, IA-13, IA-14, and IA-15. Each story is in the 'TO DO' state. A vertical orange bar marks the end of the sprint. At the bottom of the grid, there is a button '+ Create Epic'. The top right of the screen has options for 'Give feedback', 'Share', 'Export', and more.

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.

The screenshot shows the Jira Software interface for the 'IKEA App' project. The left sidebar has 'Backlog' selected under 'PLANNING'. The main area is titled 'Backlog' and shows two sections: 'Sprints' and 'Backlog'. The 'Sprints' section shows 'IA Sprint 2' with 4 issues: 'Homepage', 'Displays all the products details', 'Search tab', and 'Customers searches the required products'. The 'Backlog' section shows 6 issues: 'Customers profile tab', 'Displays the customers personal details.', and 'Location tracking'. On the right side, there are buttons for 'Start sprint' and 'Create sprint'. The top right of the screen has options for 'Share', 'Insights', and more.



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.

The screenshot shows the Jira Software Timeline view for the project 'IKEA APP123'. On the left, a sidebar menu is open with options like 'PLANNING' (selected), 'Timeline' (highlighted with a red box), 'Backlog', 'Active sprints', 'Reports', 'Issues' (highlighted with a red box), and 'Components'. The main area displays a timeline from SEP to NOV. Under the 'Sprints' section, there is an epic titled 'IA-1 IKEA' which contains 11 user stories (IA-1 to IA-11). Each user story has a status indicator: IA-2, IA-3, IA-4, IA-5, IA-6, IA-7, IA-8, IA-9, IA-10, and IA-11 are all marked as 'TO DO'. IA-11 has a longer vertical orange bar extending downwards, indicating it is a parent issue for other tasks. At the bottom right of the timeline, there are buttons for 'Today', 'Weeks', 'Months' (selected), 'Quarters', and a date range selector.

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

The screenshot shows the Jira Software Issues view for the project 'IKEA APP123'. On the left, a sidebar menu is open with options like 'Filters' (selected), 'All issues', 'My open issues', 'Reported by me', 'Open issues', 'Done issues', 'Viewed recently', 'Resolved recently', 'Updated recently', and 'View all filters'. The main area shows a list of issues under the epic 'IA-1 IKEA'. The first three issues (IA-2, IA-3, IA-4) have their status dropdown menus open, with 'DONE' selected (indicated by a red box). The fourth issue, 'IA-4 Login or sign up', has its status dropdown menu open, showing 'IN PROGRESS' (also indicated by a red box). Other issues listed include 'IA-5 Displays the logo of the application', 'IA-6 Homepage', 'IA-7 Displays the product de...', 'IA-8 Payment', 'IA-9 Online payment methods', 'IA-10 Location', and 'IA-11 Track the products deli...'. The right side of the screen shows detailed information for the selected issue, including fields for Assignee (Unassigned), Reporter (Brinda Krishnamurthy), Labels (None), Priority (Medium), Epic Name (IKEA), and Status updates (Status updates icon).

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.

The screenshot shows the Jira Software Issues page for the project 'IKEA APP123'. The page displays a list of tasks with columns for Type, Key, Summary, Assignee, Reporter, Priority, Status, Resolution, and Due Date. There are 11 tasks listed, each with a unique key (IA-1 to IA-11) and a brief description. The 'Status' column shows various states like Unresolved, Done, and In Progress. The 'Resolution' column shows the date when the task was resolved. The 'List View' button at the top right of the table is highlighted with a red box.

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

The image contains two side-by-side screenshots of the Jira Software Project Settings page for the 'IKEA 123' project.

- Left Screenshot:** Shows the 'Project settings' option highlighted with a red box in the sidebar under the 'PLANNING' section.
- Right Screenshot:** Shows the 'Issue types' section highlighted with a red box in the main content area under the 'Details' tab.

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”.

The screenshot shows the Jira Software interface for creating a new issue type. On the left, there's a sidebar with project settings and issue types (Epic, Story, Bug, Task, Subtask). The 'Story' option is selected and highlighted with a red box. The main area shows the 'Story' configuration page with fields like 'Summary' (REQUIRED) and 'Acceptance Criteria'. A large grid of field types is on the right, with 'Paragraph' highlighted with a red box. The 'Save changes' button at the bottom right is also highlighted with a red box.

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 be always sprinting so you can deliver working software, while continuously learning and improving.

The screenshot shows a Jira backlog item for 'IKEA'. The 'Description' field contains the text 'IKEA is an online retail manufacturing company'. Below it, the 'Child Issues' section is expanded, listing several sub-tasks: 'Splash screen', 'IA-3: Displays the logo of the application', 'IA-4: Login or sign up', 'IA-5: Login with your Gmail-id', 'IA-6: Homepage', 'IA-7: Displays the product details', and 'IA-8: Payment'. Each sub-task has a status indicator (e.g., DONE, IN PROGRESS) and a progress bar.

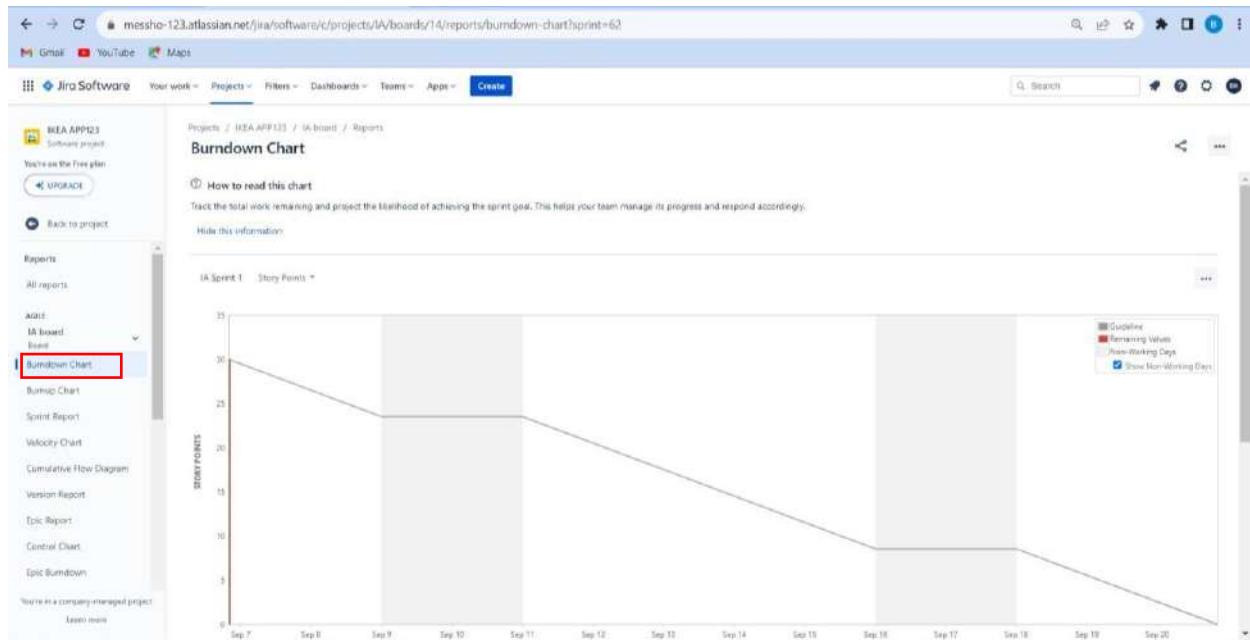
B. Backlog Refinement: JIRA Backlog Refinement otherwise called JIRA Backlog Grooming is a key process in Scrum. Backlog refinement is the process of reviewing, ranking, and editing your product backlog. Backlog refinement is an important tool in your product development process because it helps your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

The screenshot shows the Jira Software interface for the 'IA board'. On the left, there's a sidebar with sections for 'PLANNING' (IA board, Board, Timeline, Backlog, Active sprints, Reports), 'DEVELOPMENT' (Code), and 'COMPONENTS'. The main area is titled 'Backlog' and shows 'IA Sprint 1' with 6 issues. The backlog items are:

- Splash screen
- Displays the logo of the application
- Login or sign up
- Login with your Gmail-id
- Homepage
- Displays the product details

To the right, there's a sidebar with '6 issues Estimate 0' and a 'Start sprint' button.

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 project the likelihood of achieving the sprint goal. By tracking the remaining work throughout the iteration, a team can manage its progress, and respond to trends accordingly.



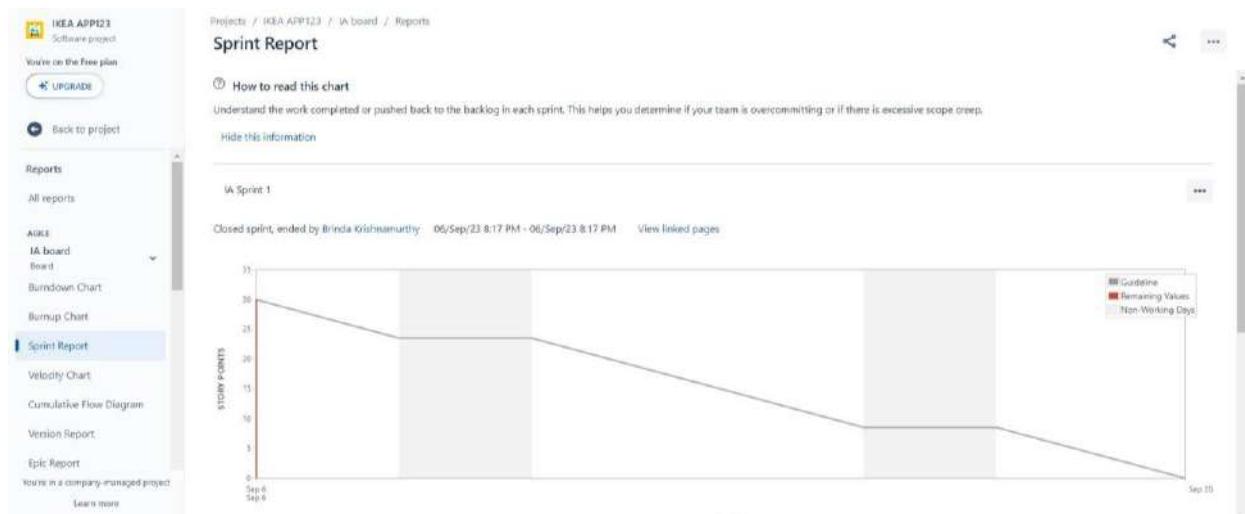
D. Sprint Demo: In a traditional scrum method, the sprint demo comes at the end of a sprint. A sprint demo is your team's chance to show off to the business stakeholders. The work that they've done is important, and you want to share it with the people who are most excited about it. The best sprint demos 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.

The screenshot shows the Jira Software interface with the URL mesho-123.atlassian.net/projects/I1/selectedItem=com.atlassian.jira.projects-plugin%3Areport-page. The top navigation bar includes links for Gmail, YouTube, Maps, and various Jira software features like Your work, Projects, Filters, Dashboards, Teams, Apps, and Create. A search bar is also present.

The main content area displays the 'All reports' page for the 'IKEA APP123' project. On the left, there's a sidebar with sections for Agile (IA board, Board, Burndown Chart, Burnup Chart, Sprint Report, Velocity Chart, Cumulative Flow Diagram, Version Report) and Reports (All reports). The 'Sprint Report' section is currently selected and highlighted with a red box.

The central area shows several report cards:

- Burndown Chart**: Track the total work remaining and project the likelihood of achieving the sprint goal.
- Burnup Chart**: Track the total scope independently from the total work done.
- Sprint Report**: Understand the work completed or pushed back to the backlog in each sprint. This helps you determine if your team is overcommitting or if there is excessive scope creep. **(This card is highlighted with a red box)**
- Velocity Chart**: Track the amount of work completed from sprint to sprint. This helps you determine your team's velocity and estimate the work your team can realistically achieve in future sprints.
- Cumulative Flow Diagram**
- Version Report**
- Epic Report**
- Control Chart**



E. Sprint Retrospective: The retrospective is a meeting that takes place at the end of a sprint. The length of the meeting depends on the time scale and complexity of the iteration. 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.

IKEA App

Overview

Reflect on past work and identify opportunities for improvement by following the instructions for the Retrospective Play.

Page Properties

Date	06-09-2023
Team	Management Team
Participants	@Brinda Krishnamurthy, Deepthi, Bhoomika, Anikitha, Deekshitha, Brundhashree, Harshini, Nischitha.

Retrospective

Add your Start doing, Stop doing, and Keep doing items to the table below. We'll use these to talk about how we can improve our process going forward.

Start doing	Stop doing	Keep doing
• e.g., Sparring work earlier in the development cycle	• e.g., Scheduling meetings without a clear agenda	• e.g., Sharing progress at daily standups

Action items

Add 1-2 follow-up action items to help the team apply what they learned in the retrospective:

Type your action; use @ to assign to someone.

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 in a 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

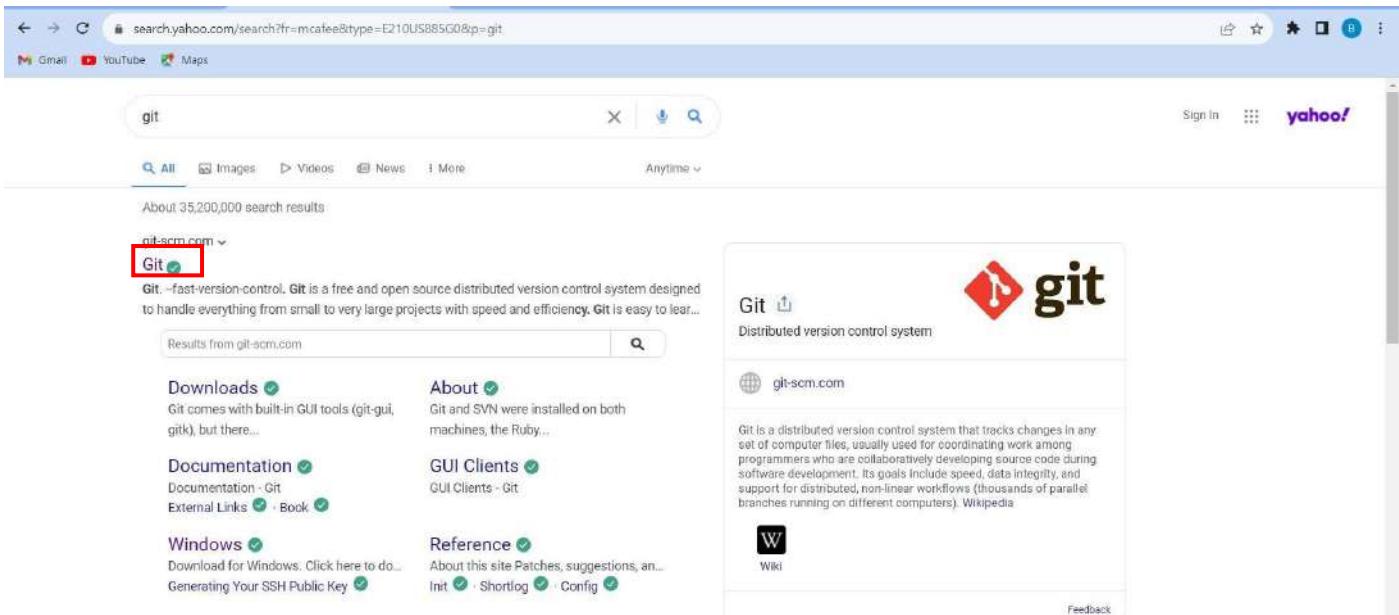
Sl. No.	Test Steps	Test Data	Expected Result	Actual Result	Status
1.	Create a Search bar based on SEO technology.	Searching of required products, materials or items.	Displaying of the products based on search results.	Searching of the available products in that platform.	Pass
2.	To check whether all 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
----	---------------------------	--	--	--	------

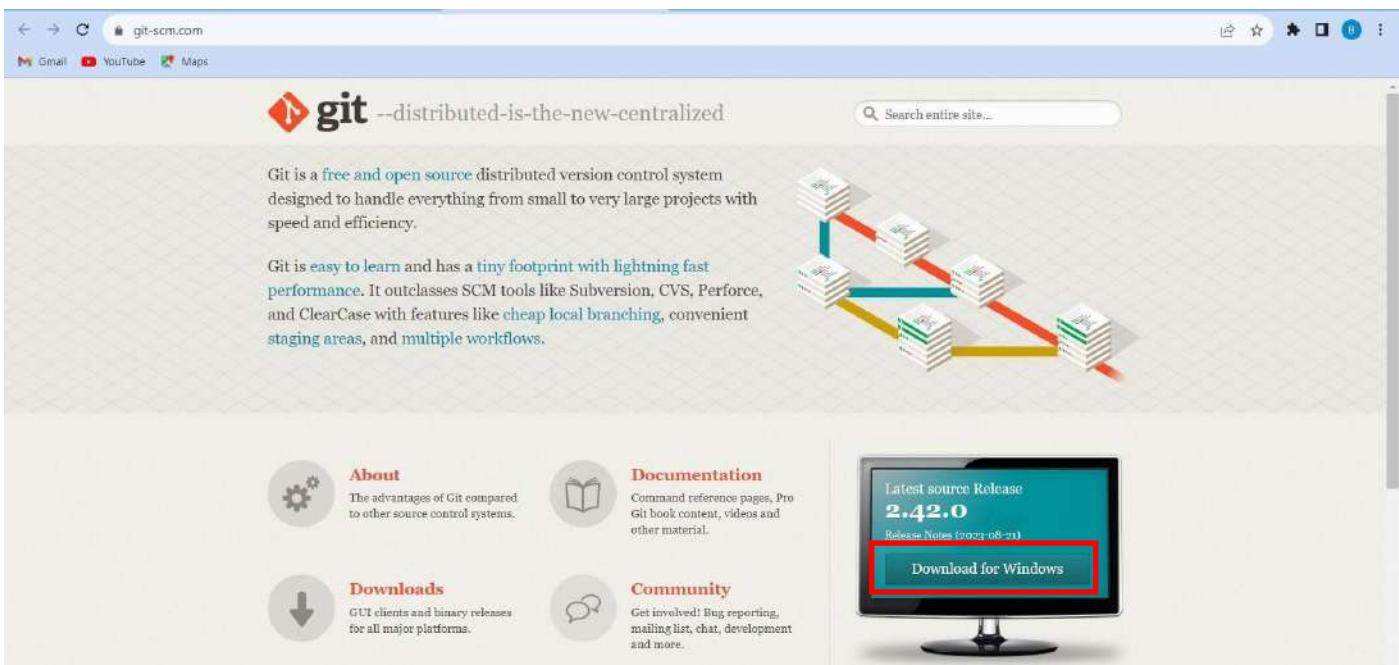
WEEK – 3

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’.



Search entire site...

About
Documentation
Downloads
GUI Clients
Logos
Community

The entire [Pro Git book](#) written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Download for Windows

[Click here to download](#) the latest (2.42.0) 64-bit version of Git for Windows. This is the most recent maintained build. It was released 18 days ago, on 2023-08-30.

Other Git for Windows downloads

Standalone Installer

[32-bit Git for Windows Setup](#).

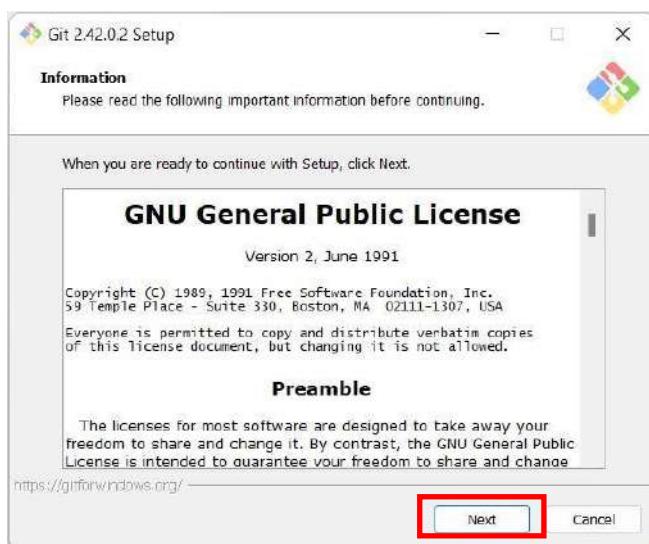
[64-bit Git for Windows Setup](#).

Portable ("thumbdrive edition")

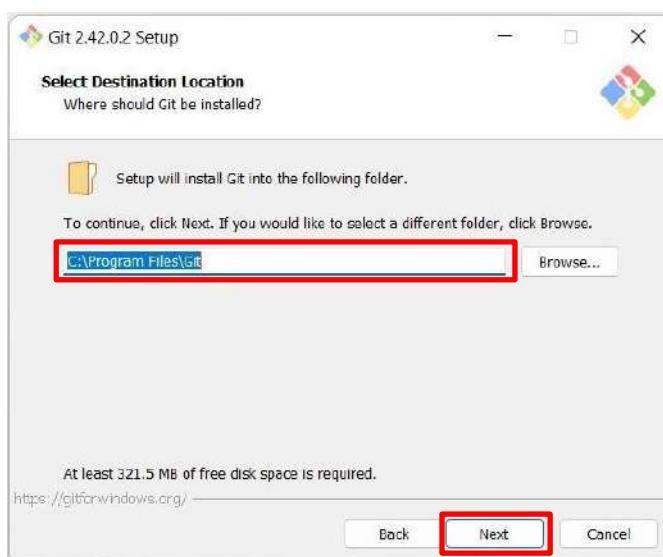
[32-bit Git for Windows Portable](#).

[64-bit Git for Windows Portable](#).

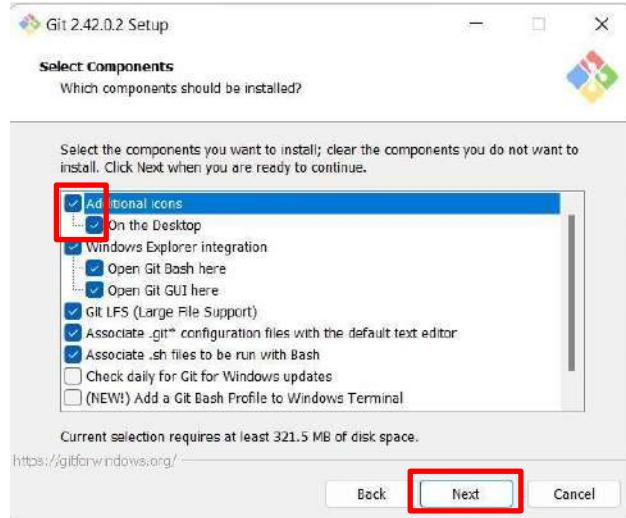
Step 4: Click on the downloaded file & click on 'Next'.



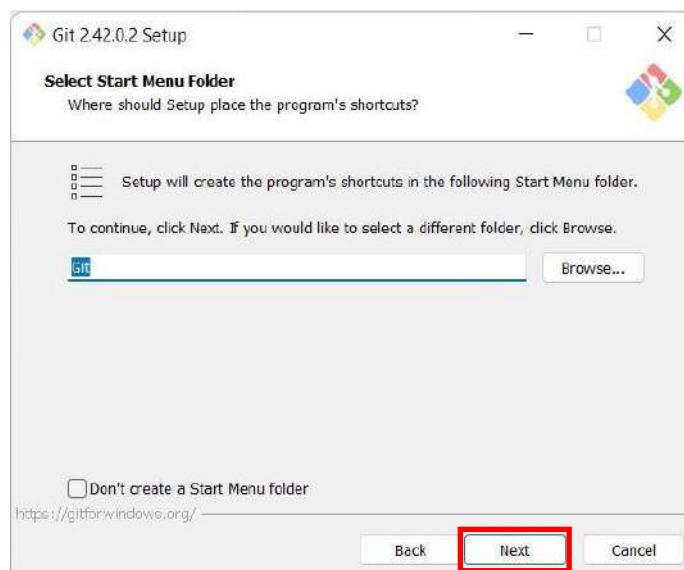
Step 5: Choose the file location to save this file & click on 'Next'.



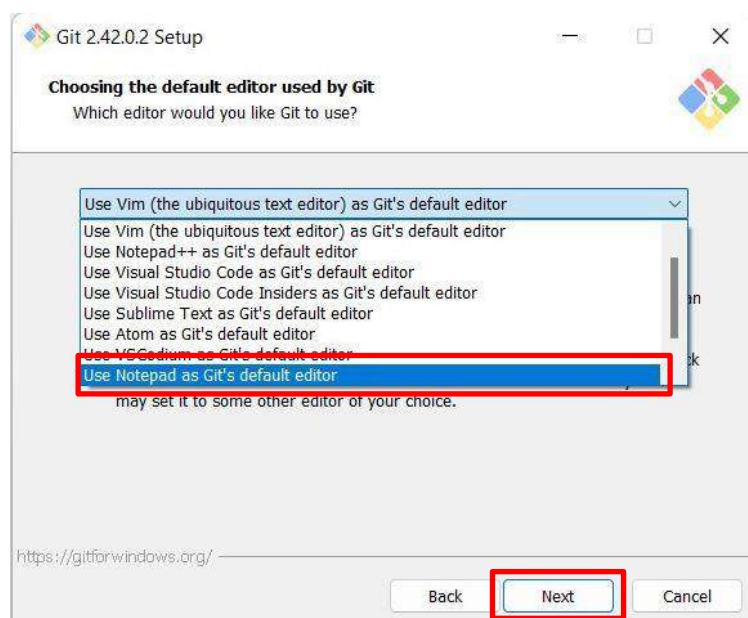
Step 6: Enable the top 2 checkboxes & click on 'Next'.



Step 7: Then click on ‘Next’.

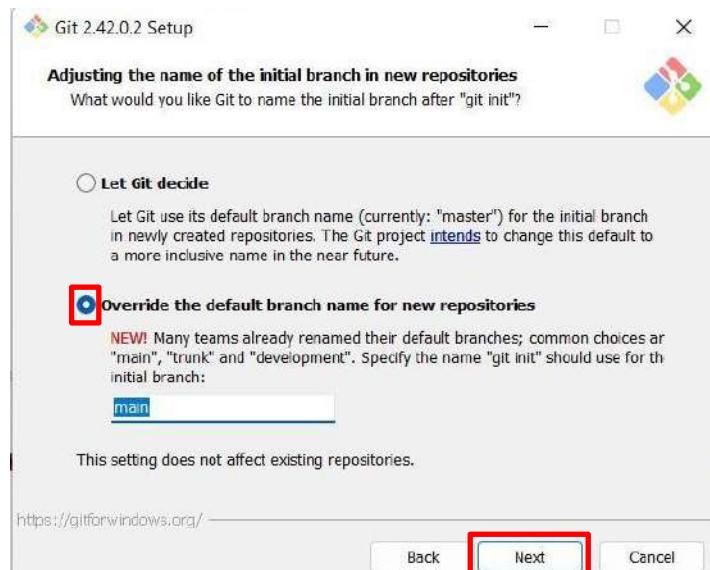


Step 8: Now select the ‘Use Notepad ad Git’s default editor’ option in the dropdown list & click on ‘Next’.

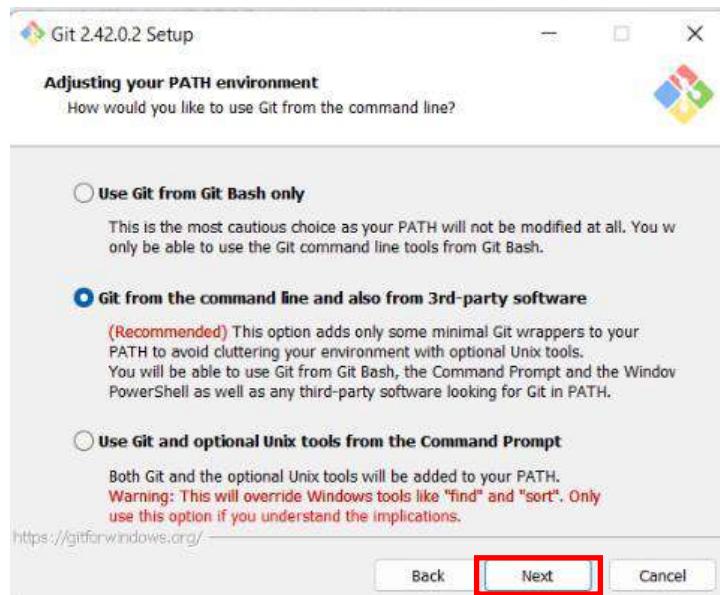


Step 9: Enable the 2nd checkbox ‘Override the default branch name for new repositories’ & click on

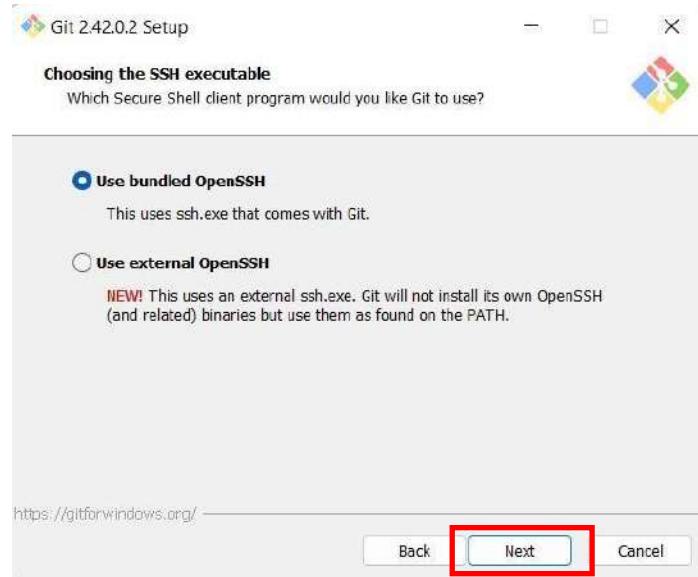
'Next';



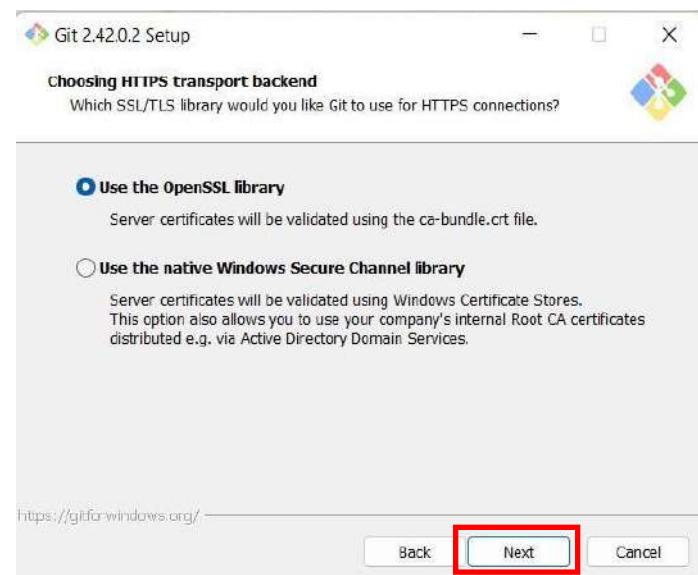
Step 10: Now click on 'Next'.



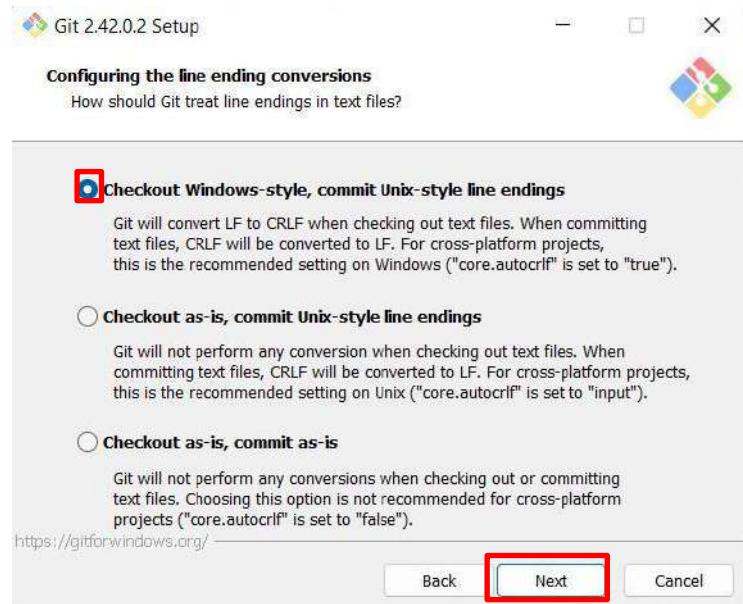
Step 11: Click on 'Next'.



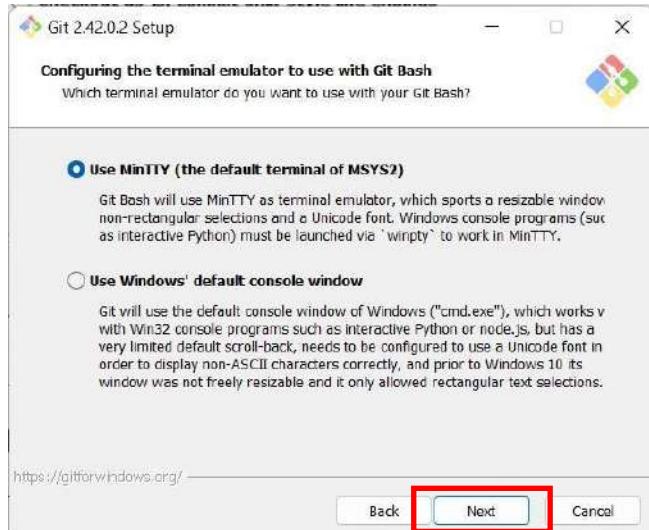
Step 12: Click on 'Next'.



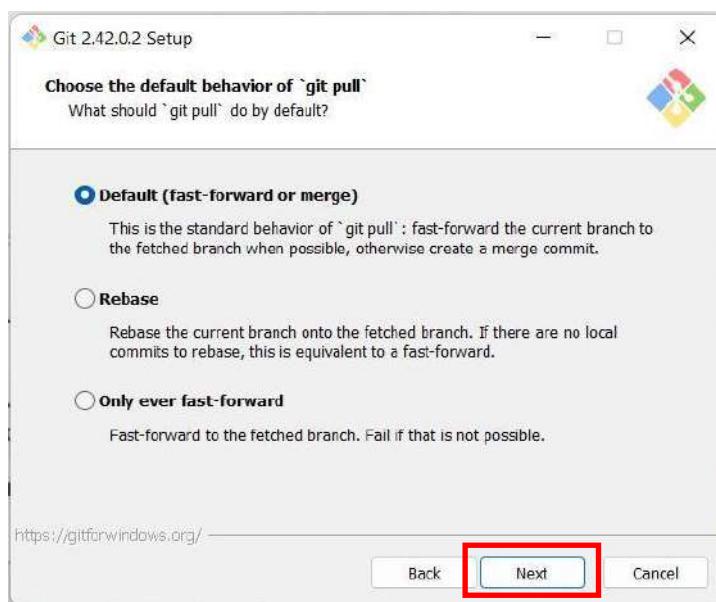
Step 13: Enable the 1st 'Checkout Windows-Style, Commit Unix-Style line endings' & click on 'Next'.



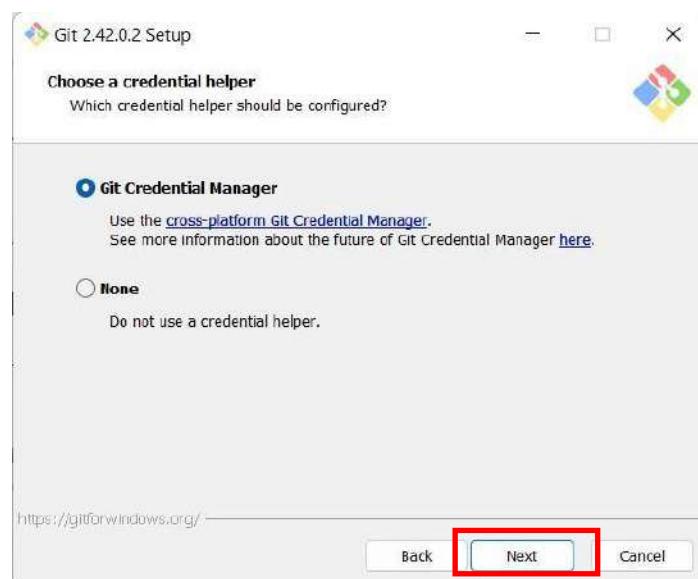
Step 14: Now enable the first option & click on ‘Next’.



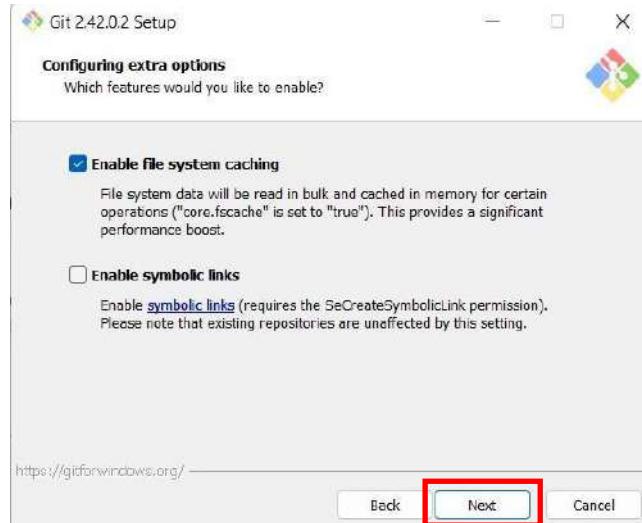
Step 15: Then enable the first option & click on ‘Next’.



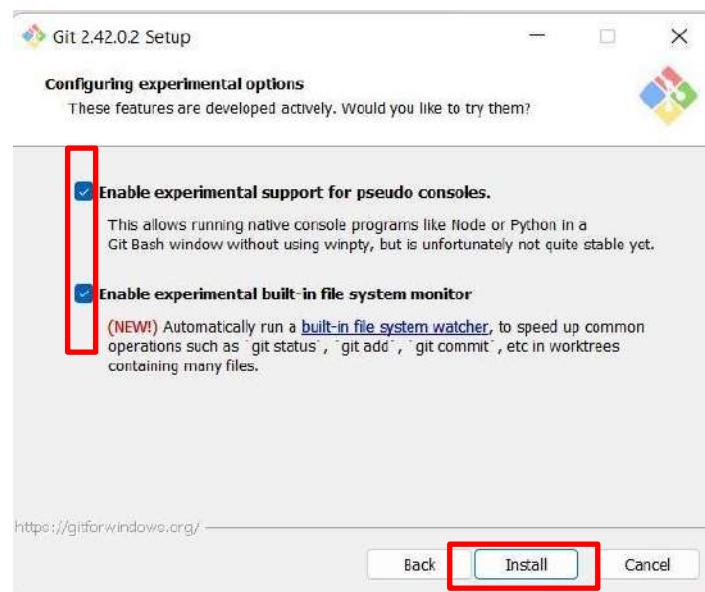
Step 16: Enable the first option & click on ‘Next’.



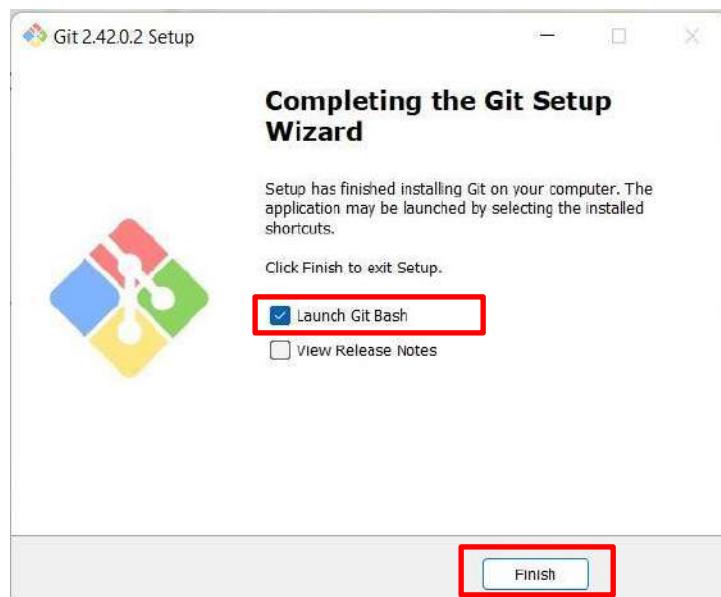
Step 17: Enable the first option & click on ‘Next’.



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



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



- **Creating a repository**

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

```
hk923@HK MINGW64 ~/OneDrive/Desktop/bash1
$ git init
Initialized empty Git repository in C:/Users/hk923/OneDrive/Desktop/bash1/.git/
```

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

```
hk923@HK MINGW64 ~/OneDrive/Desktop/bash1 (main)
$ ls -al
total 29
drwxr-xr-x 1 hk923 197609 0 Sep 20 18:51 .
drwxr-xr-x 1 hk923 197609 0 Sep 20 18:50 ..
drwxr-xr-x 1 hk923 197609 0 Sep 20 18:51 .git/
-rw-r--r-- 1 hk923 197609 20 Sep 20 18:51 example.txt
```

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

```
hk923@HK MINGW64 ~/OneDrive/Desktop/bash1 (main)
$ git status
On branch main
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    example.txt
nothing added to commit but untracked files present (use "git add" to track)
```

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

```
hk923@HK MINGW64 ~/OneDrive/Desktop/bash1 (main)
$ git add example.txt
```

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

```
hk923@HK MINGW64 ~/OneDrive/Desktop/bash1 (main)
$ git status
On branch main
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   example.txt
```

Step 25: Enter the command ‘**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
[main (root-commit) 7201c3c] First Commit
 1 file changed, 1 insertion(+)
 create mode 100644 example.txt
```

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.

```
hk923@HK MINGW64 ~/OneDrive/Desktop/bash1 (main)
$ git config --global user.email "brindakrishnamurthy476@gmail.com"
hk923@HK MINGW64 ~/OneDrive/Desktop/bash1 (main)
$ git config --global user.name "Brinda"
```

Step 27: Enter ‘**git commit -m “First Commit” example.txt**’ to commit the file.

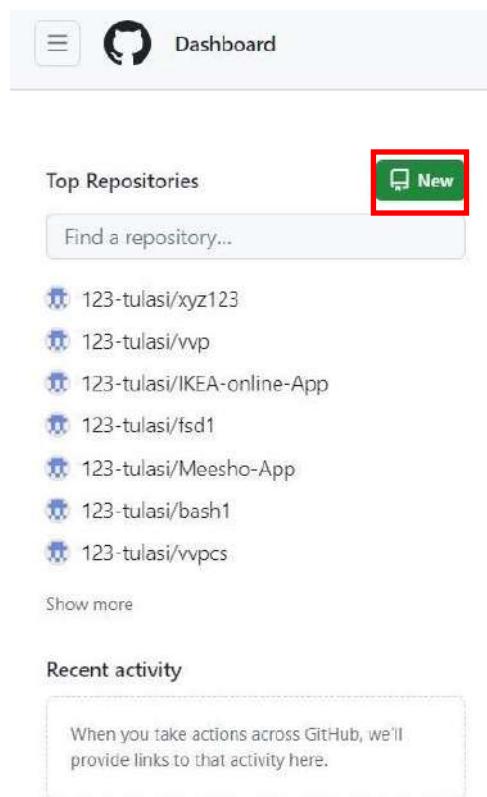
```
hk923@HK MINGW64 ~/OneDrive/Desktop/bash1 (main)
$ git commit -m "First Commit" example.txt
On branch main
nothing to commit, working tree clean
```

Step 28: Then enter ‘git log’ to view the committed file.

```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (main)
$ git log
commit 7201c4c4ea9f12e2f2b6cf50e25312fa1cbb503d (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



Create a new repository

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

Required fields are marked with an asterisk (*).

Owner *



Repository name *

gitbash1

gitbash1 is available.

Great repository names are short and memorable. Need inspiration? How about friendly-fishstick ?

Description (optional)

First Commit on Git Bash

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

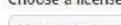
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore



Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license



A license tells others what they can and can't do with your code. Learn more about licenses.

① You are creating a public repository in your personal account.

Create repository

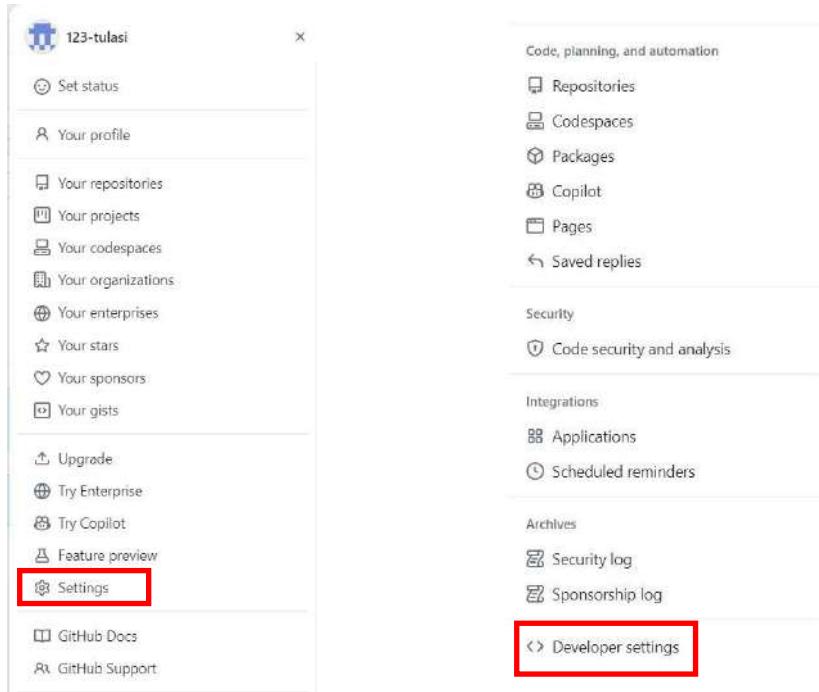
Step 30: Now copy the GitHub location and open it in New tab.

The screenshot shows a web browser window with the GitHub URL `github.com/123-tulasi/gitbash1` highlighted in red. The page content includes:

- Set up GitHub Copilot**: Use GitHub's AI pair programmer to autocomplete suggestions as you code. [Get started with GitHub Copilot](#)
- Add collaborators to this repository**: Search for people using their GitHub username or email address. [Invite collaborators](#)
- Quick setup — if you've done this kind of thing before**:
 - [Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/123-tulasi/gitbash1.git>
 - Get started by creating a new file or uploading an existing file. We recommend every repository include a `README`, `LICENSE`, and `.gitignore`.
- ...or create a new repository on the command line**:

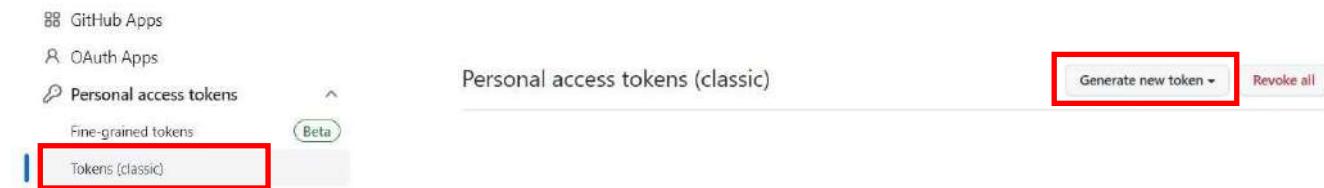
```
echo "# gitbash1" >> README.md
git init
git add README.md
git commit -m "First commit"
git branch -M main
git remote add origin https://github.com/123-tulasi/gitbash1.git
git push -u origin main
```

Step 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 then click on ‘Generate Token’.

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

New Token

What's this token for?

Expiration *

30 days

The token will expire on Fri, Oct 20 2023

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input checked="" type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> admin:enterprise	Full control of enterprises
<input type="checkbox"/> manage_runners:enterprise	Manage enterprise runners and runner groups
<input type="checkbox"/> manage_billing:enterprise	Read and write enterprise billing data
<input type="checkbox"/> readenterprise	Read enterprise profile data
<input checked="" type="checkbox"/> audit_log	Full control of audit log
<input type="checkbox"/> read:audit_log	Read access of audit log
<input checked="" type="checkbox"/> codespace	Full control of codespaces
<input type="checkbox"/> codespace:secrets	Ability to create, read, update, and delete codespace secrets
<input type="checkbox"/> copilot	Full control of GitHub Copilot settings and seat assignments
<input type="checkbox"/> manage_billing:copilot	View and edit Copilot for Business seat assignments
<input checked="" type="checkbox"/> project	Full control of projects
<input type="checkbox"/> read:project	Read access of projects
<input checked="" type="checkbox"/> admin:gpg_key	Full control of public user GPG keys
<input type="checkbox"/> write:gpg_key	Write public user GPG keys
<input type="checkbox"/> read:gpg_key	Read public user GPG keys
<input checked="" type="checkbox"/> admin:ssh_signing_key	Full control of public user SSH signing keys
<input type="checkbox"/> write:ssh_signing_key	Write public user SSH signing keys
<input type="checkbox"/> read:ssh_signing_key	Read public user SSH signing keys

Generate token

Cancel

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

Personal access tokens (classic)

[Generate new token](#)[Revoke all](#)

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your personal access token now. You won't be able to see it again!

✓ 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 

[Delete](#)

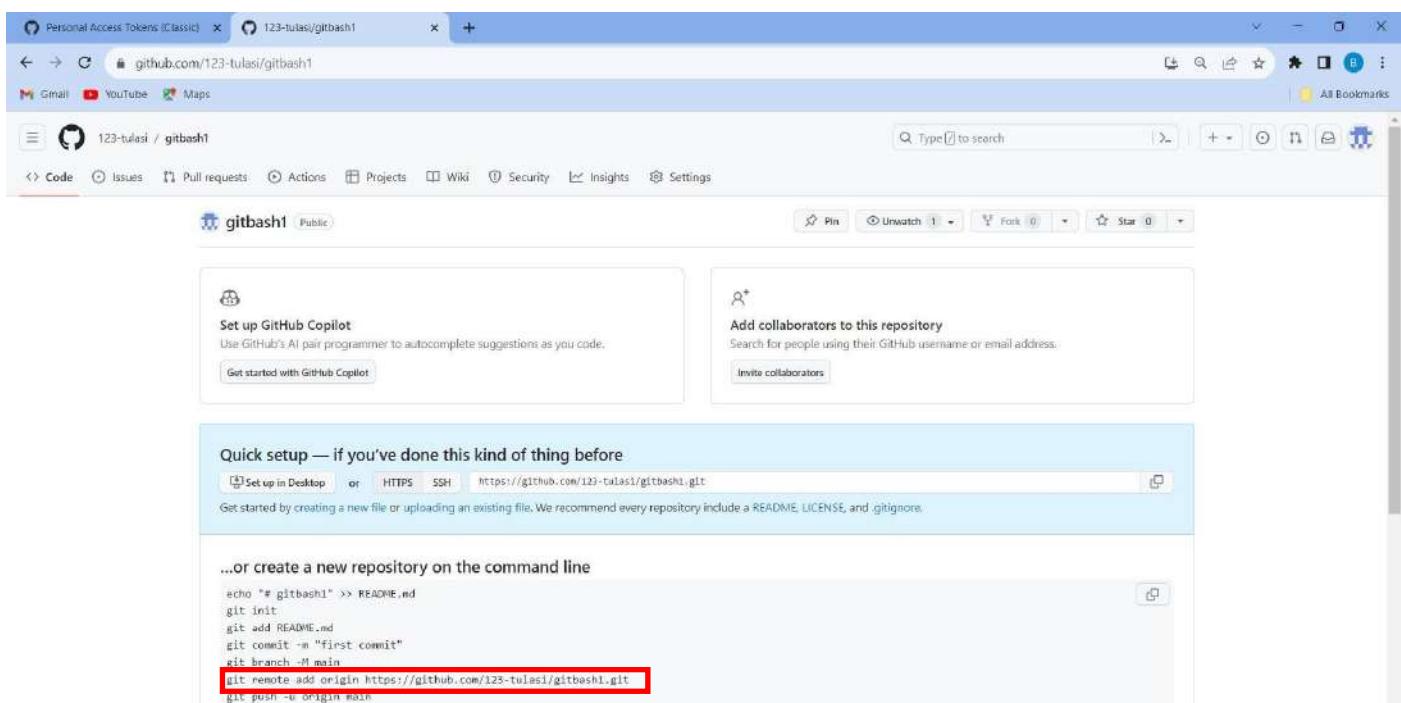
New first token — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, delete:packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:packages

Expires on **Thu, Oct 19 2023**.

First new Token — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, delete:packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:packages

Expires on **Thu, Oct 19 2023**.

Step 35: Also copy the below command '**git remote add origin https://github.com/tulasi/gitbash1.git**' & paste that link in the same notepad.



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

```
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
git remote add origin https://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@github.com/123-tulasi/gitbash1
```

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

```
hk923@HK MINGW64 ~/OneDrive/Desktop/gitbash1 (main)
$ git remote add origin https://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@github.com/123-tulasi/gitbash1
```

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
```

- **Creating a new branch and adding new text files.**

Step 41: Give ‘git branch newbranch1’ to create a new branch. And ‘git checkout newbranchd1’ to switch 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 870beb] 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 870beb376a3eadf46ca2ca1254d9c451e4ce79 (HEAD -> main, newbranch1)
Author: Brinda sbrindakrishnamurthy476@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.



- **Working in GitHub**

GitHub: GitHub, Inc. is a platform and cloud-based service for software development and version control using 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.

The screenshot shows the GitHub Home page. On the left, there's a sidebar with 'Top Repositories' and a 'New' button highlighted with a red box. The main area has sections for 'Updates to your homepage feed' and 'Start writing code'. Under 'Start writing code', there's a 'Start a new repository' section where you can name it '123-tulasi / name your new repository...'. It offers two options: 'Public' (Anyone on the internet can see this repository) and 'Private' (You choose who can see and commit to this repository). Below that is a 'Introduce yourself with a profile README' section with a sample README file and a 'Create' button. On the right, there's a 'Latest changes' sidebar with several recent updates. A blue banner for 'UNIVERSE23' is visible at the top right.

Step 2: Give a name to your repository & some description. Then enable ‘Add a README file’ and click on ‘Create repository’.

The screenshot shows the 'Create a new repository' form. The 'Repository name' field is set to 'Ikea app1234' and is highlighted with a red box. The 'Description (optional)' field contains 'IKEA is an online retail manufacturing company.' The 'Public' radio button is selected. The 'Initialize this repository with:' section has the 'Add a README file' checkbox checked and is also highlighted with a red box. At the bottom, the 'Create repository' button is highlighted with a red box.

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

The screenshot shows a GitHub repository page for 'Ikea-app1234'. At the top, there are navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the header, there's a search bar and a pin/unwatch/fork/star button. The main area shows a 'main' branch with 1 commit. A red box highlights the 'Add file' button, which has a dropdown menu showing 'Create new file' and 'Upload files'. To the right, there's an 'About' section for IKEA, a 'Releases' section with no releases published, and a sidebar with repository statistics.

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

The screenshot shows the GitHub code editor for the 'app' file in the 'main' branch. The code editor contains a snippet of shell commands. A red box highlights the 'Commit changes...' button at the top right of the editor. Other buttons visible include 'Edit', 'Preview', and 'Cancel changes'.

Step 5: Add the extended description and click on ‘Commit changes’.

The screenshot shows the 'Commit changes' dialog box. It includes fields for 'Commit message' (with a placeholder 'Create app') and 'Extended description' (containing the text 'Creating a new text file with using commands.'). At the bottom, there are two radio button options: 'Commit directly to the main branch' (selected) and 'Create a new branch for this commit and start a pull request'. Below these options is a link 'Learn more about pull requests'. The dialog box also features 'Cancel' and 'Commit changes' buttons, with the latter being highlighted by a red box.

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

- **Using Version control system.**

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 some related files to it. Then press on ‘**Publish release**’.

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

A screenshot of a GitHub release page for 'Version 0.2.6.2'. The page includes a search bar at the top right. Below the title, it shows the release date ('123-tulasi released this now'), version ('v0.2.6'), and commit hash ('13670e8'). A note states, 'The purpose of this version is to enhance a better User interface experience to work in a better manner.' Under the 'Assets' section, there are three items listed: 'GitHub-Logo.png' (22.7 kB, now), 'Source code (.zip)' (5 minutes ago), and 'Source code (.tar.gz)' (5 minutes ago). The 'Source code (.zip)' and 'Source code (.tar.gz)' items are highlighted with a red box.

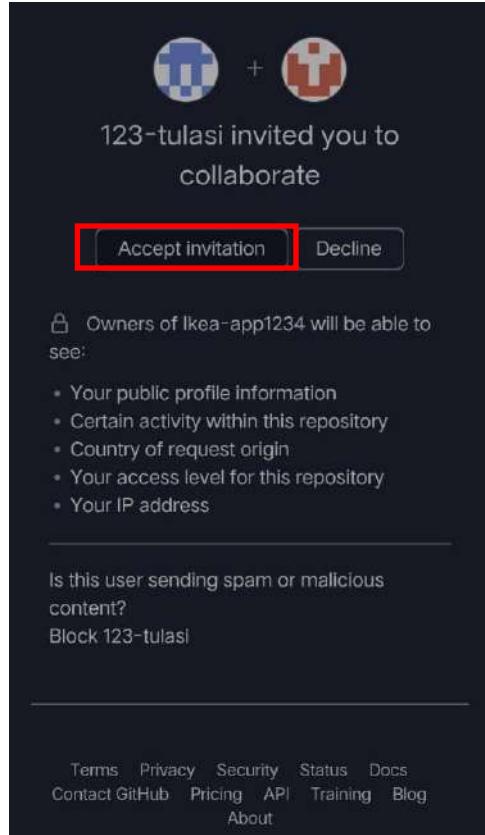
Step 10: Go to Settings □ Collaborators □ Add people.

A screenshot of the GitHub repository settings page for 'ikea-app1234'. The 'Settings' tab is highlighted with a red box. On the left, a sidebar lists various settings sections: General, Access (with 'Collaborators' highlighted with a red box), Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables), and Integrations (GitHub Apps, Email notifications). The main area is titled 'Who has access' and shows 'PUBLIC REPOSITORY' status with 'DIRECT ACCESS' (0 collaborators). Below this is a 'Manage access' section with a message 'You haven't invited any collaborators yet' and a green 'Add people' button, which is also highlighted with a red box.

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

A screenshot of a modal dialog box titled 'Add a collaborator to Ikea-app1234'. It contains an input field with the email address 'rakshithamr705@gmail.com' and a large green button at the bottom labeled 'Add rakshithamr705@gmail.com to this repository', which is highlighted with a red box.

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



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

Ikea-app1234 Public

123-tulasi / Ikea-app1234

Type [] to search

Code Issues Pull requests Actions Projects Wiki Security Insights

Watch Fork Star

main 1 branch 1 tag

123-tulasi Create app 1367bb8 28 minutes ago 2 commits

README.md Initial commit 33 minutes ago

app Create app 26 minutes ago

README.md

Ikea-app1234

IKEA is an online retail manufacturing company.

About

IKEA is an online retail manufacturing company.

Readme Activity 0 stars 1 watching 0 forks Report repository

Releases 1

Version 0.2.6.2 Latest 23 minutes ago

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

Amazon Web Service (AWS)

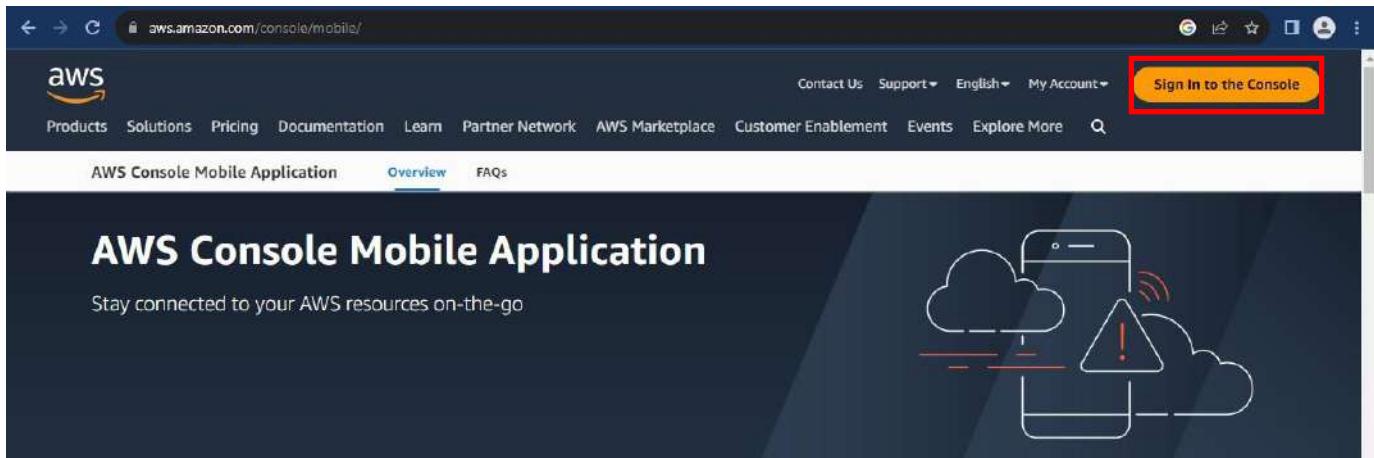
Amazon Web Services, Inc. (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay-as-you-go basis. Clients will often use this in combination with autoscaling (a process that allows a client to use more computing

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

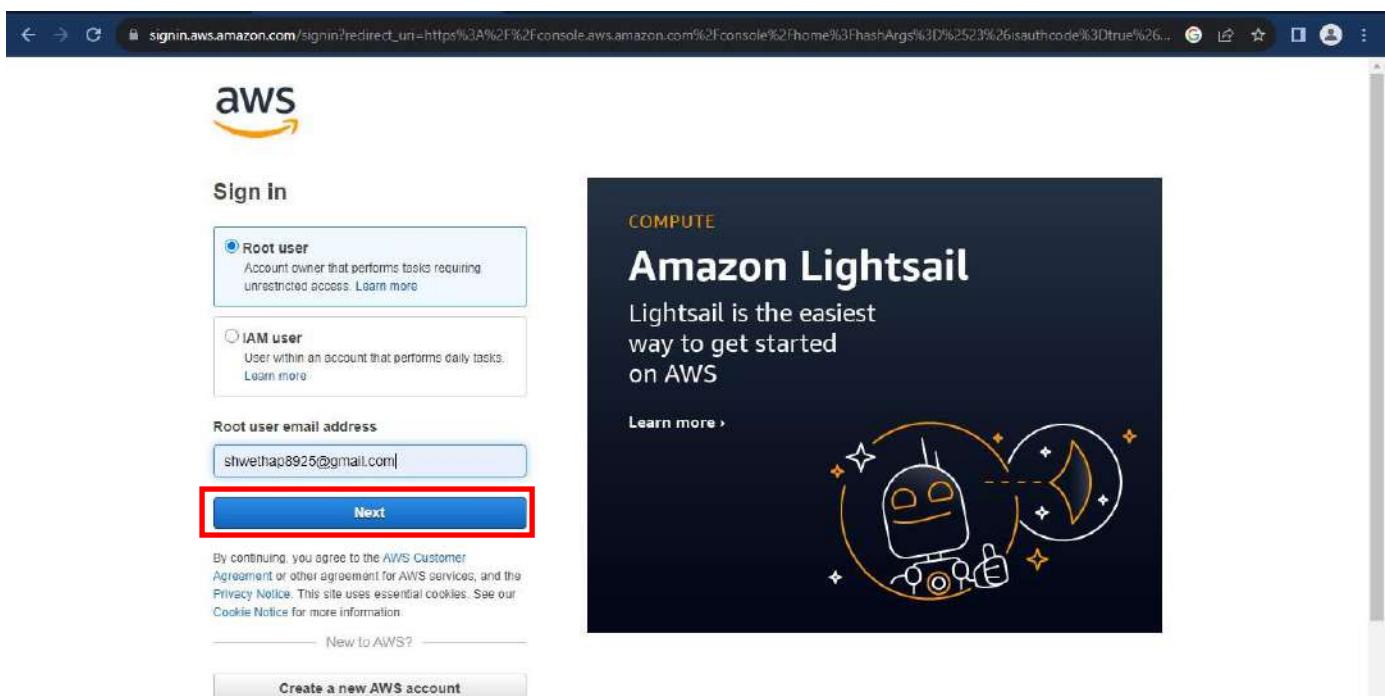
- **Create and setup a virtual machine.**

Step 1: Go to web browser and search for AWS console and click on first website provided and create a new AWS 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 the account details and then click on 'Next'.



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

The screenshot shows the AWS Home page with the search bar at the top. Below it, there are two main sections: 'Build a solution' and 'Trusted Advisor'. The 'Build a solution' section contains several items, with the first one, 'Launch a virtual machine With EC2 (2 mins)', highlighted by a red box. The 'Trusted Advisor' section shows a shield icon and a message stating 'No recommendations'.

Step 5: Give any name for the tag.

The screenshot shows the 'Launch an instance' wizard. In the 'Name and tags' step, the 'Name' field contains 'My website' and is highlighted by a red box. There is also a link 'Add additional tags'.

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

The screenshot shows the 'Quick Start' section of the AWS EC2 interface. It displays various operating system options: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE Linux. The 'Windows' option is highlighted by a red box.

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

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

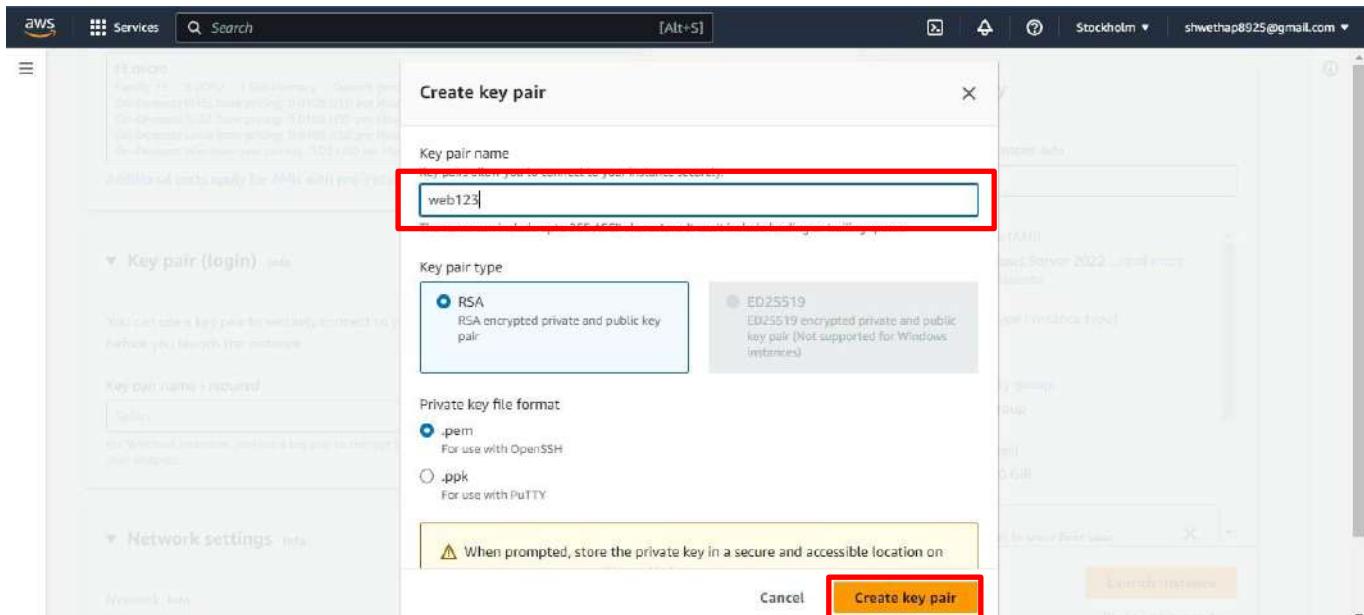
Key pair name - required

Select

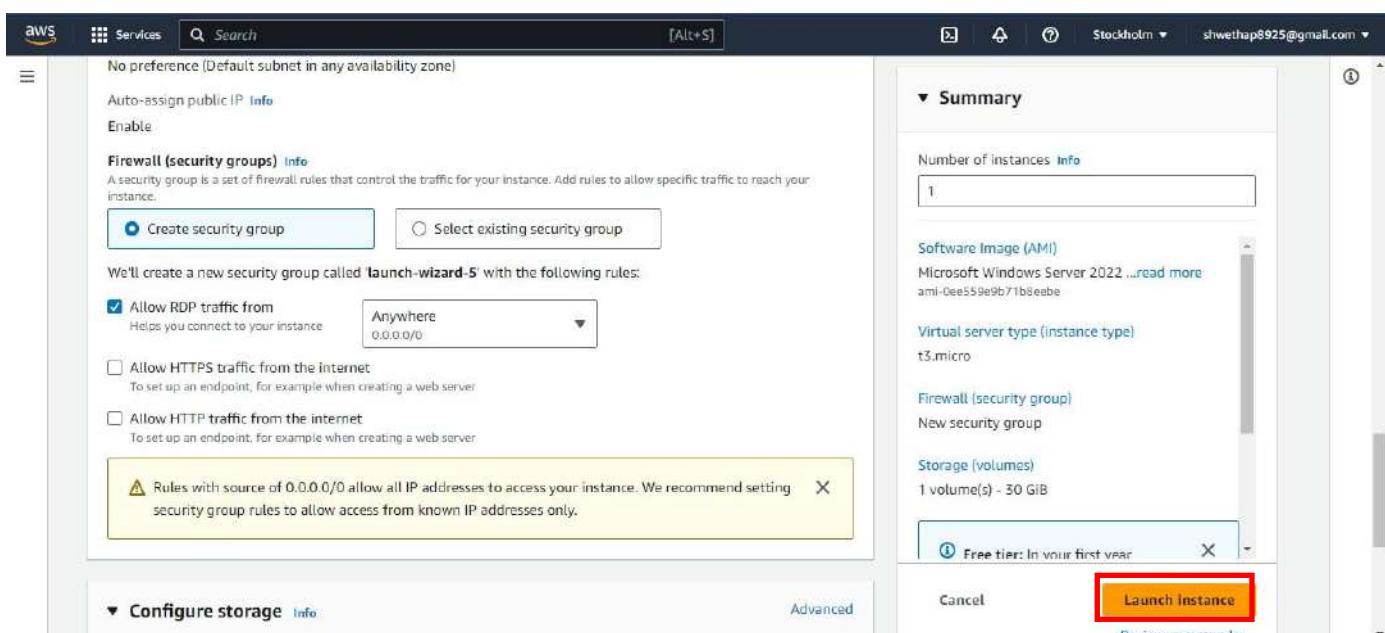
 Create new key pair

For Windows instances, you use a key pair to decrypt the administrator password. You then use the decrypted password to connect to your instance.

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.

The screenshot shows the AWS EC2 Instances page. At the top, there is a green success message: "Successfully initiated launch of instance (i-0cfdd3b275b2edaa0)". Below the message, there is a link to "Launch log".

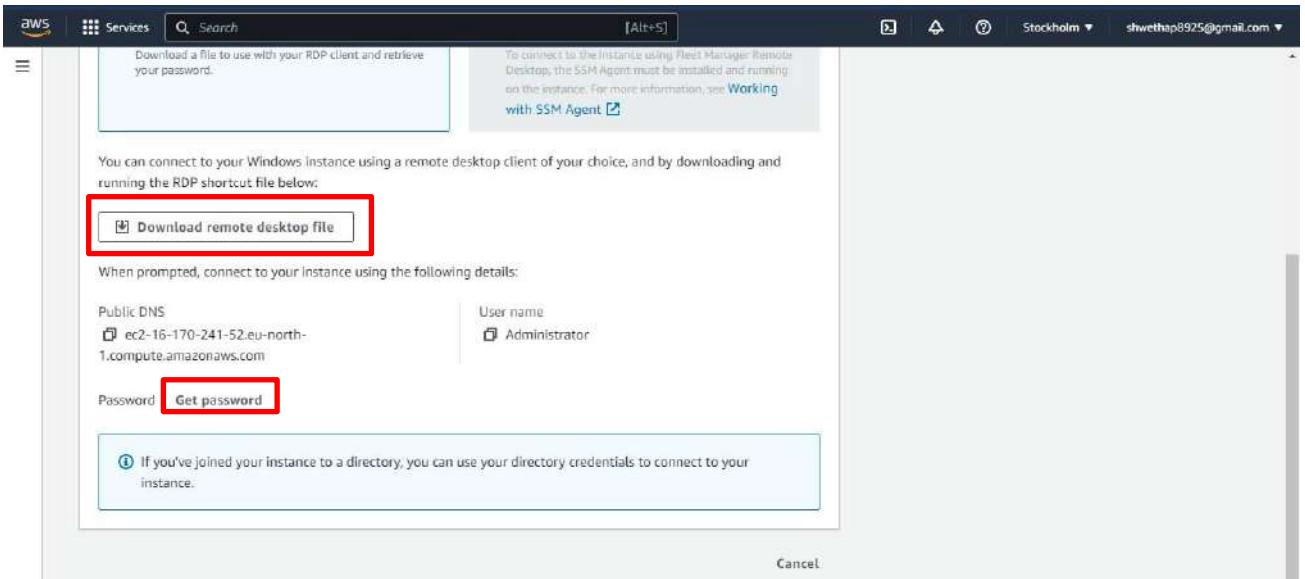
Step 11: Select the key pair created and click on ‘Connect’.

The screenshot shows the AWS EC2 Instances page. A specific instance named "My website" is selected. The "Connect" button, located at the top right of the instance list, is highlighted with a red box. The instance details pane below shows the instance ID, state, type, and other metadata.

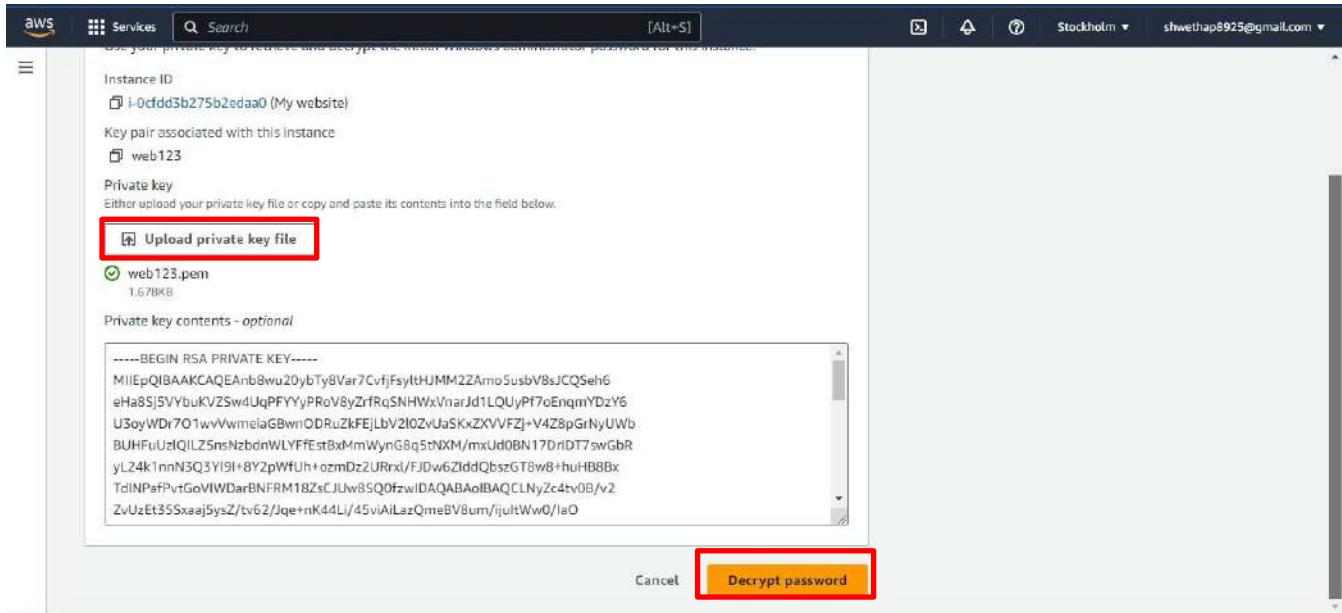
Step 12: Click on ‘RDP Client’.

The screenshot shows the "Connect to instance" page for the selected instance. The "RDP client" tab is highlighted with a red box. The page provides instructions for connecting using an RDP client or Fleet Manager, along with download links for the RDP shortcut file.

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



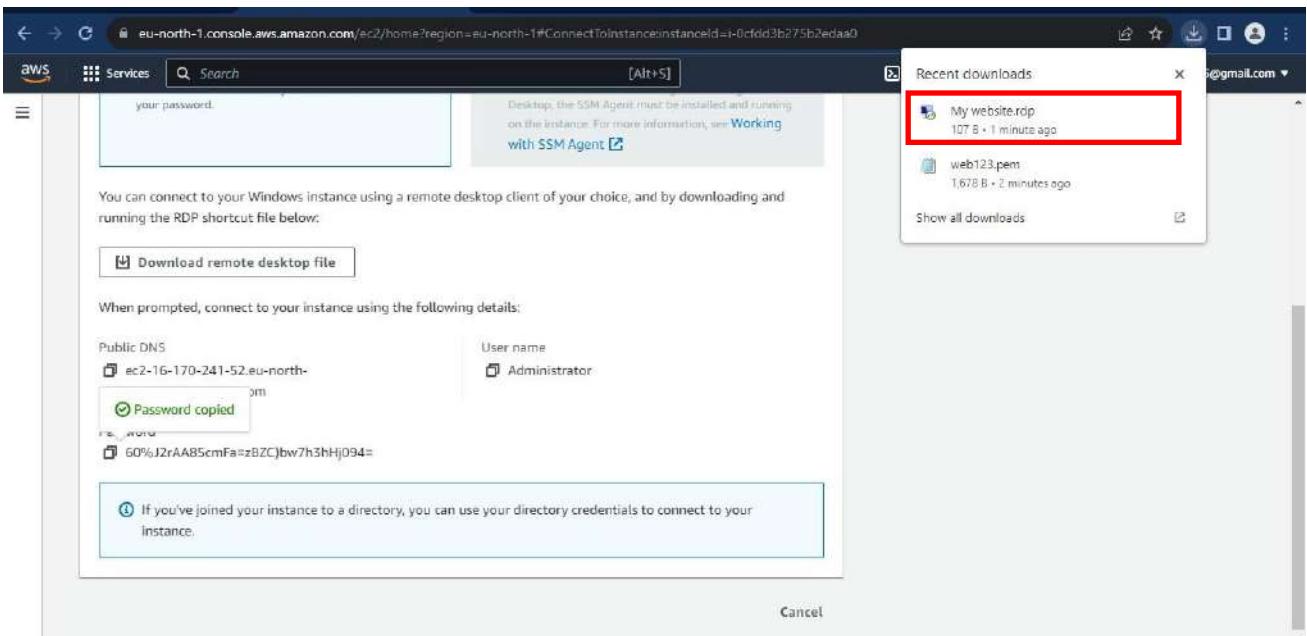
Step 14: Now click on ‘Upload Private Key File’ and open the key pair which was already downloaded. And click on ‘Decrypt password’.



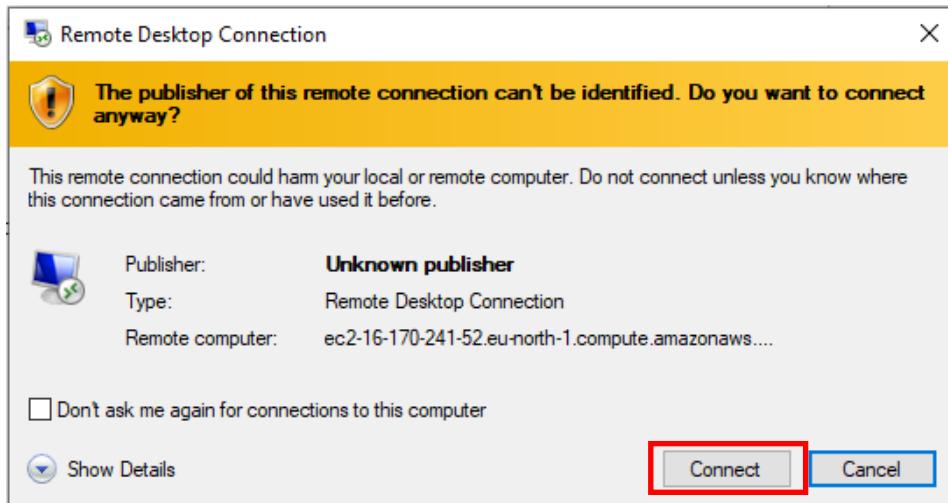
Step 15: Now copy the below password generated.



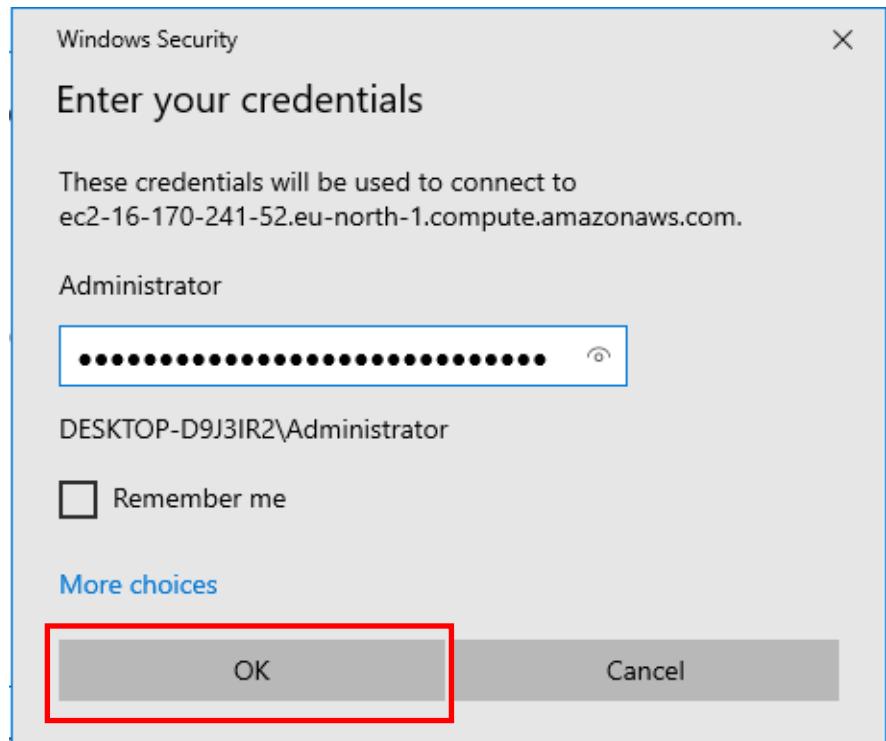
Step 16: Now open the Remote desktop file which was already downloaded.



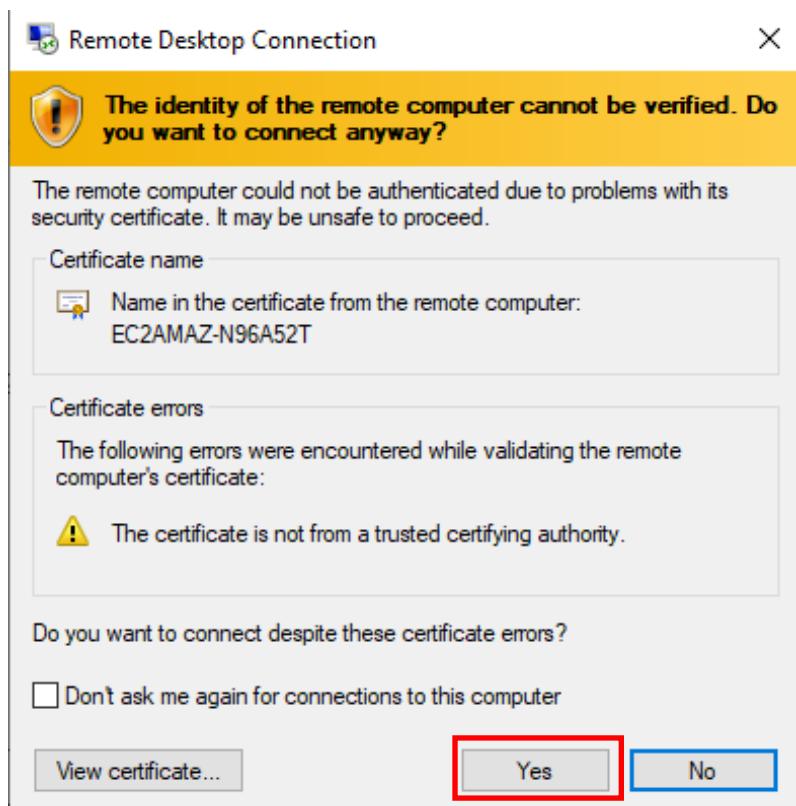
Step 17: Click on 'Connect'.



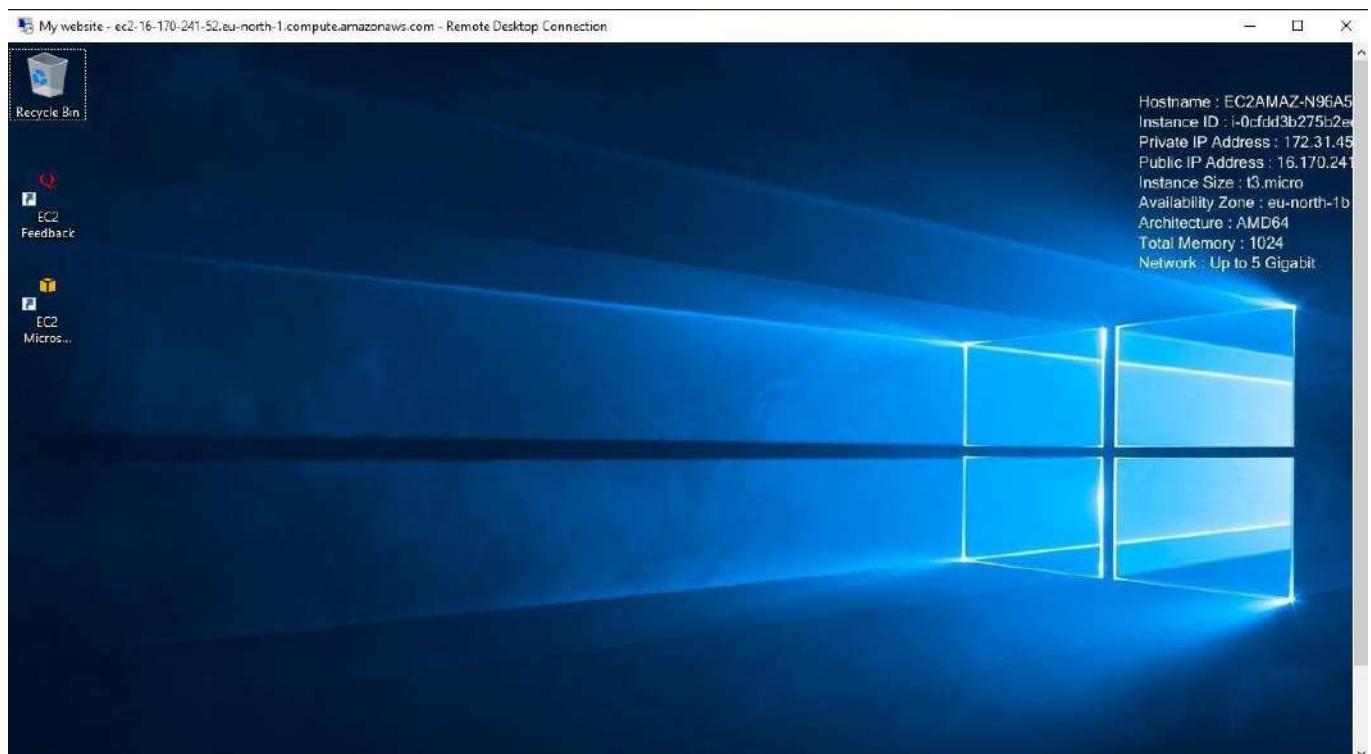
Step 18: Now paste the password which was copied in step15 and click on 'OK'.



Step 19: Click on 'Yes'.



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



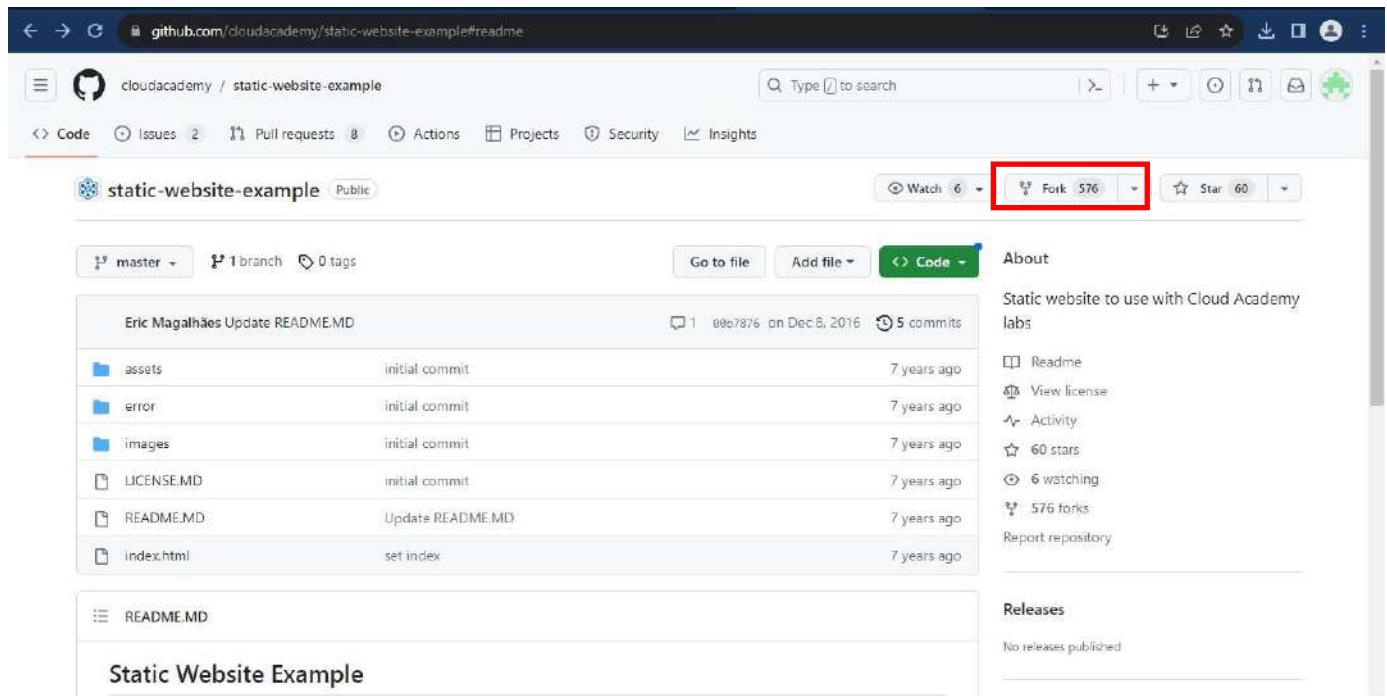
Note: Now terminate the instance created.

A screenshot of the AWS EC2 Instances page. On the left, a navigation pane shows "Instances" is selected. The main table lists two instances: "Name" (bhoomika) and "Instance ID" (i-0a6662b97aca577dd). Both instances are in "Running" state and have "t3.micro" type. The "Actions" dropdown menu for the first instance is open, with "Terminate instance" highlighted by a red box. The "Actions" dropdown for the second instance is also visible.

A screenshot of the "Terminate instance?" confirmation dialog. It contains a warning message about EBS-backed instances and the loss of root volume storage. Below it, a question asks if the user is sure they want to terminate the instances. A table shows the instance ID (i-0a6662b97aca577dd) and termination protection status (Disabled). At the bottom, a note states that instances with termination protection enabled will not be terminated. A "Terminate" button is highlighted by a red box.

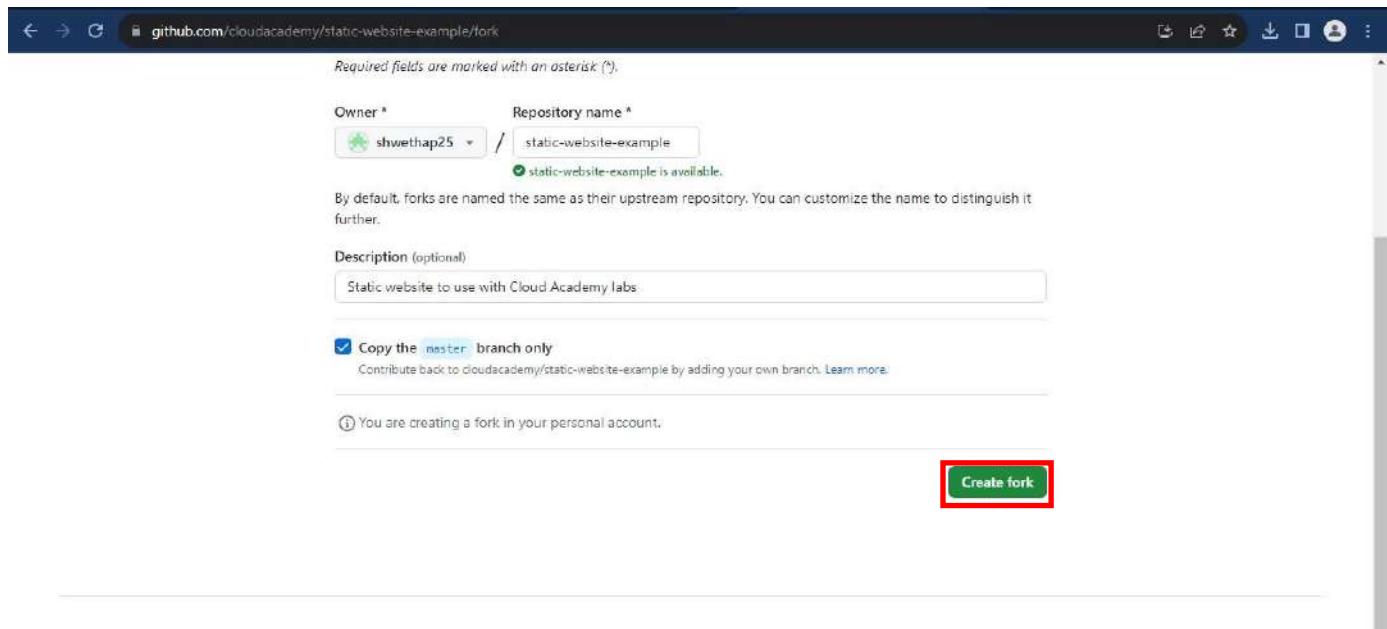
B. Create a simple Web application using Cloud Services.

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



The screenshot shows the GitHub repository page for 'static-website-example'. At the top right, there is a 'Fork' button with a count of 576 forks, which is highlighted with a red box. Below the header, there's a list of files and their commit history. On the right side, there's an 'About' section with details like 'Static website to use with Cloud Academy labs', 'Readme', 'View license', 'Activity', '60 stars', '6 watching', and '576 forks'. There's also a 'Report repository' link. At the bottom left, there's a preview of the 'README.MD' file content: 'Static Website Example'.

Step 22: Click on ‘Create Fork’.



The screenshot shows the 'Create fork' dialog box on GitHub. It asks for the 'Owner' (set to 'shwethap25') and 'Repository name' ('static-website-example'). A note says 'static-website-example is available'. It includes an optional 'Description' ('Static website to use with Cloud Academy labs'), a checked 'Copy the master branch only' checkbox, and a note about contributing back. A warning message at the bottom says 'You are creating a fork in your personal account.' At the bottom right is a large green 'Create fork' button, which is highlighted with a red box.

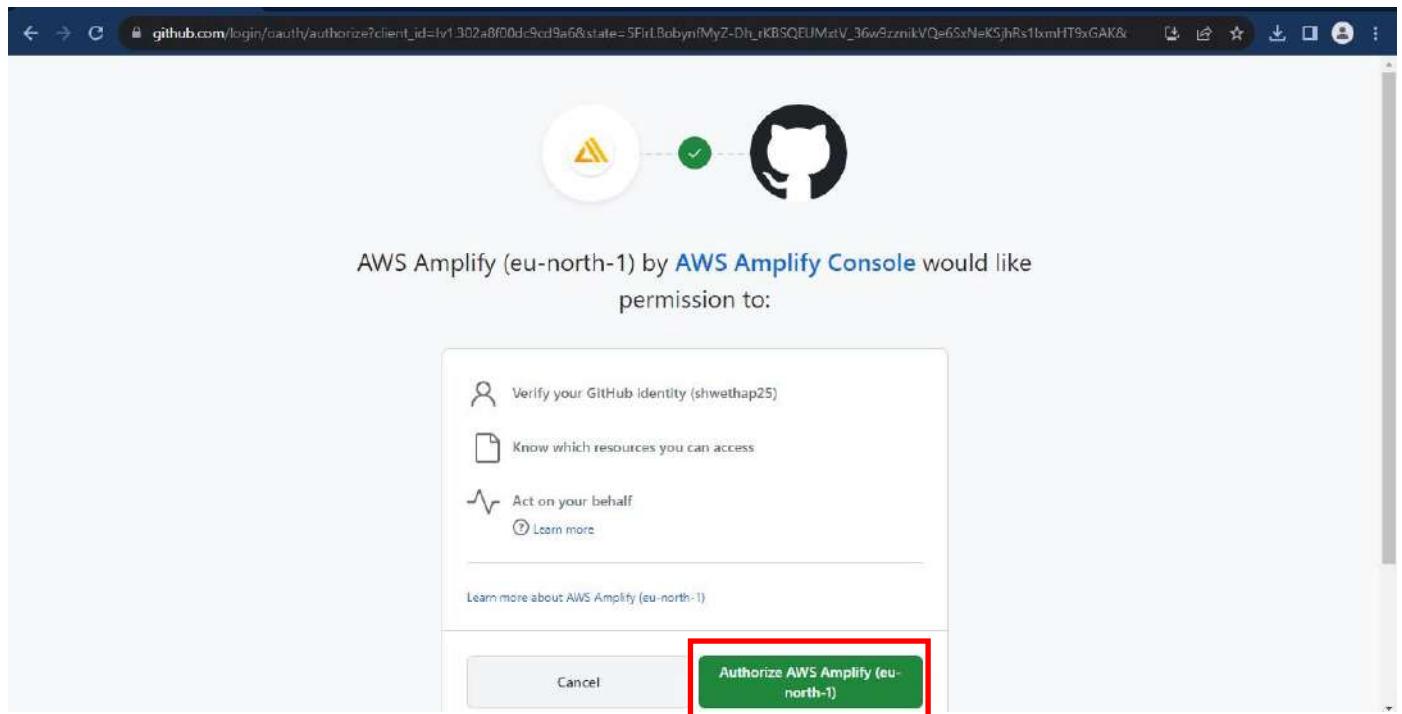
Step 23: Now go back to the AWS home screen and click on ‘Host a static web app’.

The screenshot shows the AWS Amplify console home page. On the left, there's a section titled 'Build a solution' with various options like 'Launch a virtual machine', 'Start a development project', etc. In the center, there's a red box around the 'Host a static web app' option, which is described as 'With AWS Amplify Console (2 mins)'. To the right, there's a 'Trusted Advisor' section with a shield icon and a message stating 'No recommendations'. Below that, it says 'This could be because you have not run Trusted Advisor checks, or you don't have AWS Business or AWS Enterprise support plans.' At the bottom, there are links for 'Go to Trusted Advisor', 'Explore AWS', 'Latest announcements', and 'Recent AWS blog posts'.

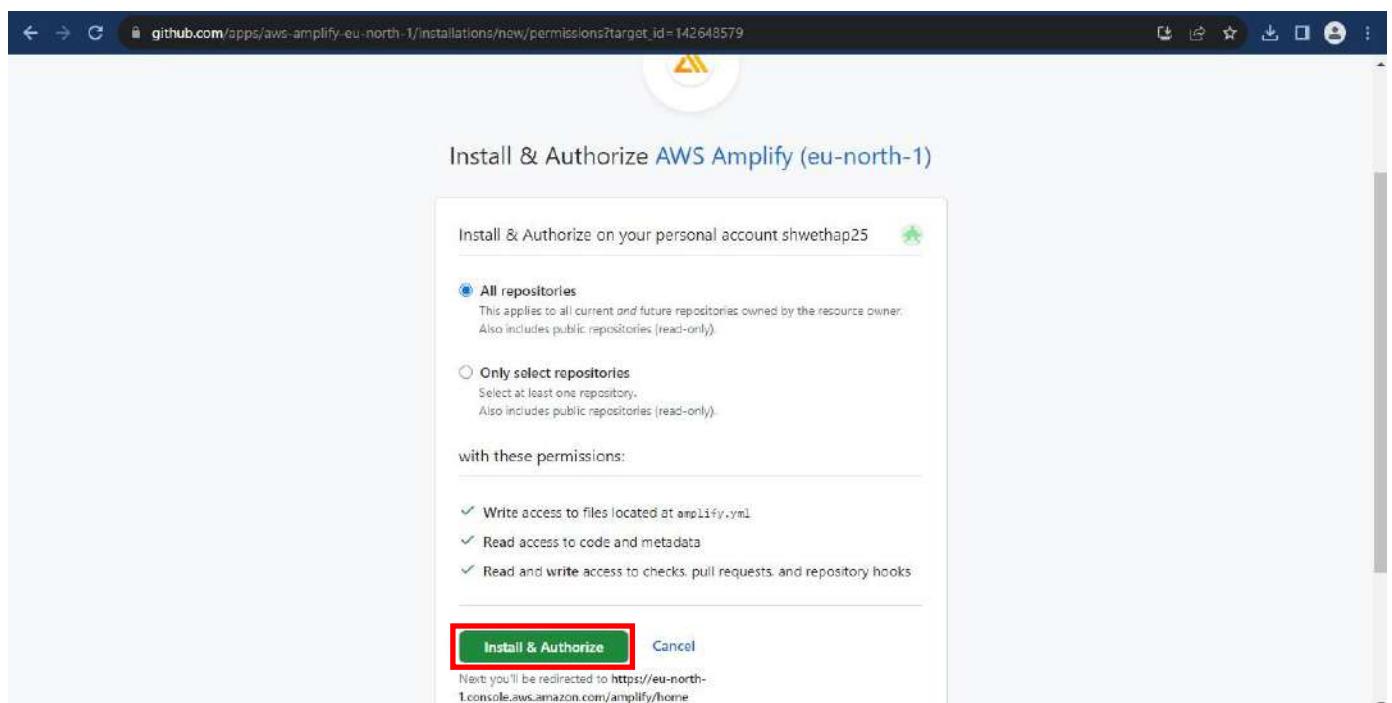
Step 24: Select ‘GitHub’ as the Host web app. And click on ‘Continue’.

The screenshot shows the 'From your existing code' configuration screen in the AWS Amplify console. It lists several options: 'GitHub' (selected), 'Bitbucket', 'GitLab', 'AWS CodeCommit', and 'Deploy without Git provider'. A red box highlights the 'GitHub' option. Below the options, a note says 'Amplify Hosting requires read-only access to your repository.' At the bottom right, there's a yellow 'Continue' button, which is also highlighted with a red box.

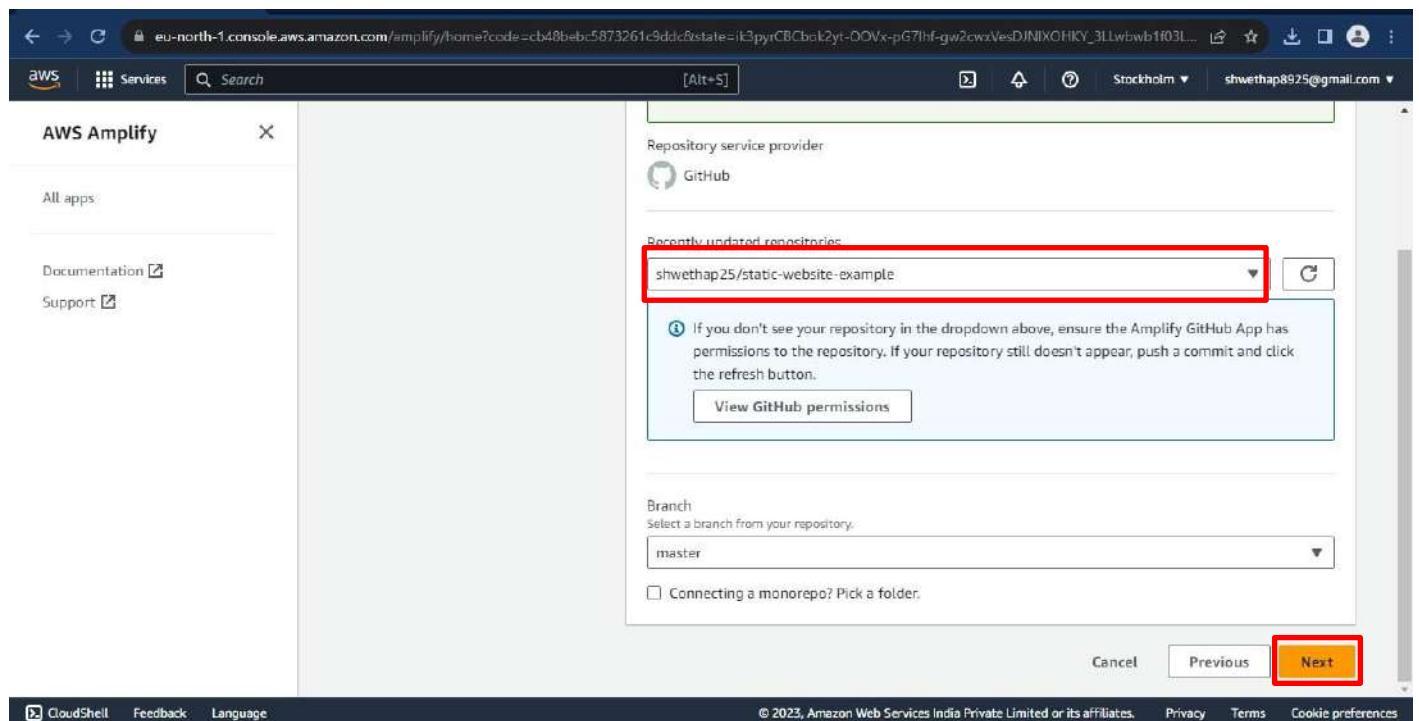
Step 25: Now click on ‘Authorize AWS Amplify (eu-north-1)’.



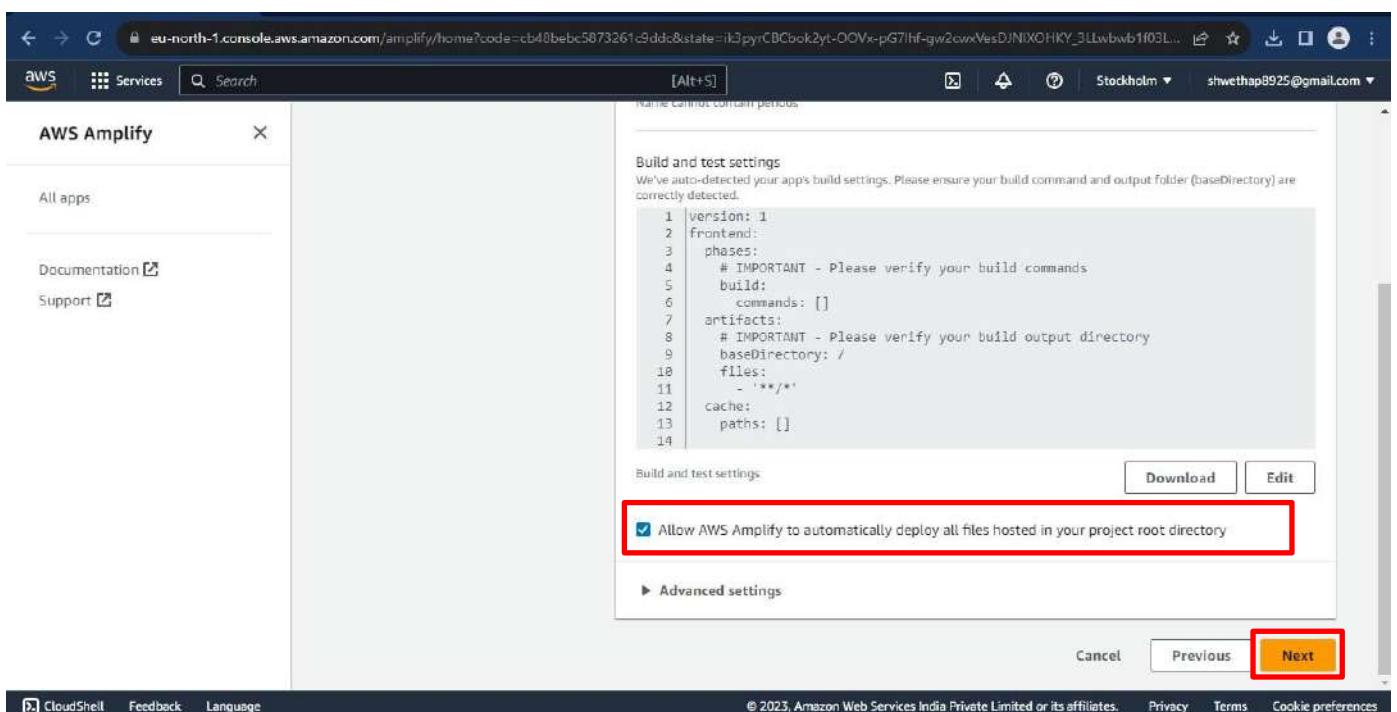
Step 26: Then click on ‘Install & Authorize’.



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’.

The screenshot shows the AWS Amplify Step 3 Review screen. It displays the following details:

- Repository service:** GitHub
- Repository:** shwethap25/static-website-example
- Branch:** master
- Branch environment:** Application root

App settings:

- App name:** static-website-example
- Framework:** Web
- Build image:** Using default image
- Build settings:** Auto-detected settings will be used
- Environment variables:** None

At the bottom right, there are buttons for **Cancel**, **Previous**, and **Save and deploy**. The **Save and deploy** button is highlighted with a red box.

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

The screenshot shows the AWS Amplify Hosting environments screen for the **static-website-example** app. It displays the following information:

- Hosting environments:** master
- Continuous deploys set up (Edit):** A small preview window showing a green checkmark and a smiley face.
- Deployment status:** Provision, Build, Deploy (all three steps are marked with green checkmarks and connected by green arrows). This section is highlighted with a red box.
- Deployment URL:** <https://master..amplifyapp.com> (The URL is highlighted with a red box).
- Last deployment:** 08/09/2023, 11:32:22
- Last commit:** This is an autogenerated message | Auto-build | GitHub - master
- Previews:** Disabled

At the bottom right, there are buttons for **CloudShell**, **Feedback**, **Language**, and **Cookie preferences**. The **CloudShell** button is highlighted with a red box.

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 other users 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 the AWS search bar and click on the first option.

The screenshot shows the AWS Management Console search results for 'iam'. The 'IAM' service card is highlighted with a red box. Other services listed include IAM Identity Center (successor to AWS Single Sign-On), Resource Access Manager, and AWS App Mesh.

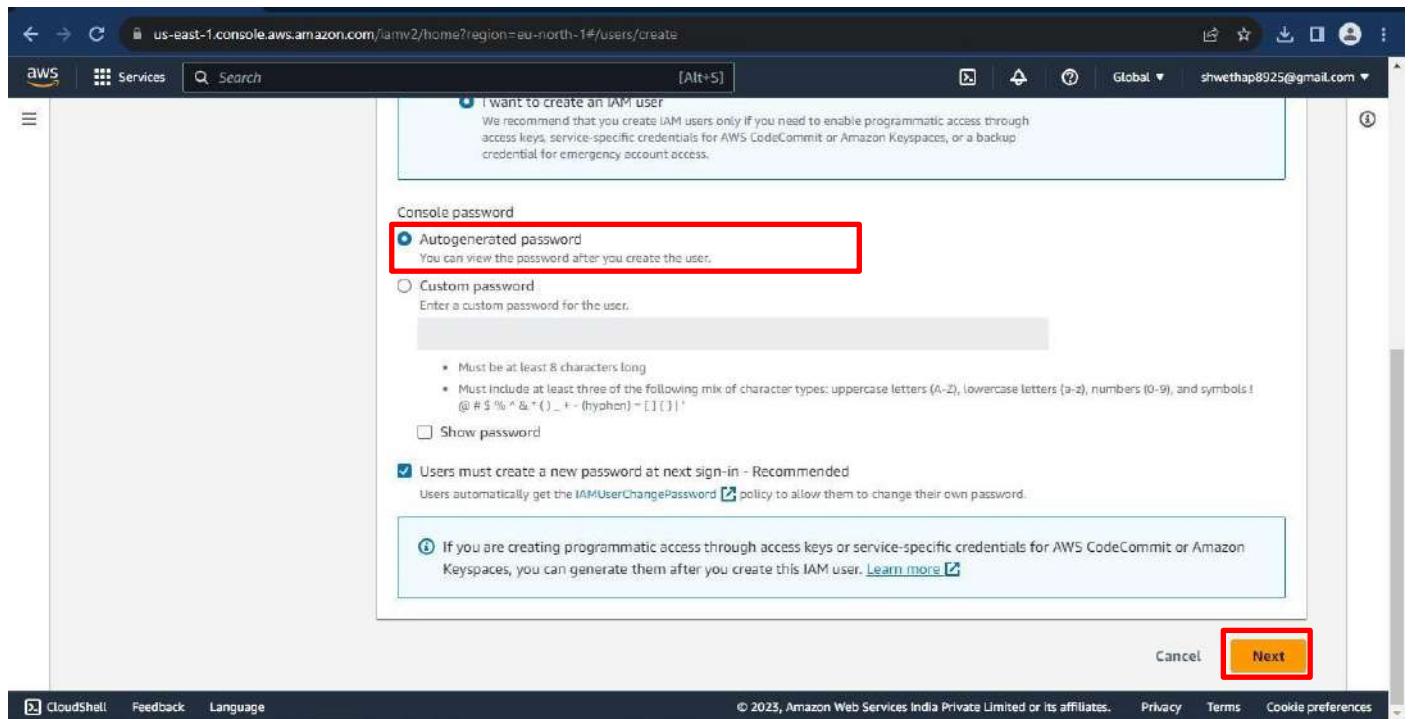
Step 33: Click on ‘Users’ and give the name of the User. Now click on ‘Create User’.

The screenshot shows the AWS IAM service interface. On the left, there's a navigation sidebar with 'Identity and Access Management (IAM)' selected. Under 'Access management', 'Users' is also selected and highlighted with a red box. In the main content area, the 'Users' section is displayed with a search bar containing 'deekshitha' (also highlighted with a red box) and a 'Create user' button (highlighted with an orange box). The results table shows 'No resources to display'.

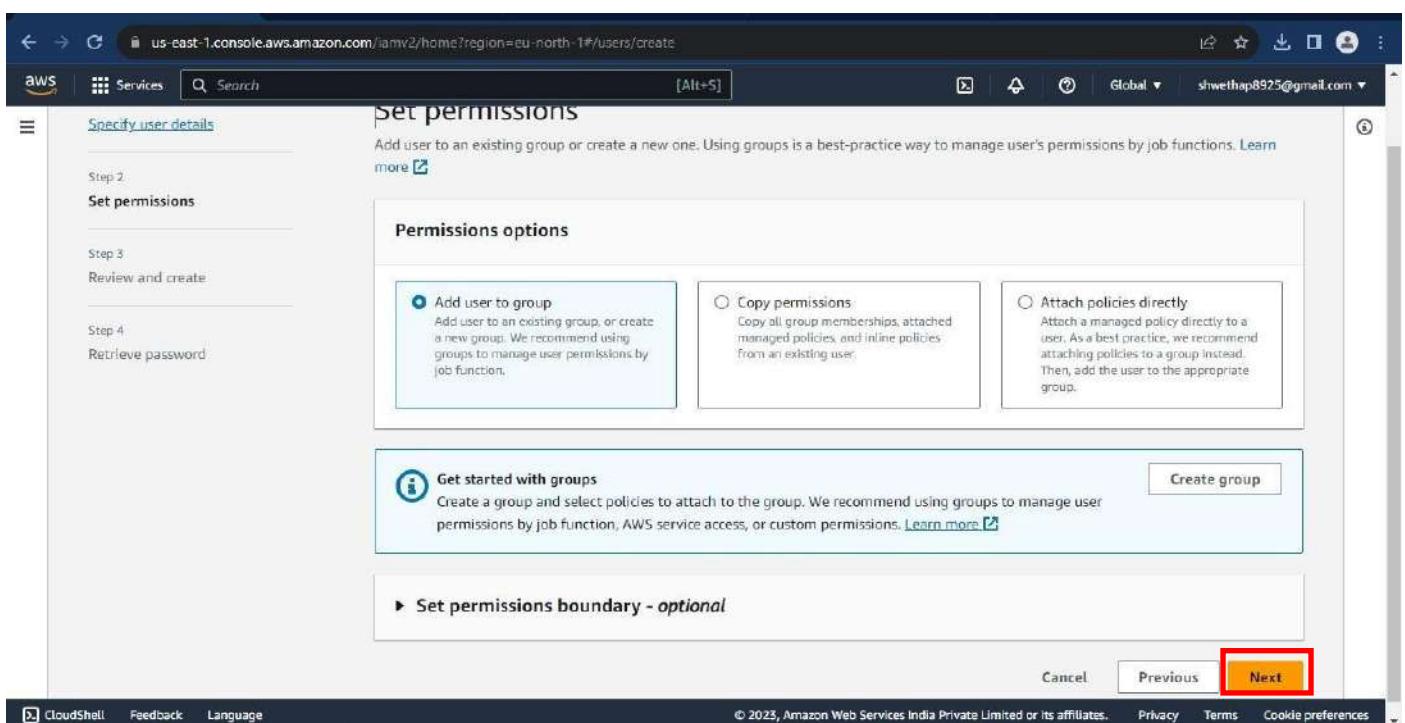
Step 34: Now select ‘I want to create an IAM user’,

The screenshot shows the 'Specify user details' step of the IAM user creation wizard. On the left, a sidebar lists steps: Step 1 (selected and highlighted with a red box), Step 2, Step 3, and Step 4. The main area is titled 'Specify user details' and contains a 'User details' section. It shows a 'User name' input field with 'deekshitha' (highlighted with a red box) and a note about character restrictions. A checkbox for 'Provide user access to the AWS Management Console - optional' is checked. Below it, a question 'Are you providing console access to a person?' has two options: 'Specify a user in Identity Center - Recommended' (unchecked) and 'I want to create an IAM user' (checked and highlighted with a red box). A note below the second option states: 'We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.'

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



Step 36: Click on 'Next'.



Step 37: Now click on 'Create User'.

Step 3:
Review and create

User name: deekshitha
Console password type: Autogenerated
Require password reset: Yes

Step 4:
Retrieve password

Permissions summary

Name	Type	Used as
IAMUserChangePassword	AWS managed	Permissions policy

Tags - optional
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.
[Add new tag](#)
You can add up to 50 more tags.

Cancel Previous **Create user**

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

User created successfully
You can view and download the user's password and email instructions for signing in to the AWS Management Console.

IAM > Users > Create user

Step 1: Specify user details

Step 2: Set permissions

Step 3: Review and create

Step 4: Retrieve password

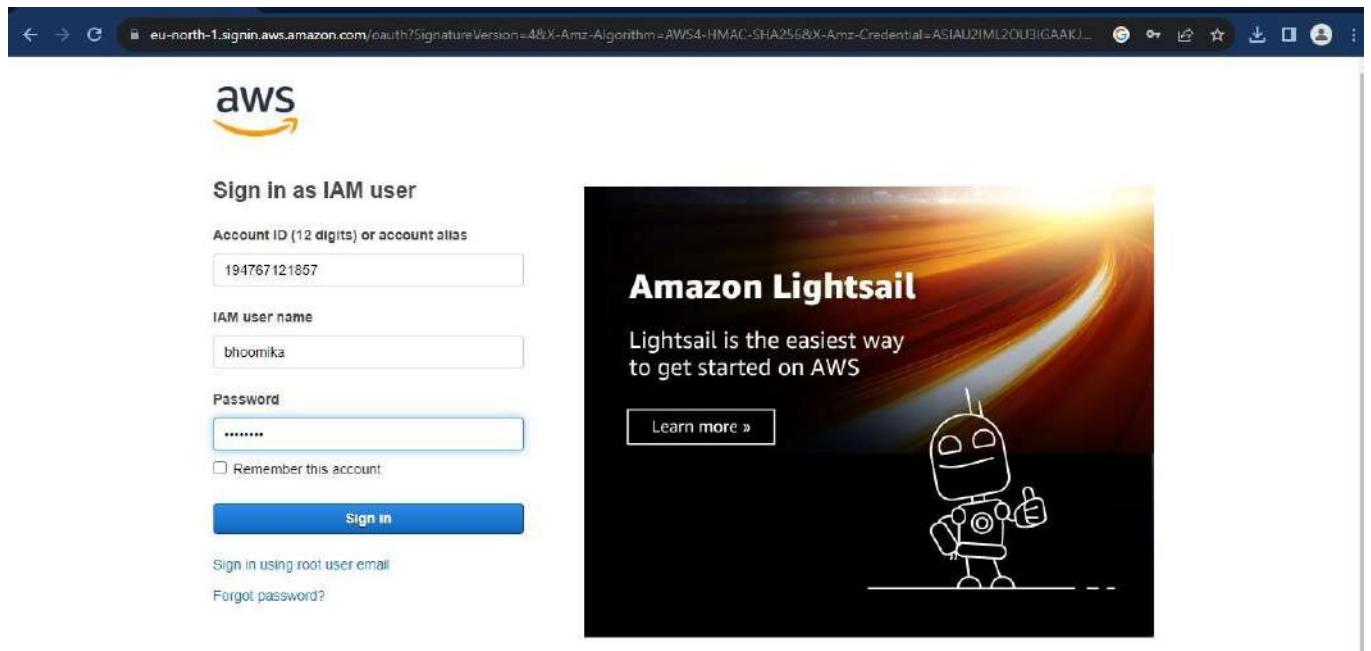
Retrieve password
You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

Console sign-in details

Console sign-in URL	https://194767121857.signin.aws.amazon.com/console	Email sign-in instructions
User name	bhoomika	
Console password	***** Show	

Cancel Download .csv file Return to users list

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 **Confirm password change**.

The screenshot shows the 'Change password' form. It requires the 'Old password' and 'New password' fields to be filled. Below them is a 'Retype new password' field. A large blue 'Confirm password change' button is positioned below the retype field. At the bottom of the form, there is a link for 'Sign in using root user email' and a language selection dropdown set to 'English'.

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

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**'.

The screenshot shows the Jenkins dashboard with a search bar at the top. Below it, a form titled "Enter an item name" has the value "wow9" entered. A red box highlights this input field. Below the form, a list of project types is shown: "Freestyle project" (selected), "Pipeline", "Multi-configuration project", "Folder", and "Branch Pipeline". The "Branch Pipeline" option has a red box around its description and an "OK" button below it.

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

The screenshot shows the "Configuration" page for the "wow9" project. The left sidebar lists "General", "Source Code Management", "Build Triggers", "Build Environment", "Build Steps", and "Post-build Actions". The "General" tab is selected. The "Description" field contains the text "This is my first project.", which is highlighted with a red box. Other options like "Plain text" and "Preview" are also visible.

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

The screenshot shows the "Configuration" page for the "wow9" project. The "Build Environment" section is expanded, showing various checkboxes for workspace management. The "Build Steps" section is expanded, showing a dropdown menu with the option "Add build step". A sub-menu is open under "Add build step", listing "Execute Windows batch command", "Execute shell", "Invoke Ant", "Invoke Grails script", "Invoke top-level Maven targets", "Run with timeout", "Set build status to 'pending' on GitHub commit", and "Trigger/call builds on other projects". The "Execute Windows batch command" option is highlighted with a red box.

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

The screenshot shows the Jenkins configuration interface for a project named 'wow9'. In the 'Build Steps' section, there is a single step titled 'Execute Windows batch command'. The command entered is 'echo "This is my first application"'. A red box highlights the 'Save' button at the bottom of the configuration page.

The screenshot shows the Jenkins dashboard with the 'Project wow9' status. The status bar indicates 'This is my first project.' There are several navigation links on the left: Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. On the right, there are links for Edit description and Disable Project. Below the status, there is a 'Build History' section with links for Atom feed for all and Atom feed for failures.

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

The screenshot shows the Jenkins dashboard. A red box highlights the '+ New Item' button in the top-left corner of the main content area. The dashboard also displays a list of existing projects: 'build pipeline wow', 'build pipeline wow6', and 'build pipeline wow7'. A table below shows recent activity, including a successful build by user 'BHOOMIKA'.

S	W	Name	Last Success	Last Failure	Last Duration
		BHOOMIKA	3.6 sec #7171	N/A	0.4 sec

The screenshot shows the Jenkins 'New Item' creation dialog. A red box highlights the 'sample wow9' input field. Another red box highlights the 'Freestyle project' section, which is described as the central feature of Jenkins, combining any SCM with any build system. A third red box highlights the 'OK' button at the bottom of the dialog.

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**.

The screenshot shows the Jenkins configuration page for the 'sample wow9' project. The 'General' tab is selected, showing a 'Description' field containing 'this is my second project'. The 'Enabled' checkbox is checked. The 'Build Triggers' tab is selected, showing the 'Build after other projects are built' option checked. Under 'Projects to watch', 'wow9' is listed. The 'Save' button at the bottom is highlighted with a red box.

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**.

The screenshot shows the Jenkins dashboard. On the left sidebar, under 'Manage Jenkins', the 'Manage Jenkins' option is highlighted with a red box. The main content area displays a table of build pipelines: BHOOMIKA, BHOOMIKA R, BHOOMIKA6, Demo, demo1, and example. Each row includes columns for Status (S), Warning (W), Name, Last Success, Last Failure, and Last Duration.

S	W	Name	Last Success	Last Failure	Last Duration
✓	⚠	BHOOMIKA	4.2 sec #7202	N/A	78 ms
✓	⚠	BHOOMIKA R	10 days #2	N/A	5.3 sec
✓	⚠	BHOOMIKA6	10 days #1	N/A	0.18 sec
✓	⚠	Demo	10 days #1	N/A	0.15 sec
✗	⚠	demo1	N/A	10 days #2	4 ms
✓	⚠	example	10 days #1	N/A	2 sec

Step 10: Go to Plugins.

The screenshot shows the 'Manage Jenkins' page. Under 'System Configuration', the 'Plugins' section is highlighted with a red box. It contains a brief description: 'Add, remove, disable or enable plugins that can extend the functionality of Jenkins.' Below this, there are sections for 'Build Pipeline Plugin 1.5.8', 'Nodes', 'Tools', and 'Clouds'. A message at the top right says: 'Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.' with buttons for 'Set up agent', 'Set up cloud', and 'Dismiss'.

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

The screenshot shows the 'Plugins' page. The 'Available plugins' tab is selected and highlighted with a red box. A search bar contains the text 'build pipeline'. Below the search bar, a table lists two plugins: 'Webhook Step' and 'Pipeline timeline'. Each plugin has an 'Install' button and a release date.

Install	Name	Released
<input type="checkbox"/>	Webhook Step 173.vf1_b_93560b_977 Allows build pipelines to wait for notification from an external system before continuing.	1 yr 3 mo ago
<input type="checkbox"/>	Pipeline timeline 1.0.3 An interactive build timeline to help you visualize your build pipeline and identify bottlenecks.	4 yr 7 mo ago

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

The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are dropdowns for 'Build Queue (1)' (showing 'BHOOMIKA') and 'Build Executor Status' (showing 1 Idle and 2 Idle). At the top right, there's a search bar, a user icon for 'Bhoomika R', and a 'log out' link. In the center, there's a table titled 'All' showing build pipelines: 'build pipeline wow', 'build pipeline wow6', and 'build pipeline wow?'. A red box highlights the 'New View' button at the top right of the table area.

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

Create.

The screenshot shows the 'New view' creation page. The 'Name' field is filled with 'build pipeline wow9' and has a red box around it. The 'Type' section has a radio button for 'Build Pipeline View' which is selected and highlighted with a red box. Below this, there are descriptions for 'List View' and 'My View'. At the bottom, a blue 'Create' button is highlighted with a red box.

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

The screenshot shows the 'Configure' page for the 'build pipeline wow9' view. The 'Name' field is set to 'build pipeline wow9'. The 'Description' field contains the text 'This is fifth project.' and is highlighted with a red box. Below the description, there are options for 'Plain text' and 'Preview'. Further down, there are checkboxes for 'Filter build queue' and 'Filter build executors', and a 'Build Pipeline View Title' field. At the bottom, there's a 'Pipeline Flow' section.

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

This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs. This is the only out-of-the-box supported layout mode, but is open for extension.

Upstream / downstream config

Select Initial Job ?

sample test wow9

BHOOMIKA
BHOOMIKA6
Demo
demo1
Trig
sample delay wow
sample delay wow6
Build
sample delay wow7
sample delay wow9
sample release wow
sample release wow6
sample release wow7
sample test wow
sample test wow6
sample test wow7
Restart
sample test wow9

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

No Of Displayed Builds ?

1
2
3
5
10
20
50
100
200
500
No header

Do not show any column headers

Refresh frequency (in seconds) ?

3

URL for custom CSS files

OK Apply

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

Jenkins

Dashboard > build pipeline wow9 >

Build Pipeline

This is fifth project.

Trigger a Pipeline Run Pipeline History Configure Add Step Delete Manage

This view has no jobs associated with it. You can either add some existing jobs to this view or create a new job in this view.

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

This screenshot shows the Jenkins Build Pipeline page. At the top, there's a navigation bar with links for 'Trigger a Pipeline', 'Pipeline History', 'Configure', 'Add Step', 'Delete', and 'Manage'. Below the navigation bar, a green header bar displays the build information: '#1 sample test wow9' (Build #1), 'Sep 11, 2023 23:07:54 PM' (Timestamp), '0:12 sec' (Duration), and 'Bhoomika R' (User). A red box highlights the 'console' link at the end of the green bar.

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.

This screenshot shows the Jenkins Console Output page for build #1 of the 'sample test wow9' pipeline. The left sidebar has options: Status, Changes, Console Output (which is selected and highlighted in grey), View as plain text, Edit Build Information, Delete build '#1', and Next Build. The main content area shows a summary of the build: 'Started by user Bhoomika R', 'Running as SYSTEM', 'Building in workspace C:\ProgramData\Jenkins\workspace\sample test wow9', and 'Finished: SUCCESS'. At the bottom, it says 'Console output for sample test wow9 #1'.

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 > ...
1  <html>
2  |   <body>
3  |   |   <head>
4  |   |   |       <link rel="stylesheet" href="style.css">
5  |   |   </head>
6  |   <script>
7  |   |       function validateform(){
8  |   |       var name=document.myform.name.value;
9  |   |       var password=document.myform.password.value;
10 |   |       if (name==null || name=="")
11 |   |       {
12 |   |           alert("Name can't be blank")
13 |   |           return false;
14 |   |       }
15 |   |       else if(password.length<6)
16 |   |       {
17 |   |           alert("Password must be at least 6 characters long")
18 |   |           return false;
19 |   |       }
20 |   |   </script>
21 |   <body><center>
22 |   |   <form name="myform" method="post" action="valid.html" onsubmit="return validateform()">
23 |   |       Enter your Name: <input type="text" name="name"><br><br>
24 |   |       Enter your Password: <input type="password" name="password"><br><br>
25 |   |       <input type="submit" value="Register">
26 |   |   </form>
27 |   </body></center>
28 |
29 </html>
30
```

```
< valid.html > ...
1  <html>
2  |   <body>
3  |   |   <h1>Validation Successfull</h1>
4  |   |   </body>
5  </html>
```

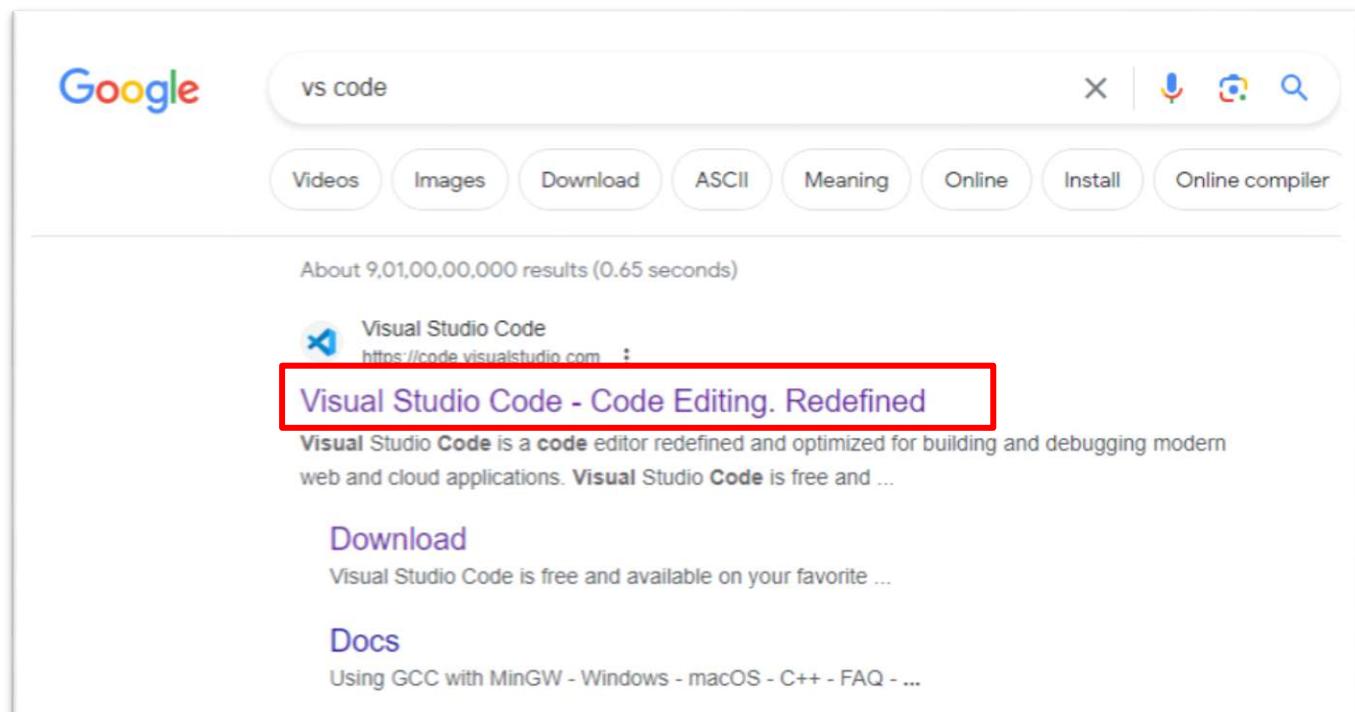
B) Using CSS file.

```
# style.css > ...
1  body {
2      background-color: #rgb(159, 203, 204);
3      margin: 100px;
4      padding: 50px;
5  }
```

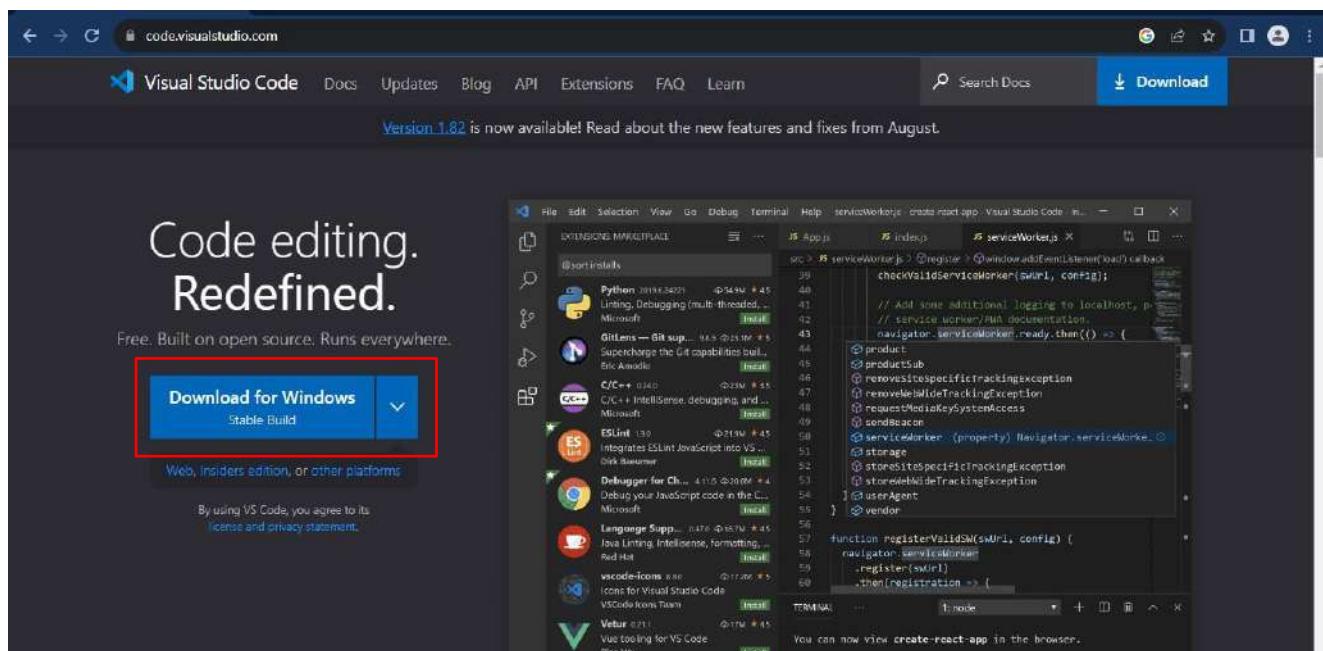
2. Setting up the Environment and tools for Front End Development.

- Installation of VS Code.**

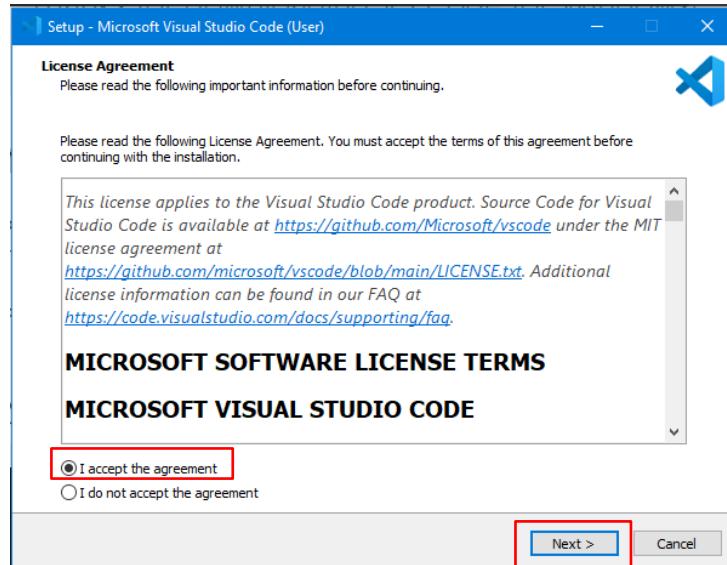
Step 1: Go to Web Browser and search for VS Code. Then click on the first link.



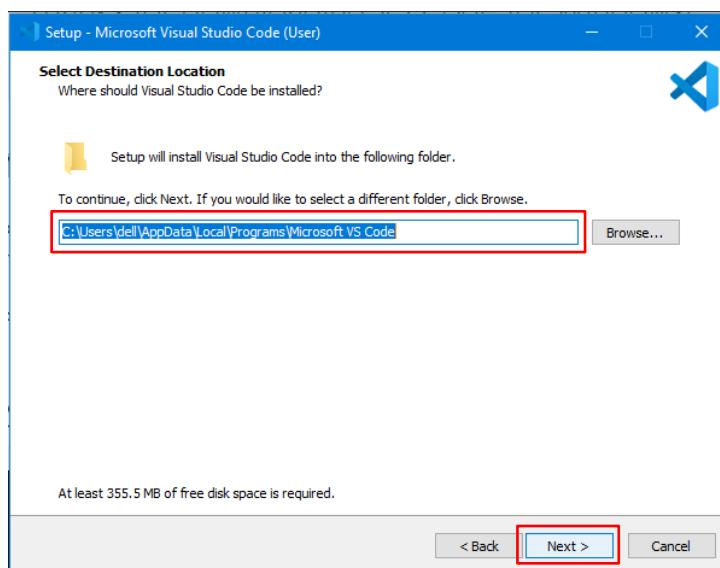
Step 2: Click on ‘Download for Windows’.



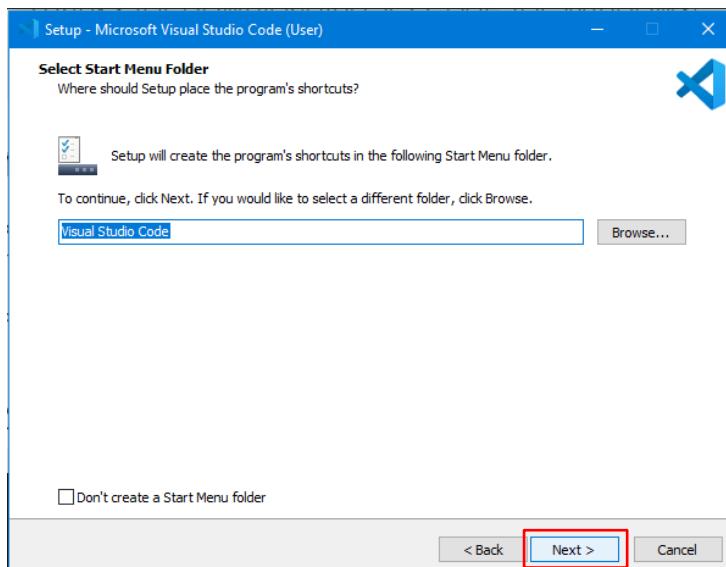
Step 3: Enable the first option ‘I accept the agreement’ & click on ‘Next’.



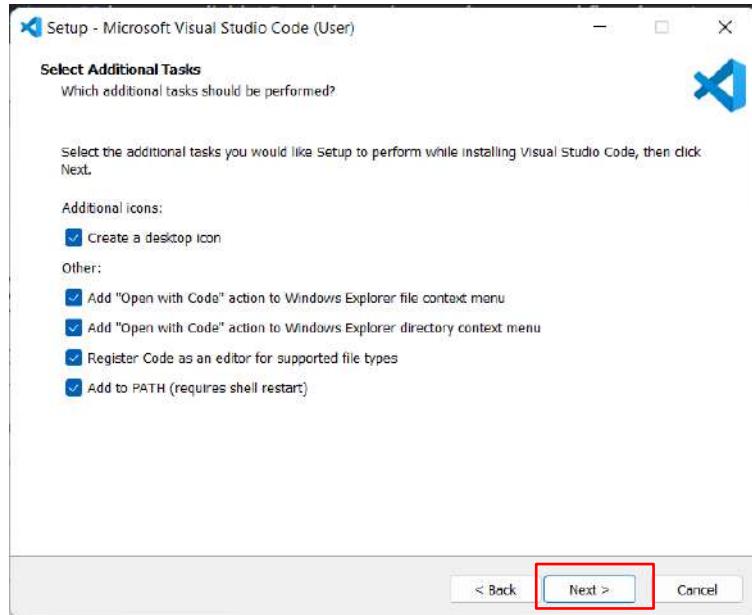
Step 4: Choose the location to save the file & click on 'Next'.



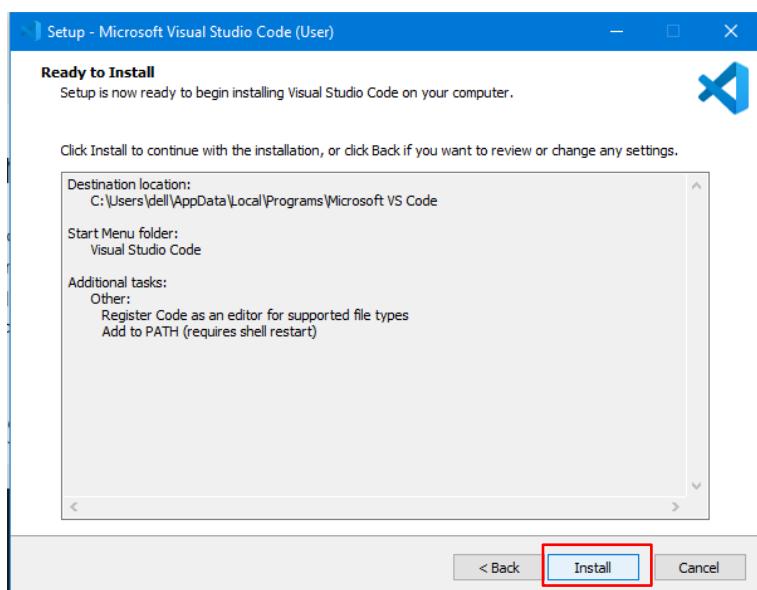
Step 5: Select a folder name or click on 'Next'.



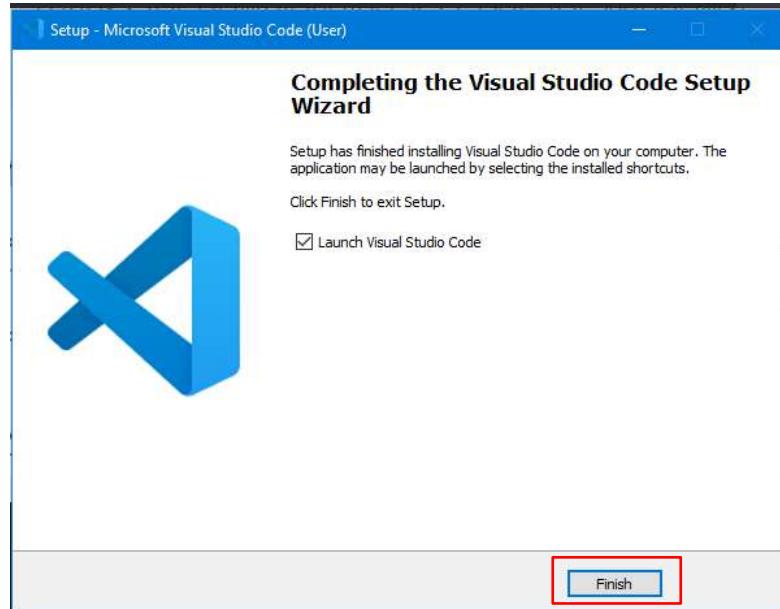
Step 6: Enable all the checkboxes and click on 'Next'.



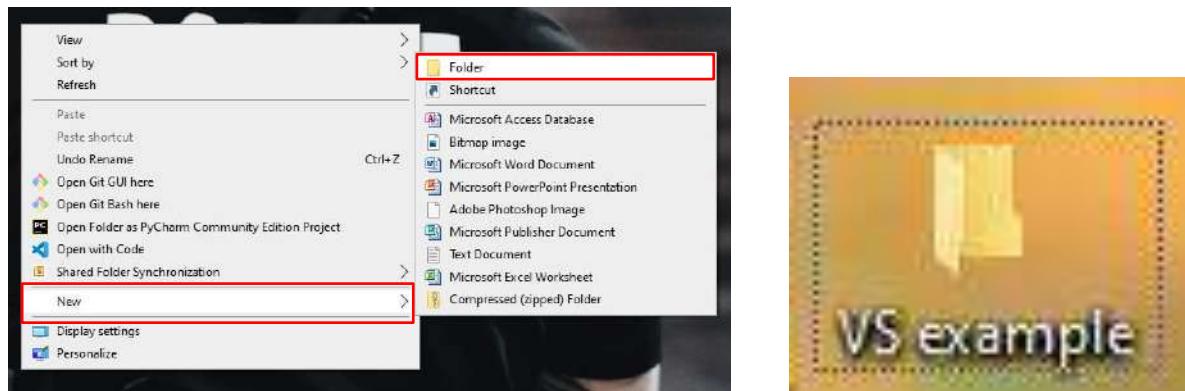
Step 7: Now click on 'Install'.



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



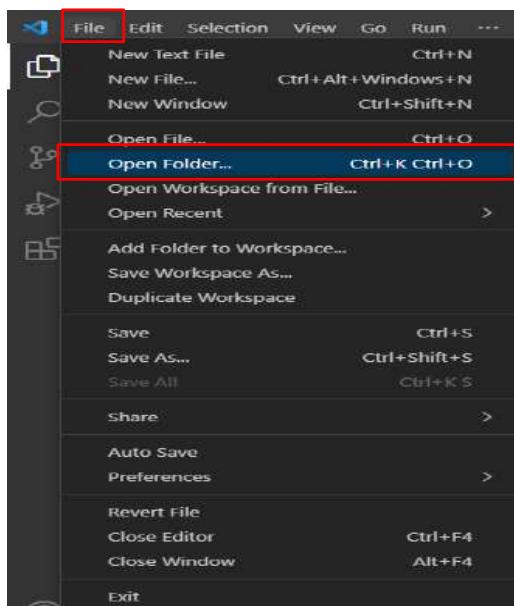
Step 9: Go to Desktop □ Right click □ New □ Folder □ Folder name.

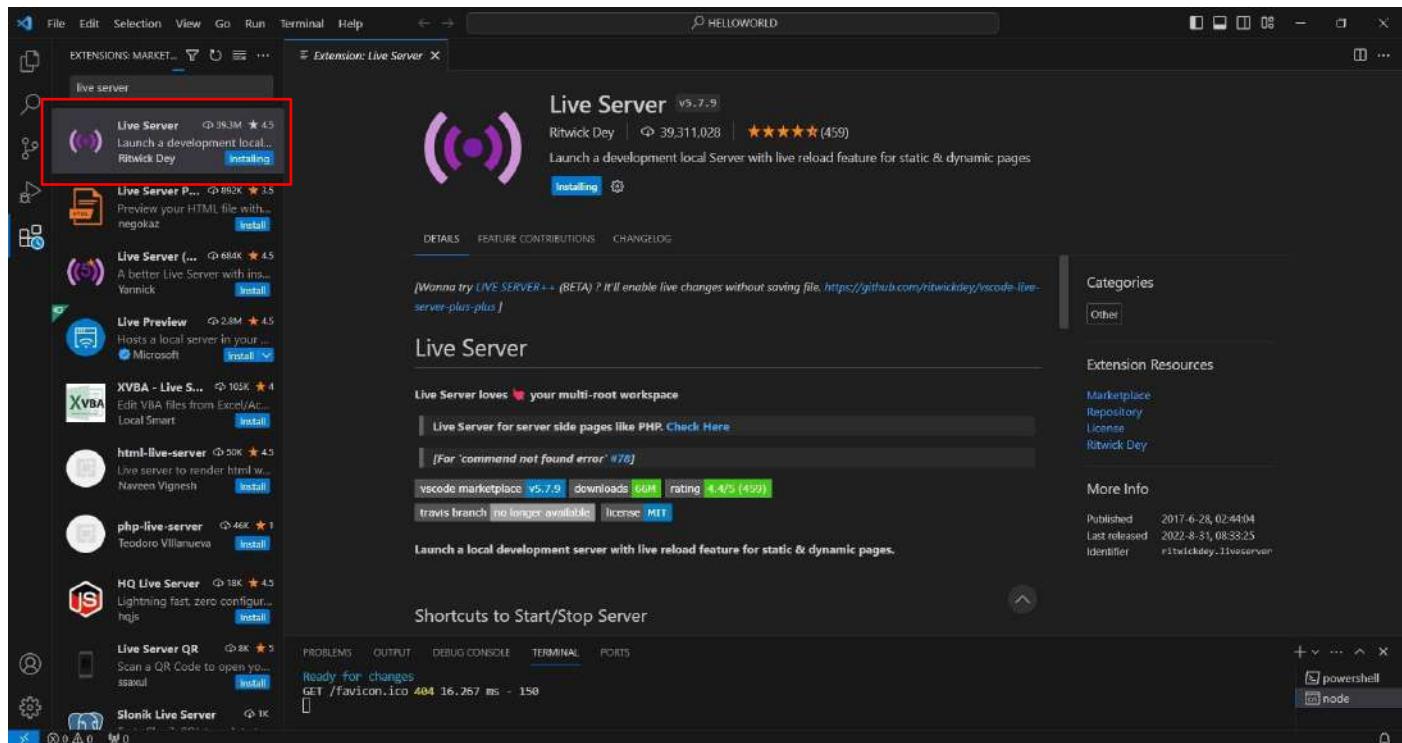


Step 10: Launch the VS Code □ click on File □ Open Folder □ Select the newly created folder.

- **VS Code extensions**

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
1  <html>
2    <head>
3      <title>Registration form</title>
4      <script>
5        function passvalues()
6        {
7          var name=document.getElementById("name").value;
8          var email=document.getElementById("email").value;
9          var address=document.getElementById("address").value;
10         localStorage.setItem("name",name);
11         localStorage.setItem("email",email);
12         localStorage.setItem("address",address);
13         return;
14     }
15   </script>
16 </head>
17 <body><center>
18   <h1>Registration form</h1>
19   <form action="details.html">
20     <fieldset>
21       <legend>Registration</legend>
22       <label>Name</label>
23       <input type="text" id="name"/><br><br>
24       <label>Email ID</label>
25       <input type="email" id="email"/><br><br>
26       <label>Address</label>
27       <input type="address" id="address"/><br><br>
28       <input type="submit" value="submit" onclick="passvalues()"/></center>
29     </fieldset>
30   </form>
31 </body>
32 </html>

```

Details Form:

```

⑦ details.html > ⚡ html
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <title>Details</title>
5   </head>
6   <body>
7       <form>
8           Your Name is:<p id="name"></p><br>
9           Your Email is:<p id="email"></p><br>
10          Your Address is:<p id="address"></p>
11          <script>
12              document.getElementById("name").innerHTML=localStorage.getItem("name");
13              document.getElementById("email").innerHTML=localStorage.getItem("email");
14              document.getElementById("address").innerHTML=localStorage.getItem("address");
15          </script>
16      </form>
17  </body>
18  </html>

```

Final Output:

The screenshot shows a web browser window with the title "Registration Form". Inside the form, there are three input fields: "Name" containing "Brinda", "Email - id" containing "Bri123@gmail.com", and "Address" containing "Vijayanagar". A "submit" button is visible at the bottom of the form.

The screenshot shows a web browser window with the title "Details". The page contains the following text output:
 Your name is:
 Brinda
 Your Email Id is:
 Bri123@gmail.com
 Your Address is:
 Vijayanagar

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.

Node.js® is an open-source, cross-platform JavaScript runtime environment.

Download for Windows (x64)

18.18.0 LTS
Recommended For Most Users

20.7.0 Current
Latest Features

For information about supported releases, see the [release schedule](#).

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

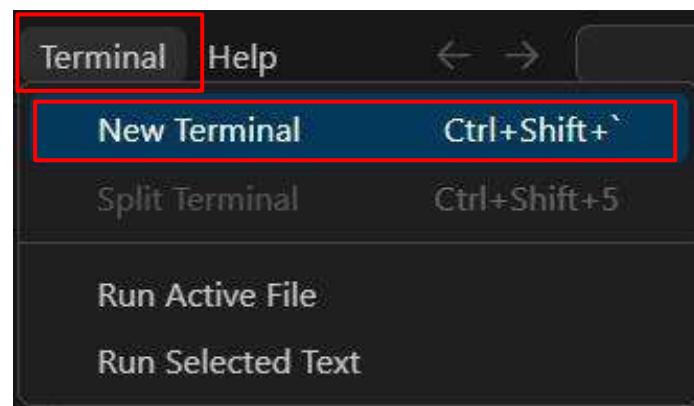
C:\Users\hk923>npm install typescript@latest -g
changed 1 package in 1s

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

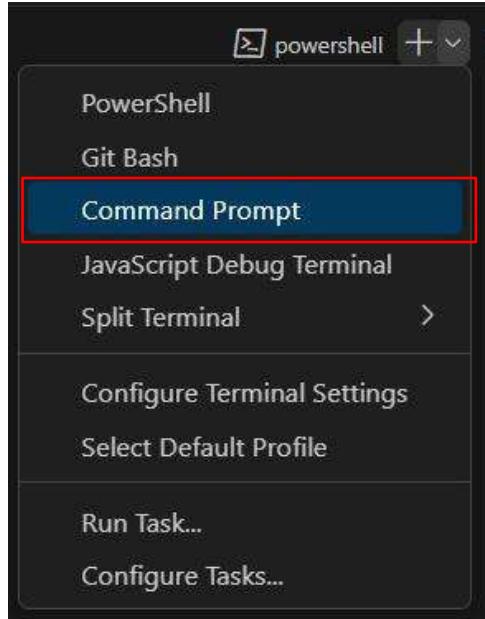
C:\Users\hk923>npm install -g live-server
[██████████] / idealTree:send: sill placeDep node_modules/live-server ms@2.0.0 OK for: debug@2.6.9 want: 2.0.0
```

- **Installation of typescript inside the VS code live server.**

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.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

C:\Users\hk923\OneDrive\Desktop\HELLOWORLD>npm install typescript --save.dev
up to date, audited 2 packages in 2s
found 0 vulnerabilities
```

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

```
C:\Users\hk923\OneDrive\Desktop\HELLOWORLD>npm install typescript -g
changed 1 package in 1s
C:\Users\hk923\OneDrive\Desktop\HELLOWORLD>npm install typescript@latest -g
changed 1 package in 2s
C:\Users\hk923\OneDrive\Desktop\HELLOWORLD>tsc -v
Version 5.2.2
```

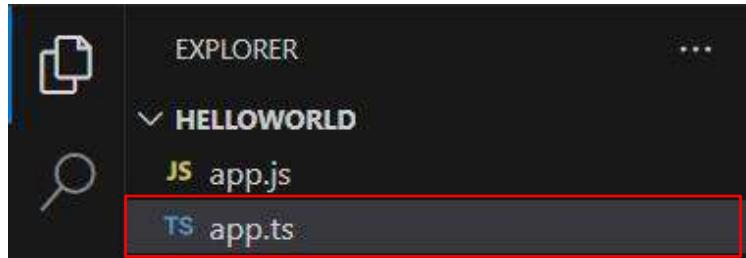
Step 7: Then enter '**npm install -g live-server**' to install the Live Server.

```
C:\Users\hk923\OneDrive\Desktop\HELLOWORLD>npm install -g live-server
[=====] \ idealTree:anymatch: sill placeDep node_modules/live-server normalize-path@3.0.0 OK for: chokidar@2.1.8 want: ^3.0.0
```

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

```
C:\Users\hk923\OneDrive\Desktop\TypeScript ex>live-server
Serving "C:\Users\hk923\OneDrive\Desktop\TypeScript ex" at http://127.0.0.1:8080
Ready for changes
GET /favicon.ico 404 13.798 ms - 150
^C
```

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.



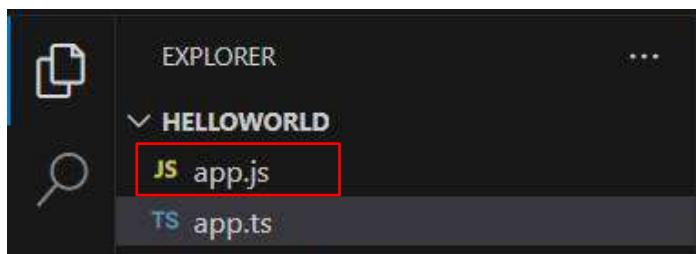
The screenshot shows the VS Code interface. The Explorer sidebar on the left has a tree view with a folder named 'HELLOWORLD'. Inside the folder, there are two files: 'app.js' (marked with a JS icon) and 'app.ts' (marked with a TS icon). The 'app.ts' file is highlighted with a red border. The main editor area on the right displays the code for 'app.ts':

```
TS app.ts      X  ↗ index.html  
TS app.ts > ...  
1 let message: string = 'Hello, World!';  
2 console.log(message);
```

Step 10: Go back to the Terminal and enter ‘tsc app.ts’.

```
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.



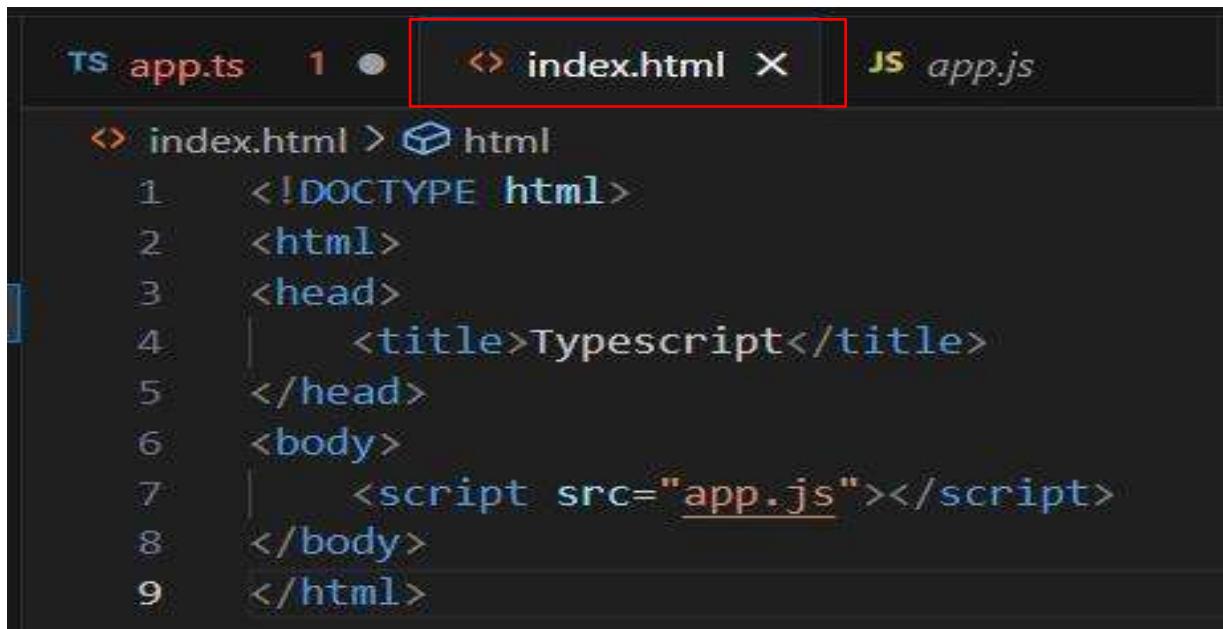
The screenshot shows the VS Code interface. The Explorer sidebar on the left has a tree view with a folder named 'HELLOWORLD'. Inside the folder, there are two files: 'app.js' (marked with a JS icon) and 'app.ts' (marked with a TS icon). The 'app.js' file is highlighted with a red border. The main editor area on the right is not visible in this specific screenshot.

Step 12: Use the command ‘node app.js’ to run that file.

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

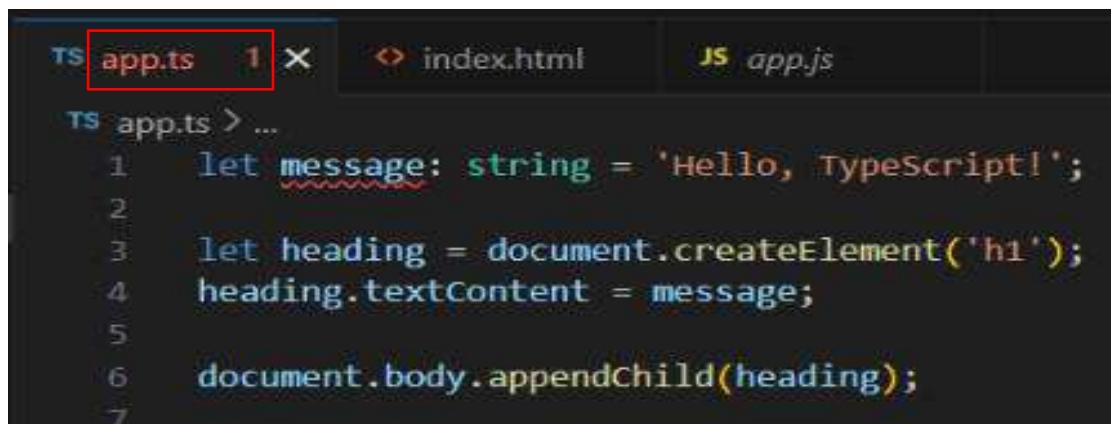
- Usage of TypeScript & Live Server to run the ‘Hello World’ program.

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



```
<> index.html > html
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <title>TypeScript</title>
5     </head>
6     <body>
7       <script src="app.js"></script>
8     </body>
9   </html>
```

Step 14: Then make changes to the ‘app.ts’ code as shown below.



```
ts app.ts > ...
1 let message: string = 'Hello, TypeScript!';
2
3 let heading = document.createElement('h1');
4 heading.textContent = message;
5
6 document.body.appendChild(heading);
7
```

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**.

The screenshot shows the Visual Studio Code interface with three tabs open: 'app.ts', 'index.html', and 'app.js'. The 'index.html' tab is active, displaying the following code:

```
<!DOCTYPE html>
<html>
<head>
    <title>TypeScript</title>
</head>
<body>
    <script src="app.js"></script>
</body>
</html>
```

A context menu is open over the title bar of the 'index.html' tab. The menu items are:

- Go to Definition F12
- Go to References Shift+F12
- Peek >
- Find All References Shift+Alt+F12
- Rename Symbol F2
- Change All Occurrences Ctrl+F2
- Format Document Shift+Alt+F
- Refactor... Ctrl+Shift+R
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Open with Live Server Alt+L Alt+O** (highlighted with a red box)
- Stop Live Server Alt+L Alt+C
- Command Palette... Ctrl+Shift+P

Output:



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

A. Login.html

```
app1 > public > login.html
1  <!DOCTYPE html>
2  <html>
3  <head>
4  | <title>Login Page</title>
5  </head>
6  <body>
7  | <h1>Login</h1>
8  | <form id="login-form">
9  | | <label for="username">Username:</label>
10 | | <input type="text" id="username" name="username" required><br>
11 | | <label for="password">Password:</label>
12 | | <input type="password" id="password" name="password" required><br>
13 | | <input type="submit" value="Login">
14 | </form>
15 | <p id="message"></p>
16 | <script src="login.js"></script>
17 </body>
18 </html>
```

B. Login.js

```
app1 > src > JS login.js
1  document.getElementById("login-form").addEventListener("submit", function (event) {
2    event.preventDefault();
3
4    // Get user input
5    var username = document.getElementById("username").value;
6    var password = document.getElementById("password").value;
7
8    // Load JSON data
9    fetch('login.json')
10   .then(response => response.json())
11   .then(data => {
12     const users = data.users;
13     const user = users.find(u => u.username === username && u.password === password);
14
15     if (user) {
16       document.getElementById("message").textContent = "Login successful!";
17     } else {
18       document.getElementById("message").textContent = "Invalid username or password.";
19     }
20   })
21   .catch(error => console.error('Error loading JSON: ', error));
22 })|
```

C. Login.json

```
app1 > src > {} login.json > ...
1  [
2   "users": [
3     {
4       "username": "user1",
5       "password": "password1"
6     },
7     {
8       "username": "user2",
9       "password": "password2"
10    }
11  ]
12 ]
```

Output:

Please leave some space here while writing in record.

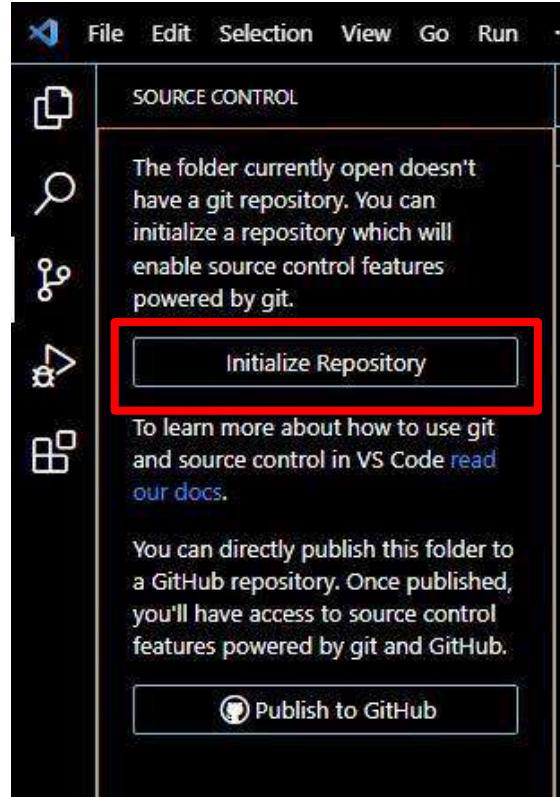
And continue from here.

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

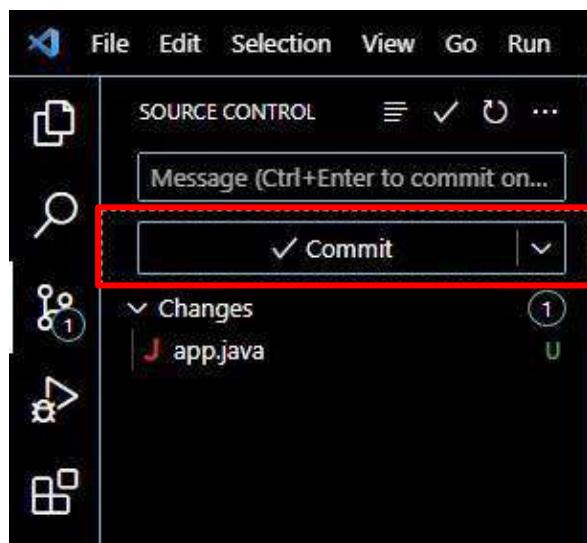
Step 1: Create an empty folder in the Desktop and open that folder via VS Code. Then create a new file named ‘app.java’ & add some code to it.



Step 2: Go to Extensions □ Initialize Repository.



Step 3: Now click on ‘Commit’.



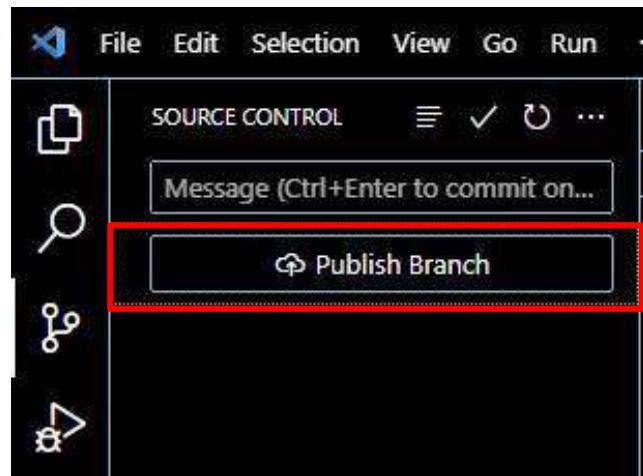
Step 4: Enter a command as ‘first commit’ & Save it.

A screenshot of the Visual Studio Code interface. The title bar shows 'app.java U' and 'COMMIT_EDITMSG'. The main editor area displays a commit message template:

```
1 first commit
2 # Please enter the commit message for your changes. Lines starting
3 # with '#' will be ignored, and an empty message aborts the commit.
4 #
5 # On branch main
6 #
7 # Initial commit
8 #
9 # Changes to be committed:
10 #   new file: app.java
11 #
12 [
```

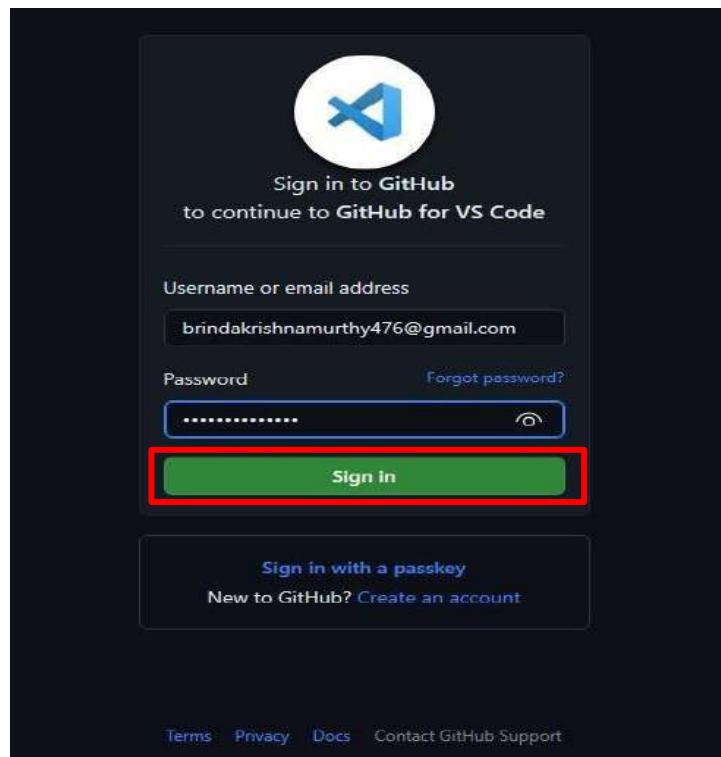
The status bar at the bottom right shows 'git: 1 file, 1 changes to commit'.

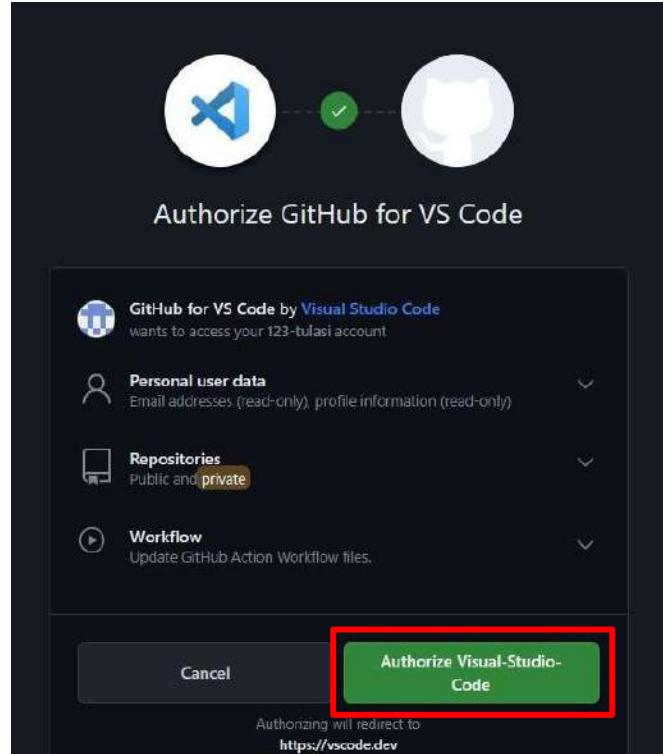
Step 5: Then click on ‘Publish Branch’.



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

Step 7: then click on ‘Authorize Visual-Studio-Code’.

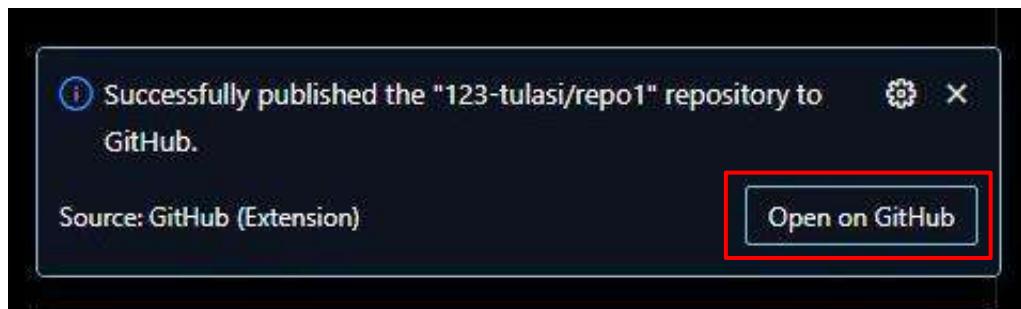




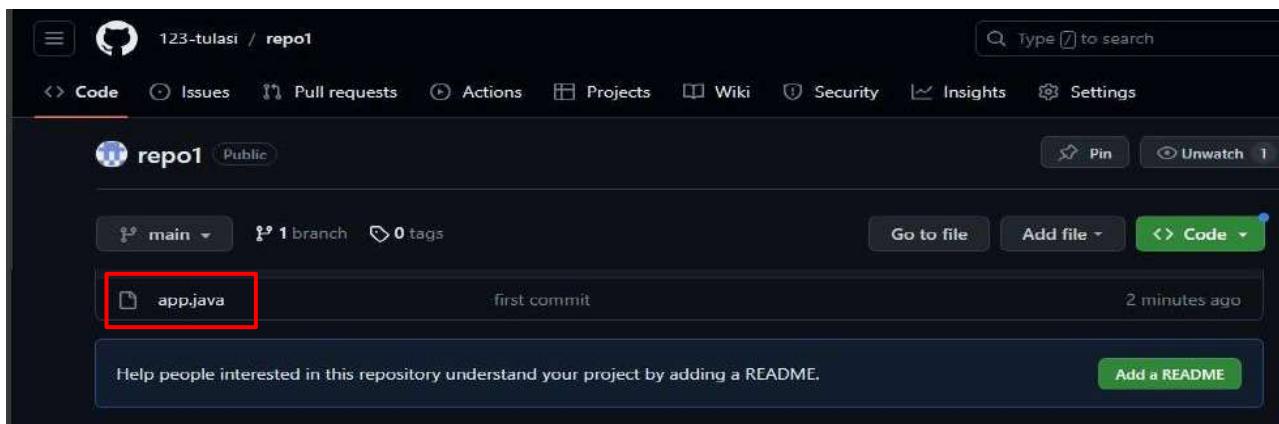
Step 8: Select ‘Publish to GitHub public repository’.



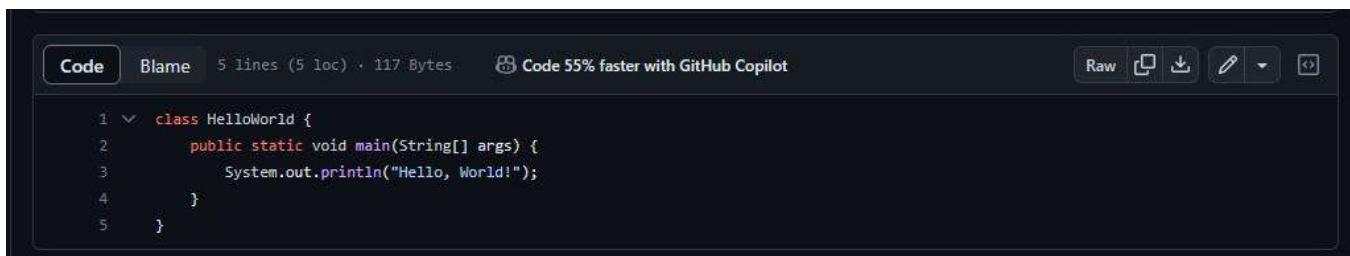
Step 9: Click on ‘Open on GitHub’.



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



Step 11: The code is made visible here.

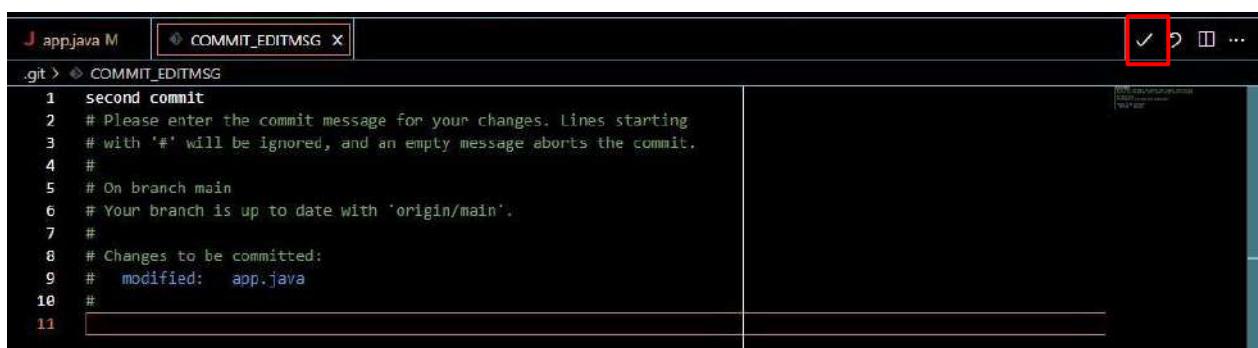


- **Synchronising Changes to the code.**

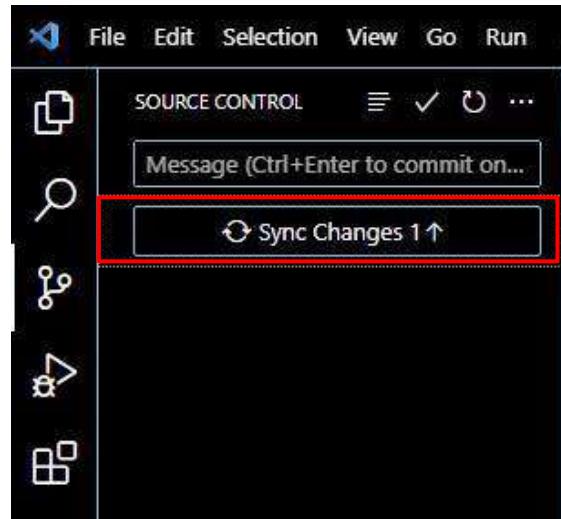
Step 12: Do some changes to your code & click on ‘Commit’.



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.

A screenshot of a GitHub commit history. The commit message is: "Initial commit". It shows one file change: "src/main/java/com/example/helloworld/HelloWorld.java" with 5 additions and 0 deletions. The commit was made by "GitHub User" on "2023-09-15".

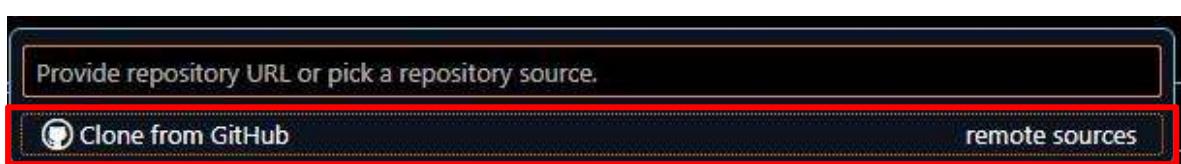
```
1 class HelloWorld {  
2     public static void main(String[] args) {  
3         System.out.println("Hello, World! github");  
4     }  
5 }
```

- **Cloning an empty repository from VS Code to GitHub.**

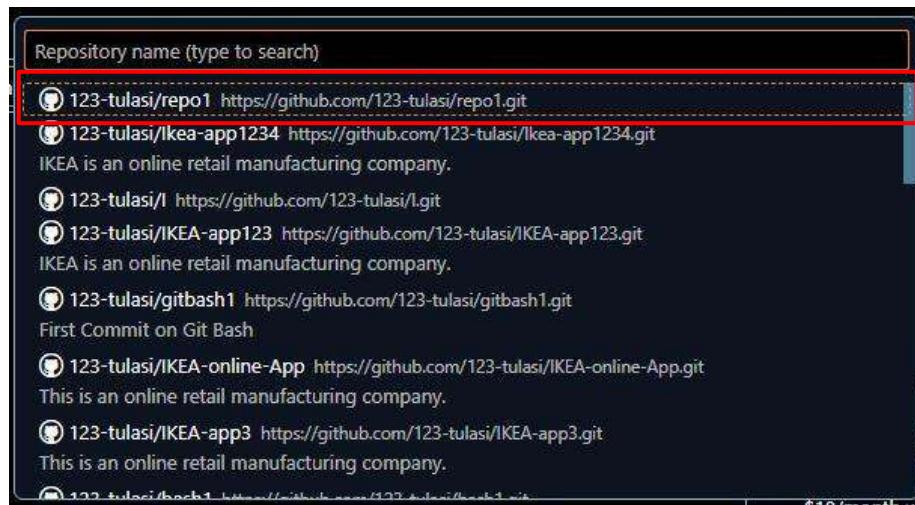
Step 16: Click on ‘Clone Git Repository’.



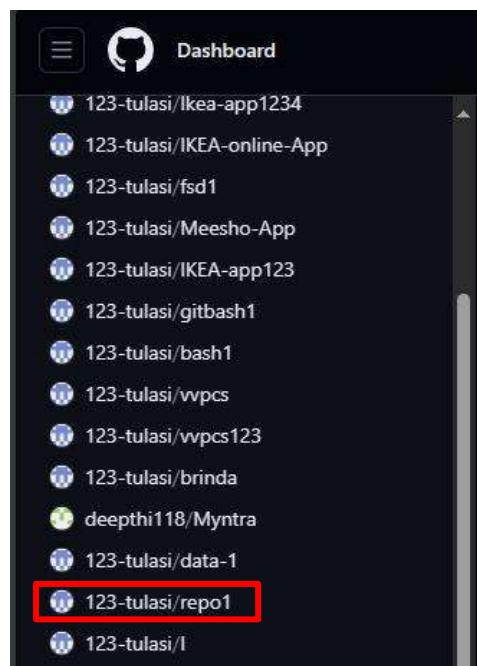
Step 17: Now click on ‘Clone from GitHub’.



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.

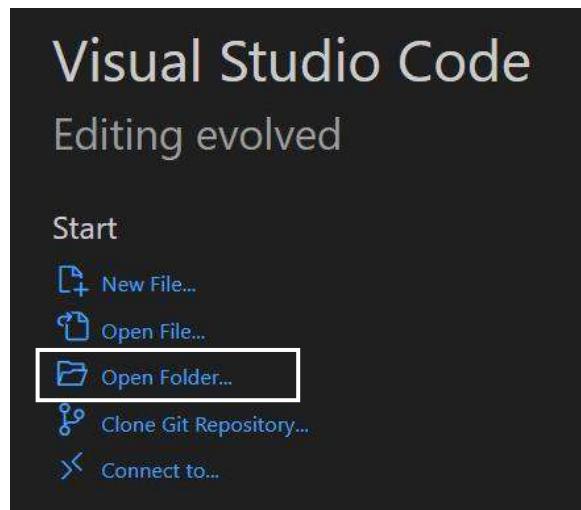
A screenshot of a GitHub code editor interface. The file 'app.java' is open. The code is as follows:

```
1 class HelloWorld {  
2     public static void main(String[] args) {  
3         System.out.println("Hello, World! github");  
4     }  
5 }
```

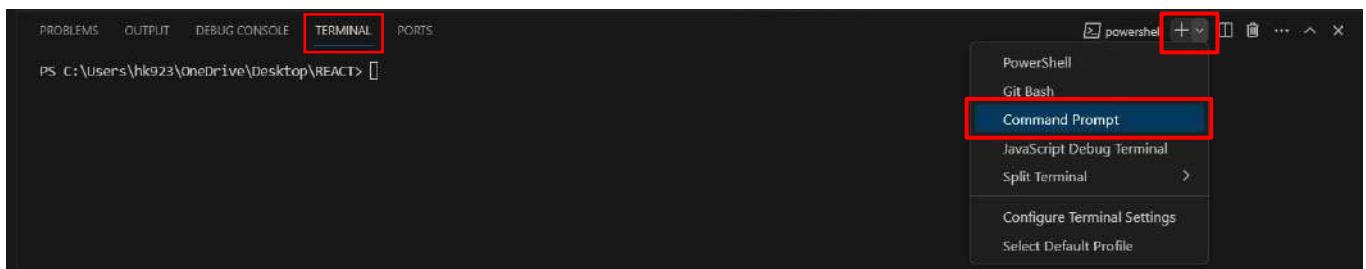
WEEK – 5

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 Terminal □ New Terminal □ Select Command Prompt.



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

A screenshot of the Visual Studio Code terminal window. The terminal tab is active. The output shows the following text:

```
(c) Microsoft Corporation. All rights reserved.  
C:\Users\hk923\OneDrive\Desktop\REACT>npm init  
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.  
  
See `npm help init` for definitive documentation on these fields  
and exactly what they do.  
  
Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.
```

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

A screenshot of the Visual Studio Code terminal window. The terminal tab is active. The output shows the following text:

```
C:\Users\hk923>npm -v  
9.8.1
```

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:**

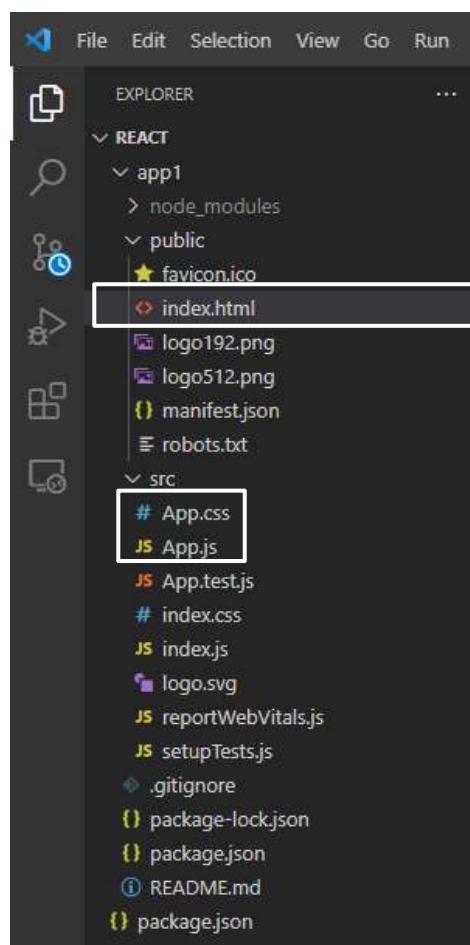
Step 6: Enter the command ‘**npx create-react-app**’ to create a React Project & give the folder name as **app1**.

```
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: 5.11 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 > JS App.js > ...  
1 import './App.css';  
2  
3 function App() {  
4   return (  
5     <div className="App">  
6       <header className="App-header">  
7         <p>  
8           <h1>Welcome to React !</h1>  
9           This is the Welcome page of React.  
10          </p>  
11        </header>  
12      </div>  
13    );  
14  }  
15  
16 export default App;  
17 .
```

```
# 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
```

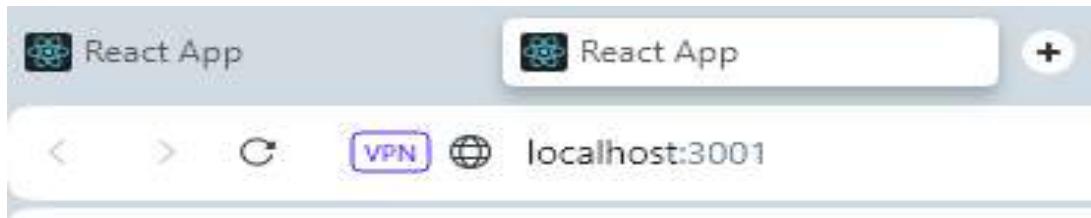
Welcome to React !

This is the Welcome page of React.

2. React Components

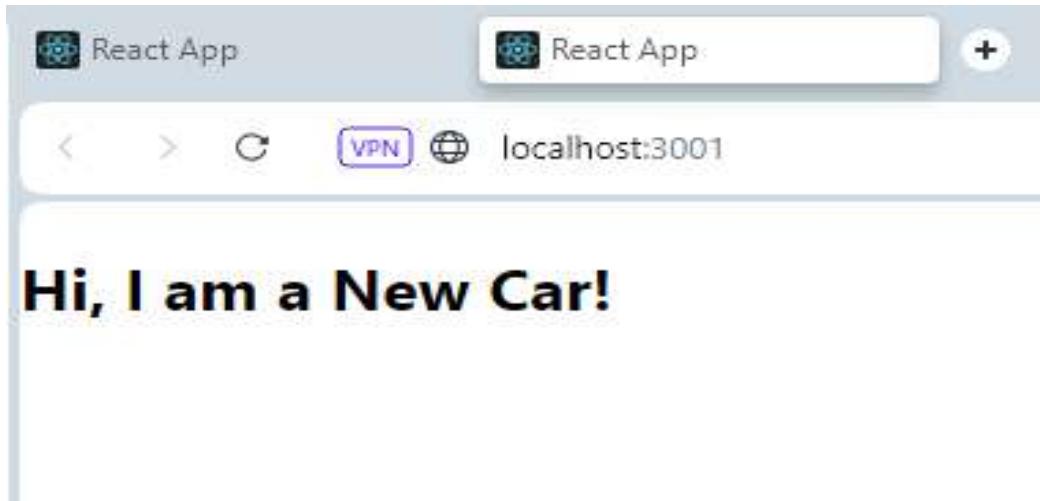
- **Functional Components:** Functional components are simply javascript functions. We can create a functional component in React by writing a javascript function. These functions may or may not receive data as parameters. The functional component is also known as a stateless component because they do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



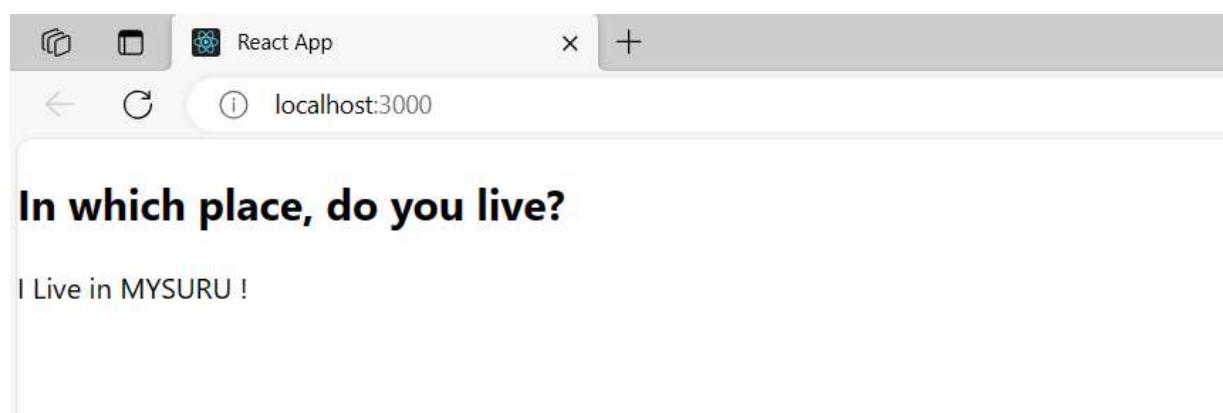
- **Class Component:** A class component are little more complex then functional component. A class component must include the extends React.Component statement. The class component is also known as a stateful component because they can hold or manage local state.

```
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
20
```

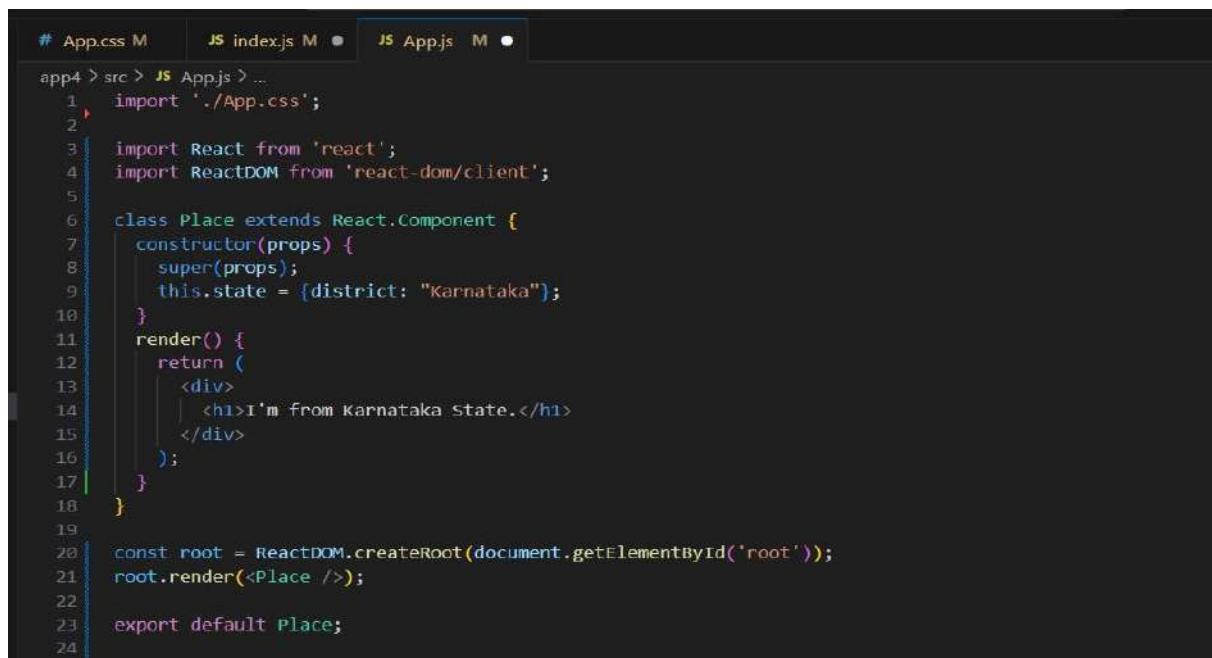


- **React Props:** Props is short for properties and they are used to pass data between React components. React's data flow between components is uni-directional (from parent to child only).

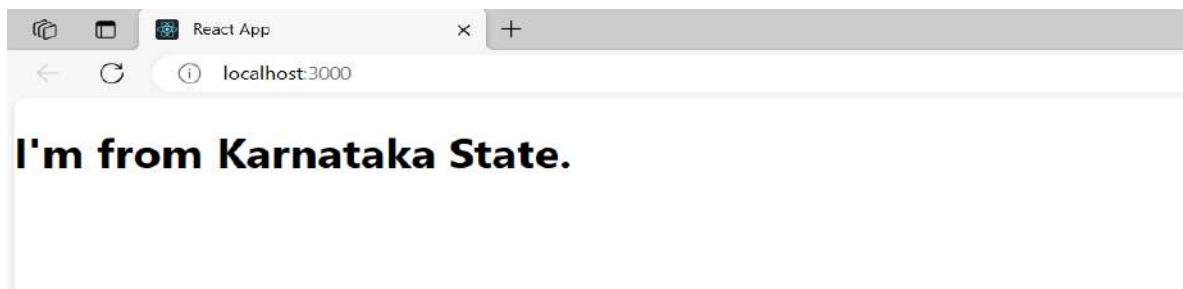
```
# App.css M JS index.js M JS App.js M ●
app4 > src > JS App.js > ...
1 import './App.css';
2
3 import React from 'react';
4 import ReactDOM from 'react-dom/client';
5
6 function City(props) {
7   return (
8     <p>
9       <h2>In which place, do you live?</h2>
10      I Live in {props.place} !</p>
11    );
12 }
13
14 const myElement = <City place="MYSURU" />;
15
16 const root = ReactDOM.createRoot(document.getElementById('root'));
17 root.render(myElement);
18
19 export default City;
20
21
```



- **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.

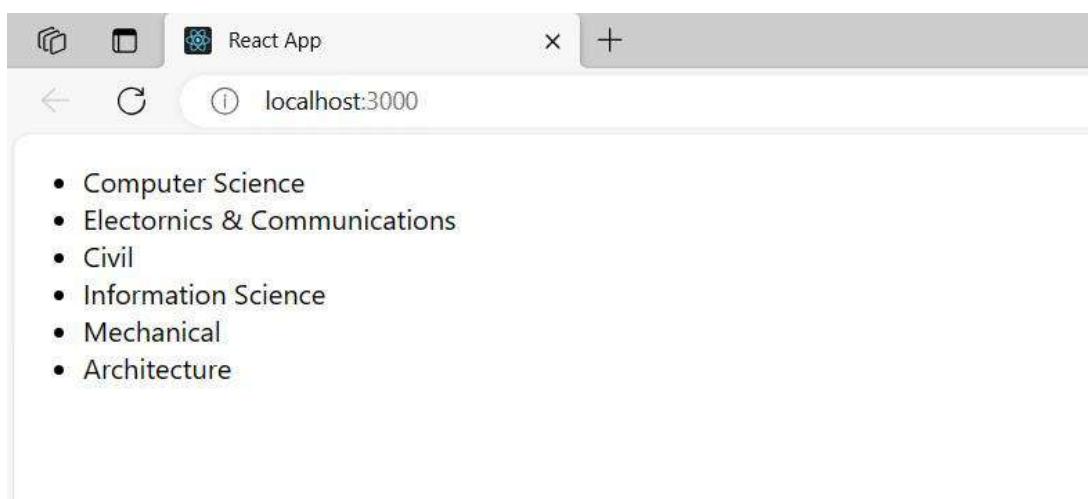


```
# App.css M JS index.js M JS App.js M
app4 > src > JS App.js > ...
1 import './App.css';
2
3 import React from 'react';
4 import ReactDOM from 'react-dom/client';
5
6 class Place extends React.Component {
7   constructor(props) {
8     super(props);
9     this.state = {district: "Karnataka"};
10 }
11 render() {
12   return (
13     <div>
14       <h1>I'm from Karnataka state.</h1>
15     </div>
16   );
17 }
18
19 const root = ReactDOM.createRoot(document.getElementById('root'));
20 root.render(<Place />);
21
22 export default Place;
```



- **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 React to figure out how to update a list, be it adding, updating, or deleting an item in a list.

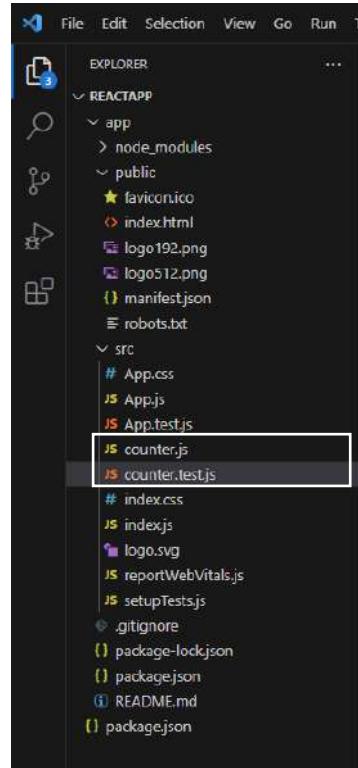
```
# App.css M JS index.js JS App.js M X
app4 > src > JS App.js > default
1 import './App.css';
2
3 import React from 'react';
4 import ReactDOM from 'react-dom/client';
5
6 function App() {
7   const Movies = [
8     { id: 1, name: 'Computer Science' },
9     { id: 2, name: 'Electronics & Communications' },
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.

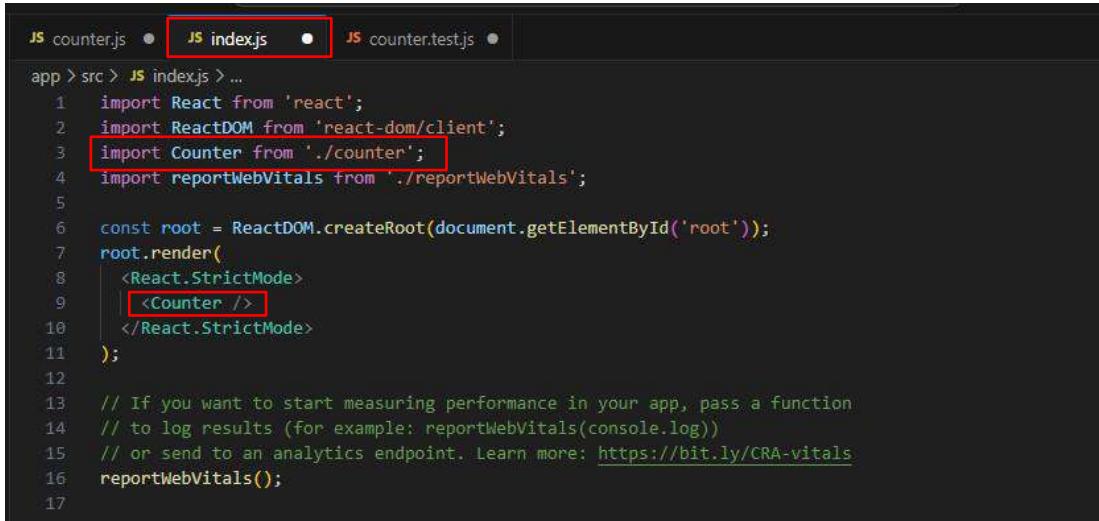


```
JS counter.js ● JS index.js ● JS counter.test.js ●
app > src > JS counter.js > ...
1 import React, { useState } from "react";
2 const Counter = () => {
3   const [counter, setCounter] = useState(0);
4   const incrementCounter = () => {
5     setCounter((prevCounter) => prevCounter + 1);
6   };
7   const decrementCounter = () => {
8     setCounter((prevCounter) => prevCounter - 1);
9   };
10  return (
11    <>
12      <button data-testid="increment" onClick={incrementCounter}>
13        +
14      </button>
15      <p data-testid="counter">{counter}</p>
16      <button data-testid="decrement" onClick={decrementCounter}>
17        -
18      </button>
19    </>
20  );
21 }
22 export default Counter;
23
```

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

```
JS counter.js ● JS index.js ● JS counter.test.js ●
app > src > JS counter.test.js > ...
1 import { render, fireEvent, screen } from "@testing-library/react";
2 import Counter from "./counter";
3 //test block
4 test("increments counter", () => {
5   //render the component on virtual dom
6   render(<Counter />);
7   //select the elements you want to interact with
8   const counter = screen.getByTestId("counter");
9   const incrementBtn = screen.getByTestId("increment");
10  //interact with those elements
11  fireEvent.click(incrementBtn);
12  //assert the expected result
13  expect(counter).toHaveTextContent("1");
14 });
15
16
```

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



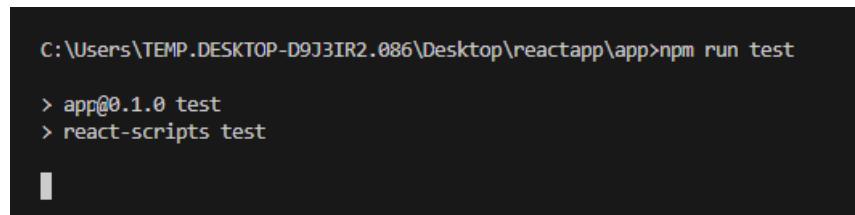
The screenshot shows a code editor with three tabs: 'counter.js', 'index.js' (which is the active tab), and 'counter.test.js'. The 'index.js' code is as follows:

```
JS counter.js • JS index.js • JS counter.test.js
app > src > JS index.js ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import Counter from './counter';
4 import reportWebVitals from './reportWebVitals';

5
6 const root = ReactDOM.createRoot(document.getElementById('root'));
7 root.render(
8   <React.StrictMode>
9     <Counter />
10   </React.StrictMode>
11 );
12
13 // If you want to start measuring performance in your app, pass a function
14 // to log results (for example: reportWebVitals(console.log))
15 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
16 reportWebVitals();
17
```

Lines 3, 9, and 16 are highlighted with red boxes.

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

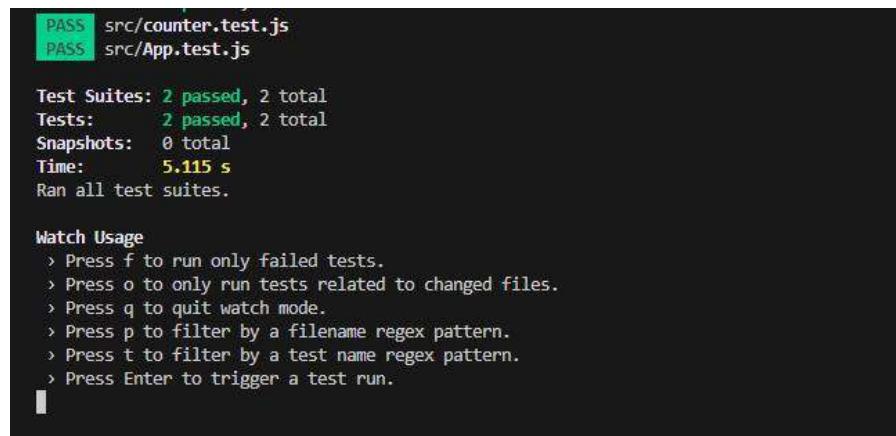


```
C:\Users\TEMP.DESKTOP-D9J3IR2.086\Desktop\reactapp\app>npm run test

> app@0.1.0 test
> react-scripts test

|
```

Output:



```
PASS  src/counter.test.js
PASS  src/App.test.js

Test Suites: 2 passed, 2 total
Tests:    2 passed, 2 total
Snapshots: 0 total
Time:      5.115 s
Ran all test suites.

Watch Usage
> Press f to run only failed tests.
> Press o to only run tests related to changed files.
> Press q to quit watch mode.
> Press p to filter by a filename regex pattern.
> Press t to filter by a test name regex pattern.
> Press Enter to trigger a test run.

|
```

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.

```
JS home.js  X JS index.js  # App.css
app1 > src > JS home.js > ↗ Form
  1 import { useState } from 'react';
  2 import './App.css';
  3 export default function Form() {
  4   const [name, setName] = useState('');
  5   const [email, setEmail] = useState('');
  6   const [password, setPassword] = useState('');
  7   const [submitted, setSubmitted] = useState(false);
  8
  9   const handleName = (e) => {
 10     setName(e.target.value);
 11   };
 12   const handleEmail = (e) => {
 13     setEmail(e.target.value);
 14   };
 15   const handlePassword = (e) => {
 16     setPassword(e.target.value);
 17   };
 18   const handleSubmit = (e) => {
 19     e.preventDefault();
 20     if (name === "" || email === "" || password === "") {
 21       alert("Please enter all the fields");
 22     } else {
 23       setSubmitted(true);
 24     }
 25     setName(e.target.value);
 26   };
 27   const successMessage = () => {
 28
 29     if (submitted)
 30       return (
 31         <div className="success">
 32           <h1>User {name} successfully registered!!</h1>
```

```
JS home.js • JS index.js • # App.css
app1 > src > JS home.js > ...
29     if (submitted)
30         return (
31             <div className="success">
32                 <h1>User {name} successfully registered!!</h1>
33             </div>
34     );
35     return (
36         <div className="form">
37             <div>
38                 <h1>User Registration</h1>
39             </div>
40             <div className="messages">
41                 {successMessage()}
42             </div>
43             <form>
44                 <fieldset>
45                     <label className="label">Name</label>
46                     <input onChange={handleName} className="input" value={name} type="text" /><br><br>
47                     <label className="label">Email</label>
48                     <input onChange={handleEmail} className="input" value={email} type="email" /><br><br>
49                     <label className="label">Password</label>
50                     <input onChange={handlePassword} className="input" value={password} type="password" /><br><br>
51                     <button onClick={handleSubmit} className="btn" type="submit">submit
52                 </fieldset>
53             </form>
54         </div>
55     );
56 }
57 }
```

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

```
JS home.js • JS index.js • # App.css •
app1 > src > # App.css > ...
1 .input {
2     width:30%;
3     padding:12px 20px;
4     margin:8px 0;
5     display:inline-block;
6     border:1px solid #ccc;
7     border-radius:4px;
8     box-sizing:border-box;
9 }
10
11
```

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

```
JS home.js • JS index.js • # App.css
app1 > src > JS index.js > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import Home from './home';
5 import reportWebVitals from './reportWebVitals';

6 const root = ReactDOM.createRoot(document.getElementById('root'));
7 root.render(
8     <React.StrictMode>
9         <Home />
10    </React.StrictMode>
11 );
12
13 // If you want to start measuring performance in your app, pass a function
14 // to log results (for example: reportWebVitals(console.log))
15 // or send to an analytics endpoint. Learn more: https://bit.ly/cRA-vitals
16 reportWebVitals();
17
18
19
20
```

Output:

React App x | +

localhost:3000

User Registration

Name

Email

Password 

React App x | +

localhost:3000

User Registration

User Brinda successfully registered!!

Name

Email

Password

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.

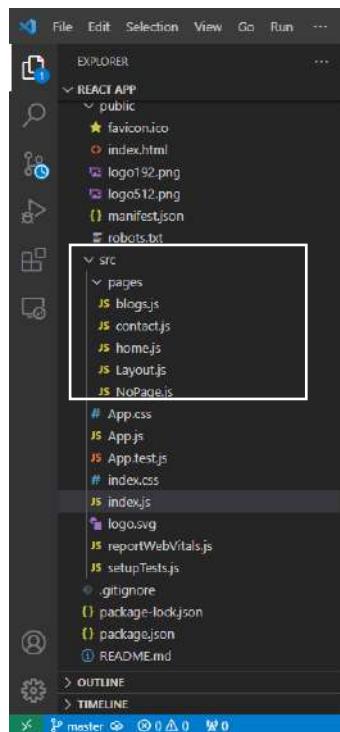
Step 1: Open the VS Code and launch the React on the local server.

• 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.

```
JS index.js •
router > src > JS index.js > ...
1 import ReactDOM from "react-dom/client";
2 import { BrowserRouter, Routes, Route } from "react-router-dom";
3 import Layout from "./pages/Layout";
4 import Home from "./pages/home";
5 import Blogs from "./pages/blogs";
6 import Contact from "./pages/contact";
7 import NoPage from "./pages/NoPage";
8 export default function App() {
9   return (
10     <BrowserRouter>
11       <Routes>
12         <Route path="/" element={<Layout />}>
13           <Route index element={<Home />} />
14           <Route path="blogs" element={<Blogs />} />
15           <Route path="contact" element={<Contact />} />
16           <Route path="*" element={<NoPage />} />
17         </Route>
18       </Routes>
19     </BrowserRouter>
20   );
21 }
22 const root = ReactDOM.createRoot(document.getElementById('root'));
23 root.render(<App />);
24
```

Step 5: Now add code to ‘App.css’ file to style your application.

```
JS index.js • # App.css •
router > src > # App.css > li a:hover:not(.active)
1 ul {
2   list-style-type: none;
3   margin: 0;
4   padding: 0;
5   overflow: hidden;
6   background-color: #00AA6D;
7 }
8
9 li {
10   float: left;
11   border-right:1px solid #bbb;
12 }
13
14 li a {
15   display: block;
16   color: white;
17   text-align: center;
18   padding: 14px 16px;
19   text-decoration: none;
20 }
21
22 li a:hover:not(.active) {
23   background-color: #111;
24 }
25
```

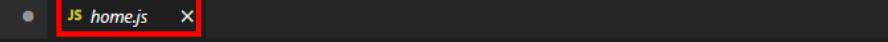
Step 6: Write the below code to ‘blogs.js’ file inside the ‘Pages’ folder.

```
JS index.js • JS blogs.js •
router > src > pages > JS blogs.js > [o] default
1 const Blogs = () => {
2   |   return <h1>Blog Articles</h1>
3 };
4 export default Blogs;
```

Step 7: Write the below code to ‘contact.js’ file inside the ‘Pages’ folder.

```
JS index.js • JS contact.js •
router > src > pages > JS contact.js > [o] default
1 const contact =() => {
2   |   return <h1>contact me</h1>
3 };
4 export default contact;
```

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



JS index.js ● JS home.js X

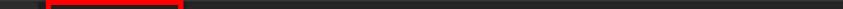
router > src > pages > JS home.js > [e] default

```
1 const Home =() => {
2   return <h1>Home</h1>
3 };
4 export default Home;
```

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

```
JS index.js • JS Layout.js ✘
router > src > pages > JS Layout.js > [x] default
  1 import { Outlet, Link } from "react-router-dom";
  2 const Layout = () => {
  3   return (
  4     <>
  5       <nav>
  6         <ul>
  7           <li> <Link to="/">Home</Link>
  8           <li>
  9             <Link to="/blogs">Blogs</Link>
 10           <li>
 11             <Link to="/contact">contact</Link>
 12           </li>
 13         </ul>
 14       </nav>
 15       <Outlet />
 16     </>
 17   )
 18 };
 19
 20
 21
 22
 23 export default Layout;
 24
```

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



```
JS index.js • JS NoPage.js X
router > src > pages > JS NoPage.js > default
1 const NoPage = () => {
2   return <h1>404</h1>;
3 };
4 export default NoPage;
5
6
7
```

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.

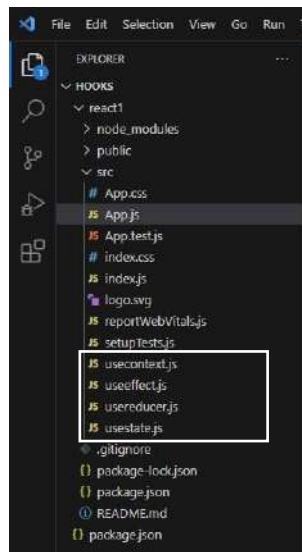
Syntax: `const myValue = useContext(MyContext);`

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.

```
react1 > src > JS App.js ...  
1 import Name from './usestate';  
2  
3 function App() {  
4   return (  
5     <div>  
6       <Name/>  
7     </div>  
8   );  
9 }  
10  
11 export default App;  
12  
13  
14 |
```

Step 4: Then add code in ‘usestate.js’ file within the ‘src’ folder.

```
react1 > src > JS usestate.js ...  
1 import { useState } from "react";  
2  
3 function Name() {  
4   const [name, setName] = useState("VVP_CS_FSD");  
5   const changeName = () => {  
6     setName("Full stack development");  
7   };  
8  
9   return (  
10     <div>  
11       <p>My name is {name}</p>  
12       <button onClick={changeName}> Click me </button>  
13     </div>  
14   );  
15 }  
16  
17 export default Name;  
18  
19
```

Output:



- **Example of UseEffect:**

Step 5: Write the below code in the ‘App.js’ file inside the ‘src’ folder.

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

Step 6: Then add code in ‘useeffect.js’ file within the ‘src’ folder.

```
JS App.js ● JS useeffect.js ✘
react1 > src > JS useeffect.js > ...
1 import React, { useState, useEffect } from 'react';
2
3 function CounterExample() {
4   const [count, setCount] = useState(0);
5
6   // Similar to componentDidMount and componentDidUpdate:
7   useEffect(() => {
8     // Update the document title using the browser API
9     document.title = `You clicked ${count} times`;
10   });
11
12   return (
13     <div>
14       <p>You clicked {count} times</p>
15       <button onClick={() => setCount(count + 1)}>
16         Click me
17       </button>
18     </div>
19   );
20 }
21
22 export default CounterExample;
```

Output:



- **Example of UseContext:**

Step 7: Write the below code in the ‘App.js’ file inside the ‘src’ folder.

```
JS App.js ● JS usecontext.js
react1 > src > JS App.js > ...
1 import Component1 from './usecontext';
2
3 function App() {
4   return (
5     <div>
6       <Component1/>
7     </div>
8   );
9 }
10
11 export default App
12
```

A screenshot of a code editor showing the content of the 'App.js' file. The file contains a single component named 'App' which returns a div containing a component named 'Component1'. The 'usecontext' file is shown in the tab bar but is not currently selected.

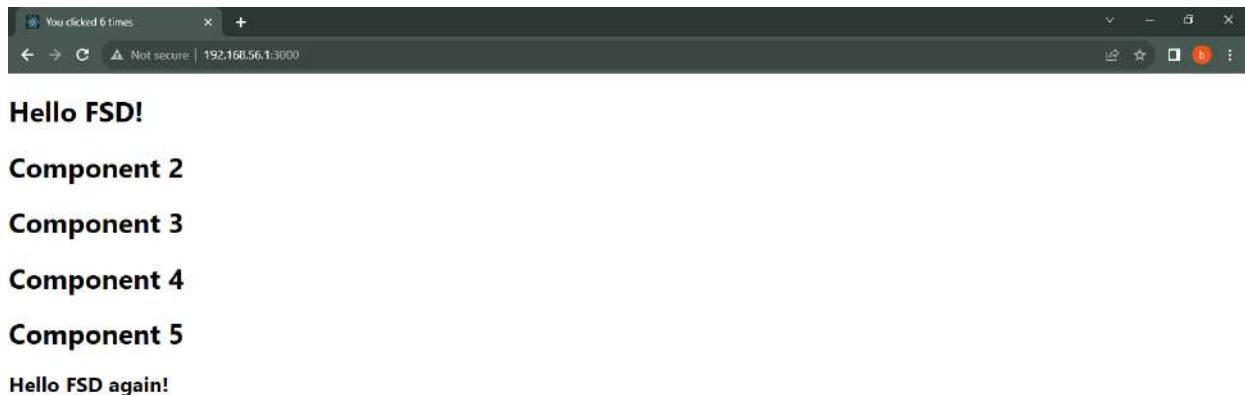
Step 8: Then add code in the ‘usecontext.js’ file within the ‘src’ folder.

```
JS App.js ● JS usecontext.js ✘
react1 > src > JS usecontext.js > ⚡ Component3
1 import { useState, createContext, useContext } from "react";
2
3 const UserContext = createContext();
4
5 function Component1() {
6   const [user, setUser] = useState("FSD");
7
8   return (
9     <UserContext.Provider value={user}>
10       <h1>Hello {user}!</h1>
11       <Component2 />
12     </UserContext.Provider>
13   );
14 }
15
16 function Component2() {
17   return (
18     <>
19       <h1>Component 2</h1>
20       <Component3 />
21     </>
22   );
23 }
24
25 function Component3() {
26   return (
27     <>
28       <h1>Component 3</h1>
29       <Component4 />
30     </>
31   );
32 }
33
34 function Component4() {
35   return (
36     <>
37       <h1>Component 4</h1>
38       <Components />
39     </>
40   );
41 }
```

A screenshot of a code editor showing the content of the 'usecontext.js' file. The file defines a context provider 'UserContext' and four components: 'Component1', 'Component2', 'Component3', and 'Component4'. 'Component1' sets the initial user state to 'FSD' and provides it to 'Component2'. 'Component2' contains 'Component3'. 'Component3' contains 'Component4'. The 'usecontext' file is shown in the tab bar and is currently selected.

```
JS App.js ● JS usecontext.js X
react1 > src > JS 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.

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

Step 10: Then add code in the ‘userreducer.js’ file within the ‘src’ folder.

```
JS App.js • JS usereducer.js •
react1 > src > JS usereducer.js > ...
1 import React, { useReducer } from "react";
2
3 // Defining the initial state and the reducer
4 const initialState = 0;
5 const reducer = (state, action) => {
6   switch (action) {
7     case "add":
8       return state + 1;
9     case "subtract":
10       return state - 1;
11     case "reset":
12       return 0;
13     default:
14       throw new Error("Unexpected action");
15   }
16 };
17
18 const AppReducer = () => {
19   // Initialising useReducer hook
20   const [count, dispatch] = useReducer(reducer, initialState);
21   return (
22     <div>
23       <h2>{count}</h2>
24       <button onClick={() => dispatch("add")}>
25         add
26       </button>
27       <button onClick={() => dispatch("subtract")}>
28         subtract
29       </button>
30       <button onClick={() => dispatch("reset")}>
31         reset
32       </button>
33     </div>
34   );
35 };
36 export default AppReducer;
37
```

Output:

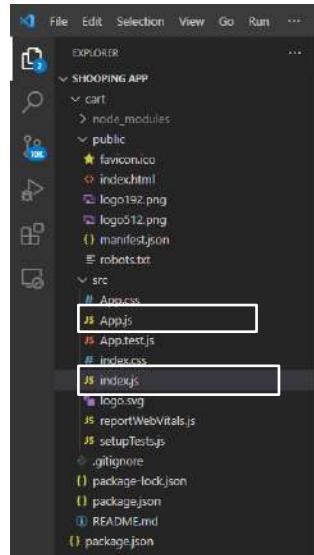
The screenshots show a browser window with three tabs, each displaying a different state of a simple counter application running at localhost:3000.

- Tab 1:** Shows the initial state with a count of 0. Below the count, there are three buttons: "add", "subtract", and "reset".
- Tab 2:** Shows the state after three "add" button clicks. The count is now 6. The same three buttons are present below the count.
- Tab 3:** Shows the state after three "subtract" button clicks. The count is now 3. The same three buttons are present below the count.

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.

```
JS App.js ● JS index.js ●
cart > src > JS index.js > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6 const root = ReactDOM.createRoot(document.getElementById('root'));
7 root.render(
8   <React.StrictMode>
9     <App/>
10   </React.StrictMode>
11 );
12
13
```

Step 3: Then write the below code in the ‘App.js’ within the ‘src’ folder.

```
JS App.js ● JS index.js
cart > src > JS App.js > ...
1 import { useState } from "react";
2 function App() {
3   const [list, setList] = useState([]);
4   const [value, setValue] = useState("");
5   const addToList = () => {
6     let tempArr = list;
7     tempArr.push(value);
8     setList(tempArr);
9     setValue("");
10   };
11   const deleteItem = (index) => {
12     let temp = list.filter((item, i) => i !== index);
13     setList(temp);
14   };
15   return (
16     <div className="App">
17       <fieldset>
18         <h>Add Product to List</h><br><br>
19         <input type="text" value={value} onChange={(e) => setValue(e.target.value)} />
20         <button onClick={addToList}> Click to Add </button><br><br><br>
21         <h>Product Catalog</h><br><br>
22         <ol>
23           <li onClick={() => deleteItem(1)}>{item} </li>
24         </ol>
25         <h>Click on Product to Delete</h><br><br>
26       </fieldset></div>
27     );
28 } 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.

About 116,000,000 search results

[Java | Oracle](#)

Download Import Oracle Java License Information... Download Java for Windows

Update Yes, updating to Java 7, using Auto Upd... Security Remove Older Versions

Java Uninstall Tool Java Uninstall Tool - Java | Oracle Help

Oracle Academy Oracle Academy recently expanded its curriculum to include...

What is Java What is Java technology and why do I n... Support Options Using Java

Linux 64-Bit Installation I... Unpack the tarball and install Java tar zxvf...

Java (software platform) Set of several computer software products and specifications

See results about

[Java \(programming language\)](#) Concept

[Java version history](#) Wikimedia list article

[java.com](#)

Java is a set of computer software and specifications developed by James Gosling at Sun Microsystems that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise

Step 2: Now click on 'Download Java'.

The screenshot shows the official Java website at java.com/en/. The header includes links for 'Download', 'Developer Resources', and 'Help'. A search bar is on the right. Below the header, a message says 'Manual update required for some Java 8 users on macOS'. A large heading 'Get Java for desktop applications' is followed by a green 'Download Java' button, which is highlighted with a red box. Below the button are links for 'What is Java?' and 'Uninstall help'. To the right is a photo of a woman smiling while using a laptop. A dark banner at the bottom asks if you're a developer looking for JDK downloads, with links for 'OpenJDK Early Access Builds' and 'Java SE Development Kit'. The footer contains copyright information and links for 'Select Language', 'Support', 'Privacy', 'Cookie Preferences', 'Terms of Use', and 'Trademarks'.

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**'.

The screenshot shows the Java download page at java.com/en/download/. The left sidebar has a 'Help Resources' section with links for 'What is Java?', 'Remove older versions', 'Disable Java', 'Error messages', 'Troubleshoot Java', and 'Other help'. The main content area features a section for '64-bit Java for Windows' with a green 'Download Java' button, which is highlighted with a red box. This section includes a link to 'Version 8 Update 381 (filesize: 62.63 MB)' and a note about why Java 8 is recommended. Below this is a yellow box containing 'Important Oracle Java License Information' and a note about the license change starting April 16, 2019. It also mentions that commercial license and support is available with a low-cost Java SE Subscription. At the bottom, there's a note about acknowledging the terms of the Oracle Technology Network License Agreement.

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



Step 5: Wait until the installation is completed



Step 6: Click on 'close'.



- **Setting up of Environment Variables for Java.**

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.



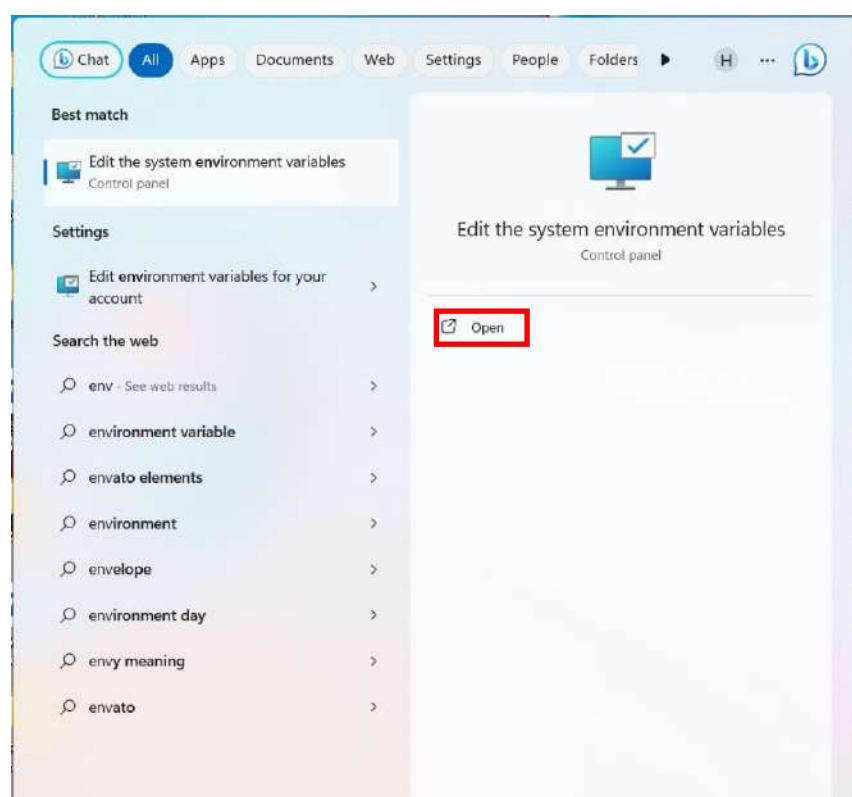
Step 2: Now click on 'bin' folder.

Name	Date modified	Type	Size
bin	9/5/2023 7:50 AM	File folder	
conf	9/5/2023 7:50 AM	File folder	
include	9/5/2023 7:50 AM	File folder	
jmods	9/5/2023 7:50 AM	File folder	
legal	9/5/2023 7:50 AM	File folder	
lib	9/5/2023 7:50 AM	File folder	
jenkinserr	9/29/2023 8:26 AM	Text Document	169 KB
jenkins	8/23/2023 7:45 AM	Application	606 KB
jenkins.exe	8/23/2023 1:34 PM	Configuration.Sou...	1 KB
jenkins.out	9/6/2023 9:41 PM	Text Document	1 KB
Jenkins.war	8/23/2023 1:26 PM	WAR file	87,435 KB
jenkins.wrapper	9/6/2023 9:41 PM	Text Document	4 KB
jenkins	9/5/2023 7:52 AM	XML Document	3 KB
LICENSE	9/5/2023 7:50 AM	File	7 KB
README	9/5/2023 7:50 AM	File	1 KB
release	9/5/2023 7:50 AM	File	2 KB

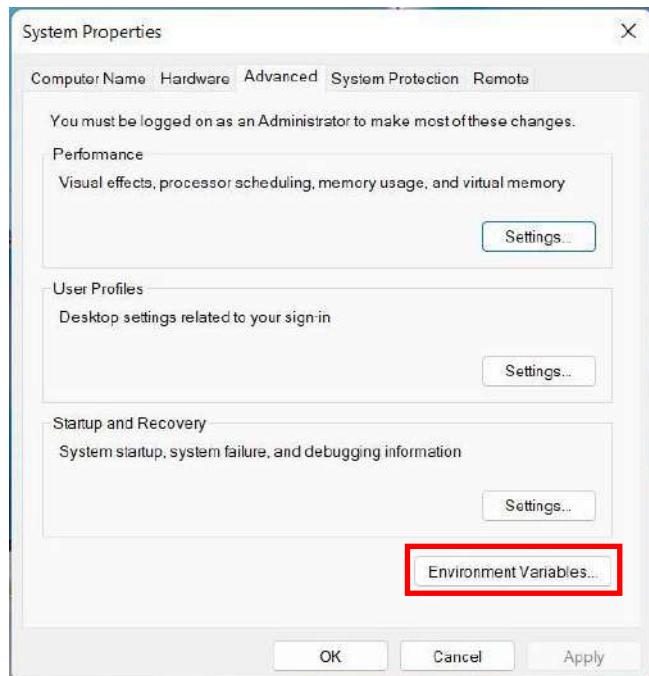
Step 3: Then copy the copy the bin location here.



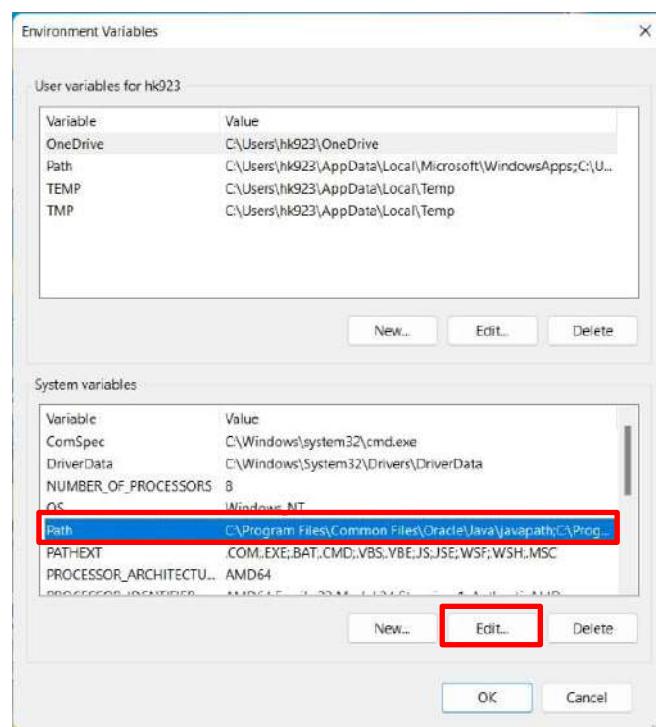
Step 4: Go to windows search bar Search for '**Environment Variables**' & click on Open



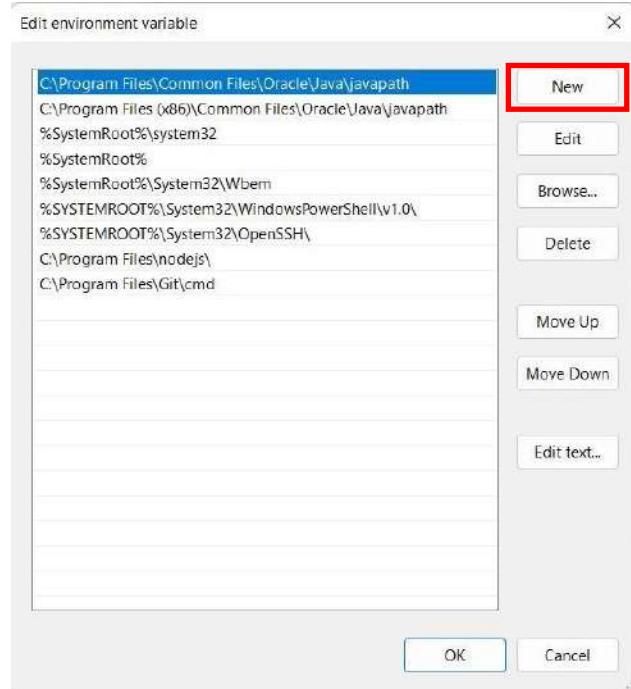
Step 5: Now click on ‘Environment Variables’ here.



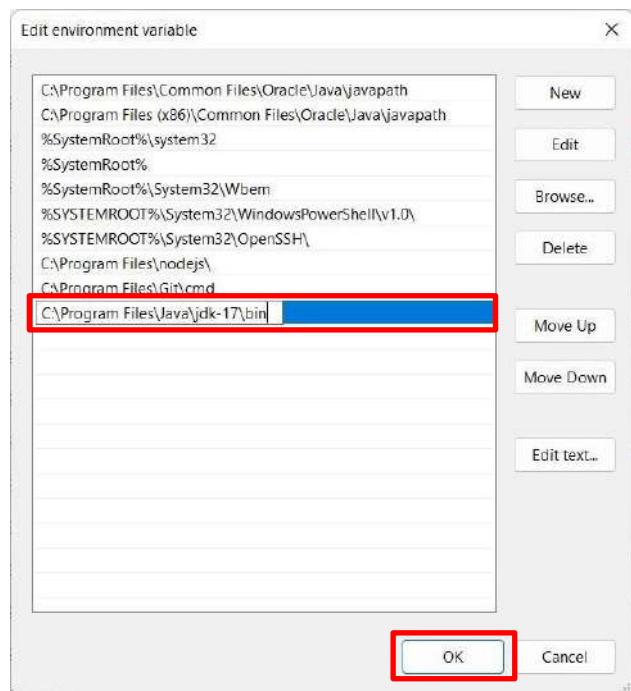
Step 6: Under System Variables, select Path location & click on ‘Edit’.



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.

```
C:\> Command Prompt
Microsoft Windows [Version 10.0.22000.2295]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hk923>java -version
java version "17.0.8" 2023-07-18 LTS
Java(TM) SE Runtime Environment (build 17.0.8+9-LTS-211)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.8+9-LTS-211, mixed mode, sharing)
```

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

```

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

C:\Users\vk923>javac
Usage: javac <options> <source files>
where possible options include:
  -<filename>           Read options and filenames from file
  -Akey[=value]           Options to pass to annotation processors
  --add-modules <module>[,<module>]*   Root modules to resolve in addition to the initial modules, or all modules
  on the module path if <module> is ALL-MODULE-PATH.
  --boot-class-path <path>, --bootclasspath <path>
  Override location of bootstrap class files
  --class-path <path>, -classpath <path>, -cp <path>
  Specify where to find user class files and annotation processors
  -d <directory>         Specify where to place generated class files
  -deprecation           Output source locations where deprecated APIs are used
  -enable-preview         Enable preview language features. To be used in conjunction with either -source or --release.
  -encoding <encoding>    Specify character encoding used by source files
  -endorseddirs <dirs>    Override location of endorsed standards path
  -extdirs <dirs>          Override location of installed extensions
  -g <lines,vars,source>   Generate all debugging info
  -g:none                Generate only some debugging info
  -g:none                Generate no debugging info
  -h <directory>         Specify where to place generated native header files
  -help, -help, -?        Print this help message
  -help-extra, -X         Print help on extra options
  -implicit:(none,class) Specify whether or not to generate class files for implicitly referenced files

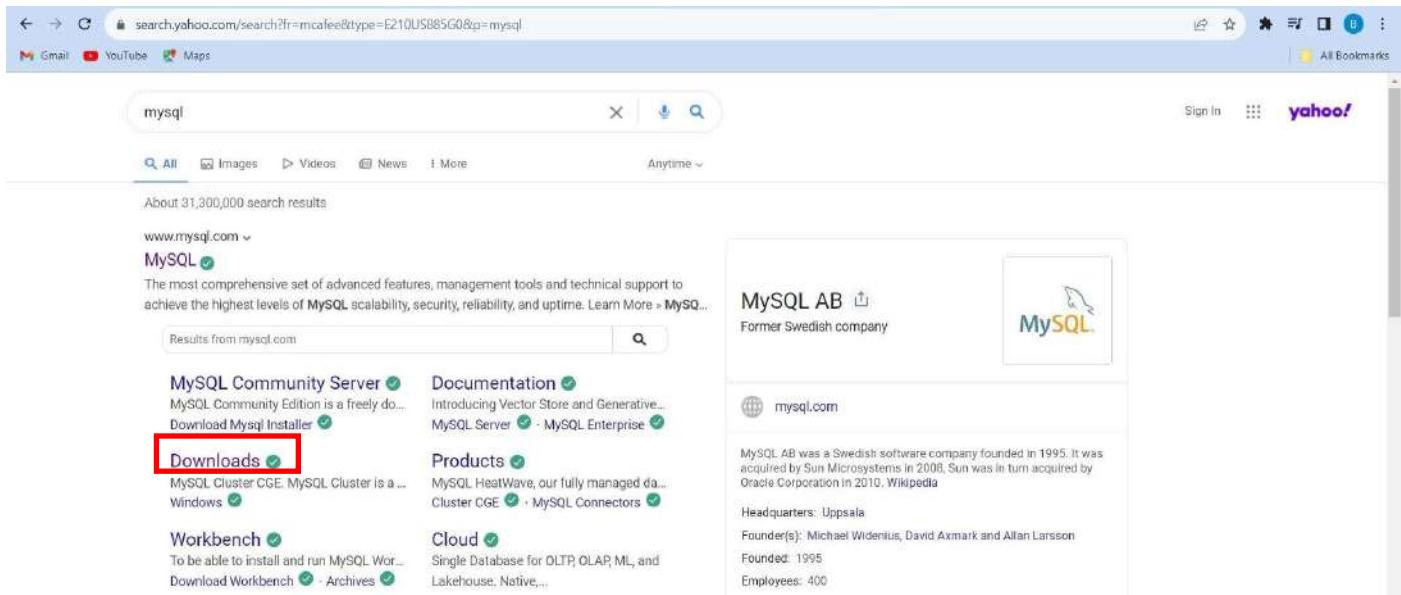
```

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’.



The screenshot shows a Yahoo search results page for 'mysql'. The search bar contains 'mysql'. Below it, there are filters for 'All', 'Images', 'Videos', 'News', and 'More'. The results count is 'About 31,300,000 search results'. The first result is 'www.mysql.com' with a link to 'MySQL'. The page content includes sections for 'MySQL Community Server', 'Documentation', 'Products', 'Cloud', and 'Workbench'. Under 'MySQL Community Server', there is a 'Downloads' link which is highlighted with a red box. The right side of the page has a sidebar for 'MySQL AB' with information about the company's history, headquarters, founders, and employees.

Step 2: Click on ‘MySQL Community (GPL) Downloads’.

MySQL Newsletter

[Subscribe »](#)

[Archive »](#)

Free Webinars

What's New with MySQL and MySQL HeatWave
(Announcements at CloudWorld)
Thursday, October 05, 2023

Virtual Conference: Machine Learning for Beginners - From Data to Insights
Thursday, October 05, 2023

MySQL Security from Data Protection to Regulation Compliance
Thursday, October 19, 2023

[More »](#)

MySQL Enterprise Edition

MySQL Enterprise Edition includes the most comprehensive set of advanced features, management tools and technical support for MySQL.

[Learn More »](#)

[Customer Download »](#)

[Trial Download »](#)

MySQL Cluster CGE

MySQL Cluster is a real-time open source transactional database designed for fast, always-on access to data under high throughput conditions.

- [MySQL Cluster](#)
- [MySQL Cluster Manager](#)
- [Plus, everything in MySQL Enterprise Edition](#)

[Learn More »](#)

[Customer Download »](#) (Select Patches & Updates Tab, Product Search)

[Trial Download »](#)

[MySQL Community \(GPL\) Downloads »](#)

Step 3: Now click on ‘MySQL Installer for Windows’.

④ MySQL Community Downloads

- [MySQL Yum Repository](#)
- [MySQL APT Repository](#)
- [MySQL SUSE Repository](#)
- [MySQL Community Server](#)
- [MySQL Cluster](#)
- [MySQL Router](#)
- [MySQL Shell](#)
- [MySQL Operator](#)
- [MySQL NDB Operator](#)
- [MySQL Workbench](#)
- [MySQL API \(libmysqlclient\)](#)
- [Connector/C++](#)
- [Connector/J](#)
- [Connector/.NET](#)
- [Connector/Node.js](#)
- [Connector/ODBC](#)
- [Connector/Python](#)
- [MySQL Native Driver for PHP](#)
- [MySQL Benchmark Tool](#)
- [Time zone description tables](#)
- [Download Archives](#)
- [MySQL Installer for Windows](#)

ORACLE © 2023 Oracle.

[Privacy](#) | [Do Not Sell My Info](#) | [Terms of Use](#) | [Trademark Policy](#) | [Cookie Preferences](#)

Step 4: Then select the second option & click on ‘Download’.

④ MySQL Community Downloads

[MySQL Installer](#)

The screenshot shows the MySQL Community Downloads page with the 'MySQL Installer' section selected. At the top, there are tabs for 'General Availability (GA) Releases' and 'Archives'. Below the tabs, it says 'MySQL Installer 8.0.34'. A note states: 'MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.' There are dropdown menus for 'Select Version' (set to '8.0.34') and 'Select Operating System' (set to 'Microsoft Windows'). Below these are two download options:

Version	File Type	Size	Action
Windows (x86, 32-bit), MSI Installer	(mysql-installer-web-community-8.0.34.0.msi)	8.0.34 2.4M	Download
Windows (x86, 32-bit), MSI Installer	(mysql-installer-community-8.0.34.0.msi)	8.0.34 331.3M	Download

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

④ MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- » Fast access to MySQL software downloads
- » Download technical White Papers and Presentations
- » Post messages in the MySQL Discussion Forums
- » Report and track bugs in the MySQL bug system



No thanks, just start my download.

ORACLE © 2023 Oracle

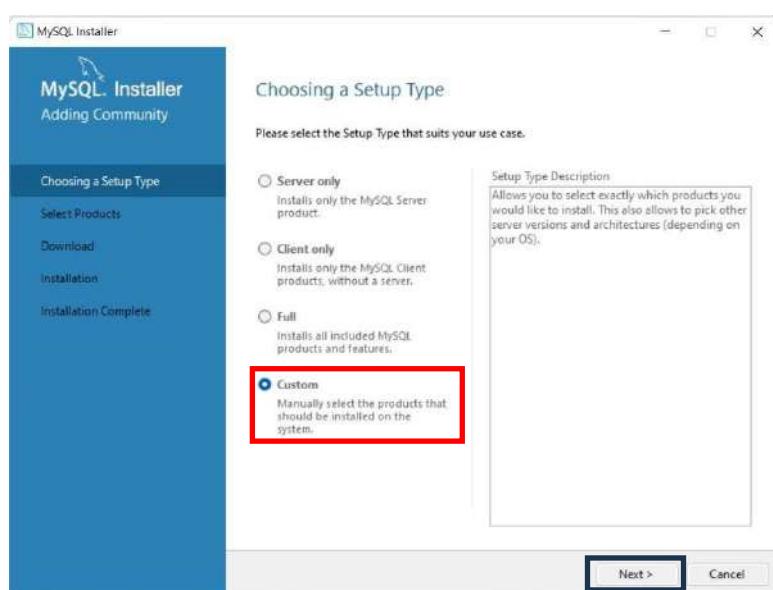
[Privacy / Do Not Sell My Info](#) | [Terms of Use](#) | [Trademark Policy](#) | [Cookie Preferences](#)

Step 6: Validating the internet connections.

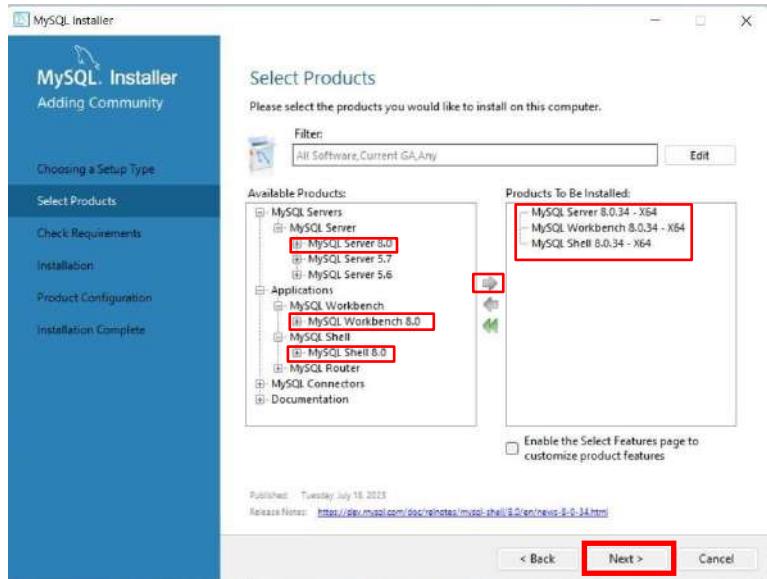


Installing the MySQL on both client & server side interfaces.

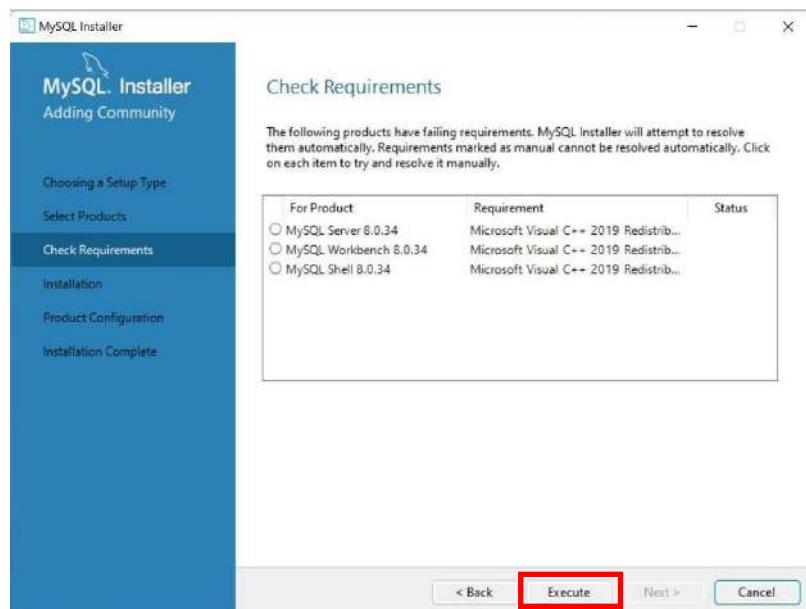
Step 7: Select ‘Custom’ & click on ‘Next’.



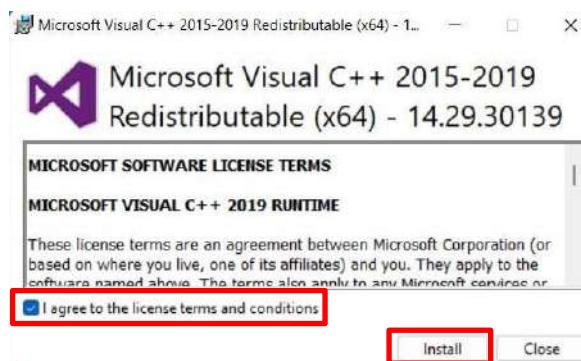
Step 8: Select the MySQL products from Available products to Products to be installed stage & click on ‘Next’.



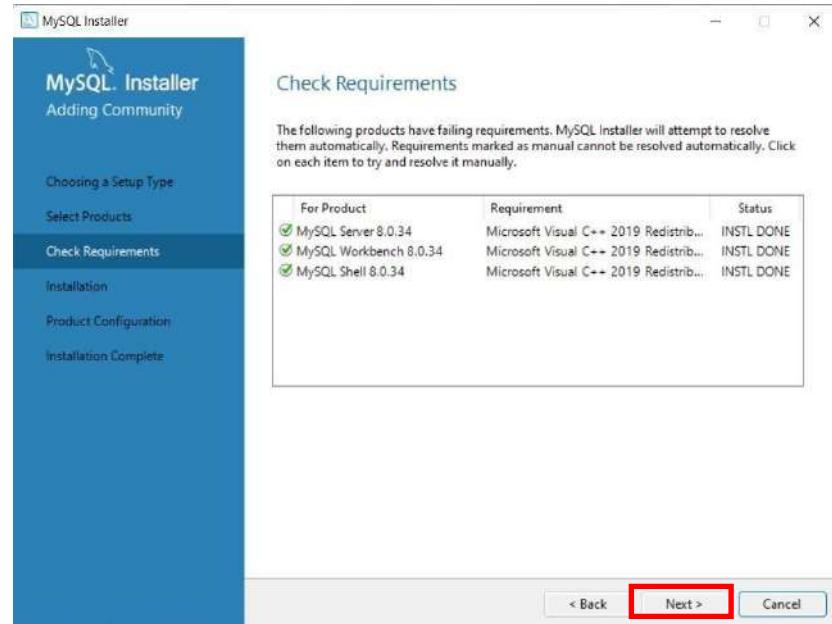
Step 9: Click on ‘Execute’.



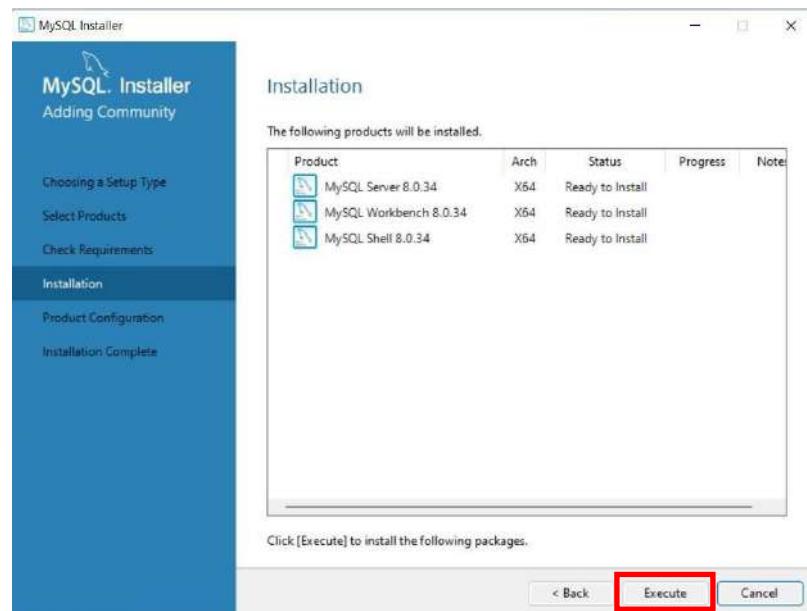
Step 10: Enable the option below & click on ‘Install’.



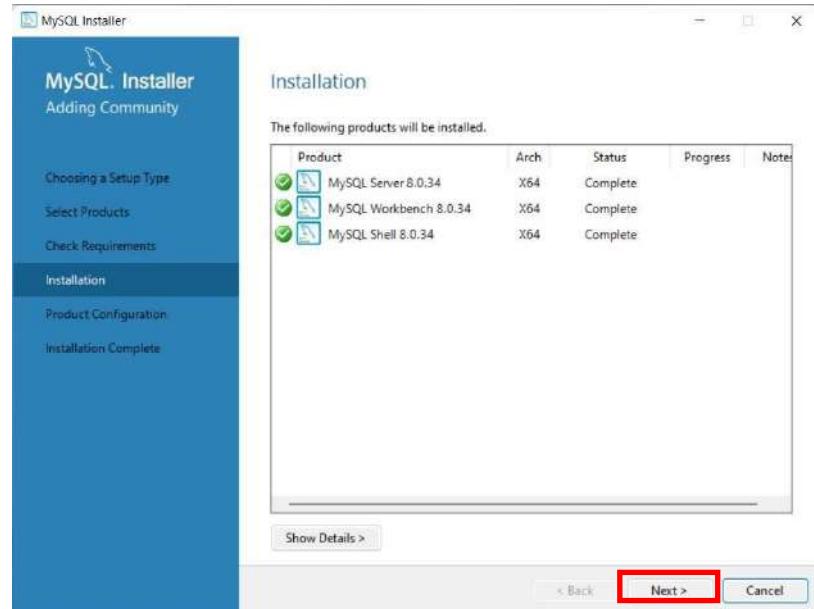
Step 11: After the installation of the products are complete, click on ‘Next’.



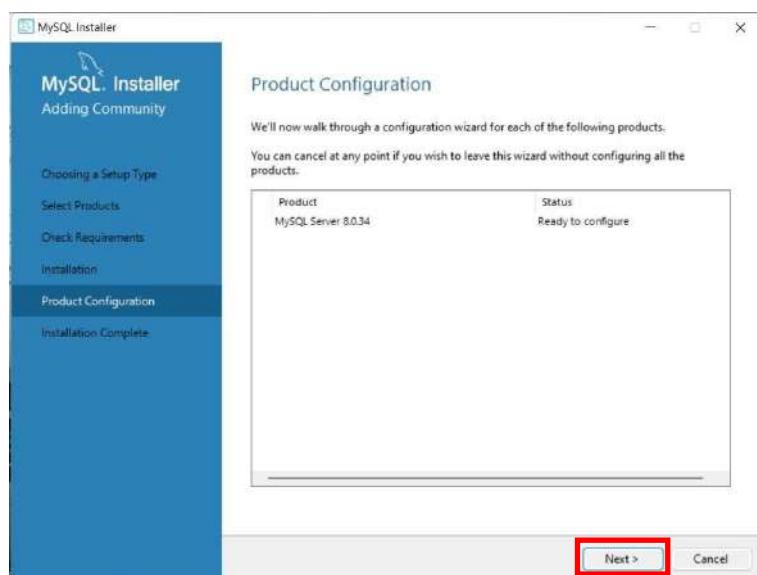
Step 12: Click on ‘Execute’.



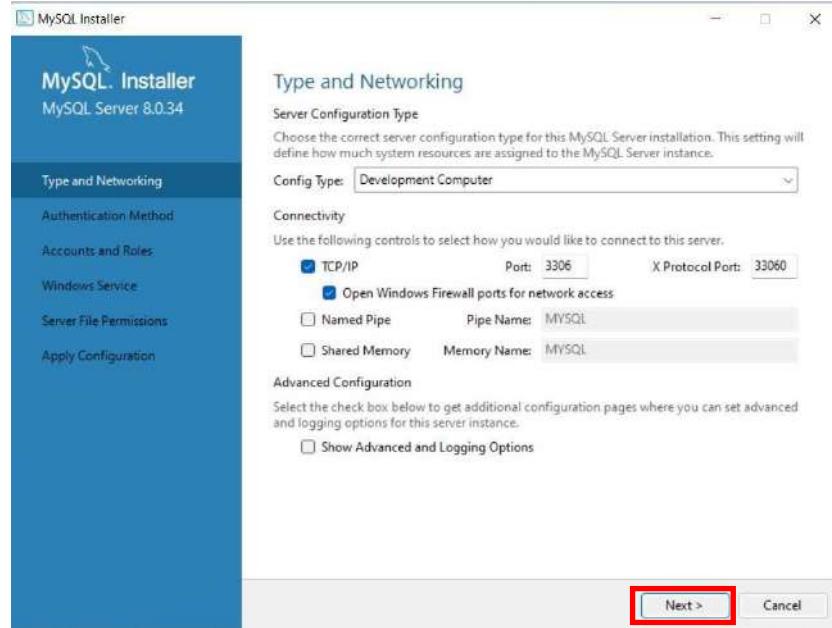
Step 13: After the complete execution, click on ‘Next’.



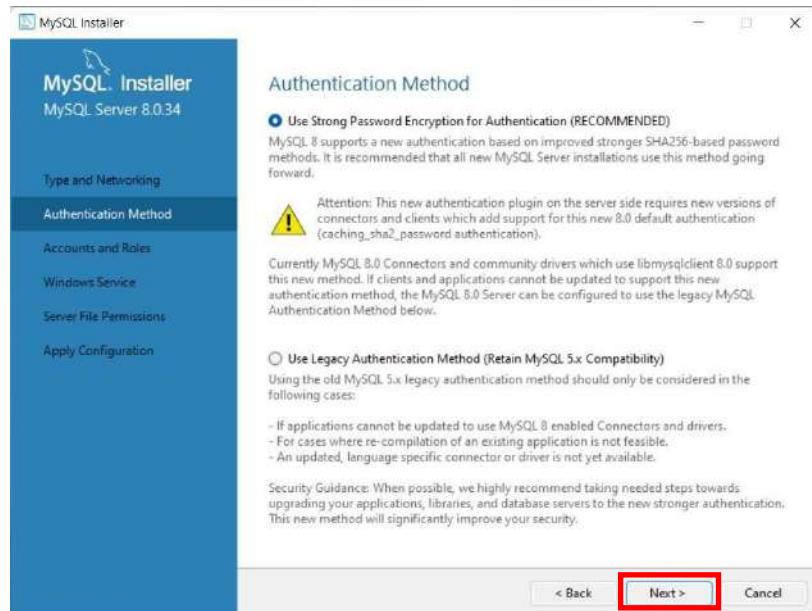
Step 14: Click on 'Next'.



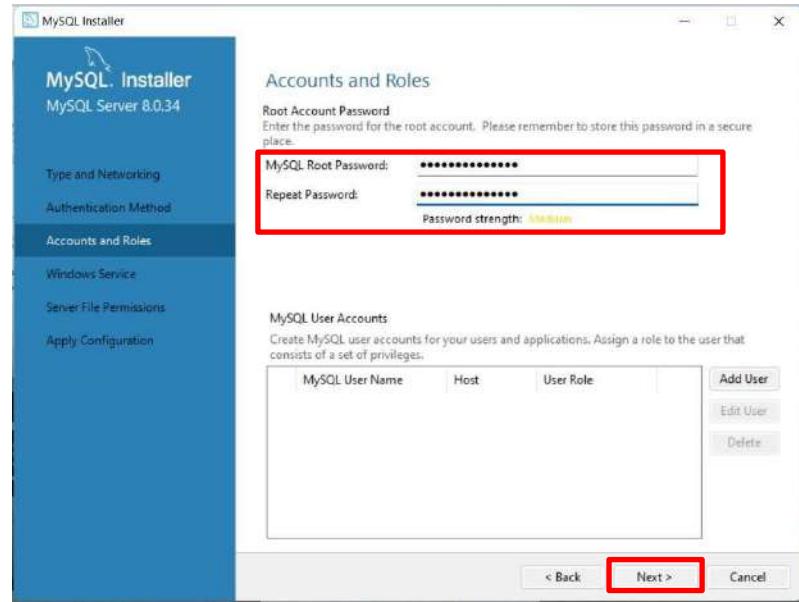
Step 15: Don't change the default selections, simply click on 'Next'.



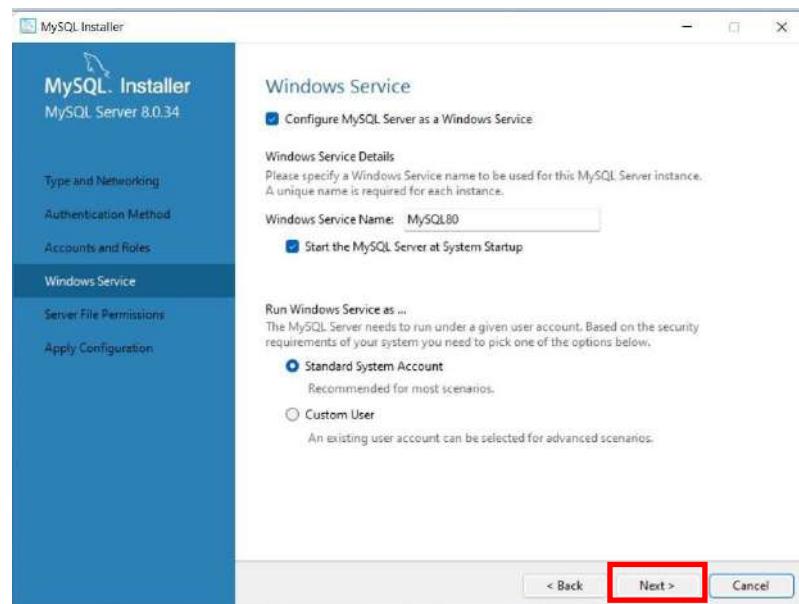
Step 16: Click on ‘Next’.



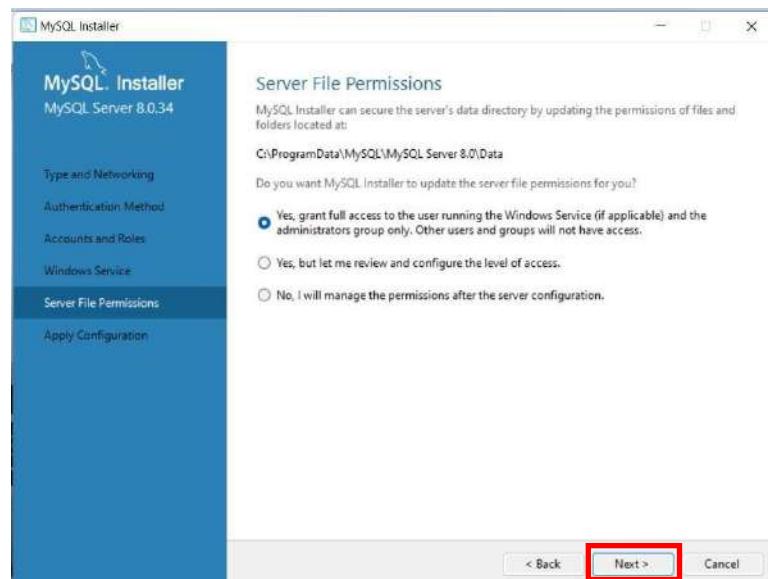
Step 17: Create a strong password for MySQL server & click on ‘Next’.



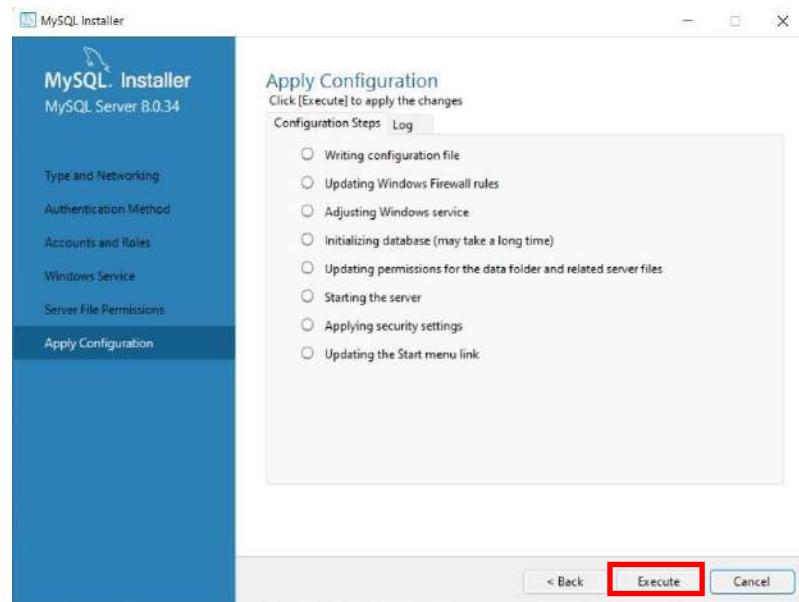
Step 18: Then click on ‘Next’.



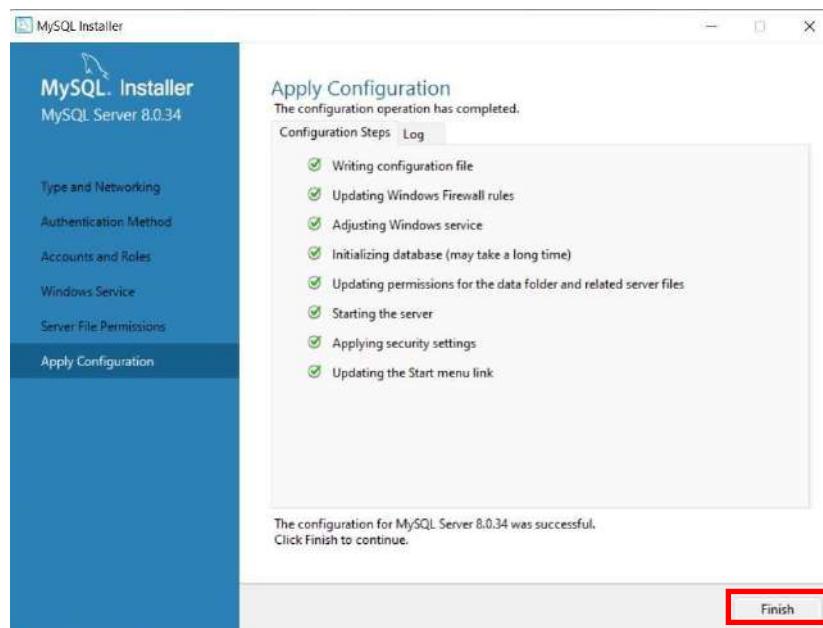
Step 19: Now click on ‘Next’.



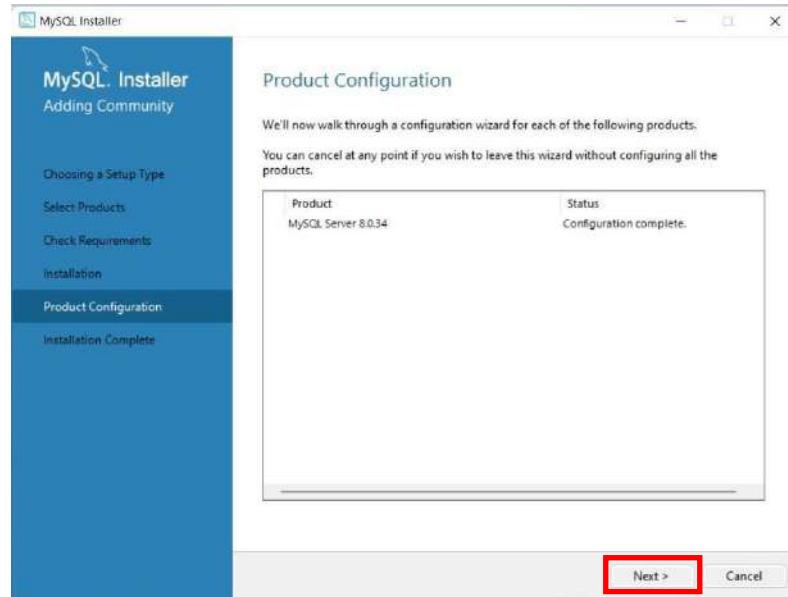
Step 20: Click on ‘Execute’.



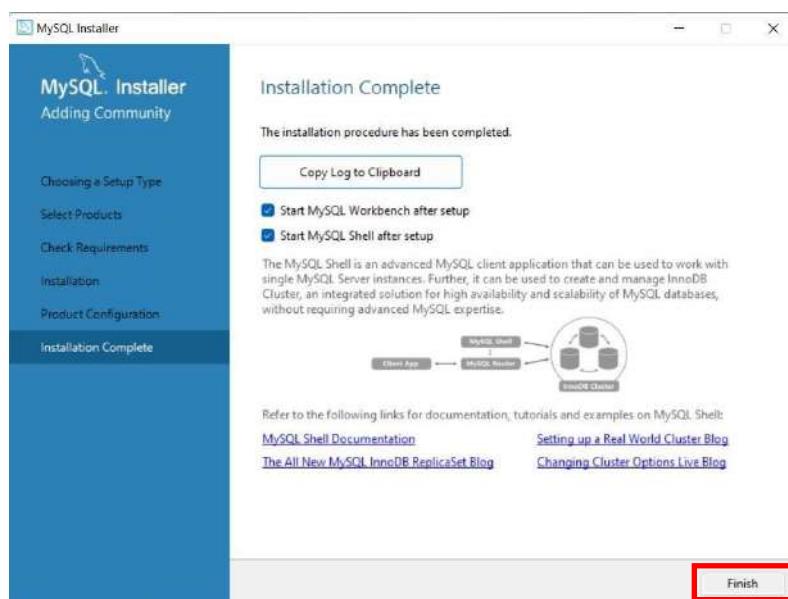
Step 21: Simply click on ‘Finish’.



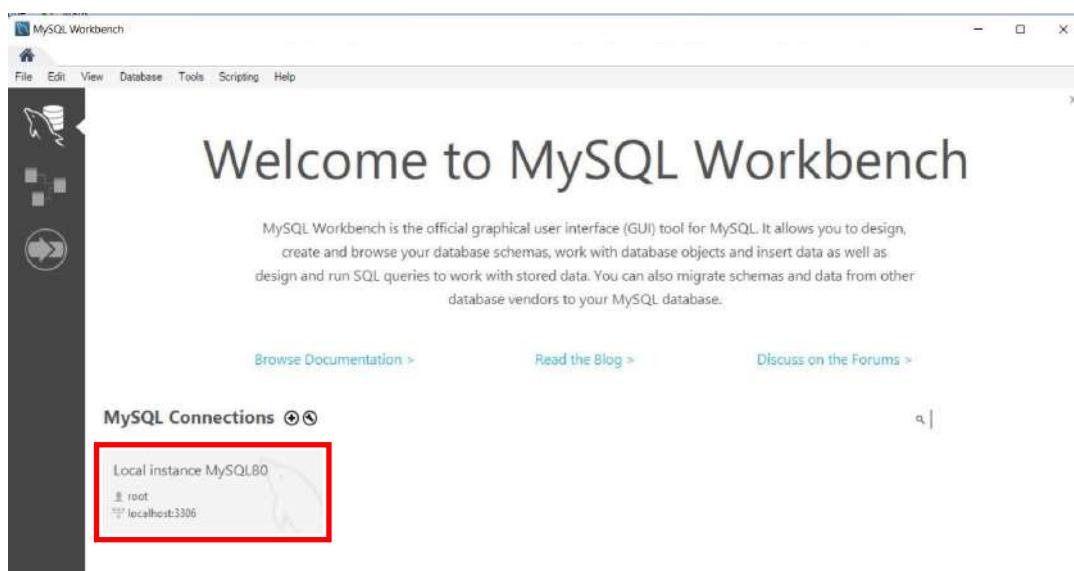
Step 22: Click on ‘Next’.



Step 23: Click on 'Finish'.



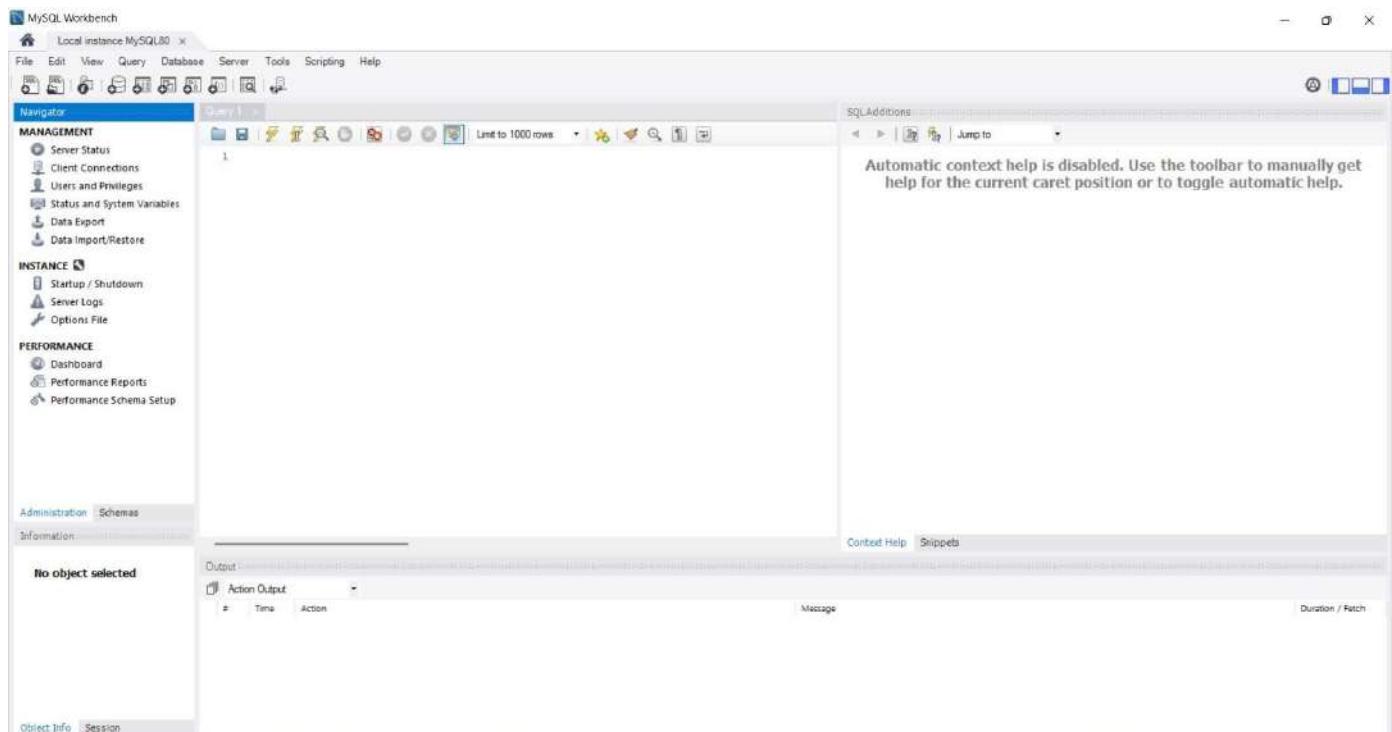
Step 24: Now click on 'Local instance MySQLBO' here.



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.

```
Command Prompt - mysql -u root -p
Microsoft Windows [Version 10.0.22000.2295]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hk923>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 8.0.34 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

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

```
C:\ Command Prompt - mysql -u root -p  
Microsoft Windows [Version 10.0.22000.2295]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\hk923>mysql --version  
mysql Ver 8.0.34 for Win64 on x86_64 (MySQL Community Server - GPL)
```

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

```
C:\Users\hk923>mysql -u root -p  
Enter password: *****  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 21  
Server version: 8.0.34 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2023, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

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

```
C:\ Command Prompt - mysql -u root -p  
  
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| sys |  
+-----+  
4 rows in set (0.09 sec)
```

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

```
mysql> create database Employee;  
Query OK, 1 row affected (0.08 sec)
```

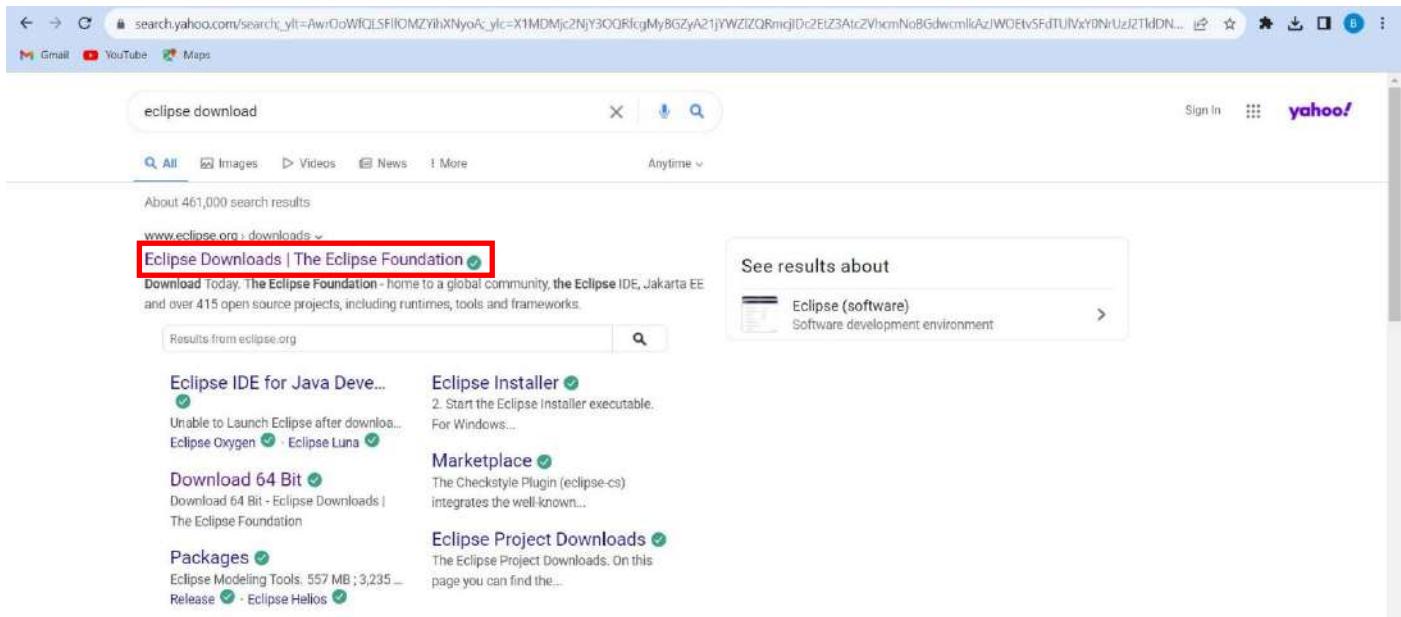
Step 5: Again enter the command ‘**show databases**’ to view the created databases as shown below.

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

WEEK – 7

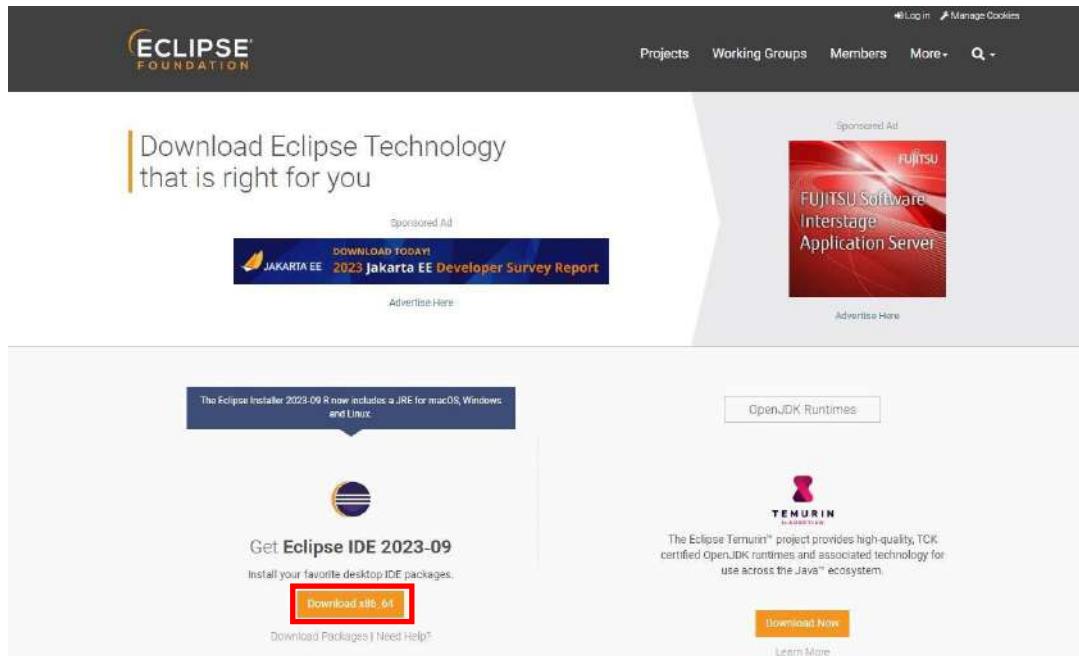
1. Installation of Eclipse IDE for Enterprise Java and Web Developers latest version on Windows.

Step 1: Go to Web Browser □ Search for ‘Eclipse download’ & click on the first link here.



The screenshot shows a Yahoo search results page for 'eclipse download'. The search bar at the top contains 'eclipse download'. Below it, there are filters for 'All', 'Images', 'Videos', 'News', and 'More'. The results section starts with a summary: 'About 461,000 search results' and 'www.eclipse.org › downloads'. The first result is 'Eclipse Downloads | The Eclipse Foundation' (highlighted with a red box), followed by a brief description: 'Download Today. The Eclipse Foundation - home to a global community, the Eclipse IDE, Jakarta EE and over 415 open source projects, including runtimes, tools and frameworks.' Below this are several other links: 'Eclipse IDE for Java Dev...', 'Eclipse Installer', 'Marketplace', 'Eclipse Project Downloads', and 'Packages'.

Step 2: Now click on ‘Download x86_64’ option.



The screenshot shows the Eclipse Foundation website. At the top, there's a navigation bar with 'Log in', 'Manage Cookies', 'Projects', 'Working Groups', 'Members', 'More', and a search bar. The main content area features a banner with 'Download Eclipse Technology that is right for you'. Below it, there's a 'Sponsored Ad' for 'FUJITSU Software Interstage Application Server'. The central part of the page is divided into two main sections: 'Get Eclipse IDE 2023-09' on the left and 'The Eclipse Temurin™ project' on the right. The 'Get Eclipse IDE 2023-09' section includes a note about the installer including a JRE for macOS, Windows, and Linux, a 'Download x86_64' button (which is highlighted with a red box), and links for 'Download Packages' and 'Need Help?'. The 'The Eclipse Temurin™ project' section includes a note about providing high-quality, TCK-certified OpenJDK runtimes, a 'Download Now' button, and a 'Learn More' link.

Step 3: Then click on ‘Download’.



[Log in](#) [Manage Cookies](#)

Projects

Working Groups

Members

More

Q

Home / Downloads / Eclipse downloads - Select a mirror

All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.

[Download](#)

Download from: Japan - Japan Advanced Institute of Science and Technology ([https](https://))

File: [eclipse-inst-jre-win64.exe](#) SHA-512

>> Select Another Mirror

Sponsored Ad



[Advertise Here](#)

Step 4: Open the Downloaded file.



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

The screenshot shows the **eclipseinstaller** website by Oomph. A search bar at the top has the placeholder "type filter text". Below it is a list of Eclipse IDE variants:

- Eclipse IDE for Java Developers**: Tools for any Java developer, including a Java IDE, Git client, XML Editor, Maven and Gradle integration.
- Eclipse IDE for Enterprise Java and Web Developers**: Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, YAML, Markdown, Web Services, JPA and Data Tools, Maven and Gradle. This option is highlighted with a red border.
- Eclipse IDE for C/C++ Developers**: An IDE for C/C++ developers.
- Eclipse IDE for Embedded C/C++ Developers**: An IDE for Embedded C/C++ developers. It includes managed cross build plug-ins (Arm and RISC-V) and debug plug-ins (SEGGER J-Link, OpenOCD,...).
- Eclipse IDE for PHP Developers**: The essential tools for any PHP developer, including PHP language support, Git client and editors for JavaScript, TypeScript, HTML, CSS and...

Step 6: Now click on ‘Install’.

The screenshot shows the configuration screen for the **Eclipse IDE for Enterprise Java and Web Developers**. It includes the following settings:

- Java 17+ VM**: Set to `C:\Program Files\Java\jdk-17`.
- Installation Folder**: Set to `C:\Users\hk923\eclipse\jee-2023-09\`.
- checkboxes**:
 - create start menu entry
 - create desktop shortcut
- INSTALL** button: A large orange button with a downward arrow icon, which is highlighted with a red border.

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.

A screenshot of a Google search results page. The search query "spring initializr" is entered in the search bar. The top result is a link to "Spring Initializr" from "https://start.spring.io". This link is highlighted with a red rectangle. Below the link, the text "Initializer generates spring boot project with just what you need to start quickly!" is visible. The "Spring Initializr" link is also highlighted with a red rectangle. The page shows approximately 3,51,000 results in 0.28 seconds.

Step 2: Now select the below options.

A screenshot of the Spring Initializr web interface. At the top, there's a navigation menu with three horizontal bars and the text "spring initializr". Below this, there are two main sections: "Project" and "Language". Under "Project", there are two radio buttons: "Gradle - Groovy" and "Gradle - Kotlin", with "Gradle - Groovy" being selected. Under "Language", there are three radio buttons: "Java" (selected), "Kotlin", and "Groovy". A red box highlights the "Java" button. Below these sections are "Spring Boot" and "Dependency" sections. In the "Spring Boot" section, several radio buttons are shown: "3.2.0 (SNAPSHOT)", "3.2.0 (M3)", "3.1.5 (SNAPSHOT)" (selected, highlighted with a green dot), "3.1.4", "3.0.12 (SNAPSHOT)", "3.0.11", "2.7.17 (SNAPSHOT)", and "2.7.16". A red box highlights the "3.1.4" button. The "Dependency" section is partially visible at the bottom.

Step 3: In the Project Metadata section, give the name as ‘springboot’.

Project Metadata

Group	com.springboot
Artifact	spring boot
Name	spring boot
Description	Demo project for Spring Boot
Package name	com.springboot.spring boot
Packaging	<input checked="" type="radio"/> Jar <input type="radio"/> War
Java	<input type="radio"/> 21 <input checked="" type="radio"/> 17 <input type="radio"/> 11 <input type="radio"/> 8

Step 4: Now click on ‘ADD DEPENDENCIES’.

Dependencies

ADD DEPENDENCIES... CTRL + B



Step 5: Under the ‘Web’ section, select the ‘Spring Web’.

Web, Security, JPA, Actuator, Devtools... Press Ctrl for multiple adds

DEVELOPER TOOLS

WEB

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

Spring Reactive Web
Build reactive web applications with Spring WebFlux and Netty

Spring for GraphQL
Build GraphQL applications with Spring for GraphQL and GraphQL Java.

Rest Repositories
Exposing Spring Data repositories over REST via Spring Data REST.

Spring Session
Provides an API and implementations for managing user session information.

Step 7: Now click on ‘GENERATE’ option.

GENERATE CTRL + G

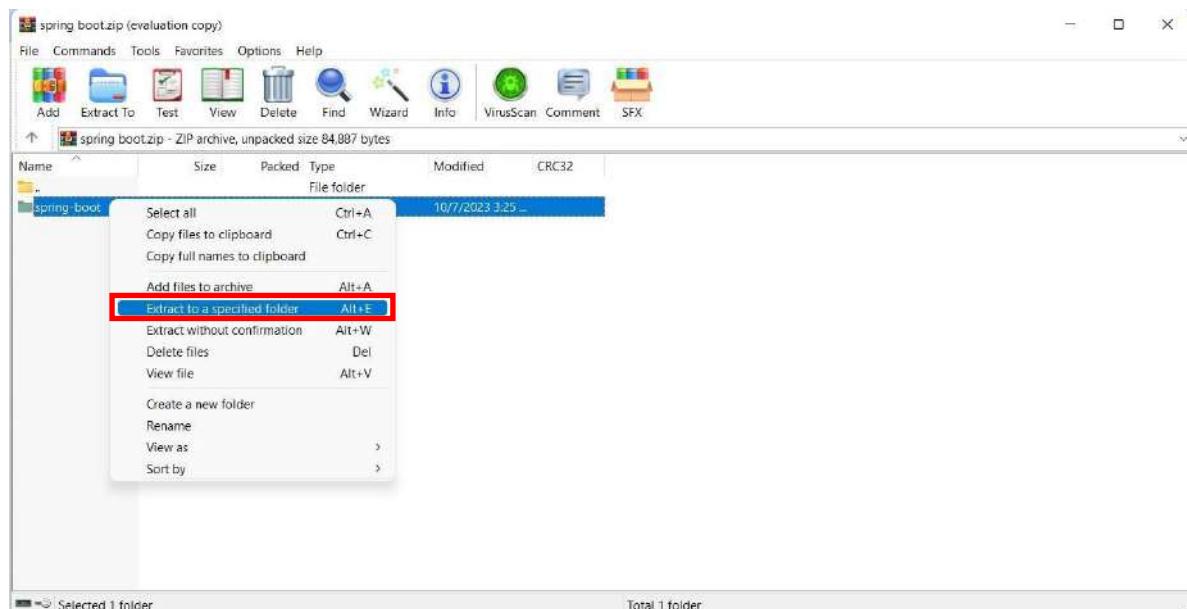
EXPLORE CTRL + SPACE

SHARE...

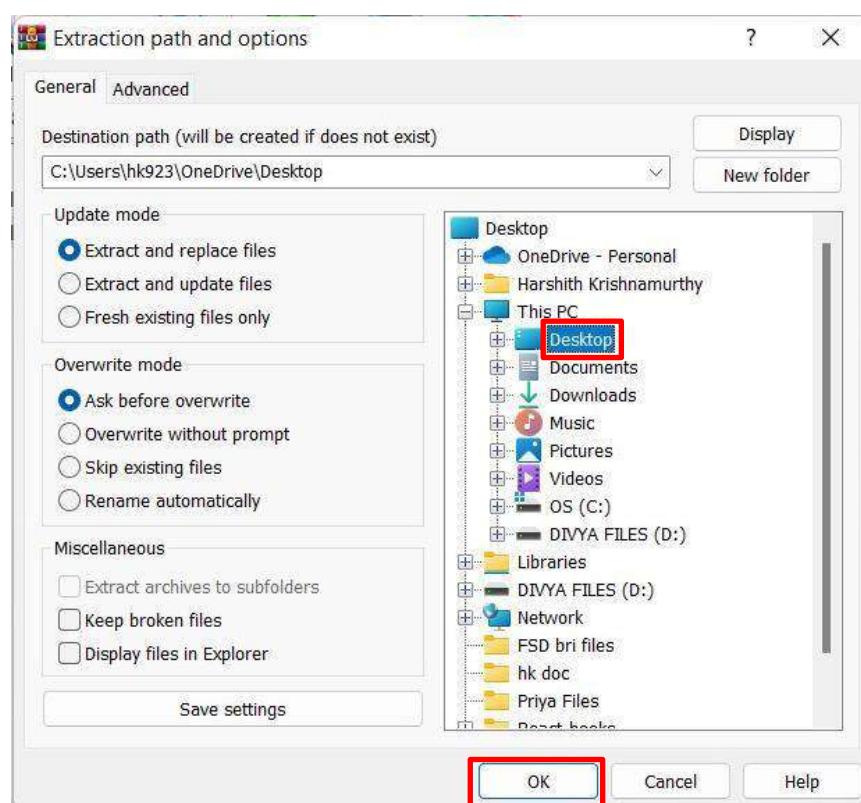
Step 8: Click on the zip file downloaded.



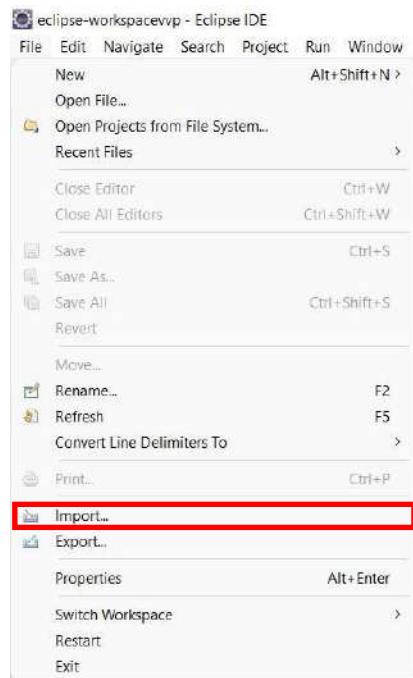
Step 9: Now extract the zip file which was downloaded.



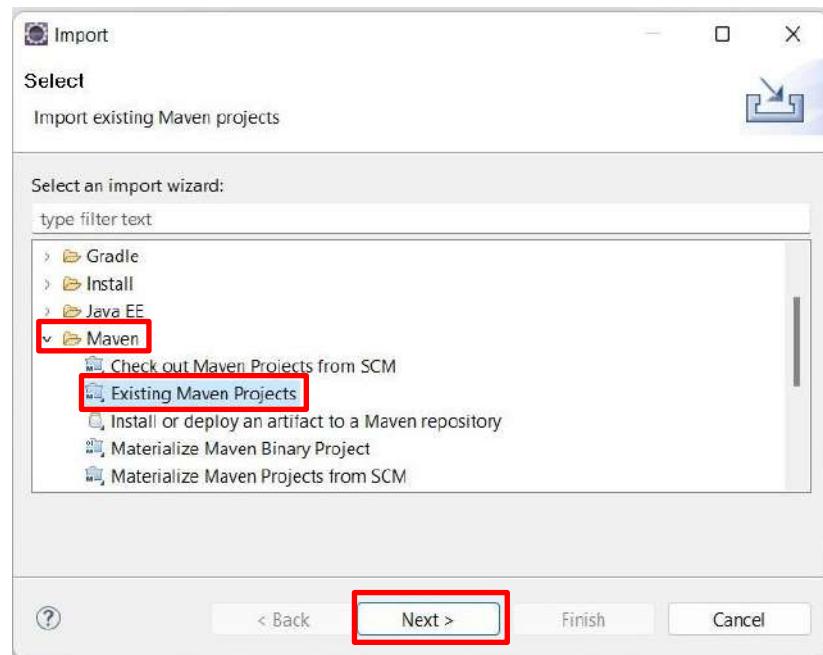
Step 10: Specify the location to save the file & click on 'OK'.



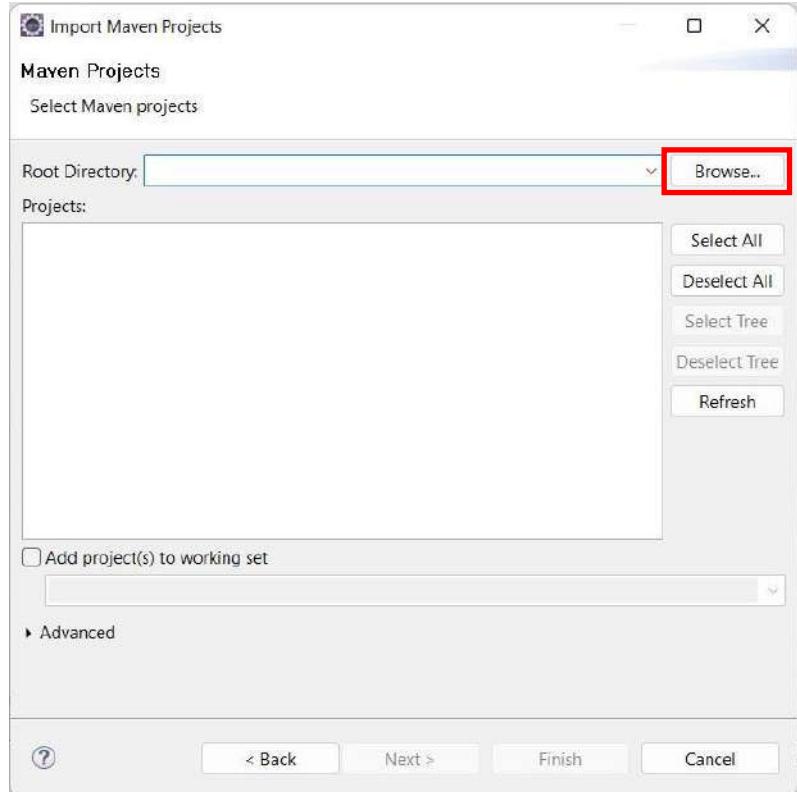
Step 11: Now launch the Eclipse IDE, go to **File** → click on ‘Import’ option.



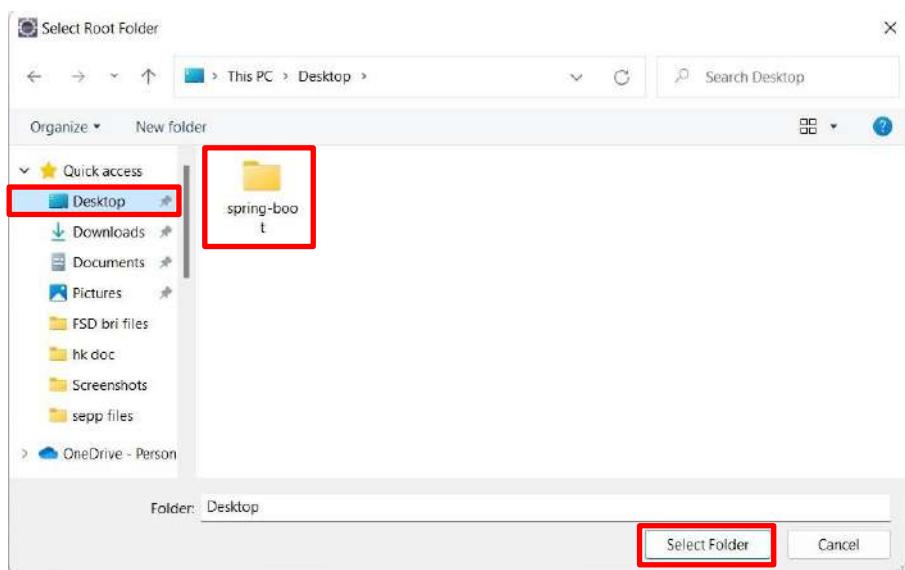
Step 12: Then select the ‘Existing Maven Projects’ under the ‘Maven’ folder & click on ‘Next’.



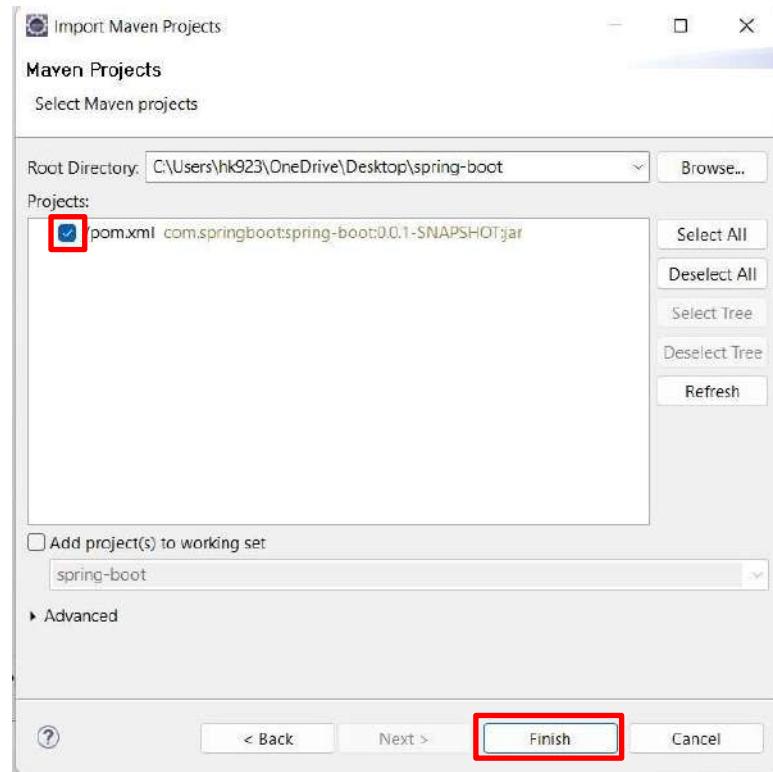
Step 13: Click on ‘Browse’ to specify the Root Directory.



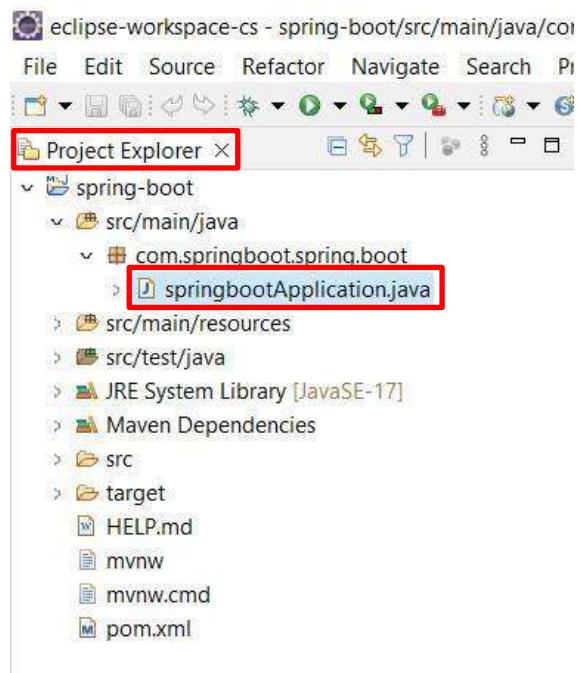
Step 14: Now select the location of the file where the zip file was extracted.



Step 15: Now click on 'Finish'.



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



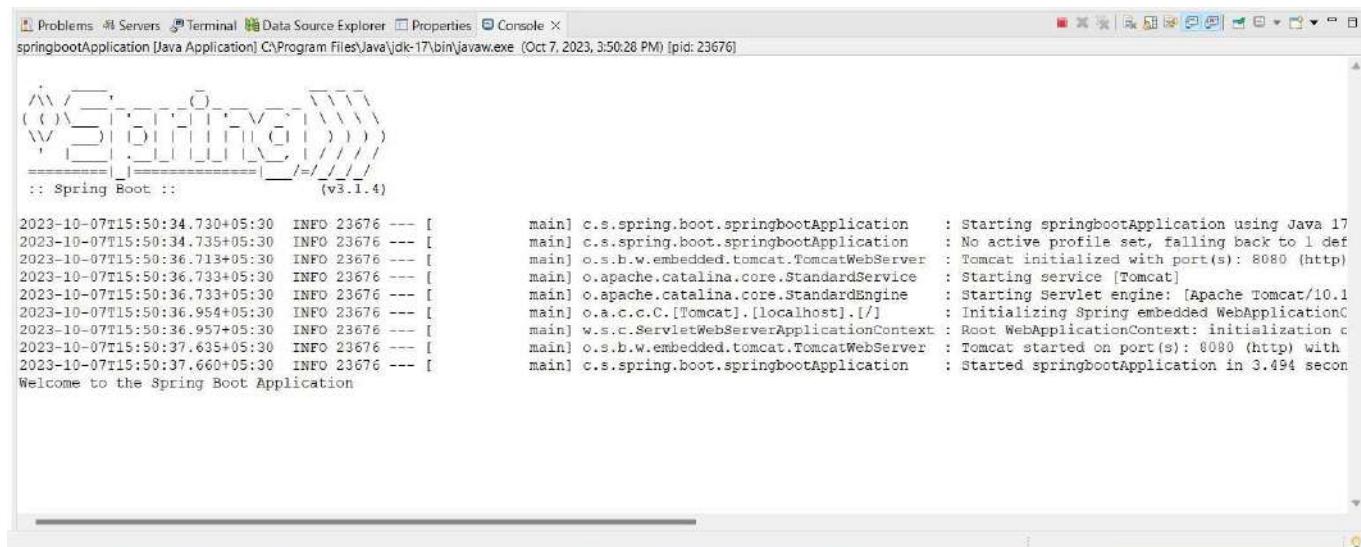
Step 17: Then add the below code to this file & Save it.

springbootApplication.java X

```
1 package com.springboot.spring.boot;
2
3+import org.springframework.boot.SpringApplication;□
4
5
6 @SpringBootApplication
7 public class springbootApplication {
8
9@   public static void main(String[] args) {
10      SpringApplication.run(springbootApplication.class, args);
11      System.out.println("Welcome to the Spring Boot Application");
12   }
13 }
14
```

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:

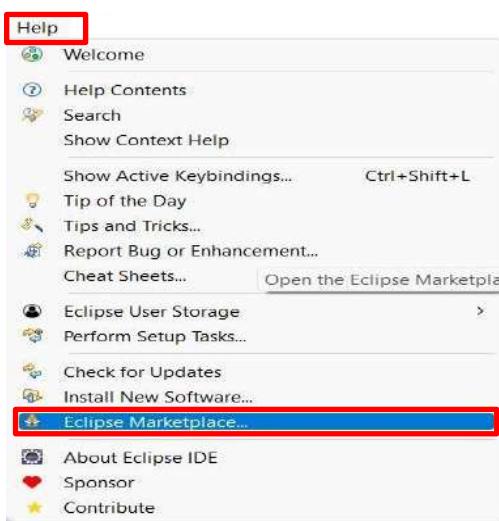


```
Problems Servers Terminal Data Source Explorer Properties Console ×
springbootApplication [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Oct 7, 2023, 3:50:26 PM) [pid: 23676]

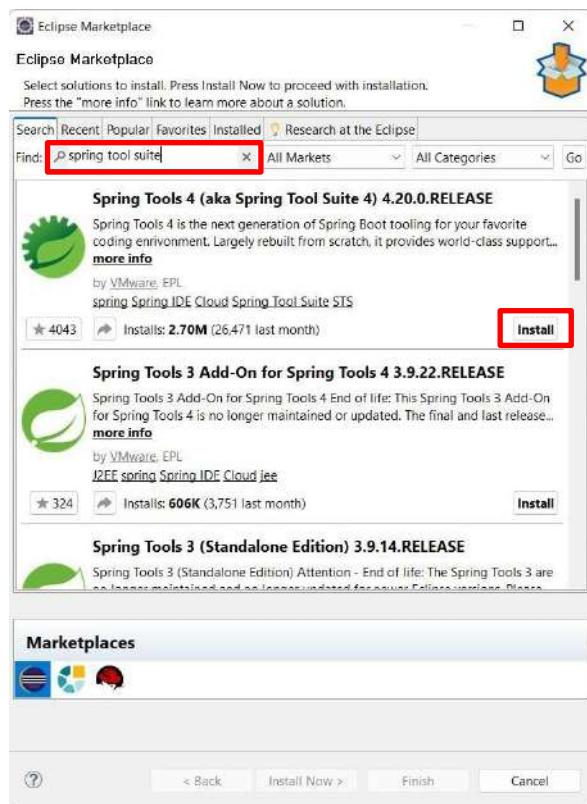
:: Spring Boot :: (v3.1.4)

2023-10-07T15:50:34.730+05:30 INFO 23676 --- [           main] c.s.spring.boot.springbootApplication : Starting springbootApplication using Java 17
2023-10-07T15:50:34.735+05:30 INFO 23676 --- [           main] c.s.spring.boot.springbootApplication : No active profile set, falling back to 1 def
2023-10-07T15:50:36.713+05:30 INFO 23676 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-10-07T15:50:36.733+05:30 INFO 23676 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-10-07T15:50:36.733+05:30 INFO 23676 --- [           main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1
2023-10-07T15:50:36.954+05:30 INFO 23676 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationC
2023-10-07T15:50:36.957+05:30 INFO 23676 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization c
2023-10-07T15:50:37.635+05:30 INFO 23676 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with
2023-10-07T15:50:37.660+05:30 INFO 23676 --- [           main] c.s.spring.boot.springbootApplication : Started springbootApplication in 3.494 secon
Welcome to the Spring Boot Application
```

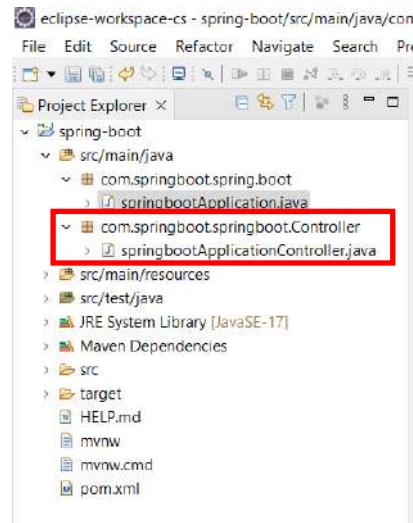
Step 19: Go to ‘Help’ □ ‘Eclipse Market place’ for downloading the required spring tools.



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 ‘springbootApplication.java’ file to view the output.

The screenshot shows the Eclipse IDE interface. In the top left, there are two tabs: 'springbootApplication.java' and 'springbootController.java'. The 'springbootController.java' tab is highlighted with a red box. The code editor displays Java code for a Spring Boot controller. The outline view on the right shows the package 'springboot.Controller' containing the class 'springbootController' with a method 'HelloWorld()'. The terminal view at the bottom shows the application's startup logs.

```
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 }
```

Spring boot application logs:

```
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]
2023-10-07T16:30:35.306+05:30 INFO 13128 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1
2023-10-07T16:30:35.310+05:30 INFO 13128 --- [main] w.s.c.ServletWebServerApplicationContext : Initializing Spring embedded WebApplicationContext
2023-10-07T16:30:36.221+05:30 INFO 13128 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Root WebApplicationContext: initialization completed in 4 ms
2023-10-07T16:30:36.249+05:30 INFO 13128 --- [main] c.s.spring.boot.springbootApplication : Tomcat started on port(s): 8080 (http) with
2023-10-07T16:30:45.224+05:30 INFO 13128 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[] : Started springbootApplication in 4.397 secon
2023-10-07T16:30:45.224+05:30 INFO 13128 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Spring DispatcherServlet 'dispa
2023-10-07T16:30:45.226+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
```

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

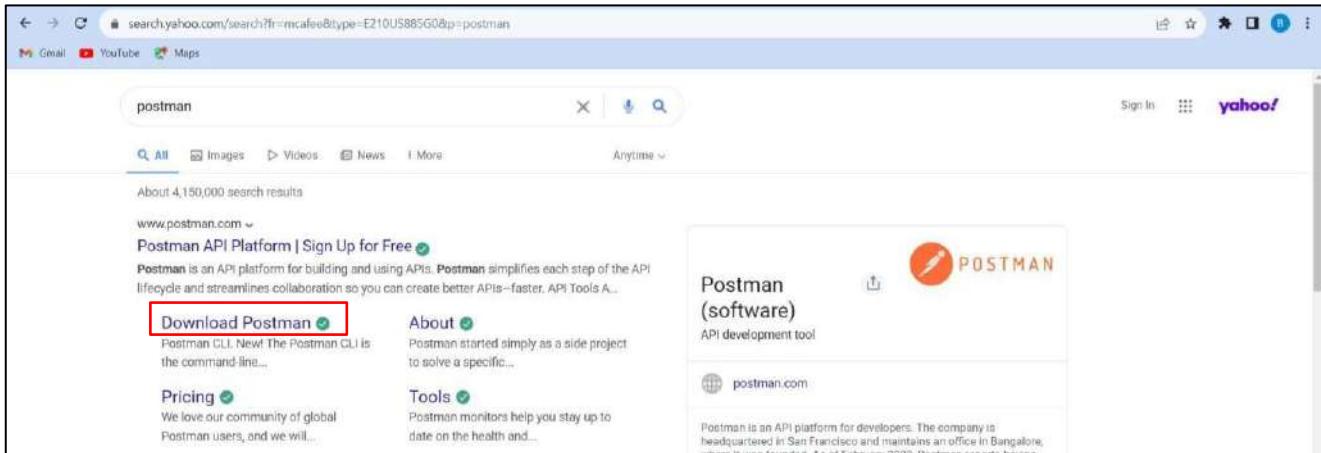
Output:



WEEK – 8

1. How to create RESTful service & Install Postman

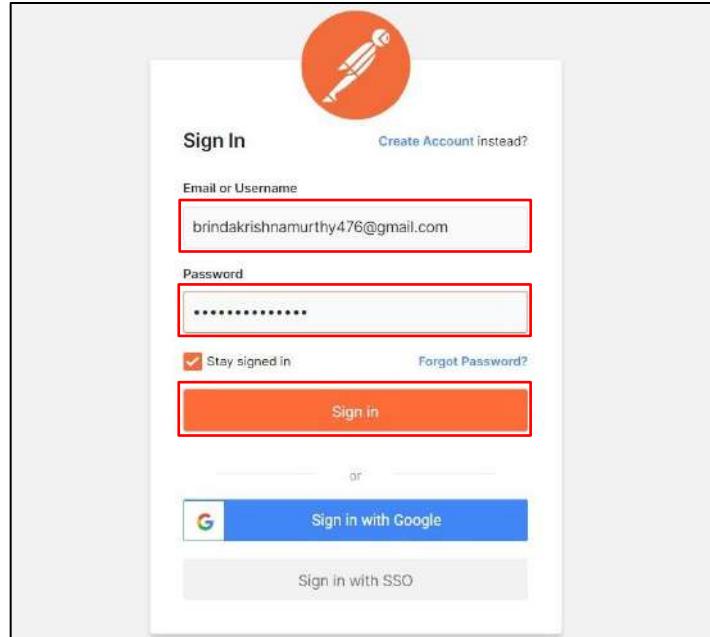
Step 1: Go to Web Browser □ search for ‘Postman’ □ select the second option here.



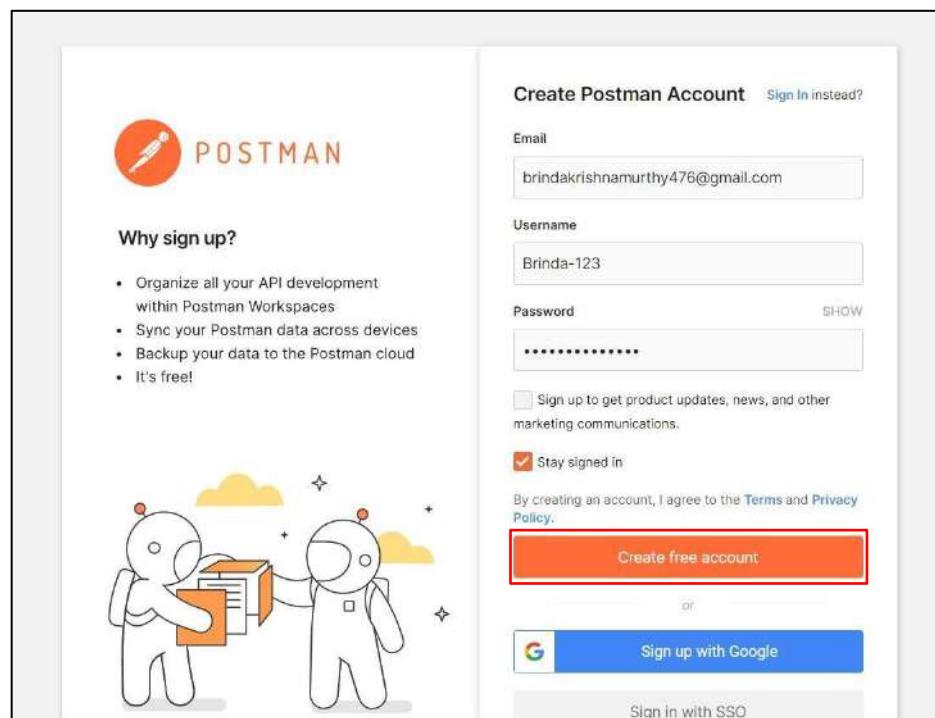
Step 2: Now click on ‘Windows 64-bit’ option to download for windows OS.

A screenshot of the Postman website's download page. The top navigation bar includes links for Product, Pricing, Enterprise, Resources and Support, and Public API Network. On the right side, there are buttons for Contact Sales, Sign In, and Sign Up for Free. The main content area features a large heading "Download Postman" and a sub-headline: "Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman." Below this, there's a section titled "The Postman app" with a sub-instruction: "Download the app to get started with the Postman API Platform." A prominent orange button labeled "Windows 64-bit" is highlighted with a red box. To the right, there's a screenshot of the Postman application interface showing a collection of API endpoints and a request editor.

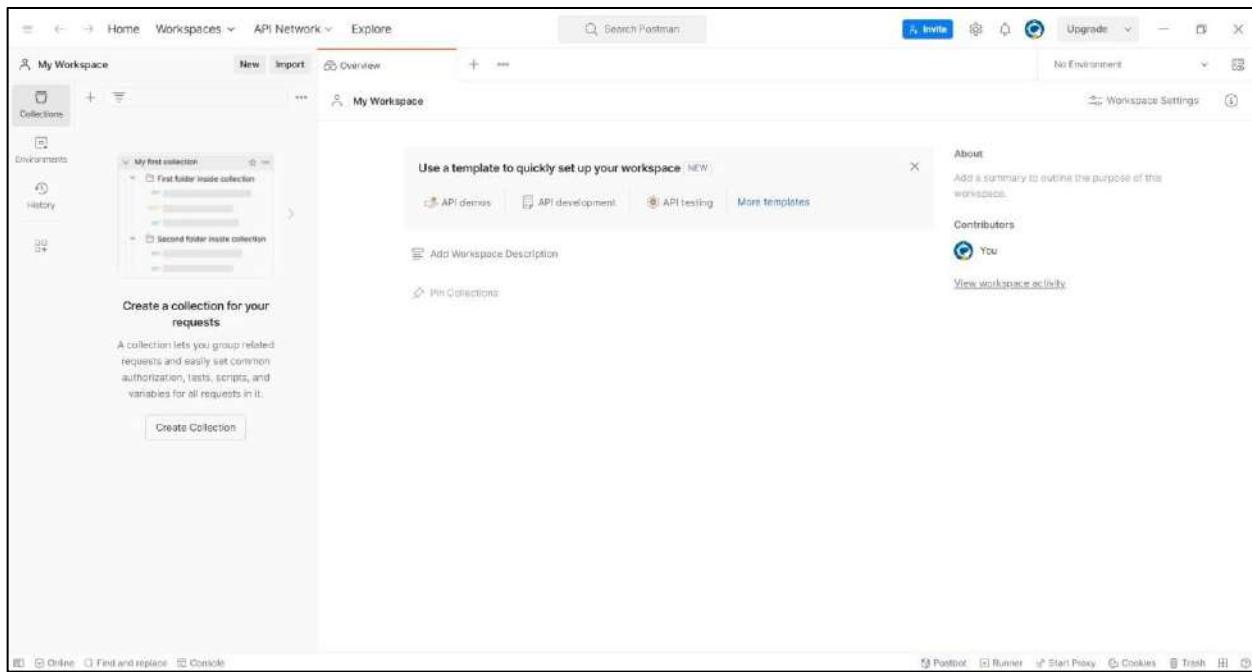
Step 3: You can simply sign to your account by giving your Email-id and Password here.



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**

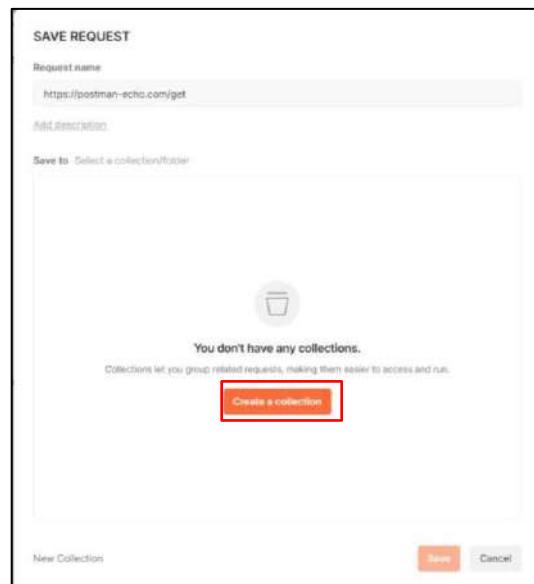
HTTP Methods:

- **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.

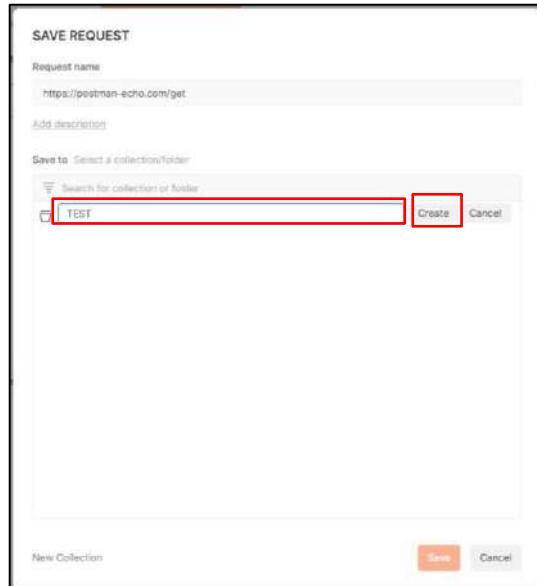
Step 1: Now click on ‘Create Collection’ here.

The screenshot shows the Postman application interface. In the top left, there's a navigation bar with 'Home', 'Workspaces', 'API Network', and 'Explore'. A search bar says 'Search Postman' with a magnifying glass icon. On the right, there are buttons for 'Invite', 'Upgrade', and workspace settings. Below the navigation, the 'My Workspace' section is visible, showing 'Collections' and 'Environments'. A 'My Workspace' panel on the right contains a 'Create a collection for your requests' section with a note about collections and a 'Create Collection' button, which is highlighted with a red box. A modal window titled 'Use a template to quickly set up your workspace' offers options like 'API demos', 'API development', 'API testing', and 'More templates'. At the bottom, there are links for 'Add Workspace Description', 'Pin Collections', 'About', 'Contributors', 'View workspace activity', and workspace status indicators.

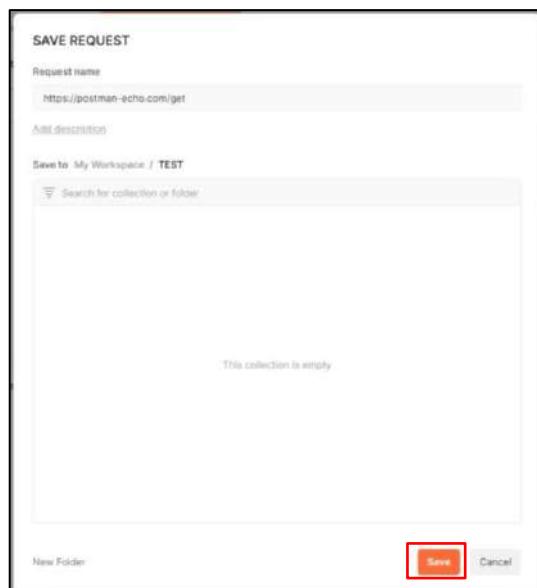
Step 2: Again click on ‘Create a Collection’.



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.

The screenshot shows the Postman interface with a 'GET' request selected. The URL field contains 'https://postman-echo.com/get'. A red box highlights the URL field. The 'Send' button is also highlighted with a red box.

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

The screenshot shows the Postman interface with a 'GET' request selected. The URL field contains 'https://postman-echo.com/get'. A red box highlights the 'Tests' tab. Another red box highlights the test script code in the 'Tests' tab:

```
pm.test("This is Test API", () =>{  
    pm.expect(pm.response.code).to.eql(200)  
});
```

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

The screenshot shows the Postman application interface. In the top navigation bar, there are links for Home, Workspaces, API Network, and Explore. A search bar is located at the top right. Below the navigation, the 'My Workspace' section is visible, showing a collection named 'TEST'. A specific test case is selected, titled 'GET / https://postman-echo.com/get'. The request details show a GET method and the URL https://postman-echo.com/get. The 'Body' tab is selected, displaying the raw JSON response. The response body is a large object containing various headers and parameters. At the bottom of the response pane, it says 'Status: 200 OK' with a red box highlighting it. Other tabs like Cookies, Headers, and Test Results are also present. The bottom of the screen shows the Postman footer with links for Online, Find and replace, and Console.

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

This screenshot shows a context menu for a collection named 'TEST' in the Postman workspace. The menu options include Share, Move, Run collection (which is highlighted with a red box), Generate tests (BETA), Edit, Add request, Add folder, Monitor collection, Mock collection, Create a fork (Ctrl+Alt+F), Create pull request, Merge changes, Pull changes, View changelog, View documentation, Rename (Ctrl+E), Duplicate (Ctrl+D), Export, and Manage roles. The bottom of the screen shows the Postman footer with links for Online, Find and replace, and Console.

Step 10: Then click on ‘Run Test’ here.

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a 'Collections' section with a 'TEST' item expanded. Under 'TEST', there is a single API endpoint: 'GET https://postman-echo.com/get'. On the right side, the 'Run' configuration panel is open. It includes sections for 'Iterations' (set to 1), 'Delay' (set to 0 ms), and 'Data' (with a 'Select File' button). There are also options for 'Persist responses for a session' and 'Advanced settings'. A large red box highlights the 'Run TEST!' button at the bottom of the panel.

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

The screenshot shows the 'TEST - Run results' page in Postman. At the top, it says 'TEST - Run results' and 'Ran today at 21:04:34 - View all runs'. Below this is a summary table with columns: Source (Runner), Environment (none), Iterations (1), Duration (1s 412ms), All tests (1), and Avg. Resp. Time (666 ms). Under 'All Tests: Passed (1) Failed (0) Skipped (0)', it shows an 'Iteration 1' log for a 'GET https://postman-echo.com/get' request. The log entry shows '200 OK' status with a response time of 666 ms. A green bar indicates the test was a 'PASS'. A red box highlights the 'View Summary' link at the top right of the results table.

Step 12: Now select ‘+’ option & select ‘POST’ method here.

The screenshot shows the 'Untitled Request' screen in Postman. At the top, it says 'Untitled Request'. Below that is a search bar with 'GET https://postman-echo.com/get' and a 'Send' button. To the left of the search bar is a dropdown menu showing 'GET' (highlighted in blue) and 'POST' (highlighted with a red box). Below the dropdown are other method options: PUT, PATCH, DELETE, HEAD, and OPTIONS. A note at the bottom says 'Type a new method'.

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.

```
pm.test("This is create Test API", () =>{  
    pm.expect(pm.response.code).to.eql(200)  
});
```

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.

```
pm.test("This is update Test API", () =>{  
    pm.expect(pm.response.code).to.eql(200)  
});
```

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.

The screenshot shows the Postman interface with a red box highlighting the 'TEST' tab. Inside the 'TEST' tab, there is a code editor with the following JavaScript test script:

```

pm.test('This is delete Test API', () => {
  pm.expect(pm.response.code).to.eq(200)
})

```

The status bar at the bottom of the Postman window displays "Status: 200 OK".

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

The screenshot shows the Postman interface with a red box highlighting the 'TEST - Run results' tab. The results table shows the following data:

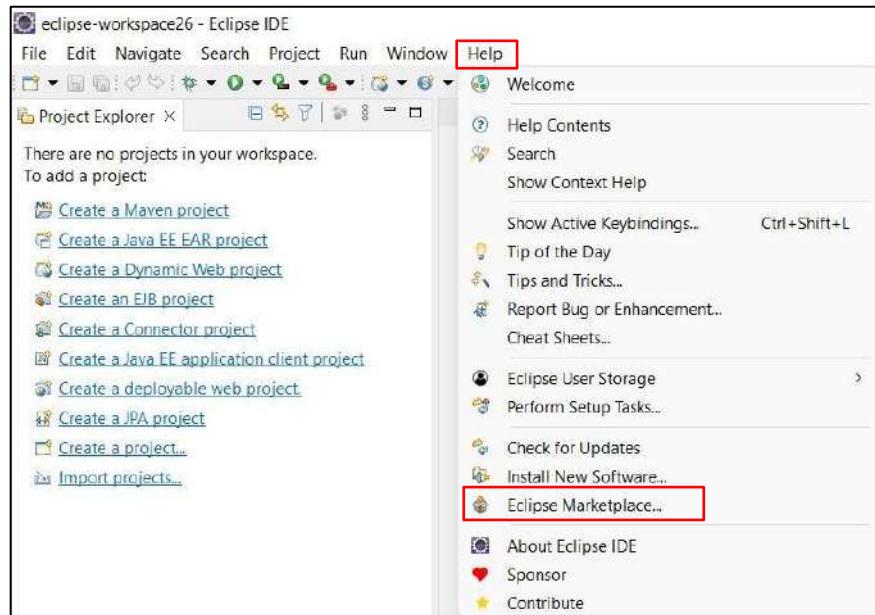
Source	Environment	Iterations	Duration	All Tests	Avg. Resp. Time
Runner	none	1	3s 490ms	4	774 ms

Below the table, it says "All Tests: Passed (4) Failed (0) Skipped (0)". The results for each iteration are listed:

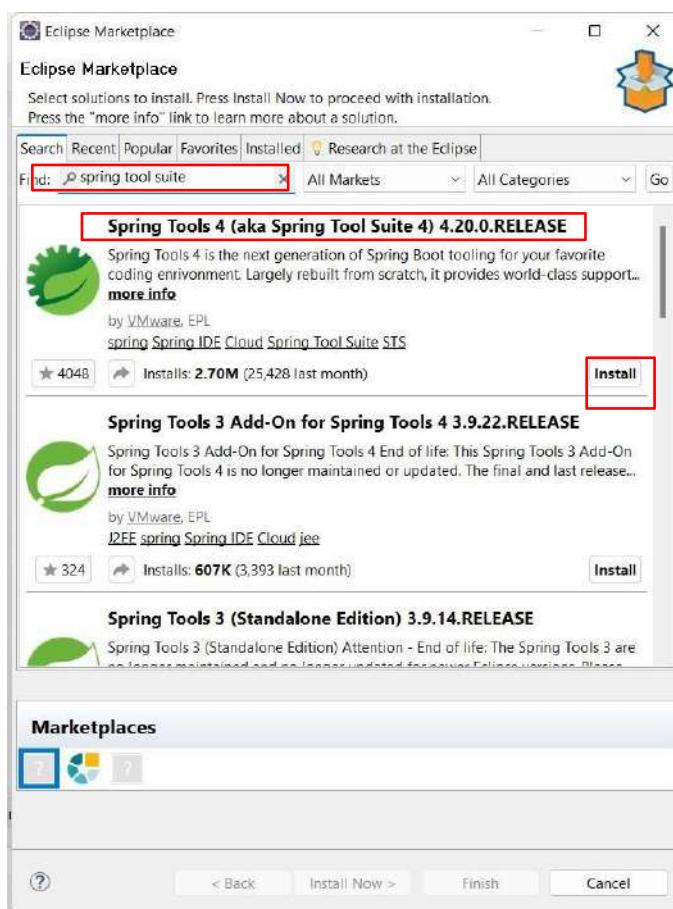
- Iteration 1
 - GET https://postman-echo.com/get
https://postman-echo.com/get
PASS This is Test API
 - POST https://postman-echo.com/post
https://postman-echo.com/post
PASS This is create Test API
 - PUT https://postman-echo.com/put
https://postman-echo.com/put
PASS This is update Test API
 - DELETE https://postman-echo.com/delete
https://postman-echo.com/delete
PASS This is delete Test API

2. Create REST controller for CRUD operations Versioning Spring REST APIs

Step 1: Open ‘Eclipse IDE for Enterprise edition’ & go to → Help → Eclipse Marketplace as shown below.

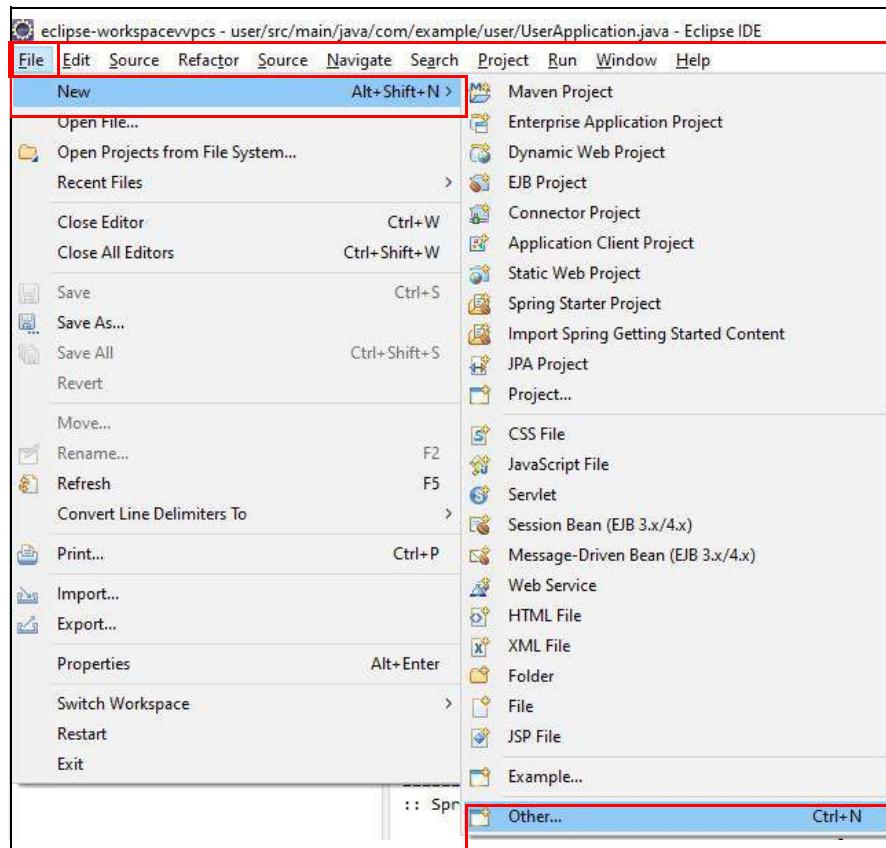


Step 2: Now search for ‘Spring tool suite’ & click on ‘Install’ to install the latest version.

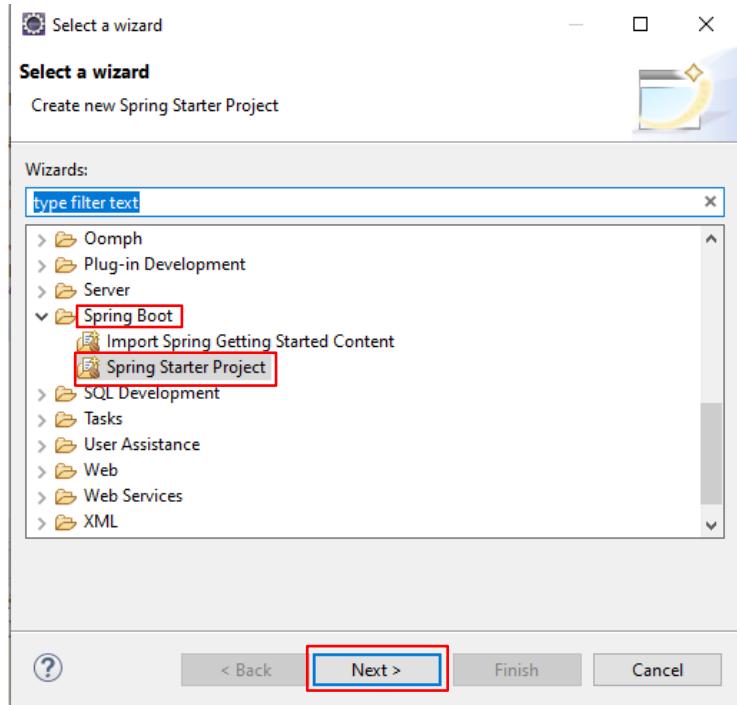


Configure using SpringBoot application property file

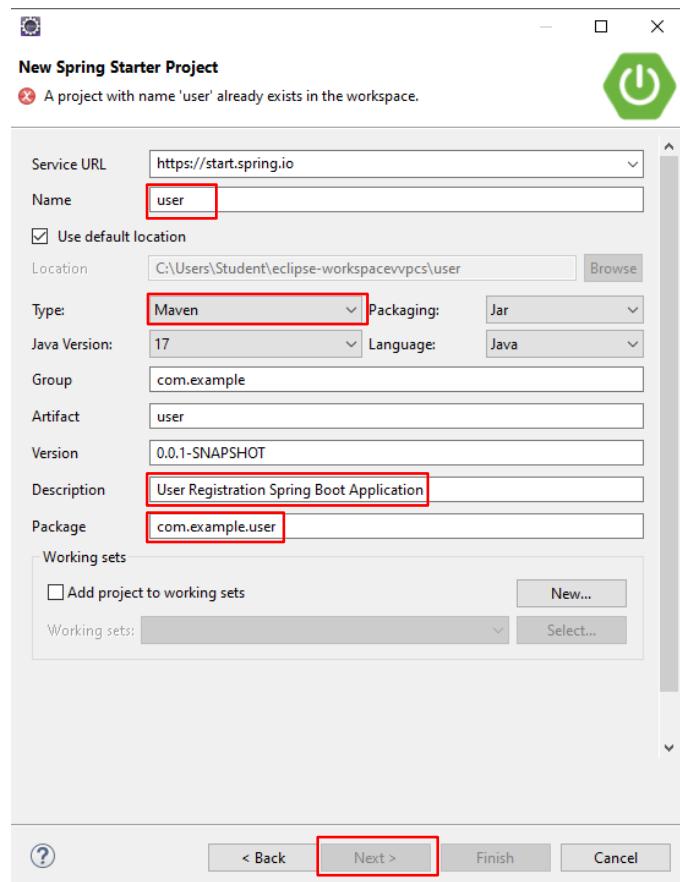
Step 3: After the installation is complete, go to → File → New → Other option as shown below.



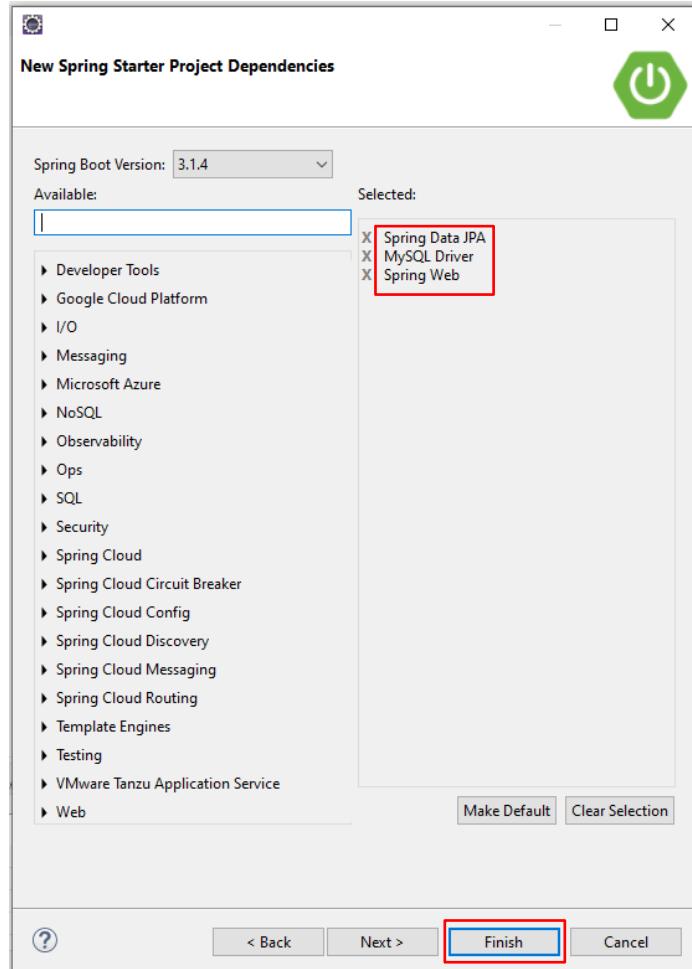
Step 4: Now select ‘Spring Starter Project’ & click on ‘Next’ option.



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’.



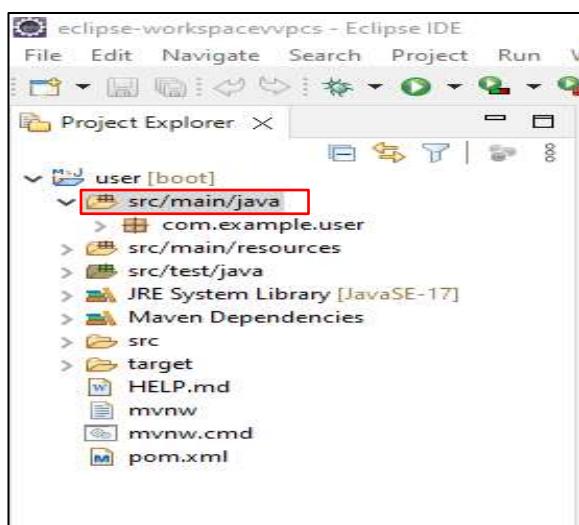
Step 6: Now add all the required dependencies as shown below & click on ‘Finish’.



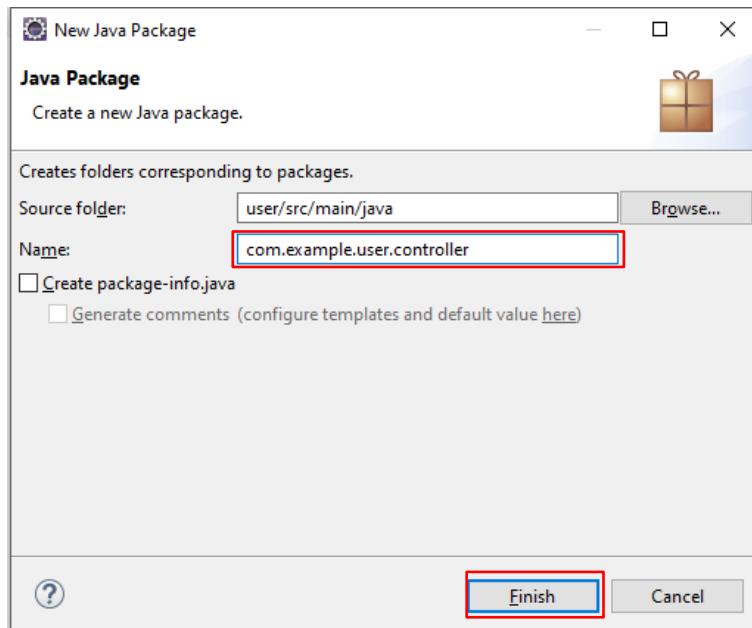
Spring REST – creating Spring REST controller

- Controller Layer (handling request and responses)
- Service Layer (Application business logic)
- Repository layer (Communicate with DB)

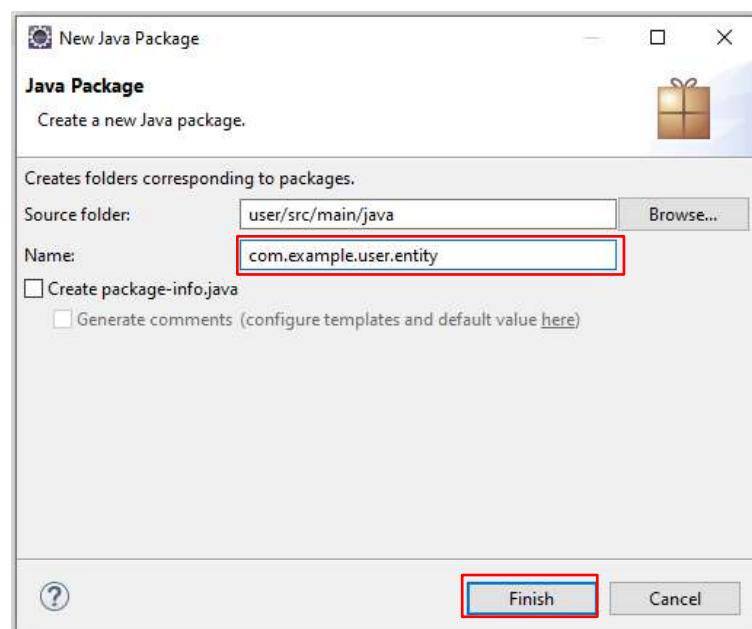
Step 7: Then select the ‘src’ folder New Package.



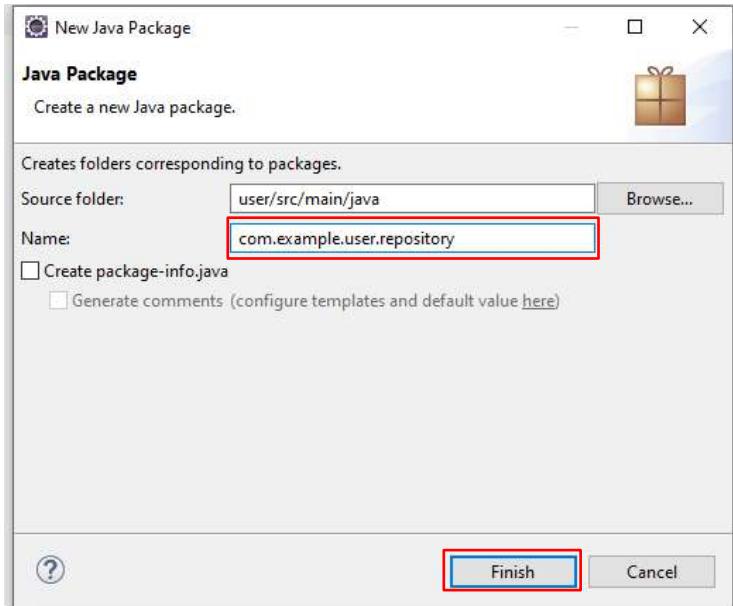
Step 8: Create a new package named ‘com.example.user.controller’ & click on ‘Finish’.



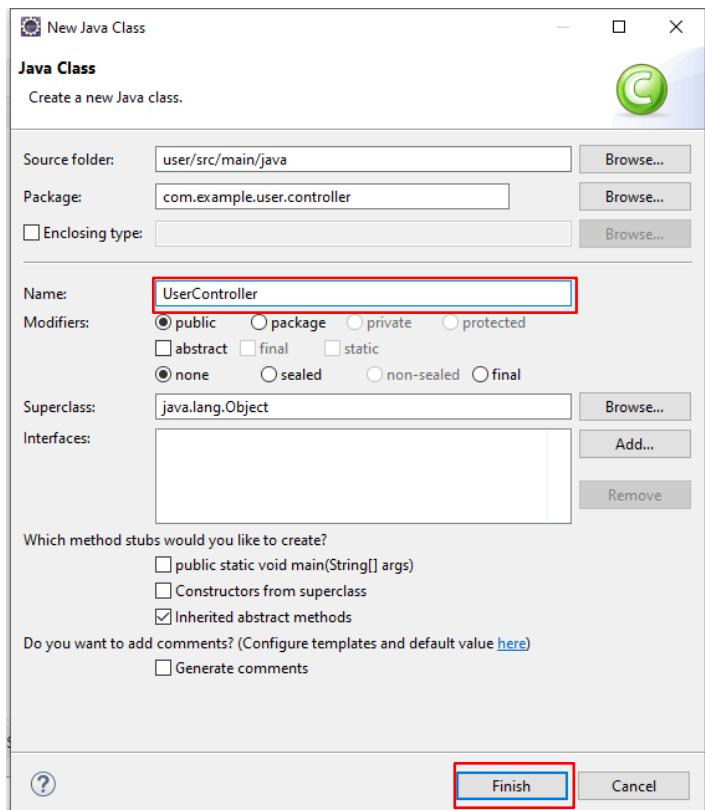
Step 9: Create a new package named ‘com.example.user.entity’ & click on ‘Finish’.



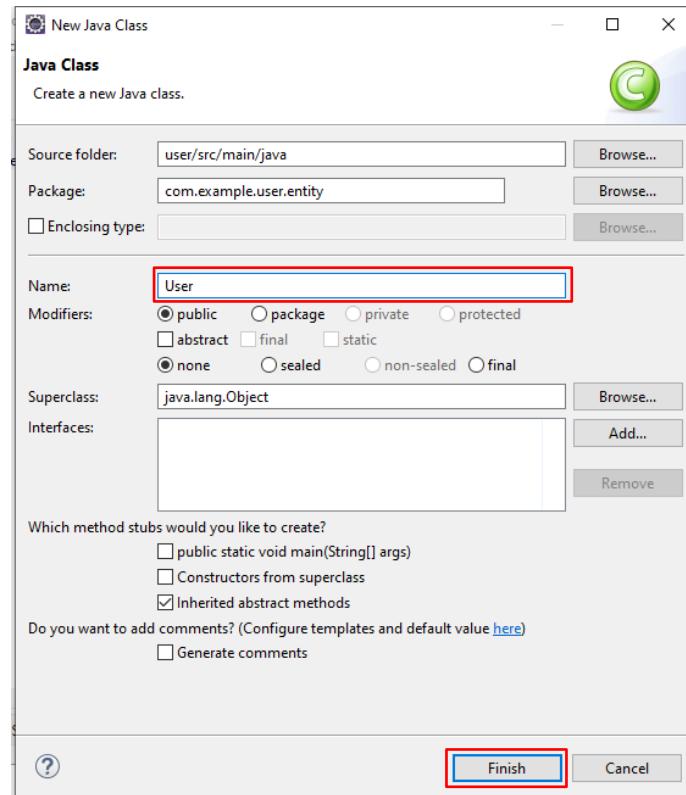
Step 10: Create a new package named ‘com.example.user.repository’ & click on ‘Finish’.



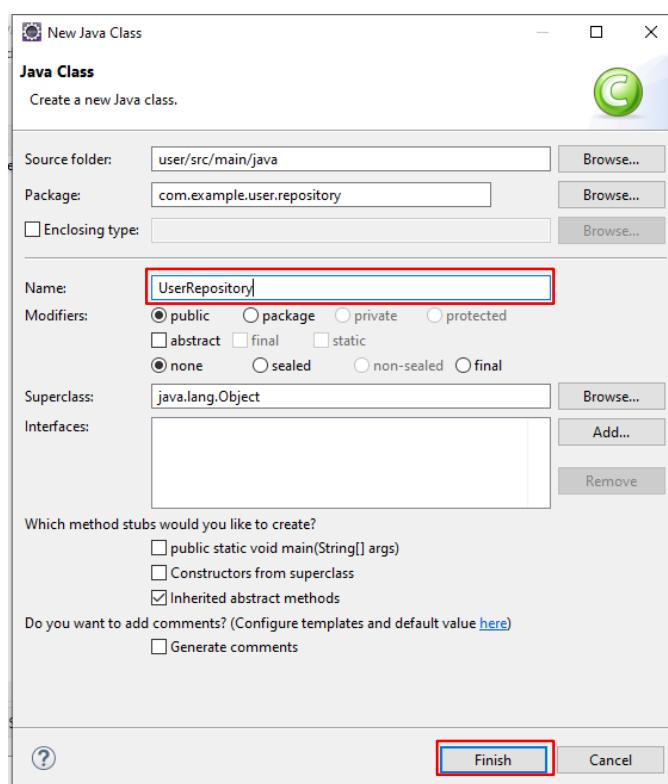
Step 11: Now right click on the ‘com.example.user.controller’ package New Class & create a new class named ‘UserController’ & then click on ‘Finish’.



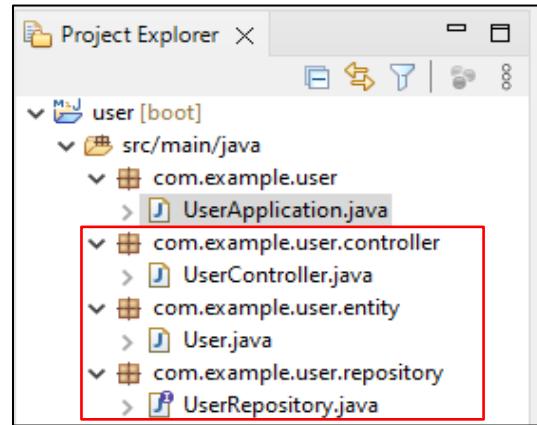
Step 12: Now right click on the ‘com.example.user.entity’ package New Class & create a new class named ‘User’ & then click on ‘Finish’.



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 X User.java UserRepository.java application.properties UserApplication.java
1 package com.example.user.controller;
2 import java.util.List;
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.http.ResponseEntity;
5 import org.springframework.web.bind.annotation.DeleteMapping;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.PathVariable;
8 import org.springframework.web.bind.annotation.PostMapping;
9 import org.springframework.web.bind.annotation.PutMapping;
10 import org.springframework.web.bind.annotation.RequestBody;
11 import org.springframework.web.bind.annotation.RequestMapping;
12 import org.springframework.web.bind.annotation.RestController;
13
14 import com.example.user.entity.User;
15 import com.example.user.repository.UserRepository;
16
17 @RestController
18 @RequestMapping("/users")
19 public class UserController {
20     @Autowired
21     private UserRepository userRepository;
22
23     @GetMapping
24     public List<User> getAllUser()
25     {
26         return this.userRepository.findAll();
27     }
28
29     @GetMapping("/{id}")
30     public User getUserById(@PathVariable(value="id") long userId) {
31         return this.userRepository.findById(userId).orElseThrow();
32     }
33

```

```

36     @PostMapping
37     public User createUser(@RequestBody User user)
38     {
39         return this.userRepository.save(user);
40     }
41     @PutMapping("/{id}")
42     public User updateUser(@RequestBody User user,@PathVariable("id") long userId)
43     {
44         User ex=this.userRepository.findById(userId).orElseThrow();
45         ex.setFirstname(user.getFirstname());
46         ex.setLastname(user.getLastname());
47         return this.userRepository.save(ex);
48     }
49     @DeleteMapping("/{id}")
50     public ResponseEntity<User> deleteUser(@PathVariable("id") long userId)
51     {
52         User ex=this.userRepository.findById(userId).orElseThrow();
53         this.userRepository.delete(ex);
54         return ResponseEntity.ok().build();
55     }
56 }
57

```

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

```

1 package com.example.user.entity;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.GeneratedValue;
5 import jakarta.persistence.GenerationType;
6 import jakarta.persistence.Id;
7 import jakarta.persistence.Table;
8
9 @Entity
10 @Table(name="user")
11 public class User {
12     public User () {
13     }
14     @Id
15     @GeneratedValue(strategy=GenerationType.AUTO)
16     private Long id;
17     private String firstname;
18     private String lastname;
19     public Long getId() {
20         return id;
21     }
22     public void setId(Long id) {
23         this.id = id;
24     }
25     public String getFirstname() {
26         return firstname;
27     }
28     public void setFirstname(String firstname) {
29         this.firstname = firstname;
30     }
31     public User(Long id, String firstname, String lastname) {
32         super();
33         this.id = id;
34         this.firstname = firstname;
35         this.lastname = lastname;
36     }

```

```

37     public String getLastname() {
38         return lastname;
39     }
40     public void setLastname(String lastname) {
41         this.lastname = lastname;
42     }
43 }
44

```

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

```

1 package com.example.user.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.example.user.entity.User;
7
8 @Repository
9 public interface UserRepository extends JpaRepository<User,Long>
10 {
11 }
12

```

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



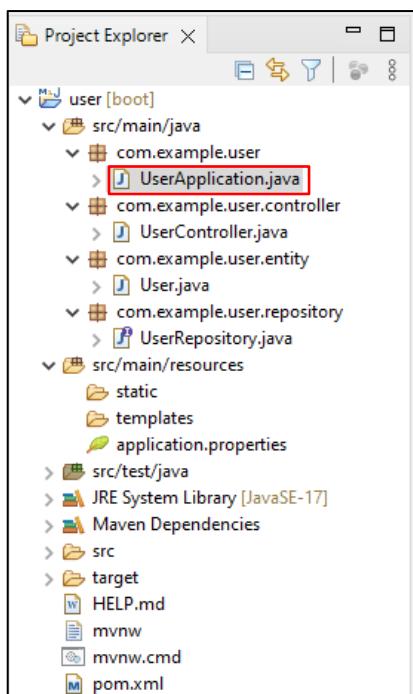
Step 19: Now add code to **application.properties** file as shown below.

```
spring.datasource.url=jdbc:mysql://localhost:3307/data
spring.datasource.username=root
spring.datasource.password>Vvp_dlt
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update
```

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

```
package com.example.user;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class UserApplication {
    public static void main(String[] args) {
        SpringApplication.run(UserApplication.class, args);
        System.out.println("Full Stack Development");
    }
}
```

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.

```

\\ / .
(( )) \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_
\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ 
:: Spring Boot : (v3.1.4)

2023-10-13T14:47:37.247+05:30 INFO 15408 --- [main] com.example.user.UserApplication      : Starting UserApplication using Java 17.0.8 wit
2023-10-13T14:47:37.259+05:30 INFO 15408 --- [main] com.example.user.UserApplication      : No active profile set, falling back to 1 defau
2023-10-13T14:47:40.637+05:30 INFO 15408 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in
2023-10-13T14:47:40.637+05:30 INFO 15408 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 15
2023-10-13T14:47:43.274+05:30 INFO 15408 --- [main] o.s.d.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-10-13T14:47:43.311+05:30 INFO 15408 --- [main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-10-13T14:47:43.315+05:30 INFO 15408 --- [main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.1
2023-10-13T14:47:43.675+05:30 INFO 15408 --- [main] o.a.c.c.C.[Tomcat].[localhost].[]     : Initializing Spring embedded WebApplicationContext
2023-10-13T14:47:43.682+05:30 INFO 15408 --- [main] w.s.c.ServletWebServerApplicationContext  : Root WebApplicationContext: initialization com
2023-10-13T14:47:44.141+05:30 INFO 15408 --- [main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [nam
2023-10-13T14:47:44.308+05:30 INFO 15408 --- [main] org.hibernate.Version                  : HHH000412: Hibernate ORM core version 6.2.9.Fi
2023-10-13T14:47:44.314+05:30 INFO 15408 --- [main] org.hibernate.cfg.Environment       : HHH000406: Using bytecode reflection optimizer
2023-10-13T14:47:44.682+05:30 INFO 15408 --- [main] o.h.b.i.BytecodeProviderInitiator    : HHH000221: Bytecode provider name : bytебuddy
2023-10-13T14:47:45.026+05:30 INFO 15408 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo  : HHH000182: No default (no-argument) constructo
2023-10-13T14:47:45.675+05:30 INFO 15408 --- [main] com.zaxxer.hikari.HikariDataSource   : HHH000021: Bytecode provider name : bytебuddy
2023-10-13T14:47:45.949+05:30 INFO 15408 --- [main] com.zaxxer.hikari.HikariPool          : HHH000021: Bytecode provider name : bytебuddy
2023-10-13T14:47:45.954+05:30 INFO 15408 --- [main] com.zaxxer.hikari.HikariDataSource   : HHH000221: Bytecode provider name : bytебuddy
2023-10-13T14:47:47.612+05:30 INFO 15408 --- [main] o.h.b.i.BytecodeProviderInitiator    : HHH000021: Bytecode provider name : bytебuddy
2023-10-13T14:47:48.123+05:30 INFO 15408 --- [main] o.h.m.i.EntityInstantiatorPojoStandard: HHH000182: No default (no-argument) constructo
2023-10-13T14:47:48.512+05:30 INFO 15408 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator  : HHH0000490: Using JtaPlatform implementation: [
2023-10-13T14:47:48.790+05:30 INFO 15408 --- [main] j.LocalContainerEntityManagerFactoryBean: HHH000182: No default (no-argument) constructo
2023-10-13T14:47:49.467+05:30 WARN 15408 --- [main] JpaBaseConfiguration$JpaWebConfiguration: HHH000182: No default (no-argument) constructo
2023-10-13T14:47:50.327+05:30 INFO 15408 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer: HHH000204: Processing PersistenceUnitInfo [nam
2023-10-13T14:47:50.363+05:30 INFO 15408 --- [main] com.example.user.UserApplication      : Tomcat started on port(s): 8080 (http) with cc
Full Stack Development

```

- 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.

```

MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \q.
Your MySQL connection id is 141
Server version: 8.0.34 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| vvp |
+-----+
5 rows in set (7.18 sec)

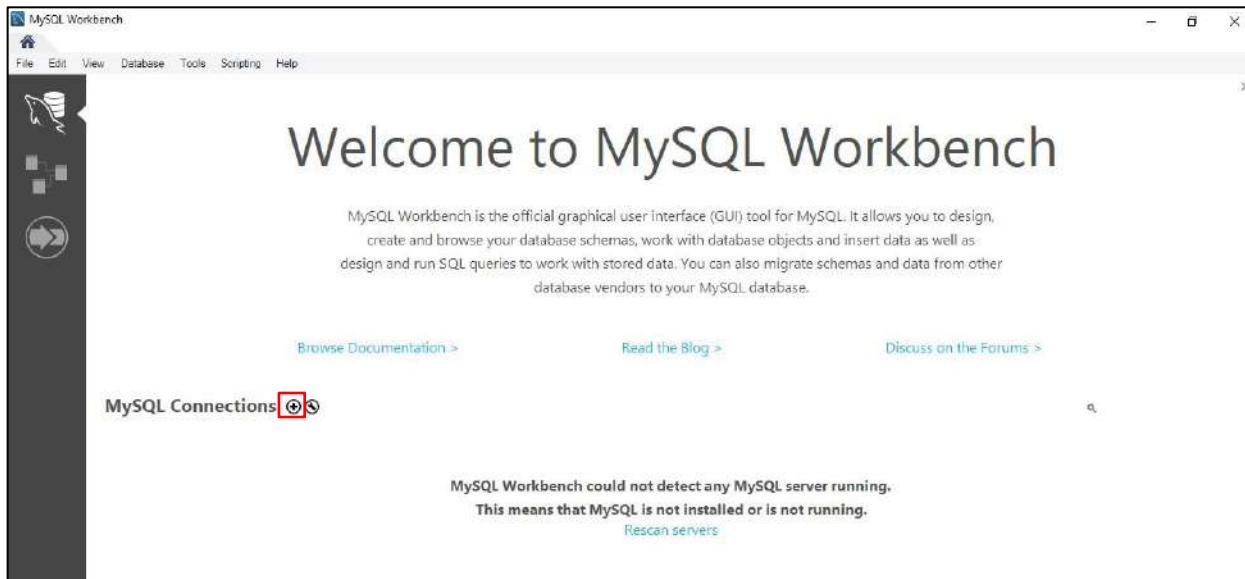
mysql> create database data2;
Query OK, 1 row affected (0.09 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| data2 |
| information_schema |
| mysql |
| performance_schema |
| sys |
| vvp |
+-----+
6 rows in set (0.00 sec)

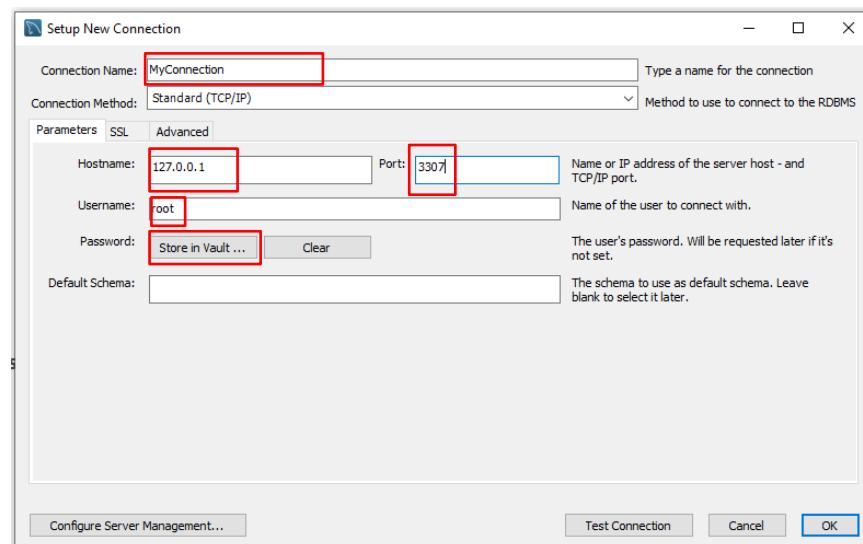
```

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

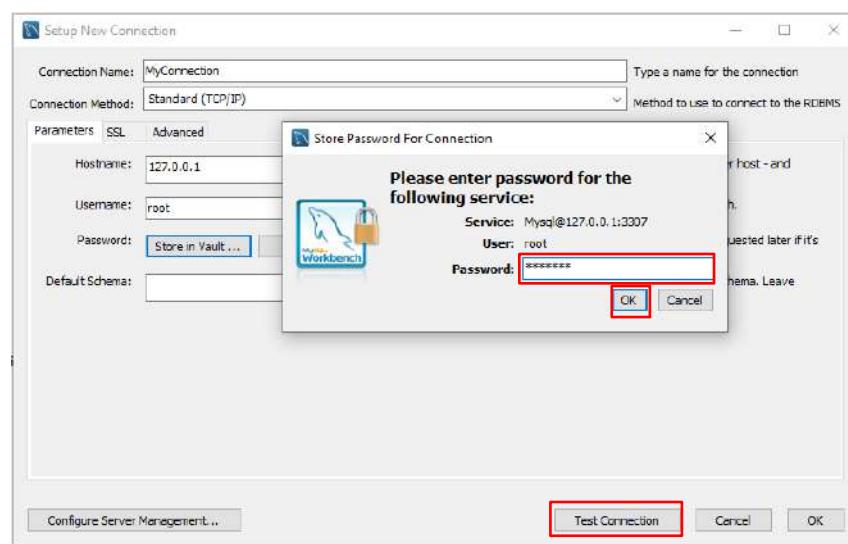
Step 24: Go to ‘+’ option to add new connection.



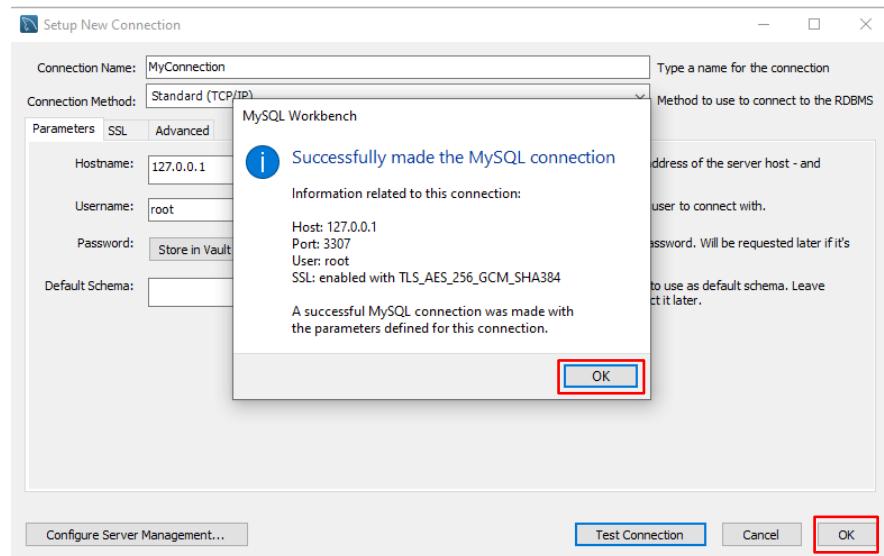
Step 25: Give a name to your connection as ‘MyConnection’ □ select the Hostname as ‘localhost’ or select the default ip address of your local computer □ give the port no. which you’ve entered while installation of MySQL Workbench (3307) □ Username as ‘root’.



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



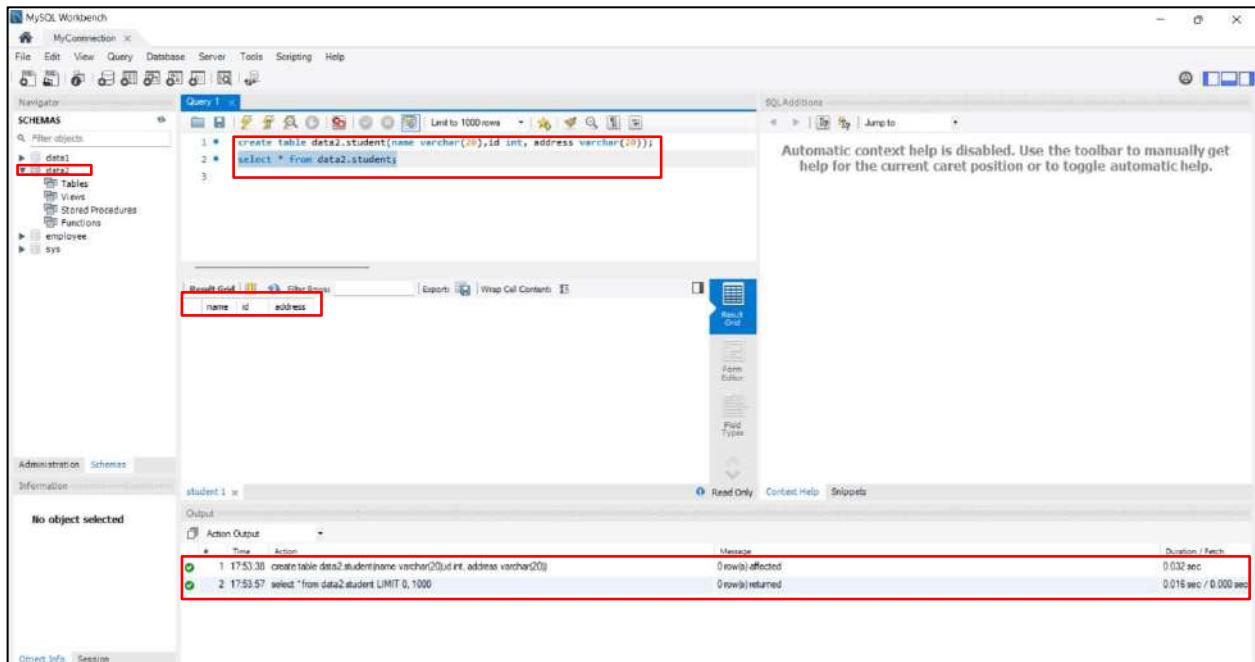
Step 27: Now click on ‘OK’ option as shown below.



- 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.



- **Performing CRUD operations using Postman RESTful Service.**

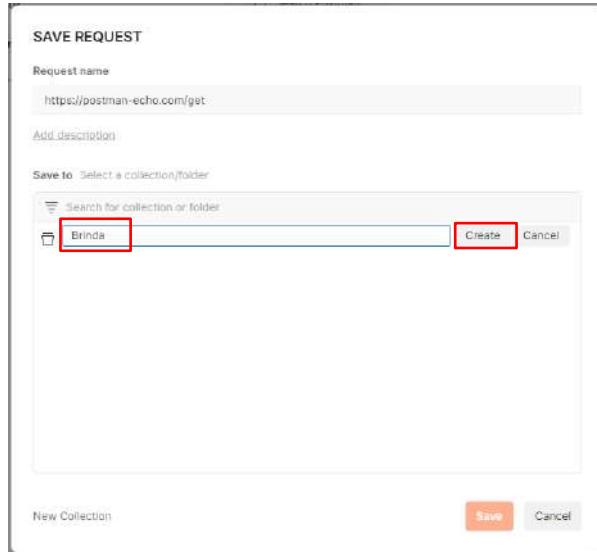
Step 29: Login to the Postman & click on ‘Create collection’ here.

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing 'Collections' (with 'My first collection' expanded to show 'First folder inside collection' and 'Second folder inside collection'), 'Environments', and 'History'. Below this, a callout box says 'Create a collection for your requests' with a note: 'A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.' A red box highlights the 'Create Collection' button at the bottom of this section. The main workspace shows an 'Untitled Request' with a 'GET' method and an empty URL field. Below the request area is a 'Response' section featuring a cartoon character holding a rocket. At the bottom of the screen, there are various status icons and a toolbar.

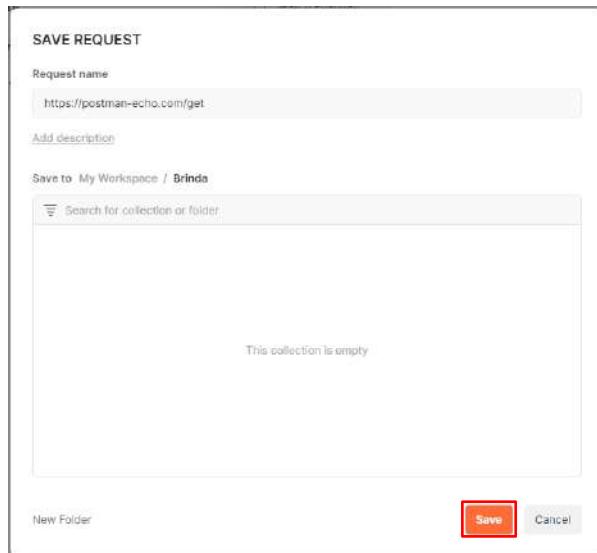
Step 30: Again click on ‘Create a collection’ here.

This screenshot shows the 'SAVE REQUEST' dialog box. It has fields for 'Request name' (set to 'https://postman-echo.com/get') and 'Add description'. Under 'Save to', it says 'Select a collection/folder' and shows a list with a trash icon. A message below states 'You don't have any collections.' A red box highlights the 'Create a collection' button at the bottom of the dialog. At the very bottom are 'New Collection', 'Save', and 'Cancel' buttons.

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.

The screenshot shows the Postman interface with a successful API call. The URL is set to `localhost:8080/users`. The response body is displayed in JSON format:

```
1: {  
2:   "timestamp": "2023-10-14T11:29:07.148+00:00",  
3:   "status": 500,  
4:   "error": "Internal Server Error",  
5:   "path": "/users"  
6: }
```

Buttons highlighted in red are 'Save' and 'Send'. The status bar at the bottom indicates 'Status: 500 Internal Server Error'.

Step 34: Select ‘POST’ method.

The screenshot shows the Postman interface with the 'POST' method selected from a dropdown menu. The URL is set to `localhost:8080/users`. The status bar at the bottom indicates 'Enter the URL, and click Send to get a response.'

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.

The screenshot shows the Postman interface. A POST request is made to `localhost:8080/users`. The request body is set to `JSON` and contains the following data:

```

1: {
2:   "id": 1,
3:   "firstname": "FSD",
4:   "lastname": "CS"
5: }

```

The response status is `200 OK`.

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.

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```

1: use details;
2: select * from user;

```

The results grid shows the following data:

	<code>id</code>	<code>firstname</code>	<code>lastname</code>
1	1	FSD	CS
2	2	APF	PostMan

The status bar at the bottom right indicates the duration of the operation.

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.

The screenshot shows the Postman application interface. The top navigation bar includes Home, Workspaces, API Network, Explore, and a search bar labeled "Search Postman". On the right, there are buttons for Invite, Help, Upgrade, and a user profile icon.

The left sidebar displays "My Workspace" with sections for Collections, Environments, and History. Under "Collections", there is a "Brinda" collection containing three items: "GET localhost:8080/users", "POST localhost:8080/users", and "PUT localhost:8080/users/{id}".

The main workspace shows a "PUT /localhost:8080/users/2" request. The "Body" tab is selected, showing a JSON payload:

```
1 {  
2   "id": 2,  
3   "firstname": "API",  
4   "lastname": "PostMan"  
5 }
```

Below the body, the response status is "Status: 200 OK", "Time: 370 ms.", and "Size: 211 B". There are also "Save as example" and "Copy" buttons.

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.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with tables like student, user, and user_log. The central area has a query editor with the following content:

```
use data1;
select * from user;
```

The results grid shows the following data:

ID	firstname	lastname
1	APT	Postman

The bottom pane shows the execution history for the 'user' table:

Action	Time	Action Output	Message	Duration / Fetch
use data1	10:17:15.32		0 rows affected	0.000 sec
select * from user LIMIT 0, 1000	11:17:15.37		1 rows returned	0.000 sec / 0.000 sec
select * from user LIMIT 0, 1000	12:17:18.25		2 rows returned	0.000 sec / 0.000 sec
select * from user LIMIT 0, 1000	13:17:20.52		2 rows returned	0.000 sec / 0.000 sec
select * from user LIMIT 0, 1000	14:17:21.48		2 rows returned	0.000 sec / 0.000 sec
select * from user LIMIT 0, 1000	15:17:22.31		2 rows returned	0.000 sec / 0.000 sec

Step 39: Select ‘DELETE’ method & then enter the command with the id no. specified as

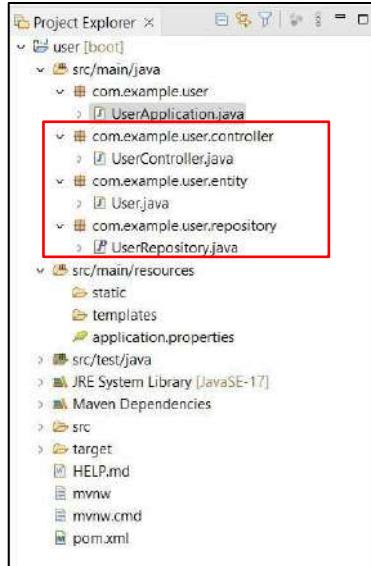
'localhost:8080/users/1' → select **Body** → **Text type - JSON** → enter the below code as shown below → Save it & Send.

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 X User.java UserRepository.java application.properties UserApplication.java
1 package com.example.user.controller;
2 import java.util.List;
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.http.ResponseEntity;
5 import org.springframework.web.bind.annotation.DeleteMapping;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.PathVariable;
8 import org.springframework.web.bind.annotation.PostMapping;
9 import org.springframework.web.bind.annotation.PutMapping;
10 import org.springframework.web.bind.annotation.RequestBody;
11 import org.springframework.web.bind.annotation.RequestMapping;
12 import org.springframework.web.bind.annotation.RestController;
13
14 import com.example.user.entity.User;
15 import com.example.user.repository.UserRepository;
16
17 @RestController
18 @RequestMapping("/users")
19 public class UserController {
20     @Autowired
21     private UserRepository userRepository;
22
23     @GetMapping
24     public List<User> getAllUser() {
25         return this.userRepository.findAll();
26     }
27
28     @GetMapping("/{id}")
29     public User getUserById(@PathVariable(value="id") long userId) {
30         return this.userRepository.findById(userId).orElseThrow();
31     }
32
33     @PostMapping
34     public User createUser(@RequestBody User user) {
35         return this.userRepository.save(user);
36     }
37
38 }

39     @PutMapping("/{id}")
40     public User updateUser(@RequestBody User user, @PathVariable("id") long userId)
41     {
42         User ex=this.userRepository.findById(userId).orElseThrow();
43         ex.setFirstname(user.getFirstname());
44         ex.setLastname(user.getLastname());
45         ex.setRegisternumber(user.getRegisternumber());
46         ex.setDOB(user.getDOB());
47         ex.setAddress(user.getAddress());
48         ex.setDepartment(user.getDepartment());
49         return this.userRepository.save(ex);
50     }
51     @DeleteMapping("/{id}")
52     public ResponseEntity<User> deleteUser(@PathVariable("id") long userId)
53     {
54         User ex=this.userRepository.findById(userId).orElseThrow();
55         this.userRepository.delete(ex);
56         return ResponseEntity.ok().build();
57     }
58 }

```

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

```

1 package com.example.user.entity;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.GeneratedValue;
5 import jakarta.persistence.GenerationType;
6 import jakarta.persistence.Id;
7 import jakarta.persistence.Table;
8
9 @Entity
10 @Table(name="user")
11 public class User {
12     public User() {
13     }
14     @Id
15     @GeneratedValue(strategy=GenerationType.AUTO)
16     private Long id;
17     private String firstname;
18     private String lastname;
19     private String Registernumber;
20     private String DOB;
21     private String Address;
22     private String Department;
23     public Long getId() {
24         return id;
25     }
26     public void setId(Long id) {
27         this.id = id;
28     }
29     public String getFirstname() {
30         return firstname;
31     }
32     public void setFirstname(String firstname) {
33         this.firstname = firstname;
34     }
35     public String getLastname() {
36         return lastname;
37     }
38     public void setLastname(String lastname) {
39         this.lastname = lastname;
40     }

```

```

41     public String getRegisternumber() {
42         return Registernumber;
43     }
44     public void setRegisternumber(String registernumber) {
45         Registernumber = registernumber;
46     }
47     public String getDOB() {
48         return DOB;
49     }
50     public void setDOB(String dob) {
51         DOB = dob;
52     }
53     public String getAddress() {
54         return Address;
55     }
56     public void setAddress(String address) {
57         Address = address;
58     }
59     public String getDepartment() {
60         return Department;
61     }
62     public void setDepartment(String department) {
63         Department = department;
64     }
65     public User(Long id, String firstname, String lastname, String registernumber, String dob, String address,
66                 String department) {
67         super();
68         this.id = id;
69         this.firstname = firstname;
70         this.lastname = lastname;
71         Registernumber = registernumber;
72         DOB = dob;
73         Address = address;
74         Department = department;
75     }
76 }

```

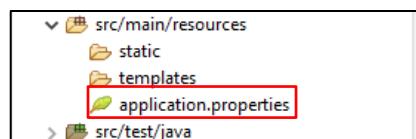
Step 4: Now add code to **UserRepository.java** file as shown below.

```

1 package com.example.user.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.example.user.entity.User;
7
8 @Repository
9 public interface UserRepository extends JpaRepository<User, Long> {
10 }
11

```

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.

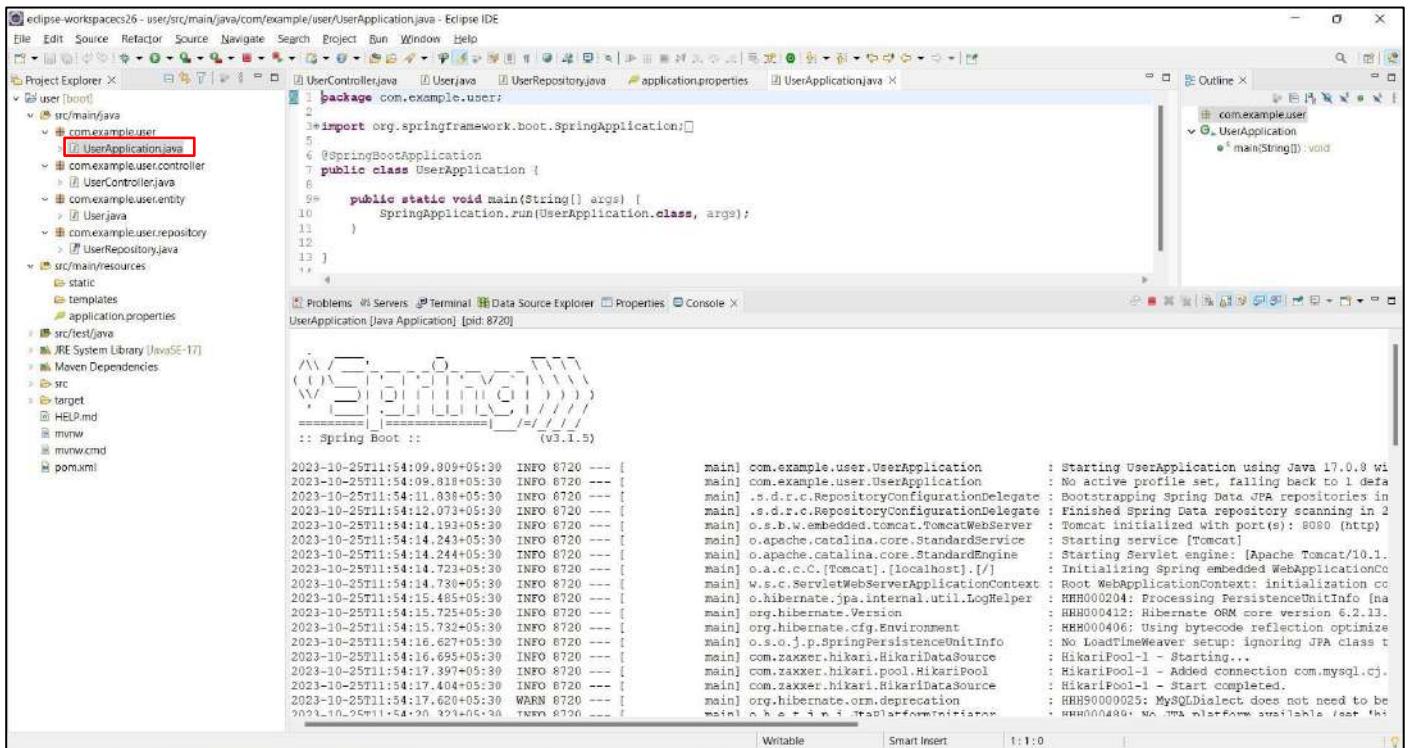
```

1 spring.datasource.url=jdbc:mysql://localhost:3306/register
2 spring.datasource.username=root
3 spring.datasource.password=Vp@123
4 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
5 spring.jpa.hibernate.ddl-auto=update
6 server.port=8080
7

```

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

Now the server is running successfully.



```
package com.example.user;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
public class UserApplication {
    public static void main(String[] args) {
        SpringApplication.run(UserApplication.class, args);
    }
}
```

The screenshot shows the Eclipse IDE interface with the UserApplication.java file open in the editor. The file contains the main method of a Spring Boot application. The 'UserApplication.java' file is highlighted with a red border. The code is as follows:

```
package com.example.user;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
public class UserApplication {
    public static void main(String[] args) {
        SpringApplication.run(UserApplication.class, args);
    }
}
```

The console output window shows the application starting up, including logs from Spring Boot, Tomcat, and Hibernate. The log output is as follows:

```
2023-10-25T11:54:09.809+05:30 INFO 8720 --- [main] com.example.user.UserApplication : Starting UserApplication using Java 17.0.8
2023-10-25T11:54:09.810+05:30 INFO 8720 --- [main] com.example.user.UserApplication : No active profile set, falling back to 'dev'
2023-10-25T11:54:11.036+05:30 INFO 8720 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in 2ms
2023-10-25T11:54:12.073+05:30 INFO 8720 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 2ms
2023-10-25T11:54:14.193+05:30 INFO 8720 --- [main] o.s.d.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-10-25T11:54:14.243+05:30 INFO 8720 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-10-25T11:54:14.244+05:30 INFO 8720 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.17]
2023-10-25T11:54:14.723+05:30 INFO 8720 --- [main] o.a.c.c.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-10-25T11:54:15.730+05:30 INFO 8720 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1 ms
2023-10-25T11:54:15.725+05:30 INFO 8720 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: register]
2023-10-25T11:54:15.732+05:30 INFO 8720 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.2.13.Final
2023-10-25T11:54:16.627+05:30 INFO 8720 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo : HHH000406: Using bytecode setup/ reflection optimize
2023-10-25T11:54:16.695+05:30 INFO 8720 --- [main] com.zaxxer.hikari.HikariDataSource : HHH000204: Processing PersistenceUnitInfo [name: register]
2023-10-25T11:54:17.397+05:30 INFO 8720 --- [main] com.zaxxer.hikari.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@391a0c4a
2023-10-25T11:54:17.404+05:30 INFO 8720 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-10-25T11:54:17.620+05:30 WARN 8720 --- [main] org.hibernate.orm.deprecation : HHH90000025: MySQLDialect does not need to be configured; MySQL platform available (set this)
```

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



```
mysql> create database register;
Query OK, 1 row affected (0.02 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| data2              |
| information_schema |
| mysql              |
| performance_schema |
| register           |
| sys                |
| vips               |
+--------------------+
7 rows in set (0.01 sec)
```

The screenshot shows the MySQL 8.0 Command Line Client interface. A command is being entered to create a new database named 'register'. After executing the command, the 'show databases;' command is run to list all databases. The output shows the newly created 'register' database along with other existing databases like 'data2', 'information_schema', 'mysql', 'performance_schema', 'sys', and 'vips'. The log output is as follows:

```
mysql> create database register;
Query OK, 1 row affected (0.02 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| data2              |
| information_schema |
| mysql              |
| performance_schema |
| register           |
| sys                |
| vips               |
+--------------------+
7 rows in set (0.01 sec)
```

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.

The screenshot shows the Postman application interface. In the center, there's a request configuration for a 'GET' method to 'localhost:8080/users'. The 'Send' button at the top right and the 'status: 200 OK' message in the results section are both highlighted with red boxes.

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.

The screenshot shows the MySQL Workbench interface. In the central query editor window, there are two lines of SQL code: 'use register;' and 'select * from user;'. The first line, 'use register;', is highlighted with a red box. The results grid below shows a table with columns: id, address, dob, department, registernumber, firstname, lastname. The 'register' schema is selected in the left sidebar.

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.

The screenshot shows the Postman interface. In the left sidebar, there's a collection named 'My Workspace' with a 'Register' folder containing several API endpoints: GET localhost:8080/users, POST localhost:8080/users, PUT localhost:8080/users/2, and DELETE localhost:8080/users/2. The main area shows a POST request to 'localhost:8080/users'. The 'Body' tab is selected, showing a JSON payload:

```

1
2   "id": 55,
3   "firstname": "Brinda",
4   "lastname": "K",
5   "registernumber": "488CS21607",
6   "dob": "02-04-2005",
7   "address": "VV Mohalla",
8   "department": "Computer Science"

```

The response status is 200 OK, and the response body is identical to the request body.

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.

The screenshot shows the MySQL Workbench interface. The 'Navigator' pane shows the 'register' schema with tables like user, user_seq, Views, and Stored Procedures. The 'Query1' pane contains the following SQL code:

```

1 * use register;
2 * select * from user;

```

The 'Result Grid' shows the following data:

	id	address	dob	department	registernumber	firstname	lastname
*	55	VV Mohalla	02-04-2005	Computer Science	488CS21607	Brinda	K

The 'Session' pane at the bottom shows the execution history with 16 rows returned.

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.

The screenshot shows the Postman interface. In the top navigation bar, 'localhost:8080/users' is selected. The main area shows a POST request to 'localhost:8080/users'. The 'Body' tab is selected, showing a JSON payload:

```

1   {
2     "id": 56,
3     "firstname": "Bhoomika",
4     "lastname": "R",
5     "address": "Vinayaknagar",
6     "registernumber": "488CS21006",
7     "dob": "06-02-2005",
8     "department": "Computer Science"
9   }

```

The response status is 200 OK. Buttons for 'Save' and 'Send' are highlighted with red boxes.

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.

The screenshot shows the MySQL Workbench interface. The connection is named 'MyConnection1'. The 'register' schema is selected. A query is run:

```

1 * use register;
2 * select * from user;
3
4
5

```

The results grid shows two rows of data:

	id	address	dob	department	registernumber	firstname	lastname
55	VV Mokkala	02-04-2005	Computer Science	488CS21007	Bridha	K	
56	Vineyaknagar	06-02-2005	Computer Science	488CS21006	Bhoomika	R	

The bottom section shows the 'Output' pane with a table of execution history:

Action	Time	Action	Message	Duration / Fetch
1	9 12:28:38	select * from user LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
2	10 12:30:21	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
3	11 12:31:44	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
4	12 12:31:59	select * from user LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
5	13 12:37:45	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
6	14 12:38:19	select * from user LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
7	15 12:47:19	select * from user LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
8	16 12:48:05	select * from user LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
9	17 12:49:40	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

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.

The screenshot shows the Postman interface. In the top navigation bar, 'Home', 'Workspaces', 'API Network', and 'Explore' are visible. The main area shows a collection named 'My Workspace' with a 'Register' item. A PUT request is selected with the URL 'localhost:8080/users/56'. The 'Body' tab is selected, showing JSON data:

```

1   {
2     "id": 56,
3     "firstname": "Deepthi",
4     "lastname": "M",
5     "registernumber": "488CS521012",
6     "dob": "18-09-2005",
7     "address": "Loknayaknagar",
8     "department": "Computer Science"
9   }

```

The 'Send' button is highlighted with a red box. Below the request, the status bar shows 'Status: 200 OK'.

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.

The screenshot shows the MySQL Workbench interface. The 'Schemas' tree on the left shows the 'register' schema with its tables: 'data2', 'user', 'user_seq', 'Views', 'Stored Procedures', and 'Functions'. The 'user' table is selected. The 'Query' editor at the top has the following SQL code:

```

1 * use register;
2 * select * from user;

```

The 'Result Grid' below shows the 'user' table with the following data:

	id	address	dob	department	registernumber	firstname	lastname
1	55	W.Mebula	02-08-2005	Computer Science	488CS521007	Rekha	K
2	56	Loknayaknagar	18-09-2005	Computer Science	488CS521012	Deepthi	M

The 'Output' pane at the bottom shows the execution history:

Action	Time	Action	Message	Duration / Fetch
10	12:30:21	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
11	12:31:44	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
12	12:31:59	select * from user LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
13	12:37:45	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
14	12:38:19	select * from user LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
15	12:47:19	select * from user LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
16	12:48:05	select * from user LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
17	12:49:40	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
18	12:50:47	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

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.

The screenshot shows the Postman interface. In the top navigation bar, 'My Workspace' is selected. On the left sidebar, under 'Environments', there is a single entry: 'localhost:8080'. In the main workspace, a collection named 'Register' is expanded, showing four requests: 'GET localhost:8080/users', 'POST localhost:8080/users', 'PUT localhost:8080/users/2', and 'DELETE localhost:8080/users/2'. The 'DELETE' request is highlighted with a red box. The URL field contains 'localhost:8080/users/58'. The 'Send' button is also highlighted with a red box. Below the URL field, the 'Query Params' section is visible. At the bottom of the screen, the status bar shows 'Status: 200 OK' and 'Time: 23 ms'. The bottom right corner of the status bar is also highlighted with a red box.

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.

The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left displays the database structure. A connection named 'MyConnection1' is selected. Under the 'register' schema, the 'Tables' section shows 'user', 'user_seq', and 'views'. The 'user' table is selected. The 'Query 1' tab in the center contains the following SQL code:

```
1 * use register;
2 * select * from user;
```

The result grid shows one row of data:

ID	Address	Dob	Department	Registernumber	Firstname	Lastname
55	VV Mohalla	02-04-2005	Computer Science	488CS21007	Brinda	K

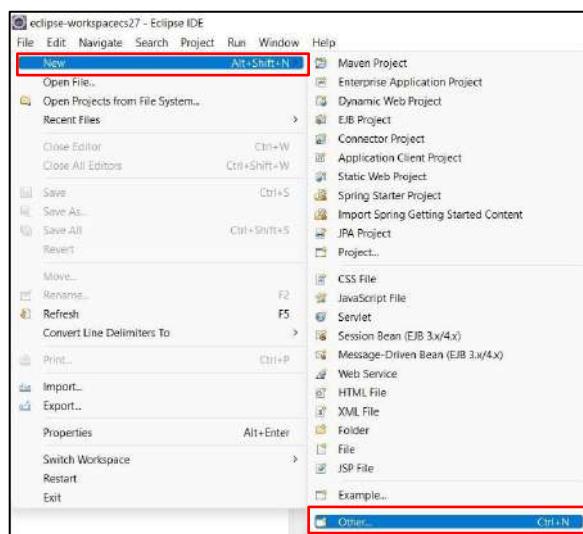
The 'Output' pane at the bottom shows the execution history with 19 entries, all of which completed successfully with 0.000 sec / 0.000 sec duration.

WEEK – 9

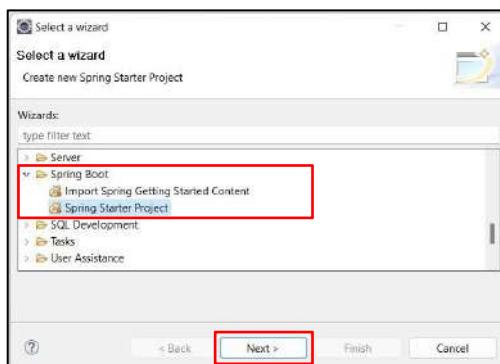
1. Build user authentication flow and authorization using SpringSecurity.

Step 1: Open ‘Eclipse IDE for Enterprise edition’ & go to → Help → Eclipse Marketplace as shown below.

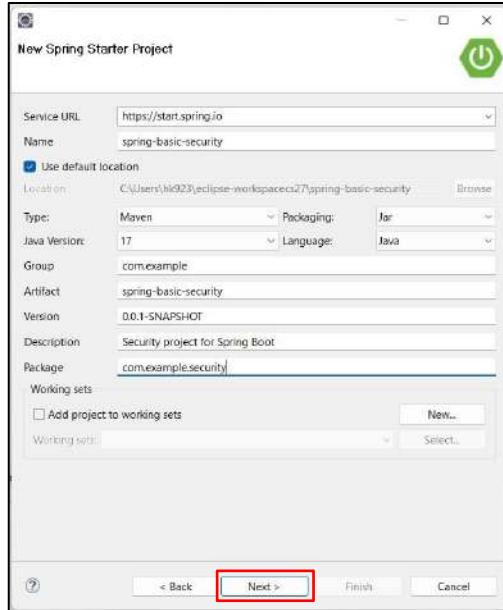
Now search for ‘Spring tool suite’ & click on ‘Install’ to install the latest version. After the installation is complete, go to → File → New → Other option as shown below.



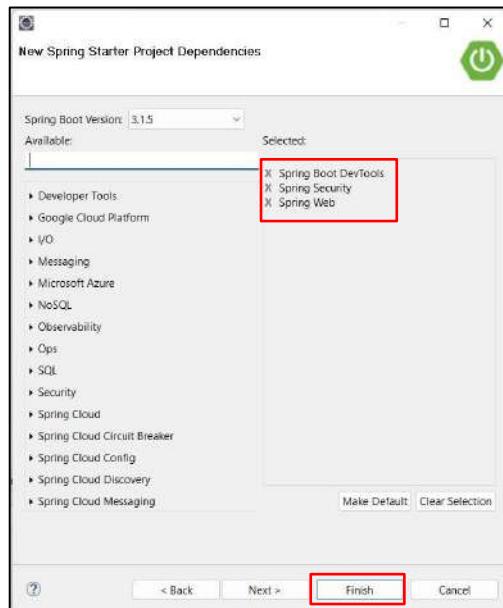
Step 2: Now select ‘Spring Starter Project’ & click on ‘Next’ option.



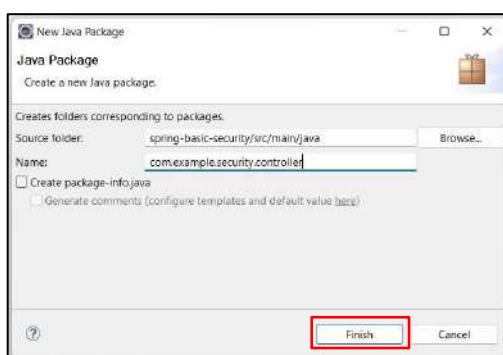
Step 3: Give the Name as ‘User’ → select ‘Maven’ → Description as User Registration Spring Boot Application → Package as ‘spring-basic-security’ & then click on ‘Next’.



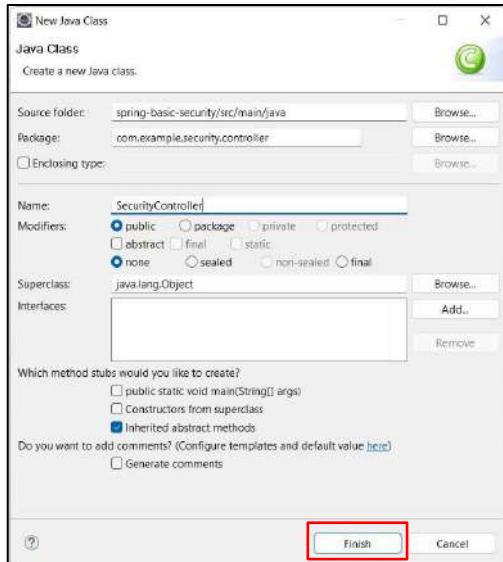
Step 4: Now add all the required dependencies as shown below & click on 'Finish'.



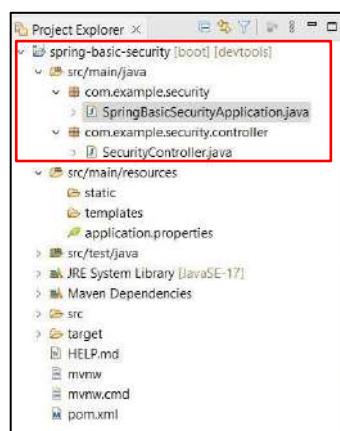
Step 5: Create a new package named 'com.example.security.controller' & click on 'Finish'.



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.

```
SecurityController.java X SpringBasicSecurityApplication.java application.properties
1 package com.example.security.controller;
2 import org.springframework.web.bind.annotation.GetMapping;
3 import org.springframework.web.bind.annotation.RestController;
4 @RestController
5 public class SecurityController {
6     @GetMapping("/")
7     public String welcome() {
8         return ("Welcome to SpringBoot Security Application");
9     }
10 }
11
```

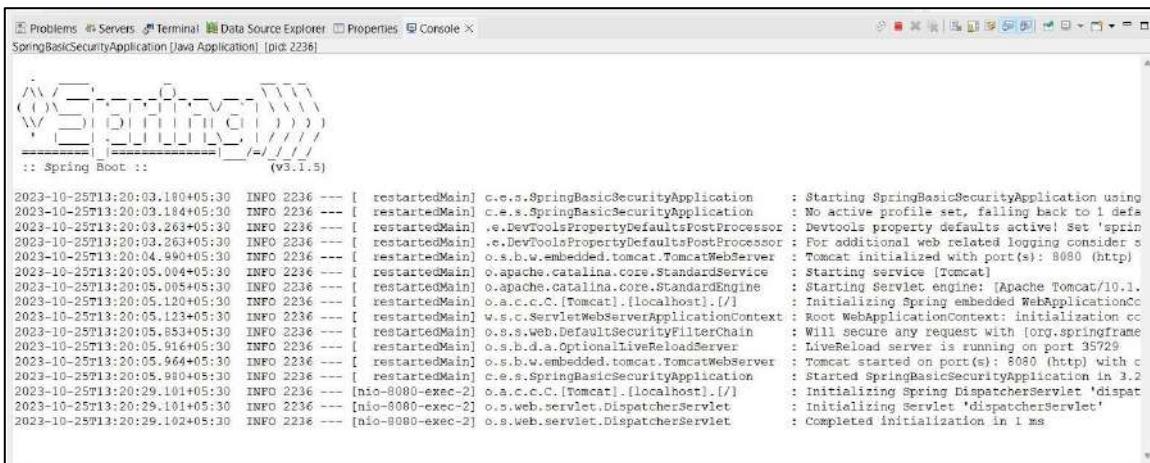
Step 9: Now add code to SpringBasicSecurityApplication.java file as shown below.

```
SecurityController.java X *SpringBasicSecurityApplication.java application.properties
1 package com.example.security;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class SpringBasicSecurityApplication {
7
8     public static void main(String[] args) {
9         SpringApplication.run(SpringBasicSecurityApplication.class, args);
10    }
11 }
12
```

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

```
SecurityController.java X SpringBasicSecurityApplication.java application.properties X
1 spring.security.user.name=user
2 spring.security.user.password=password
3 server.port=8080
4
```

Step 11: Start the server as shown below.

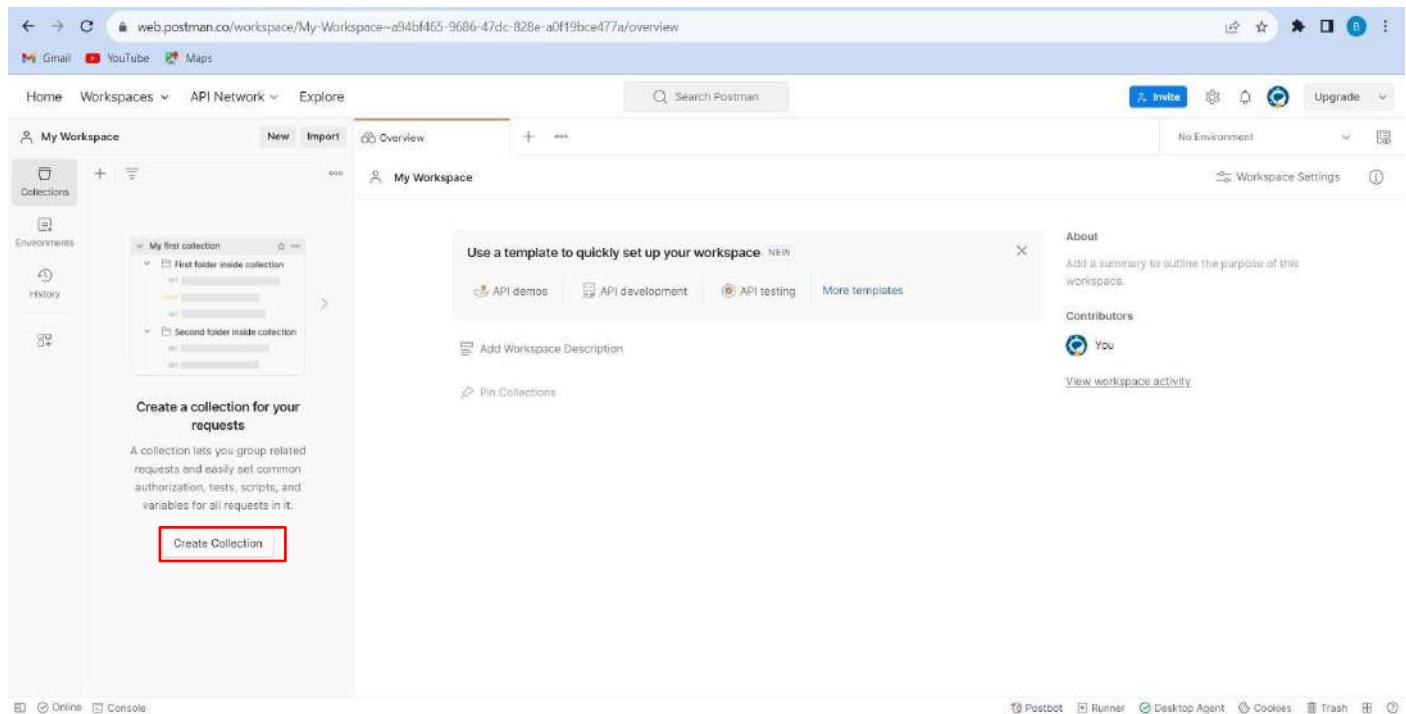


The screenshot shows the Eclipse IDE interface with the 'Servers' view open. The terminal window displays the logs for the 'SpringBasicSecurityApplication' Java Application. The logs indicate the application is starting up, including the configuration of Tomcat and the loading of Spring Basic Security Application classes. The log output is as follows:

```
Spring Basic Security Application [Java Application] [pid: 2236]
:: Spring Boot :: (v3.1.5)

2023-10-25T13:20:03.180+05:30 INFO 2236 --- [ restartedMain] c.e.s.SpringBasicSecurityApplication : Starting SpringBasicSecurityApplication using
2023-10-25T13:20:03.184+05:30 INFO 2236 --- [ restartedMain] c.e.s.SpringBasicSecurityApplication : No active profile set, falling back to 1 defa
2023-10-25T13:20:03.263+05:30 INFO 2236 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring
2023-10-25T13:20:03.263+05:30 INFO 2236 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider s
2023-10-25T13:20:04.990+05:30 INFO 2236 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-10-25T13:20:05.004+05:30 INFO 2236 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-10-25T13:20:05.120+05:30 INFO 2236 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[] : Starting Servlet engine: [Apache Tomcat/10.1.
2023-10-25T13:20:05.123+05:30 INFO 2236 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Initializing Spring embedded WebApplicationCon
2023-10-25T13:20:05.853+05:30 INFO 2236 --- [ restartedMain] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframework
2023-10-25T13:20:05.916+05:30 INFO 2236 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2023-10-25T13:20:05.964+05:30 INFO 2236 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with c
2023-10-25T13:20:05.980+05:30 INFO 2236 --- [nio-8080-exec-2] c.e.s.SpringBasicSecurityApplication : Started SpringBasicSecurityApplication in 3.2
2023-10-25T13:20:29.101+05:30 INFO 2236 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Initializing DispatcherServlet 'dispatcher'
2023-10-25T13:20:29.101+05:30 INFO 2236 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
```

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



The screenshot shows the Postman workspace interface. On the left, there's a sidebar with 'Collections' selected. In the main area, a 'My Workspace' section shows a collection named 'My first collection' with two folders: 'First folder inside collection' and 'Second folder inside collection'. Below this, there's a 'Create a collection for your requests' section with a note about collections and a 'Create Collection' button, which is highlighted with a red box. A modal dialog titled 'Use a template to quickly set up your workspace' is open, showing options like 'API demos', 'API development', 'API testing', and 'More templates'. The 'About' section of the workspace has a note to add a summary and a 'View workspace activity' link. At the bottom, there are various workspace settings and a toolbar.

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.

The screenshot shows the Postman interface. A red box highlights the 'Send' button in the top right corner of the main request configuration area. The URL field contains 'localhost:8080/'. The status bar at the bottom indicates 'Status: 401 Unauthorized'.

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

The screenshot shows the Postman interface with the 'Authorization' tab selected. A red box highlights the 'Basic Auth' option in the dropdown menu under the 'Type' section. The status bar at the bottom indicates 'Status: 401 Unauthorized'.

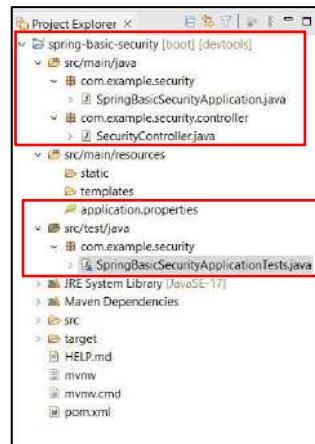
Step 15: Now enter Username & Password as mentioned below. Then Save it & click on Send to view the output.

The screenshot shows the Postman interface. In the left sidebar, 'My Workspace' is selected. A collection named 'Security' is expanded, showing an endpoint 'localhost:8080/'. The main area shows a 'GET' request to 'localhost:8080/'. The 'Authorization' tab is selected, showing 'Basic Auth' with 'Username' set to 'user' and 'Password' set to 'password'. The 'Send' button is highlighted with a red box. Below the request, the status bar shows 'Status: 200 OK'. The response body is displayed in a red box, containing the text: '1 Welcome to SpringBoot Security Application'.

- Implementing Junit for the above Security API**

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’.



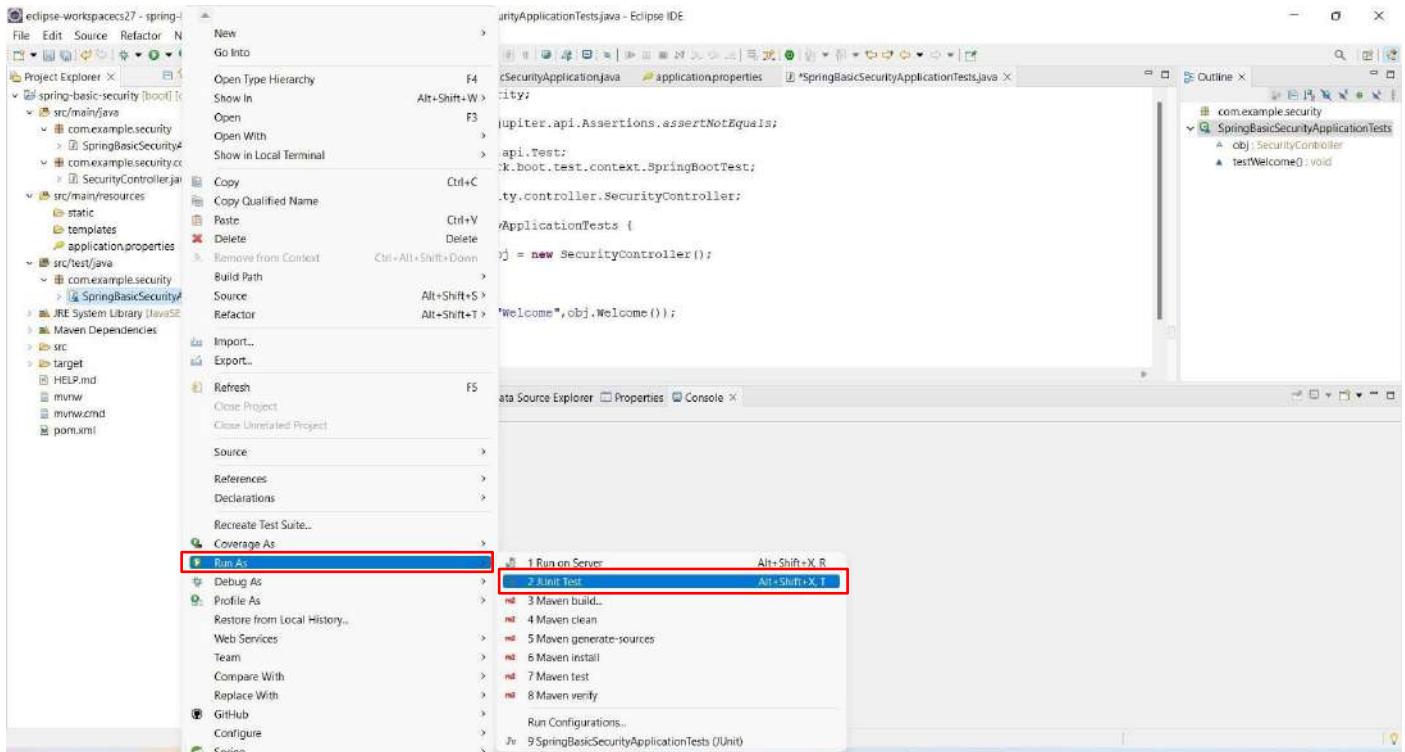
Step 17: Now add code to ‘SpringBasicSecurityApplicationTests.java’ as shown below.

```

1 package com.example.security;
2
3 import static org.junit.jupiter.api.Assertions.assertNotEquals;
4
5 import org.junit.jupiter.api.Test;
6 import org.springframework.boot.test.context.SpringBootTest;
7
8 import com.example.security.controller.SecurityController;
9
10 @SpringBootTest
11 class SpringBasicSecurityApplicationTests {
12
13     SecurityController obj = new SecurityController();
14
15     @Test
16     void testWelcome() {
17         assertNotEquals("Welcome", obj.Welcome());
18     }
}

```

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



Step 19: View the output as shown below.

```

14:52:52.725 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could not detect default configuration classes
14:52:52.855 [main] INFO org.springframework.boot.test.context.SpringBootTestBootstrapper -- Found @SpringBootConfiguration com.example.security
14:52:53.135 [main] INFO org.springframework.boot.devtools.restart.RestartApplicationListener -- Restart disabled due to context in which it is running

:: Spring Boot ::      (v3.1.5)

2023-10-25T14:52:53.484+05:30  INFO 2732 --- [           main] .e.s.SpringBasicSecurityApplicationTests : Starting SpringBasicSecurityApplicationTests
2023-10-25T14:52:53.486+05:30  INFO 2732 --- [           main] .e.s.SpringBasicSecurityApplicationTests : No active profile set, falling back to 1 defa
2023-10-25T14:52:55.676+05:30  INFO 2732 --- [           main] o.s.s.w.DefaultSecurityFilterChain : Will secure any request with [org.springframework
2023-10-25T14:52:55.757+05:30  INFO 2732 --- [           main] .e.s.SpringBasicSecurityApplicationTests : Started SpringBasicSecurityApplicationTests i
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended

```

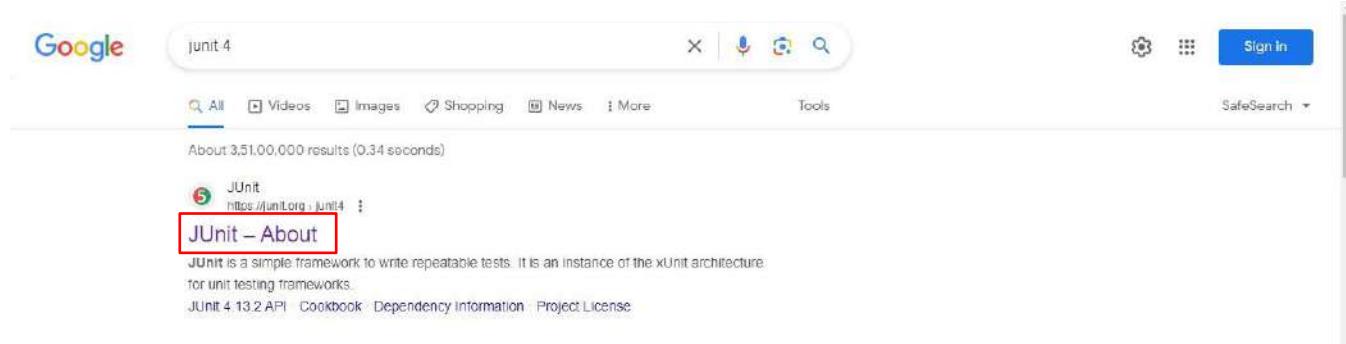
Step 20: The Junit has executed successfully.

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.

The screenshot shows the Eclipse IDE JUnit View. It displays the test results for 'SpringBasicSecurityApplicationTests'. The summary shows 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. Below this, a tree view shows the test class 'SpringBasicSecurityApplicationTests' and the specific test method 'testWelcome()'. The status bar at the bottom right of the window shows a green progress bar, indicating that the test was successful.

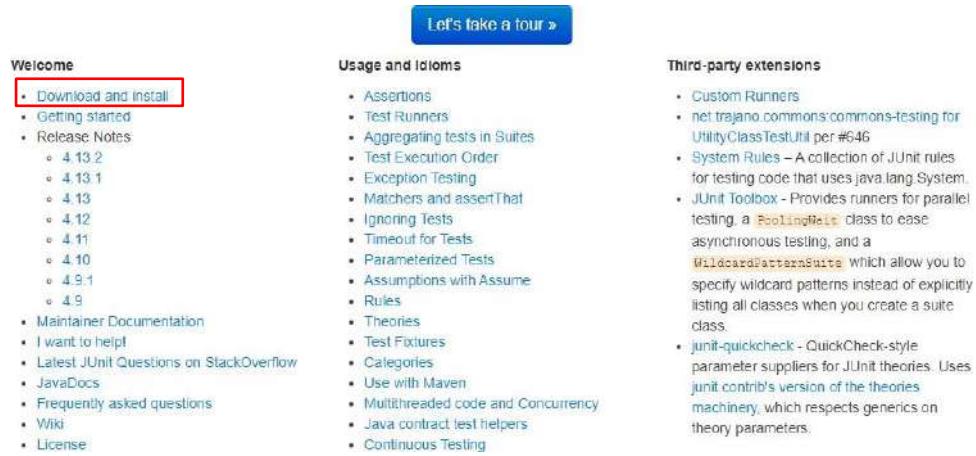
2. Writing Junit test cases for CRUD operations

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



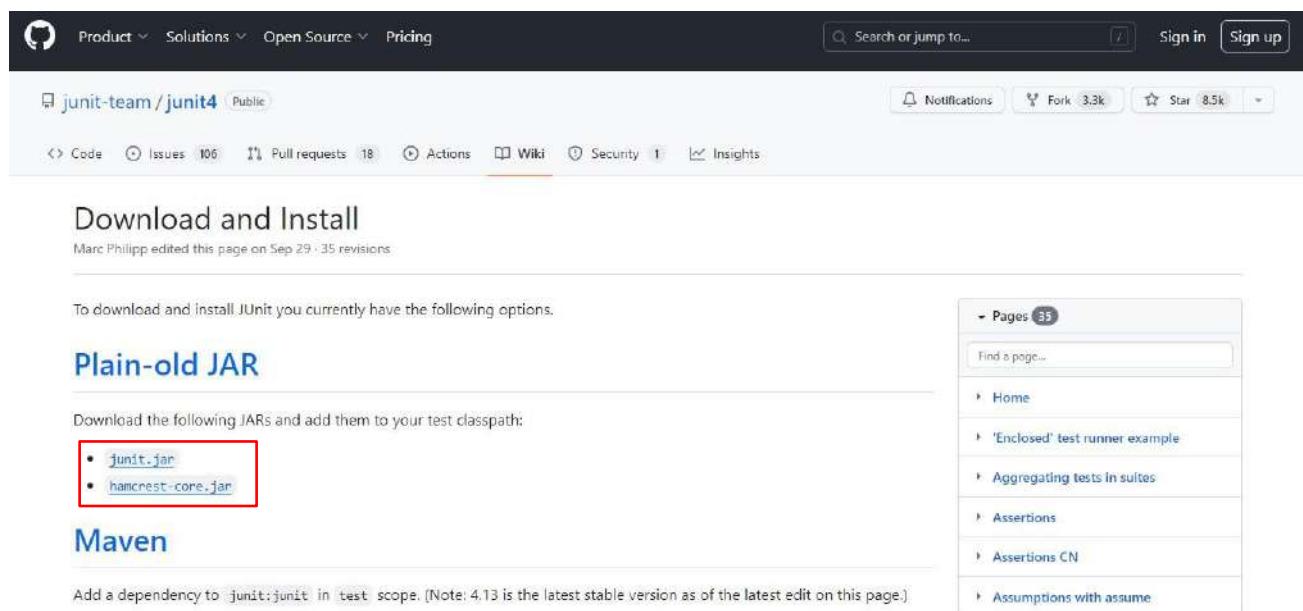
A screenshot of a Google search results page. The search term 'junit 4' is entered in the search bar. The top result is a link to the JUnit website: <https://junit.org/junit4>. The title 'JUnit – About' is displayed next to the link, and it is highlighted with a red box. Below the title, a brief description states: 'JUnit is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks.' There are also links for 'JUnit 4.13.2 API', 'Cookbook', 'Dependency Information', and 'Project License'.

Step 2: Click on ‘Download & Install’ option.



A screenshot of the JUnit website. The 'Welcome' section on the left has a red box around the 'Download and install' link. Other links in this section include 'Getting started', 'Release Notes' (with versions 4.13.2, 4.13.1, 4.13, 4.12, 4.11, 4.10, 4.9.1, 4.9), 'Maintainer Documentation', 'I want to help!', 'Latest JUnit Questions on StackOverflow', 'JavaDocs', 'Frequently asked questions', 'Wiki', and 'License'. The 'Usage and Idioms' section in the center lists various testing concepts like Assertions, Test Runners, etc. The 'Third-party extensions' section on the right lists various third-party tools and libraries.

Step 3: Now click on ‘junit.jar’ file here.



A screenshot of the GitHub repository for [junit-team/junit](#). The repository has 3.3k forks and 8.5k stars. The 'Download and Install' page shows options for 'Plain-old JAR' and 'Maven'. Under 'Plain-old JAR', there are two download links: 'junit.jar' and 'hamcrest-core.jar', with 'junit.jar' highlighted by a red box. The 'Maven' section shows a dependency example: 'Add a dependency to `junit:junit` in `test` scope. (Note: 4.13 is the latest stable version as of the latest edit on this page.)'. A sidebar on the right shows a list of pages including 'Home', 'Enclosed' test runner example, 'Aggregating tests in suites', 'Assertions', 'Assertions CN', and 'Assumptions with assume'.

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-.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-.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	60
junit-4.13.2-.pom.md5	2021-02-13 16:31	32
junit-4.13.2-.pom.sha1	2021-02-13 16:31	40

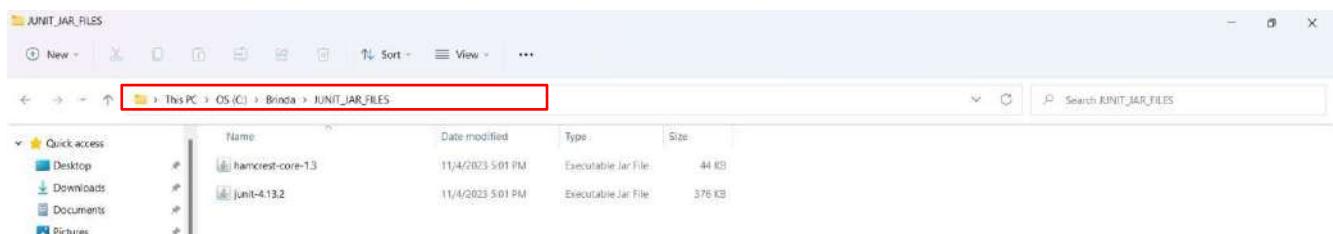
Step 5: Also select ‘hamcrest-core.jar’ file and download the ‘hamcrest-core-1.3.jar’ version here.

org/hamcrest/hamcrest-core/1.3

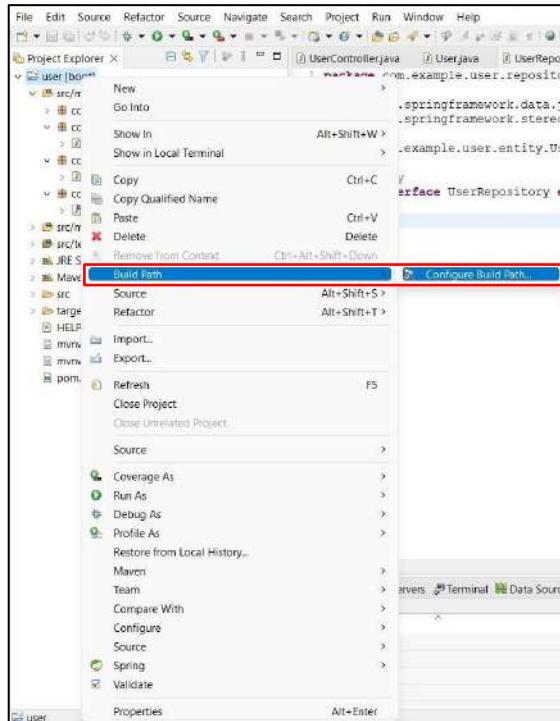
...		
hamcrest-core-1.3-javadoc.jar	2012-07-09 21:08	242519
hamcrest-core-1.3-javadoc.jar.asc	2012-07-09 21:08	499
hamcrest-core-1.3-javadoc.jar.asc.md5	2012-07-09 21:08	32
hamcrest-core-1.3-javadoc.jar.asc.sha1	2012-07-09 21:08	40
hamcrest-core-1.3-javadoc.jar.md5	2012-07-09 21:08	32
hamcrest-core-1.3-javadoc.jar.sha1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

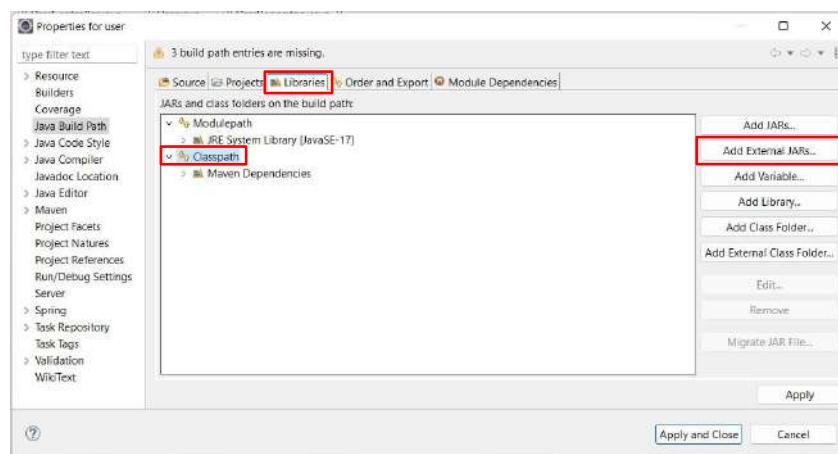
‘JUNIT_JAR_FILES’ & copy the downloaded jar files & paste it here.



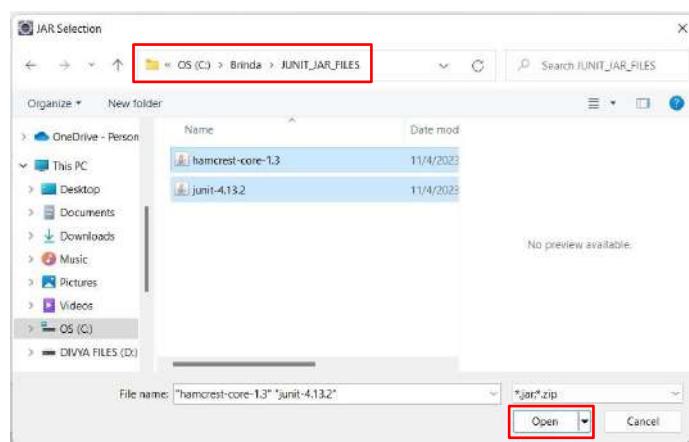
Step 7: Go to Eclipse user Build Path Configure Build Path here.



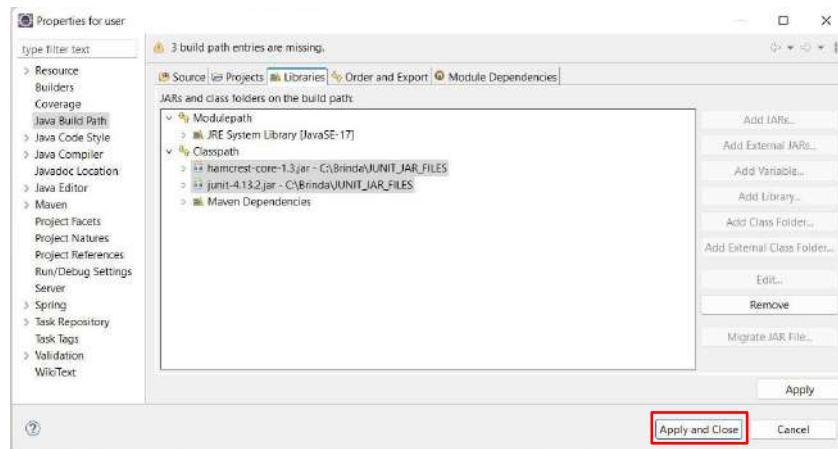
Step 8: Go to Libraries □ Classpath □ Add External JARs.. here.



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.

```
mysql> create database junit;
Query OK, 1 row affected (0.02 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| data2   |
| information_schema |
| junit    |
| mysql    |
| performance_schema |
| register |
| sys      |
| vvp      |
+-----+
8 rows in set (0.00 sec)
```

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

ID	email	firstname	lastname
1	lalit	PSO	CS
2	junit123@gmail.com	JUNIT	JAR
102	vvp123@gmail.com	JAVA	VVP
103	Bnil23@gmail.com	Brida	K
104	Bhoomi123@gmail.com	Bhoomika	R
105	Deepu123@gmail.com	Deepthi	M
106	comp456@gmail.com	Computer	Science
107	nils	nils	nils

Output:

Time	Action	Message	Duration / Fetch
2 17:24:56	select * from junit LIMIT 0, 1000	0 rows() returned	0.016 sec / 0.000 sec
3 18:12:18	use junit	0 row(s) affected	0.015 sec
4 18:12:26	select from user LIMIT 0, 1000	Error Code: 1146. Table 'junit.user' doesn't exist.	0.000 sec
5 18:12:39	select * from user LIMIT 0, 1000	Error Code: 1146. Table 'junit.user' doesn't exist.	0.000 sec
6 18:19:08	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
7 18:24:59	select * from user LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
8 18:26:51	select * from user LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
9 18:27:22	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
10 19:36:18	select * from user LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec

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

The screenshot shows the Eclipse IDE interface. The Project Explorer view on the left has a red box around the package 'com.example.user'. The main editor view on the right contains the code for 'UserController.java'.

```
1 package com.example.user.controller;
2 import java.util.List;
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.http.ResponseEntity;
5 import org.springframework.web.bind.annotation.DeleteMapping;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.PathVariable;
8 import org.springframework.web.bind.annotation.PostMapping;
9 import org.springframework.web.bind.annotation.PutMapping;
10 import org.springframework.web.bind.annotation.RequestBody;
11 import org.springframework.web.bind.annotation.RequestMapping;
12 import org.springframework.web.bind.annotation.RestController;
13
14 import com.example.user.entity.User;
15 import com.example.user.repository.UserRepository;
16
17 @RestController
18 @RequestMapping("/users")
19 public class UserController {
20     @Autowired
21     private UserRepository userRepository;
22
23     @GetMapping
24     public List<User> getAllUser() {
25         return this.userRepository.findAll();
26     }
27
28     @GetMapping("/{id}")
29     public User getUserById(@PathVariable("value=id") long userId) {
30         return this.userRepository.findById(userId).orElseThrow();
31     }
32 }
33
```

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

```
35
36     @PostMapping
37     public User createUser(@RequestBody User user) {
38         return this.userRepository.save(user);
39     }
40
41     @PutMapping("/{id}")
42     public User updateUser(@RequestBody User user, @PathVariable("id") long userId) {
43         User ex=this.userRepository.findById(userId).orElseThrow();
44         ex.setFirstname(user.getFirstname());
45         ex.setLastname(user.getLastname());
46         ex.setEmailD(user.getEmailID());
47         return this.userRepository.save(ex);
48     }
49
50     @DeleteMapping("/{id}")
51     public ResponseEntity<User> deleteUser(@PathVariable("id") long userId) {
52         User ex=this.userRepository.findById(userId).orElseThrow();
53         this.userRepository.delete(ex);
54         return ResponseEntity.ok().build();
55     }
56 }
57
```

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

```

1 package com.example.user.entity;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.GeneratedValue;
5 import jakarta.persistence.GenerationType;
6 import jakarta.persistence.Id;
7 import jakarta.persistence.Table;
8
9 @Entity
10 @Table(name="user")
11 public class User {
12     @Id
13     @GeneratedValue(strategy=GenerationType.AUTO)
14     private Long id;
15     private String firstname;
16     private String lastname;
17     private String emailID;
18
19     public User() {
20     }
21
22     public Long getId() {
23         return id;
24     }
25     public void setId(Long id) {
26         this.id = id;
27     }
28     public String getFirstname() {
29         return firstname;
30     }
31     public void setFirstname(String firstname) {
32         this.firstname = firstname;
33     }
34     public String getLastname() {
35         return lastname;
36     }
37     public void setLastname(String lastname) {
38         this.lastname = lastname;
39     }
40     public String getEmailID() {
41         return emailID;
42     }
43
44     public void setEmailID(String emailID) {
45         this.emailID = emailID;
46     }
47     public User(Long id, String firstname, String lastname, String emailID) {
48         super();
49         this.id = id;
50         this.firstname = firstname;
51         this.lastname = lastname;
52         this.emailID = emailID;
53     }
54 }
55

```

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

```

1 package com.example.user.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.example.user.entity.User;
7
8 @Repository
9 public interface UserRepository extends JpaRepository<User, Long>
10 {
11 }
12
13

```

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

```

1 package com.example.user;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class UserApplication {
7
8     public static void main(String[] args) {
9         SpringApplication.run(UserApplication.class, args);
10    }
11
12
13 }
14

```

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

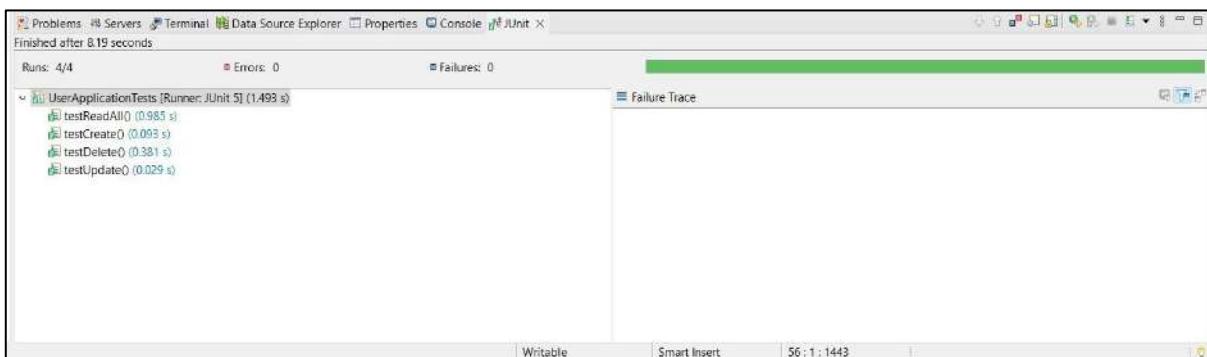
```
① *UserController.java ② *User.java ③ *UserRepository.java... ④ *UserApplication.j... ⑤ *application.prop... ⑥ *UserApplicationT... ⑦
```

1 spring.datasource.url=jdbc:mysql://localhost:3306/junit
2 spring.datasource.username=root
3 spring.datasource.password=Vvp_dlt
4 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
5 spring.jpa.hibernate.ddl-auto=update
6
7

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.

```
1 UserController.java  2 User.java  3 UserRepository.java  4 UserApplication.java  5 application.prop...  6 "UserApplicationT... X
1 package com.example.user;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4
5 @SpringBootTest
6 class UserApplicationTests {
7     @Autowired
8     UserRepository userRepo;
9
10    @Test
11    public void testCreate() {
12        User u= new User();
13        u.setid(1L);
14        u.setFirstname("VVVFSDS");
15        u.setLastname("FSD");
16        u.setEmail("DDDD@1234.com");
17        userRepo.save(u);
18        assertThat(userRepo.findById(1L).get());
19    }
20
21    @Test
22    public void testReadAll() {
23        List<User> list=userRepo.findAll();
24        assertThat(list).size().isGreaterThan(0);
25    }
26
27    @Test
28    public void testUpdate() {
29        User u=userRepo.findById(106L).get();
30        u.setFirstname("Computer");
31        u.setLastname("Science");
32        userRepo.save(u);
33        assertEquals("Computer",userRepo.findById(106L).get().getFirstname());
34    }
35
36    @Test
37    public void testDelete() {
38        userRepo.deleteById(2L);
39        assertThat(userRepo.existsById(2L)).isFalse();
40    }
41 }
```

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.

MySQL Workbench

MyConnection26

File Edit View Query Database Server Tools Scripting Help

Navigator

Query 1

```
1 * use junit;
2 * select * from user;
```

Result Grid | Filter Rows: | Edit | Export/Imports | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Context Help | Snippets

	id	email	firstname	lastname
1	1	DDDD@1234.com	VPS0SD	FSD
102	102	vvp123@gmail.com	JAVA	VVP
103	103	Bri123@gmail.com	Brinda	K
104	104	Bhoomi123@gmail.com	Bhoomika	R
105	105	Deepu123@gmail.com	Deepthi	M
106	106	comp56@gmail.com	Computer	Science
152	152	DDDD@1234.com	VPS0SD	FSD
202	202	DDDD@1234.com	VPS0SD	FSD
*	1000	1000	1000	1000

user 7 - x

Administration Schemas Information

Schema: junit

Action Output

Time	Action	Message	Duration / Fetch
3 18:12:18	use junit	0 rows affected	0.015 sec
4 18:12:26	select * from user LIMIT 0, 1000	Error Code: 1146: Table 'junit.user' doesn't exist.	0.000 sec
5 18:12:39	select * from user LIMIT 0, 1000	Error Code: 1146: Table 'junit.user' doesn't exist.	0.000 sec
6 18:19:08	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
7 18:24:59	select * from user LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
8 18:26:51	select * from user LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
9 18:27:22	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
10 19:38:18	select * from user LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
11 19:47:33	select * from user LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

3. Installation of Mongodb, mongosh, compass.

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

search.yahoo.com/search?_ylt=Awr2XzefkZl8PoTrZJXNy0A;_ylc=X1MDMjc2NjY3OQRfcgMyBGZyA21jYWZlZQRmcjDc2ltG9wBgdwcmIka2tCQjsjtM1VRU5DRmZRSWxNczlqbEEb19...?o=1

mongodb download

About 124,000 search results

Ad related to: mongodb download

www.couchbase.com

Couchbase Capella vs Atlas - Use SQL Queries On JSON Docs

Price Performance Improves w/ Scale, Do More For Less. Get Started, Try Capella DBaaS Free. Configure For Multi-Cluster, Multi-Region & Multi-Cloud. Get Started and Try Free Today! Integrated w/ Kubernetes · Develop with Agility · Perform at Any Scale · Memory-First Architecture

Full-Text Search for JSON

Easy To Manage, Fully Integrated. Within A Scalable NoSQL Database.

Why Use SQL For JSON?

Develop Apps Quickly Leveraging SQL. Skills With The Power Of NoSQL.

Couchbase Capella DBaaS

Real-time Memory 1st Architecture. Ensures Millisecond Response.

www.mongodb.com/try · Download

Download MongoDB Community Server | MongoDB

MongoDB Community Server Download. The Community version of our distributed database offers a flexible document data model along with support for ad-hoc queries, secondary indexing, and re...

MongoDB Database Tools

MongoDB Command Line Database Tools

MongoDB Atlas

Work with your data as code Documents

MongoDB Inc. American software company

mongoDB

mongodb.com

MongoDB, Inc. is an American software company that develops and provides commercial support for the source-available database MongoDB, a NoSQL database that stores data in JSON-like documents with flexible schemas. Wikipedia

Headquarters: New York, NY

Founder(s): Dwight Merriman, Eliot Horowitz, Kevin P. Ryan

Employees: 4,619

Stock price: \$343.11 (NASDAQ:GM) Yahoo Finance

\$343.11 +14.11 (+4.29%) As of Fri, Nov 3, 2023 3:00PM EST Market closed.

More on Yahoo Finance

English (United States)
English (India)

To switch input methods, press Windows key + space.

Step 2: Click on ‘Download’ here.

MongoDB

Products Solutions Resources Company Pricing

Sign In Try Free

MongoDB Atlas

MongoDB Enterprise Advanced

MongoDB Community Edition

MongoDB Community Server

MongoDB Community
Kubernetes Operator

Tools

Atlas SQL Interface

Mobile & Edge

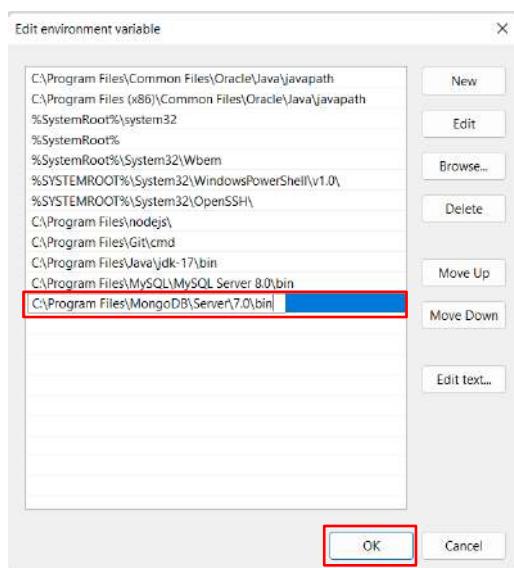
Version 7.0.2 (current)

Platform Windows x64

Package msi

Download Copy link More Options

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



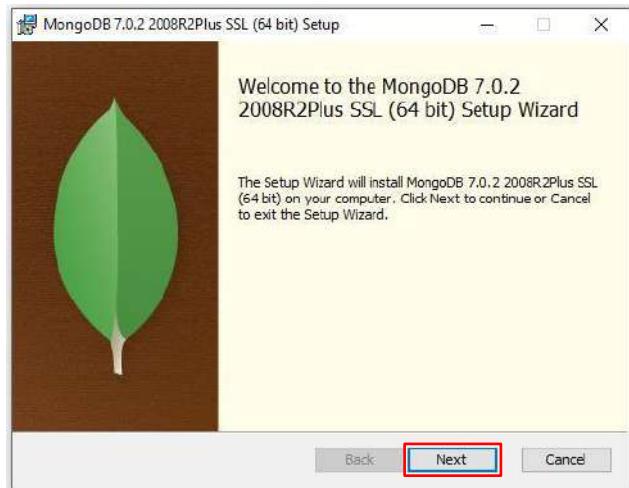
Step 3: Go to web browser □ search for ‘mongodb compass download (GUI)’ □ click on the first link here & download the MongoDB Compass.

A screenshot of a Yahoo search results page. The search query is "mongodb compass download gui". The results show various MongoDB-related links, including "MongoDB Compass Download (GUI)" which is highlighted with a red box. Other results include "MongoDB Atlas", "Community Server", "MongoDB Shell", and "M001". A sidebar on the right lists "Related searches" such as "download and install mongodb compass", "install mongodb compass windows 10", and "mongodb compass install". A "People also search for" section is also visible.

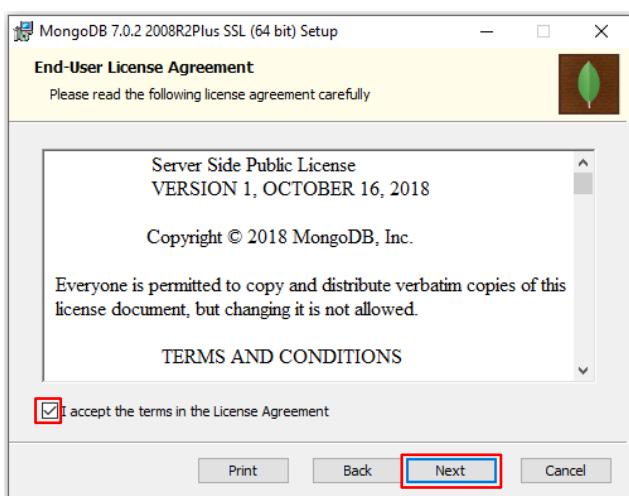
Step 4: Go to web browser □ search for ‘mongodb shell download’ □ click on the first link here & download the MongoDB Shell.

A screenshot of a Yahoo search results page. The search query is "mongodb shell download". The results show various MongoDB-related links, including "MongoDB Shell Download" which is highlighted with a red box. Other results include "MongoDB Shell", "Install mongosh – MongoDB Shell", and "MongoDB Shell | MongoDB". A sidebar on the right lists "Related searches" such as "mongodb shell download windows 10", "install mongodb shell windows 10", and "mongodb shell install".

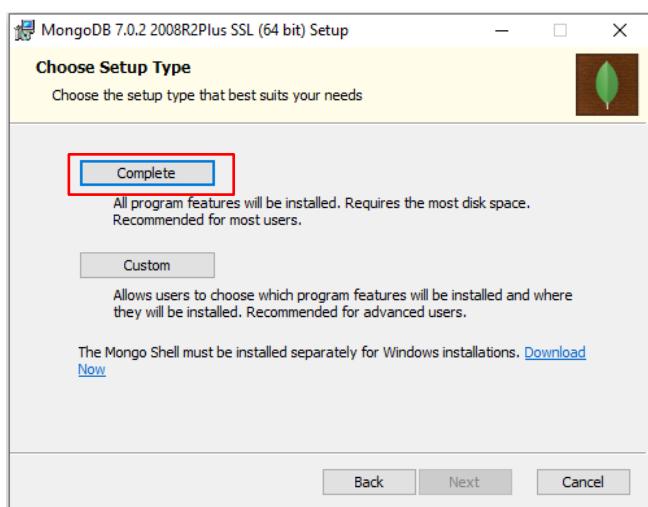
Step 5: Follow the ongoing steps to install mongodb compass, click on ‘Next’.



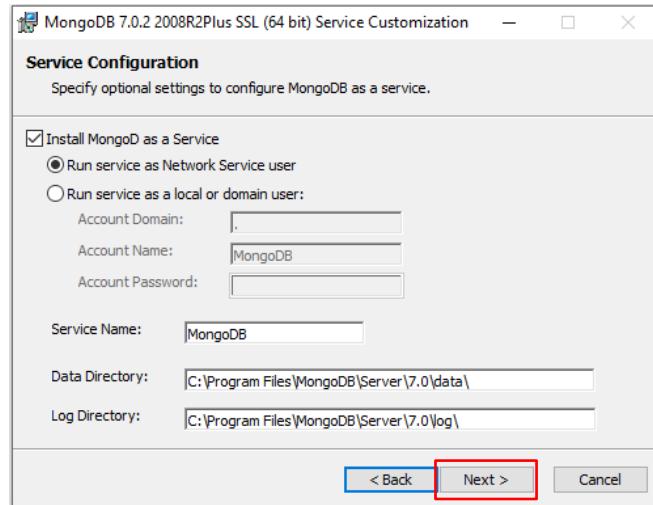
Step 6: Enable the checkbox & click on 'Next'.



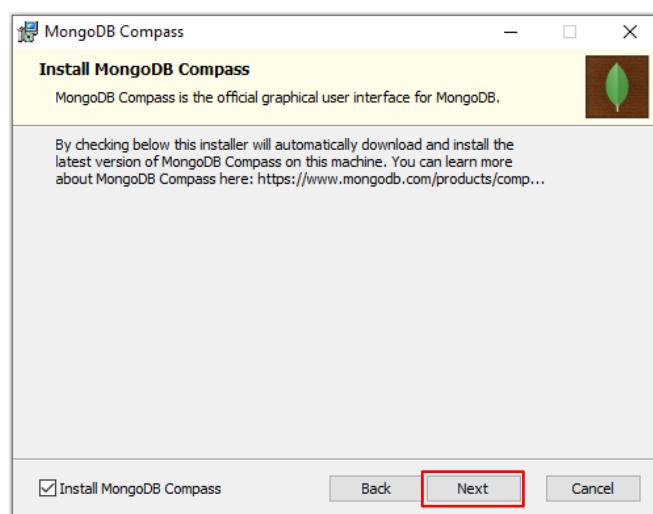
Step 7: Click on 'complete' here.



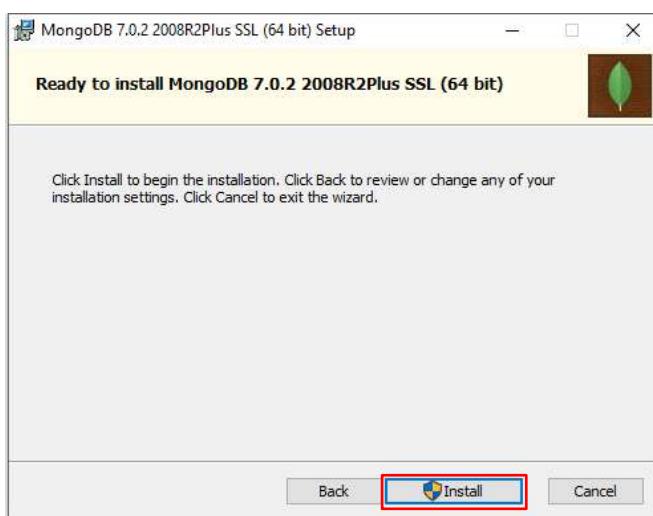
Step 8: Just click on 'Next' here.



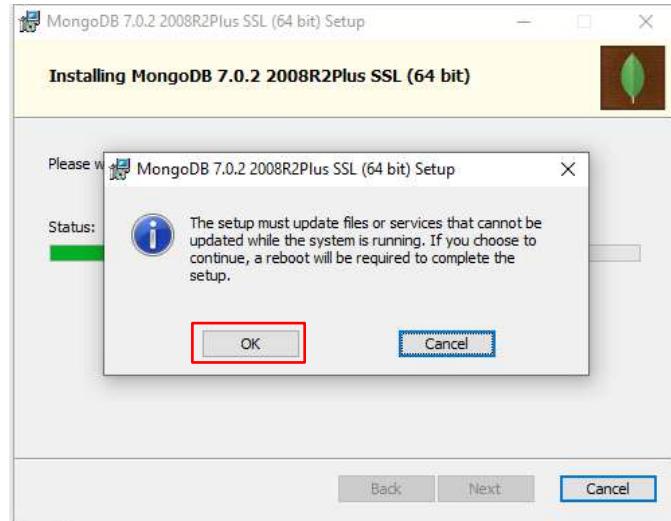
Step 9: Click on ‘Next’.



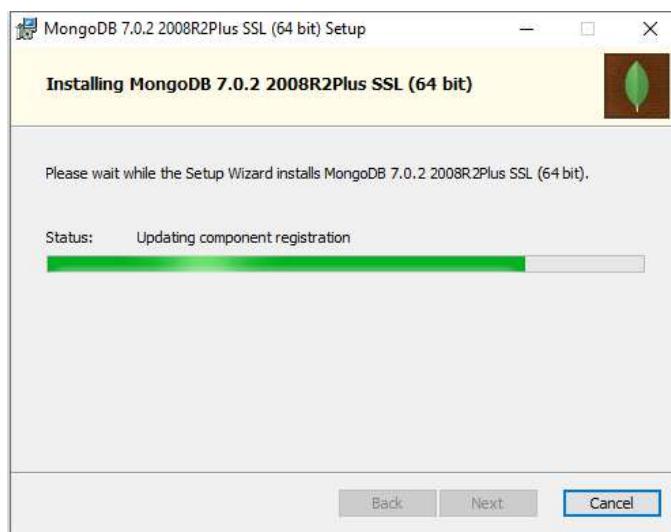
Step 10: Now click on ‘Install’.



Step 11: Click on ‘OK’ here.



Step 12: Wait until the installation is complete.



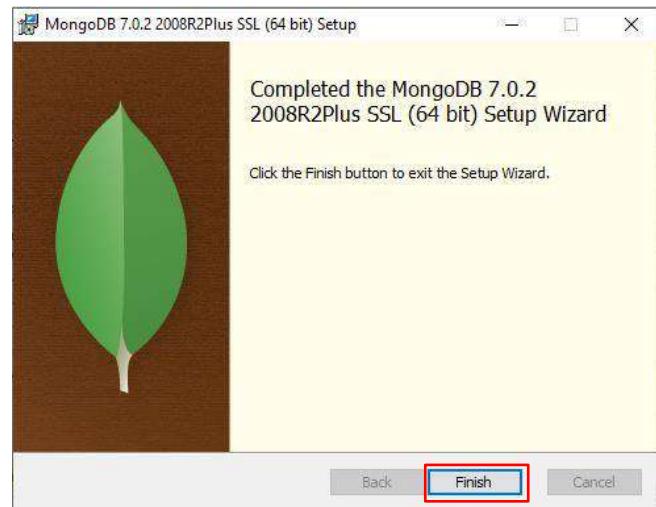
Step 13: MongoDB Compass is ready to launch.



MongoDB Compass is being installed.

It will launch once it is done.

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!

```

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Please enter a MongoDB connection string (Default: mongodb://localhost/):
Current Mongosh Log ID: 654684cf996ef8170f348008
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/mongosh-shell/

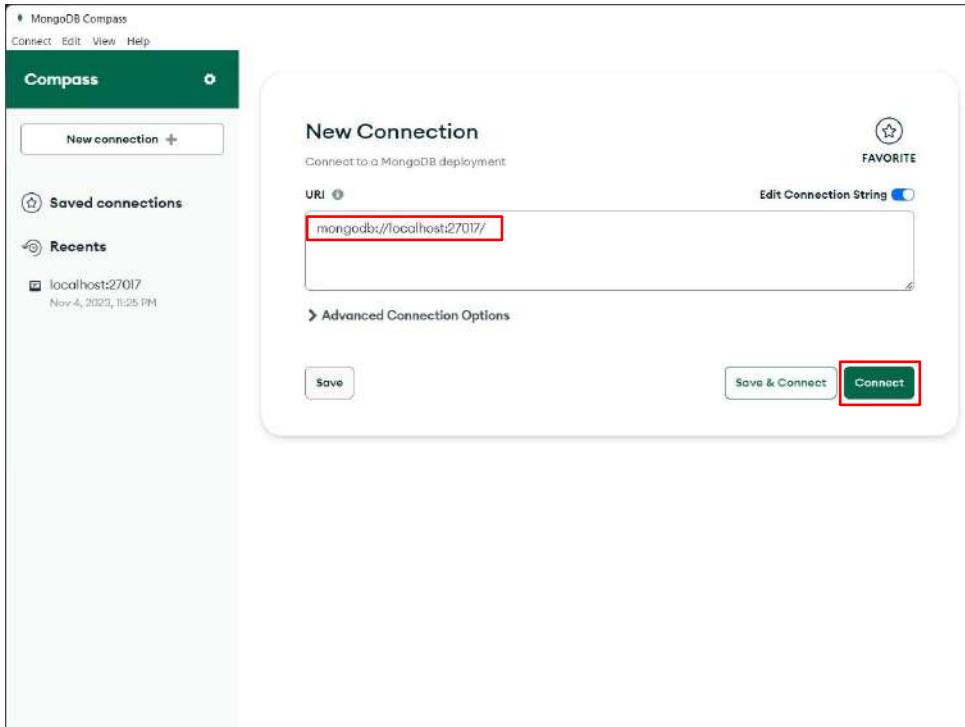
To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2023-11-04T23:16:27.845+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

test> show dbs
admin 40.00 KiB
config 12.00 KiB
local 40.00 KiB
test> use fsd
switched to db fsd
Fsd> db.createCollection("Student")
{ ok: 1 }
Fsd> show dbs
admin 40.00 KiB
config 12.00 KiB
Fsd  8.00 KiB
local 40.00 KiB
Fsd> use test
switched to db test
test> show dbs
admin 40.00 KiB
config 12.00 KiB
Fsd  8.00 KiB
local 40.00 KiB
test>

```

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.

The screenshot shows the MongoDB Compass interface connected to the 'fsd.Student' collection. The left sidebar shows databases like 'admin', 'config', 'local', and 'fsd' (which contains 'Student'). The main area shows the 'fsd.Student' collection with 0 documents and 1 index. It has tabs for 'Documents', 'Aggregations', 'Schema', 'Indexes', and 'Validation'. A search bar and filter dropdown are at the top. In the center, there's a table with one row of data. Below the table, a note says 'This collection has no data' and 'It only takes a few seconds to import data from a JSON or CSV file.' A green 'Import Data' button is at the bottom. At the bottom of the collection view, there's a 'Documents' section with a tree view and a 'Find' button. The 'ADD DATA' button in the top right of the collection view is highlighted with a red box.

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> db.data.insert({“name”:“MongoDB”})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '_id': 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.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Please enter a MongoDB connection string (Default: mongodb://localhost/):
Current Mongosh Log ID: 654b222e2af46a11f69e0f9a
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
2023-11-08T09:10:30.752+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> show dbs
BookStore 72.00 KiB
admin 40.00 KiB
config 108.00 KiB
cs 8.00 KiB
dmns 48.00 KiB
desktop 72.00 KiB
dlt 48.00 KiB
fsd 104.00 KiB
local 72.00 KiB
network 80.00 KiB
sepp 8.00 KiB
test 40.00 KiB
test> use crud
switched to db crud
crud> db.createCollection("Student")
{ ok: 1 }
crud> show collections
Student
crud> show dbs
BookStore 72.00 KiB
admin 40.00 KiB
config 108.00 KiB
crud 8.00 KiB
cs 8.00 KiB
dmns 48.00 KiB
desktop 72.00 KiB
dlt 48.00 KiB
```

➤ 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 the db.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: 'OS'
  }
]

```

➤ 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: 'OS'
  }
]

```

➤ 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.

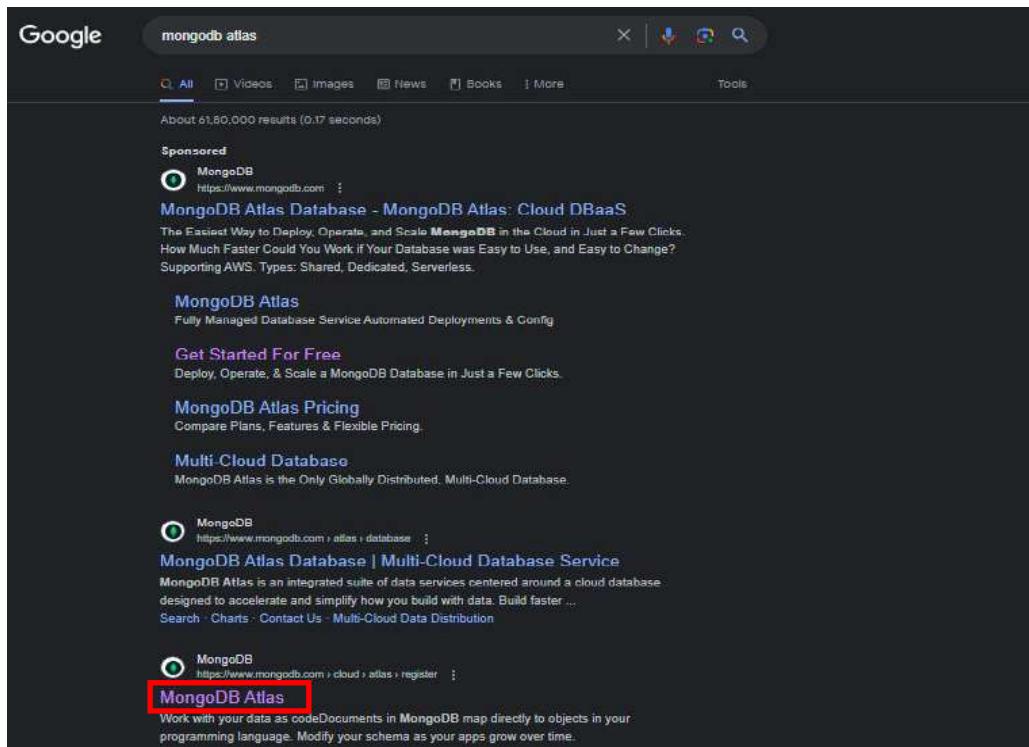
Syntax:The basic syntax of skip() method is as follows –

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

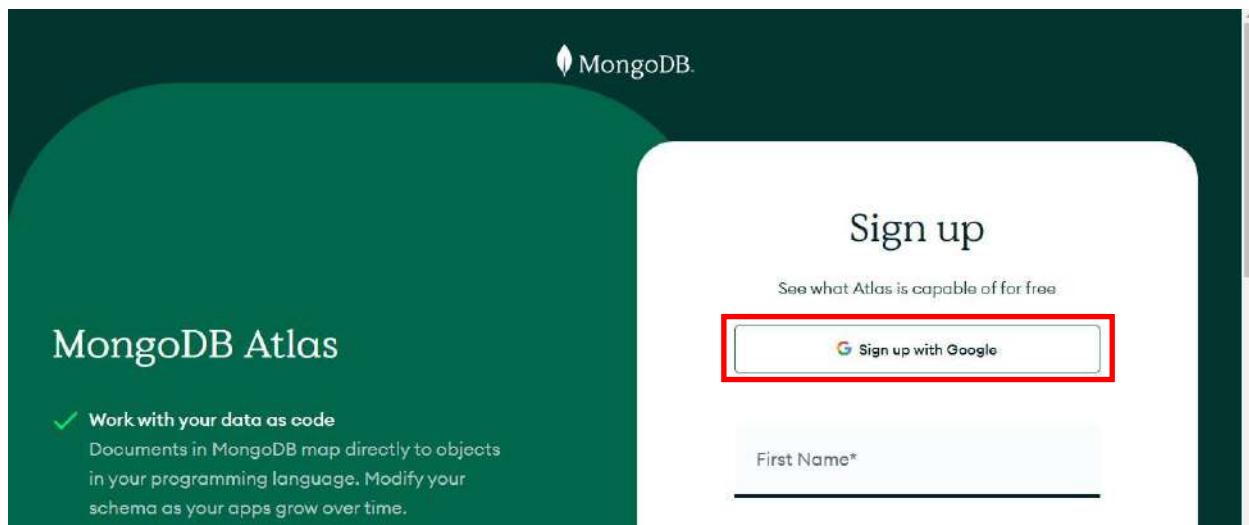
```
crud> db.Student.find({}, {"ussn":1, _id:0}).limit(2).skip(2)
[ { ussn: '488CS123' }, { ussn: '123VVP488' } ]
crud> db.Student.find({}, {"ussn":1, _id:0}).limit(2).skip(1)
[ {}, { ussn: '488CS123' } ]
```

4. How to Run Mongo DB on Cloud

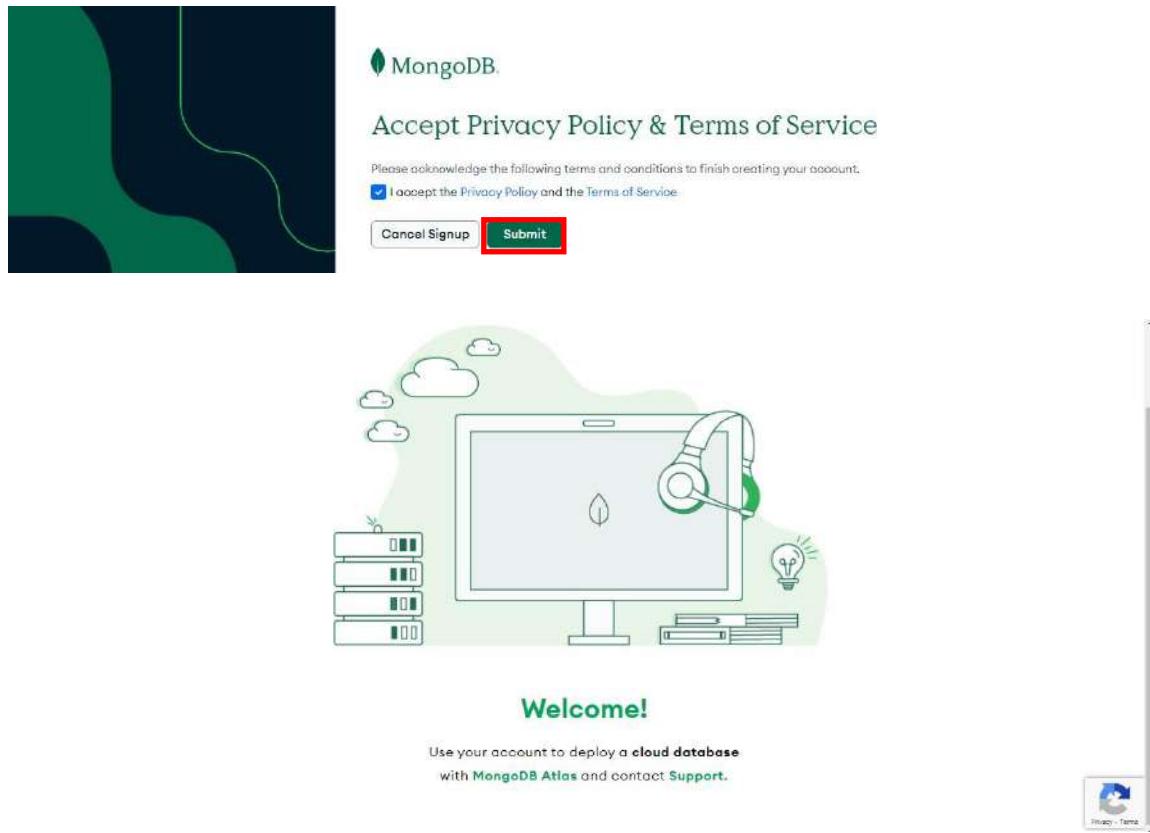
Step 1: Go to Web browser and search for ‘MongoDB Atlas’ & click on the link as shown below.



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.

 **Atlas**

Welcome to Atlas. Let's build something great.

Help us tailor your experience by taking a minute to answer the questions below.

GETTING TO KNOW YOU

What is your primary goal?

Learn MongoDB

How long have you been developing software with MongoDB?

Less than a year

GETTING TO KNOW YOUR PROJECT

What programming language are you primarily building on MongoDB with?

Java

What kind(s) of data will your project use?
You can choose as many as you want

Not aware X

Will your application include any of the following architectural models?
You can choose as many as you want

Mobile X

Finish

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

The screenshot shows the MongoDB deployment interface. At the top, it says "Deploy your database". Below that, there are three plan options: M10 (\$0.08/hour), SERVERLESS (\$0.10/1M reads), and M0 (FREE). The M0 plan is highlighted with a red border. Under each plan, there is a brief description and resource details (Storage, RAM, vCPU). Below the plans, there is a "Provider" section with buttons for AWS (which is highlighted with a red border), Google Cloud, and Azure. A "FREE" label is present. A large green "Create" button is at the bottom, also highlighted with a red border. There is a link "Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime." and a link "View deployment details".

Step 6: Enter the username & password & then click on ‘Create User’.

The screenshot shows the MongoDB "Create User" page. On the left, there is a sidebar with "Data Federation", "Search", "Stream Processing", "SECURITY" (which is highlighted with a teal background), "Quickstart", "Backup", "Database Access", "Network Access", and "Advanced". Below that is a "New On Atlas" button with the number "4" and a "Goto" button. The main area has a heading "Create a database user using a username and password. Users will be given the *read and write* to any database privilege by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password." It contains fields for "Username" (atlas@123) and "Password" (atlas@123), both of which are highlighted with red boxes. There are also "Autogenerate Secure Password" and "Copy" buttons. A green "Create User" button is at the bottom, also highlighted with a red border. A note below says "This password contains special characters which will be URL-encoded."

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

Add entries to your IP Access List

Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the [Network Access Page](#).

IP Address

Description

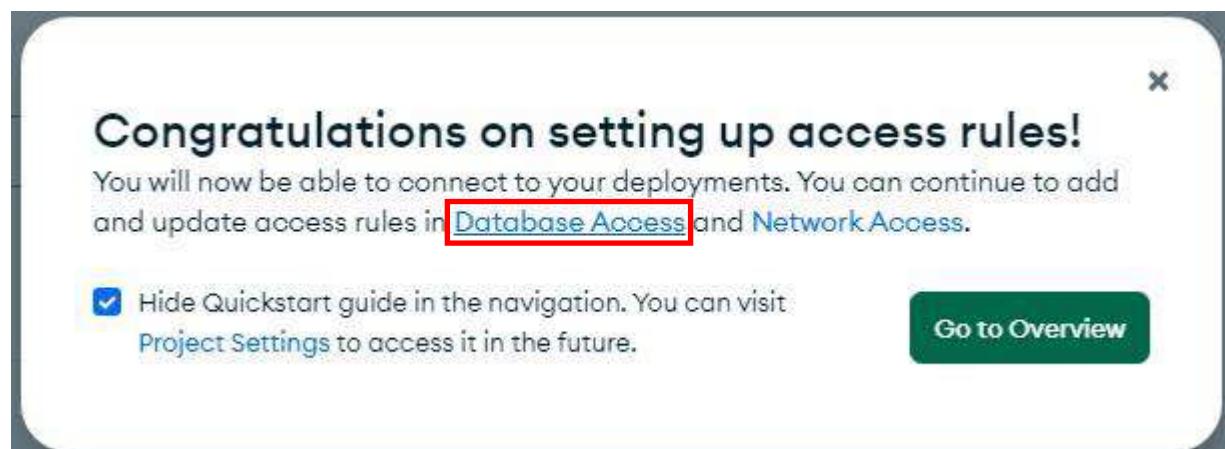
IP Access List

Description

117.211.23.160/32

My IP Address

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



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

The screenshot shows the MongoDB Atlas interface under the 'Data Services' tab. On the left sidebar, 'Project 0' is selected. In the main area, 'Database Access' is chosen, and 'Database Users' is the active tab. A table lists a single user: 'User Name: atlas', 'Authentication Method: SCRAM', and 'MongoDB Roles: readWriteAnyDatabase@admin'. The 'Actions' column for this user has two buttons: 'EDIT' and 'DELETE', which are highlighted with a red box.

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

This is a modal dialog titled 'Edit User: atlas@admin'. It contains several sections: 'Authentication Method' (Password is selected), 'Password Authentication' (username 'atlas' and 'Edit Password' button), 'Database User Privileges' (instructions about roles), 'Built-in Role' (dropdown set to 'I SELECTED'), 'Custom Roles' (instructions about pre-defined roles), 'Specific Privileges' (instructions about collection-level privileges), and 'Restrict Access to Specific Clusters/Federated Database Instances' (a toggle switch is off). At the bottom right are 'Cancel' and 'Update User' buttons, with 'Update User' also highlighted with a red box.

Step 11: Select ‘Database’ & click on ‘Connect’ option here.

The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with various project management and service options like Overview, DEPLOYMENT (highlighted with a red box), SERVICES, SECURITY, etc. In the main area, it says 'Database Deployments' and 'DEEPTHIS ORG - 2023-11-09 > PROJECT 0'. It features a search bar, an 'Edit Config' button, and a '+ Create' button. Below this, there's a section for loading sample datasets to Cluster0, with a 'Load sample dataset' button and a 'Dismiss' button. The 'Connect' button in the top navigation bar is also highlighted with a red box.

Step 12: Now select ‘Compass’ here.

This is a step-by-step connection wizard. Step 1: Set up connection security (done). Step 2: Choose a connection method (done). Step 3: Connect (in progress). The 'Connect to your application' section shows a 'Drivers' option. The 'Access your data through tools' section lists 'Compass' (highlighted with a red box), 'Shell', 'MongoDB for VS Code', and 'Atlas SQL'. At the bottom are 'Go Back' and 'Close' buttons.

Connect to Cluster0

Set up connection security 2 3

Choose a connection method

Connect

Connect to your application

Drivers Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)

Access your data through tools

Compass Explore, modify, and visualize your data with MongoDB's GUI

Shell Quickly add & update data using MongoDB's Javascript command-line interface

MongoDB for VS Code Work with your data in MongoDB directly from your VS Code environment

Atlas SQL Easily connect SQL tools to Atlas for data analysis and visualization

Go Back Close

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

Connect to Cluster0

Set up connection security

Choose a connection method

3 Connect

Connecting with MongoDB Compass

I don't have MongoDB Compass installed

I have MongoDB Compass installed

1. Choose your version of Compass

1.12 or later

See your Compass version in "About Compass".

2. Copy the connection string, then open MongoDB Compass

mongodb+srv://atlas:<password>@cluster0.twrdvku.mongodb.net/



Replace <password> with the password for the atlas user.

When entering your password, make sure that any special characters are URL-encoded.

RESOURCES

[Connect with Compass](#)

[Access your Database Users](#)

[Import and Export Data](#)

[Troubleshoot Connections](#)

[Go Back](#)

[Close](#)

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

MongoDB Compass
Connect Edit View Help

Compass

New connection +

Saved connections

Recents

localhost:27017
Nov 9, 2023, 11:16 AM

New Connection

Connect to a MongoDB deployment



FAVORITE

URI

Edit Connection String

mongodb+srv://atlas:<password>@cluster0.twrdvku.mongodb.net/

> Advanced Connection Options

Save

Save & Connect

Connect

Step 14: Then give the password & click on ‘Connect’.

The screenshot shows the Compass MongoDB client interface. On the left, there's a sidebar with 'Saved connections' and 'Recents'. The main area is titled 'New Connection' with the sub-instruction 'Connect to a MongoDB deployment'. A 'URI' field contains the value 'mongodb+srv://atlas:Vvp_dli@cluster0.twrdvku.mongodb.net/'. To the right of the URI is a 'Edit Connection String' toggle switch. At the bottom right of the dialog are two buttons: 'Save & Connect' and 'Connect', with 'Connect' being highlighted by a red box.

Step 15: Click on ‘+’ icon here.

The screenshot shows the Compass MongoDB client interface on the 'cluster0.twrdvku...' dashboard. It features a sidebar with 'My Queries' and 'Databases'. The 'Databases' section lists 'admin' and 'local' databases. A red box highlights the '+' icon located next to the database list.

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

Create Database

Database Name

college

Collection Name

vvp

Time-Series

Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

➤ Additional preferences (e.g. Custom collation, Capped, Clustered collections)

Cancel

Create Database

Step 17: Now click on ‘ADD DATA’ & select ‘insert document’ here.

The screenshot shows the MongoDB interface with the database 'cluster0.twrdvk...' selected. In the left sidebar, the 'Databases' section lists 'admin', 'college', 'vvp' (which is currently selected and highlighted in green), and 'local'. The main area displays the 'college.vvp' collection. At the top of the collection view, there are tabs for 'Documents', 'Aggregations', 'Schema', 'Indexes', and 'Validation'. Below these tabs is a search bar and a filter dropdown. The main content area contains a table with one document. The first column of the table has a 'ADD DATA' button, which is highlighted with a red box. Underneath this button is a sub-menu with two options: 'Import JSON or CSV file' and 'Insert document', also both highlighted with red boxes. To the right of the table, there is a small icon of a document with a green checkmark. Below the table, a message states 'This collection has no data' and provides instructions: 'It only takes a few seconds to import data from a JSON or CSV file.' At the bottom right of the collection view, there is a green 'Import Data' button.

Step 18: Insert dome document & click on ‘insert’ option.

Insert Document

To collection college.vvp

The screenshot shows the 'Insert Document' dialog box. At the top right are 'VIEW' and '{}' buttons. Below them are two icons: a magnifying glass and a grid. The main area contains a code editor with the following JSON document:

```
1  /**
2  * Paste one or more documents here
3  */
4  {
5  "_id": {
6      "$oid": "654c7aeb9a84cb58920365f5"
7  },
8  "ussn": "488CS123",
9  "firstname": "fsd",
10 "lastname": "cs"
11 }
```

At the bottom right are 'Cancel' and 'Insert' buttons. The 'Insert' button is highlighted with a red box.

Step 19: Your inserted document is being displayed here.

The screenshot shows the MongoDB dashboard. On the left, there's a sidebar with 'My Queries', 'Databases' (with 'vvp' selected), and 'Collections' (with 'college.vvp' selected). The main area shows the 'college.vvp' collection details: 1 DOCUMENTS and 1 INDEXES. Below this, there are tabs for 'Documents', 'Aggregations', 'Schema', 'Indexes', and 'Validation'. A search bar and a filter dropdown are present. The 'Documents' tab shows a table with one row. The row data is highlighted with a red box:

_id: ObjectId('654c7aeb9a84cb58920365f5')	ussn: "488CS123"	firstname: "fsd"	lastname: "cs"
---	------------------	------------------	----------------

Step 20: Click on 'Browse Collections' here.

The screenshot shows the MongoDB Atlas interface. On the left, a sidebar lists 'Project 0' and various services like 'Database', 'Data Lake', 'SERVICES', 'Device Sync', 'Triggers', 'Data API', and 'Data Federation'. The main area is titled 'Database Deployments' for 'DEEPTHIS.ORG - 2023-11-09 > PROJECT 0'. It features a search bar, a 'Load sample datasets to Cluster0' button with a green arrow icon, and a message about sample data. Below these are buttons for 'Edit Config', '+ Create', 'Load sample dataset', and 'Dismiss'. At the bottom, there are tabs for 'Cluster0' (selected), 'Connect', 'View Monitoring', 'Browse Collections' (highlighted with a red box), and '...'. To the right, it says 'FREE' and 'SHARED'.

Step 21: Observe the output as shown below.

The screenshot shows the 'Collections' tab for 'Cluster0'. It displays 'COLLECTIONS: 1' and 'COLL. NAMESPACES: 1'. The 'college' collection is selected. The 'vvp' namespace is also listed. The 'college.vvp' collection details are shown: STORAGE SIZE: 20KB, LOGICAL DATA SIZE: 77B, TOTAL DOCUMENTS: 1, INDEXES: TOTAL SIZE: 20KB. Below are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. An 'INSERT DOCUMENT' button is available. A 'Filter' input field contains 'Type a query: { Field: 'value' }'. The 'QUERY RESULTS: 1-1 OF 1' section shows a single document: '_id: ObjectId('654c7aeb9a64cb58929365f5'), ussn: "488CS123", firstname: "fsd", lastname: "cs"'.

Step 22: Now click on ‘connect’ option here.

The screenshot shows the 'Database Deployments' page for 'Cluster0'. It has the same layout as the previous screenshot, including the sidebar, 'Database Deployments' title, and 'Browse Collections' button. The 'Connect' button is highlighted with a red box at the bottom of the main content area.

Step 23: Select ‘Shell’ option here.

Connect to Cluster0

Set up connection security

Choose a connection method

Connect

Connect to your application



Drivers

Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)



Access your data through tools



Compass

Explore, modify, and visualize your data with MongoDB's GUI



Shell

Quickly add & update data using MongoDB's Javascript command-line interface



MongoDB for VS Code

Work with your data in MongoDB directly from your VS Code environment



Atlas SQL

Easily connect SQL tools to Atlas for data analysis and visualization



[Go Back](#)

[Close](#)

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

Connect to Cluster0

Set up connection security

Choose a connection method

3
Connect

I don't have the MongoDB Shell installed

I have the MongoDB Shell installed

1. Select your mongo shell version

To check your Mongo shell version, run:

mongosh --version or mongo --version

mongosh (2.0 or later) ▾

2. Run your connection string in your command line

Use this connection string in your application

mongosh "mongodb+srv://cluster0.twrdvku.mongodb.net/" --apiVersion 1 --username atlas



You will be prompted for the password for the Database User, **atlas**. When entering your password, make sure all special characters are [URL encoded](#).

RESOURCES

[Add Data in the Shell](#)

[Access your Database Users](#)

[Troubleshoot Connections](#)

[Go Back](#)

[Close](#)

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

C:\Users\vvpcs\Downloads\mongosh-2.0.2-win32-x64\mongosh-2.0.2-win32-x64\bin

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

```
mongosh mongdb+srv//<credentials>@cluster0.twrdvku.mongodb.net/
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vvpcs>cd C:\Users\vvpcs\Downloads\mongosh-2.0.2-win32-x64\mongosh-2.0.2-win32-x64\bin

C:\Users\vvpcs\Downloads\mongosh-2.0.2-win32-x64\mongosh-2.0.2-win32-x64\bin>mongosh "mongodb+srv://cluster0.twrdvku.mongodb.net/" --apiVersion 1 --username atlas
Enter password: *****
Current Mongosh Log ID: 654c7c3853a95ebbe7d954be
Connecting to: mongodb+srv://<credentials>@cluster0.twrdvku.mongodb.net/?appName=mongosh+2.0.2
Using MongoDB: 6.0.11 (API Version 1)
Using Mongosh: 2.0.2

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

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.

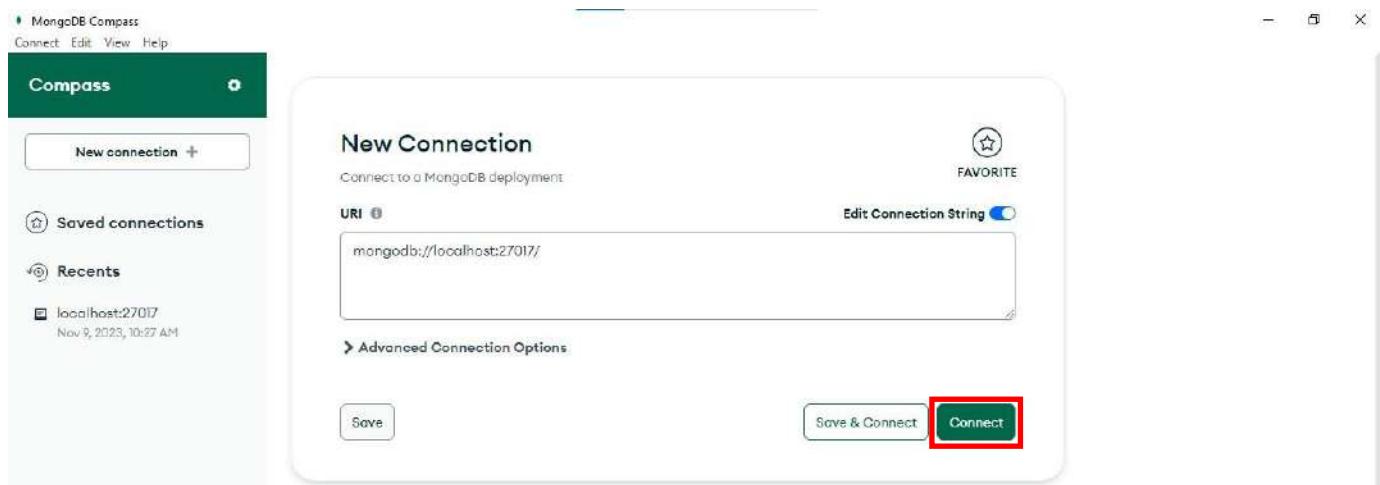
The screenshot shows the MongoDB Atlas interface. At the top, it displays 'DEEPTHIS.ORG - 2023-11-09 > PROJECT 0 > DATABASES'. Below this, there's a cluster overview with 'Cluster0' selected. The 'Collections' tab is active, showing the 'college.vvp' collection. The collection details are: STORAGE SIZE: 20KB, LOGICAL DATA SIZE: 7B, TOTAL DOCUMENTS: 1, INDEXES TOTAL SIZE: 20KB. Below this, there are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. A search bar at the top says 'Type a query: { field: 'value' }'. At the bottom, there are buttons for 'Reset', 'Apply', and 'More Options'. The document '_id: ObjectId("654c7aeb9a84cb58920365f5"), ussn: "488CS123", firstname: "Fsd", lastname: "CS"' is listed under 'QUERY RESULTS: 1-1 OF 1', and it is highlighted with a red box.

5. Aggregation concept in MongoDB.

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: 654c71b14ba60b6bc808df3
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.



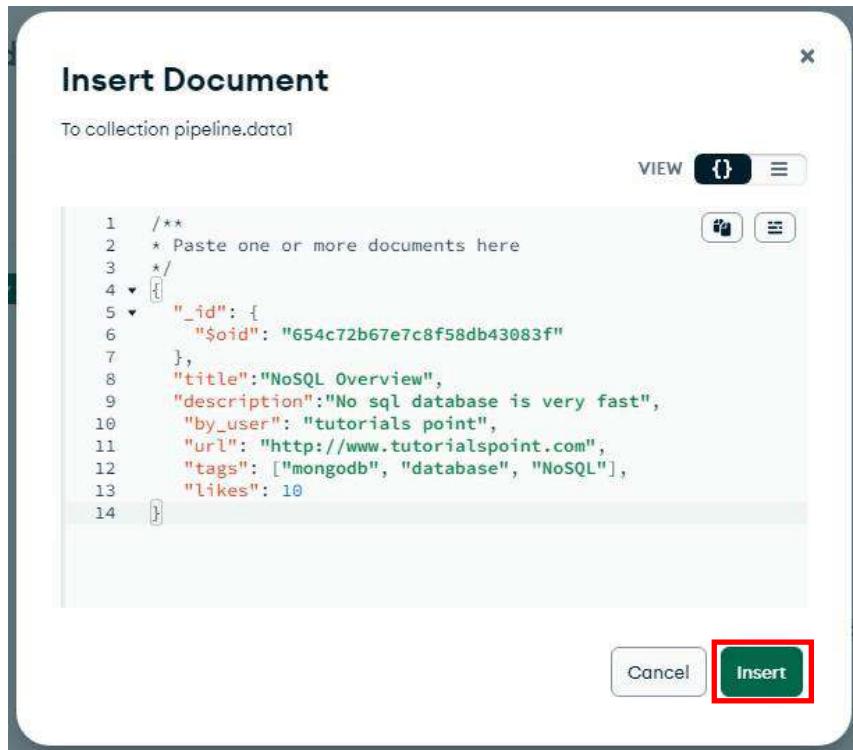
Step 3: Select the created database and collection here.

The screenshot shows the MongoDB Compass interface. At the top, there's a green header bar with the text "localhost:27017" and a leaf icon. Below the header is a sidebar with "My Queries" and "Databases" sections. A search bar is present. The main area displays a tree view of databases and collections. The "pipeline" database is expanded, showing its collections: "data1" and "test". The "data1" collection is highlighted with a red box.

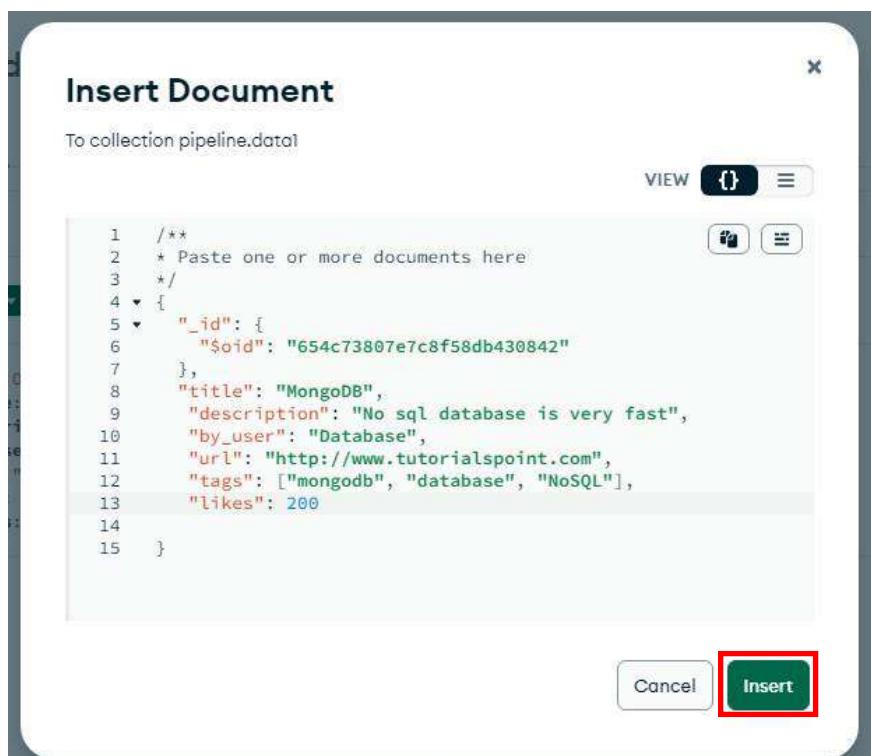
Step 4: Now click on ‘ADD DATA’ & select ‘insert document’ option here.

The screenshot shows the MongoDB Compass interface for the "pipeline.data1" collection. At the top, there's a navigation bar with "Documents", "Aggregations", "Schema", "Indexes", and "Validation". Below the navigation is a search bar and a toolbar with "Explain", "Reset", "Find", and "Options". The main area shows "0 DOCUMENTS" and "1 INDEXES". At the bottom left, there's a "ADD DATA" button with a dropdown menu, and a "Import JSON or CSV file" section containing an "Insert document" button, which is also highlighted with a red box.

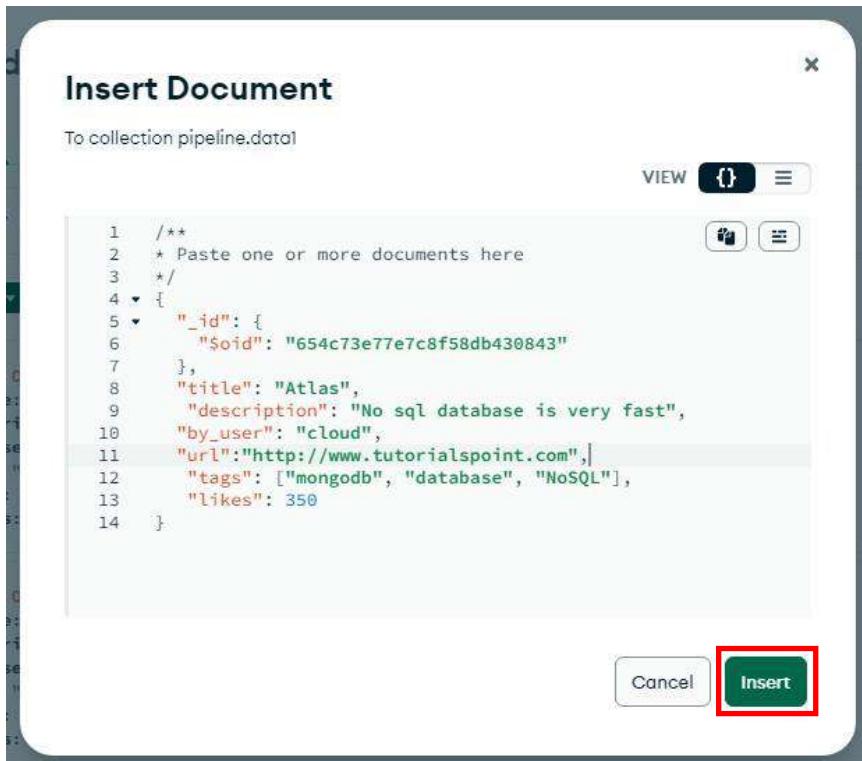
Step 5: Now insert first document & click on ‘Insert’ here.



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.

	_id	title	description	by_user	url
1	ObjectId('654c72b67e7c8f58db4...')	"NosQL Overview"	"No sql database is very fast"	"tutorialspoint"	"http://www.tutorialspoint.com"
2	ObjectId('654c73807e7c8f58db4...')	"MongoDB"	"No sql database is very fast"	"Database"	"http://www.tutorialspoint.com"
3	ObjectId('654c73e77e7c8f58db4...')	"Atlas"	"No sql database is very fast"	"cloud"	"http://www.tutorialspoint.com"

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

pipeline.data1

3 DOCUMENTS 1 INDEXES

Documents **Aggregations** Schema Indexes Validation

Pipeline Your pipeline is currently empty. Need help getting started? [Generate aggregation](#) Explain Export Run More Options

Untitled [SAVE](#) [CREATE NEW](#) [EXPORT TO LANGUAGE](#) PREVIEW STAGES TEXT

0 Documents in the collection

Preview of documents

+ Add Stage Learn more about aggregation pipeline stages

Step 10: Now select ‘limit’ aggregate operation here.

MongoDB Compass - localhost:27017/pipeline.data1

Connect Edit View Collection Help

localhost:27017 Aggregations pipeline.data1

My Queries Databases pipeline

Search

Documents **Aggregations** Schema Indexes Validation

Pipeline Your pipeline is currently empty. [Generate aggregation](#) Explain Export Run More Options

Untitled - modified [SAVE](#) [CREATE NEW](#) [EXPORT TO LANGUAGE](#) PREVIEW STAGES TEXT

Stage1 Select

1 \$indexStats Returns statistics regarding the use of each index for the collection.

\$limit Limits the number of documents that flow into subsequent stages.

\$lookup Performs a join between two collections.

\$match Filters the document stream to allow only matching documents to pass through to subsequent stages.

\$merge Merges the resulting documents into a collection, optionally overriding existing documents.

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

The screenshot shows the MongoDB aggregation pipeline interface. Stage 1 is named '\$limit'. The input stage has the code:

```
1 //**  
2   + Provide the number of documents to limit.  
3  
4 3
```

 The value '3' is highlighted with a red box. The output stage shows three sample documents:

```
_id: ObjectId('654c72b67e7c8f58db4...  
title: "NoSQL Overview"  
description: "No sql database is very fast"  
by_user: "tutorials point"  
url: "http://www.tutorialspoint.co...  
tags: Array (3)  
likes: 10  
  
_id: ObjectId('654c73807e7c8f58db4...  
title: "MongoDB"  
description: "No sql database is very fast"  
by_user: "Database"  
url: "http://www.tutorialspoint.co...  
tags: Array (3)  
likes: 200  
  
_id: ObjectId('654c73e77e7c8f58db4...  
title: "Atlas"  
description: "No sql database is very fast"  
by_user: "cloud"  
url: "http://www.tutorialspoint.co...  
tags: Array (3)  
likes: 350
```

Step 12: Select ‘Sort’ aggregate operation & enter the field for sorting here.

The screenshot shows the MongoDB aggregation pipeline interface. Stage 2 is named '\$sort'. The input stage has the code:

```
1 //**  
2   + Provide any number of field/order pairs.  
3  
4 {  
5   title:1  
6 }
```

 The field 'title:1' is highlighted with a red box. The output stage shows three sample documents:

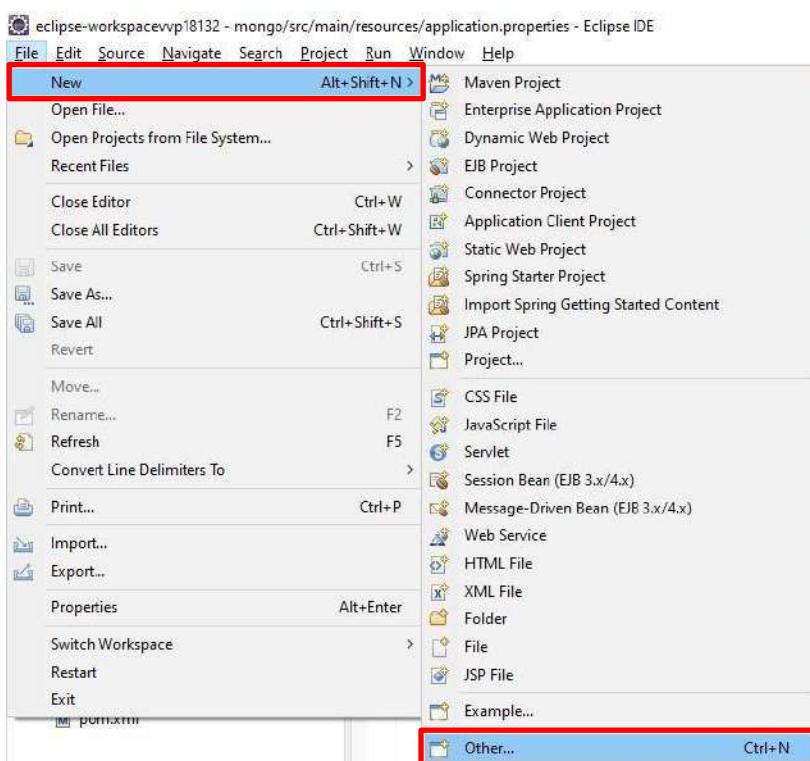
```
_id: ObjectId('654c73e77e7c8f58db4...  
title: "Atlas"  
description: "No sql database is very fast"  
by_user: "cloud"  
url: "http://www.tutorialspoint.co...  
tags: Array (3)  
likes: 350  
  
_id: ObjectId('654c73807e7c8f58db4...  
title: "MongoDB"  
description: "No sql database is very fast"  
by_user: "Database"  
url: "http://www.tutorialspoint.co...  
tags: Array (3)  
likes: 200  
  
_id: ObjectId('654c72b67e7c8f58db4...  
title: "NoSQL Overview"  
description: "No sql database is very fast"  
by_user: "tutorials point"  
url: "http://www.tutorialspoint.co...  
tags: Array (3)  
likes: 10
```

Step 13: Select ‘Count’ aggregate operation & enter the field for counting here.

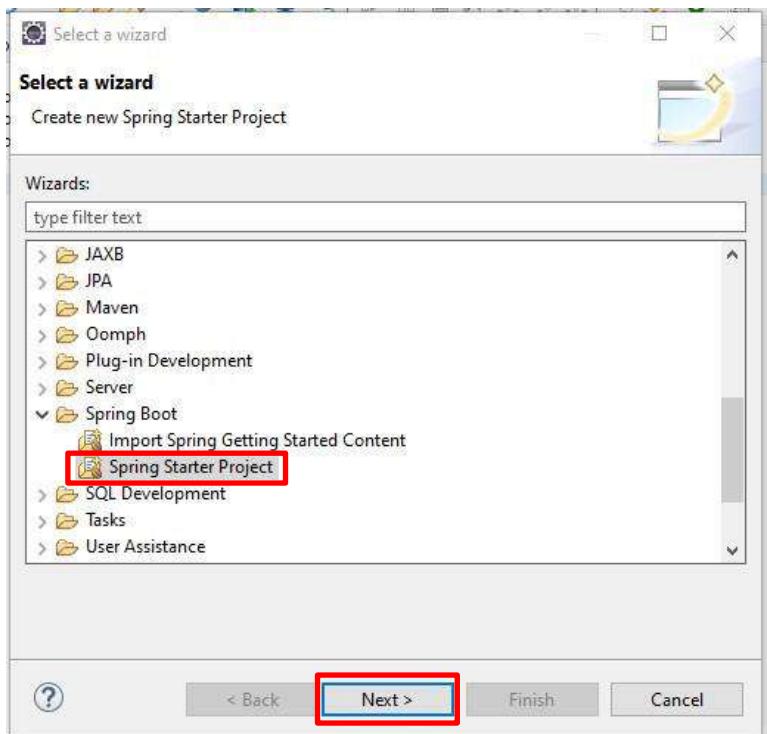
The screenshot shows the MongoDB Compass interface. At the top, there's a header with buttons for 'Stage 3: \$count', 'ENABLED' (with a toggle switch), and 'ADD STAGE'. Below this, the 'STAGE INPUT' section shows 'Sample of 3 documents' with three document snippets. The 'STAGE OUTPUT' section shows 'Sample of 1 document' with the result 'likes: 3'. A red box highlights the input field 'Scount' in the stage configuration.

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

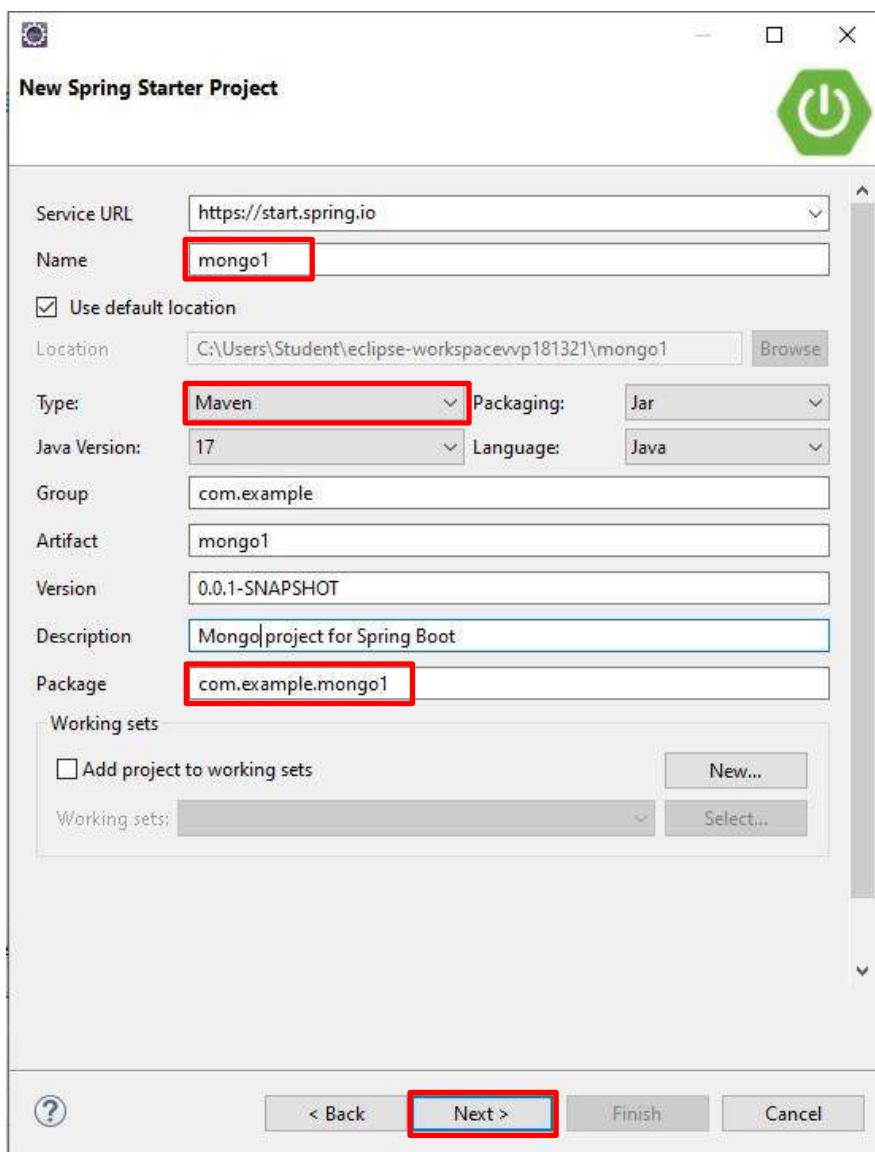
Step 1: Open the Eclipse IDE for Enterprise Edition New Other.



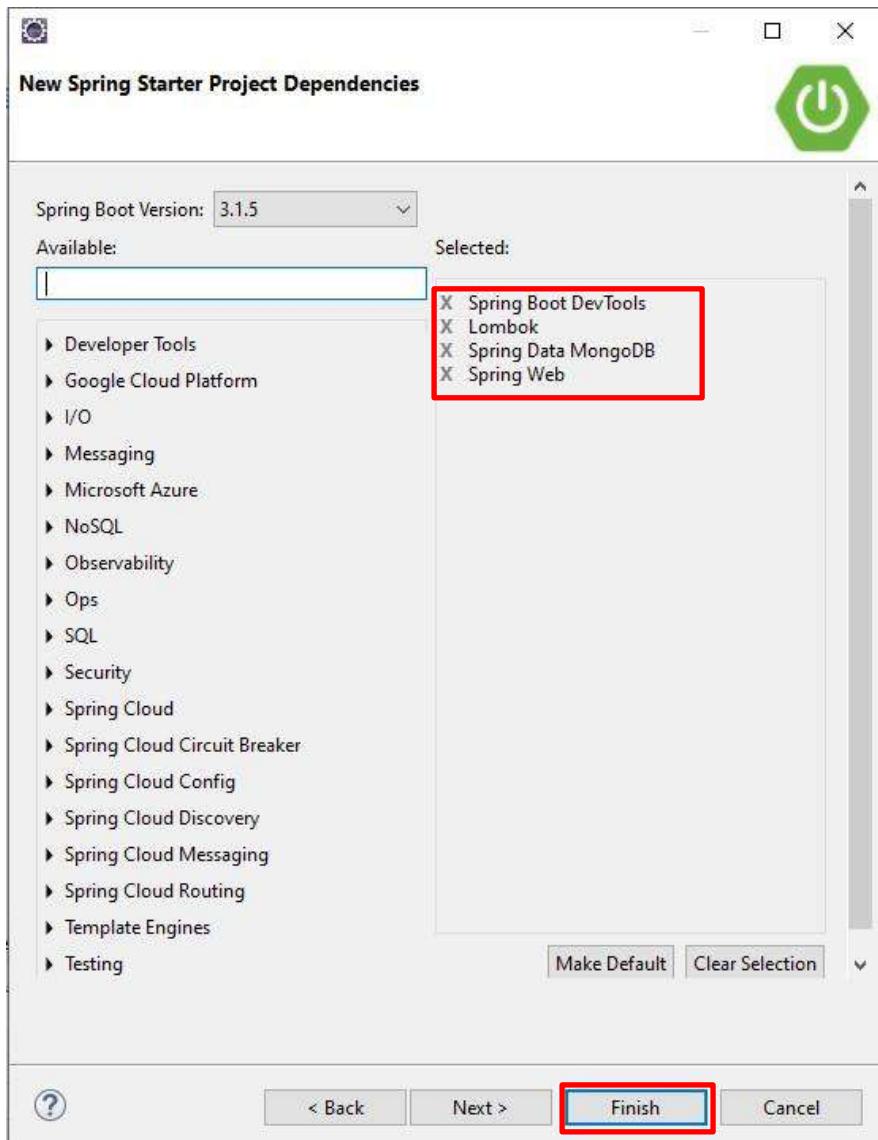
Step 2: Select ‘Spring starter project’ & click on ‘Next’.



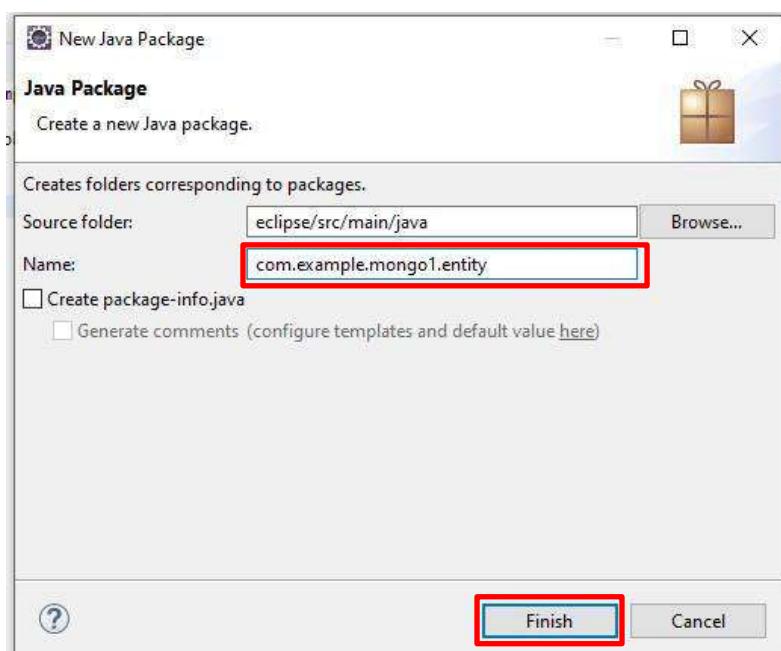
Step 3: Select the options as shown below & click on 'Next'.



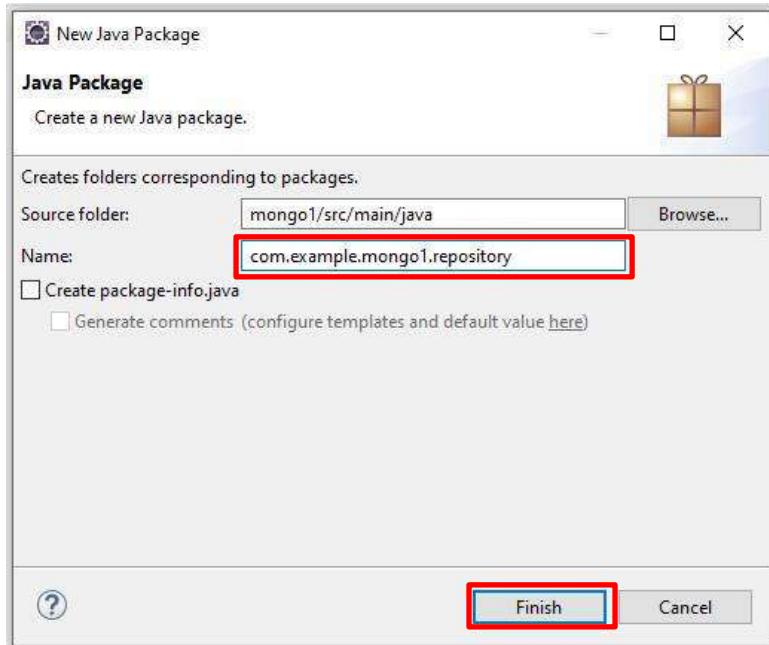
Step 4: Add 4 below mentioned dependencies & click on ‘Finish’.



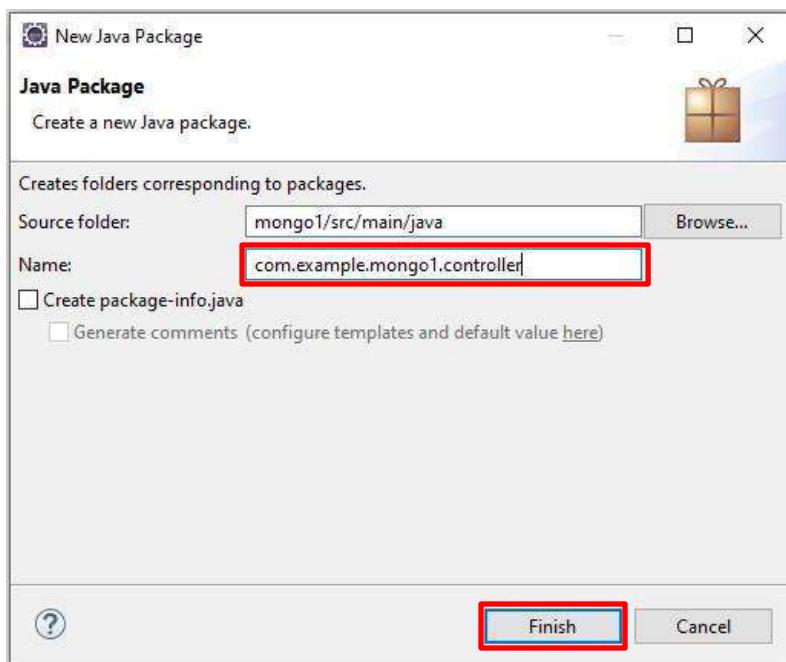
Step 5: Now create a new package named ‘com.example.mongo1.entity’ & click on ‘Finish’.



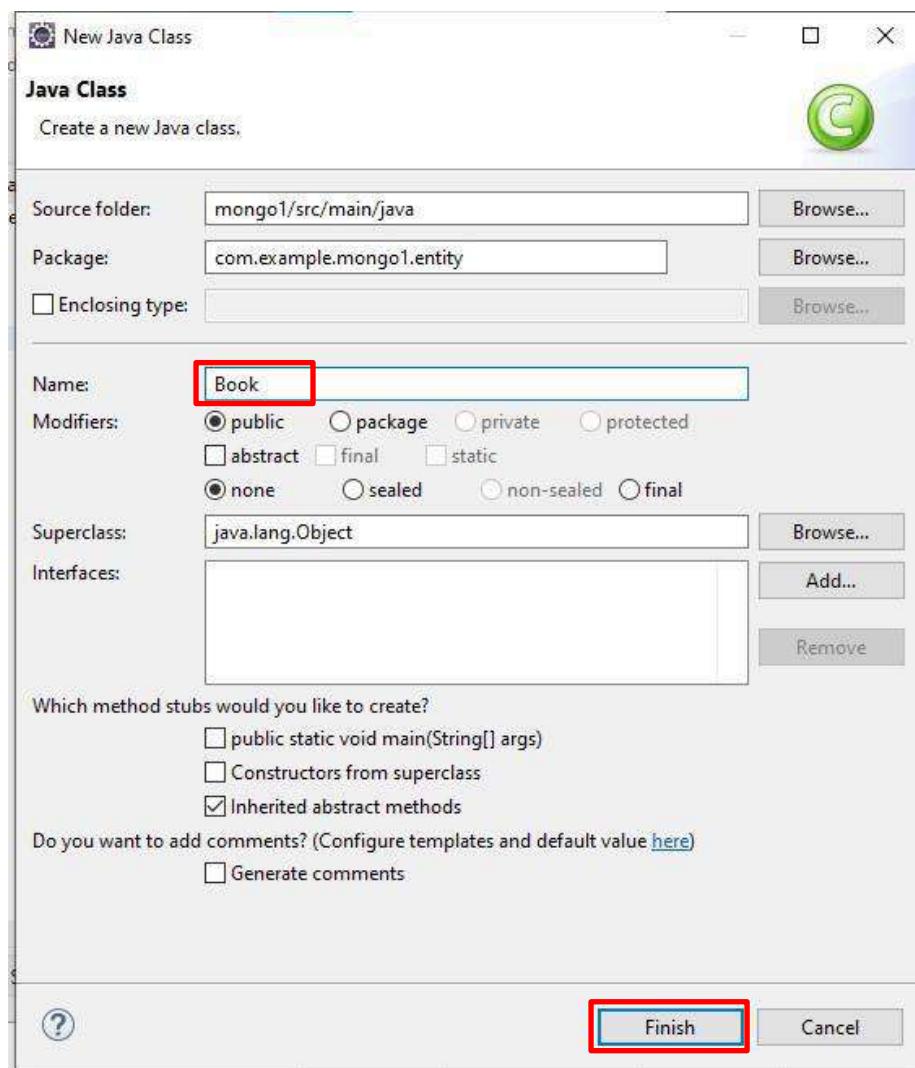
Step 6: Now create a new package named '**com.example.mongo1.repository**' & click on '**Finish**'.



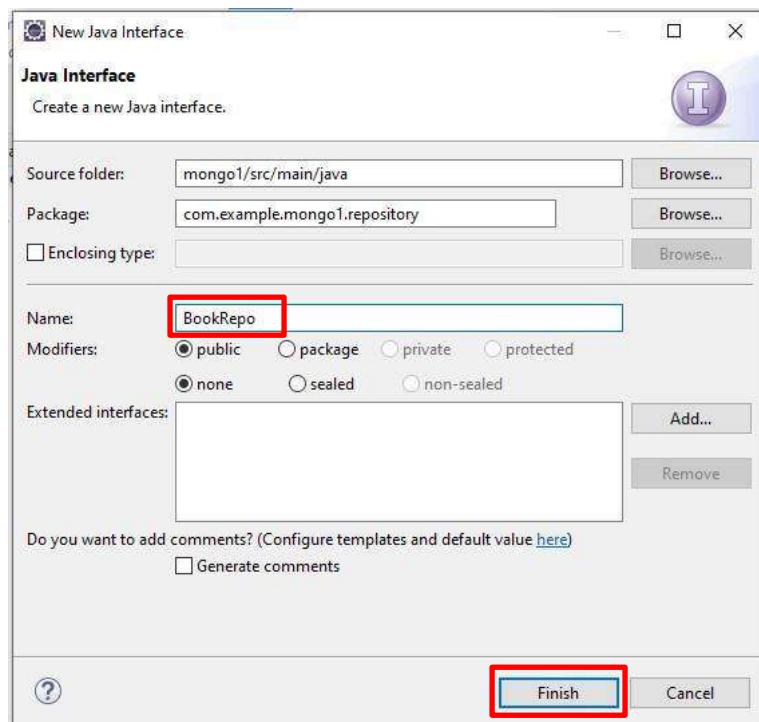
Step 7: Now create a new package named '**com.example.mongo1.controller**' & click on '**Finish**'.



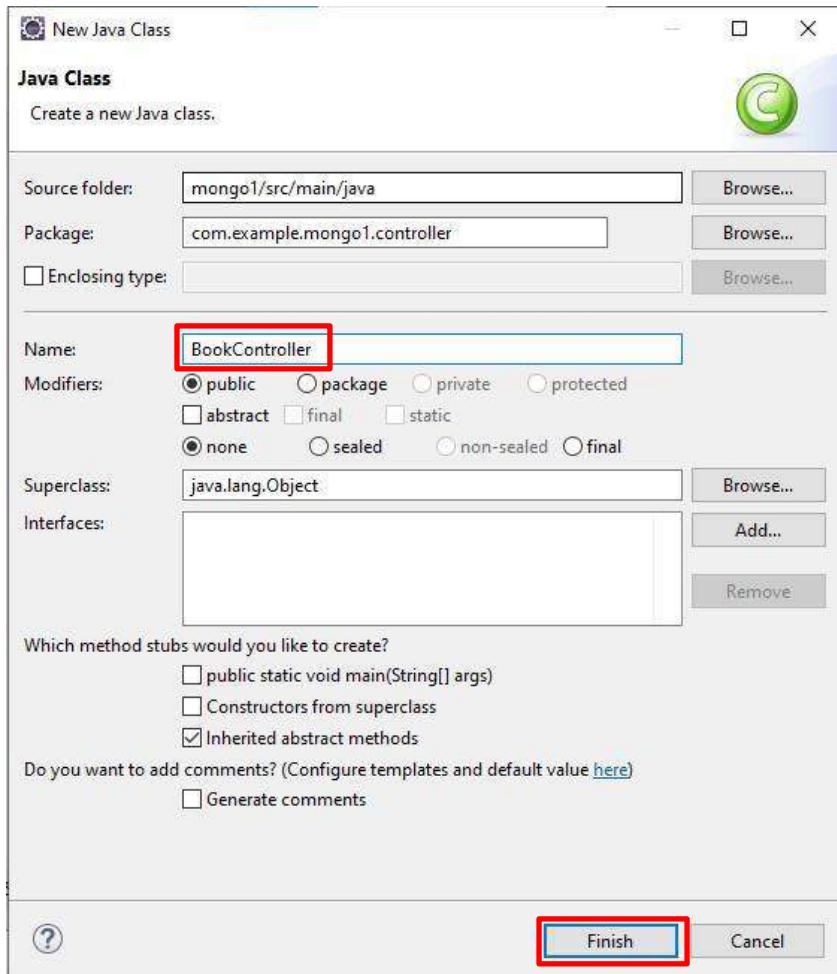
Step 8: Now create a new class named '**Book**' for the package named '**com.example.mongo1.entity**' & click on '**Finish**'.



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.

```
1 package com.example.mongo1.entity;
2 import lombok.AllArgsConstructorConstructor;
3 import lombok.Data;
4 import lombok.NoArgsConstructor;
5 import org.springframework.data.annotation.Id;
6 import org.springframework.data.mongodb.core.mapping.Document;
7 @Data
8 @NoArgsConstructor
9 @AllArgsConstructor
10 @Document(collection = "Book")
11 public class Book {
12     @Id
13     private int id;
14     private String bookName;
15     private String authorName;
16     public int getId() {
17         return id;
18     }
19     public void setId(int id) {
20         this.id = id;
21     }
22     public String getBookName() {
23         return bookName;
24     }
25     public void setBookName(String bookName) {
26         this.bookName = bookName;
27     }
28     public String getAuthorName() {
29         return authorName;
30     }
31     public void setAuthorName(String authorName) {
32         this.authorName = authorName;
33     }
34 }
35 }
```

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

```

1 package com.example.mongo1.controller;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.web.bind.annotation.DeleteMapping;
7 import org.springframework.web.bind.annotation.GetMapping;
8 import org.springframework.web.bind.annotation.PathVariable;
9 import org.springframework.web.bind.annotation.PostMapping;
10 import org.springframework.web.bind.annotation.PutMapping;
11 import org.springframework.web.bind.annotation.RequestBody;
12 import org.springframework.web.bind.annotation.RestController;
13
14 import com.example.mongo1.entity.Book;
15 import com.example.mongo1.repository.BookRepo;
16
17 @RestController
18 public class BookController {
19     @Autowired
20     private BookRepo repo;
21
22     @PostMapping("/addBook")
23     public String saveBook(@RequestBody Book book){
24         repo.save(book);
25         return "Added Successfully";
26     }
27     @GetMapping("/findAllBooks")
28     public List<Book> getBooks() {
29         return repo.findAll();
30     }
31     @PutMapping("/update/{id}")
32     public String updatebook(@RequestBody Book book,@PathVariable("id") Integer bookId)
33     {
34         Book ex=this.repo.findById(bookId).orElseThrow();
35         ex.setBookName(book.getBookName());
36         ex.setAuthorName(book.getAuthorName());
37         this.repo.save(ex);
38         return "Updated Sucessfully";
39     }
40     @DeleteMapping("/delete/{id}")
41     public String deleteBook(@PathVariable int id){
42         repo.deleteById(id);
43         return "Deleted Successfully";
44     }
45 }
46

```

Step 13: Now add code to ‘BookRepo.java’ file.

```

1 package com.example.mongo1.repository;
2 import org.springframework.data.mongodb.repository.MongoRepository;
3 import com.example.mongo1.entity.Book;
4 public interface BookRepo
5     extends MongoRepository<Book, Integer> {
6 }
7

```

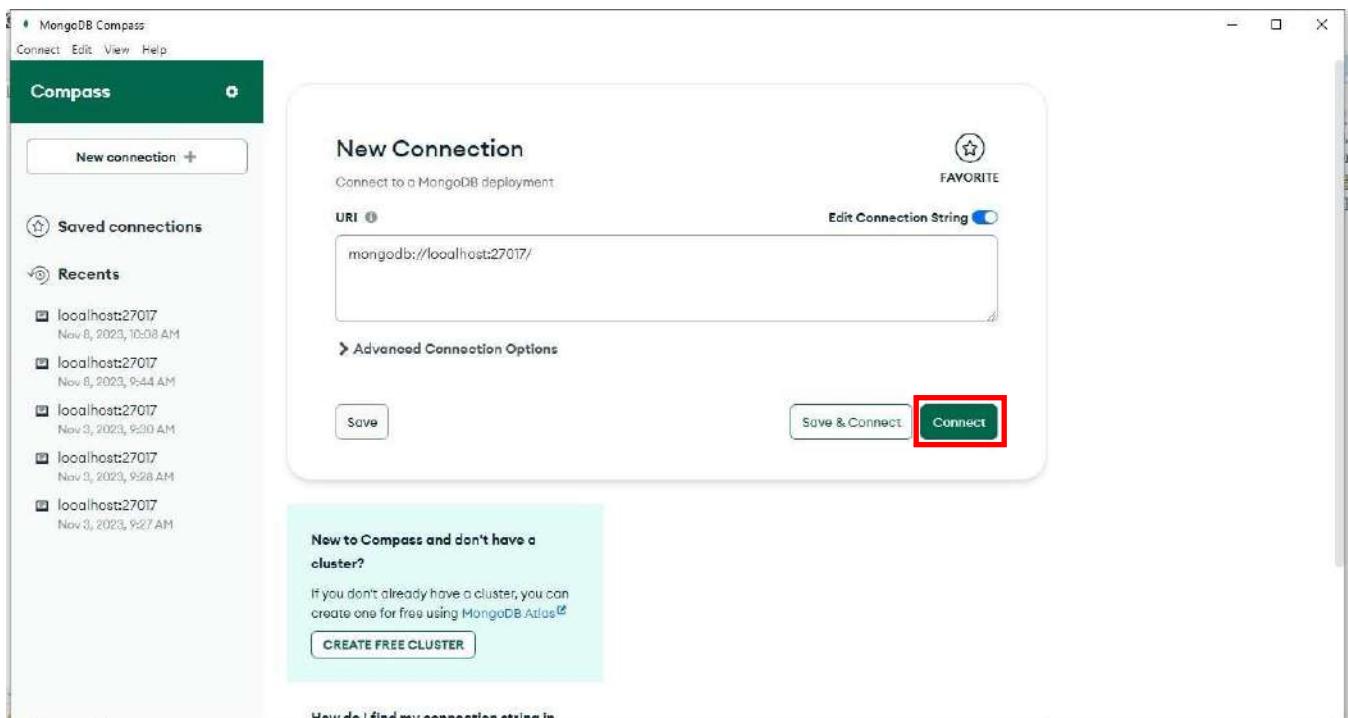
Step 14: Now add code to ‘application.properties’ file.

```

1 server.port=8080
2 spring.data.mongodb.host=localhost
3 spring.data.mongodb.port=27017
4 spring.data.mongodb.database=BookStore
5
6
7

```

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

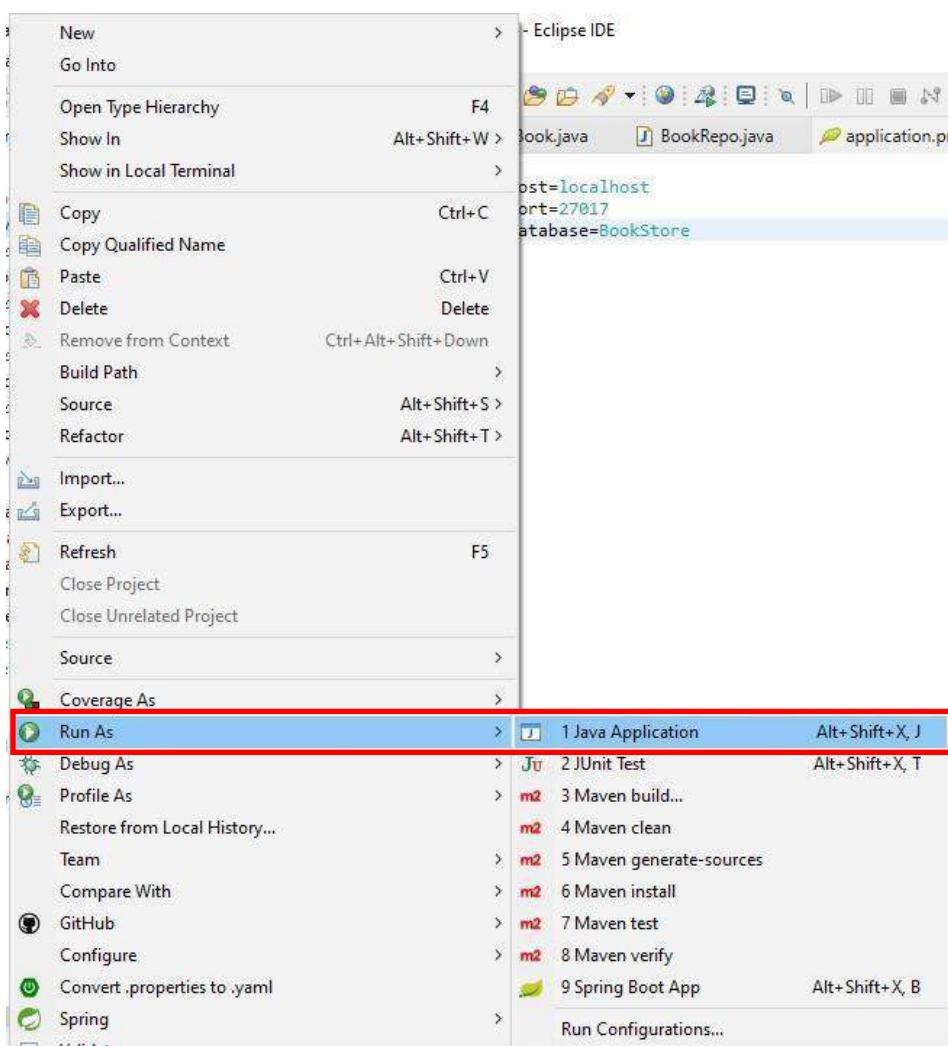
For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----
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
dmns 48.00 KiB
desktop 72.00 KiB
dlt 48.00 KiB
fsd 184.00 KiB
local 72.00 KiB
network 88.00 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.

The screenshot shows the Postman interface with the 'mongodb' collection selected. The left sidebar shows 'My Workspace' with 'Collections' (selected), 'Environments', and 'History'. The main area displays the 'Overview' tab for the 'mongodb' collection, which is currently empty. A red box highlights the word 'mongodb' in the title bar. Below it, there's a note: 'This collection is empty. Add a request to start working.' A 'View complete documentation →' link is also present.

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

The screenshot shows the Postman interface with a specific request configuration. The left sidebar shows 'My Workspace' with 'Collections' (selected). The main area shows a request for 'localhost:8080/findAllBooks' using the 'GET' method. A red box highlights the 'Send' button. The 'Body' tab is selected, showing a JSON response with one item: [{}].

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

The screenshot shows the Postman interface with a red box highlighting the request body. The request method is POST, and the URL is localhost:8080/addBook. The body contains the following JSON:

```
1: { "id": "1", "bookName": "os", "authorName": "fed" }
```

The response status is 200 OK, and the message is "Added Successfully".

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

The screenshot shows the Postman interface with a red box highlighting the request body. The request method is POST, and the URL is localhost:8080/addBook. The body contains the following JSON:

```
1: { "id": "2", "bookName": "vivo", "authorName": "java" }
```

The response status is 200 OK, and the message is "Added Successfully".

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

The screenshot shows the Postman interface. On the left, there's a sidebar with 'My Workspace' containing a 'Collections' section with a 'mongodb' entry. The main area has tabs for 'Overview', 'GET localhost:8080/findAllBooks', and 'POST localhost:8080/addBook'. The 'POST' tab is selected. Below it, the 'Body' tab is active, showing a JSON payload:

```

1 {
2   "id": "3",
3   "bookName": "python",
4   "authorName": "c_bash"
5 }

```

Below the body, the response status is 'Status: 200 OK' with a 'Time: 17 ms' and a 'Size: 182 B'. The response body says 'Added Successfully'.

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

```

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
2023-11-06T09:10:30.752+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 ~/mongosrc.js. ~/mongorc.js will not be loaded.
You may want to copy or rename ~/mongorc.js to ~/mongosrc.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
dmns 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.

The screenshot shows the Postman interface with a red box highlighting the request body. The URL is `localhost:8080/update/2`. The request method is `PUT`. The body is set to `JSON` and contains the following JSON:

```
1 {  
2   "id": "2",  
3   "bookName": "sepp",  
4   "authorName": "c++"  
5 }
```

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.

The screenshot shows the Postman interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'Collections', 'Environments', and 'History'. The main area shows a 'mongodb' collection with several requests: 'GET localhost:8080/findAllBooks', 'POST localhost:8080/addBook', 'PUT localhost:8080/update/2', and 'DELETE localhost:8080/delete/2'. The 'DELETE' request is highlighted with a red box. The request details show a 'DELETE' method and the URL 'localhost:8080/delete/2'. Below it, the 'Params' tab is selected, showing a table with one row: 'Key' (Value) and 'Description' (Description). The response tab shows a status of '200 OK' with a message 'Deleted Successfully'.

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

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } or [Generate query](#)

2 DOCUMENTS 1 INDEXES

ADD DATA EXPORT DATA Explain Reset Find Options

1 - 2 of 2

`_id: 1
bookName: "cs"
authorName: "fad"
_class: "com.example.mongo1.entity.Book"`

`_id: 3
bookName: "python"
authorName: "c_bash"
_class: "com.example.mongo1.entity.Book"`

... {}

>_MONGOSH

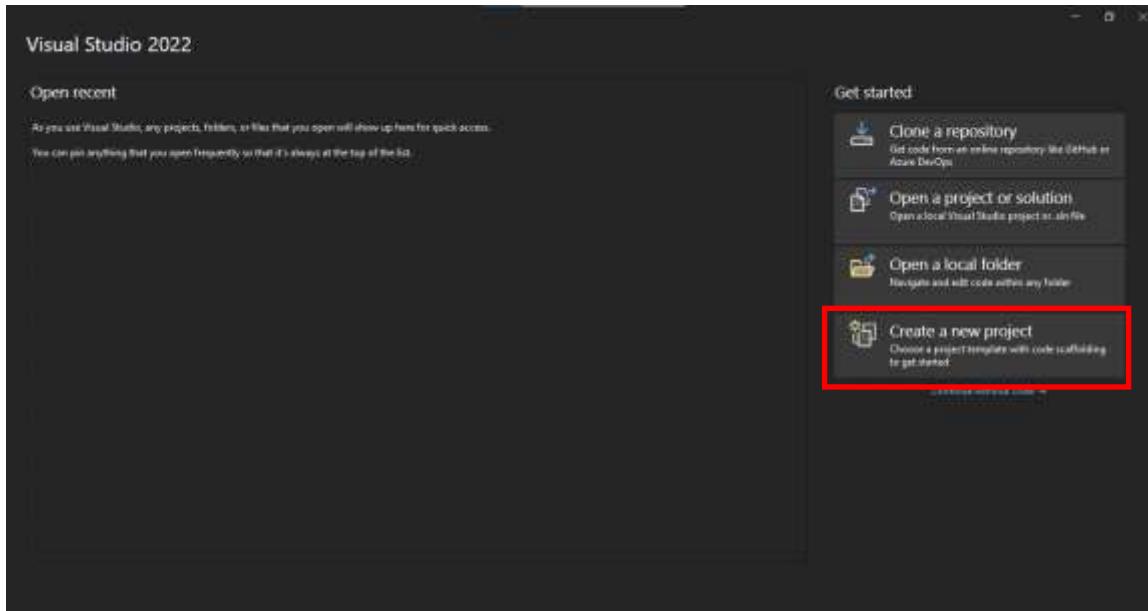
```
_id: 1
bookName: "cs"
authorName: "fad"
_class: "com.example.mongo1.entity.Book"

_id: 3
bookName: "python"
authorName: "c_bash"
_class: "com.example.mongo1.entity.Book"
```

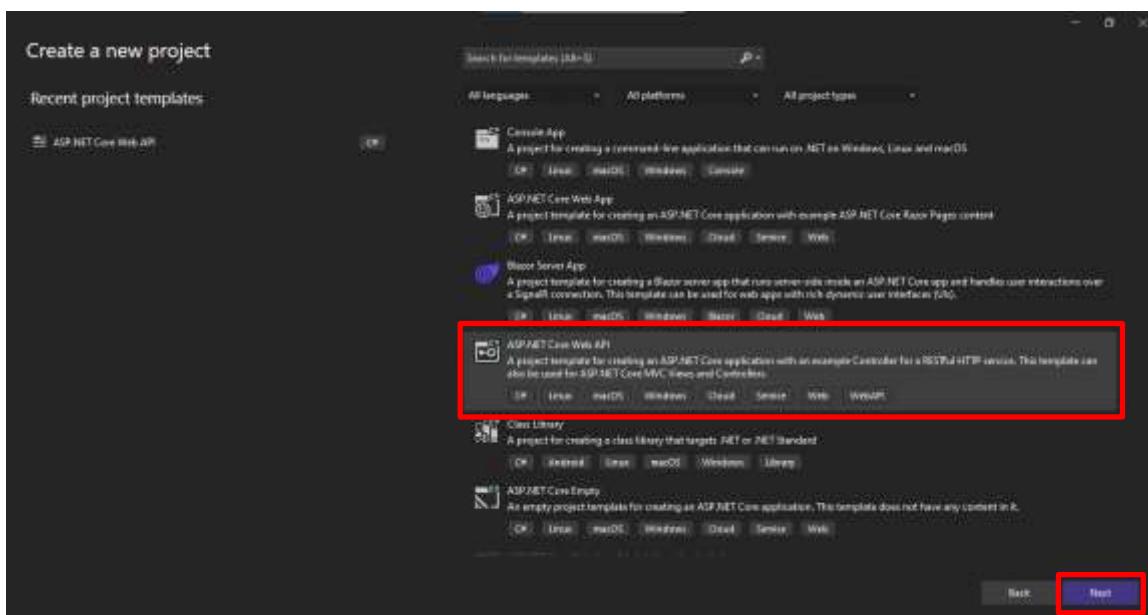
Week-11

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

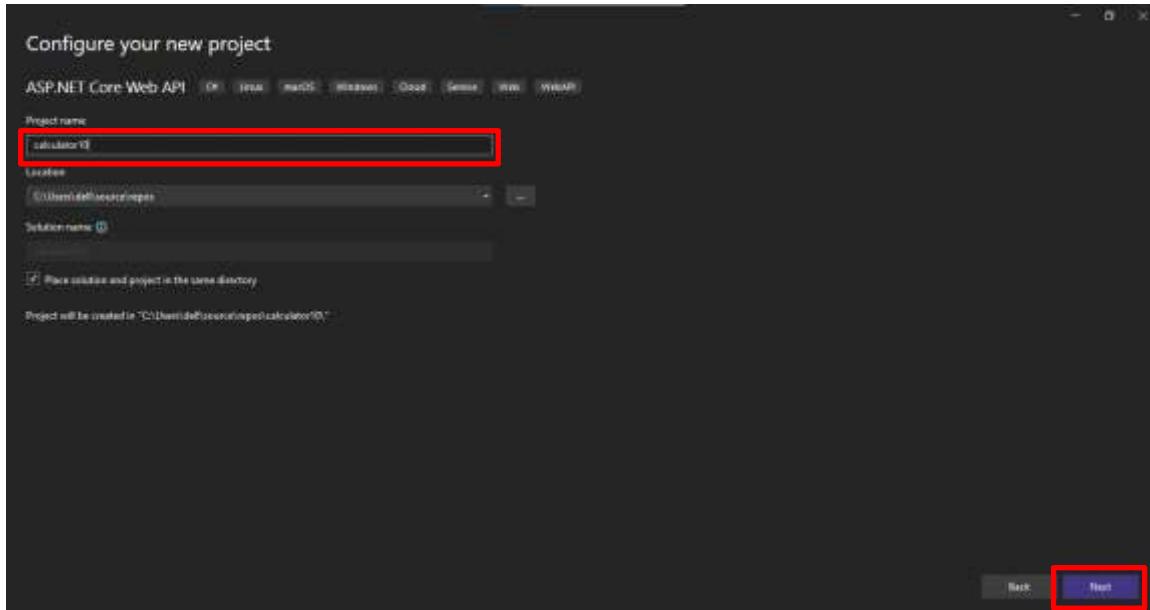
Step 1: Open Microsoft Visual Studios And click on ‘Create a new Project’.



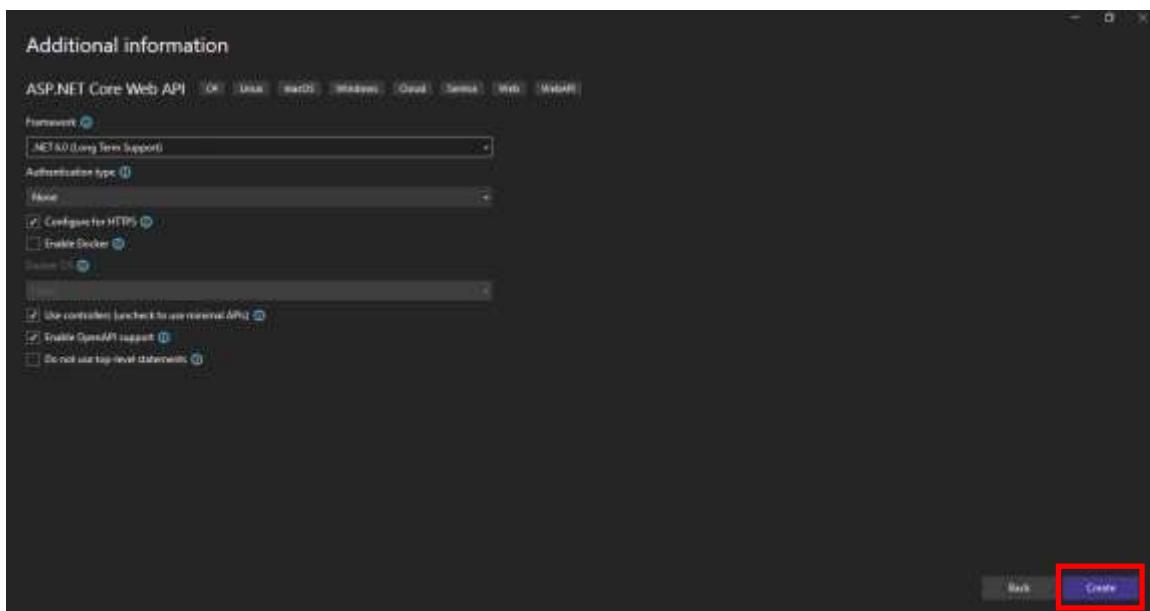
Step 2: Select ‘ASP.NETCORE Web API’ & click on ‘Next’.



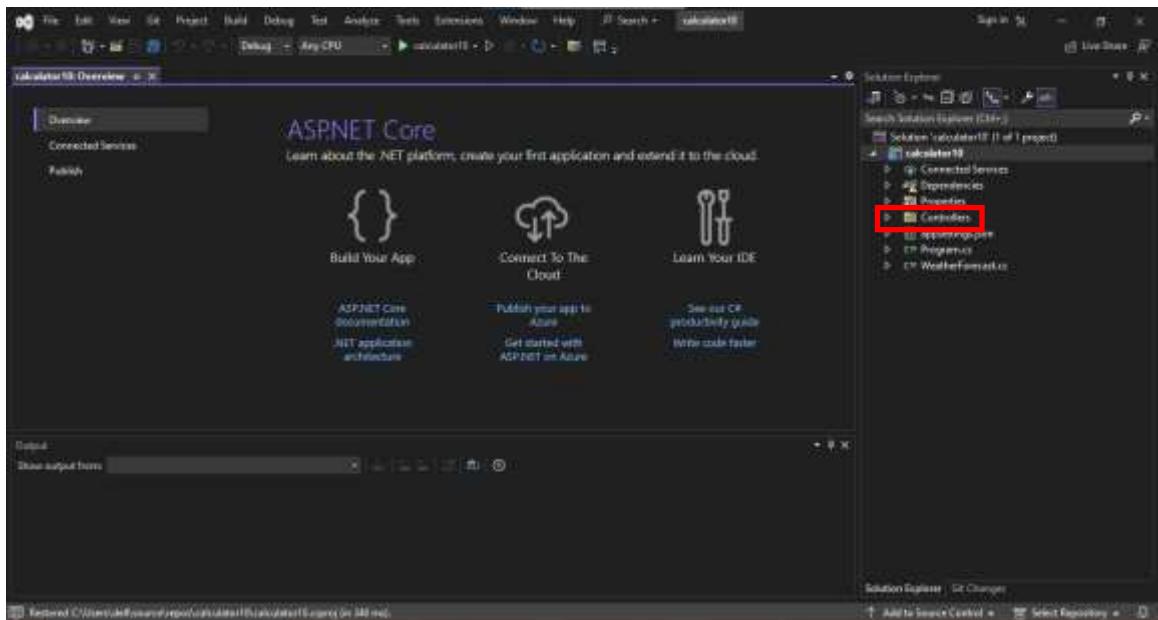
Step 3: Give a Project name & click on ‘Next’.



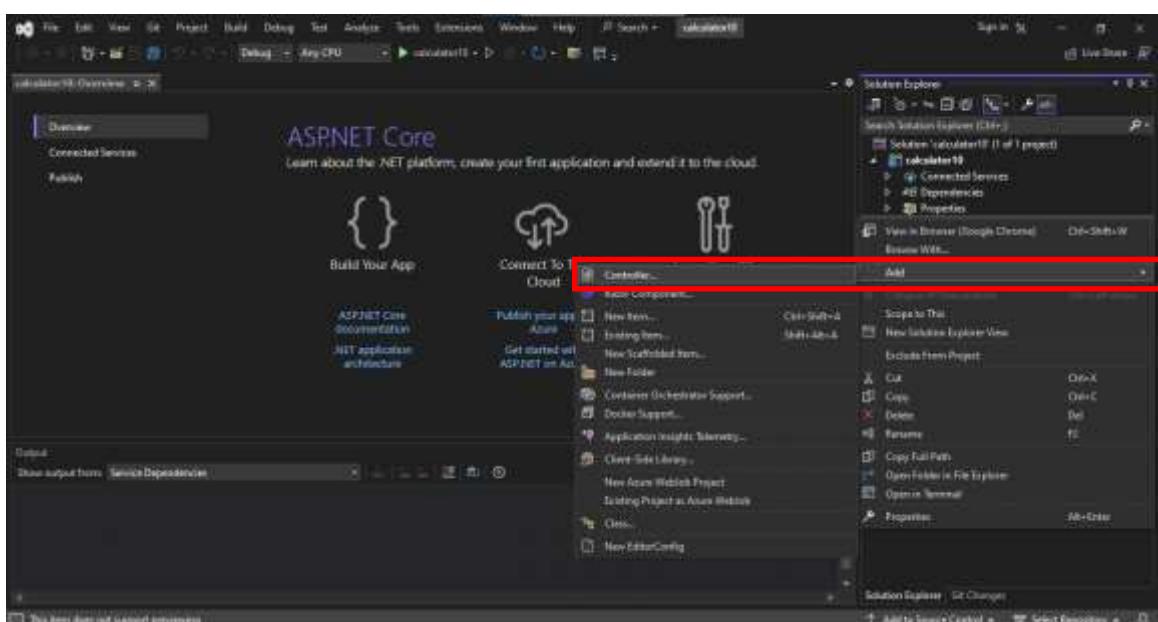
Step 4: Disable the Docker if you don't Docker Desktop installed & Click on 'Create'.



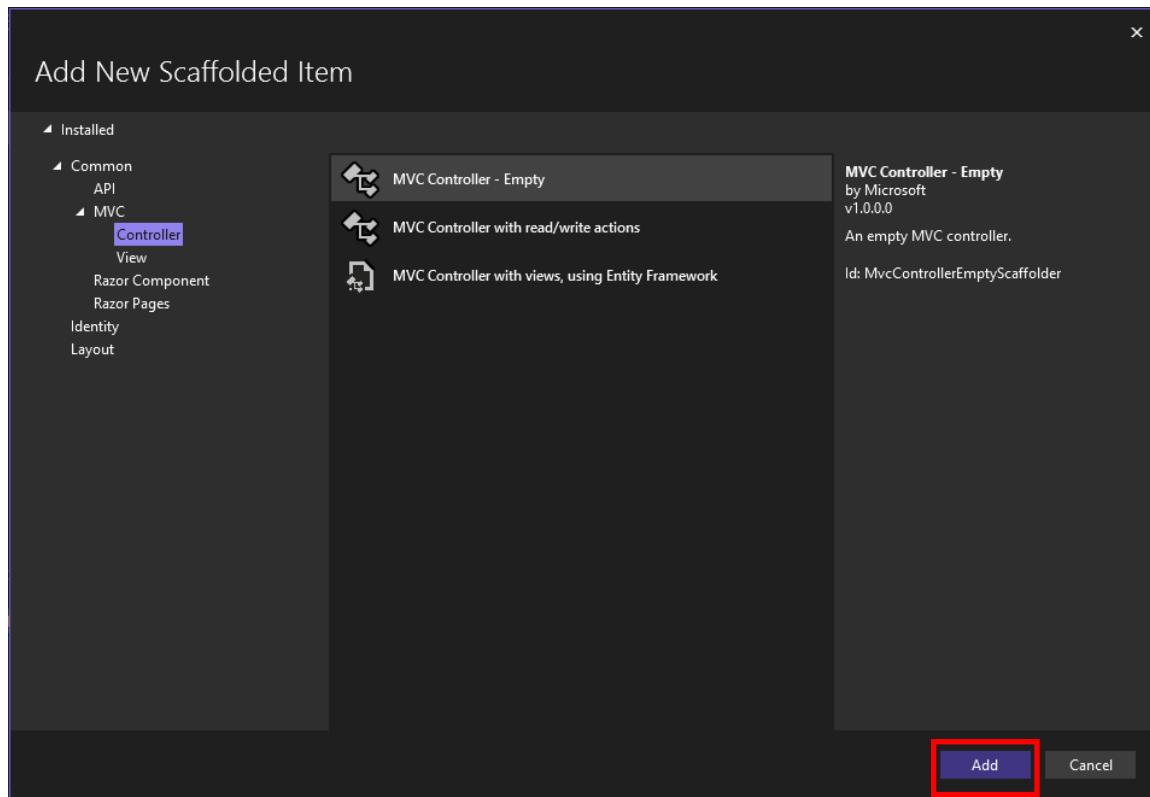
Step 5: Now right click on '**controllers**' here.



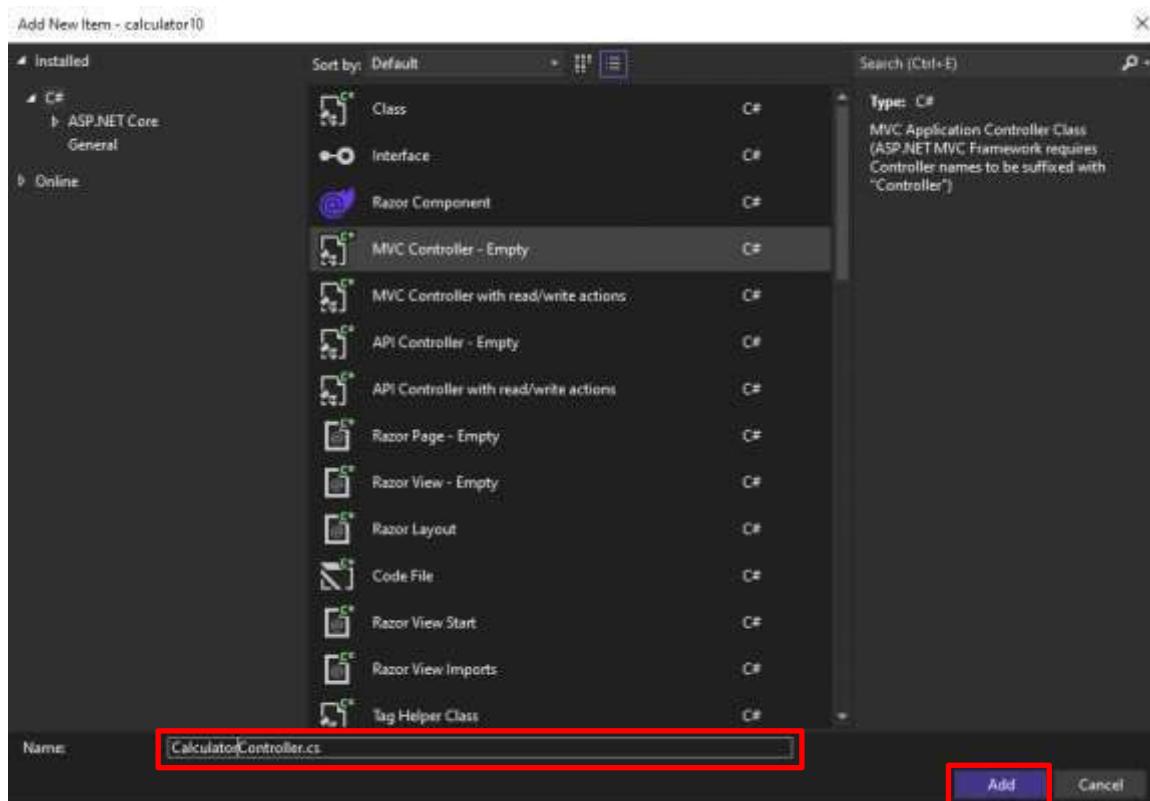
Step 6: Select **Add** & click on ‘**Controllers**’.



Step 7: Click on **Add**.



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



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

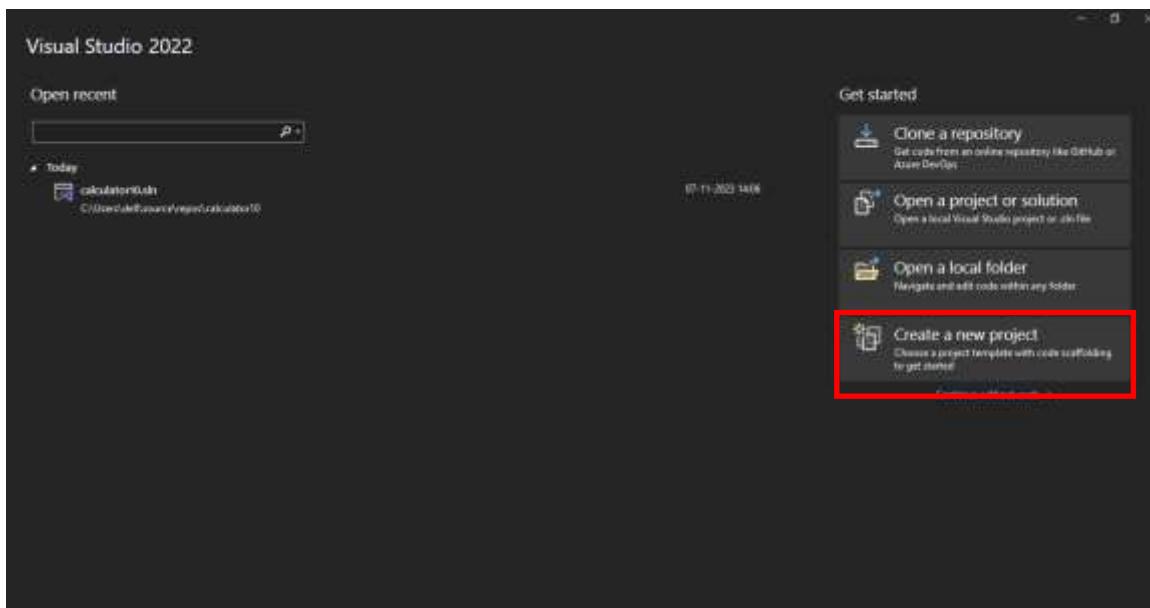
A screenshot of Microsoft Visual Studio showing the code editor and Solution Explorer. The code editor displays a C# file named 'CalculatorController.cs' with the following content:

```
calculatorController.cs
using Microsoft.AspNetCore.Mvc;
namespace calculator.Controllers
{
    [ApiController]
    [Route("calculator")]
    public class CalculatorController : ControllerBase
    {
        private static readonly int number1 = 10;
        private static readonly int number2 = 5;
        private readonly ILogger<CalculatorController> logger;

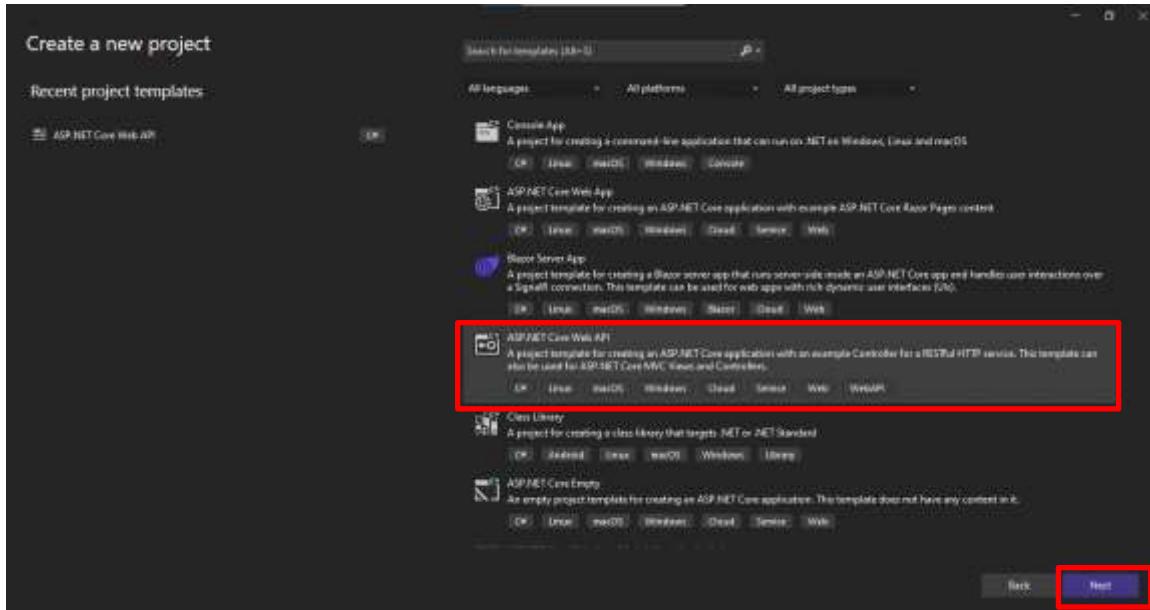
        public CalculatorController(ILogger<CalculatorController> logger)
        {
            this.logger = logger;
        }
    }
}
```

The Solution Explorer shows a single project named 'calculator10' with a 'Controllers' folder containing the 'CalculatorController.cs' file.

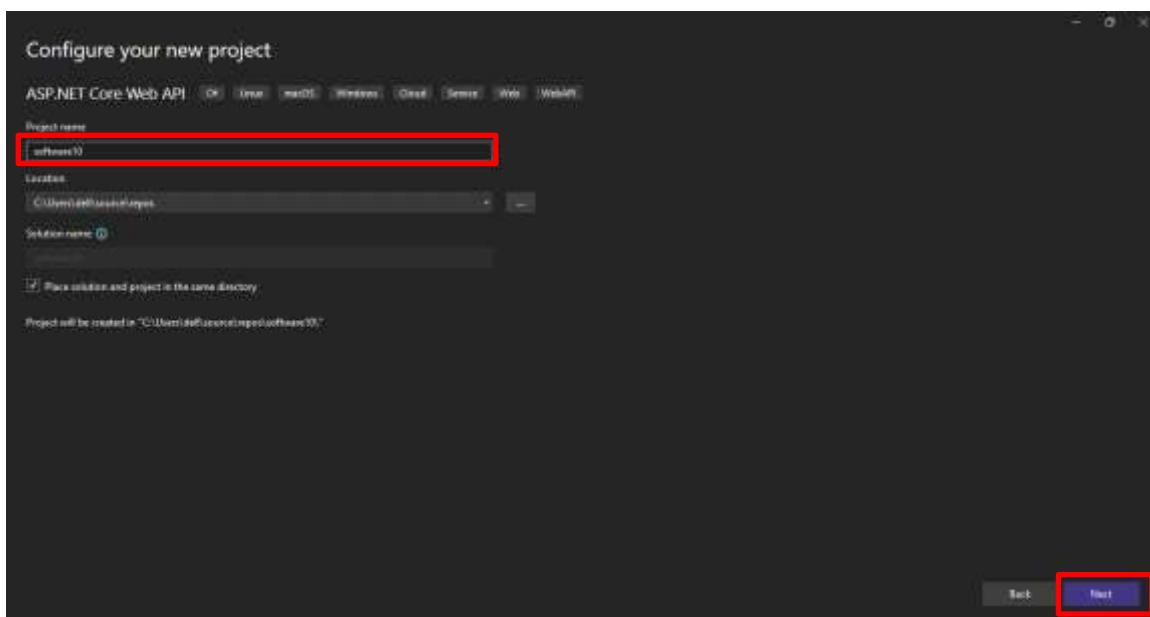
Step 10: Now close the other windows and open the Microsoft Visual Studios one more time & click on ‘Create a new Project’.



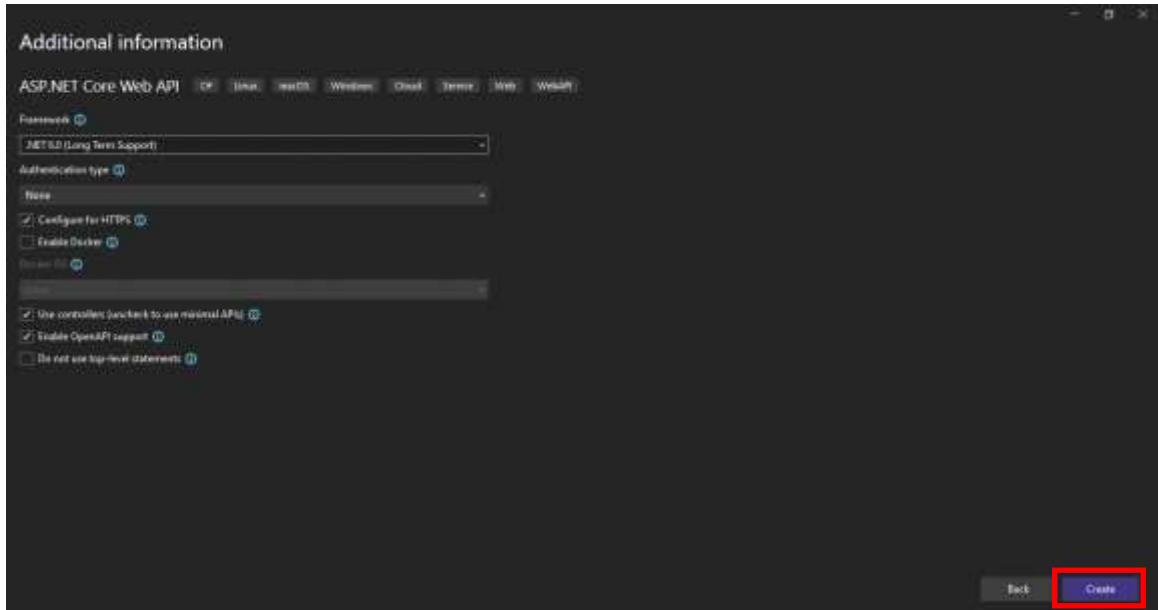
Step 11: Select ‘ASP.NETCORE Web API’ & click on ‘Next’.



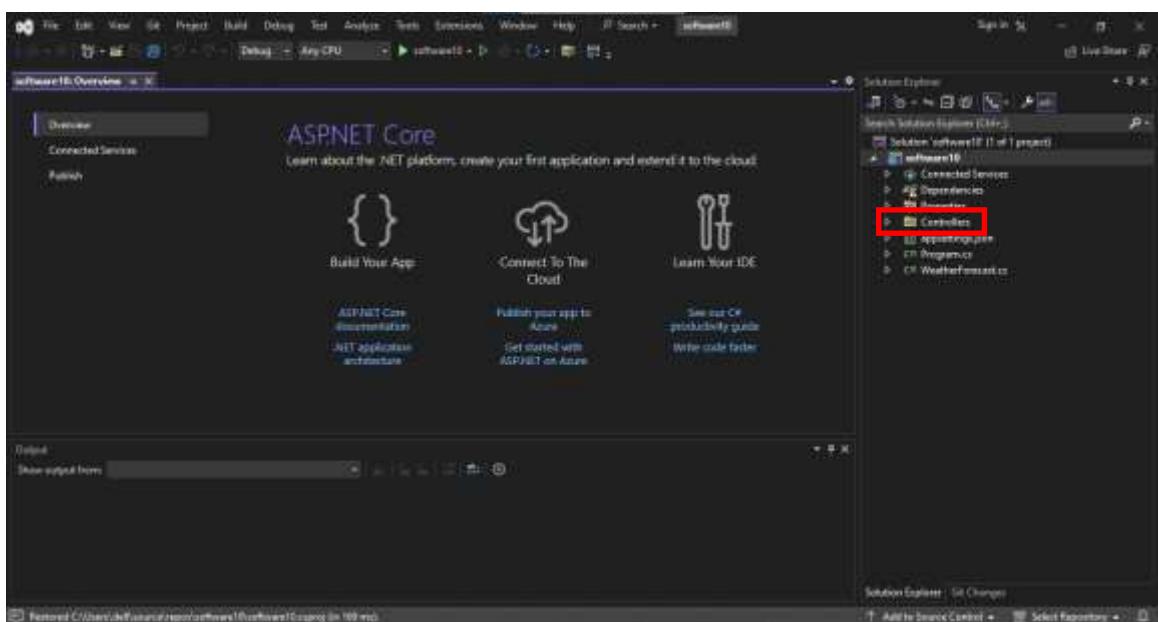
Step 12: Give a Project name & click on ‘Next’.



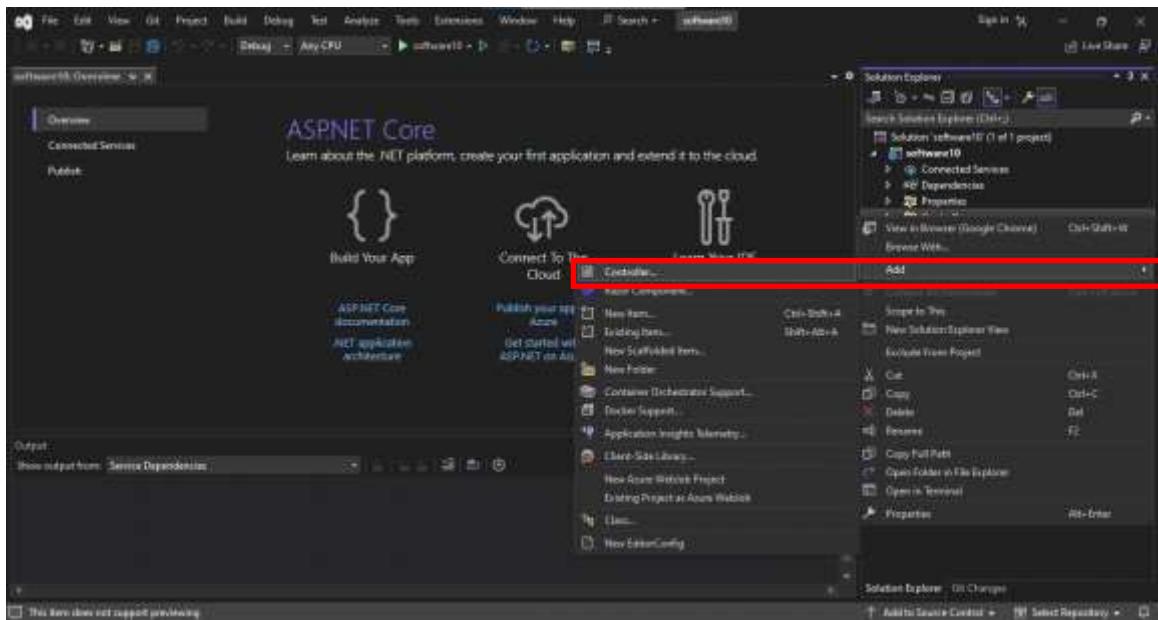
Step 13: Disable the Docker if you don't Docker Desktop installed & Click on ‘Create’.



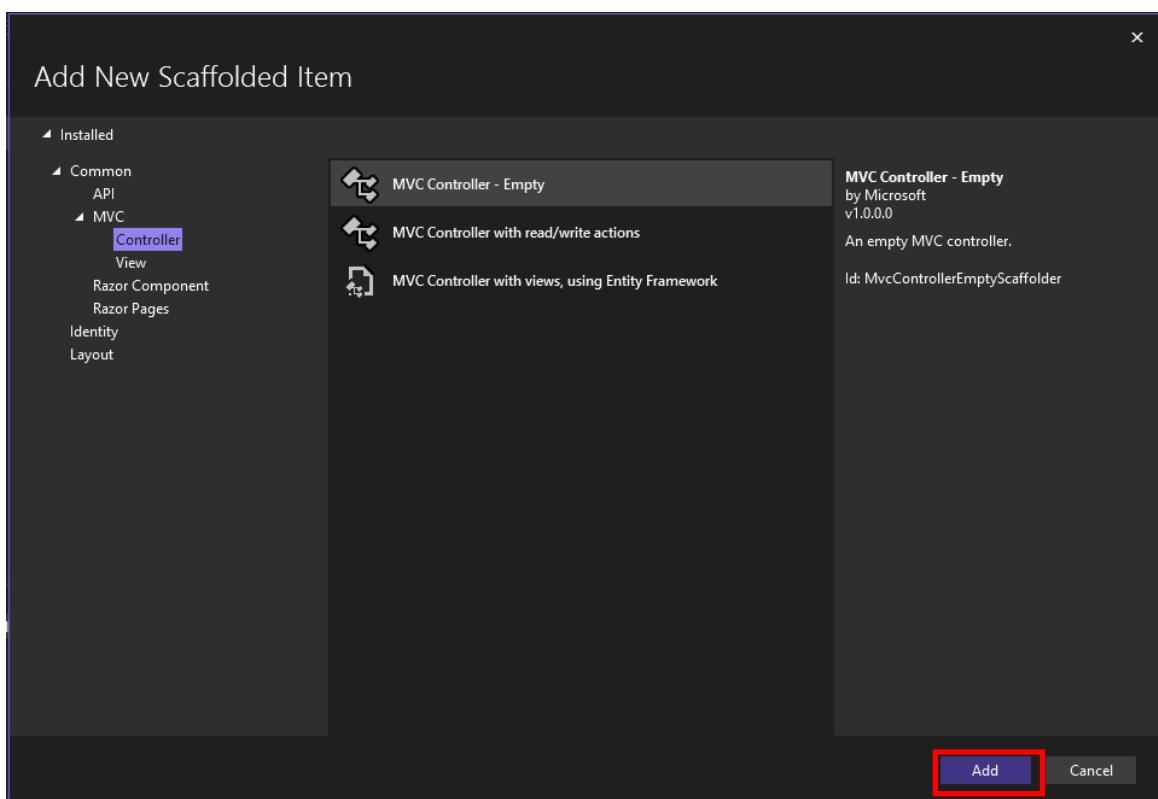
Step 14: Now right click on ‘**controllers**’ here.



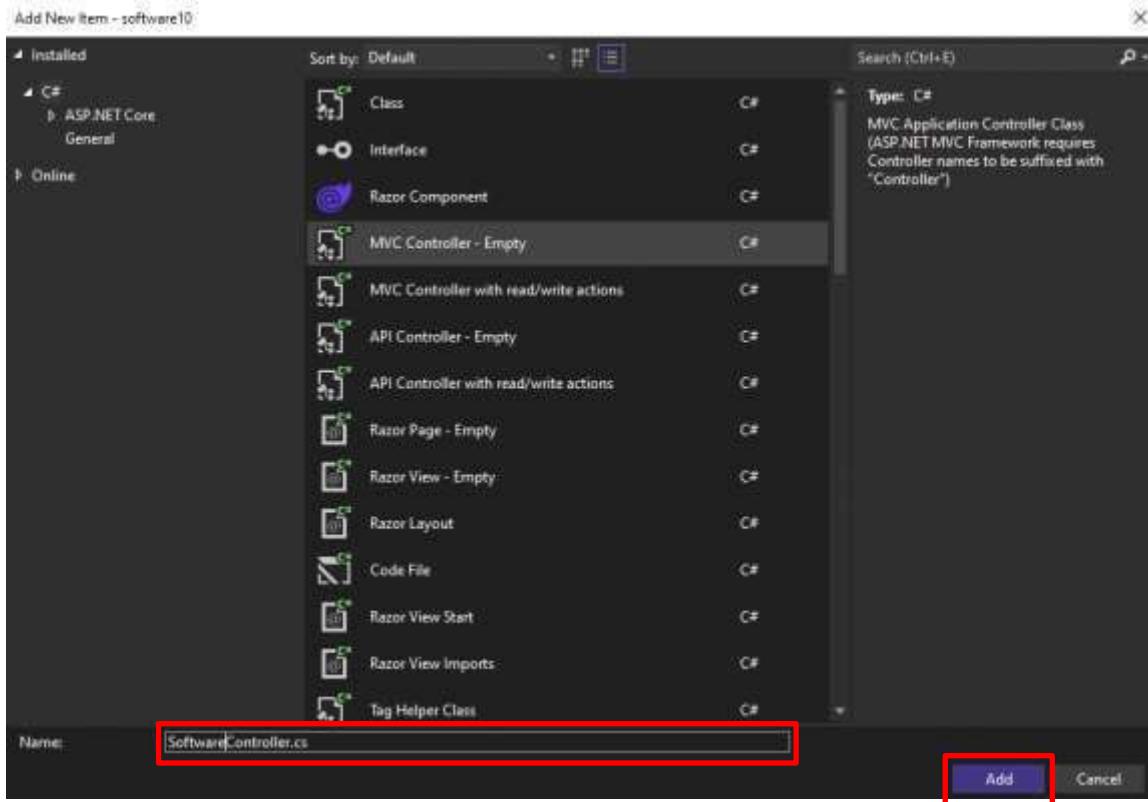
Step 15: Select **Add** & click on ‘**Controllers**’.



Step 16: Click on **Add**.



Step 17: Give a name & click on **Add**.

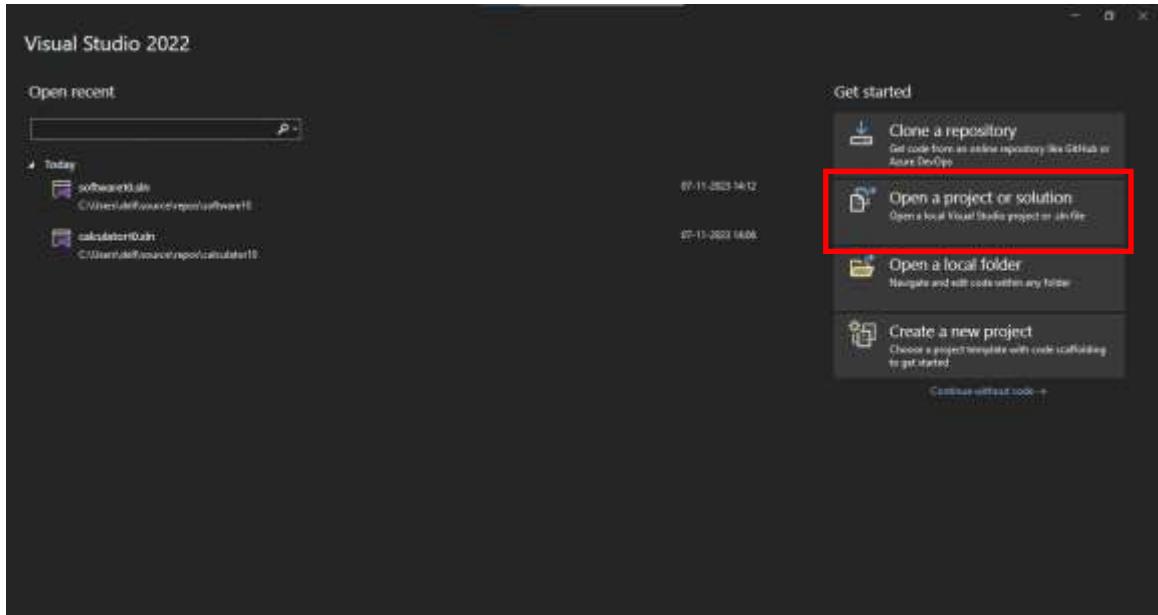


Step 18: Add code to **Software.Controller.cs** file.

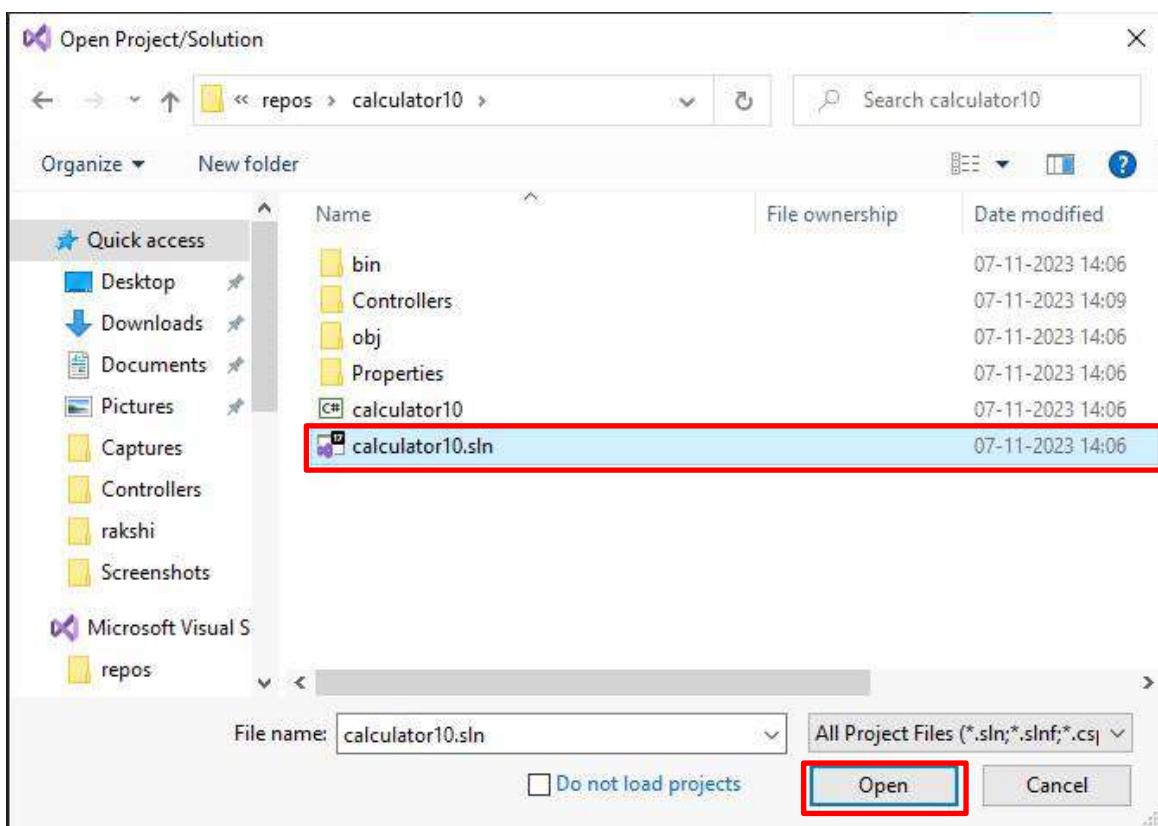
```
using Microsoft.AspNetCore.Mvc;
namespace Software.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class SoftwareController : Controller
    {
        private readonly HttpClient _client;
        private static readonly int[] numbers = new[] { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
        private readonly ILogger<SoftwareController> _logger;

        public SoftwareController(ILogger<SoftwareController> logger)
        {
            _logger = logger;
            HttpClientHandler clientHandler = new HttpClientHandler();
            clientHandler.ServerCertificateCustomValidationCallback = (sender, cert, chain, sslPolicyErrors) => {
                _client = new HttpClient(clientHandler);
            }
        }
    }
}
```

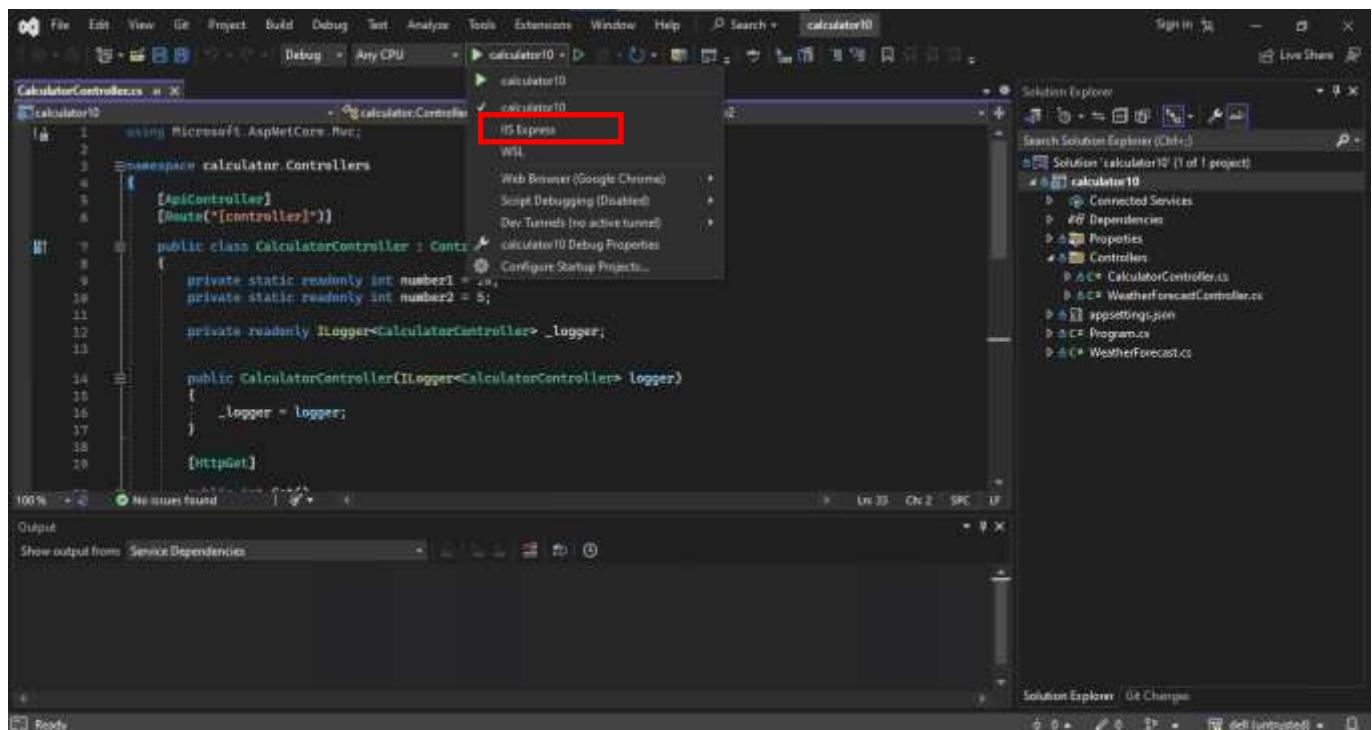
Step 19: Now once again open the Microsoft Visual Studios & click on ‘Open a Project Solution’.



Step 20: Now open the **calculator10.sln** file.



Step 21: Select **IIS Express** for running the code.



Step 22: Now click on **Add Numbers**.

A screenshot of the Swagger UI browser extension. The address bar shows 'localhost:44305/swagger/index.html'. The main content area displays the 'calculator10' API documentation. Under the 'Calculator' section, there is a 'GET /Calculator/AddNumbers' endpoint, which is highlighted with a red box. Other sections shown include 'WeatherForecast' with a 'GET /WeatherForecast' endpoint and 'Schemas' with a 'WeatherForecast' entry.

Step 23: Select **Try it out** option & enter any 2 numbers. Then click on **Execute**.

The screenshot shows the Swagger UI interface for a 'calculator10' API. A POST request is selected for the '/Calculator/AddNumbers' endpoint. The 'Parameters' section shows two parameters: 'num1' (integer, value 10) and 'num2' (integer, value 5). The 'Execute' button at the bottom is highlighted with a red box.

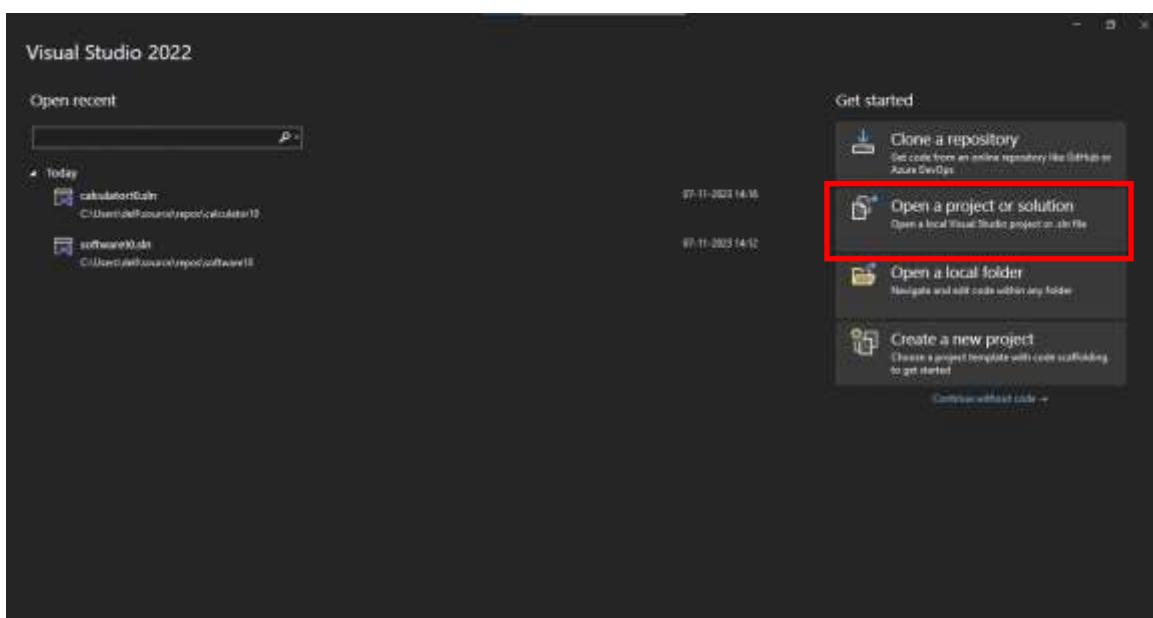
Step 24: Observe the result.

The screenshot shows the Swagger UI displaying the results of the executed POST request. The response code is 200, the response body is '15', and the response headers include 'Content-Type: application/json; charset=utf-8', 'Date: Tue, 07 Nov 2023 08:48:58 GMT', 'Server: Microsoft-IIS/10.0', and 'X-Powered-By: ASP.NET'.

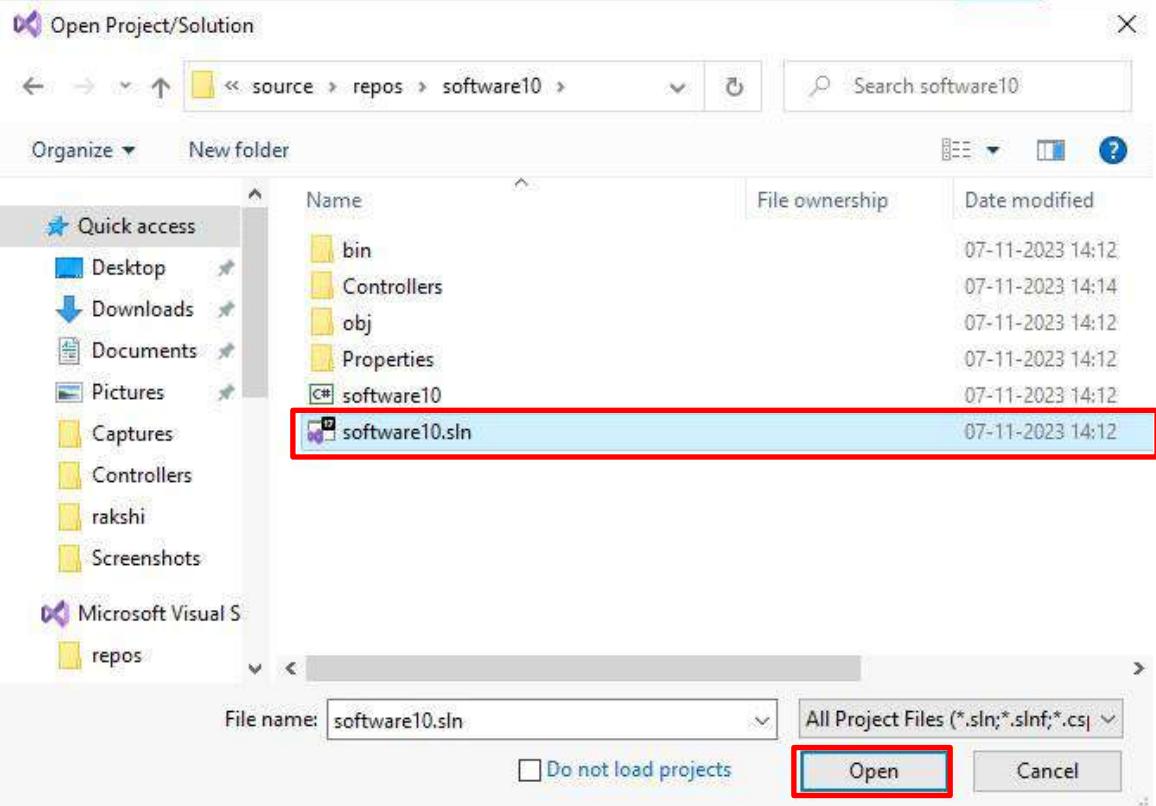
Step 25: Code execution is successful.

The screenshot shows the Swagger UI interface for a REST API. At the top, it displays the URL `localhost:44305/swagger/index.html`. Below this, the "Responses" section shows a single entry for status code 200, labeled "Success". The "Media type" dropdown is set to "text/plain". Underneath, there is a "Controls Accept Headers" section and an "Example Value" section containing the word "Success". To the right, there is a "Links" section with the message "No links". Below the responses, there is a section titled "WeatherForecast" with a "GET /WeatherForecast" button. A "Schemas" section is also visible.

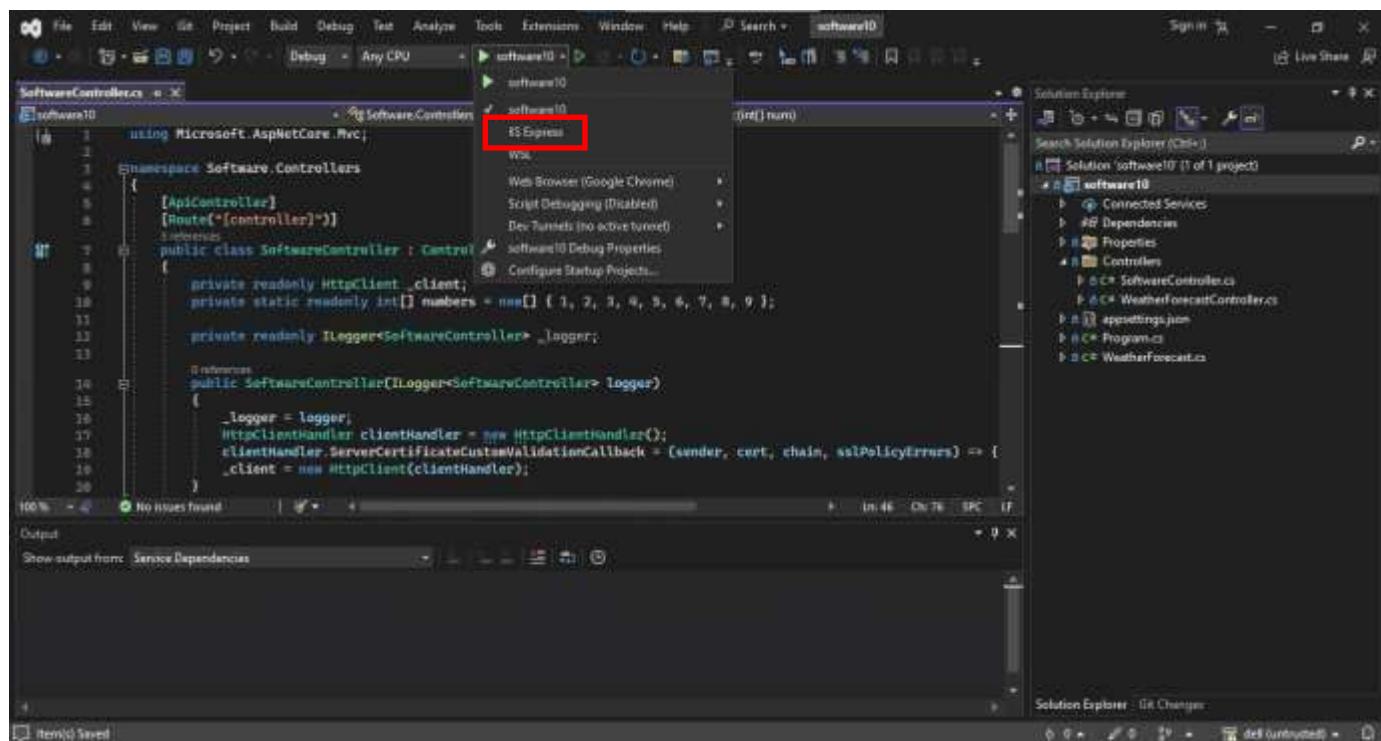
Step 26: Now one more time open the Microsoft Visual Studios & click on '**Open a Project Solution**'.



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.

The screenshot shows the Swagger UI interface for the 'software10' API. At the top, there's a navigation bar with a back/forward button, a refresh button, and a search bar labeled 'Select a definition' with 'software10 v1' selected. Below the header, the API title 'software10' is displayed with a '1.0' version and a 'GAS' badge. A link to 'http://localhost:44304/swagger/index.html' is shown. The main content area is divided into sections: 'Software' (containing a 'GET /Software' button), 'WeatherForecast' (containing a 'GET /WeatherForecast' button), and 'Schemas' (containing a 'WeatherForecast' entry). A red box highlights the 'POST /Software' button under the 'Software' section.

Step 30: Select **Try it out** option & enter any 3 numbers. Then click on **Execute**.

The screenshot shows the 'POST /Software' endpoint details in the Swagger UI. It includes a 'Parameters' section (empty) and a 'Request body' section. In the 'Request body' section, there is a text input field containing the JSON object '{ 1,2,3 }'. Below the input field is a dropdown menu set to 'application/json'. At the bottom of the request body section is a large blue 'Execute' button, which is highlighted with a red box.

Step 31: Observe the result.

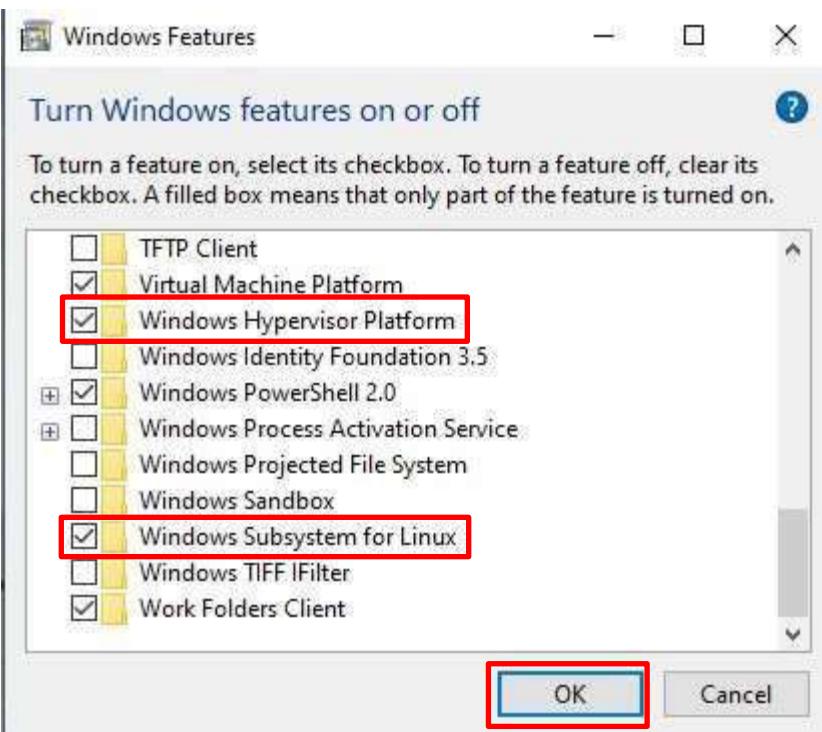
The screenshot shows the Swagger UI interface for a POST request to the endpoint `/Software`. The request body is a JSON array containing the values `1,2,3`. The response status code is 200, and the response body is empty. The response headers include `content-type: application/json; charset=utf-8`, `date: Tue, 07 Nov 2023 06:54:27 GMT`, `server: Microsoft-IIS/10.0`, and `x-powered-by: ASP.NET`.

Step 32: Code execution is successful.

The screenshot shows the Swagger UI interface for a GET request to the endpoint `/WeatherForecast`. The response status code is 200, and the response body is empty. The response headers include `Content-Type: text/plain`.

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.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\VVCE>wsl --status
Default Distribution: docker-desktop-data
Default Version: 2

C:\Users\VVCE>
```

Step 3: Now enter the command as **docker --version** to check the version.

```
C:\Users\VVCE>docker --version
Docker version 24.0.6, build ed223bc

C:\Users\VVCE>
```

Installation of Docker Desktop

Go to Web Browser □ Search for Docker □ Click on the second link below.

A screenshot of a Google search results page for the query "docker". The top result is the official Docker website, which includes a brief description of Docker as a platform for containerization and a link to "Docker Desktop". A red box highlights this link. To the right of the search results is a snippet from the Docker website with a detailed description of what Docker is.

Scroll down & select **Download and Install** option here.

A screenshot of the "Developer resources" page for Docker Desktop. The page features a "Find support" section with a "Get support →" button. Below this are four cards: "Get started", "Download and Install" (which is highlighted with a red box), "Get the latest news", and "View the Product manual".

Now click on **Docker Desktop for Windows** here.

Docker overview

Get Docker

Get started

Language-specific guides

Develop with Docker

Build with Docker

Deployment and orchestration

Educational resources

Contribute

Get Docker

Get Docker

Docker is an open platform for developing, shipping, and running applications.

Docker allows you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications.

By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

You can download and install Docker on multiple platforms. Refer to the following section and choose the best installation path for you.

Docker Desktop for Mac
A native application using the macOS sandbox security model that delivers all Docker tools to your Mac.

Docker Desktop for Windows
A native Windows application that delivers all Docker tools to your Windows computer.

Docker Desktop for Linux
A native Linux application that delivers all Docker tools to your Linux computer.

Then click on **Docker Desktop for Windows** for download.

Docker Desktop

Overview

Install Docker Desktop

Install on Mac

Understand permission requirements...

Install on Windows

Understand permission requirements...

Install on Linux

Manuals / Docker Desktop / Install Docker Desktop / Install on Windows

Install Docker Desktop on Windows

This page contains the download URL, information about system requirements, and instructions on how to install Docker Desktop for Windows.

Docker Desktop for Windows

For checksums, see [Release notes](#)

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
software             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.

```
C:\Command Prompt - python
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

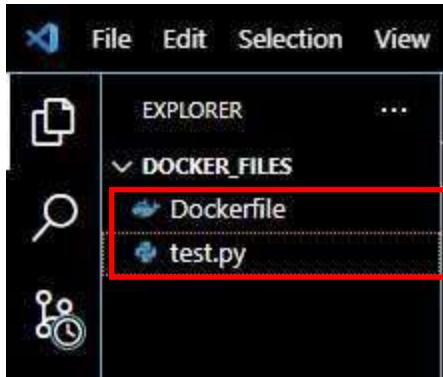
C:\Users\VVCE>python
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct  2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("welcome")
welcome
>>> -
```

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.

```
1 print("full stack development")
2 print("welcome to WP")
```

Step 11: Enter the below code to **Dockerfile** as well.

```
1 FROM python
2 WORKDIR /app
3 COPY . /app
4 CMD ["python","test.py"]
```

Step 12: Go to Terminal □ New Terminal □ enter the command as **python test.py** here.

```
PS C:\Users\WCE\Desktop\Docker_files> python test.py
full stack development
welcome to WP
PS C:\Users\WCE\Desktop\Docker_files>
```

Step 13: Now enter the command as **docker build -t firstapp .** here.

TERMINAL PORTS

TERMINAL

```
PS C:\Users\WVCE\Desktop\Docker_files> python test.py
full stack development
welcome to VVP
PS C:\Users\WVCE\Desktop\Docker_files> docker build -t firstapp .
[+] Building 4.0s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 105B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:latest
=> [1/3] FROM docker.io/library/python@sha256:7b8d65a924f596eb65306214f559253c468336bcae09fd57542977456346 0.0s
=> [internal] load build context
=> => transferring context: 197B
=> CACHED [2/3] WORKDIR /app
=> [3/3] COPY . /app
=> exporting to image
=> => exporting layers
=> => writing image sha256:0a789c6c4588e6cbc392d2003d9394c8830469b854fdd1a08afee47dbb84d68a
=> => naming to docker.io/library/firstapp
```

What's Next?

View a summary of image vulnerabilities and recommendations → docker scout quickview

PS C:\Users\WVCE\Desktop\Docker_files>

Step 14: In the folder path cmd, enter the command as **docker images** to view created image.

```
C:\Users\WVCE\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\WVCE\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.

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	testimage	thirdapp	Running	N/A	8080	1 day ago	[Edit] [Delete]
<input type="checkbox"/>	testimage	thirdapp	Exited	N/A	8080	4 days ago	[Edit] [Delete]
<input type="checkbox"/>	testpython	firstapp	Exited	N/A	8080	26 seconds ago	[Edit] [Delete]

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.

```
public class sample {
    public static void main(String args){
        System.out.println("Hello in FSQ");
        System.out.println("VOC");
    }
}
```

The screenshot shows the VS Code editor with the "sample.java" file open. The code defines a simple Java class with a main method that prints two strings to the console.

Step 20: Also enter the below code to **Dockerfile** here.

```
FROM openjdk
WORKDIR /app
COPY . /app
RUN javac sample.java
CMD ["java","sample"]
```

The screenshot shows the VS Code editor with the "Dockerfile" file open. The Dockerfile contains instructions to use the openjdk base image, set the working directory to /app, copy the current directory contents into /app, compile the sample.java file, and run it with the command "java sample".

Step 21: Go to Terminal → New Terminal → enter the command as **javac sample.java & java sample** here.

TERMINAL PORTS

PS C:\Users\WCE\Desktop\Docker_java> javac sample.java
 PS C:\Users\WCE\Desktop\Docker_java> java sample
 Welcome to FSD
 WCE
 PS C:\Users\WCE\Desktop\Docker_java>

Step 22: Enter the command as **docker build -t secondjavaapp .** to build a docker image.

```
PS C:\Users\WCE\Desktop\Docker_java> docker build -t secondjavaapp .
[+] Building 73.6s (9/9) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 122B
=> [internal] load metadata for docker.io/library/openjdk:latest
=> [1/4] FROM docker.io/library/openjdk@sha256:9b448de897d211c9e0ec635a485650aed6e28d4eca1efbc34940560a48
=> => resolve docker.io/library/openjdk@sha256:9b448de897d211c9e0ec635a485650aed6e28d4eca1efbc34940560a480
=> => sha256:9b448de897d211c9e0ec635a485650aed6e28d4eca1efbc34940560a480b3f1 1.04kB / 1.04kB
=> => sha256:fe05457a5e9b9403fbe72eeba507ae80a4237d2d2d3f219fa62ceb128482a9ee 954B / 954B
=> => sha256:71260f256d19f4ae5c762601e5301418d2516ca591103b1376f063be0b7ba056 4.46kB / 4.46kB
=> => sha256:197c1acd755131915cd019bdd58658d4445b3638f65449932c18ee39b6047c 44.56MB / 44.56MB
=> => sha256:95a27dbe0150755fca4304b4af0d7d6dd6a40ede6fdb30da8568e9e8cdf23a9 188.74MB / 188.74MB
=> => sha256:57b698b7af4b18900b53c768746b1dfb603dfb9aec1eea328fdac86d37001e2a 12.26MB / 12.26MB
=> => extracting sha256:197c1acd755131915cd019bdd58658d4445b3638f65449932c18ee39b6047c
=> => extracting sha256:57b698b7af4b18900b53c768746b1dfb603dfb9aec1eea328fdac86d37001e2a
=> => extracting sha256:95a27dbe0150755fca4304b4af0d7d6dd6a40ede6fdb30da8568e9e8cdf23a9
=> [internal] load build context
=> => transferring context: 800B
=> [2/4] WORKDIR /app
=> [3/4] COPY . /app
=> [4/4] RUN javac sample.java
=> exporting to image
=> => exporting layers
=> => writing image sha256:3736e2c68a1e8e83a2ea7e8da2f637497dc8096f1aedb7f6e1a54acc4ae4ad3f
=> => naming to docker.io/library/secondjavaapp
```

What's Next?
 View a summary of image vulnerabilities and recommendations → [docker scout quickview](#)

Step 23: In the folder path cmd, enter the command as **docker images** to view the list of created 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
software	dev	f73cadff2a7c	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\WCE\Desktop\Docker_java>docker run --name testjavaapp secondjavaapp
 Welcome to FSD
 WCE

Step 25: You can see the image has been created in Docker Desktop.

Docker Desktop Upgrade plan

Containers

Images Volumes Dev Environments Docker Spout Learning center Extensions Add Extensions

Container CPU usage: No containers are running Container memory usage: No containers are running

Search: Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
testjython	itinstans	Exited	N/A		21 minutes ago	[Actions]
dockerimage	itinstans	Exited	N/A		1 day ago	[Actions]
testimage	secondimage	Exited	N/A		4 days ago	[Actions]
testjavaapp	secondjavapro	Exited	N/A		32 seconds ago	[Actions]

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'.

https://sites.google.com/vvce.ac.in/devops

Reading data from JSON file
Writing data to JSON file
Maven dependency
Docker
Installation
Docker commands
Creating image for python application
Creating image for java application

Step 3: Now download the 2 files '**Dockerfile**' & '**test.py**' which are displayed here.

The screenshot shows a Google Drive interface. In the top left, there's a 'Drive' button and a search bar labeled 'Search in Drive'. Below the search bar, it says 'Shared with me > docker_python'. There are filters for 'Type', 'People', and 'Modified'. A red box highlights two files: 'Dockerfile' (64 bytes) and 'test.py' (51 bytes), both owned by 'Anil Kumar B...' and modified on Nov 3, 2023.

Step 4: Then copy & paste the 2 downloaded files in the new folder created as '**docker**' as shown below.

The screenshot shows a Windows File Explorer window with the path 'This PC > Desktop > docker'. The 'Downloads' folder is selected on the left. A red box highlights the 'Dockerfile' and 'test.py' files in the main list, which are both 1 KB in size and were modified on 10-11-2023 at 10:49.

Step 5: Go to Desktop docker cmd as shown below.

The screenshot shows a Windows File Explorer window with the path 'cmd'. The 'Desktop' folder is selected on the left. A red box highlights the 'Dockerfile' and 'test.py' files in the main list, which are both 1 KB in size and were modified on 10-11-2023 at 10:49.

Step 6: Now enter '**code .**' as show below.

Note: This command will directly enter to the VS Code.

The screenshot shows a terminal window with the following text:
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.
C:\Users\STUDENT\Desktop\docker>
A red box highlights the command 'code .' entered in the terminal.

Step 7: Now save the default code in '**Dockerfile**' as shown here.

```
FROM python
WORKDIR /app
COPY . /app
CMD ["python", "test.py"]
```

Step 8: Now save the default code in 'test.py' as shown here.

Note: You can change the code if you want.

```
print("first python program")
print("vvce mysore")
```

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

```
PS C:\Users\STUDENT\Desktop\docker> docker build -t pythonapp .
[+] Building 3.8s (8/8) FINISHED
--> [internal] load build definition from Dockerfile
--> [internal] load .dockerignore
--> transferring dockerfile: 10B
--> [internal] load context: 2B
--> [internal] load metadata for docker.io/library/python:latest
--> [internal] FROM docker.io/library/python@sha256:7bb05a024f596eb60306214f550215c468336bcac09fb57542977456940fcacf
--> [internal] load build context
--> [internal] transfer context: 59B
--> [internal] COPY [2/3] WORKDIR /app
--> [internal] COPY . /app
--> [internal] exporting to image
--> [internal] writing image sha256:4f876d74c1e1aef89e5a3066055ad63411e3cfeff118a0694f5a5355185ee
--> [internal] naming to docker.io/library/pythonapp

What's Next?
View a summary of image vulnerabilities and recommendations -> docker scan quickview
PS C:\Users\STUDENT\Desktop\docker> docker run --name pythoncont pythonapp
First python program
vvce mysore
PS C:\Users\STUDENT\Desktop\docker>
```

Step 10: Go to Docker Desktop --> Containers --> you can see the created container with image here.

Docker Desktop Upgrade plan

Containers

Images Volumes Dev Environments Docker-Scout Learning center Extensions Add Extensions

Container CPU usage: No containers are running.

Container memory usage: No containers are running.

Show charts

Search Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
brinda_statistic	theomarbs/statistic	Exited	N/A		8 minutes ago	[Run] [Logs] [Delete]
funny_cray_tracesview	theomarbs/tracesview	Exited	N/A		44 minutes ago	[Run] [Logs] [Delete]
related_smid_gctoolkit	theomarbs/gctoolkit	Exited	N/A		45 minutes ago	[Run] [Logs] [Delete]
testleveldb_313af05dab	theomarbs/leveldb	Exited	N/A		45 minutes ago	[Run] [Logs] [Delete]
testimage_458cc205344	theomarbs/testimage	Exited	N/A		52 minutes ago	[Run] [Logs] [Delete]
pythoncont_655e0f15729	theomarbs/pythoncont	Exited	N/A		47 seconds ago	[Run] [Logs] [Delete]

Showing 6 items

Engine running RAM 4.43 GB CPU 0.00% Not signed in v4.25.0

Step 11: Now click on 'Run' option here.

Docker Desktop Upgrade plan

pythoncont

STATUS Exited (0) (1 minute ago)

Logs Inspect Bind mounts Exec Files Stats

2023-11-19 10:59:27 first python program
2023-11-19 10:59:27 exec myscript

Step 12: You can see the output here.

Docker Desktop Upgrade plan

pythoncont

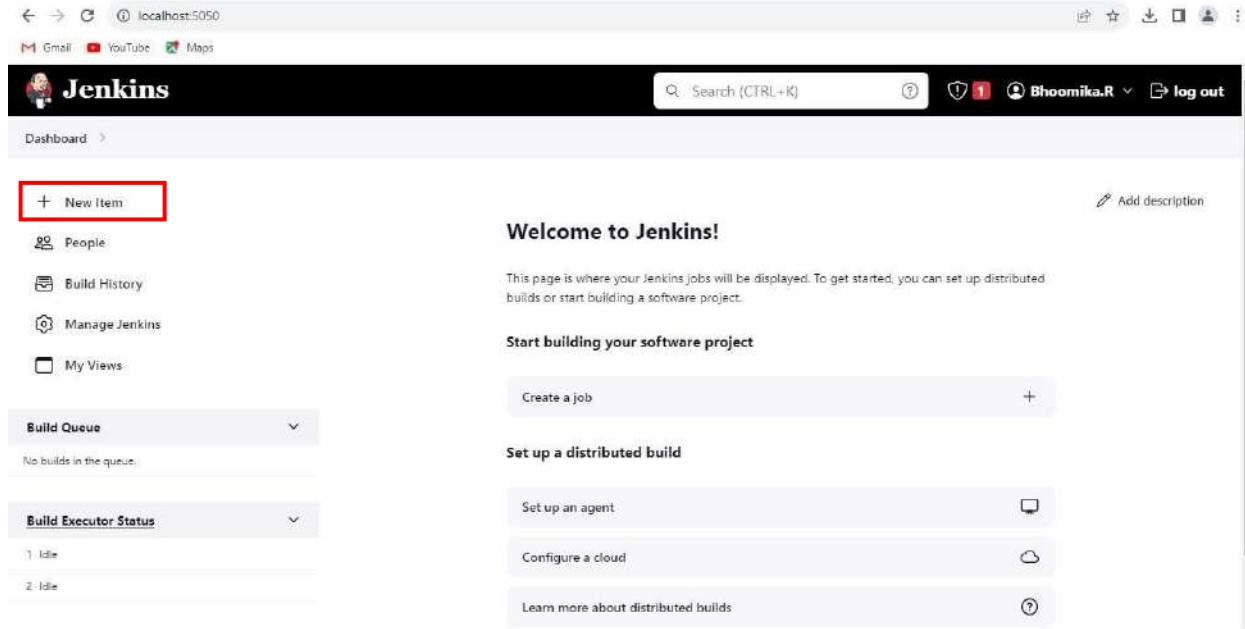
STATUS Exited (0) (6 seconds ago)

Logs Inspect Bind mounts Exec Files Stats

2023-11-19 10:59:27 first python program
2023-11-19 10:59:27 exec myscript
2023-11-19 11:01:13 first python program
2023-11-19 11:01:13 exec myscript

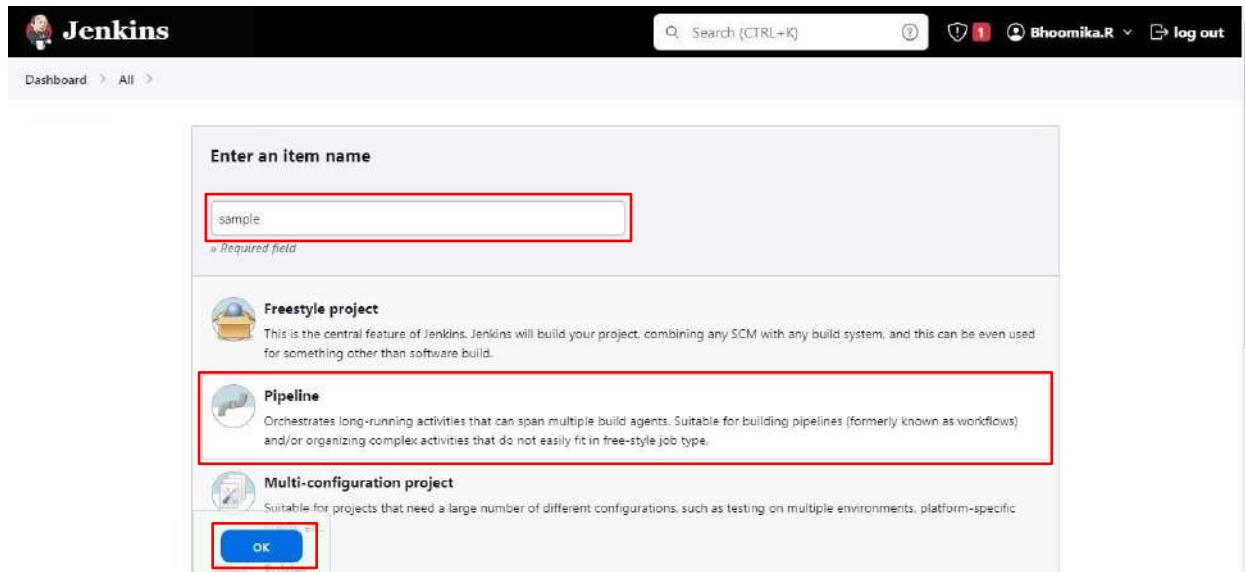
4. Jenkins Pipeline

Step 1: Login to your Jenkins Account. Then click on 'New Item' here.



The screenshot shows the Jenkins dashboard at localhost:5050. The left sidebar has a red box around the '+ New Item' button. The main area features a 'Welcome to Jenkins!' message and a 'Start building your software project' section. A 'Create a job' button is at the top right of this section. Below it are links for 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. On the left, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (two idle executors).

Step 2: Enter an item name here & select ‘Pipeline’ field. Then click on ok.



The screenshot shows the 'Enter an item name' dialog. A red box highlights the input field containing 'sample'. Another red box highlights the 'Pipeline' project type option, which is described as orchestrating long-running activities across multiple build agents. At the bottom, a red box highlights the 'OK' button.

Step 3: Scroll down and add a simple code under Script section. Then click on **Apply & Save**.

Dashboard > sample > Configuration

Configure

- General
- Advanced Project Options
- Pipeline**

Definition: Pipeline script

```

1 pipeline {
2   agent any
3   stages {
4     stage('Clone Repository') {
5       steps {
6         echo "Cloned Successfully"
7       }
8     }
9     stage('Test') {
10      steps {
11        echo "Tested Successful"
12      }
13    }
14    stage('Deploy') {
15      steps {
16        echo "Deployed Successfully"
17      }
18  }
19}

```

try sample Pipeline...

Use Groovy Sandbox ?

Pipeline Syntax

Step 4: Now click on **Build Now** option here.

Jenkins

Dashboard > sample >

Status: sample

- </> Changes
- Build Now**
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

Stage View

No data available. This Pipeline has not yet run.

Permalinks

Build History trend No builds

Step 5: Now observe the three stage views (**Clone Repository**, **Test**, **Deploy**) as shown below.

Dashboard > sample >

Status: sample

- </> Changes
- Build Now
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

Stage View

	Clone Repository	Test	Deploy
Average stage times:	1s	239ms	196ms
(Average full run time: ~12s)			
# Nov 21 09:56	1s	239ms	196ms
No Changes			

Permalinks

Nov 21, 2023, 9:56 AM Atom feed for all Atom feed for failures

Step 6: Then click on **Console Output** here.

The screenshot shows the Jenkins interface for a build named 'sample' (Build #1). The top navigation bar includes a search bar, a user icon, and a 'log out' button. The main content area displays the build status as 'Build #1 (Nov 21, 2023, 9:56:11 AM)' with a green checkmark. Below the status, there are several actions: 'Status', 'Changes', 'Console Output' (which is highlighted with a red box), 'Edit Build Information', 'Delete build '#1', 'Restart from Stage', 'Replay', 'Pipeline Steps', and 'Workspaces'. A note on the right says 'Keep this build forever'. The pipeline steps section shows a single step: 'Started by user Bhoomika.R'. The bottom right corner indicates 'REST API' and 'Jenkins 2.426.1'.

Step 7: Now observe the three stages running successfully.

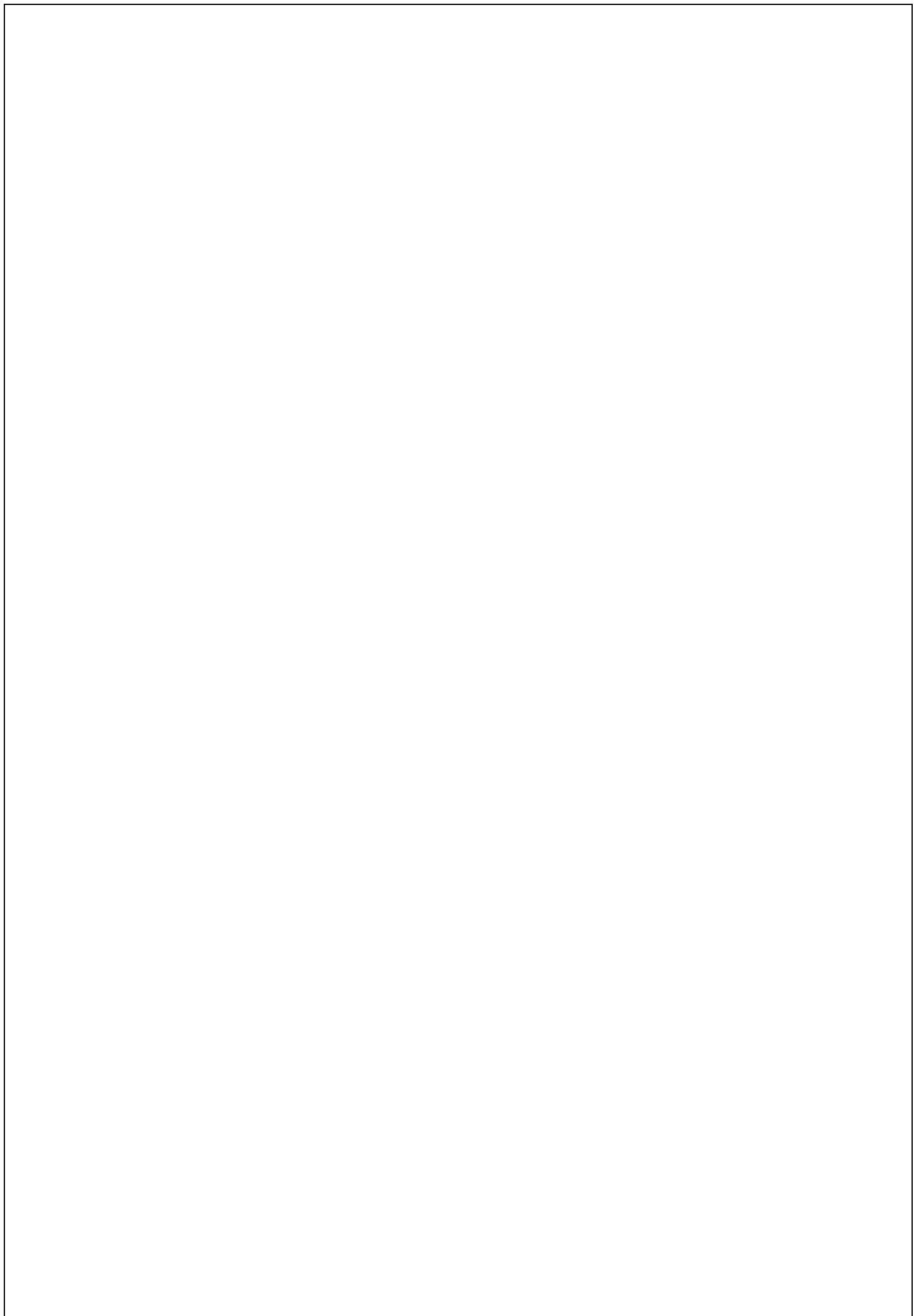
The screenshot shows the Jenkins interface for the 'Console Output' of Build #1. The left sidebar lists actions: 'Status', 'Changes', 'Console Output' (highlighted with a red box), 'Edit Build Information', 'Delete build '#1', 'Restart from Stage', 'Replay', 'Pipeline Steps', and 'Workspaces'. The main content area displays the execution log of the pipeline steps. The log shows the pipeline starting, cloning a repository, running tests, deploying, and finally finishing successfully.

```
Started by user Bhoomika.R
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\jenkins\workspace\sample
[Pipeline] {
    [Pipeline] stage
    [Pipeline] {
        [Pipeline] node
        [Pipeline] echo
        Cloned Successfully
    }
}
[Pipeline] }
[Pipeline] stage
[Pipeline] stage
[Pipeline] {
    [Pipeline] node
    [Pipeline] echo
    Test Successful
}
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] {
    [Pipeline] node
    [Pipeline] echo
    Deployed Successfully
}
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Step 8: Then click on **Pipeline Steps** to view the pipeline steps as shown below.

The screenshot shows the Jenkins interface for the 'Pipeline Steps' of Build #1. The left sidebar lists actions: 'Status', 'Changes', 'Console Output' (highlighted with a red box), 'Edit Build Information', 'Delete build '#1', 'Restart from Stage', 'Replay', 'Pipeline Steps' (highlighted with a red box), and 'Workspaces'. The main content area displays a table of pipeline steps. Each row contains the step name, duration, arguments, and status. The steps include 'Start of Pipeline', 'node', 'stage', 'stage block (Clone Repository)', 'echo', 'stage block (Test)', 'stage', 'stage block (Deploy)', and 'echo'.

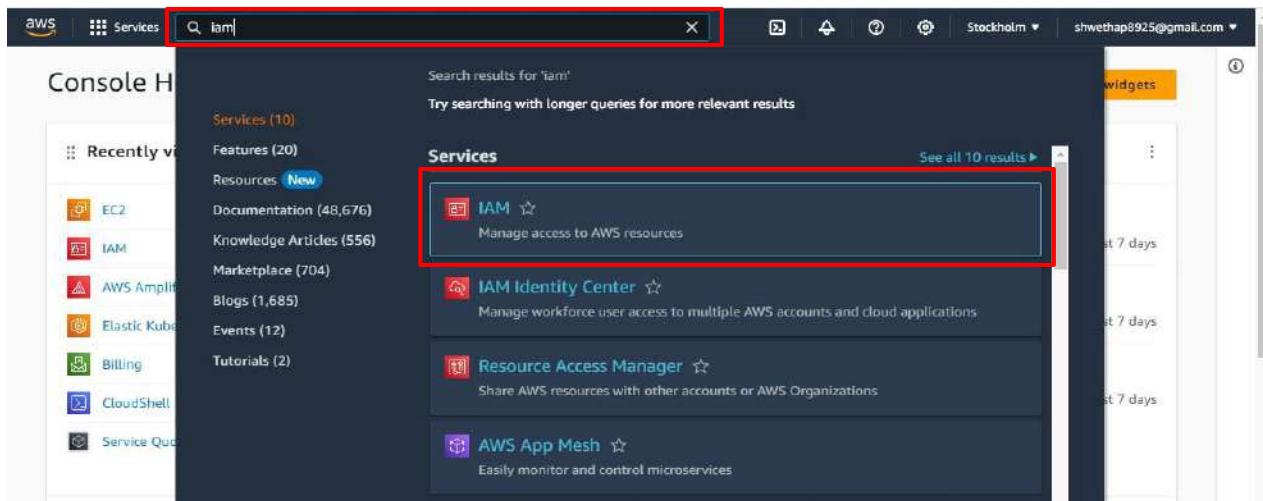
Step	Arguments	Status
Start of Pipeline - (10 sec in block)		Success
node - (5 sec in block)		Success
node block - (3.1 sec in block)		Success
stage - (1.8 sec in block)	Clone Repository	Success
stage block (Clone Repository) - (1.4 sec in block)		Success
echo - (32 ms in self)	Cloned Successfully	Success
stage - (0.3 sec in block)	Test	Success
stage block (Test) - (0.16 sec in block)		Success
echo - (15 ms in self)	Tested Successful	Success
stage - (0.2 sec in block)	Deploy	Success
stage block (Deploy) - (0.12 sec in block)		Success
echo - (15 ms in self)	Deployed Successfully	Success



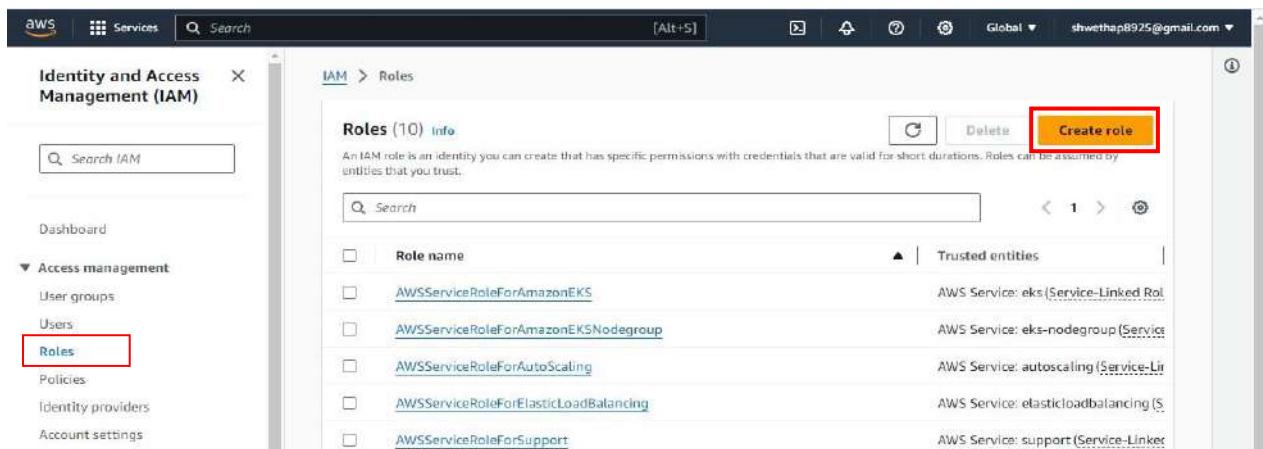
WEEK – 12

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.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
EKS

Choose a use case for the specified service.

Use case

- EKS
Allows EKS to manage clusters on your behalf.
- EKS - Cluster
Allows access to other AWS service resources that are required to operate clusters managed by EKS.
- EKS - Nodegroup
Allows EKS to manage nodegroups on your behalf.
- EKS - Fargate pod
Allows access to other AWS service resources that are required to run Amazon EKS pods on AWS Fargate.
- EKS - Fargate profile
Allows EKS to run Fargate tasks.
- EKS - Connector
Allows access to other AWS service resources that are required to connect to external clusters
- EKS Local - Outpost
Allows Amazon EKS Local to call AWS services on your behalf.

Cancel **Next**

Step 4: Click on Next.

IAM > Roles > Create role

Step 1: Select trusted entity

Step 2: Add permissions

Step 3: Name, review, and create

Add permissions [Info](#)

Permissions policies (1) [Info](#)
The type of role that you selected requires the following policy.

Policy name	Type
AmazonEKSClusterPolicy	AWS managed

▶ Set permissions boundary - optional

Cancel Previous **Next**

Step 5: Give the role name as **eks-cluster-role-103** as shown here.

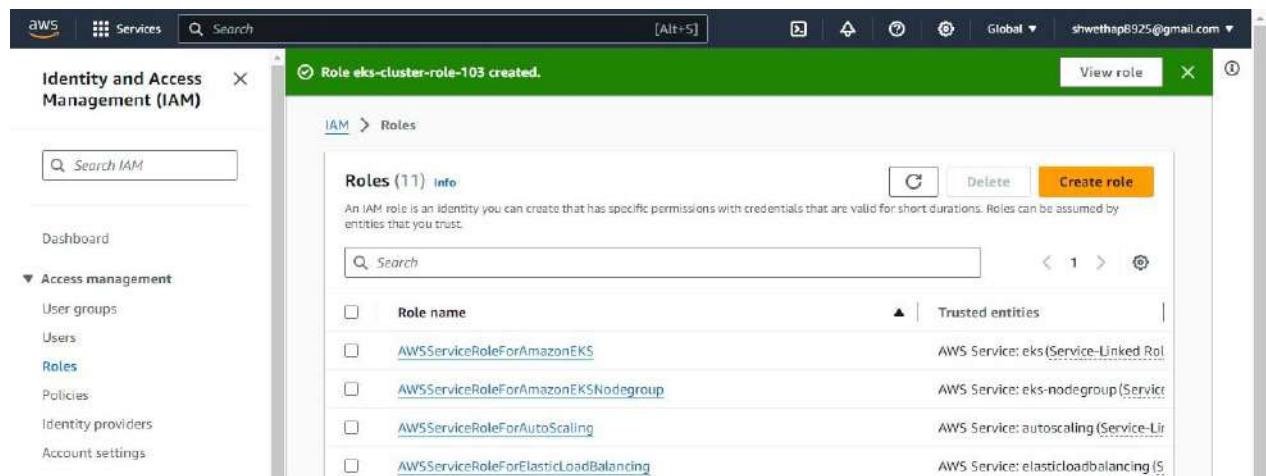
Role name
Enter a meaningful name to identify this role.
 eks-cluster-role-103
Maximum 64 characters. Use alphanumeric and '-' characters.

Description
Add a short explanation for this role.
 Allows access to other AWS service resources that are required to operate clusters managed by EKS.
Maximum 1000 characters. Use alphanumeric and '-' characters.

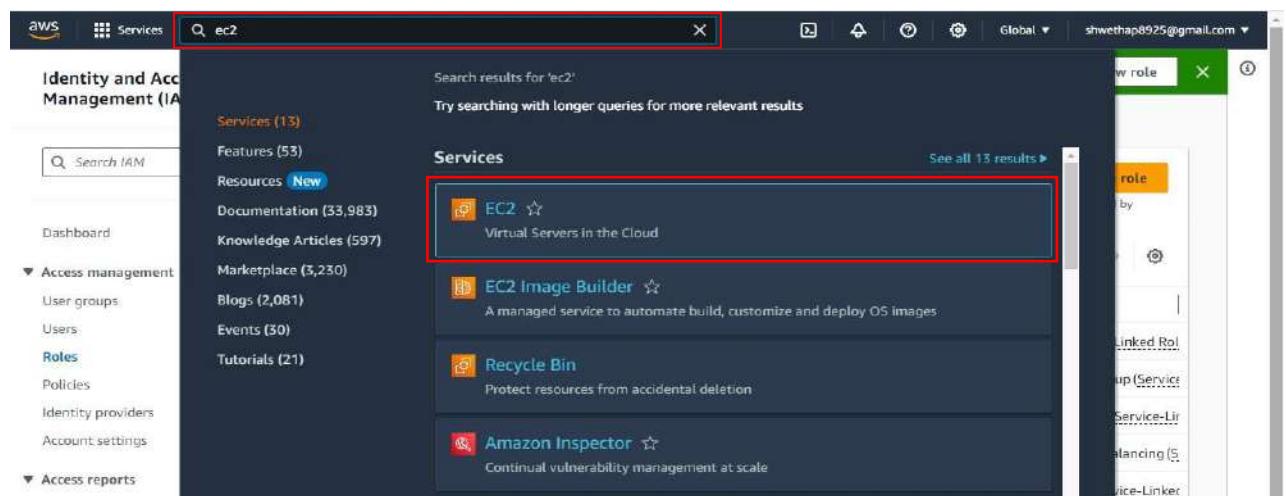
Step 6: Now click on **Create Role**.



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.

Security Groups (13) Info					
		Actions ▾		Export security groups to CSV	
<input type="text"/> Find resources by attribute or tag					
□	Name	Security group ID	Security group name	VPC ID	⋮
□	-	sg-05440db7fe2035c1c	launch-wizard-7	vpc-0de6d8080620	⋮
□	-	sg-0ffe7859fa450e8a5	launch-wizard-2	vpc-0de6d8080620	⋮
□	-	sg-0cf17dbe7f56c6297	eks-sec-grp-2023	vpc-0de6d8080620	⋮
□	-	sg-00a163a50427f899f	launch-wizard-5	vpc-0de6d8080620	⋮
□	-	sg-011fc0812f2f7881	EC2 Security-group	vpc-0de6d8080620	⋮
□	-	sg-0014e391dcdaefade	launch-wizard-3	vpc-0de6d8080620	⋮
□	-	sg-097e1163b6693b5a2	eks-sec-grp-101	vpc-0de6d8080620	⋮

Step 10: Give the name for Security group here. Now copy the same name & paste to the Description as shown here.

AWS Services Search [Alt+S] Stockholm shwethap8925@gmail.com

EC2 > Security Groups > Create security group

Create security group [Info](#)

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details
Security group name Info eks-sec-grp-103 Name cannot be edited after creation.
Description Info eks-sec-grp-103
VPC Info vpc-0de6d80806263e3f6

Step 11: Now add rules as shown below & click on **Create Security Group** here.

Outbound rules

Type Info	Protocol Info	Port range Info	Destination Info	Description - optional Info
SSH	TCP	22	Any... ▾	0.0.0.0/0 Delete
HTTP	TCP	80	Any... ▾	0.0.0.0/0 Delete
Custom TCP	TCP	8081	Any... ▾	0.0.0.0/0 Delete

Add rule

Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel Create security group

Step 12: Now the Security group is successfully created here.

Security group (sg-0fa7683c9b0cc6114 - eks-sec-grp-103) was created successfully

Details

EC2 > Security Groups > sg-0fa7683c9b0cc6114 - eks-sec-grp-103

sg-0fa7683c9b0cc6114 - eks-sec-grp-103

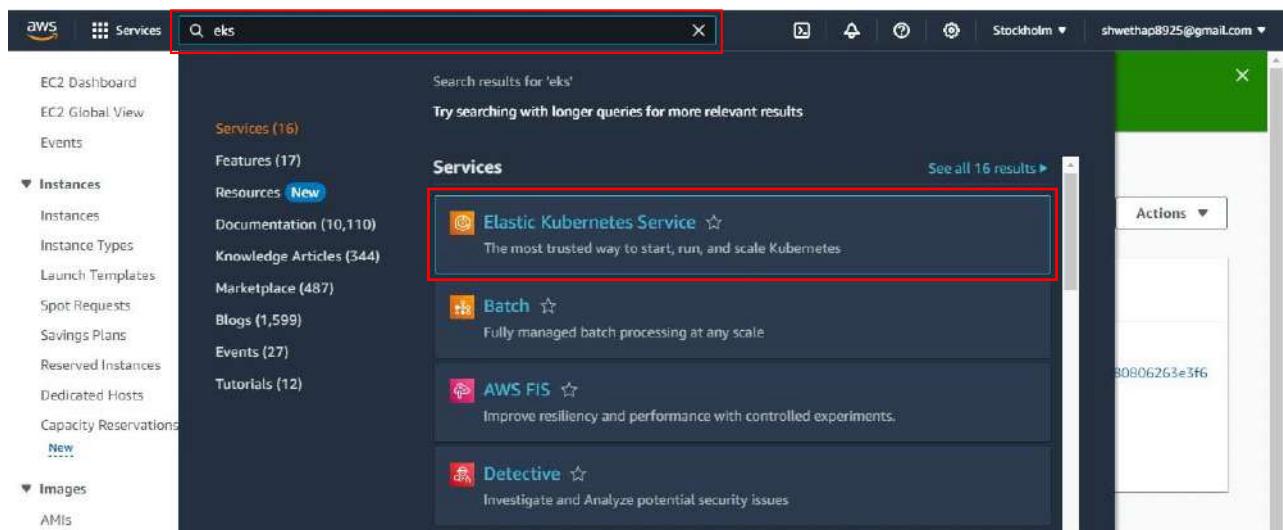
Details

Security group name eks-sec-grp-103	Security group ID sg-0fa7683c9b0cc6114	Description eks-sec-grp-103	VPC ID vpc-0de6d80806263e3f6
Owner 194767121857	Inbound rules count 0 Permission entries	Outbound rules count 3 Permission entries	

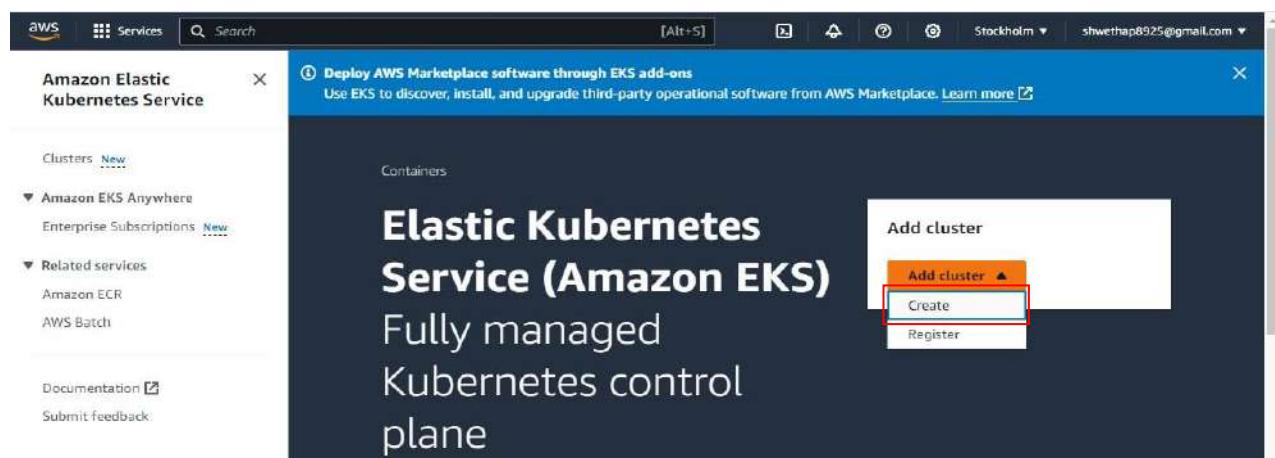
Inbound rules

No security group rules found

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 the created cluster service role here & click on **Next** here.

Cluster configuration [Info](#)

Name
Enter a unique name for this cluster. This property cannot be changed after the cluster is created.
The cluster name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 100.

Kubernetes version [Info](#)
Select Kubernetes version for this cluster.
▼

ⓘ Kubernetes version 1.25 reaches the end of standard support on May 2024. If you don't update your cluster to a later version before that date, it will automatically enter extended support. After the extended support preview ends, clusters on versions in extended support will be subject to additional fees. [Learn more](#)

Cluster service role [Info](#)
Select the IAM role to allow the Kubernetes control plane to manage AWS resources on your behalf. This property cannot be changed after the cluster is created. To create a new role, follow the instructions in the [Amazon EKS User Guide](#).

▼ [Create new role](#)

Secrets encryption [Info](#)
Once turned on, secrets encryption cannot be modified or removed.

Turn on envelope encryption of Kubernetes secrets using KMS
Envelope encryption provides an additional layer of encryption for your Kubernetes secrets.

Tags (0) [Info](#)
Each tag consists of a key and an optional value.

This cluster does not have any tags.

[Add tag](#)
You can add up to 50 tags.

[Cancel](#) [Next](#)

Step 16: Select **Public** endpoint access & click on **Next** here.

Cluster endpoint access [Info](#)
Configure access to the Kubernetes API server endpoint.

Public
The cluster endpoint is accessible from outside of your VPC. Worker node traffic will leave your VPC to connect to the endpoint.

Public and private
The cluster endpoint is accessible from outside of your VPC. Worker node traffic to the endpoint will stay within your VPC.

Private
The cluster endpoint is only accessible through your VPC. Worker node traffic to the endpoint will stay within your VPC.

[Advanced settings](#)

[Cancel](#) [Previous](#) [Next](#)

Step 17: Now click on **Next** here.

Metrics

CloudWatch Info

ⓘ You can enable CloudWatch Container Insights in your clusters through the CloudWatch Observability add-on. After your cluster is created, navigate to the add-ons tab and install CloudWatch Observability add-on to enable Container Insights and start ingesting infrastructure telemetry into CloudWatch.

Control plane logging Info

Send audit and diagnostic logs from the Amazon EKS control plane to CloudWatch Logs.

API server
Logs pertaining to API requests to the cluster.

Audit
Logs pertaining to cluster access via the Kubernetes API.

Authenticator
Logs pertaining to authentication requests into the cluster.

Controller manager
Logs pertaining to state of cluster controllers.

Scheduler
Logs pertaining to scheduling decisions.

Cancel Previous **Next**

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

Select add-ons

Review the add-ons from multiple categories, then select add-ons to enhance your cluster.

Amazon EKS add-ons (5) Info

CoreDNS Info	Amazon VPC CNI Info	kube-proxy Info
Enable service discovery within your cluster. Category networking <input checked="" type="checkbox"/> Installed by default	Enable pod networking within your cluster. Category networking <input checked="" type="checkbox"/> Installed by default	Enable service networking within your cluster. Category networking <input checked="" type="checkbox"/> Installed by default
Amazon EKS Pod Identity Agent Info	Amazon GuardDuty EKS Runtime Monitoring Info	
Install EKS Pod Identity Agent to use EKS Pod Identity to grant AWS IAM permissions to pods through Kubernetes service accounts. Category security	Install EKS Runtime Monitoring add-on within your cluster. Ensure to enable EKS Runtime Monitoring within Amazon GuardDuty. Category security	

Cancel Previous **Next**

Step 19: Again, click on **Next** here.

The screenshot shows the AWS EKS Add-ons configuration interface. It lists two add-ons: 'kube-proxy' and 'Amazon EKS Pod Identity Agent'. Both are currently installed. The 'kube-proxy' section shows it is 'Installed by default' with version v1.25.6-eksbuild.1 selected. The 'Amazon EKS Pod Identity Agent' section shows it is 'Ready to install' with version v1.0.0-eksbuild.1 selected. At the bottom, there are 'Cancel', 'Previous', and 'Next' buttons, with 'Next' highlighted in red.

Step 20: Now click on **Create** option here.

This screenshot shows a confirmation step where the user has selected specific versions for various add-ons. The table lists the add-on name and its corresponding version. The 'Create' button at the bottom is highlighted in red.

Selected add-ons version	
Add-on name	Version
coredns	v1.9.3-eksbuild.2
Add-on name	Version
vpc-cni	v1.12.2-eksbuild.1
Add-on name	Version
kube-proxy	v1.25.6-eksbuild.1
Add-on name	Version
eks-pod-identity-agent	v1.0.0-eksbuild.1

Step 21: Now the cluster has started creating as shown here.

EKS > Clusters > eks-cluster-103

eks-cluster-103

C Delete cluster

Cluster info [Info](#)

Status Creating	Kubernetes version Info 1.25	Support type Standard support until May 2024	Provider EKS
---------------------------------	---	--	-----------------

[Overview](#) [Resources](#) [Compute](#) [Networking](#) [Add-ons](#) [Authentication](#) [Observability](#) >

Details

API server endpoint	OpenID Connect provider URL	Created
...	...	a few seconds ago
Certificate authority	Cluster IAM role ARN arn:aws:iam::194767121857:role/eks-cluster-role-103	Cluster ARN arn:aws:eks:eu-north-1:194767121857:cluster/eks-cluster-103
		Platform version Info

Step 22: Now the cluster creation is successfully done here.

EKS > Clusters > eks-cluster-103

eks-cluster-103

C Delete cluster

ⓘ End of support for Kubernetes version 1.25 is May 2024. If you don't update your cluster to a later version before that date, it will automatically enter extended support. After the extended support preview ends, clusters on versions in extended support will be subject to additional fees. [Learn more](#)

Update now

Cluster info [Info](#)

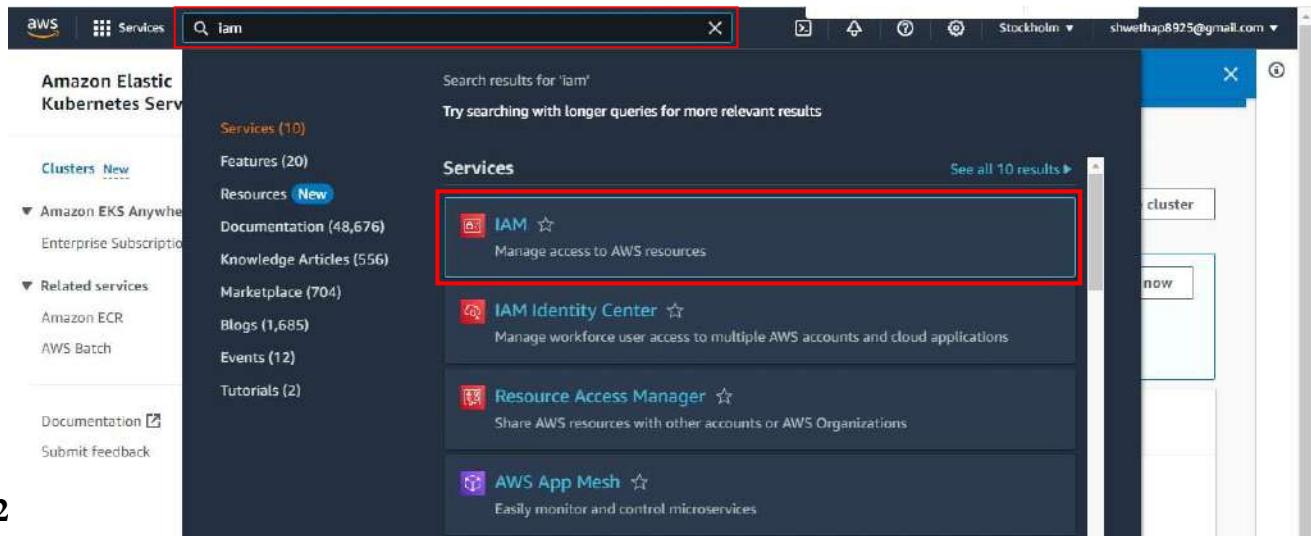
Status Active	Kubernetes version Info 1.25	Support type Standard support until May 2024	Provider EKS
-------------------------------	---	--	-----------------

[Overview](#) [Resources](#) [Compute](#) [Networking](#) [Add-ons](#) [Authentication](#) [Observability](#) >

Details

API server endpoint https://6BD7E3FD82ED8A1ED1972C4884975802.gr7.eu-north-1.eks.amazonaws.com	OpenID Connect provider URL https://oidc.eks.eu-north-1.amazonaws.com/id/6BD7E3FD82ED8A1ED1972C4884975802	Created 9 minutes ago
Certificate authority	Cluster IAM role ARN	Cluster ARN arn:aws:eks:eu-north-1:194767121857:cluster/eks-cluster-103

Step 23: Search for IAM services & click on the first link here.



Step 2

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

User groups

Users

Roles **Selected**

Policies

Identity providers

Account settings

IAM > Roles

Roles (11) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities
AWSServiceRoleForAmazonEKS	AWS Service: eks (Service-Linked Role)
AWSServiceRoleForAmazonEKSNodergroup	AWS Service: eks-nodegroup (Service-Linked Role)
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)

Create role

Step 25: Select EC2 & click on Next here.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
EC2

Choose a use case for the specified service.

Use case

- EC2**
Allows EC2 instances to call AWS services on your behalf.
- EC2 Role for AWS Systems Manager**
Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- EC2 Spot Fleet Role**
Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.
- EC2 - Spot Fleet Auto Scaling**
Allows Auto Scaling to access and update EC2 spot fleets on your behalf.
- EC2 - Spot Fleet Tagging**
Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.
- EC2 - Spot Instances**
Allows EC2 Spot Instances to launch and manage spot instances on your behalf.
- EC2 - Spot Fleet**
Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.
- EC2 - Scheduled Instances**
Allows EC2 Scheduled Instances to manage instances on your behalf.

Cancel **Next**

Step 26: Select the policy name as **AmazonEKSWorkerNodePolicy** here.

Add permissions Info

Permissions policies (3/889) Info

Choose one or more policies to attach to your new role.

Filter by Type

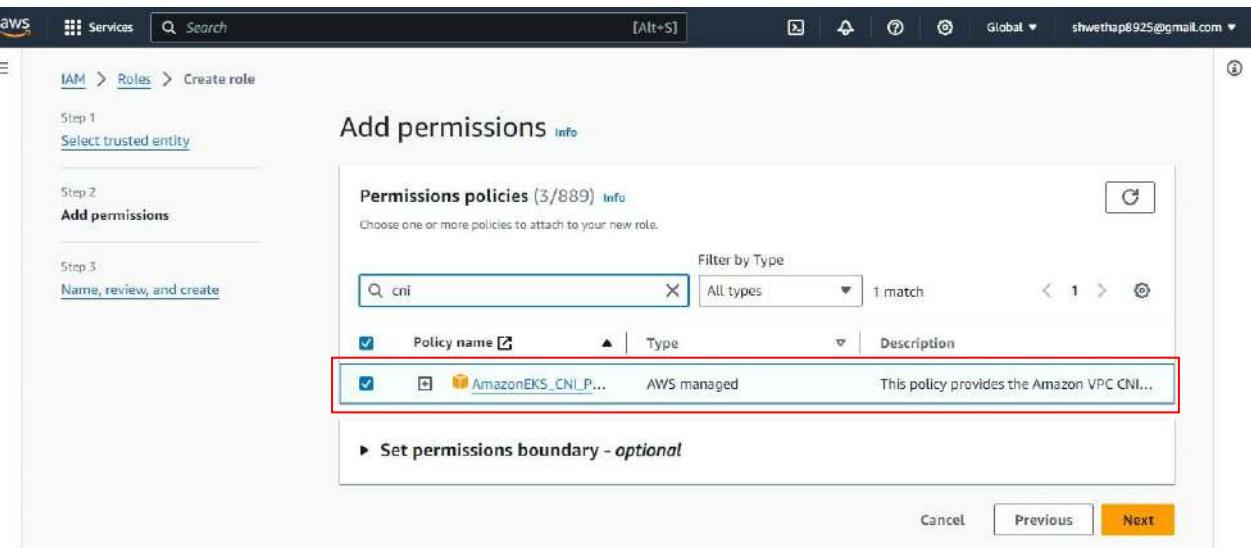
Q: ekswo All types 1 match

Policy name	Type	Description
<input checked="" type="checkbox"/> AmazonEKSWorker...	AWS managed	This policy allows Amazon EKS worker no...

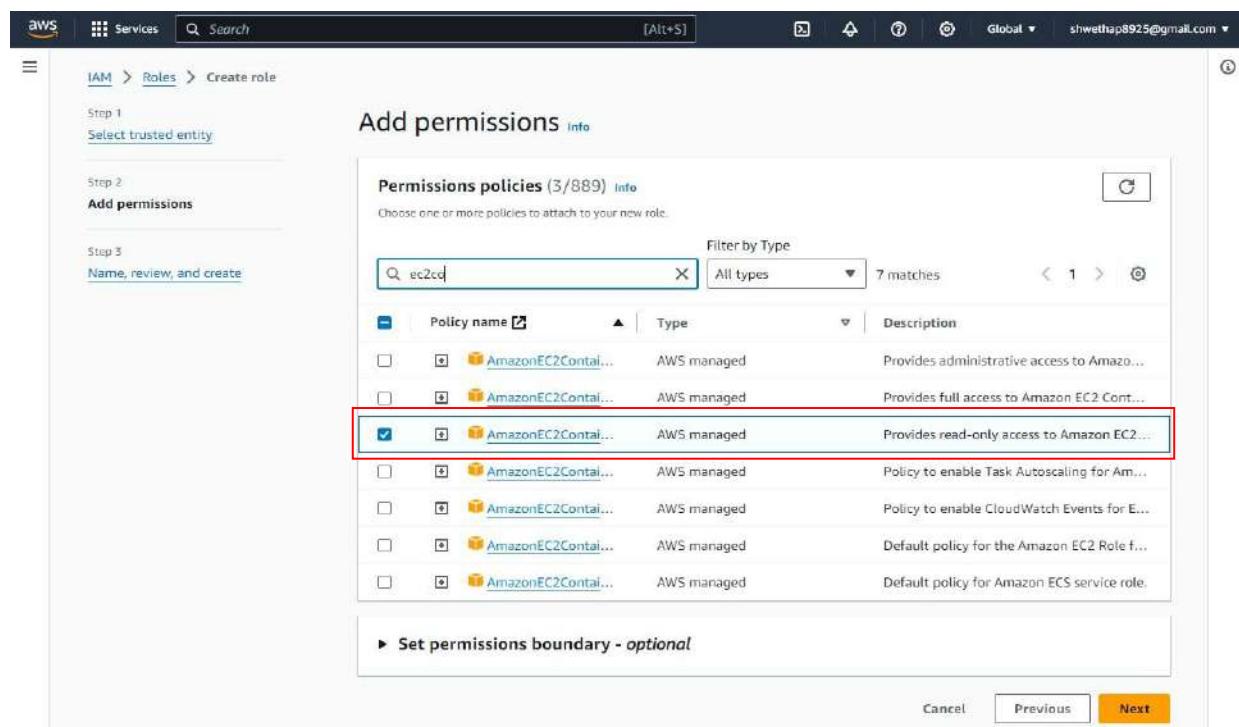
Set permissions boundary - *optional*

Cancel Previous **Next**

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.

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role
Maximum 64 characters. Use alphanumeric and '-' characters.

Description
Add a short explanation for this role.
Maximum 1000 characters. Use alphanumeric and '-' characters.

Step 30: Check that all the permissions of the policies are attached as needed. Click on **CreateRole**.

Permissions policy summary

Policy name	Type	Attached as
AmazonEC2ContainerRegistryReadOnly	AWS managed	Permissions policy
AmazonEKS_CNI_Policy	AWS managed	Permissions policy
AmazonEKSWorkerNodePolicy	AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

You can add up to 50 more tags.

Cancel

Step 31: Now the Role is successfully created.

The screenshot shows the AWS IAM Roles page. At the top, a green banner indicates that 'Role node-grp-role-103 created.' Below this, the 'Roles' section displays 12 entries. One entry, 'eks-cluster-role-103', is highlighted with a red border. The table columns include 'Role name' and 'Trusted entities'. The 'eks-cluster-role-103' row shows 'Role name' as 'eks-cluster-role-103' and 'Trusted entities' as 'AWS Service: eks'. Other entries listed include 'AWSServiceRoleForAmazonEKS', 'AWSServiceRoleForAmazonEKSNodegroup', 'AWSServiceRoleForAutoScaling', and 'AWSServiceRoleForElasticLoadBalancing'.

Role name	Trusted entities
AWSServiceRoleForAmazonEKS	AWS Service: eks (Service-Linked Role)
AWSServiceRoleForAmazonEKSNodegroup	AWS Service: eks-nodegroup (Service)
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service)
eks-cluster-role-103	AWS Service: eks
eks-cluster-role-2023	AWS Service: eks
eks-node-role-2023	AWS Service: ec2

Step 32: Search for eks & click on the created EKS cluster here.

The screenshot shows the same AWS IAM Roles page as before, but with a search filter applied. The search bar contains 'eks' and shows '6 matches'. The 'eks-cluster-role-103' entry is now highlighted with a red border and has a checked checkbox next to it. The other entries remain unselected.

Role name	Trusted entities
AWSServiceRoleForAmazonEKS	AWS Service: eks (Service-Linked Role)
AWSServiceRoleForAmazonEKSNodegroup	AWS Service: eks-nodegroup (Service)
eks-cluster-role-101	AWS Service: eks
<input checked="" type="checkbox"/> eks-cluster-role-103	AWS Service: eks
eks-cluster-role-2023	AWS Service: eks
eks-node-role-2023	AWS Service: ec2

Step 33: View the Cluster info here.

EKS > Clusters > eks-cluster-103

eks-cluster-103

C Delete cluster

ⓘ End of support for Kubernetes version 1.25 is May 2024. If you don't update your cluster to a later version before that date, it will automatically enter extended support. After the extended support preview ends, clusters on versions in extended support will be subject to additional fees. [Learn more](#)

ⓘ New versions are available for 2 add-ons. X

▼ Cluster info [Info](#)

Status Active	Kubernetes version Info 1.25	Support type Standard support until May 2024	Provider EKS
----------------------------	--	---	--------------

Step 34: Select Compute & click on Add node group option here.

< Overview | Resources | **Compute** | Networking | Add-ons | Authentication | Observability >

Nodes (0) [Info](#)

Filter Nodes by property or value

Node name	▲ Instance type	▼ Node group	▼ Created	▼ Status
No Nodes				
This cluster does not have any Nodes, or you don't have permission to view them.				

Node groups (0) [Info](#)

Edit Delete Add node group

Group name ▲ Desired size ▽ AMI release version ▽ Launch template ▽ Status ▽

No node groups

This cluster does not have any node groups.

Nodes that are not part of an Amazon EKS managed node group are not shown in the AWS console.

Add node group

Step 35: Now give the name as **eks-node-grp-104** & select the Node IAM role for that as shown below.

Configure node group Info

A node group is a group of EC2 instances that supply compute capacity to your Amazon EKS cluster. You can add multiple node groups to your cluster.

Node group configuration

These properties cannot be changed after the node group is created.

Name

Assign a unique name for this node group.

The node group name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 63.

Node IAM role Info

Select the IAM role that will be used by the nodes. To create a new role, go to the IAM console.



ⓘ The selected role must not be used by a self-managed node group as this could lead to a service interruption upon managed node group deletion.

[Learn more](#)

Step 36: Give the Node group scaling configuration to **1** here & click on **Next** here.

Node group scaling configuration

Desired size

Set the desired number of nodes that the group should launch with initially.

 nodes

Desired node size must be greater than or equal to 0.

Minimum size

Set the minimum number of nodes that the group can scale in to.

 nodes

Minimum node size must be greater than or equal to 0.

Maximum size

Set the maximum number of nodes that the group can scale out to.

 nodes

Maximum node size must be greater than or equal to 1 and cannot be lower than the minimum size.

Node group update configuration Info

Maximum unavailable

Set the maximum number or percentage of unavailable nodes to be tolerated during the node group version update.

 Number

Enter a number

 Percentage

Specify a percentage

Value

 node

Node count must be greater than 0.

[Cancel](#)[Previous](#)[Next](#)

Step 37: Click on **Next** here.

Step 1
[Configure node group](#)

Step 2
[Set compute and scaling configuration](#)

Step 3
Specify networking

Step 4
Review and create

Specify networking

Node group network configuration
These properties cannot be changed after the node group is created.

Subnets [Info](#)
Specify the subnets in your VPC where your nodes will run. To create a new subnet, go to the corresponding page in the [VPC console](#).

Select subnets [▼](#) [C](#)

subnet-0e2fe43efef13b063 X subnet-0f22fff4ce49ffc57 X

subnet-0c9de3845972c1a1c X

Configure remote access to nodes [Info](#)

Cancel Previous **Next**

Step 38: Now click on **Next** here.

Node group scaling configuration

Desired size	Minimum size	Maximum size
1 node	1 node	1 node

Node group update configuration

Maximum unavailable
1 node

Step 3: Networking [Edit](#)

Node group network configuration

Subnets
subnet-0e2fe43efef13b063
subnet-0f22fff4ce49ffc57
subnet-0c9de3845972c1a1c

Configure remote access to nodes
off

Cancel Previous **Create**

Step 39: Now the node group is successfully created.

eks-node-grp-104

Node group configuration [Info](#)

Kubernetes version 1.25	AMI type Info AL2_x86_64	Status Creating
AMI release version Info 1.25.15-20231116	Instance types t3.medium	Disk size 20 GiB

Details [Nodes](#) [Health issues](#) [0](#) [Kubernetes labels](#) [Update config](#) [Kubernetes taints](#) [Update history](#) [Tags](#)

Details

Node group ARN arn:aws:eks:eu-north-1:194767121857:nodegroup/eks-cluster-103/eks-node-grp-104/c1c60270-b46f-1488-fb75-cfc8a27067d9	Autoscaling group name ASG	Capacity type On-Demand	Subnets subnet-0e2fe43efef13b063 subnet-0f22ff14cc49fc57 subnet-0c9de3845972c1a1c
Created a few seconds ago	Node IAM role ARN arn:aws:iam::194767121857:role/node-grp-role-103	Desired size 1 node	Configure remote access to nodes off
		Minimum size 1 node	
		Maximum size 1 node	

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 install 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
```

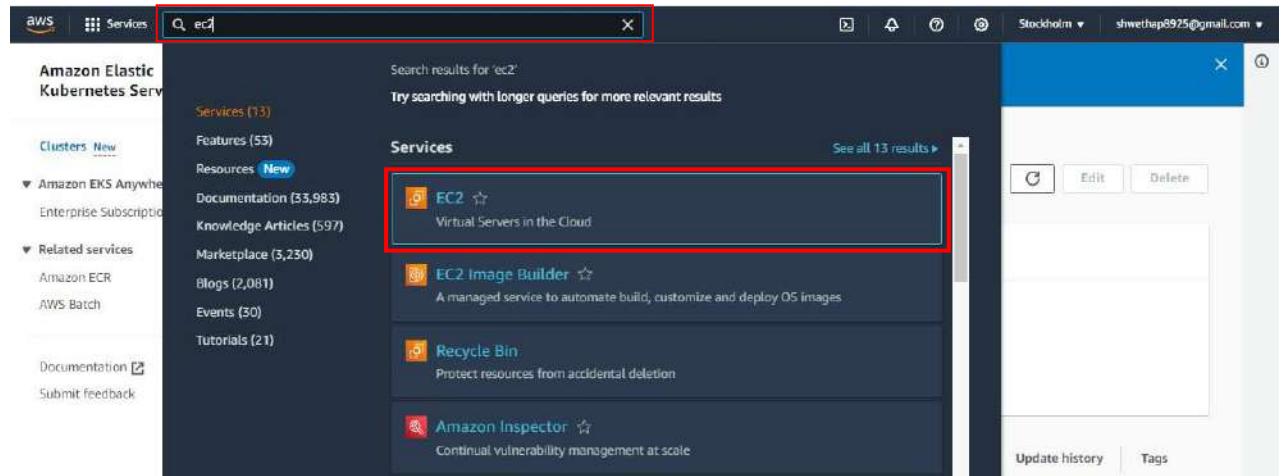
Enter key ctrl+o Enter key ctrl x

```
$ kubectl apply -f mygame-svc.yaml
```

```
$ kubectl describe svc mygame-svc
```

```
[cloudshell-user@ip-10-6-71-201 ~]$ aws sts get-caller-identity
{
    "UserId": "194767121857",
    "AccessKeyId": "AKIAJZL7V2XWQH7D6UQ",
    "Arn": "arn:aws:iam::194767121857:root"
}
[cloudshell-user@ip-10-6-71-201 ~]$ aws eks update-kubeconfig --region eu-north-1 --name eks-cluster-103
Added new config: arn:aws:eks:eu-north-1:194767121857:cluster/eks-cluster-103 to /home/cloudshell-user/.kube/config
[cloudshell-user@ip-10-6-71-201 ~]$ cat .kube/config
cat: .kube/config: No such file or directory
[cloudshell-user@ip-10-6-71-201 ~]$ sudo yum install nano -y
Loaded plugins: ovl, priorities
amzn2-core
Package nano-2.9.8-2.amzn2.0.1.x86_64 already installed and latest version
nothing to do
[cloudshell-user@ip-10-6-71-201 ~]$ nano 2048.pod.yaml
[cloudshell-user@ip-10-6-71-201 ~]$ kubectl apply -f 2048.pod.yaml
pod/2048-pod created
[cloudshell-user@ip-10-6-71-201 ~]$ kubectl get pods
NAME          STATUS    RESTARTS   AGE
2048-pod      Running   0          19s
[cloudshell-user@ip-10-6-71-201 ~]$ nano mygame-svc.yaml
[cloudshell-user@ip-10-6-71-201 ~]$ kubectl apply -f mygame-svc.yaml
service/mygame-svc created
[cloudshell-user@ip-10-6-71-201 ~]$ kubectl describe svc mygame-svc
Name:           mygame-svc
Namespace:      default
Labels:          <none>
Annotations:   <none>
Selector:       app=2048.ws
Type:           LoadBalancer
IP Family Policy: Singlestack
IP Families:   IPv4
IP:             10.100.81.12
TCPI:           10.100.81.12
LoadBalancer Ingress: ac36aeef7636464c92a077260ec8179c9-61219317.eu-north-1.elb.amazonaws.com
Port:            80/TCP
TargetPort:     80/TCP
NodePort:      32400/TCP
Endpoints:     172.31.7.108:80
Session Affinity: None
External Traffic Policy: Cluster
Events:
  Type  Reason  Age   From            Message
  ----  -----  --   --              --
  Normal  EnsuringLoadBalancer  22s  service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer  19s  service-controller  Ensured load balancer
[cloudshell-user@ip-10-6-71-201 ~]$
```

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.

The screenshot shows the AWS EC2 Load Balancers console. On the left sidebar, under the 'Load Balancing' section, 'Load Balancers' is selected. In the main pane, there are two load balancers listed:

- af9ee88a411e4bd2b...** (unchecked)
- ac36aee7636464c92a077260ec8179c9** (checked)

A modal window titled 'Load balancer: ac36aee7636464c92a077260ec8179c9' is open, displaying the following details:

Details	Listeners	Network mapping	Security	Health checks	Target instances	Monitoring	Attributes	Tags								
Details <table border="1"> <tr> <td>Load balancer type: Classic</td> <td>Status: 1 of 1 instance in service</td> <td>VPC: vpc-0de6d80806263e3f6</td> <td>Date created: November 25, 2023, 10:51 (UTC+05:30)</td> </tr> <tr> <td>Scheme: Internet-facing</td> <td>Hosted zone: Z23TAZ6LKFNMIO</td> <td>Availability Zones: subnet-02fe43efef13b063, eu-north-1c (eun1-az3), subnet-0f2fff4ce49fffc57, eu-north-1b (eun1-az2), subnet-0c9de3845972c1a1c, eu-north-1a (eun1-az1)</td> <td></td> </tr> </table>									Load balancer type: Classic	Status: 1 of 1 instance in service	VPC: vpc-0de6d80806263e3f6	Date created: November 25, 2023, 10:51 (UTC+05:30)	Scheme: Internet-facing	Hosted zone: Z23TAZ6LKFNMIO	Availability Zones: subnet-02fe43efef13b063, eu-north-1c (eun1-az3), subnet-0f2fff4ce49fffc57, eu-north-1b (eun1-az2), subnet-0c9de3845972c1a1c, eu-north-1a (eun1-az1)	
Load balancer type: Classic	Status: 1 of 1 instance in service	VPC: vpc-0de6d80806263e3f6	Date created: November 25, 2023, 10:51 (UTC+05:30)													
Scheme: Internet-facing	Hosted zone: Z23TAZ6LKFNMIO	Availability Zones: subnet-02fe43efef13b063, eu-north-1c (eun1-az3), subnet-0f2fff4ce49fffc57, eu-north-1b (eun1-az2), subnet-0c9de3845972c1a1c, eu-north-1a (eun1-az1)														

Step 43: Then copy the DNS name as shown below.

The screenshot shows the AWS Classic Load Balancer console. The left sidebar lists various services, and the 'Load Balancers' section is selected. A modal window is open for the load balancer 'ac36aee7636464c92a077260ec8179c9'. The 'Details' tab is active, showing the following configuration:

Load balancer type: Classic	Status: 1 of 1 instance in service	VPC: vpc-0de6d80806263e3f6	Date created: November 25, 2023, 10:51 (UTC+05:30)
Scheme: Internet-facing	Hosted zone: Z23TAZ6LKFNMIO	Availability Zones: subnet-02fe43efef13b063, eu-north-1c (eun1-az3), subnet-0f2fff4ce49fffc57, eu-north-1b (eun1-az2), subnet-0c9de3845972c1a1c, eu-north-1a (eun1-az1)	

A message at the bottom of the modal says 'DNS name copied' with a green checkmark icon. Below it, the copied DNS name 'ac36aee7636464c92a077260ec8179c9-611219317.eu-north-1.elb.amazonaws.com (A Record)' is highlighted with a red box. A note at the bottom right says 'Your Classic Load Balancer can now be migrated in just a few steps. Migration wizard helps by using your load balancer's current configuration to create an Application Load Balancer or Network Load Balancer. Select Launch migration wizard to get started. For the list of Elastic Load Balancing features, including which load balancers support them, see Comparison of Elastic Load Balancing Products.'

Step 44: Now paste the DNS name copied from step 43 in Chrome & play the game displayed

below.

