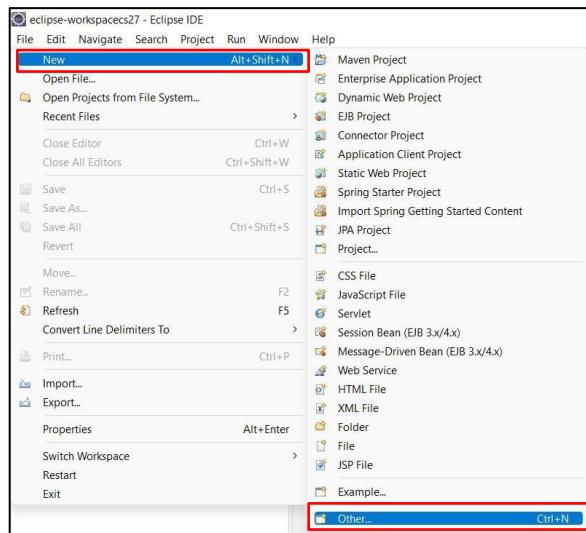


WEEK – 9

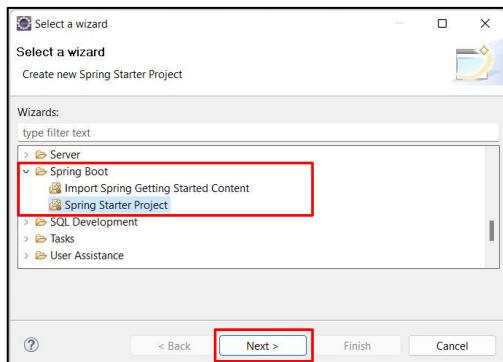
1. Build user authentication flow and authorization using SpringSecurity.

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

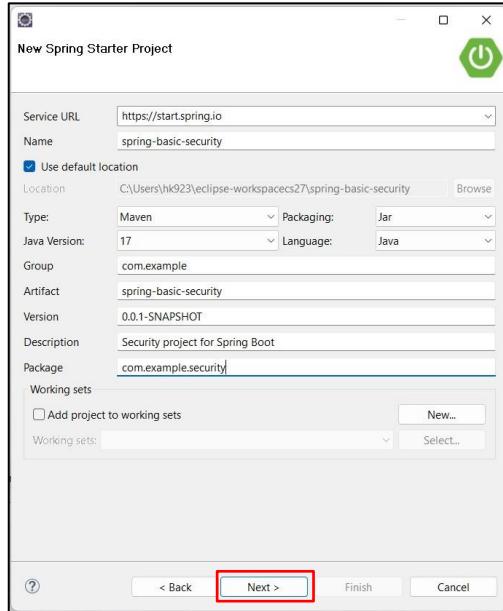
Now search for ‘Spring tool suite’ & click on ‘Install’ to install the latest version. After the installation is complete, go to → File → New → Other option as shown below.



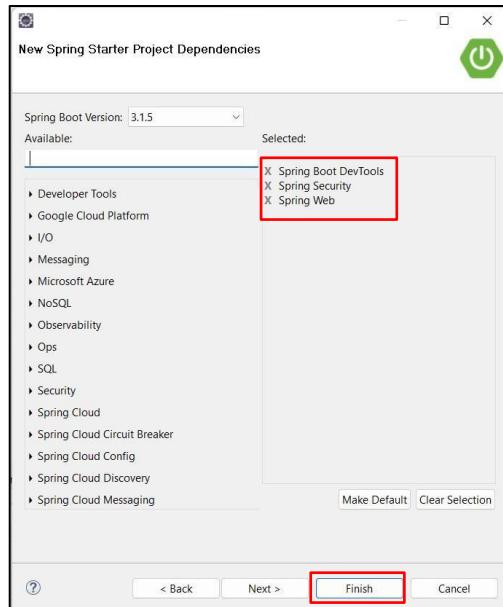
Step 2: Now select ‘Spring Starter Project’ & click on ‘Next’ option.



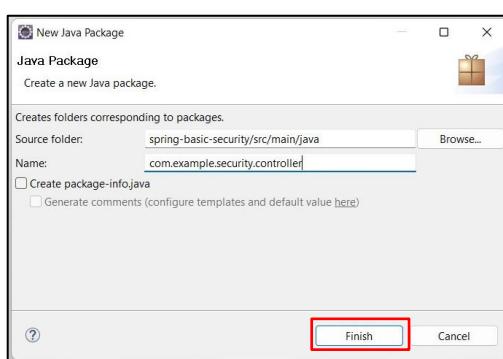
Step 3: Give the Name as ‘User’ → select ‘Maven’ → Description as User Registration Spring Boot Application → Package as ‘spring-basic-security’ & then click on ‘Next’.



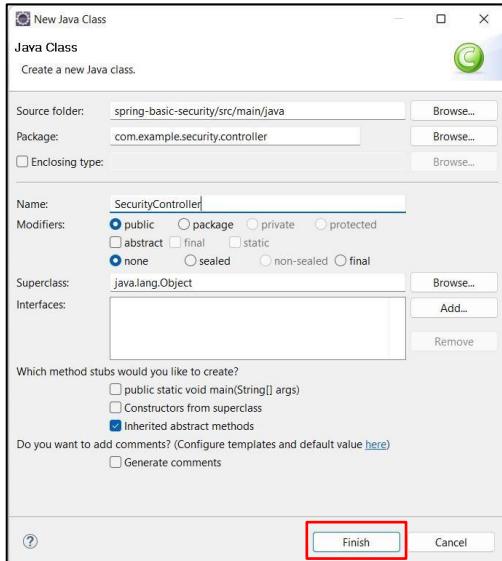
Step 4: Now add all the required dependencies as shown below & click on ‘Finish’.



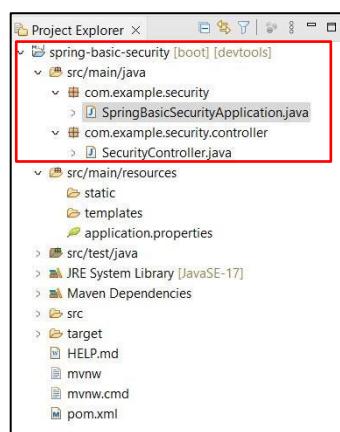
Step 5: Create a new package named ‘com.example.security.controller’ & click on ‘Finish’.



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 × 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 }
```

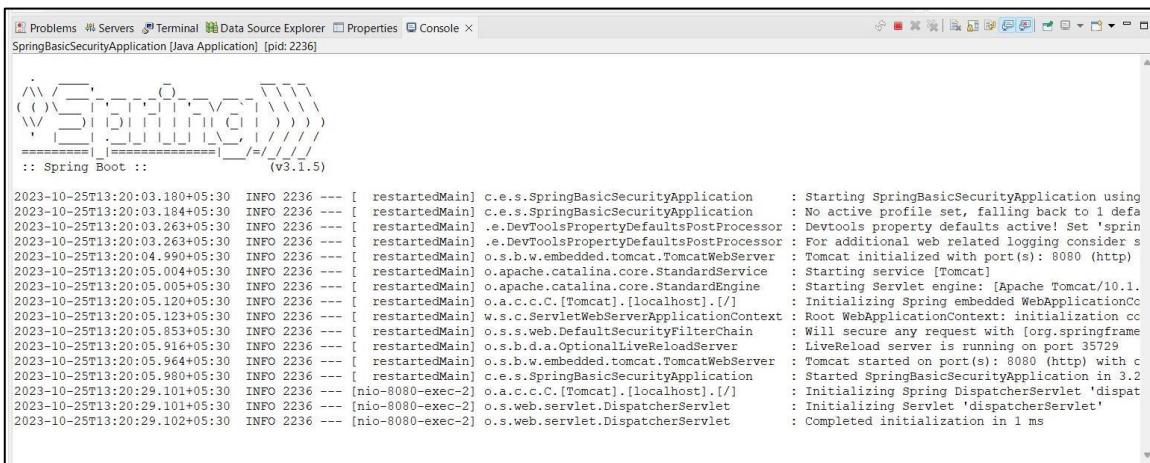
Step 9: Now add code to SpringBasicSecurityApplication.java file as shown below.

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

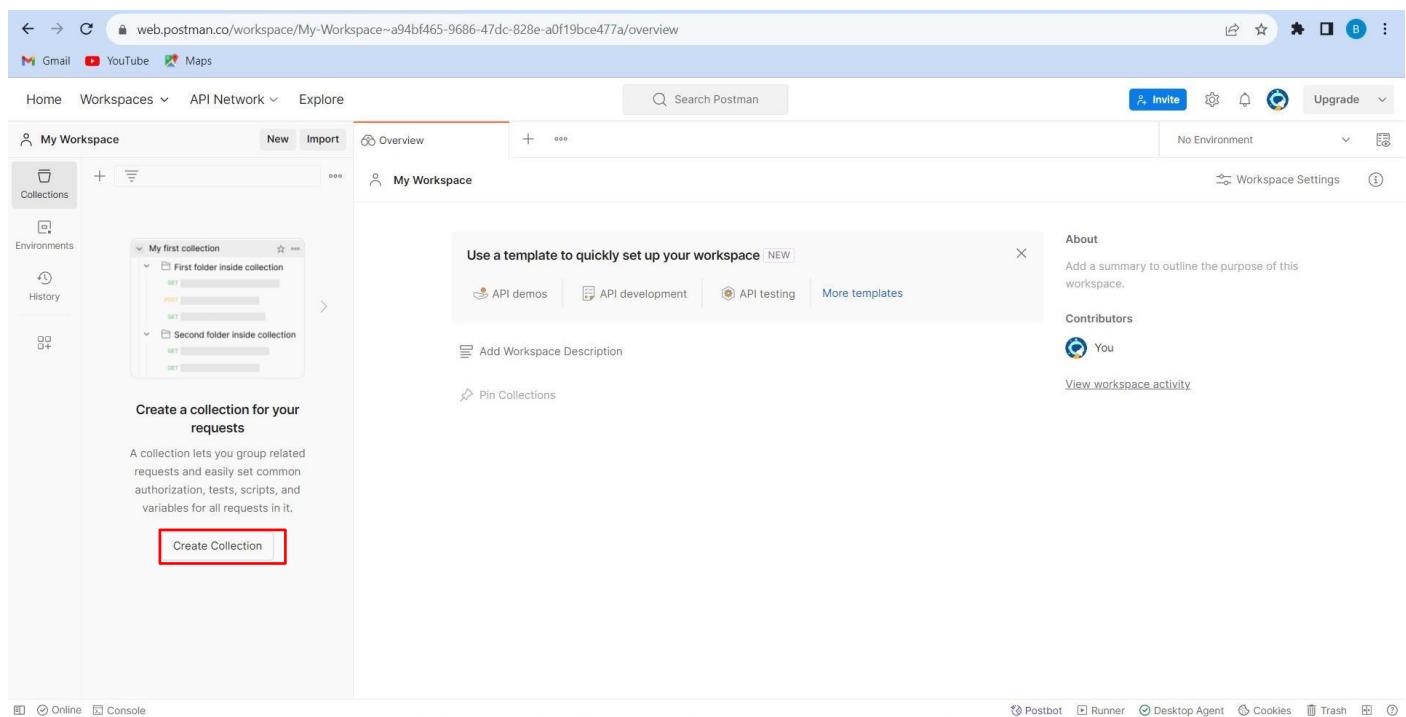
Step 11: Start the server as shown below.



The screenshot shows the Eclipse IDE interface with the 'Servers' view selected. The central pane displays the log output for the 'SpringBasicSecurityApplication [Java Application]'. The log includes Spring Boot initialization messages, Tomcat startup, and various system and application logs. Key messages include:

```
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 'sprin
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.
2023-10-25T13:20:05.853+05:30 INFO 2236 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Initializing Spring embedded WebApplicationCo
2023-10-25T13:20:05.853+05:30 INFO 2236 --- [ restartedMain] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframe
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
2023-10-25T13:20:29.101+05:30 INFO 2236 --- [nio-8080-exec-2] o.a.c.c.C.[Tomcat].[localhost].[] : 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 Spring DispatcherServlet 'dispat
2023-10-25T13:20:29.102+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 'My Workspace' containing 'Collections' (selected), 'Environments', and 'History'. The main area shows a tree view of collections: 'My first collection' with 'First folder inside collection' and 'Second folder inside collection', each containing several GET requests. 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'. Below the modal, there are buttons for 'Add Workspace Description' and 'Pin Collections'. To the right, there are sections for 'About' (with a placeholder for a workspace summary), 'Contributors' (listing 'You'), and a link to 'View workspace activity'. At the bottom, there are buttons for 'Postbot', 'Runner', 'Desktop Agent', 'Cookies', 'Trash', and other workspace management options.

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 with a collection named 'Security'. A GET request is made to 'localhost:8080/'. The 'Send' button is highlighted with a red box. The response status is '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 dropdown menu is open, and 'Basic Auth' is highlighted with a red box. Other options include Inherit auth from parent, No Auth, API Key, Bearer Token, JWT Bearer, Digest Auth, OAuth 1.0, OAuth 2.0, Hawk Authentication, AWS Signature, and NTLM Authentication [Beta].

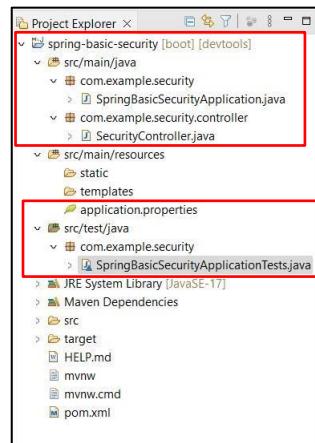
Step 15: Now enter Username & Password as mentioned below. Then Save it & click on Send to view the output.

The screenshot shows the Postman interface. A GET request is made to `localhost:8080`. In the Authorization tab, 'Basic Auth' is selected with 'Username' set to `user` and 'Password' set to `password`. The response status is `200 OK`, and the response body contains the message `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’.



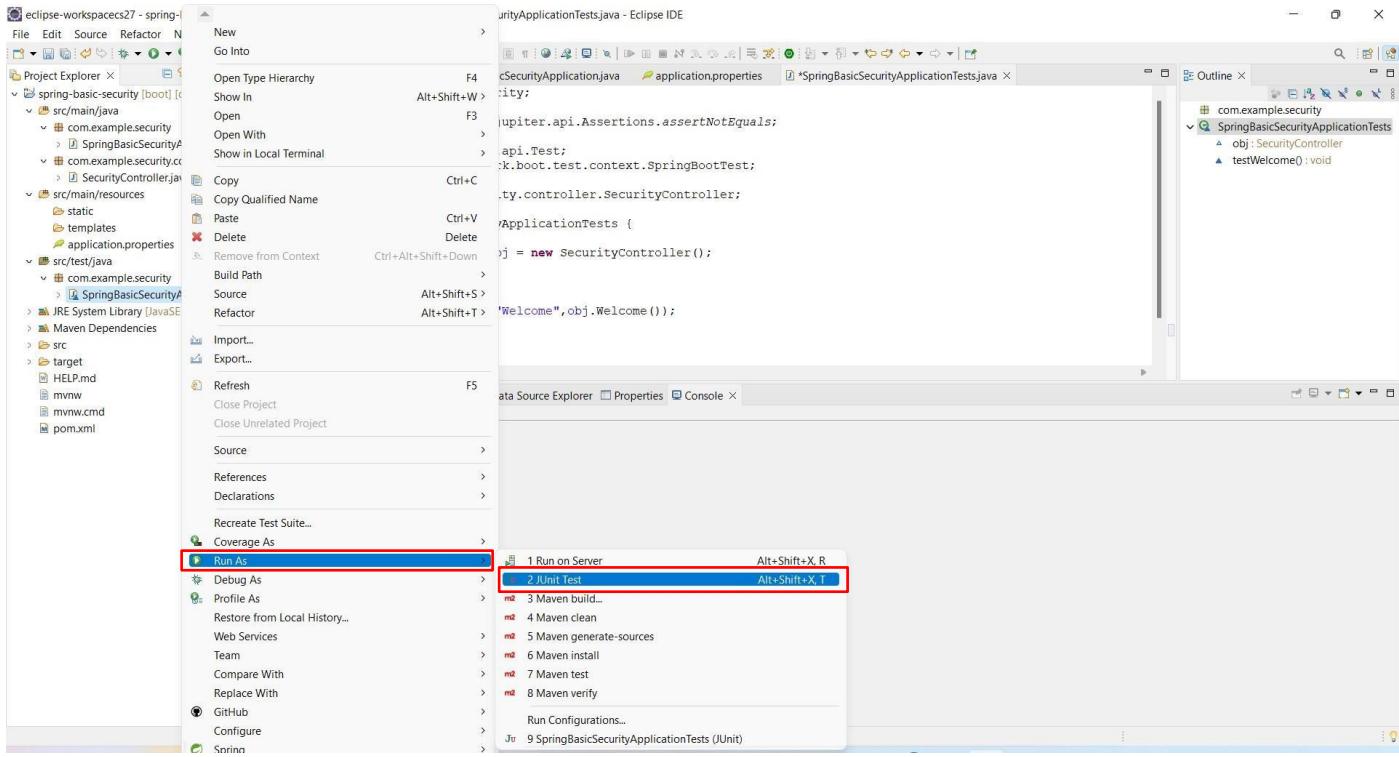
Step 17: Now add code to ‘SpringBasicSecurityApplicationTests.java’ as shown below.

```

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     }
19 }

```

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.SpringBootTestContextBootstrapper -- 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.web.DefaultSecurityFilterChain : Will secure any request with [org.springframe
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 JUnit tab. At the top, it displays 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. Below this, a green progress bar is fully filled. The main area shows a single test case: 'SpringBasicSecurityApplicationTests [Runner: JUnit 5] (1.138 s)' with a single method 'testWelcome() (1.138 s)'. The status for this method is also green, indicating it passed.

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 query 'junit 4' is entered in the search bar. The top result is a link to the JUnit website: [JUnit](https://junit.org/junit4). The title of the result is 'JUnit – About'. A red box highlights this link. Below the title, a snippet of text describes JUnit as a simple framework for writing repeatable tests. Other links in the snippet include 'JUnit 4.13.2 API', 'Cookbook', 'Dependency Information', and 'Project License'. The page also shows 'About 3,51,00,000 results (0.34 seconds)' and various search filters like 'All', 'Videos', 'Images', 'Shopping', 'News', and 'More'.

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

A screenshot of the JUnit official website. The main navigation bar includes 'Welcome', 'Usage and Idioms', and 'Third-party extensions'. The 'Welcome' section has a 'Download and install' link, which is highlighted with a red box. 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 lists various testing concepts like Assertions, Test Runners, and Matchers. The 'Third-party extensions' section lists various tools and libraries that integrate with JUnit.

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' section contains instructions for downloading JUnit. It lists two options: 'Plain-old JAR' and 'Maven'. Under 'Plain-old JAR', it says to download the JARs and add them to your test classpath. Two files are listed: [junit.jar](#) and [hamcrest-core.jar](#), both of which are highlighted with a red box. Under 'Maven', it says to add a dependency to `junit:junit` in `test` scope. On the right side, there is a sidebar with a table of contents containing links to various JUnit features like 'Pages 35', '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-sources.jar.asc	2021-02-13 16:31	833
junit-4.13.2-sources.jar.asc.md5	2021-02-13 16:31	32
junit-4.13.2-sources.jar.asc.sha1	2021-02-13 16:31	40
junit-4.13.2-sources.jar.md5	2021-02-13 16:31	32
junit-4.13.2-sources.jar.sha1	2021-02-13 16:31	40
junit-4.13.2.jar	2021-02-13 16:31	384581
junit-4.13.2.jar.asc	2021-02-13 16:31	833
junit-4.13.2.jar.md5	2021-02-13 16:31	32
junit-4.13.2.jar.asc.sha1	2021-02-13 16:31	40
junit-4.13.2.jar.md5	2021-02-13 16:31	32
junit-4.13.2.jar.sha1	2021-02-13 16:31	40
junit-4.13.2.pom	2021-02-13 16:31	27018
junit-4.13.2.pom.asc	2021-02-13 16:31	833
junit-4.13.2.pom.asc.md5	2021-02-13 16:31	32
junit-4.13.2.pom.asc.sha1	2021-02-13 16:31	40
junit-4.13.2.pom.md5	2021-02-13 16:31	32
junit-4.13.2.pom.sha1	2021-02-13 16:31	40

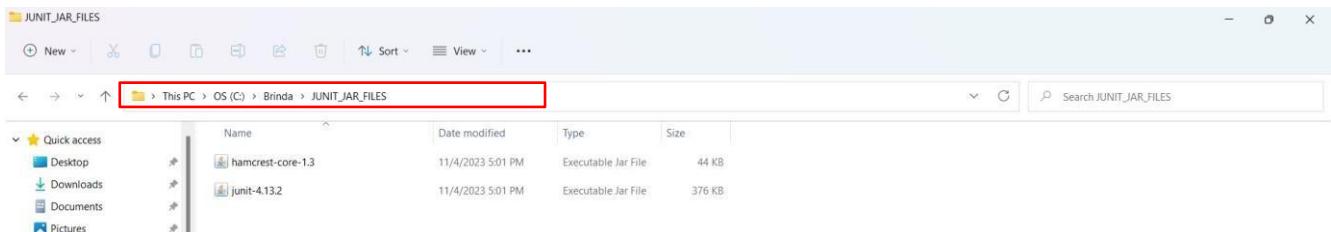
Step 5: Also select ‘hamcrest-core.jar’ file and download the ‘hamcrest-core-1.3.jar’ version here.

org/hamcrest/hamcrest-core/1.3

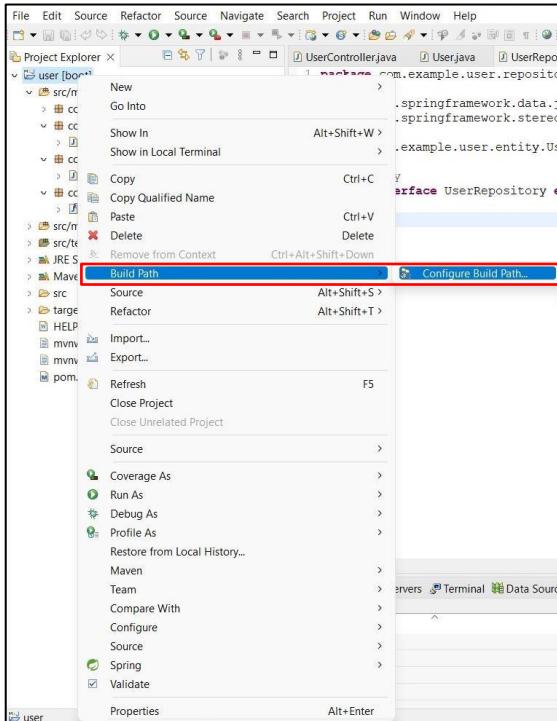
..		
hamcrest-core-1.3-javadoc.jar	2012-07-09 21:08	242519
hamcrest-core-1.3-javadoc.jar.asc	2012-07-09 21:08	499
hamcrest-core-1.3-javadoc.jar.asc.md5	2012-07-09 21:08	32
hamcrest-core-1.3-javadoc.jar.asc.sha1	2012-07-09 21:08	40
hamcrest-core-1.3-javadoc.jar.md5	2012-07-09 21:08	32
hamcrest-core-1.3-javadoc.jar.sha1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

Step 6: Now select any drive & create a new folder within that folder, create a new folder named

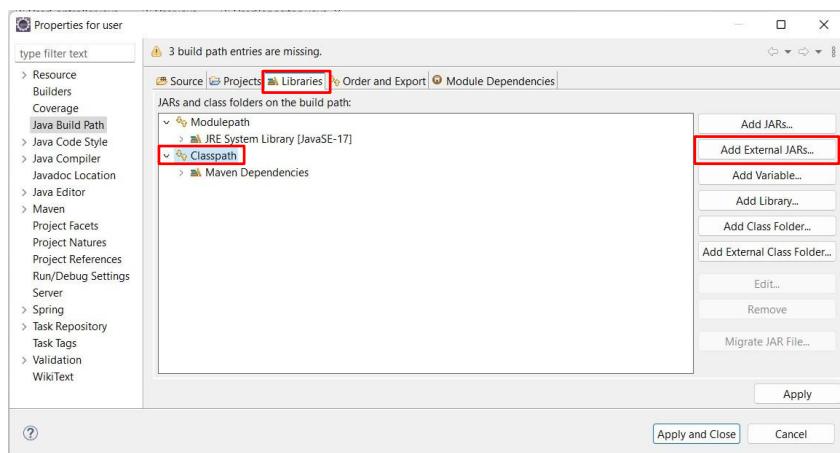
‘JUNIT_JAR_FILES’ & copy the downloaded jar files & paste it here.



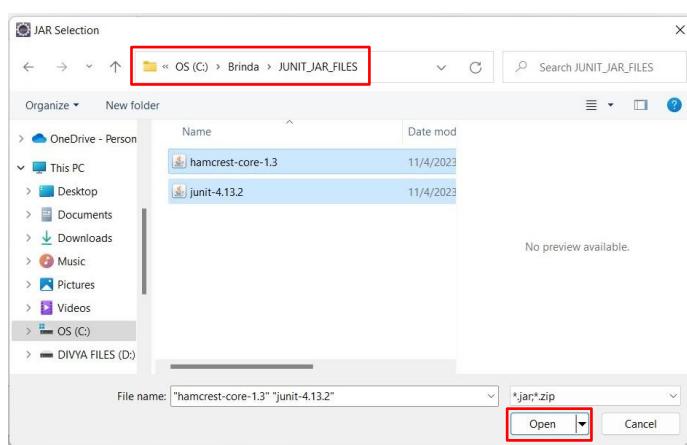
Step 7: Go to Eclipse user Build Path Configure Build Path here.



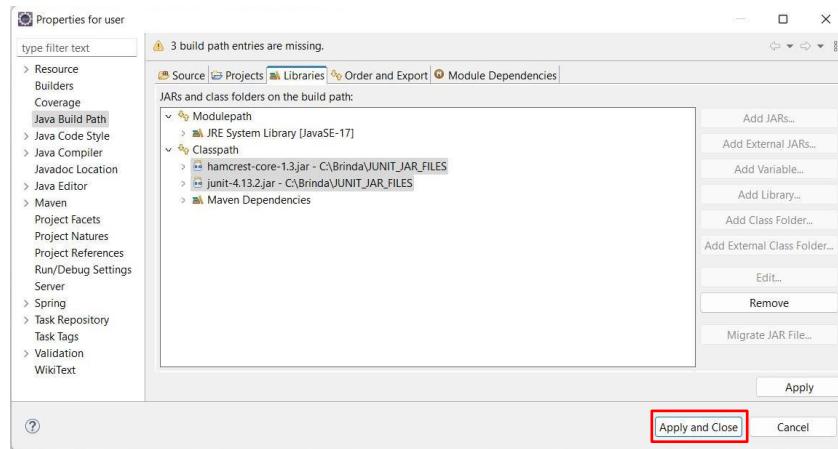
Step 8: Go to Libraries □ Classpath □ Add External JARs.. here.



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



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



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

```
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.

#	Time	Action	Message	Duration / Fetch
1	2 17:24:56	select * from junit.cs LIMIT 0, 1000	0 row(s) returned	0.016 sec / 0.000 sec
2	3 18:12:18	use junit	0 row(s) affected	0.015 sec
3	4 18:12:26	select * from user LIMIT 0, 1000	Error Code: 1146. Table 'junit.user' doesn't exist	0.000 sec
4	5 18:12:39	select * from user LIMIT 0, 1000	Error Code: 1146. Table 'junit.user' doesn't exist	0.000 sec
5	6 18:19:08	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
6	7 18:24:59	select * from user LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
7	8 18:26:51	select * from user LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
8	9 18:27:22	select * from user LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
9	10 19:38:18	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 'user [boot]' project, which contains a 'src/main/java' folder with 'com.example.user' and 'UserApplication.java'. The main editor window on the right displays the 'UserController.java' code. The code defines a REST controller for users, using annotations like @RestController, @RequestMapping, @GetMapping, and @PostMapping. It includes imports for java.util.List, org.springframework.beans.factory.annotation.Autowired, org.springframework.http.ResponseEntity, org.springframework.web.bind.annotation.DeleteMapping, org.springframework.web.bind.annotation.GetMapping, org.springframework.web.bind.annotation.PathVariable, org.springframework.web.bind.annotation.PostMapping, org.springframework.web.bind.annotation.PutMapping, org.springframework.web.bind.annotation.RequestBody, org.springframework.web.bind.annotation.RequestMapping, org.springframework.web.bind.annotation.RestController, com.example.user.entity.User, and com.example.user.repository.UserRepository.

```
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.

The screenshot shows the 'UserController.java' code with several new annotations and methods added. Lines 36 through 40 show a @PostMapping for creating a user. Lines 41 through 50 show a @PutMapping for updating a user by ID. Line 51 shows a @DeleteMapping for deleting a user by ID. The code uses @RequestBody to map the incoming JSON to a User object and @PathVariable to extract the user ID from the URL.

```
35
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     ex.setEmailID(user.getEmailID());
48     return this.userRepository.save(ex);
49 }
50     @DeleteMapping("/{id}")
51     public ResponseEntity<User> deleteUser(@PathVariable("id") long userId) {
52     }
53     User ex=this.userRepository.findById(userId).orElseThrow();
54     this.userRepository.delete(ex);
55     return ResponseEntity.ok().build();
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

```

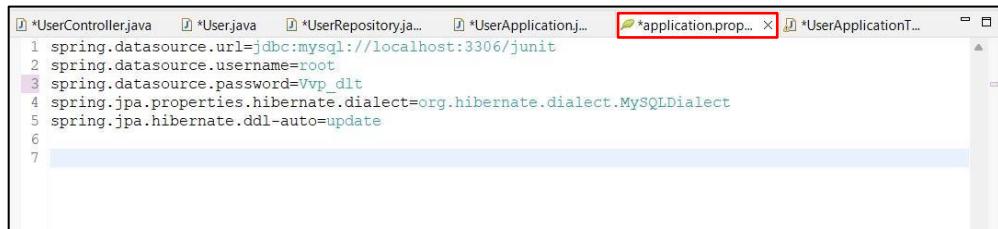
Step 17: Now add code to ‘UserApplication.java’ file where main method class present here.

```

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.



```
spring.datasource.url=jdbc:mysql://localhost:3306/junit
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 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.



```
package com.example.user;

import static org.assertj.core.api.Assertions.assertThat;
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.annotation.Rollback;
import org.springframework.transaction.annotation.Transactional;
import com.example.user.model.User;
import com.example.user.repository.UserRepository;

@SpringBootTest
public class UserApplicationTests {

    @Autowired
    UserRepository userRepo;

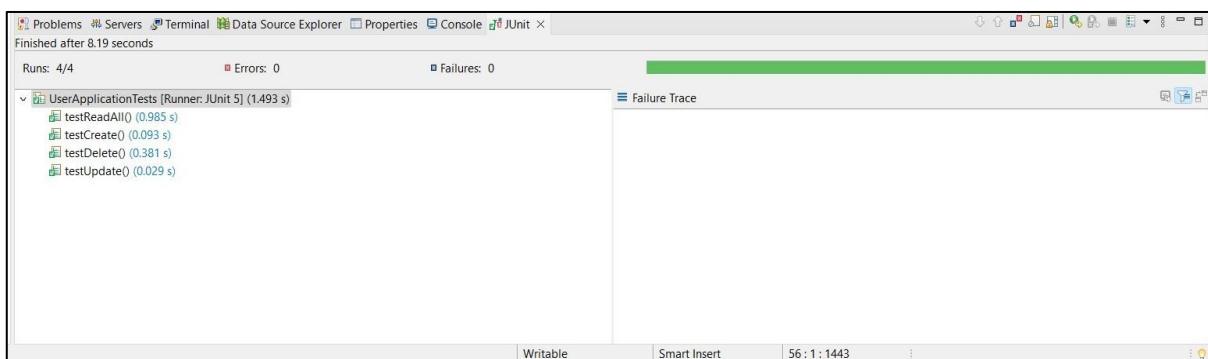
    @Test
    public void testCreate() {
        User u = new User();
        u.setId(1L);
        u.setFirstname("VBSDSD");
        u.setLastname("FSD");
        u.setEmailID("DDDD@1234.com");
        userRepo.save(u);
        assertThat(userRepo.findById(1L).get());
    }

    @Test
    public void testReadAll() {
        List<User> list = userRepo.findAll();
        assertThat(list.size()).isGreaterThan(0);
    }

    @Test
    public void testUpdate() {
        User u = userRepo.findById(106L).get();
        u.setFirstname("Computer");
        u.setLastname("Science");
        userRepo.save(u);
        assertEquals("Computer", userRepo.findById(106L).get().getFirstname());
    }

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

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

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: Result Grid Form Editor Field Types

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

	id	email	firstname	lastname
1	DDDD@1234.com	VVPSSD	FSD	
102	vvp123@gmail.com	JAVA	VVP	
103	Bri123@gmail.com	Brinda	K	
104	Bhoomi123@gmail.com	Bhoomika	R	
105	Deeputhi123@gmail.com	Deepthi	M	
106	comp456@gmail.com	Computer	Science	
152	DDDD@1234.com	VVPSSD	FSD	
202	DDDD@1234.com	VVPSSD	FSD	
*	HULL	HULL	HULL	HULL

user 7

Action Output

#	Time	Action	Message	Duration / Fetch
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	18:19:00	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=Awr.xZefkZIBPoTnZJXNyA;_ylc=X1MDMjc2NjY3OQRfcgMyBGZyA21jYWZlZQRmcjdC2ltG9wBgdwcmIka2tCQjptM1VRUj5DRmZRSWxNczlqbEEbI9... Sign In yahoo!

mongodb download

All Images Videos News More Anytime

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

mongo.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

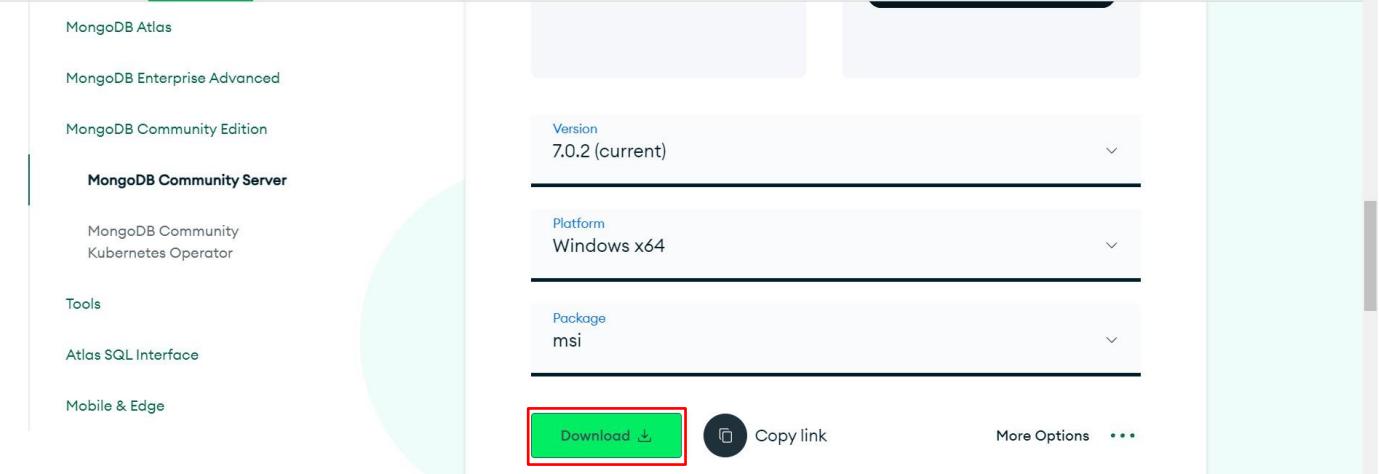
Stock price: MDB (NasdaqGM)
\$343.11
+14.11 (+4.29%)
As of Fri, Nov 3, 2023 3:00PM EST
Market closed.

More on Yahoo Finance

English (United States)
English (India)

To switch input methods, press Windows key + space.

Step 2: Click on ‘Download’ here.

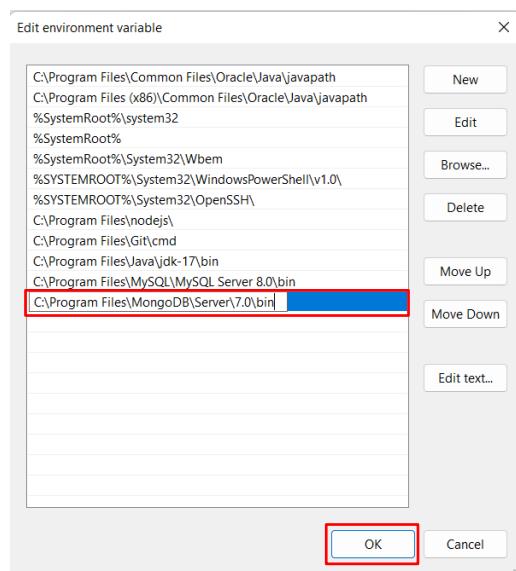


MongoDB Atlas
MongoDB Enterprise Advanced
MongoDB Community Edition
MongoDB Community Server
MongoDB Community
Kubernetes Operator
Tools
Atlas SQL Interface
Mobile & Edge

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.

About 185,000 search results

www.mongodb.com/try/download ✓
MongoDB Compass Download (GUI) | MongoDB ✓

MongoDB Compass Download (GUI) Easily explore and manipulate your database with **Compass**, the GUI for MongoDB. Intuitive and flexible, **Compass** provides detailed schema visualizations, rea...

MongoDB Atlas ✓ Work with your data as code Documents in MongoDB map...

Community Server ✓ Download MongoDB Community Server non-relational database to...

MongoDB Shell ✓ MongoDB Compass Download (GUI) Easily explore and manipulate...

M001 ✓ Introduction to MongoDB. The Introduction to MongoDB course...

Pricing ✓ MongoDB Product Pricing. App Services. All App Services in...

Compass ✓ Easily work with your data in Compass, the GUI built by -...

People also search for

- mongodb compass download gui for windows
- mongodb compass download gui 64-bit
- mongodb community server download
- postman download

mongodb compass download gui free
mongodb compass download gui version
mongodb download
mongodb compass

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

About 257,000 search results

www.mongodb.com/try/download ✓
MongoDB Shell Download | MongoDB ✓

The **MongoDB Shell** is a modern command-line experience, full with features to make it easier to work with your database. **Free download**. Try now!

Category: Data Platform

www.mongodb.com/docs/mongodb-shell/ ✓
Install mongosh – MongoDB Shell ✓

You can use the **MongoDB Shell** to connect to MongoDB version 4.2 or greater. Supported Operating Systems You can install MongoDB Shell 2.0.0 on these operating systems:

www.mongodb.com/products/tools ✓
MongoDB Shell | MongoDB ✓

MongoDB Shell. Best-in-class Database Shell Experience. The quickest way to connect to MongoDB and Atlas to work with your data and manage your data platform. **Download Now**. Contact sales....

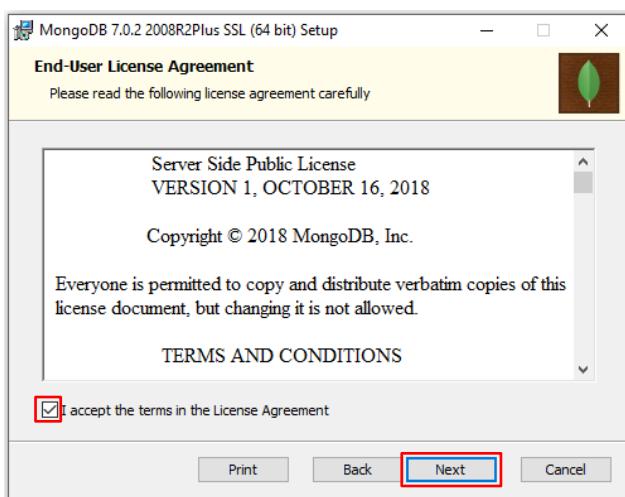
www.mongodb.com/docs/mongodb-shell/ ✓
Welcome to MongoDB Shell (mongosh) – MongoDB Shell ✓

The **MongoDB Shell**, mongosh, is a JavaScript and Node.js REPL environment for interacting with MongoDB deployments in Atlas, locally, or on another remote host. Use the **MongoDB Shell** to test...

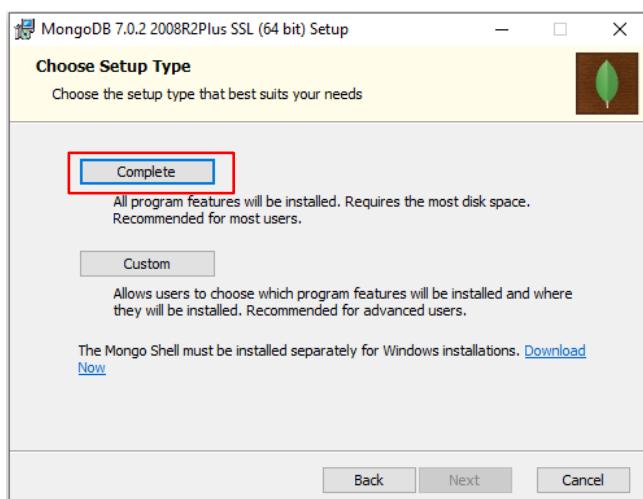
Step 5: Follow the ongoing steps to install mongodb compass, click on ‘Next’.



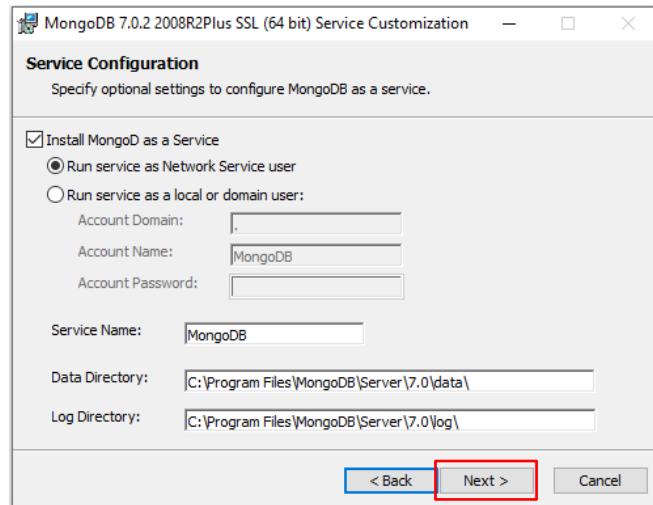
Step 6: Enable the checkbox & click on ‘Next’.



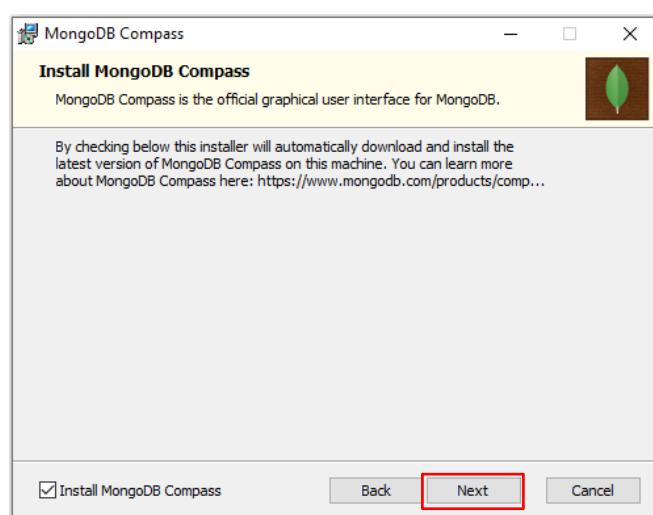
Step 7: Click on ‘complete’ here.



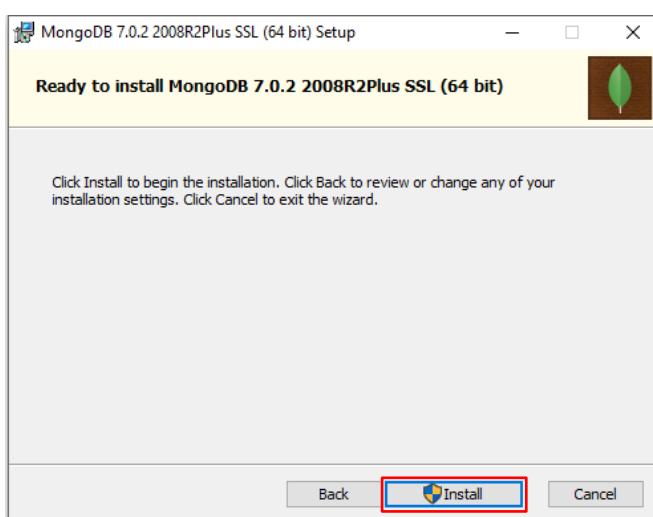
Step 8: Just click on ‘Next’ here.



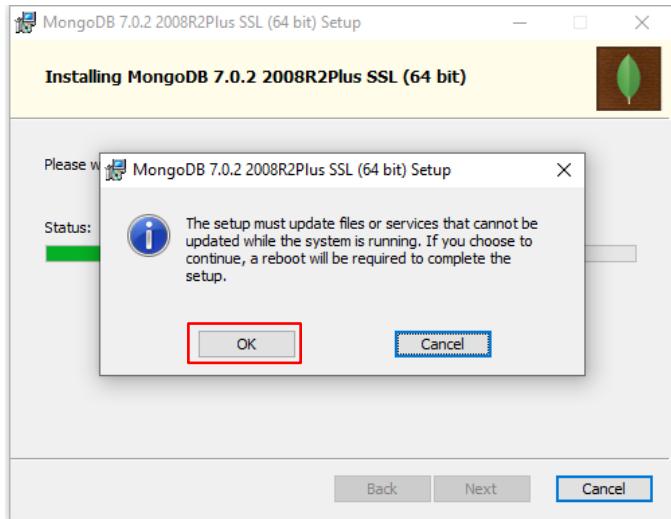
Step 9: Click on 'Next'.



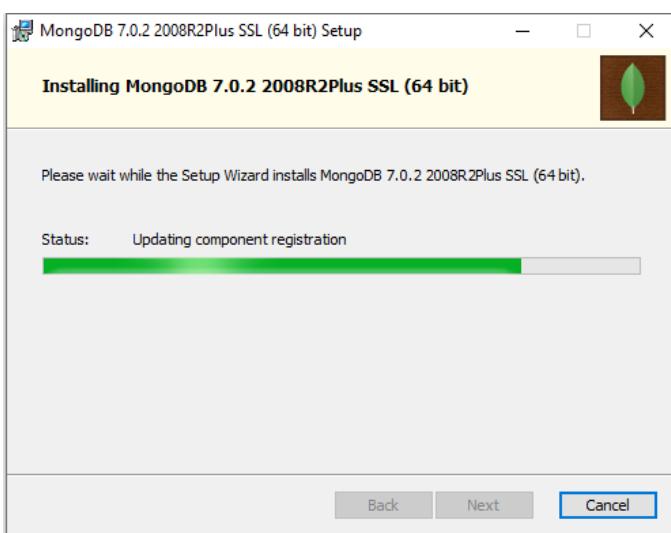
Step 10: Now click on 'Install'.



Step 11: Click on 'OK' here.



Step 12: Wait until the installation is complete.



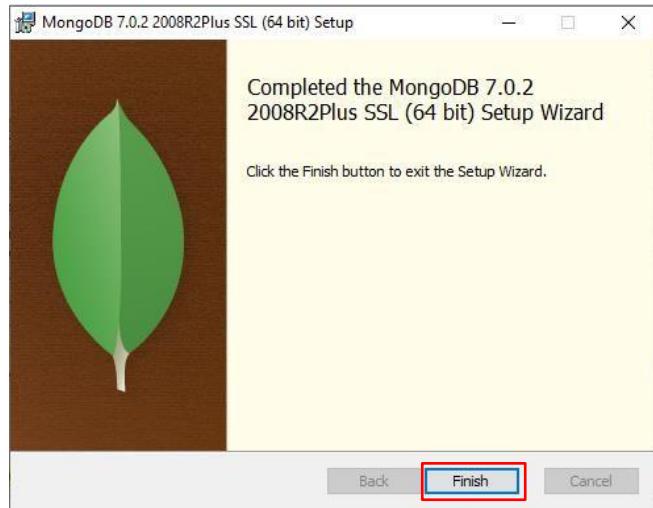
Step 13: MongoDB Compass is ready to launch.



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: 654684cf9eef8170f34808
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/

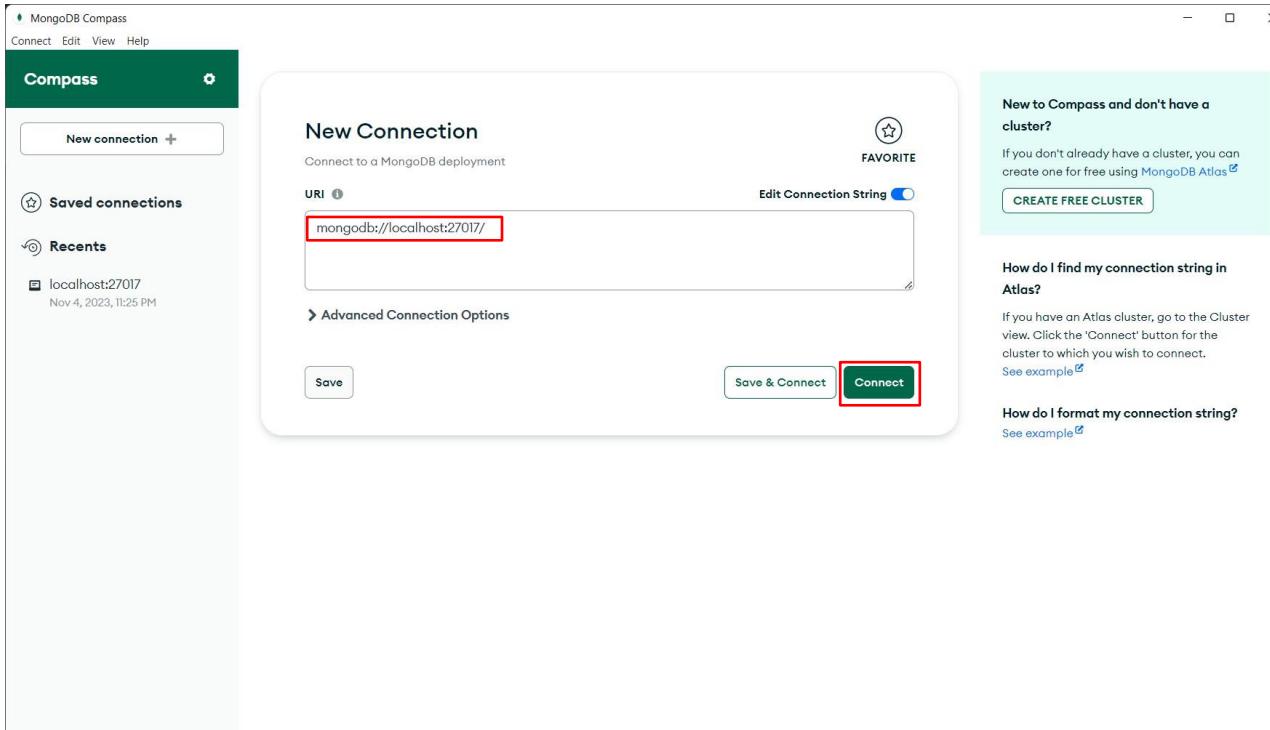
To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
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.

A screenshot of the MongoDB Compass application showing the 'fsd.Student' collection. The left sidebar lists databases: 'localhost:27017' (selected), 'admin', 'config', 'fsd' (selected), and 'local'. Under 'fsd', the 'Student' collection is selected. The main panel shows the 'Documents' tab for the 'fsd.Student' collection. At the top, there are buttons for 'Documents', 'Aggregations', 'Schema', 'Indexes', and 'Validation'. Below that is a search bar and a 'Filter' dropdown. In the center, there's a table header with columns for '_id', 'Name', 'Age', and 'Address'. A single document is listed: '_id': '5e4a2a2a1f0000001', 'Name': 'John Doe', 'Age': 20, 'Address': '123 Main St'. At the bottom, there's a note: 'This collection has no data' and an 'Import Data' button. A red box highlights the 'ADD DATA' button at the bottom left of the main panel.