**OrangeHRM**

OrangeHRM offers a comprehensive HR management system to maintain HR processes of an organization in a user-friendly interface.

**SDLC, Agile and Software Testing**

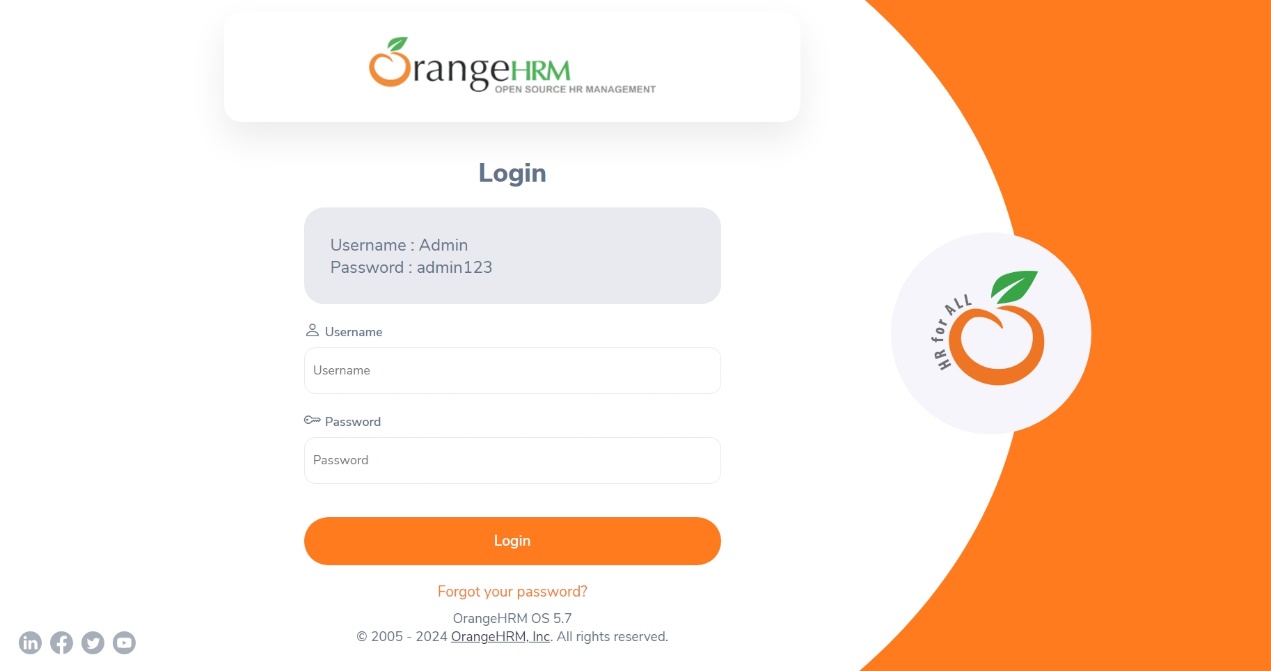


Figure 1. This is the login page for OrangeHRM portal.

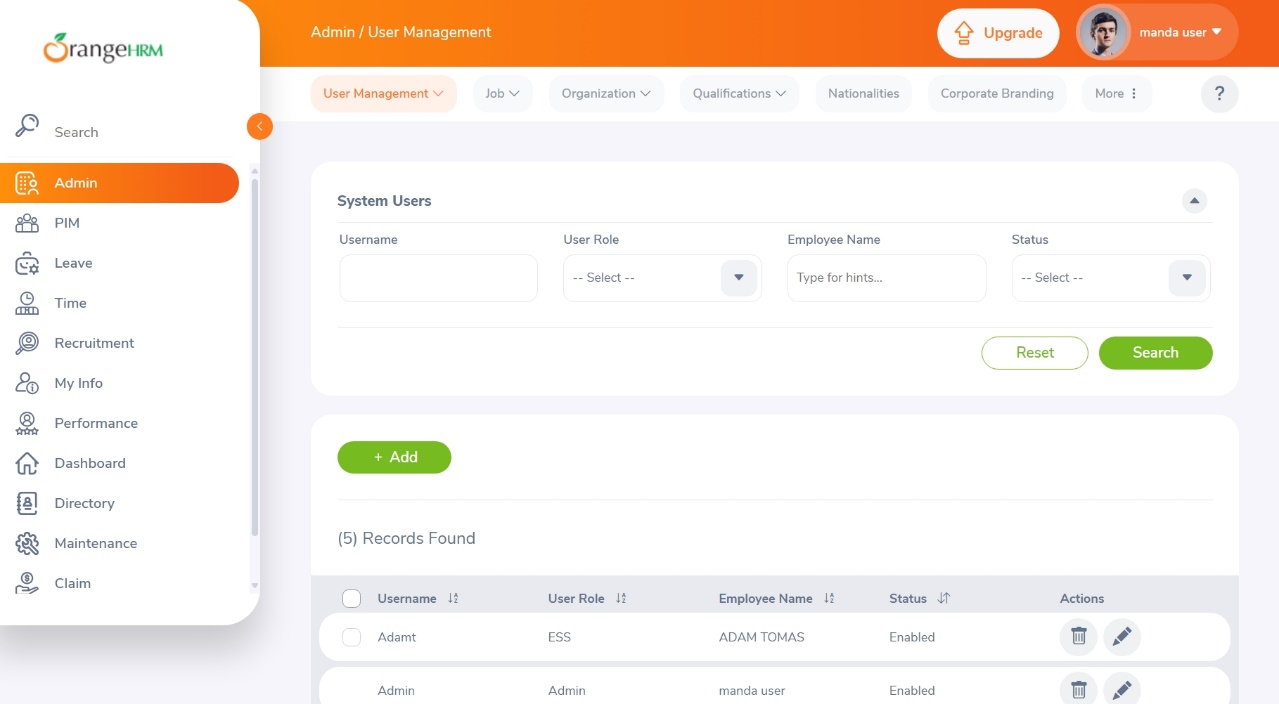


Figure 2. This the home page of the OrangeHRM Portal.

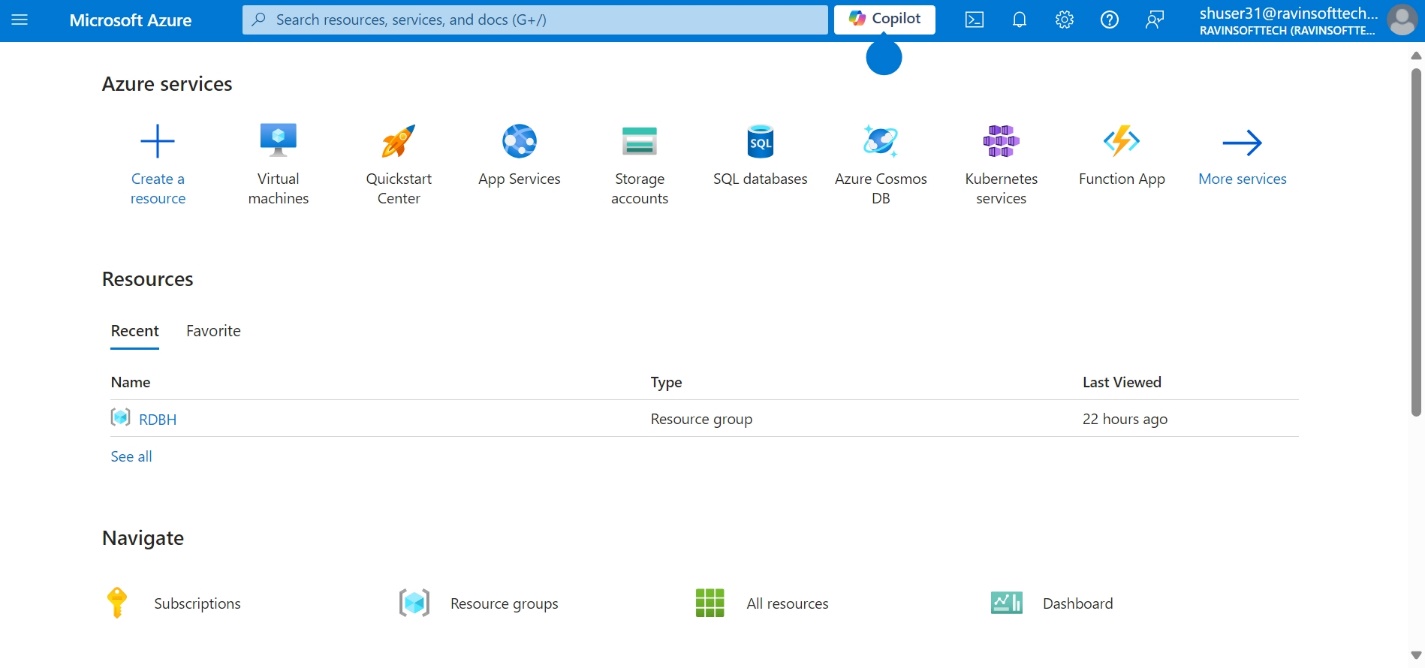


Figure 3. Here we are logged into the Microsoft Azure.

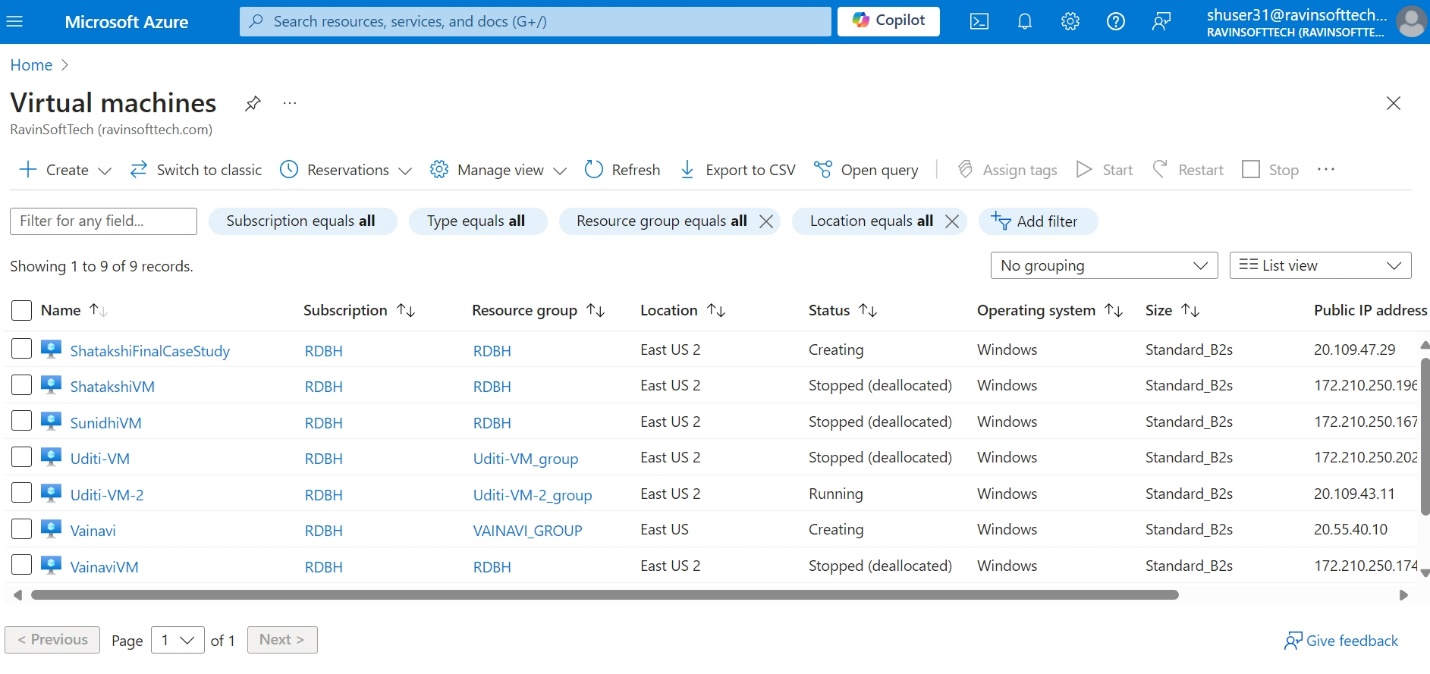


Figure 4. After clicking on the Virtual Machine button, we got the list of all the virtual machines.

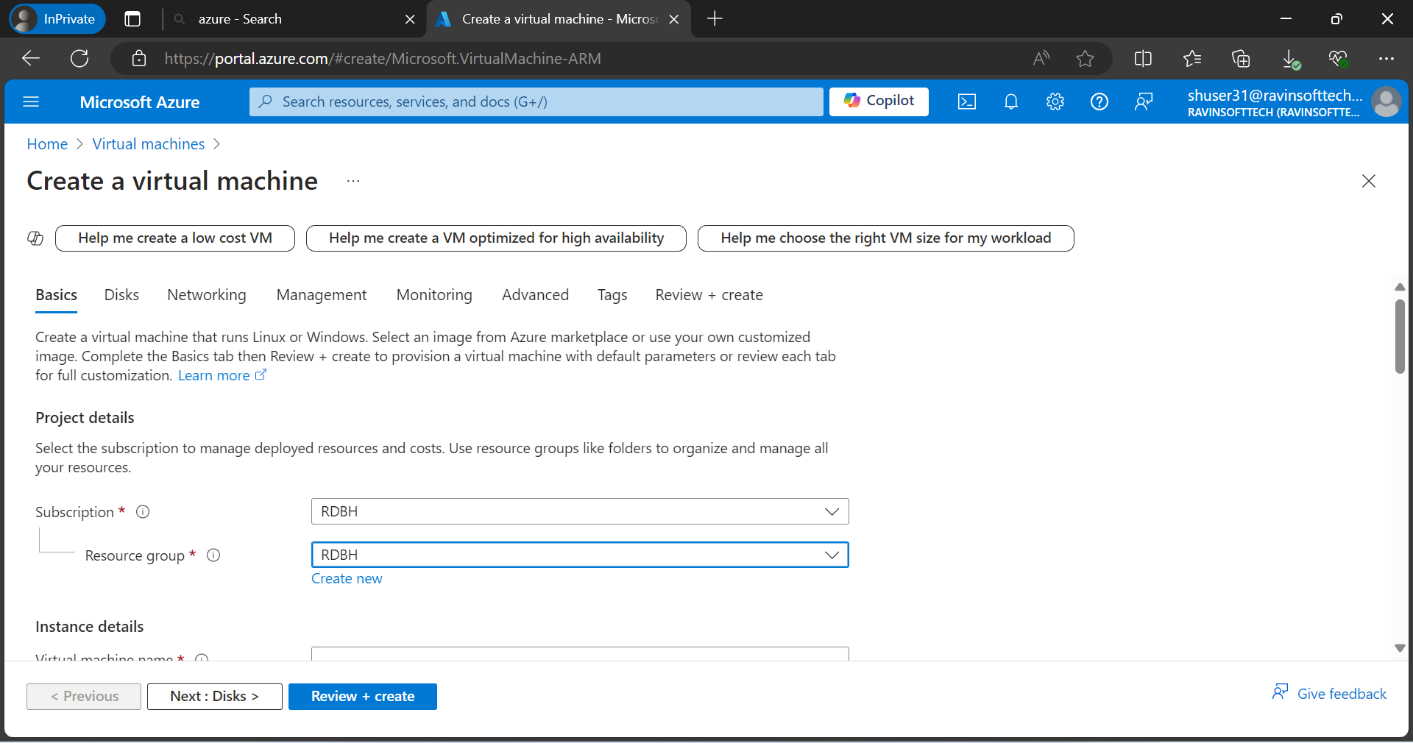


Figure 5. After clicking on the Create new VM.

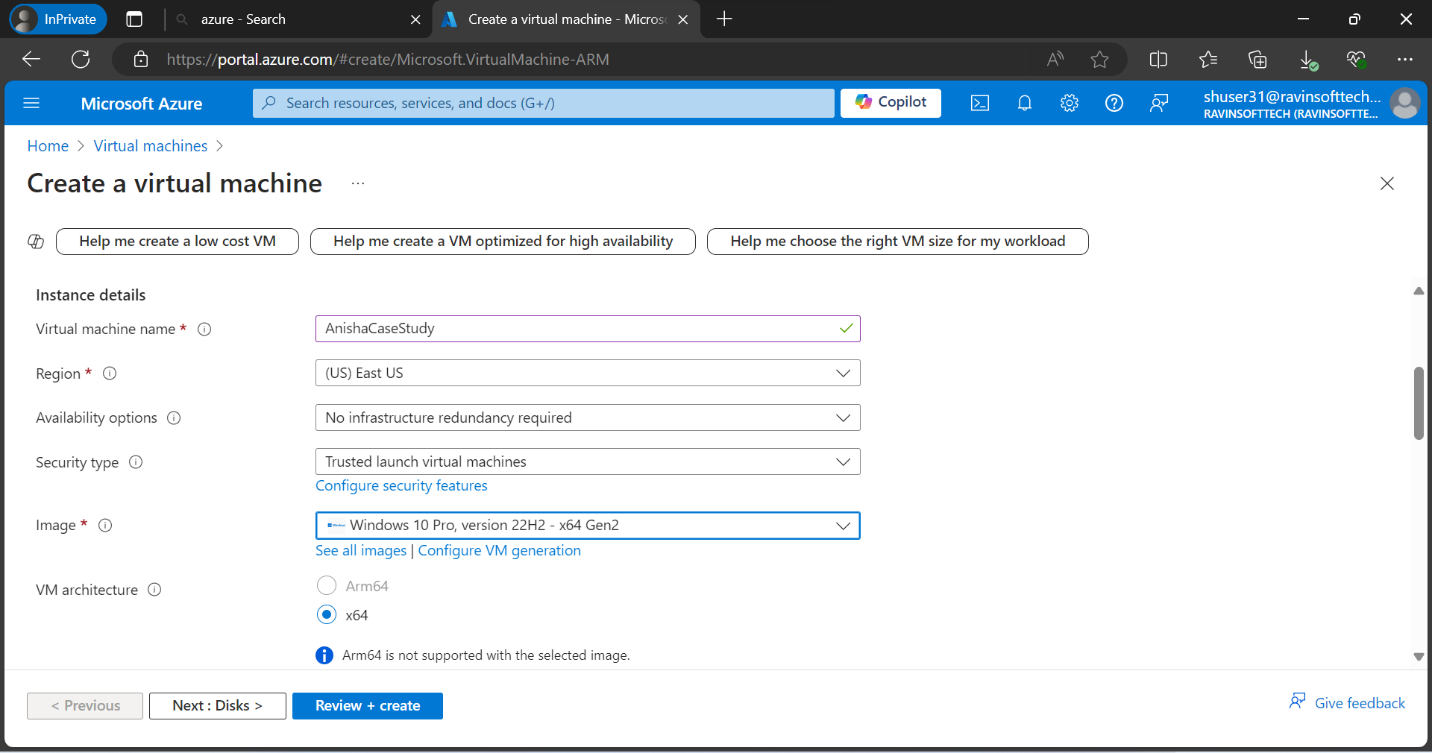


Figure 6. Name and Image of the Virtual Machine has been allocated.

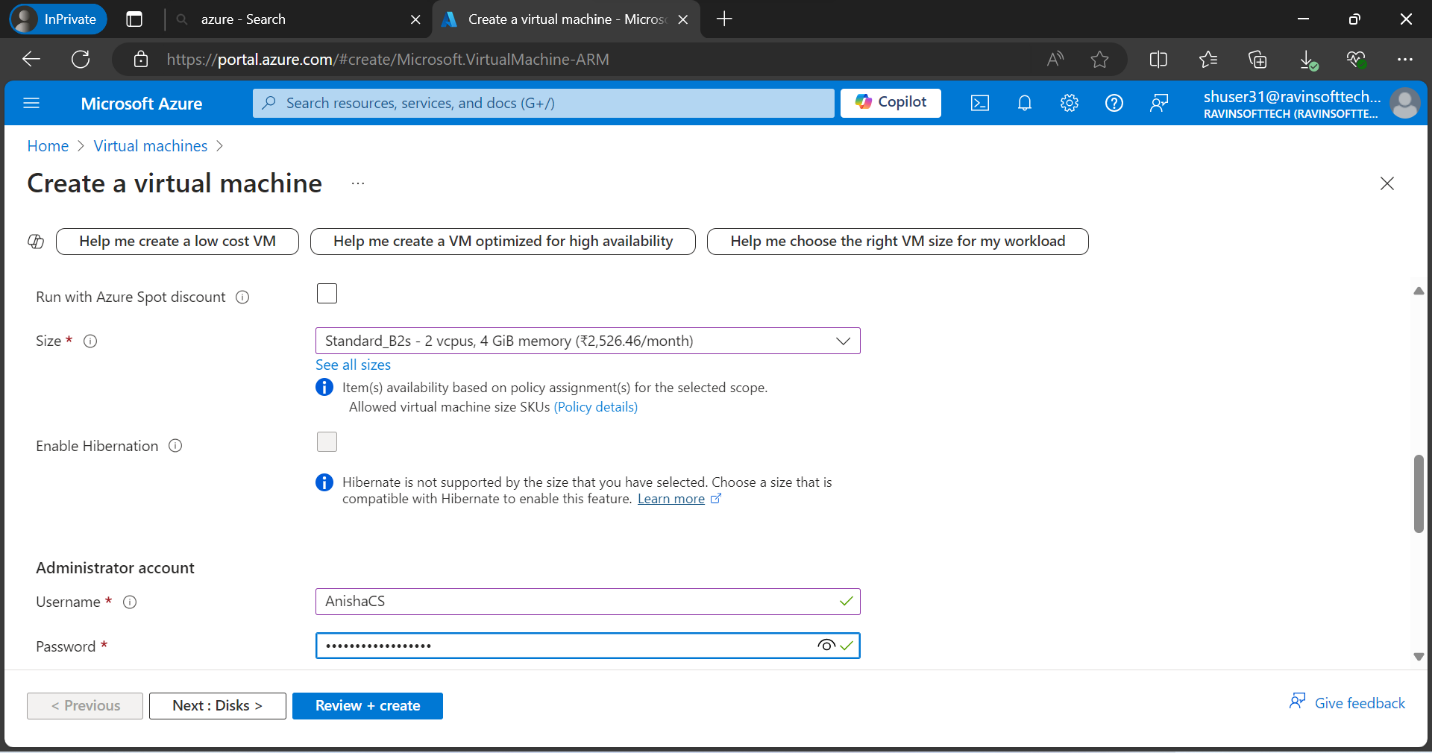


Figure 7. Size, Username and Password had been filled.

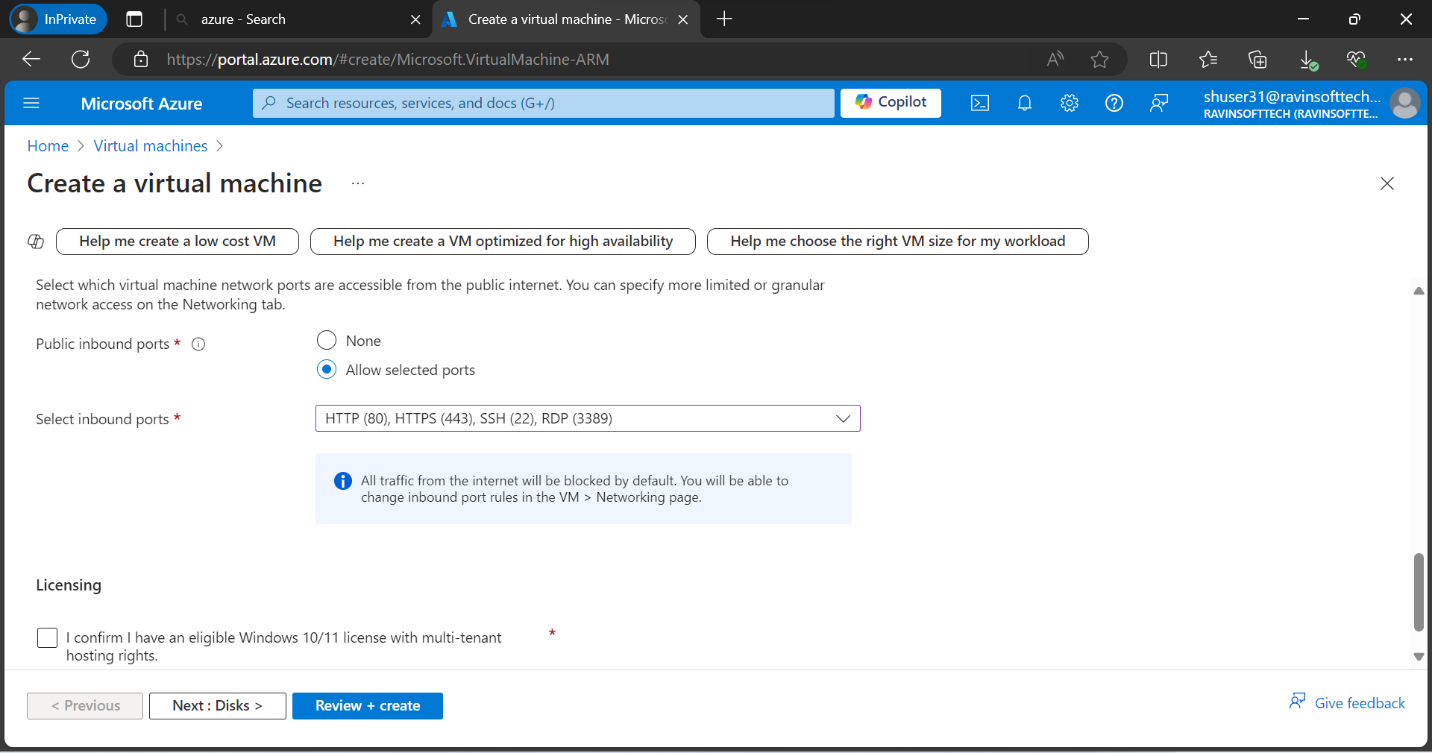


Figure 8. Public inbound ports have been selected and then virtual machine is created.

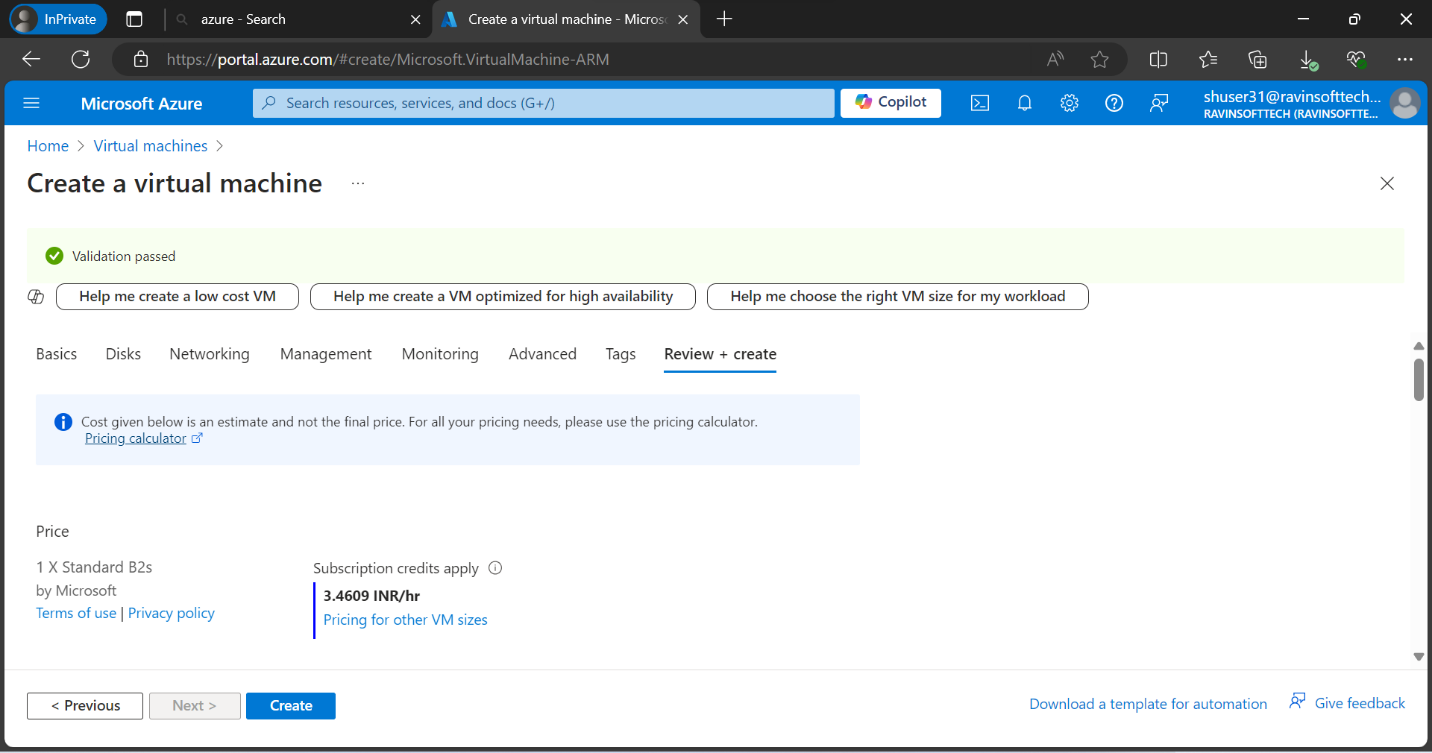


Figure 9. Virtual Machine has been created.

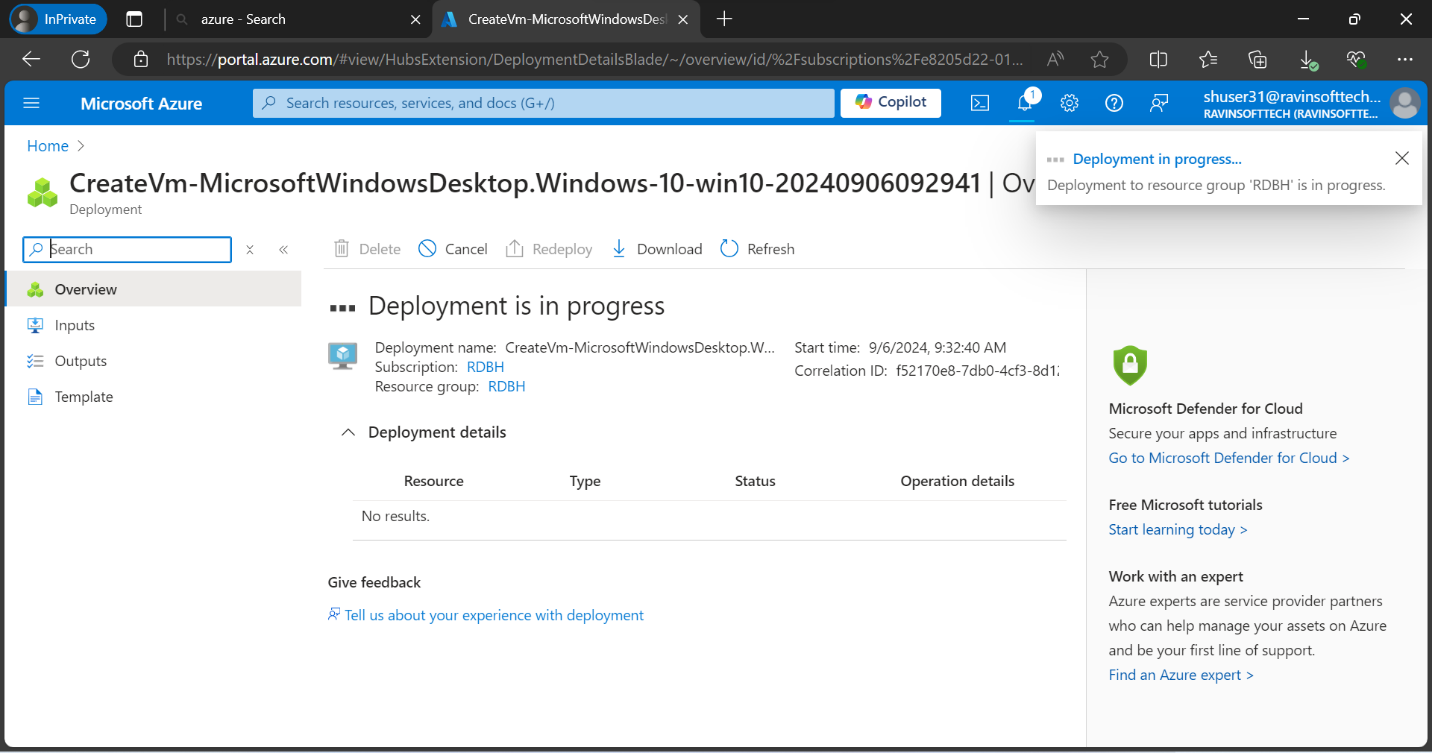


Figure 10. VM is being deployed.

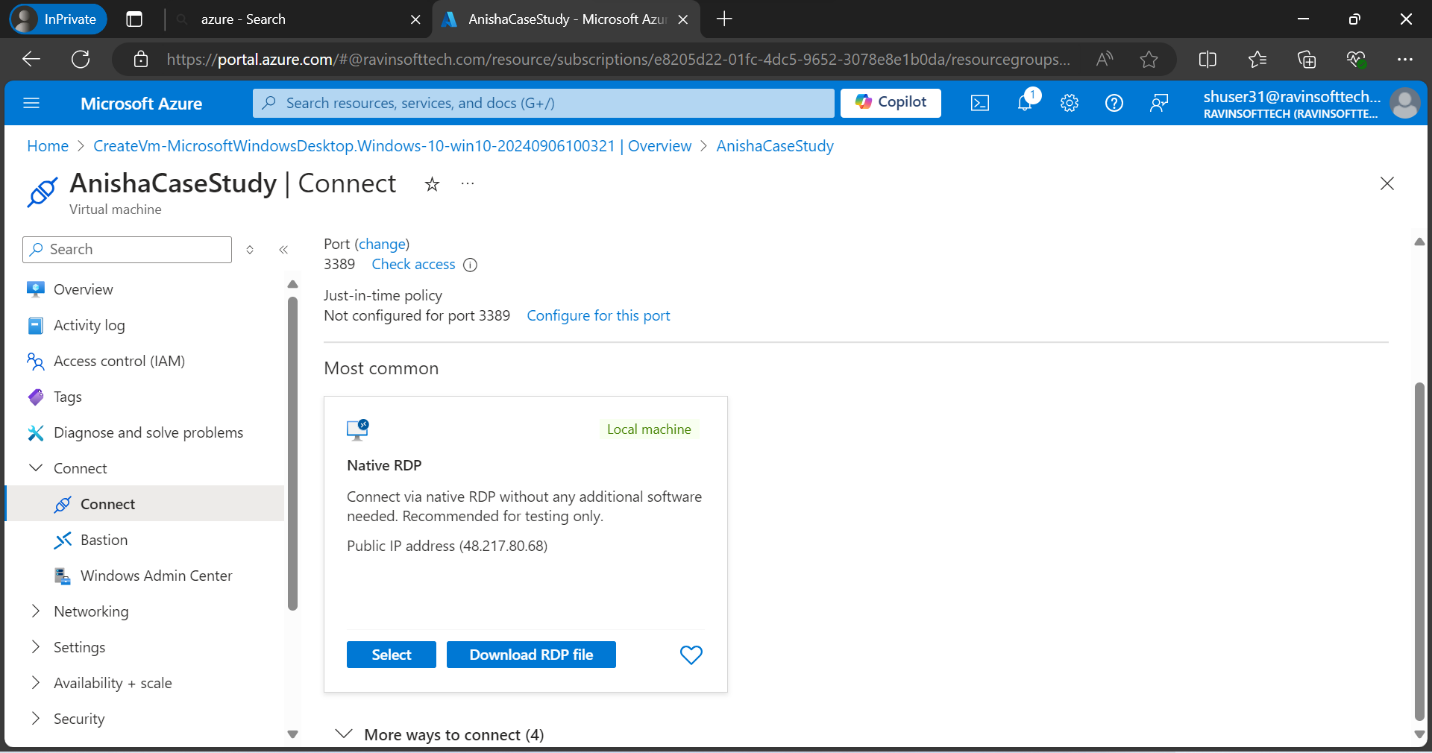
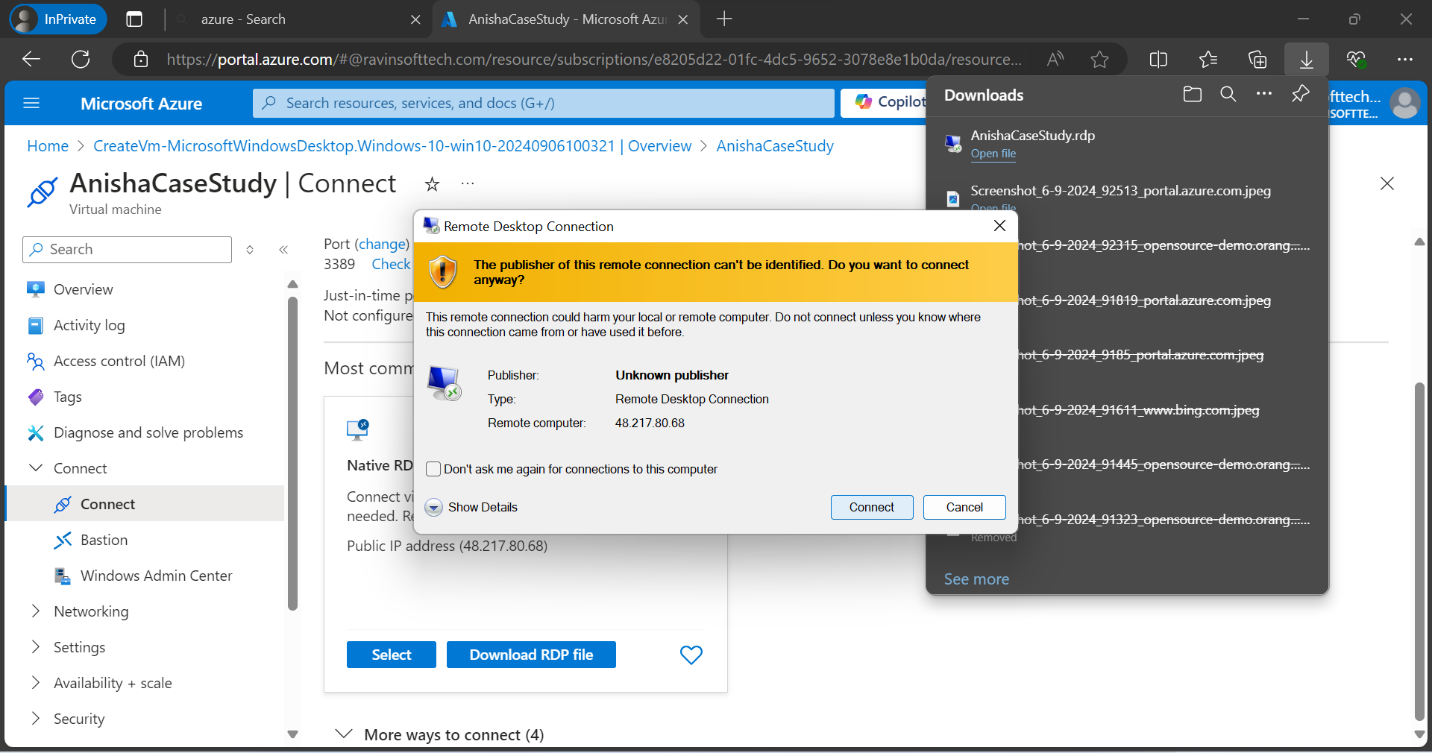


Figure 11. After deploying the VM, it was connected to the resource and then RDP file was downloaded.



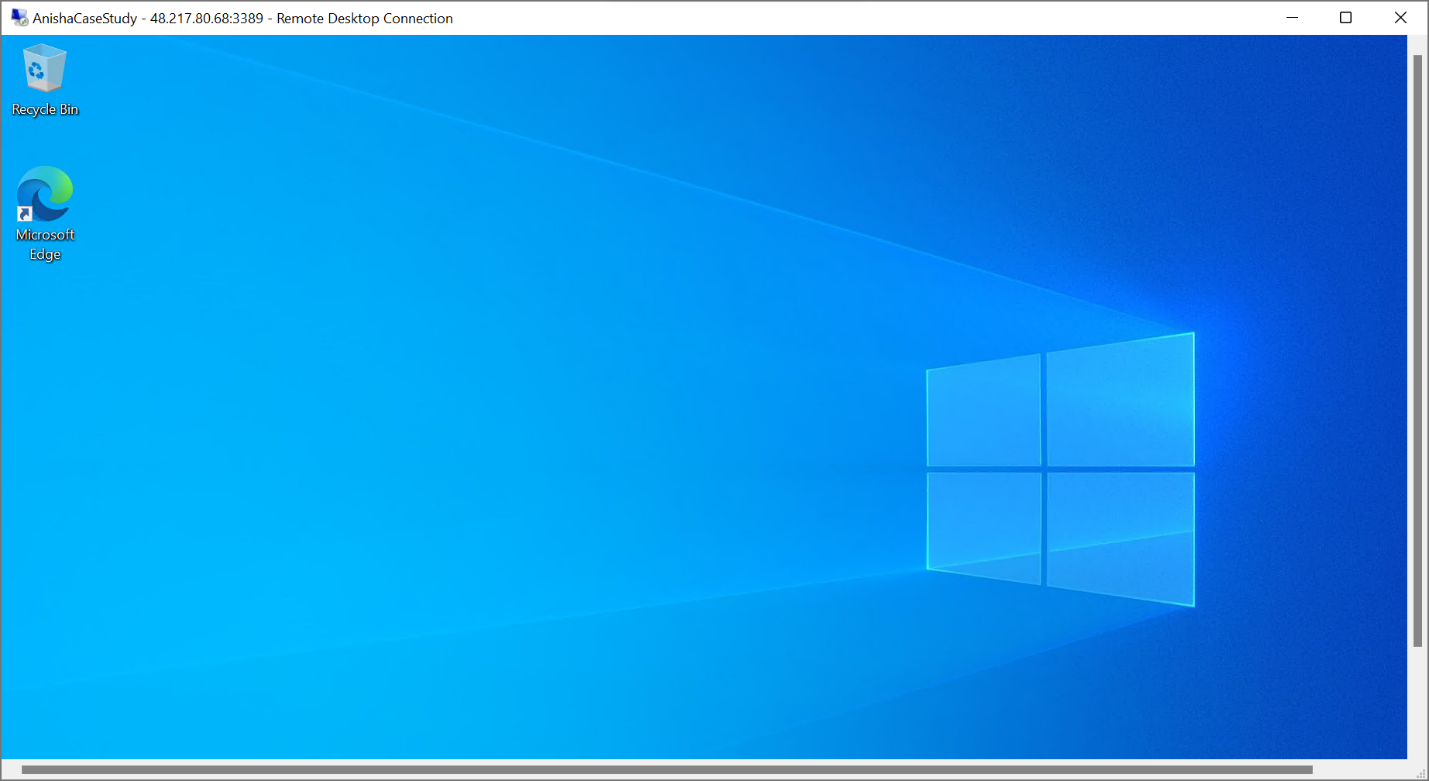


Figure 13. The virtual machine has been opened.

**User Stories**

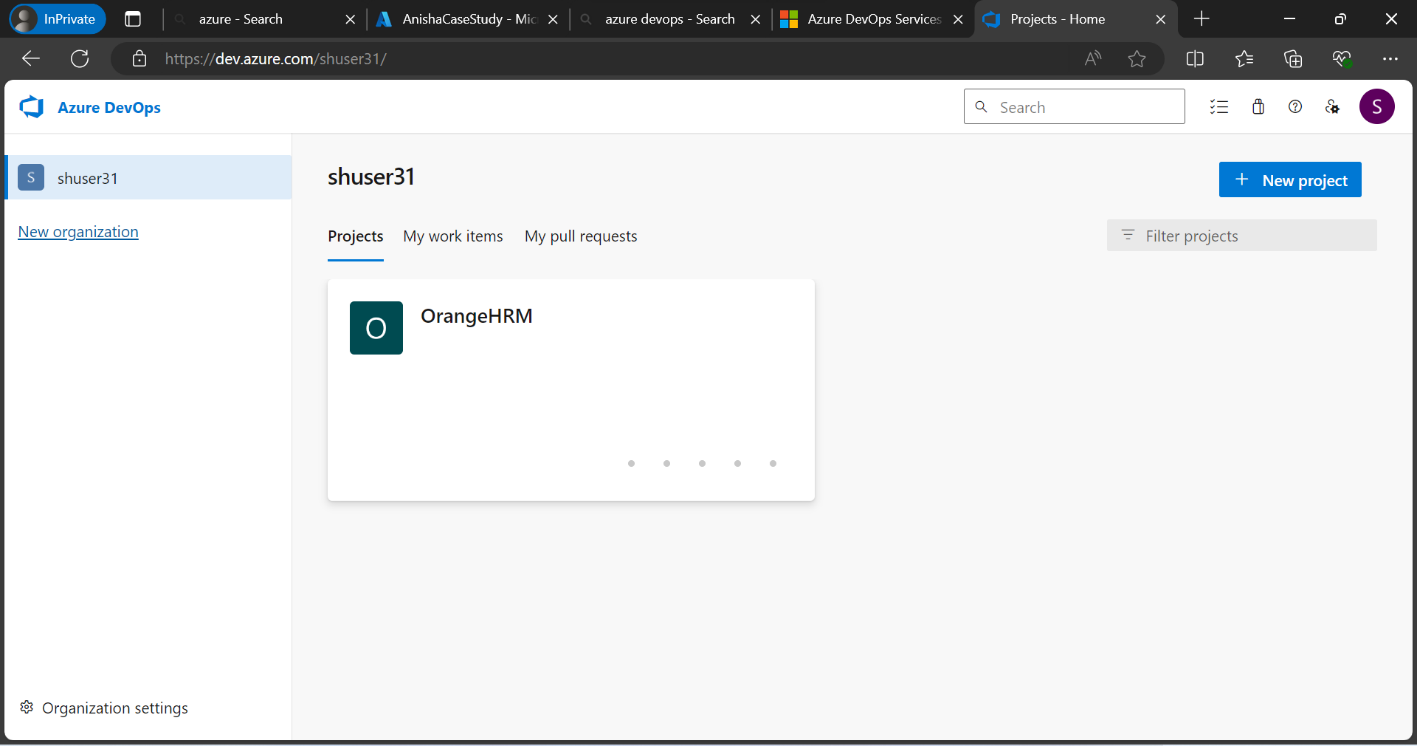


Figure 14. Azure DevOps account has been open and a new project has been created under the name of OrangeHRM.

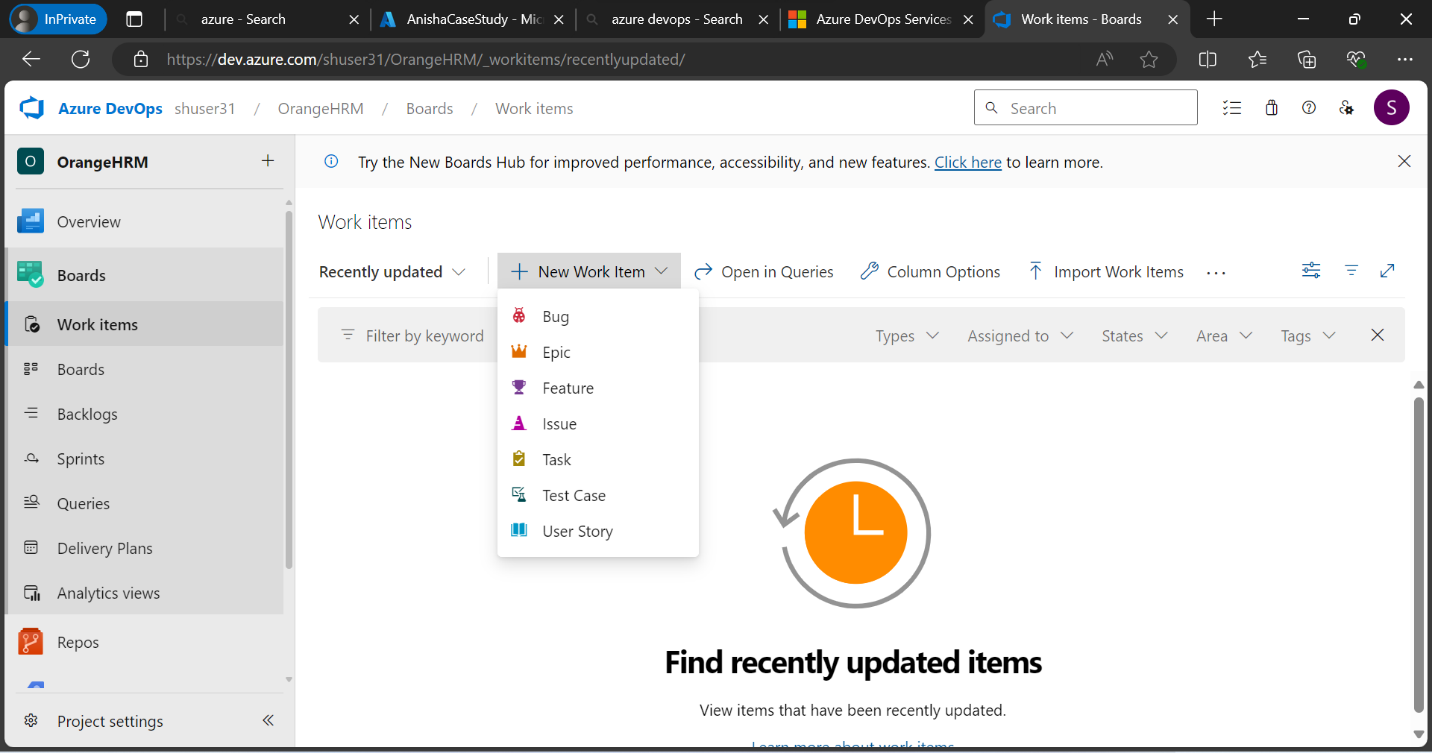


Figure 15. After creating a project New Work Items have to be added after changing the project model to Agile.

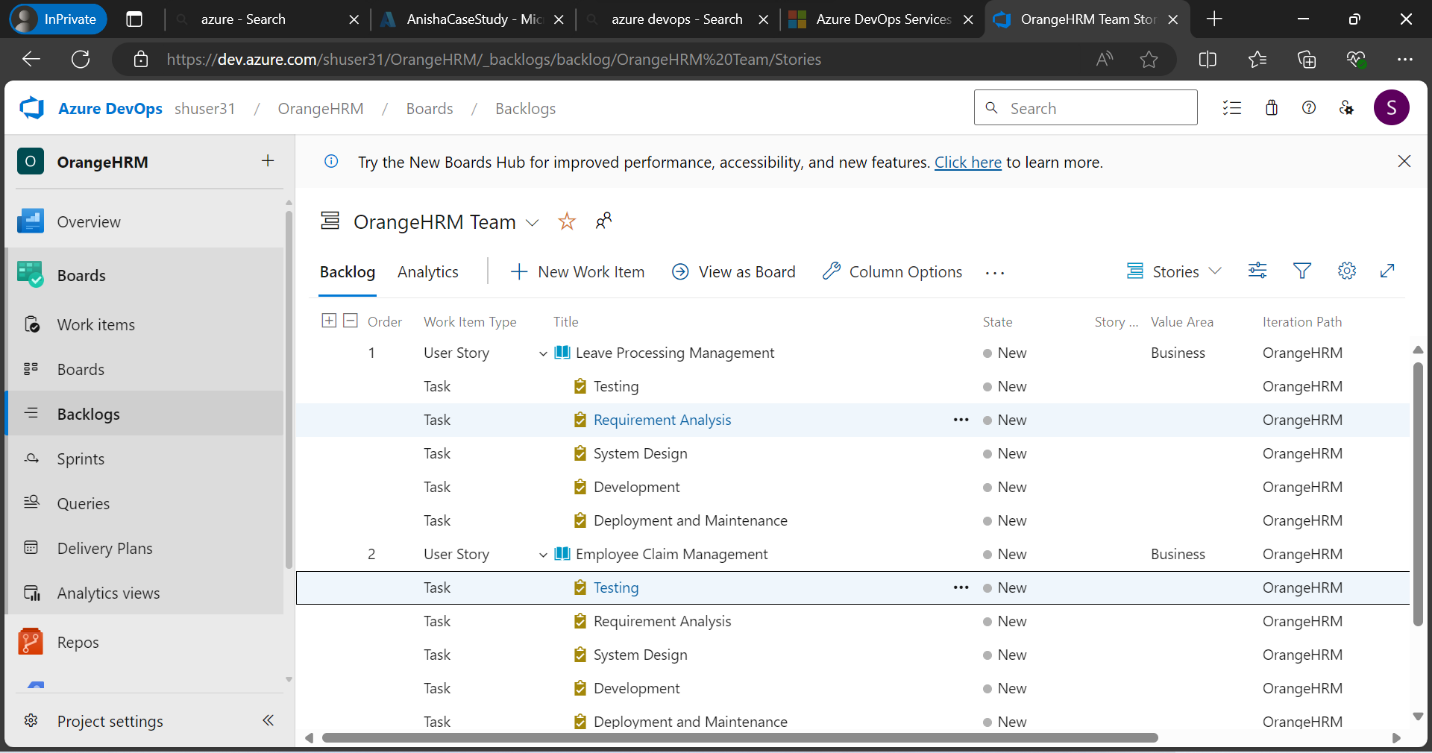


Figure 16. This is the linkage of every user story to their tasks.

1. **Leave Processing Management**  
   As a user I want to manage and grant leaves requests for the employees of my organization, so that the leave balances are accurately maintained.

**Tasks:**

1 Requirement Analysis: Gather and document detailed requirements for the leave processing/management feature, including user roles, leave types, approval workflows, and notification preferences.

2 System Design: Design the system architecture and create detailed design documents, including database schema, user interface mockups, and API specifications for the leave processing/management feature.

3. Development: Implement the leave processing/management feature based on the design documents, including frontend forms, backend workflows, and integration with existing systems.

4. Testing:  Conduct unit testing, integration testing, and user acceptance testing (UAT) to ensure the leave processing/management feature works as expected and meets the requirements.

5. Deployment and Maintenance:  Deploy the leave processing/management feature to the production environment and monitor its performance. Provide ongoing maintenance and support to address any issues or enhancements.

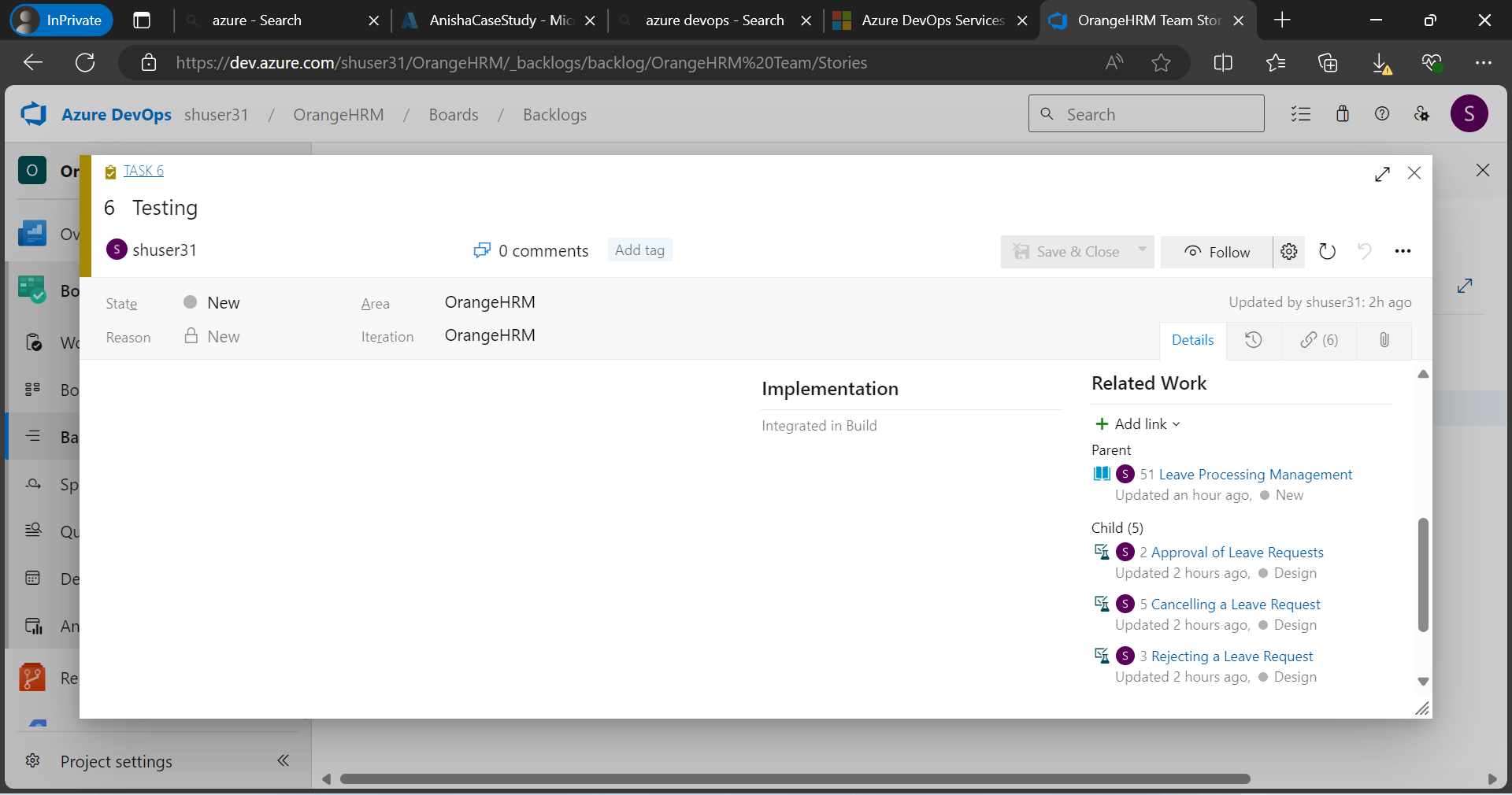


Figure 17. This is the linkage of the Testing Task for the User Case 1 for Leave Process Management.

**Test Cases:**

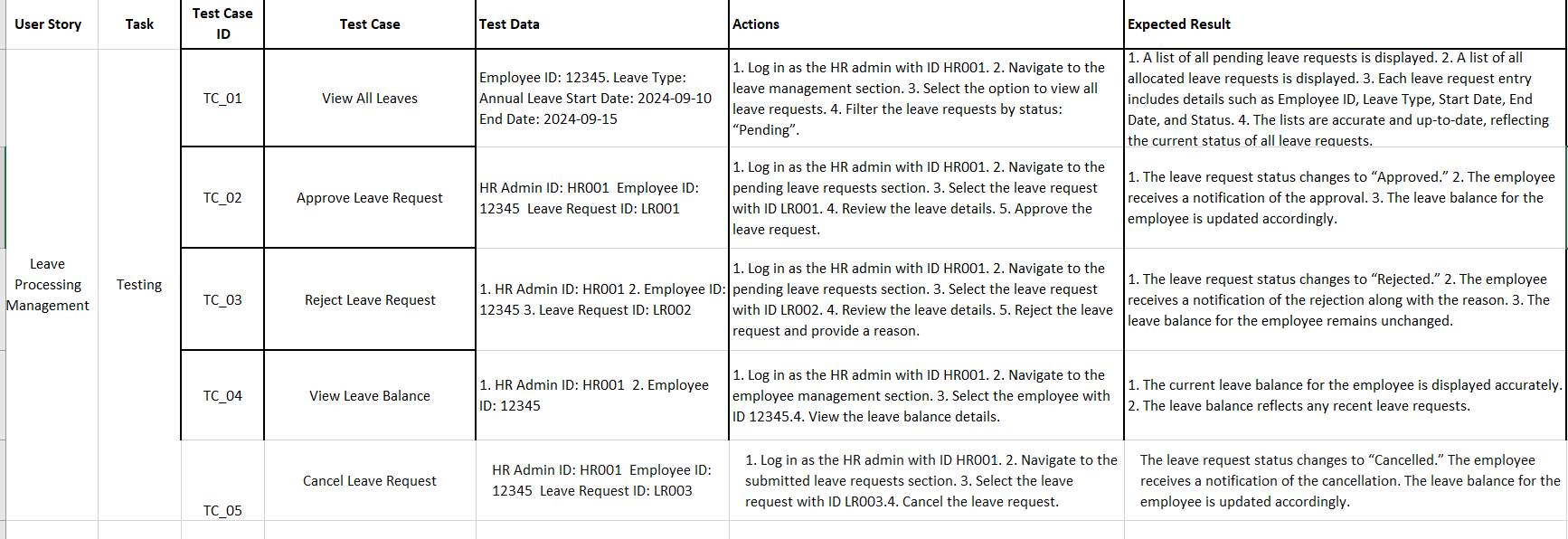


Figure 18. Test cases table.

1. **Employee Claim Management**As a user I want to manage and approve the claim requests for the employees of the organization, so they and the organization both can manage their expenses.

**Tasks:**

1 Requirement Analysis: Gather and document detailed requirements for the leave processing/management feature, including user roles, leave types, approval workflows, and notification preferences.

2 System Design: Design the system architecture and create detailed design documents, including database schema, user interface mockups, and API specifications for the leave processing/management feature.

3. Development: Implement the leave processing/management feature based on the design documents, including frontend forms, backend workflows, and integration with existing systems.

4. Testing:  Conduct unit testing, integration testing, and user acceptance testing (UAT) to ensure the leave processing/management feature works as expected and meets the requirements.

5. **Deployment:** Deploy the claim management module to the production environment and monitor for any issues, providing support as needed.

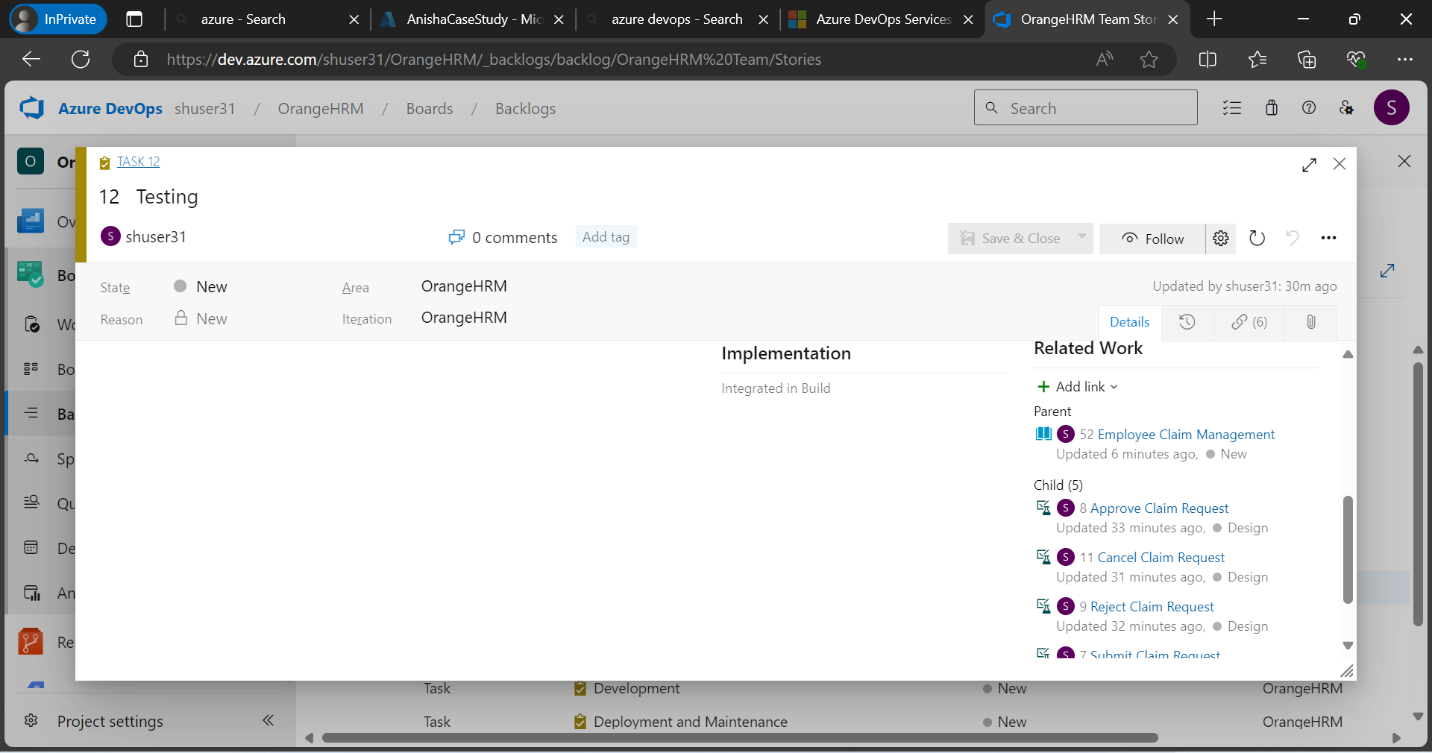


Figure 19. This is the linkage of the Testing Task for the User Case 2 for Employee Claim Management.

**Test Cases:**



Figure 20. Test cases table.

1. **Timesheet Management**As a user I want to manage and review the number of hours per week the employees of my organization have given into the work.

**Tasks:**

1 Requirement Analysis: Gather and document detailed requirements for the leave processing/management feature, including user roles, leave types, approval workflows, and notification preferences.

2 System Design: Design the system architecture and create detailed design documents, including database schema, user interface mockups, and API specifications for the leave processing/management feature.

3. Development: Implement the leave processing/management feature based on the design documents, including frontend forms, backend workflows, and integration with existing systems.

4. Testing:  Conduct unit testing, integration testing, and user acceptance testing (UAT) to ensure the leave processing/management feature works as expected and meets the requirements.

5. **Deployment:** Deploy the claim management module to the production environment and monitor for any issues, providing support as needed.

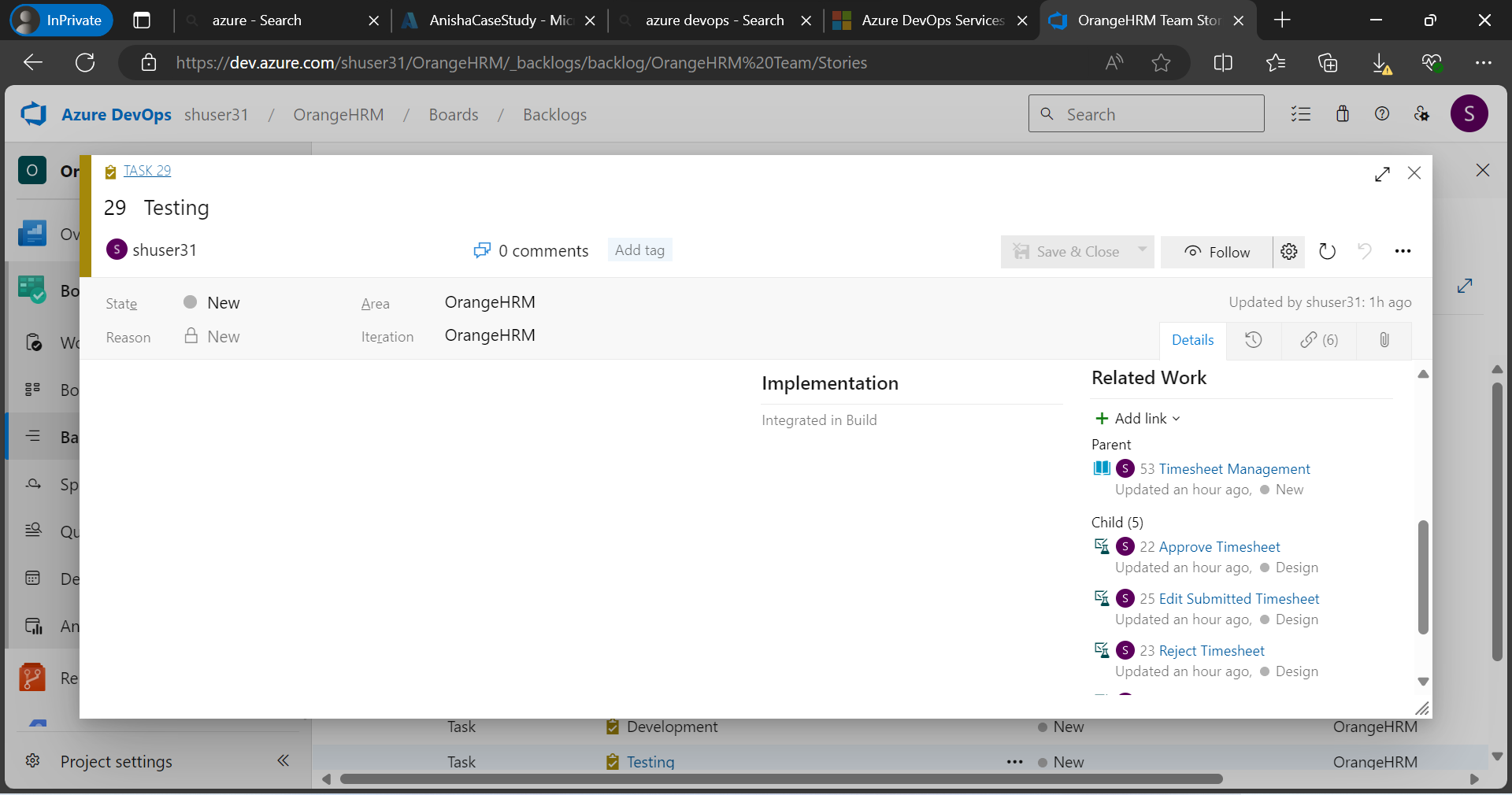


Figure 21. This is the linkage of the Testing Task for the User Case 3 for Timesheet Management.

**Test Case:**

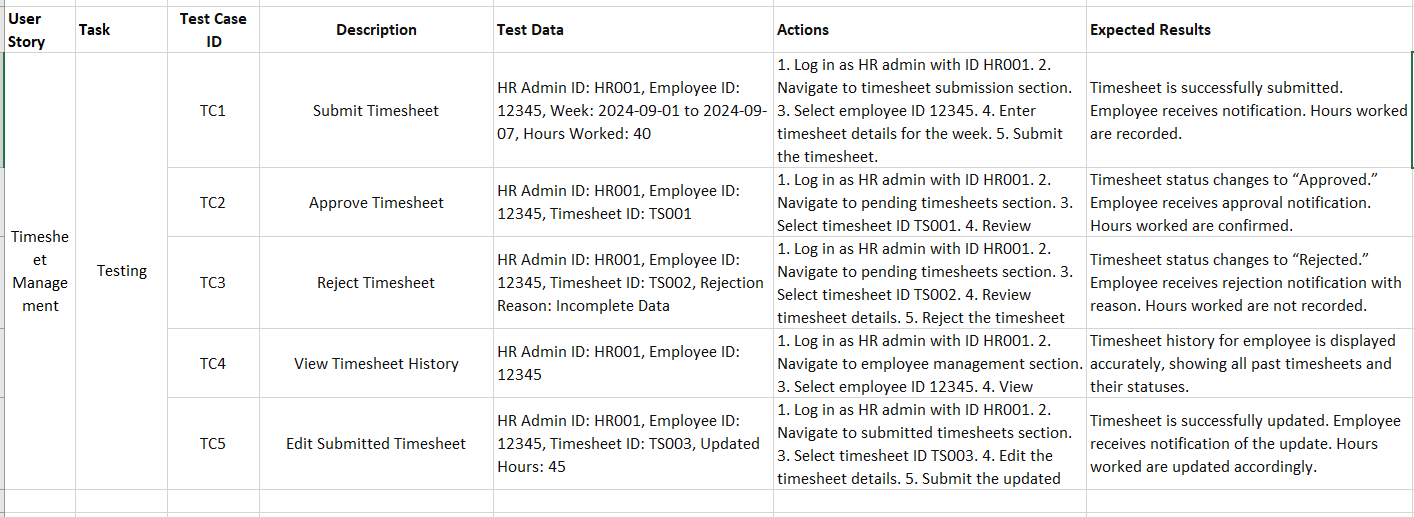
****

Figure 22. Test cases table.

1. **Employee Performance Review System**As a user I want to review the performance of the employees of my organization.

**Tasks:**

1 Requirement Analysis: Gather and document detailed requirements for the leave processing/management feature, including user roles, leave types, approval workflows, and notification preferences.

2 System Design: Design the system architecture and create detailed design documents, including database schema, user interface mockups, and API specifications for the leave processing/management feature.

3. Development: Implement the leave processing/management feature based on the design documents, including frontend forms, backend workflows, and integration with existing systems.

4. Testing:  Conduct unit testing, integration testing, and user acceptance testing (UAT) to ensure the leave processing/management feature works as expected and meets the requirements.

5. **Deployment:** Deploy the claim management module to the production environment and monitor for any issues, providing support as needed.

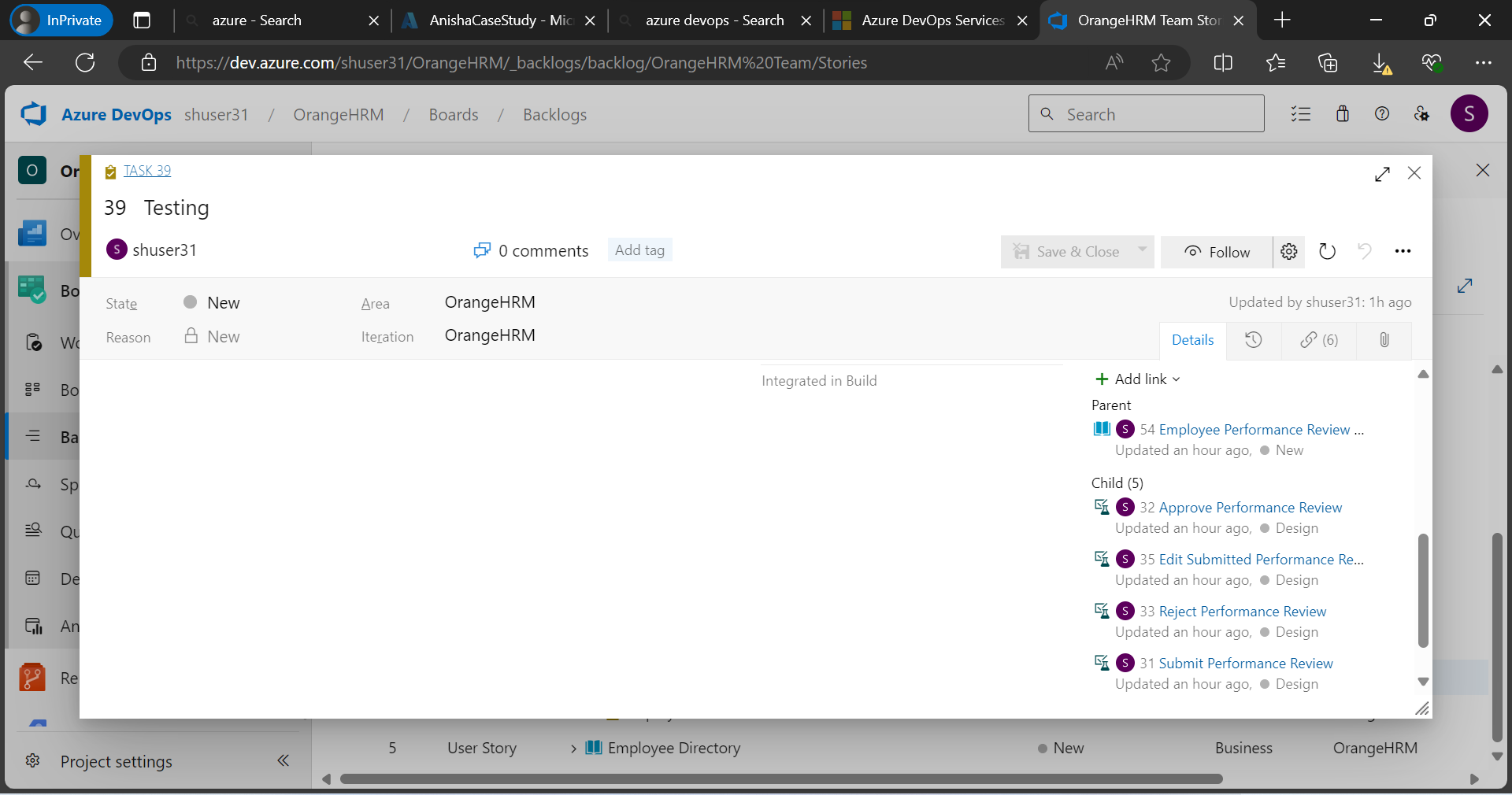
****

Figure 23. This is the linkage of the Testing Task for the User Case 4 for Employee Performance Review.

**Test case**

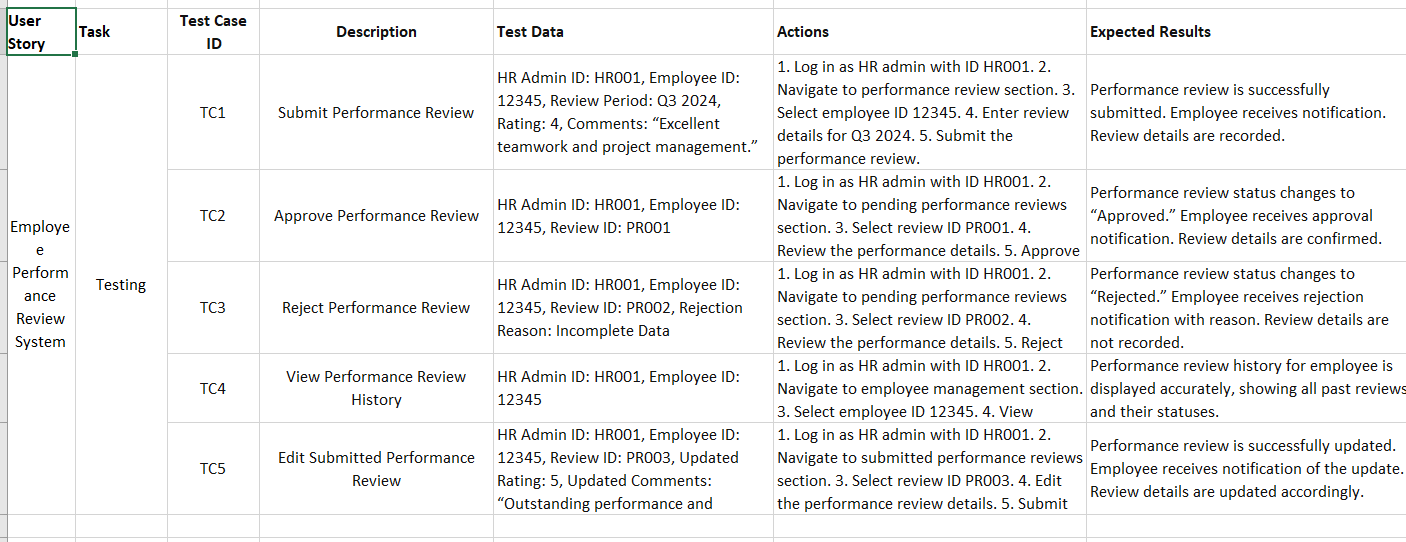
****

Figure 24. Test cases table.

1. **Employee Directory**As a user I want to get the directory of information of all the employees in my organization.

**Tasks:**

1 Requirement Analysis: Gather and document detailed requirements for the leave processing/management feature, including user roles, leave types, approval workflows, and notification preferences.

2 System Design: Design the system architecture and create detailed design documents, including database schema, user interface mockups, and API specifications for the leave processing/management feature.

3. Development: Implement the leave processing/management feature based on the design documents, including frontend forms, backend workflows, and integration with existing systems.

4. Testing:  Conduct unit testing, integration testing, and user acceptance testing (UAT) to ensure the leave processing/management feature works as expected and meets the requirements.

5. **Deployment:** Deploy the claim management module to the production environment and monitor for any issues, providing support as needed.

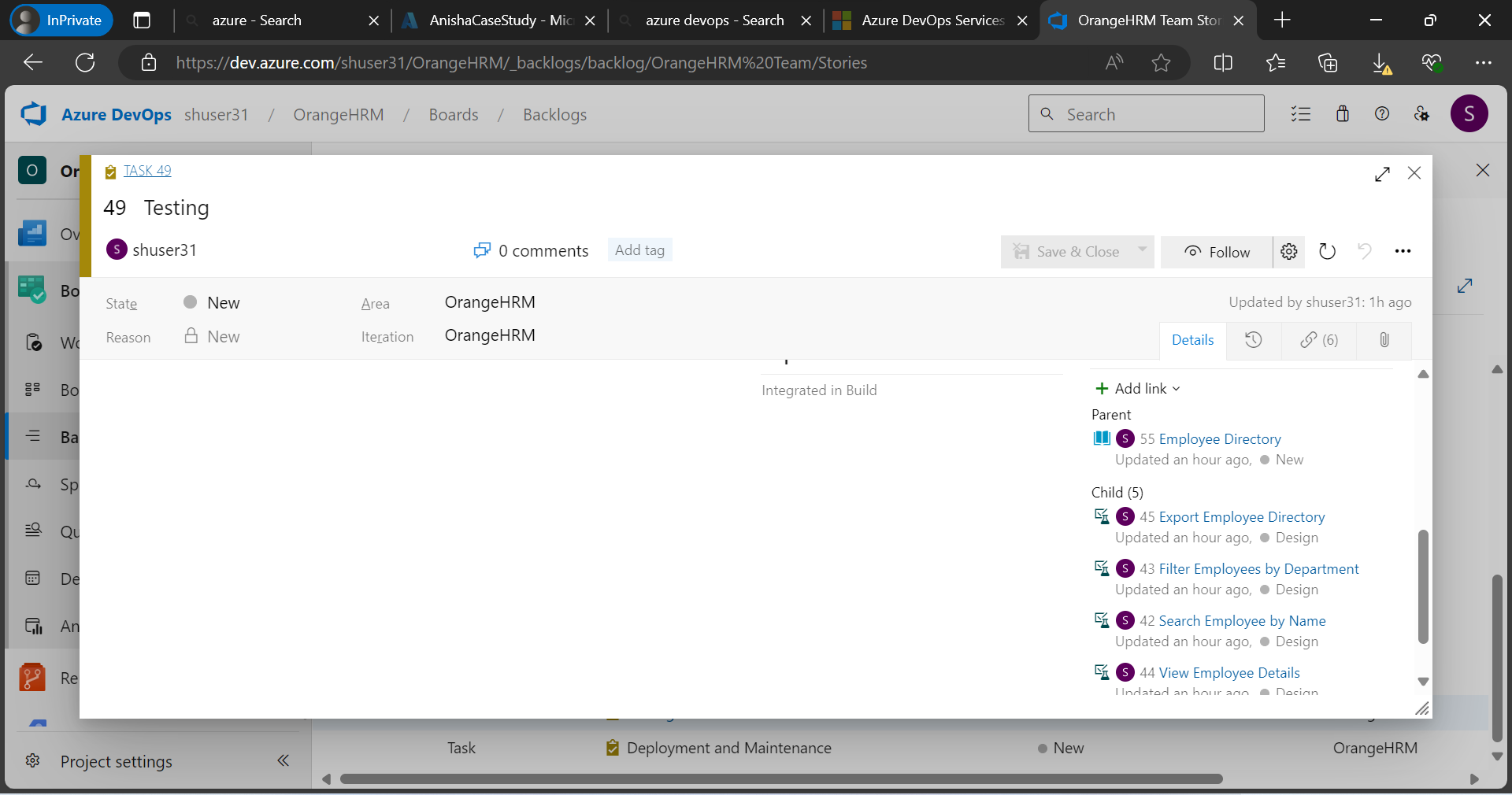


Figure 25. This is the linkage of the Testing Task for the User Case 5 for Employee Directory.

**Test Case**

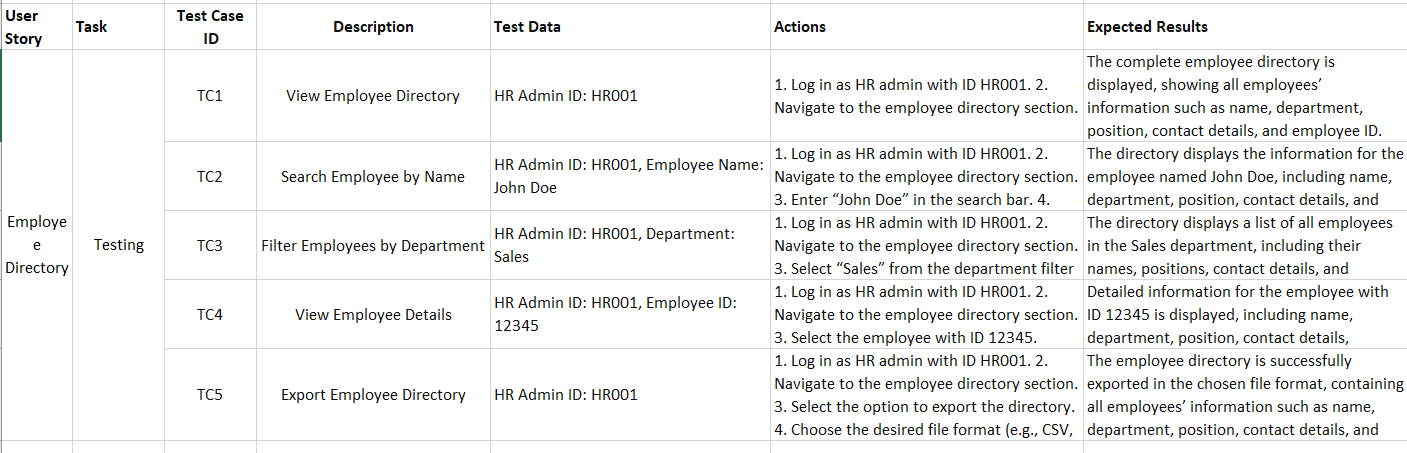


Figure 26. Test cases table.

**Defects**

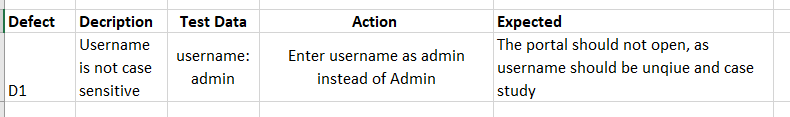
****

Figure 27. Table of Defects detected in the system

**Implementing the CI/CD Pipeline using GitHub Actions**

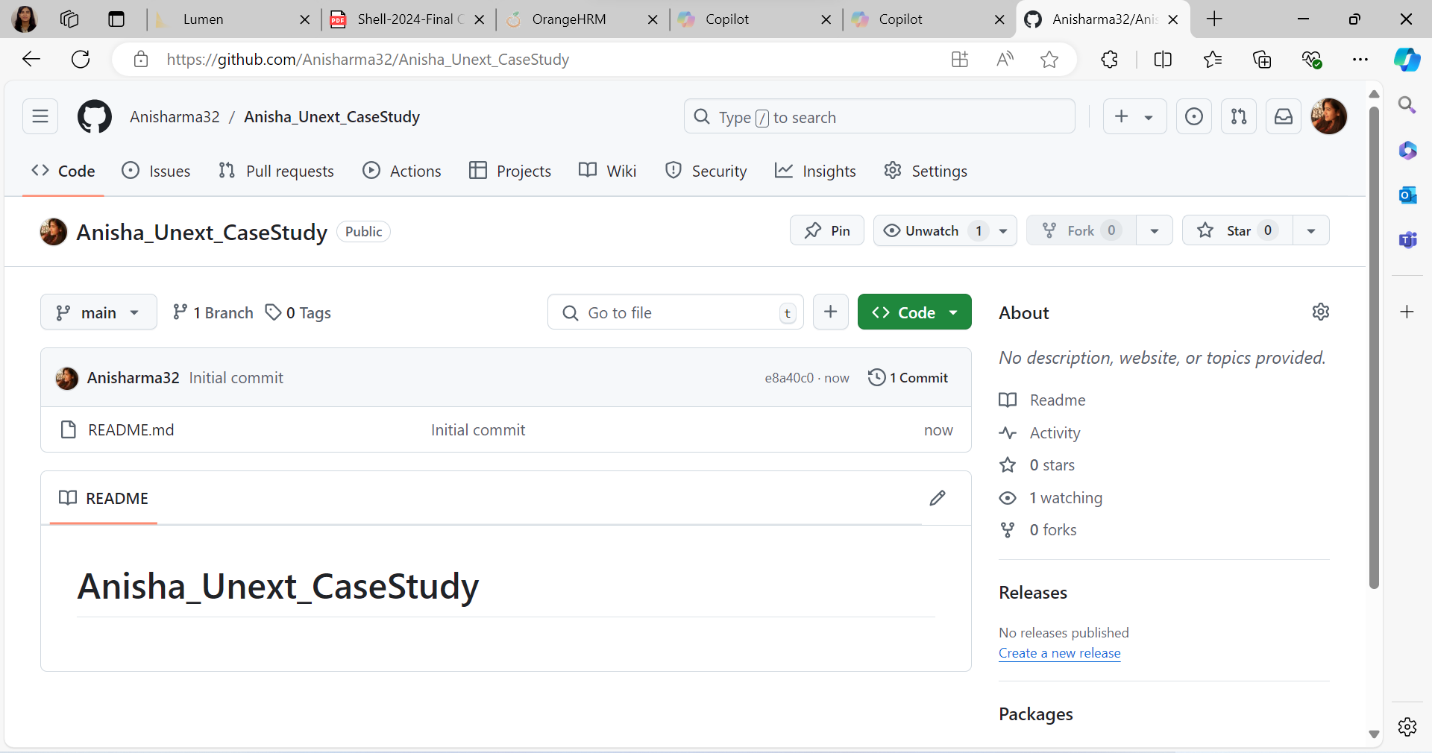


Figure 28. New github repository has been created.

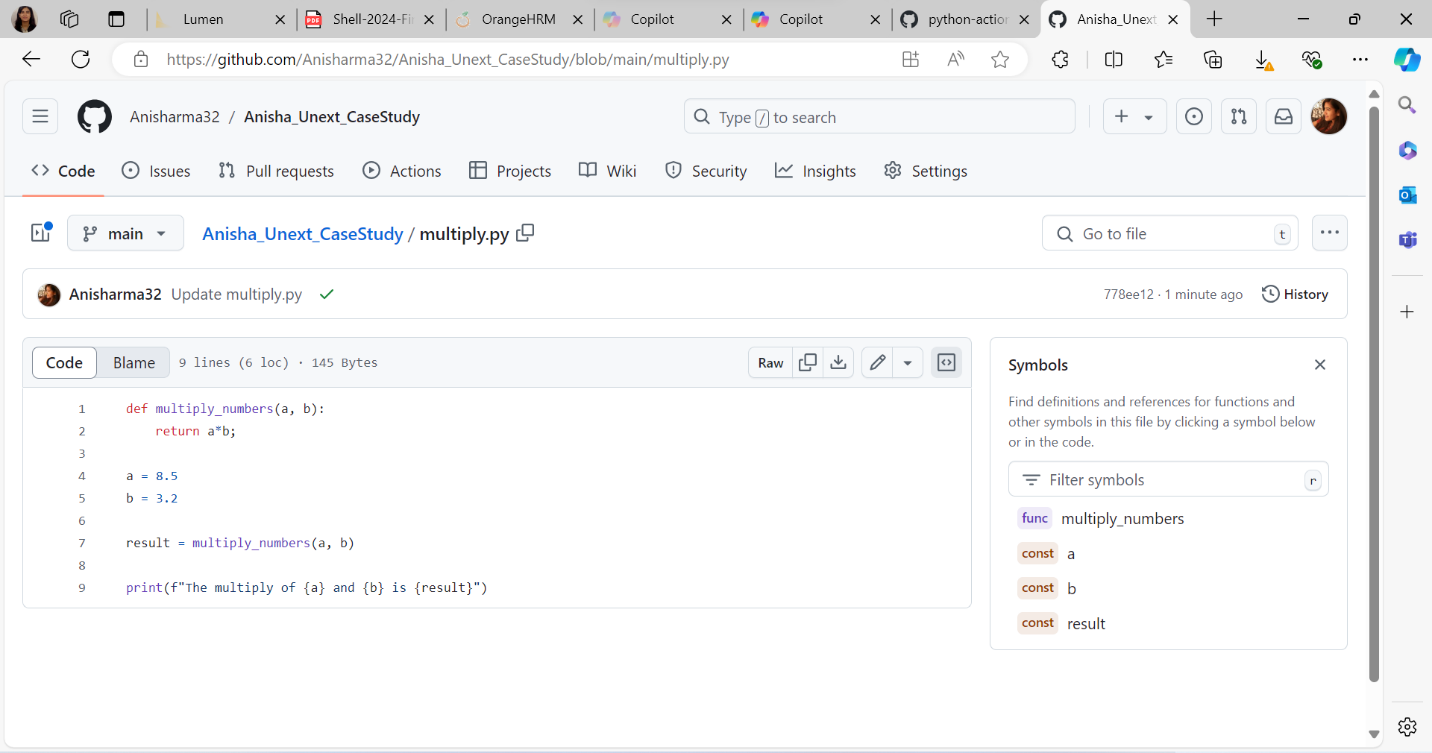


Figure 29. A python file for multiplication has been created.

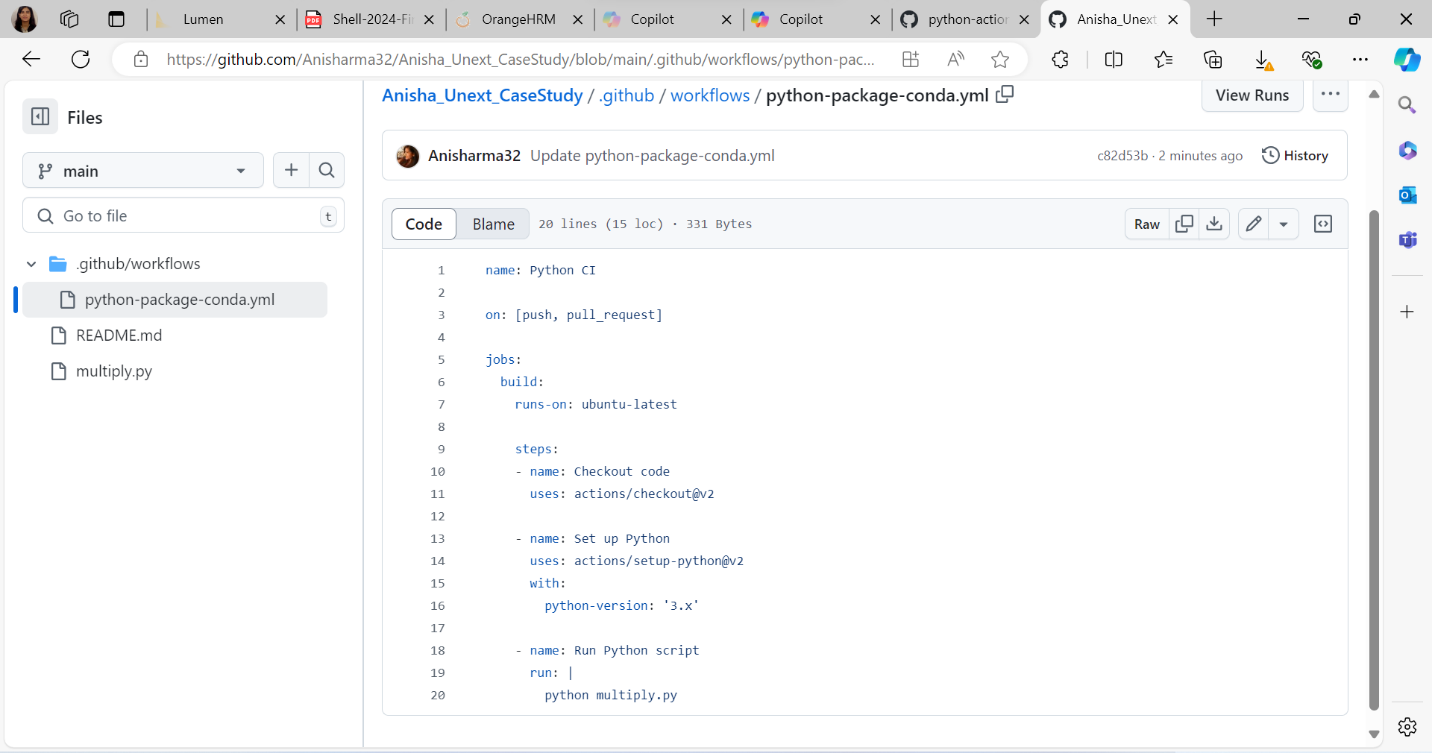


Figure 30. After the code creation, a workflow has been created and for those actions has been created in YAML file.

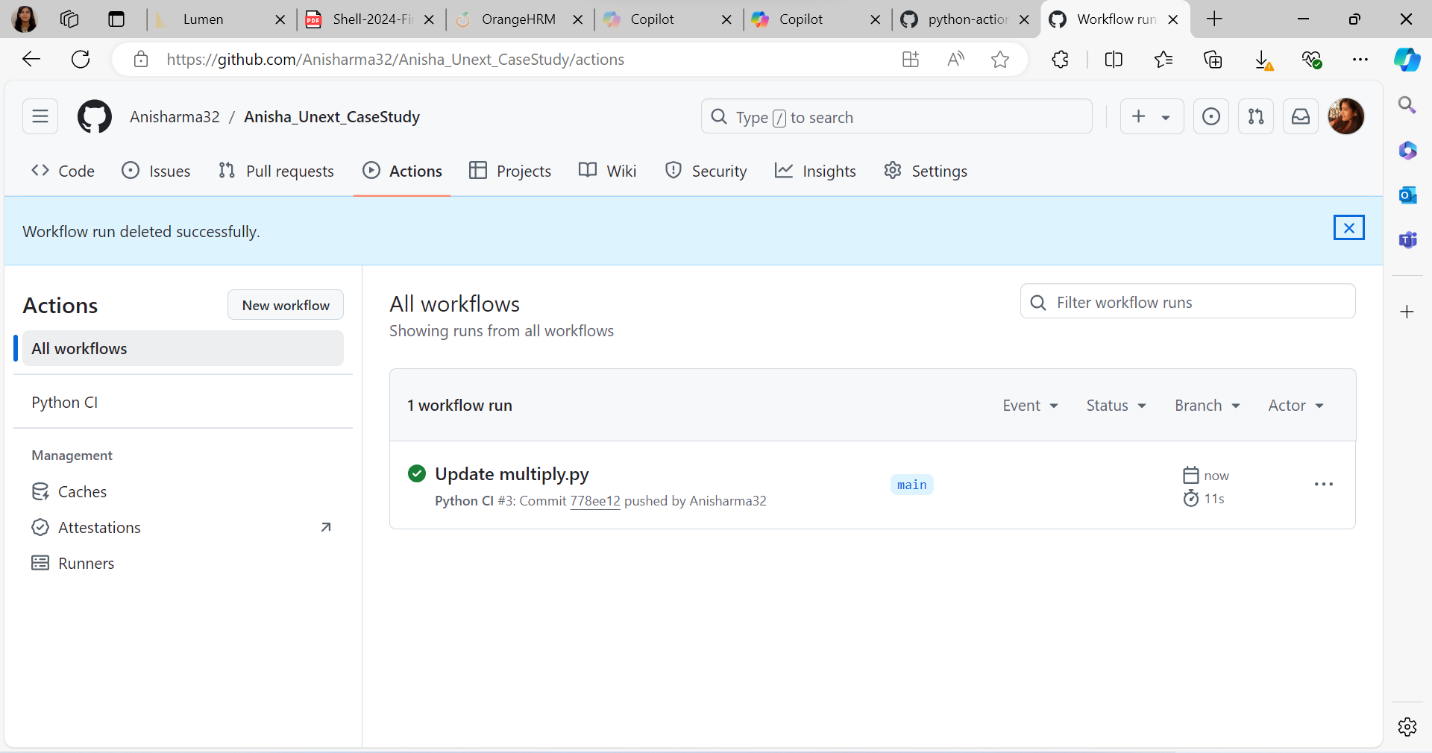


Figure 31. YAML file has been checked and python file has been updated.

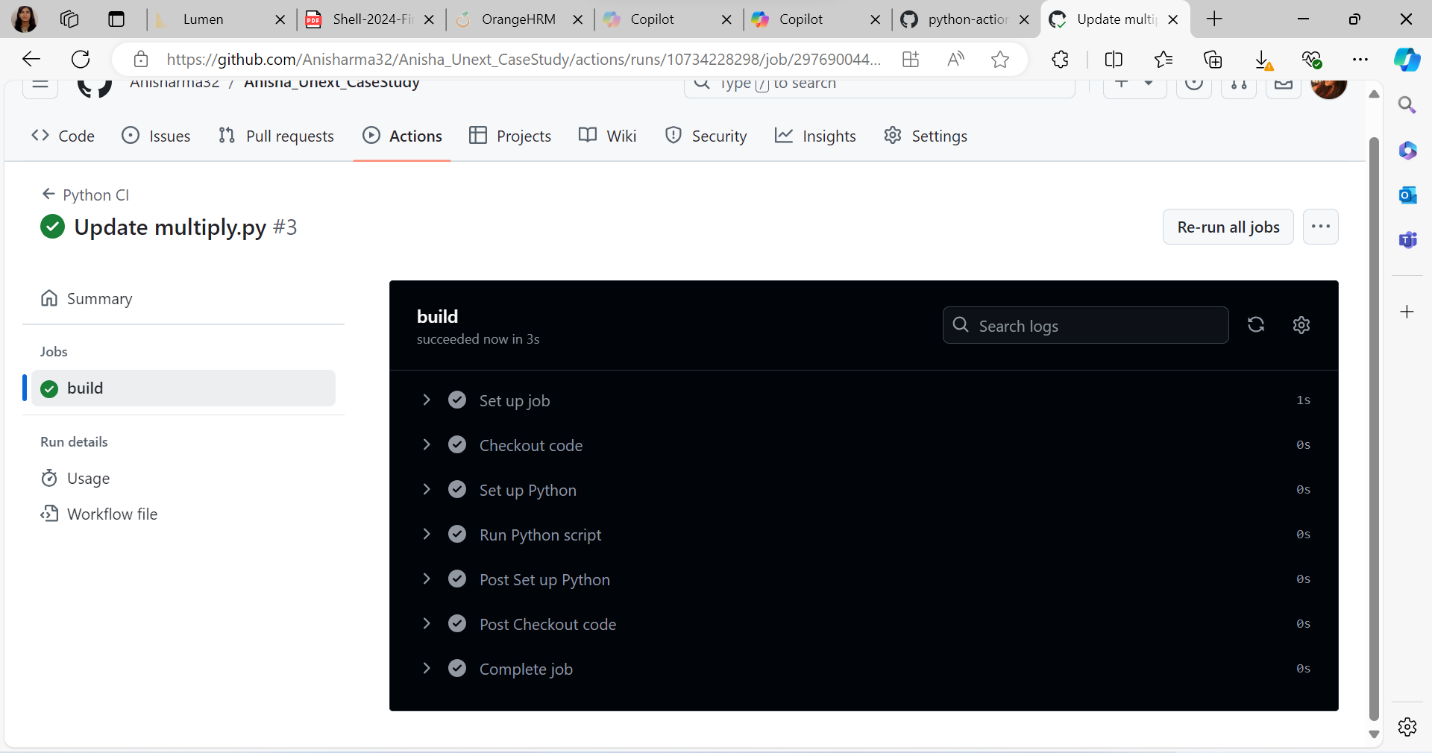


Figure 32. All the automations has been passed.

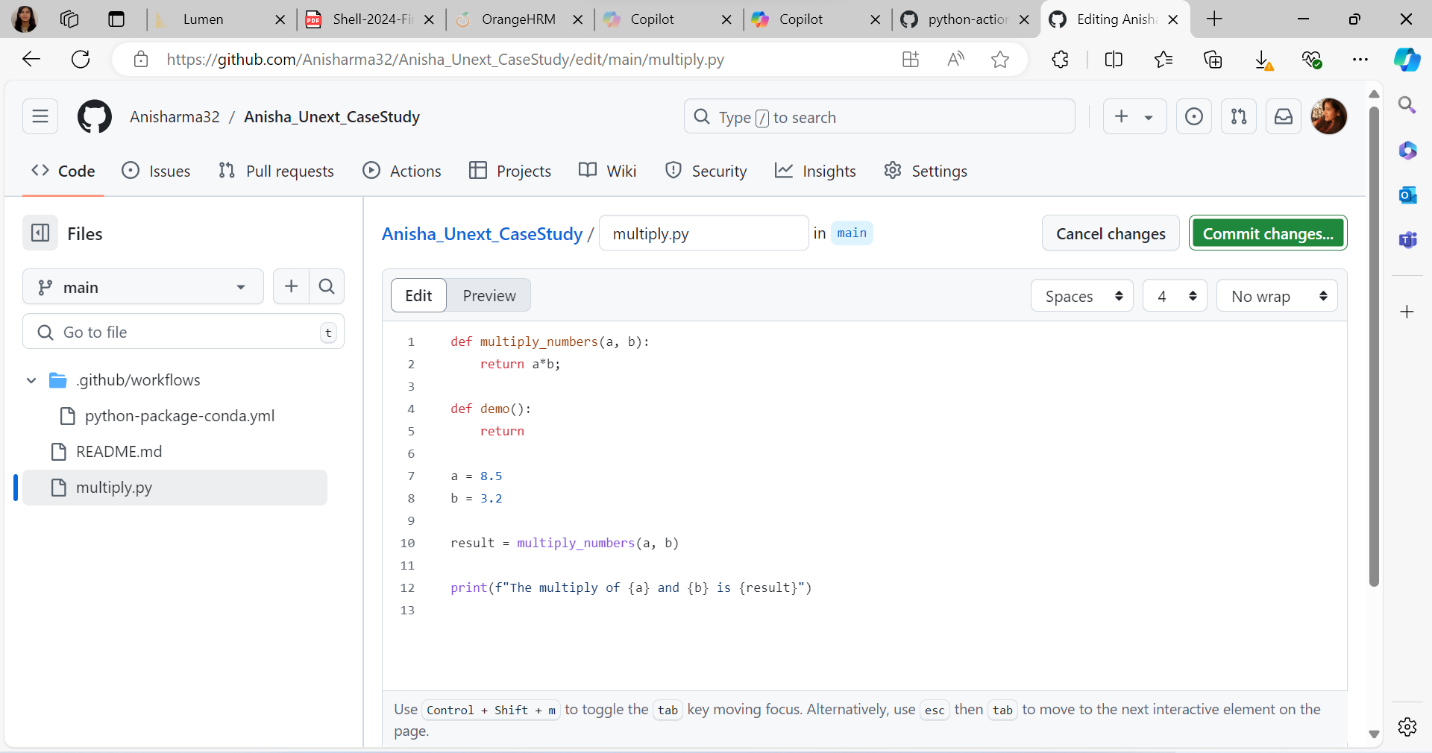


Figure 33. A function demo() has been edited in the python file.

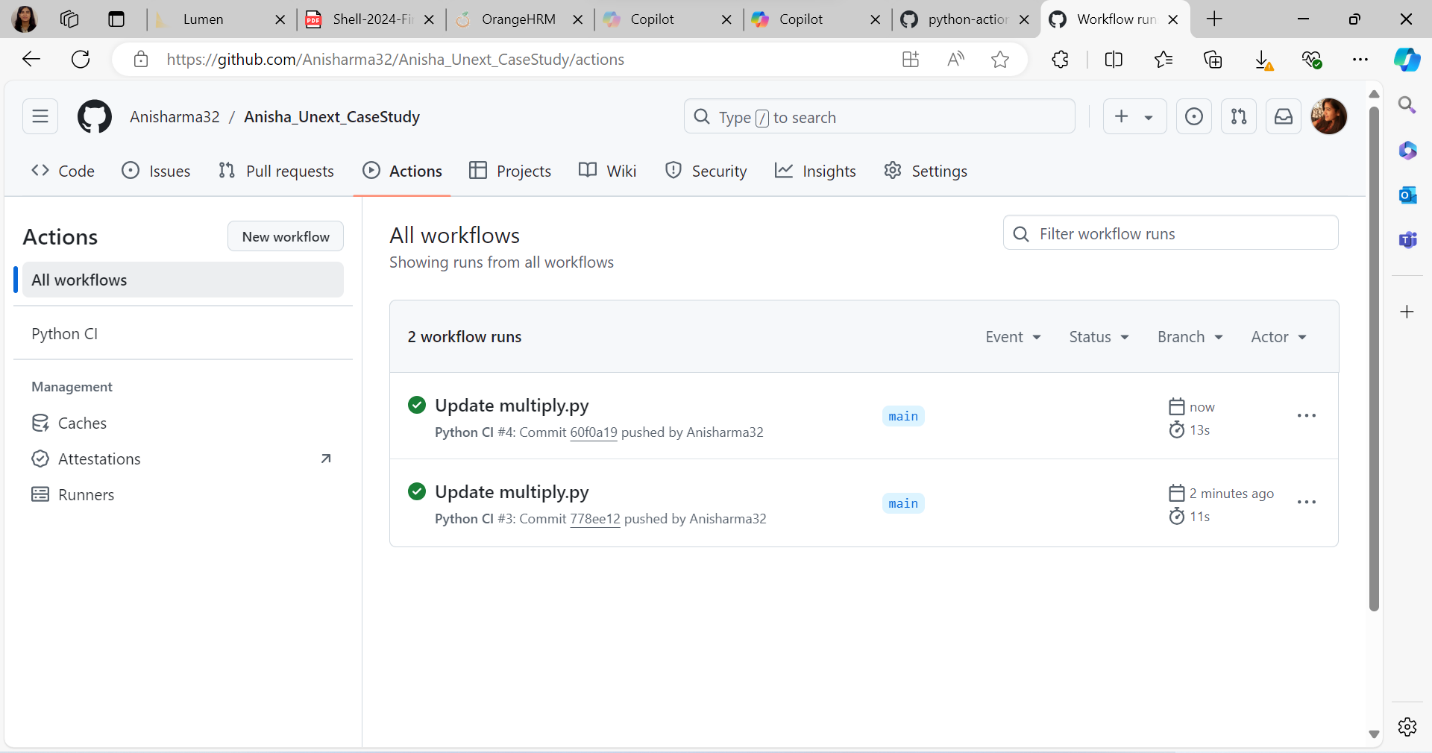


Figure 34. The updated python file is again automated.

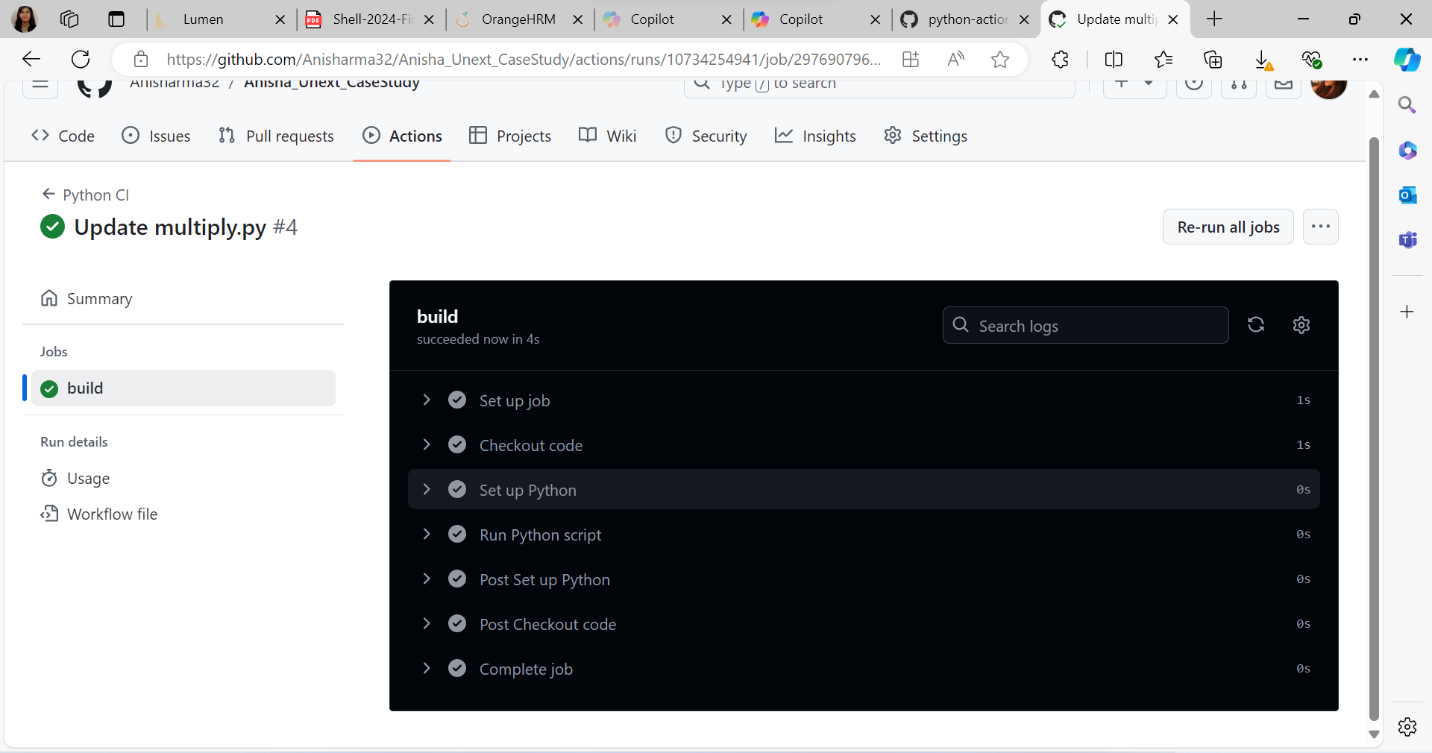


Figure 35. All the automations have been passed.