Dot Net Phase 4 – Custom Support Logger

Phase-End Project Problem Statement

Github: https://github.com/11812142/EndProject

Phase-End Project 3

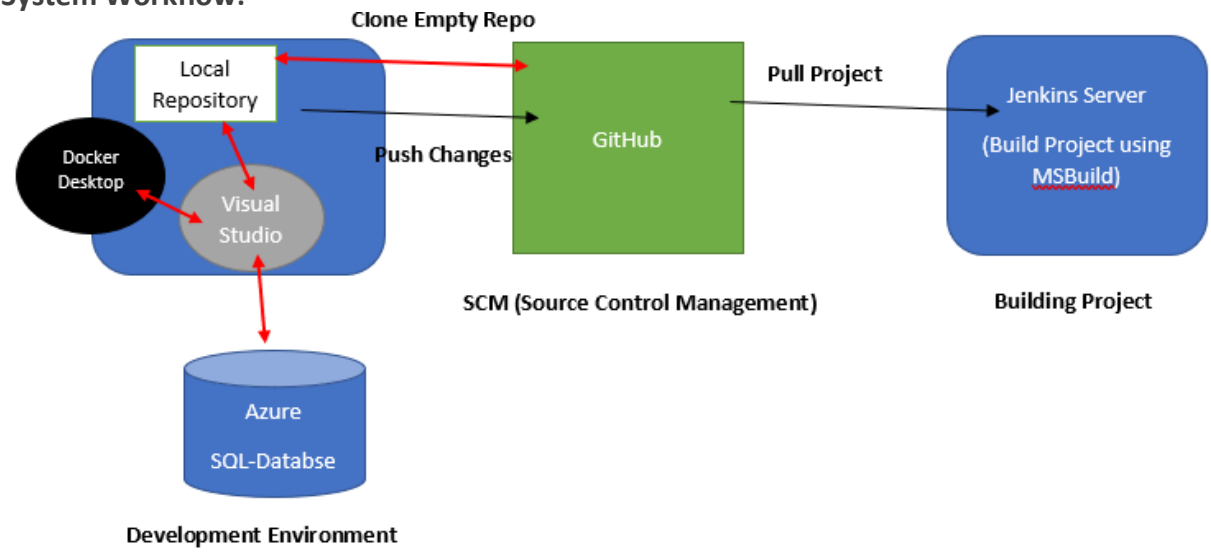
Custom Support Logger

Prerequisites: C# Basics, Basics of MVC (Model-View-Controller), Docker (Docker Desktop), NUnit (Moq Framework), Jenkins, Azure Core Services (Resource Group, Azure SQL Database), Basics of Git, and the Git extension in Visual Studio

Case Study:

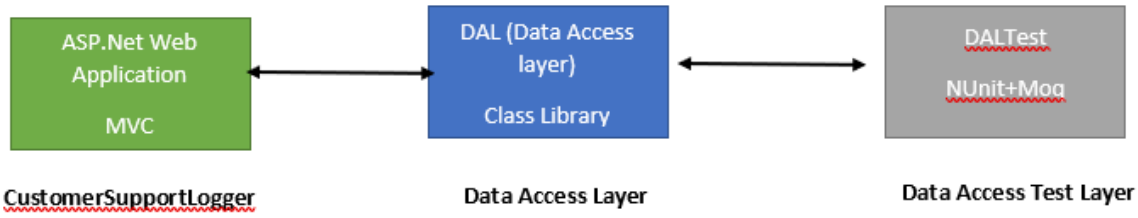
Simplona Tech. Solutions have multiple customers for their ERP application, and they have a dedicated team to provide support for this. They need to develop a web application that helps them to record their customer support executive's daily contribution toward customer support activities.

System Workflow:



- You need to create an ASP.Net MVC application within the Docker container and push it to GitHub by cloning the empty repository.
- You need to create Database Azure SQL Database to store data.
- After implementing all the functionalities, you must push the project to GitHub and then build a project using the Jenkins server by pulling the project from the GitHub repository.

Project Workflow:



1. **Project Creation:**

   Create a repository on GitHub and clone the empty repository on the local machine. Inside the local repository, create the below projects using Visual Studio:

   a. Create a class library project (DAL)

   b. Create a class library project (DALTest)

   c. Create ASP.Net Web MVC application project with Docker Support

2. **Functionalities:**

   A. Create an SQL Server Database on Azure with the structure given below:

Table: **UserInfo**

Field Name	**Type**	**Constraints**
UserId	int	Primary Key
Email	nvarchar(100)	
Password	nvarchar(20)	

Table: **CustLogInfo**

Property Name	**Type**	**Constraints**
LogId	int	Primary Key
CustEmail	nvarchar(100)	
CustName	nvarchar(50)	

LogStatus	nvarchar(50)	
UserId	int	Foreign key
Description	nvarchar(50)	

B. **DAL:** In this layer, add the entity data model by selecting SQL Server database which is created on Azure and add the below functionalities using the data repository pattern
    a. Add DAL class and write a function to validate the user (customer support executive) from the UserInfo table using entity framework
    b. Add one more DAL class to save complaint log information to CustLogInfo table using the entity framework

C. **DALTest**: In this layer, test functionalities written in DAL using NUnit and Moq framework, such as UserInfo and CustLogInfo functionalities

D. **CustomerSupportLogger:** This is an MVC application to consume functionalities you have written in DAL:
    a. Develop a login page to validate user (customer support executive) as shown in the output
    b. If the user is valid, then develop a page to add customer complaints as shown in the output
    c. Debug this application on the Docker container as shown in the output

3. Push the entire project over a GitHub repository using the Visual Studio Git extension

4. After pushing a project to GitHub, create a job in Jenkins to build a project which has been pushed over a GitHub

5. Create a **FreeStyle** project in Jenkins, and configure it as mentioned in the next few steps

6. Configure Git **Source Code Management,** add GitHub project URL, and set branch as **Main**

7. To Trigger a build, select **Poll SCM** and schedule the build in such a way that the project triggers the build process after each hour

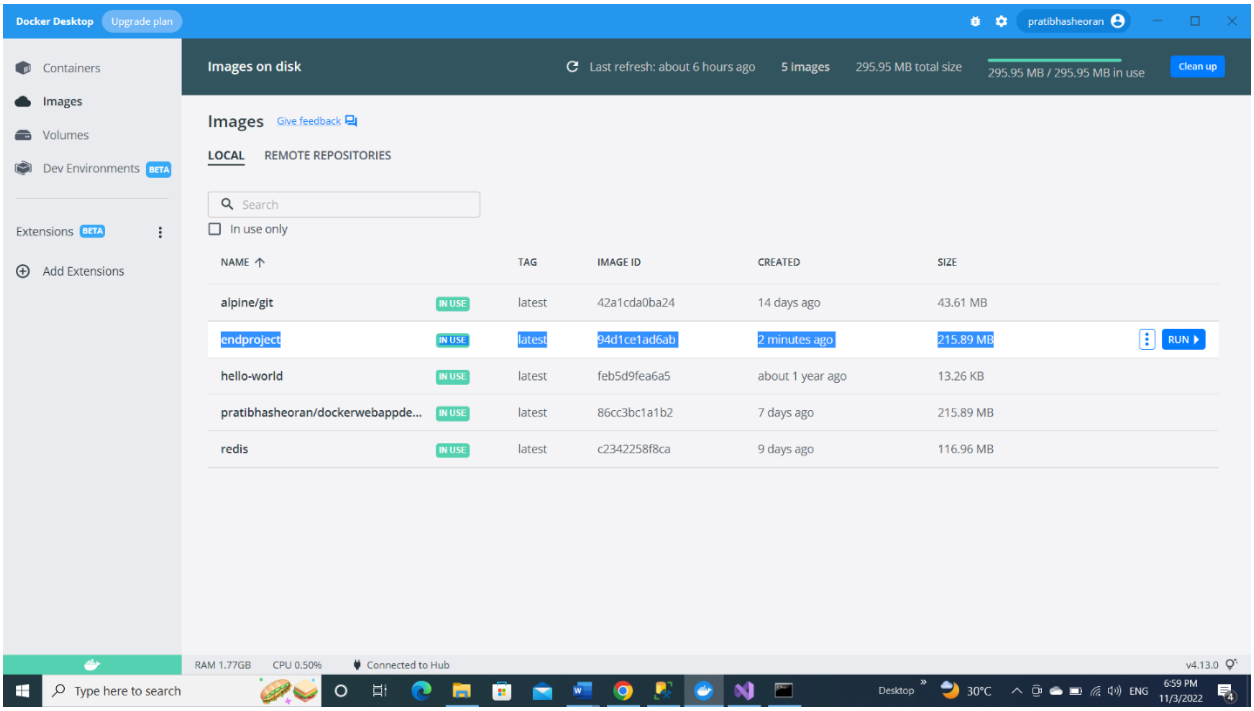
8. Set **Build** as **(Build a Visual Studio project or Solution using MSBuild)**

9. Select MSBuild version installed in Jenkins, and write .sln file name with the relative path, which exists inside the GitHub repository

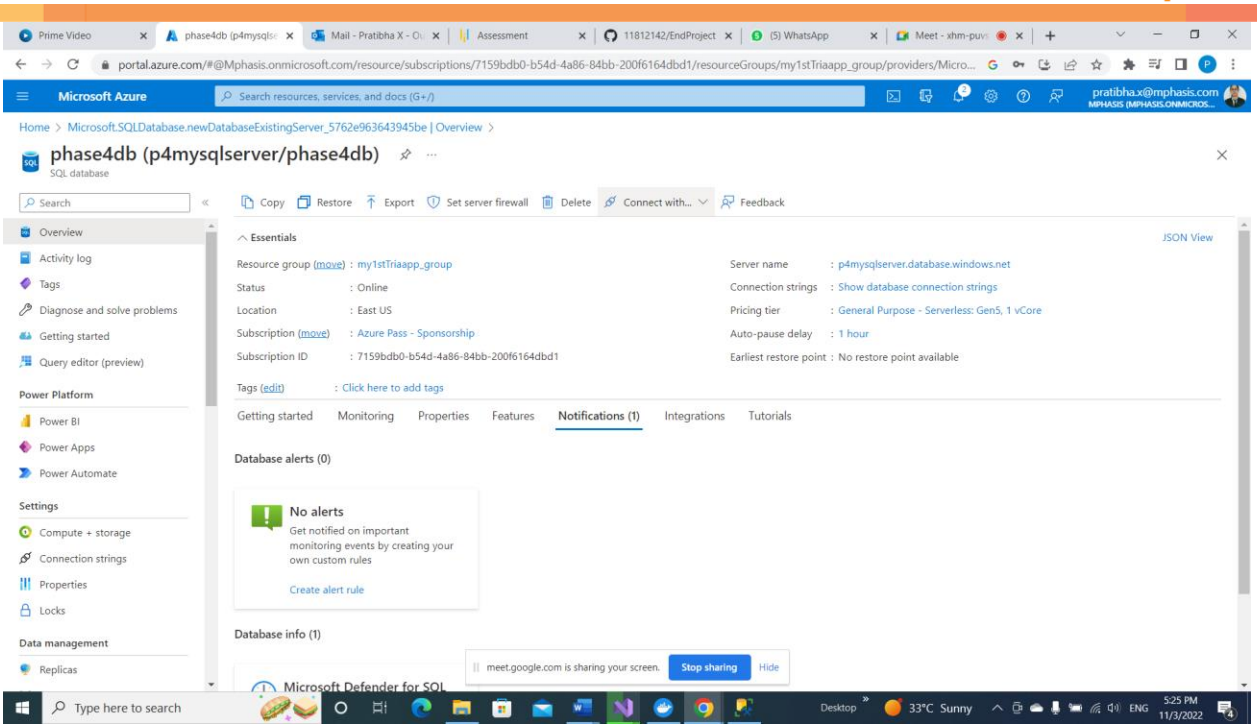
10. Build a Project in Jenkins

Sample Input/Output:

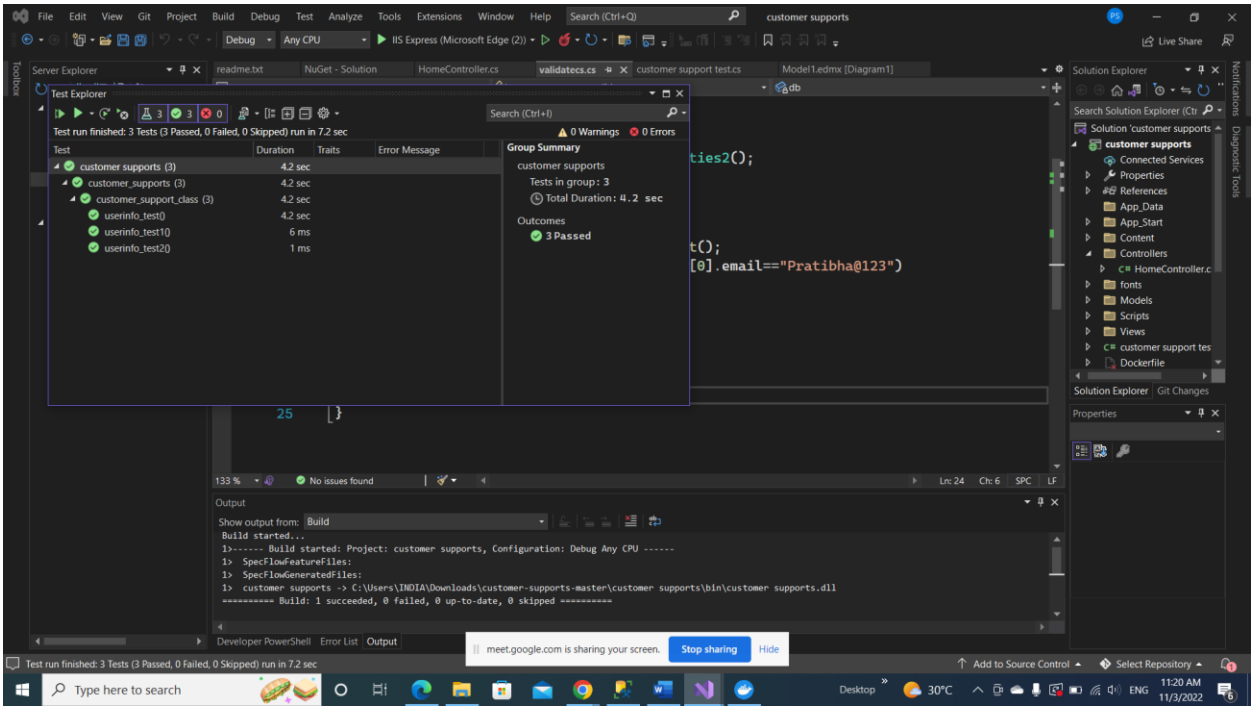
1. Docker Desktop output after creating MVC application with docker support



2. After creating SQLServer database on Azure

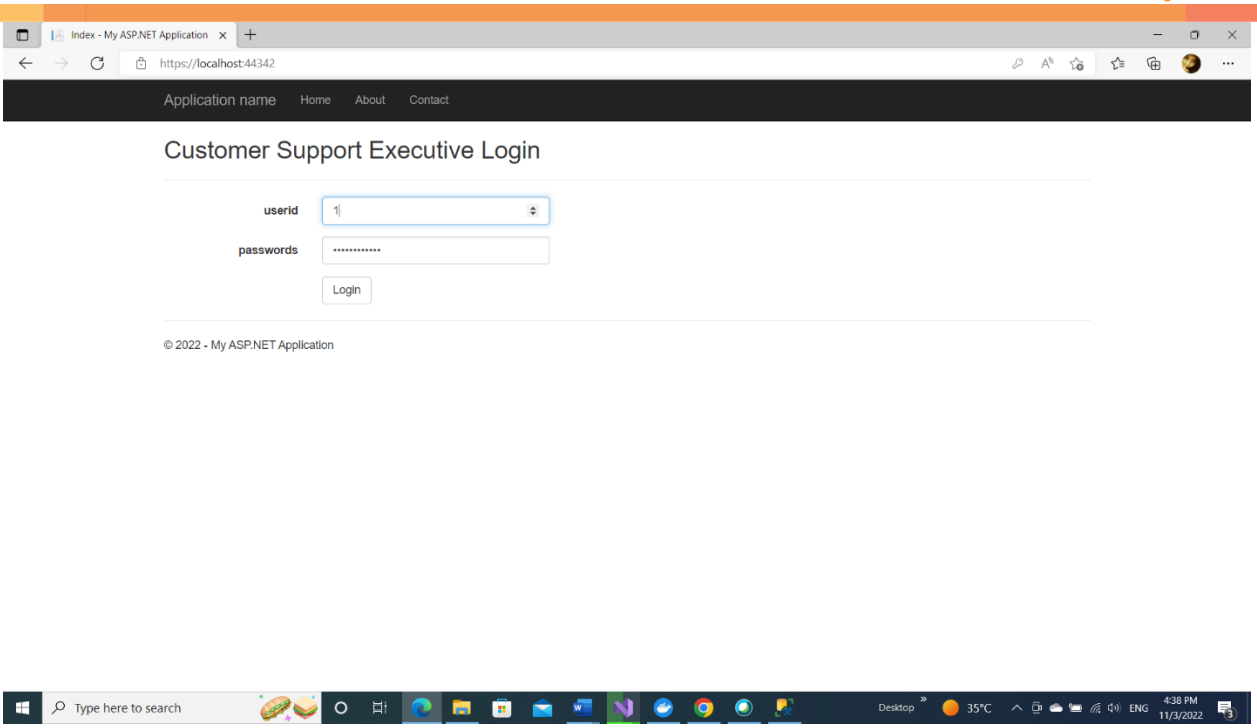


3. After testing all test cases

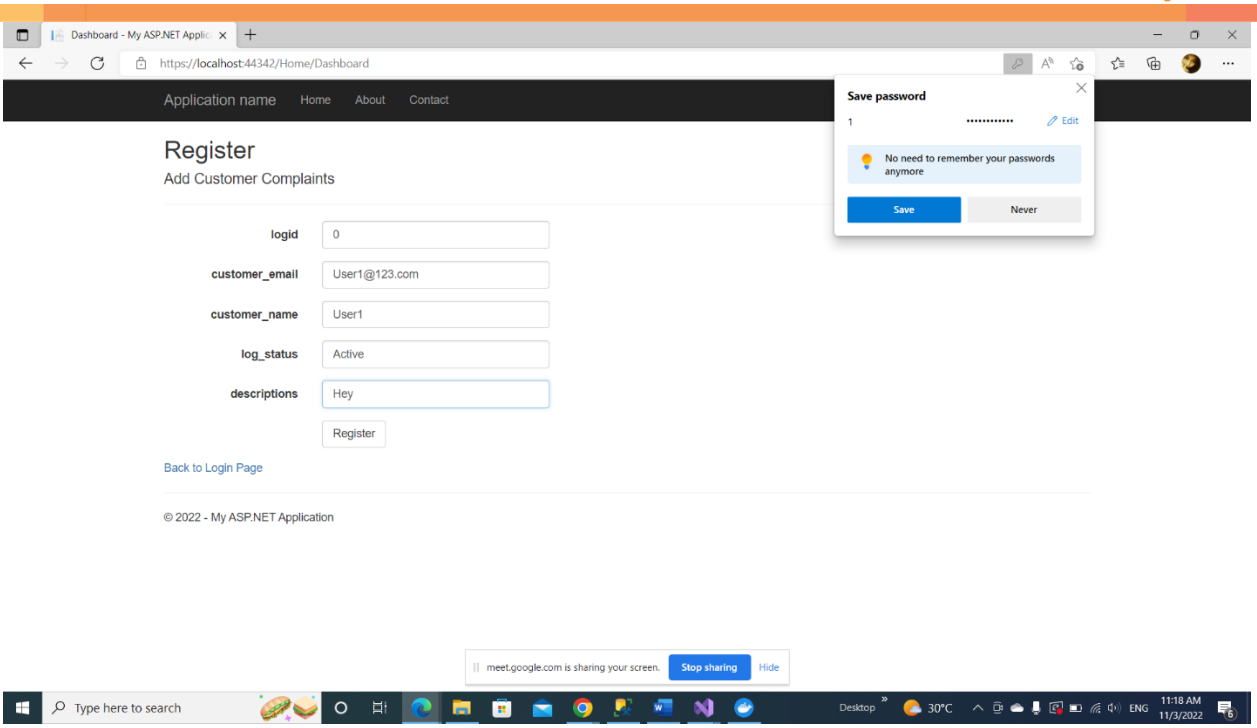


4. After executing MVC application

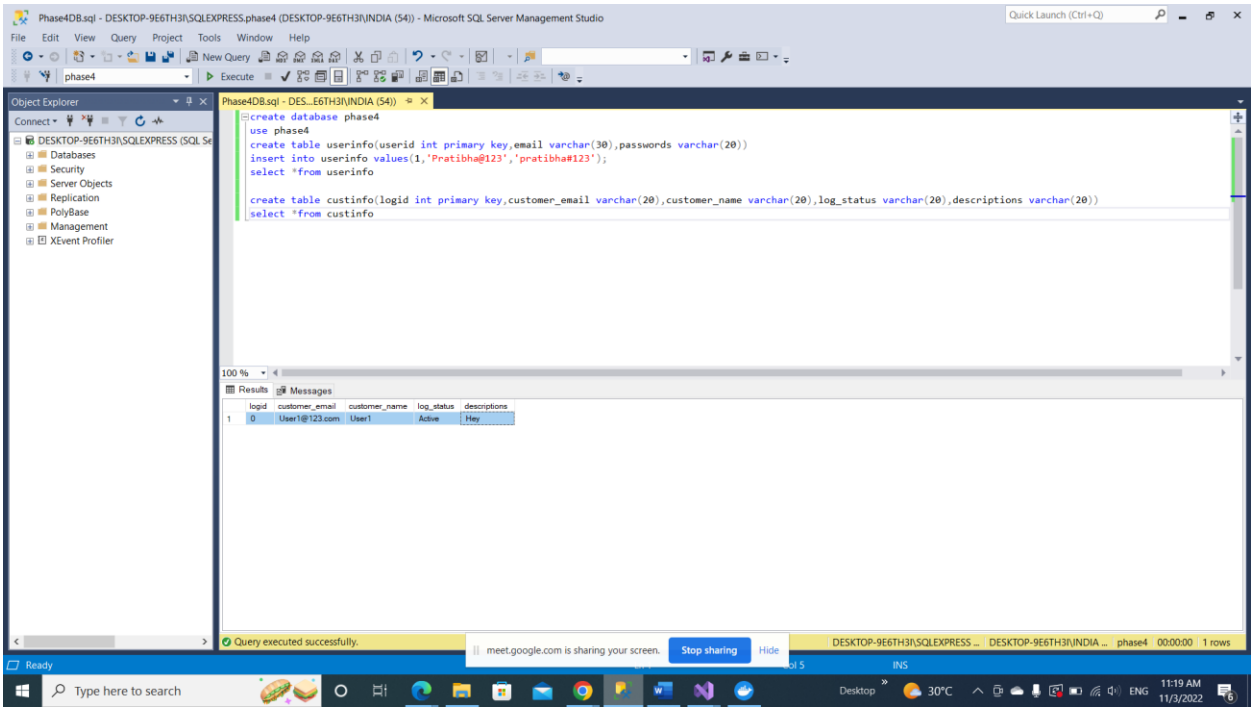
a. Customer executive login



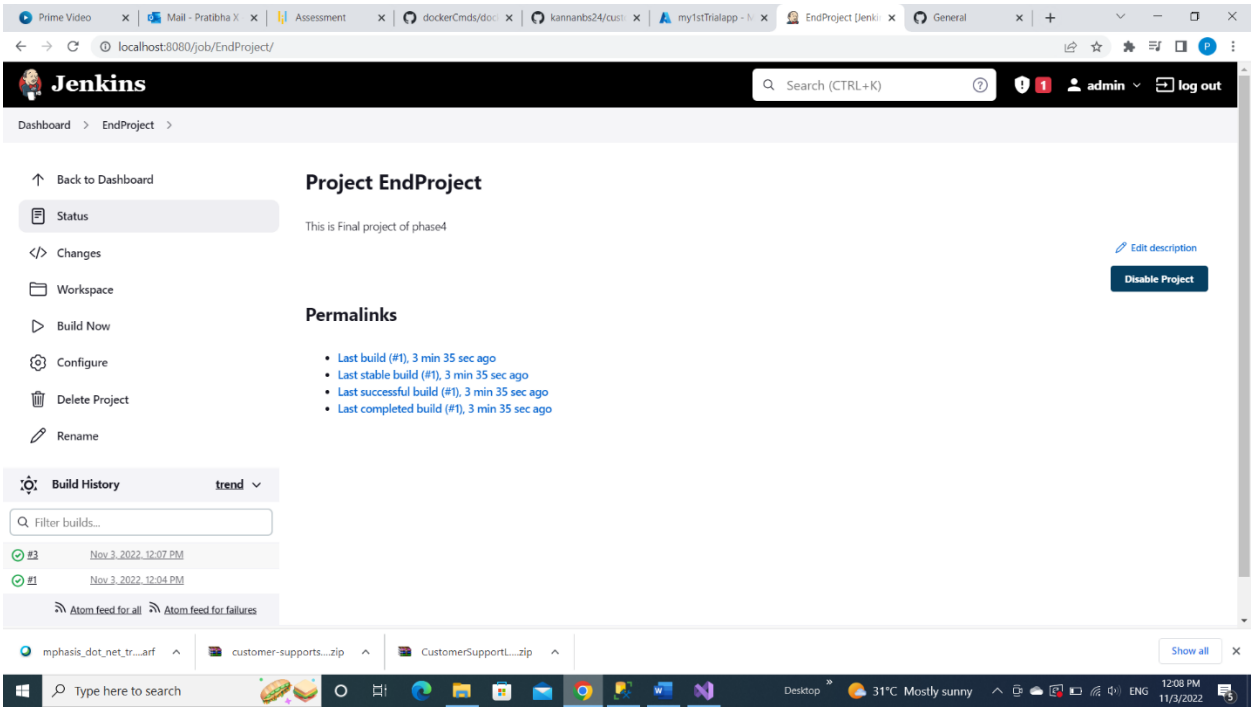
b. After saving log complaint information



c. Verifying results into the database by connecting to SQL Server Management Studio (SSMS)



5. After successfully building a project in Jenkins


Jenkins
Search (CTRL+K)
admin
log out
Dashboard
EndProject
Back to Dashboard
Status
Changes
Workspace
Build Now
Configure
Delete Project
Rename
Project EndProject
This is Final project of phase4
Edit description
Disable Project
Permalinks
Last build (#1), 3 min 35 sec ago
Last stable build (#1), 3 min 35 sec ago
Last successful build (#1), 3 min 35 sec ago
Last completed build (#1), 3 min 35 sec ago
Build History
trend
Filter builds...
#3 Nov 3, 2022, 12:07 PM
#1 Nov 3, 2022, 12:04 PM
Atom feed for all
Atom feed for failures
Show all
Type here to search
31°C Mostly sunny
ENG
12:08 PM
11/3/2022