# Customer Support Logger

## Project Overview

To streamline and manage the customer support process, create a web application that logs daily activities of the customer support executives. This guide serves as a roadmap for creating such an application.

## Objectives

Develop an ASP.Net MVC web application.

Containerize the application with Docker.

Maintain source code using Git and GitHub.

Incorporate Azure SQL Database for data storage.

Automate the build process using Jenkins.

## Project Creation & Setup

Utilizing Visual Studio, set up the following projects inside the project:

A class library project (DAL)

A Unit Test class(DALTest)

An ASP.Net Web MVC application project with Docker Support

## Database Creation & Management

Establish an SQL Server Database on Azure.

Create the necessary tables (UserInfo & CustLogInfo) with the specified fields and constraints.

## Development of the Data Access Layer (DAL)

Create the entity data model in the DAL by integrating the SQL Server database from Azure.

Implement functionalities to:

Validate the user from the UserInfo table.

Save complaint log information to the CustLogInfo table.

**Testing the DAL**

Employ NUnit and the Moq framework to test functionalities within the DAL, covering both UserInfo and CustLogInfo operations.

**Building the MVC Application**

Develop a user login page for the customer support executive.

If authenticated, present the executive with a page to log customer complaints.

Ensure that the application runs seamlessly in a Docker container.

**Source Code Management & Continuous Integration**

Commit and push your complete project to the GitHub repository via the Visual Studio Git extension.

Set up Jenkins and create a FreeStyle project.

Configure Jenkins to:

Fetch from the GitHub repository.

Build the project.

**Monitoring & Validation**

Ensure Docker Desktop correctly displays the output upon creating the MVC application with Docker support.

Post the development of the SQL Server database on Azure, cross-verify its creation.

Continuously monitor the MVC application's execution:

During the login process for the customer executive.

Upon logging complaint details.

Ensure the results synchronize with the database by using SQL Server Management Studio (SSMS).

After every Jenkins job upload, validate the outcome for correctness.

**Deliverables**

A Dockerized ASP.Net MVC web application.

A repository on GitHub containing the complete project.

An Azure SQL Database structured to hold user information and complaint logs.

Jenkins configurations and build results.

**Conclusion**

This project not only enhances our grasp of ASP.Net MVC, Docker, Azure, and Jenkins but also mirrors real-world scenarios where applications must be developed, containerized, stored in cloud databases, and continuously integrated. It's a holistic approach to understanding the end-to-end lifecycle of web application development and deployment.

Github Link: https://github.com/PrashastVats1/CustomSupportProj

Dockerhub Link: https://hub.docker.com/repository/docker/prashastvats/customsupport/general