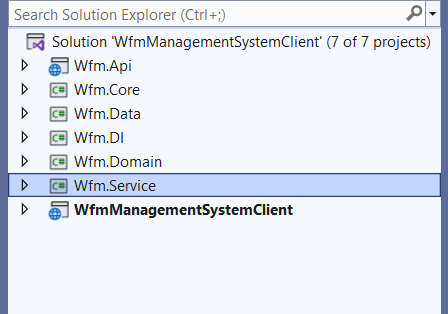
**Work Force Management - Final Document**

1. Client web app -> Dot net core MVC project
2. API calls -> API projects
3. Token Authentication for API call while Login implemented.
4. Class Library projects are created for Data , DI and service purposes.
5. Data is dynamically taken through ADO dot net concept.

**Structure Of My Solution**



**Tables Used**

1. **Employee table**

CREATE TABLE Employee (

EmpId int NOT NULL PRIMARY KEY,

EmpName varchar(50) NOT NULL,

Status varchar(50) NOT NULL,

ManagerName varchar (30),

wfm\_MemberName varchar (30),

Email varchar(250) NOT NULL ,

LockStatus varchar(50),

Experience decimal (5,0),

Profile varchar(255),

LockId int NOT NULL FOREIGN KEY REFERENCES SoftLock(LockId)

);

**2) Skill table**

CREATE TABLE Skill(

SkillId int NOT NULL PRIMARY KEY ,

SkillName varchar(30) NOT NULL

);

**3) SkillMap**

CREATE TABLE SkillMap(

EmpId int FOREIGN KEY REFERENCES Employee(EmpId) ,

SkillId int FOREIGN KEY REFERENCES Skill(SkillId)

);

**4) User**

CREATE TABLE User(

UserName varchar(50) NOT NULL PRIMARY KEY,

Password varchar(50) NOT NULL,

FirstName varchar(30) NOT NULL ,

Role varchar(30) NOT NULL,

Email varchar(50)

);

**5) SoftLock**

CREATE TABLE SoftLock(

LockId int NOT NULL PRIMARY KEY,

EmpId int NULL,

RequestedManager varchar(50) NULL,

RequestedDate DATETIME,

LockStatus varchar(50) NOT NULL,

IsLocked Bit NULL,

LastUpdatedDate DATETIME,

RequestMessage varchar(255) NOT NULL,

WfmRemark varchar(255) NOT NULL,

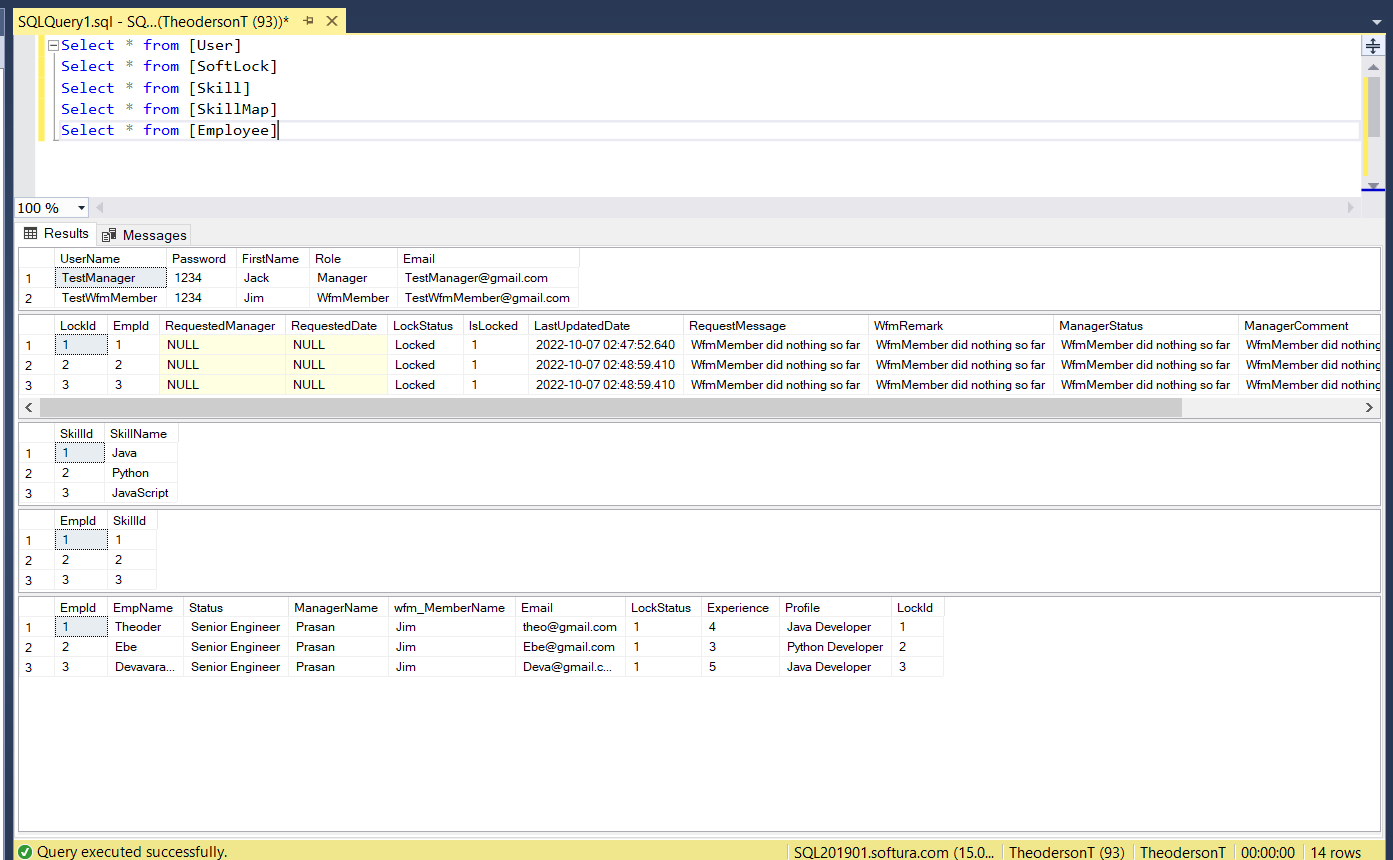
ManagerStatus varchar(50) NOT NULL,

ManagerComment varchar(255) NOT NULL,

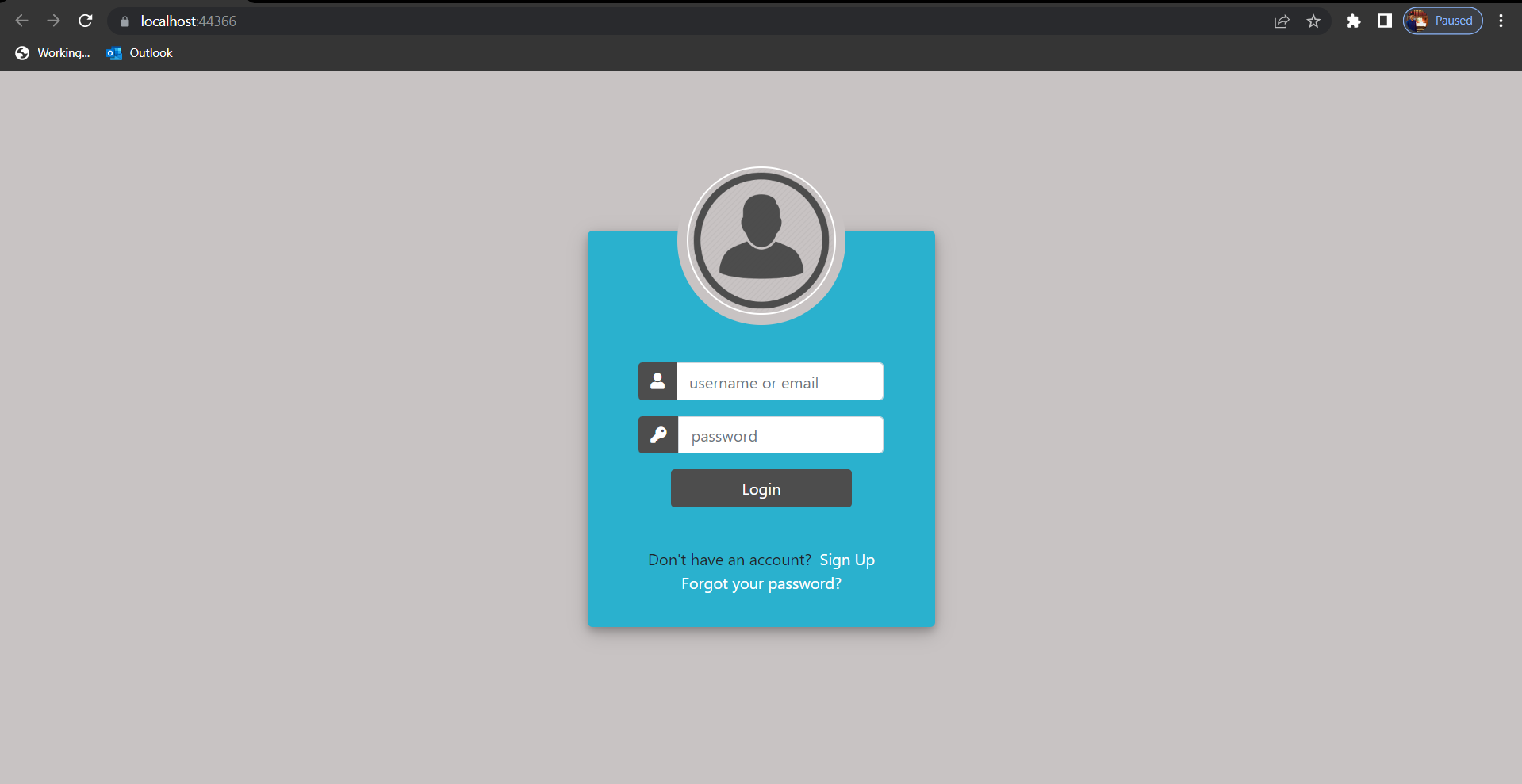
ManagerLastUpdatedDate DATETIME

)

These are the tables used for maintaining the data.



**LOGIN PAGE :**



Normal login page with Authentication to API has been done.

When User clicks in the login button,

It will call the following method.

public ActionResult Login(UserProfile user)

{

try

{

string Uri = string.Format("http://localhost:44586");

RestRequest request = new RestRequest(Uri);

var response = client.ExecuteTaskAsync<List<Core.Models.LoginCriteria>>(request);

if (response.Role == "Manager")

{

return RedirectToAction("Index", "Manager");

}

else

{

return RedirectToAction("Fail", "WfmMember");

}

}

catch

{

return View();

}

}

There is an authenticated api call. The described method will vall the api controller, which in terms call the sp for the required actions. The service inturn will call the repo method for sp execution.

The following sp will be executed. To get the user list based on the Login inputs.

CREATE PROCEDURE sp\_GetUserForLogin

(

@UserName varchar(255),

@Password varchar(255)

)

AS

BEGIN

SELECT

UserName,

Password,

FirstName,

Role

Email FROM [User]

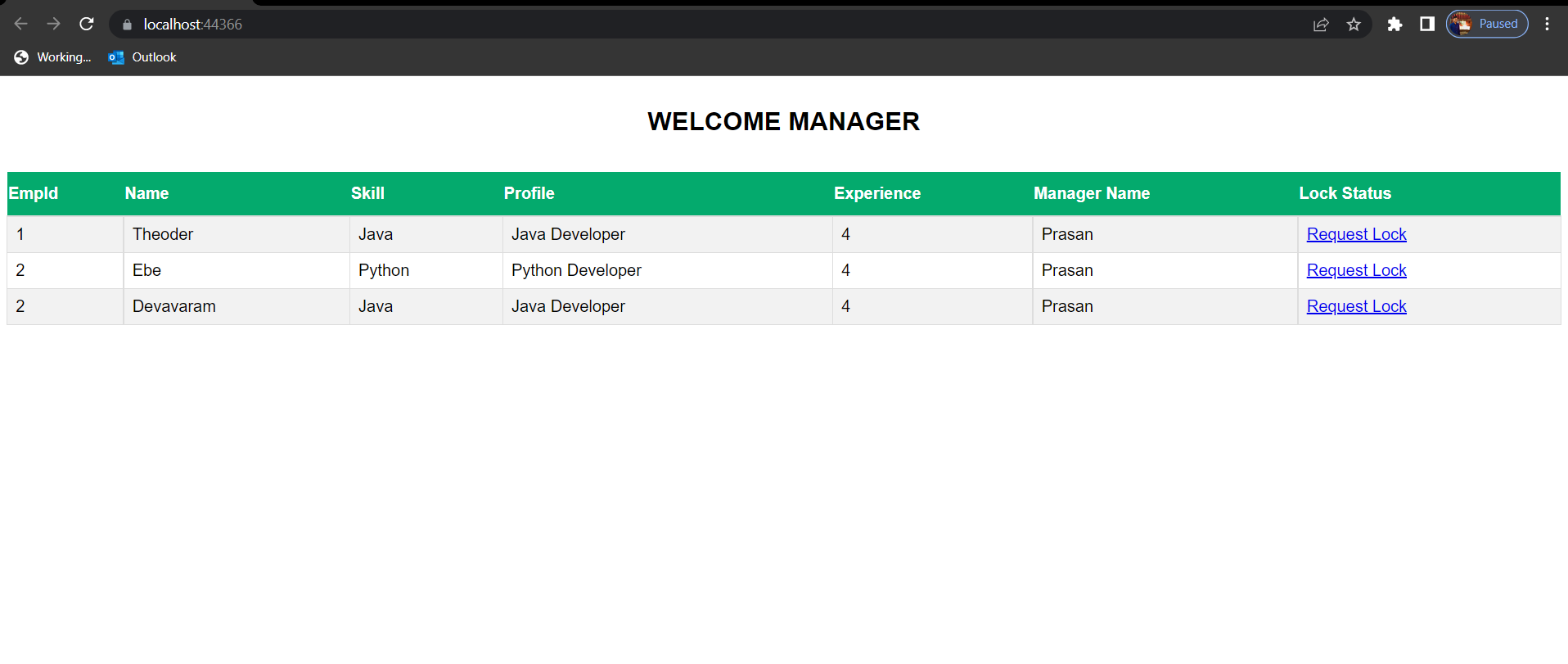
WHERE

UserName = @UserName AND Password = @Password

END

Based on the role , UI will be as Manager view and WfmMember view.

**Manager Login View**



After logging in as Manager , manager page will open. It will call the following method.

public ActionResult Index()

{

List<Manager> details = new List<Manager>();

string Uri = string.Format("http://localhost:44586");

RestRequest request = new RestRequest(Uri);

var response = client.ExecuteTaskAsync<List<Core.Models>>(request);

details=response.Result;

return View("Manager" , details);

}

The api method will call the repo method to execute the following sp to get details.

CREATE PROCEDURE sp\_GetDetailsForManagerPage

(

)

AS

BEGIN

SELECT

emp.EmpId,

emp.EmpName,

sk.SkillName,

emp.Profile,

emp.Experience,

emp.ManagerName,

sftlck.IsLocked

FROM Employee emp

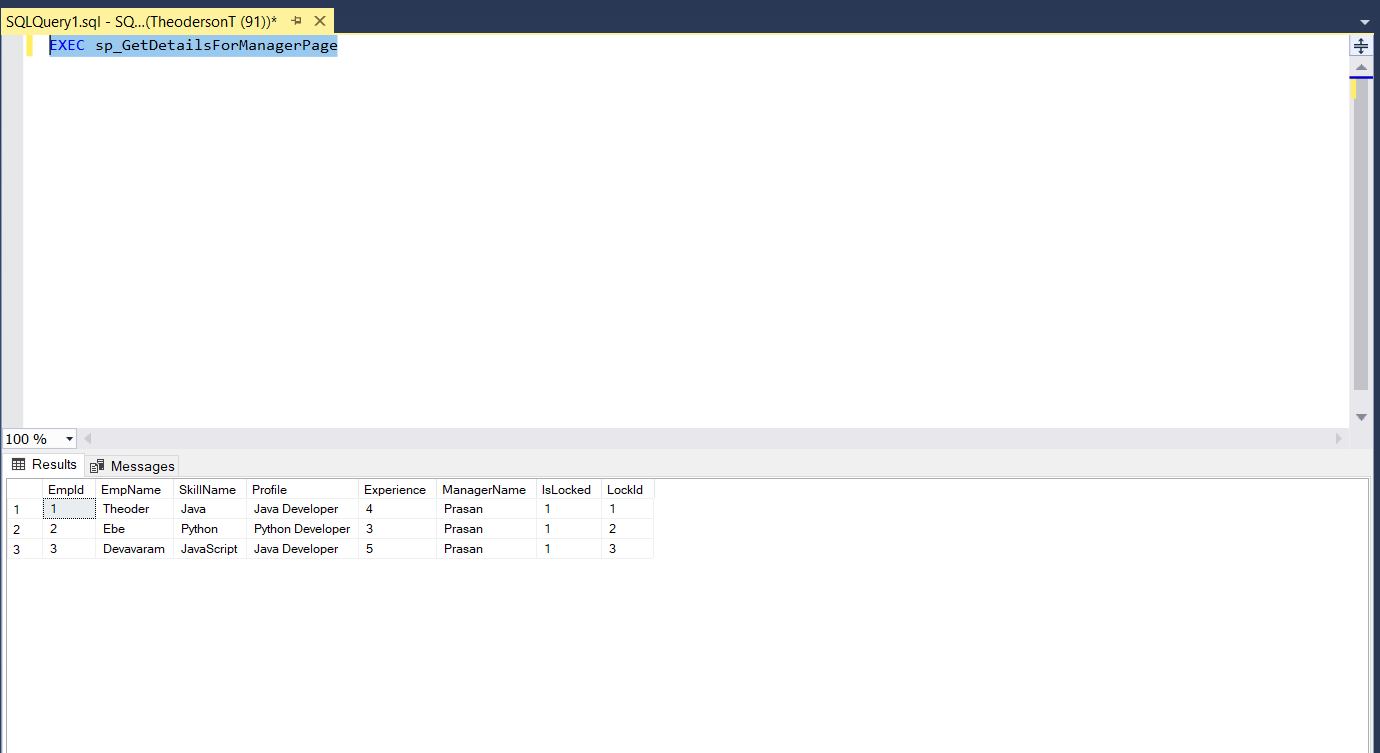
JOIN SkillMap skm ON skm.EmpId = emp.EmpId

JOIN Skill sk ON skm.SkillId = sk.SkillId

JOIN SoftLock sftlck ON sftlck.LockId = emp.LockId

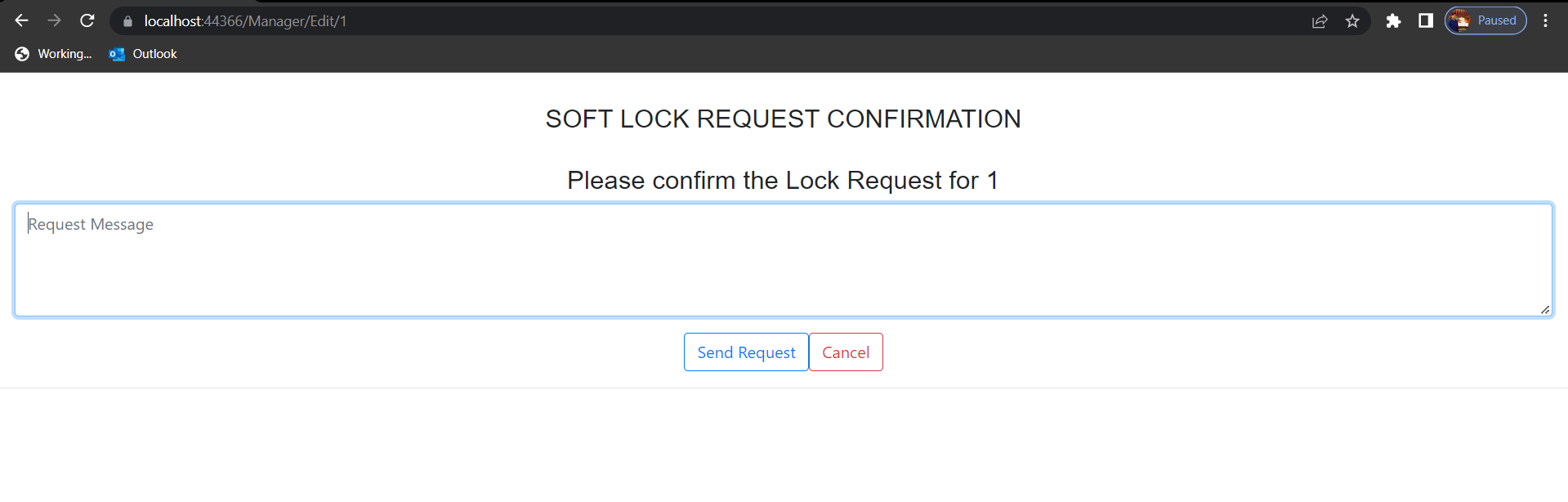
where sftlck.IsLocked = 1

END

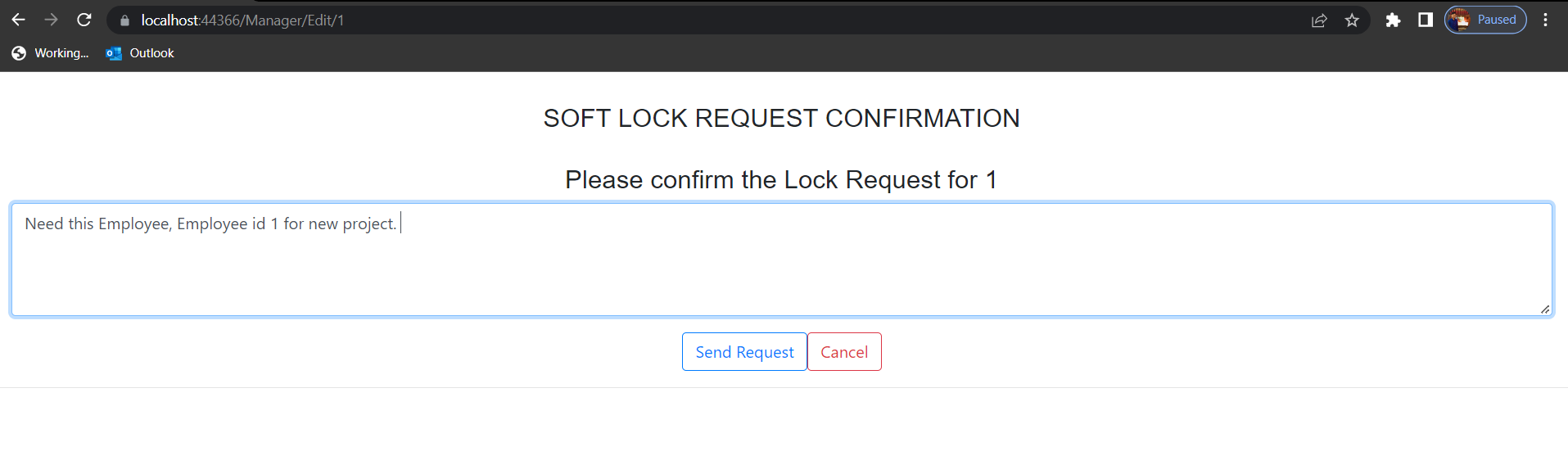


In this sp joints and conditions for the listing is handled to get the expected result.

After clicking the Lock request button, the following page will be opened.



Lock Request id value is dynamically called through view by @Model.EmpId



After sending send request it will call the following method in the client.

[HttpGet]

public IActionResult Edit(int id)

{

Manager e = details.Where(x => x.LockId == id).SingleOrDefault();

return View("EditManager",e);

}

// POST: ManagerController/Edit/5

[HttpPost]

public IActionResult Edit(Manager detail)

{

int id = detail.LockId;

Manager oldOne = details.Where(x => x.LockId == id).SingleOrDefault();

details.Remove(oldOne);

details.Add(e);

return RedirectToAction("Manager");

}

This controller will call the api controller method which in turn call the sp sp\_LockStatusChangeForManager.

CREATE PROCEDURE sp\_LockStatusChangeForManager

(

@LockId int,

@RequestMessage varchar(255)

@FirstName varchar(50)

)

AS

BEGIN

UPDATE SoftLock

SET

RequestedDate = GETDATE(),

LockStatus = 'Pending',

RequestMessage = @RequestMessage,

LastUpdatedDate = GETDATE(),

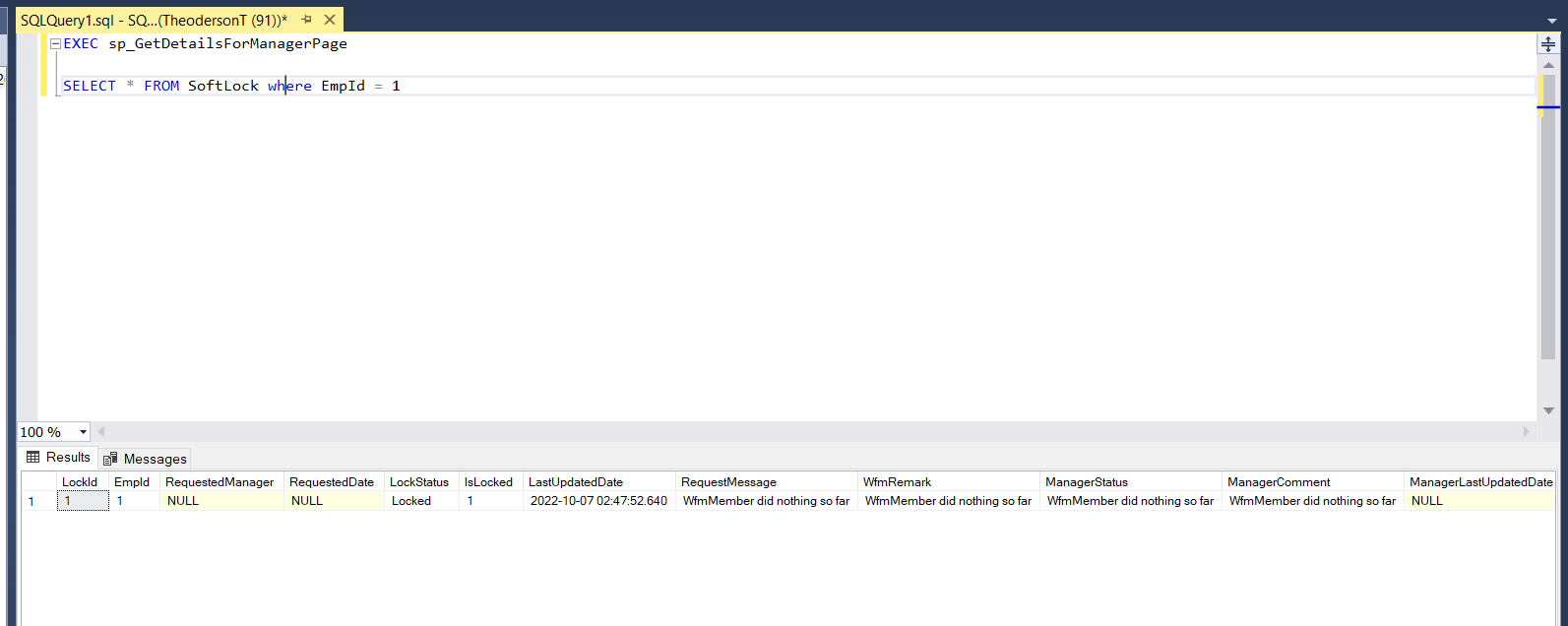
RequestedManager = @FirstName

WHERE

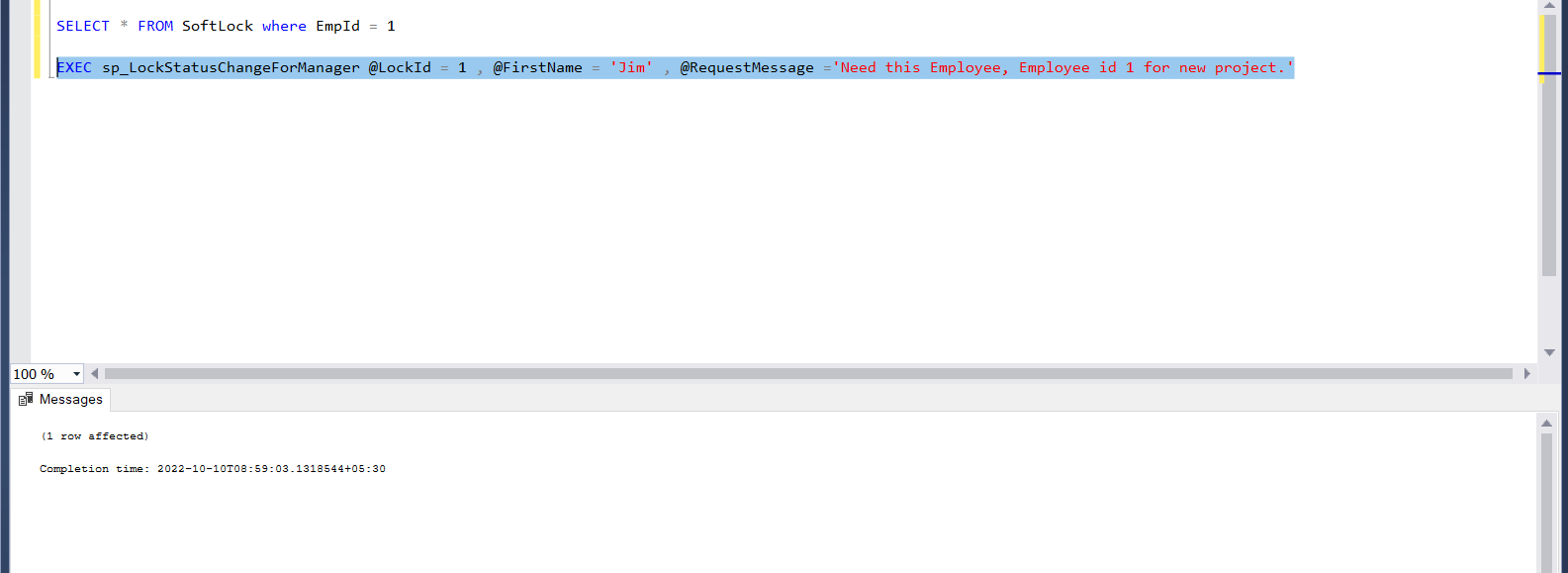
LockId = @LockId

END

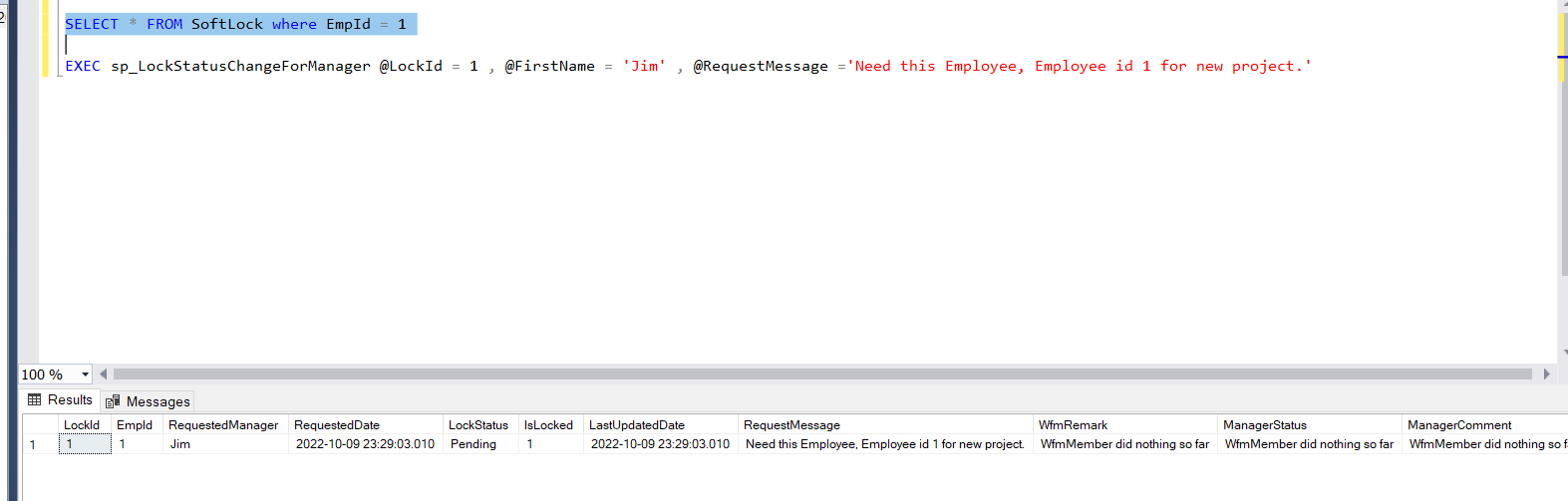
BEFORE UPDATE :



**AFTER EXECUTING sp\_LockStatusChangeForManager:**



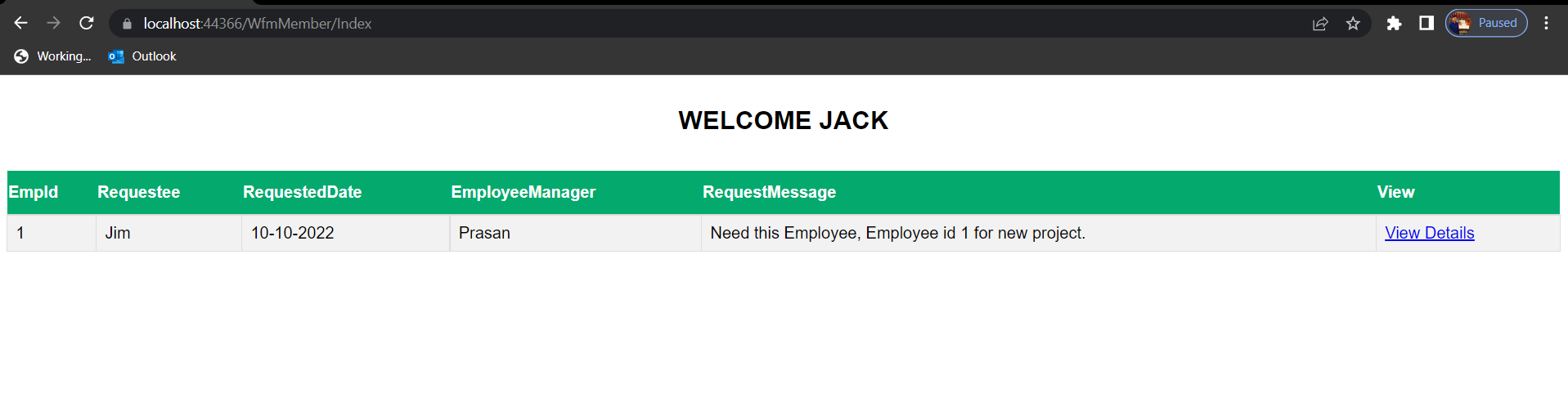
**AFTER UPDATE :**



After sending the request, means after update of soft lock table , it will redirect to the initial page.

WFM MANAGER FIRST PAGE :

After logging in as Wfm Member , this page will open. It will call the following method.



After logging in as Manager , manager page will open. It will call the following method.

public ActionResult Index()

{

string Uri = string.Format("http://localhost:44586");

RestRequest request = new RestRequest(Uri);

var response = client.ExecuteTaskAsync<List<Core.Models>>(request);

details = response.Result;

return View("WfmMember", details);

}

The api method will call the repo method to execute the following sp to get details.

CREATE PROCEDURE sp\_wfmMemberLoginGetDetails

(

@UserName varchar(255),

)

AS

BEGIN

SELECT

emp.EmpId,

sftlck.RequestedManager,

sftlck.RequestedDate,

emp.ManagerName,

sftlck.RequestMessage

FROM

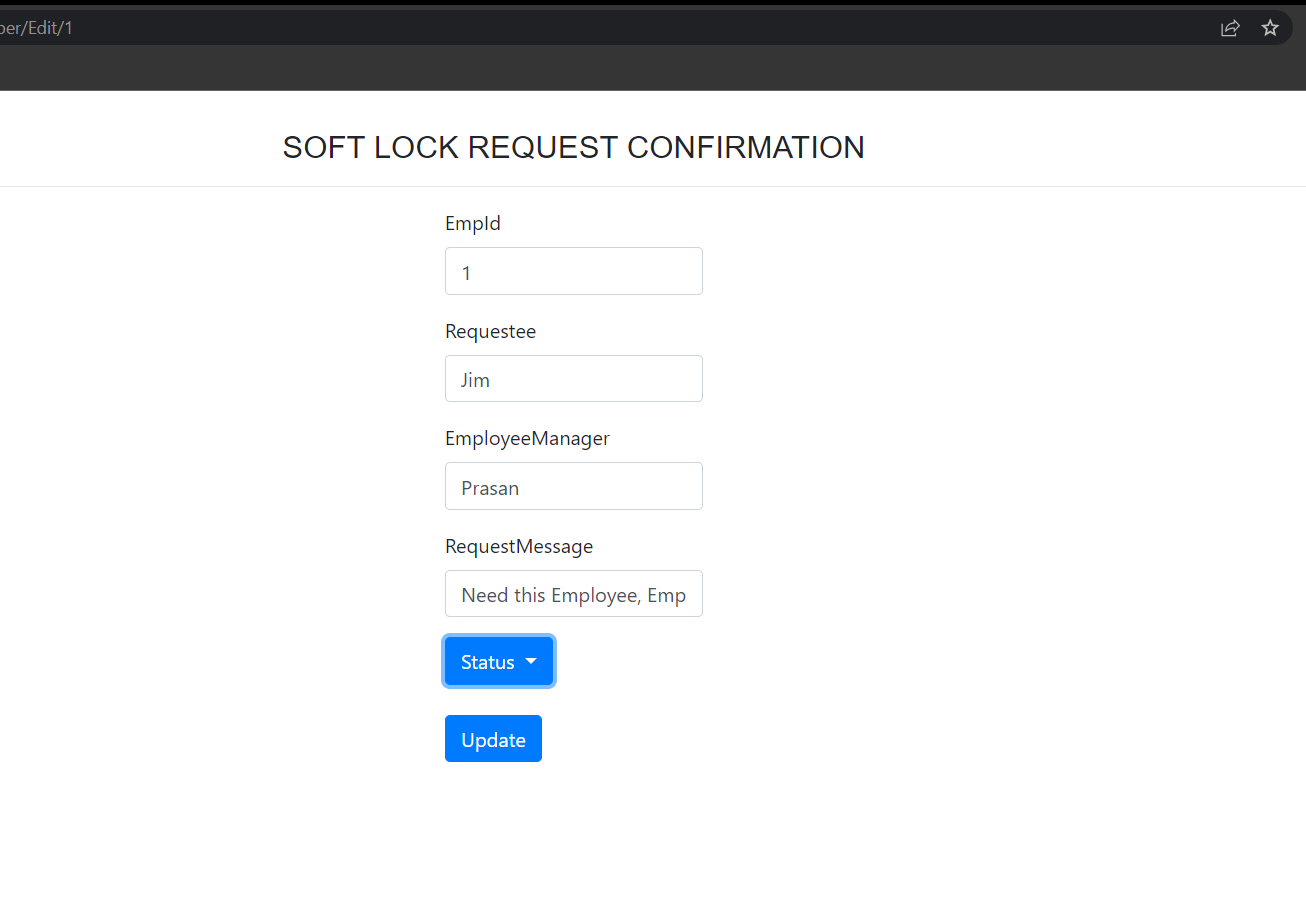
Employee emp

JOIN SoftLock sftlck ON SoftLock.LockId = Employee.LockId

WHERE emp.wfm\_MemberName = @UserName

END

After clicking the view details the following page will be open.



By clicking the status dropdown method, the Wfm Member can select the options. He can approve and reject.

public ActionResult Edit(int id)

{

WfmMember e = details.Where(x => x.EmpId == id).SingleOrDefault();

return View("WfmMemberEdit", e);

}

// POST: WfmMemberController/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(WfmMember details)

{

try

{

int id = details.EmpId;

string Uri = string.Format("http://localhost:44586");

RestRequest request = new RestRequest(Uri);

var response = client.ExecuteTaskAsync<List<Core.Models>>(request);

return View("Index");

}

catch

{

return View();

}

}

It will call the following sp for the update.

CREATE PROCEDURE sp\_ApprovalStatusForSendRequestByWfmMember

(

@EmpId int ,

@ManagerStatus varchar(255),

@ManagerComment varchar(255)

)

AS

BEGIN

DECLARE @LockId int

Select @LockId = LockId FROM Employee where EmpId = @EmpId

UPDATE SoftLock

SET

IsLocked = CASE WHEN @ManagerStatus = 'Unlock' THEN 0 Else 1 END,

LockStatus = @ManagerStatus,

LastUpdatedDate = GETDATE() ,

ManagerLastUpdatedDate = GETDATE(),

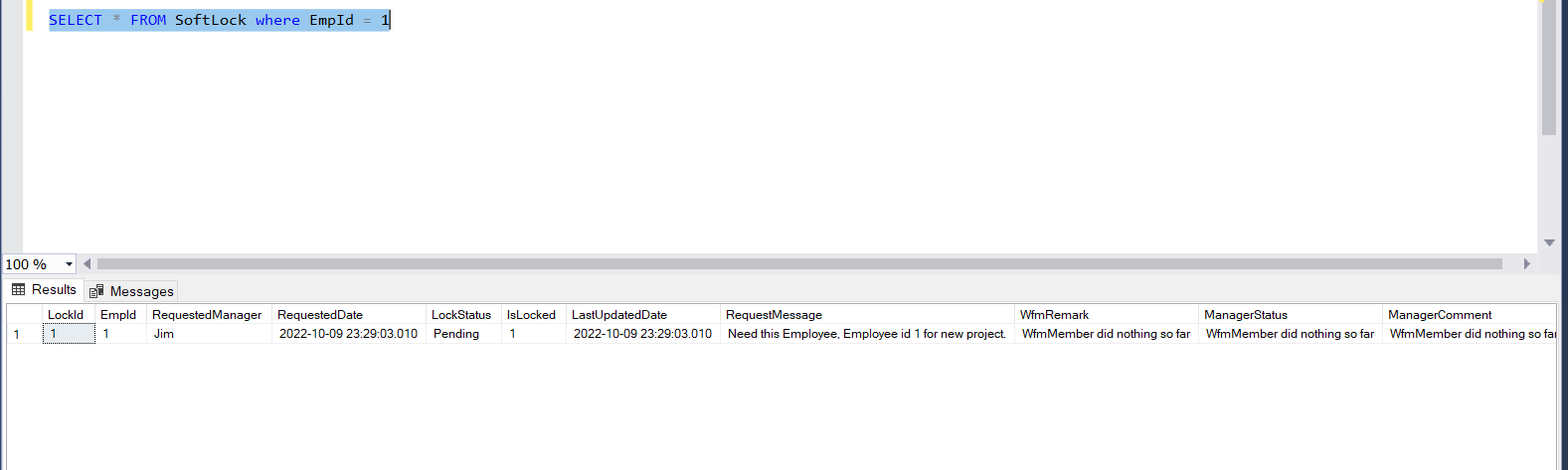
ManagerComment = @ManagerComment

WHERE

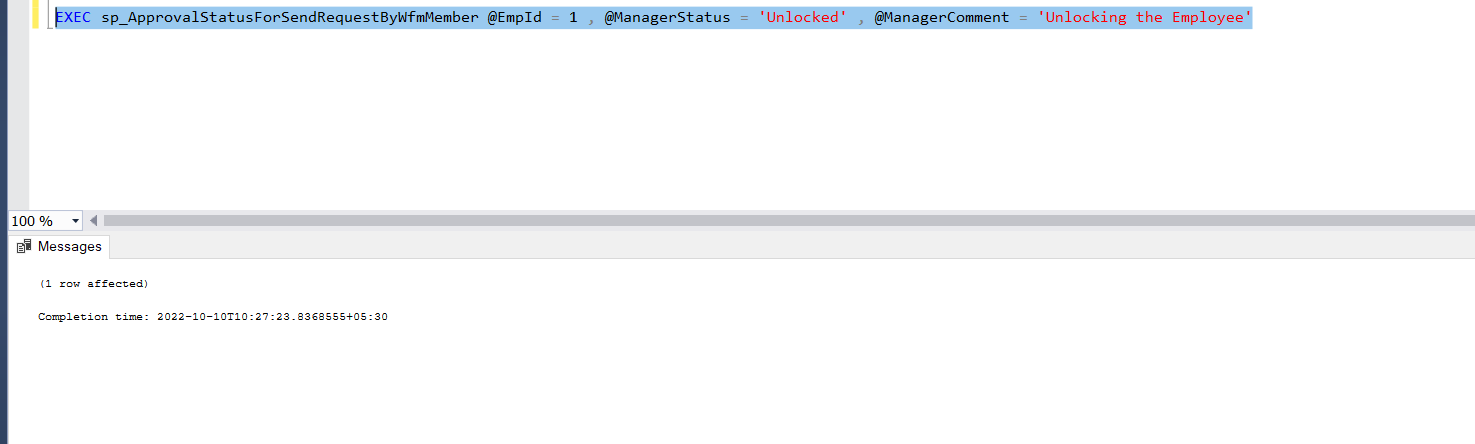
LockId = @LockId

END

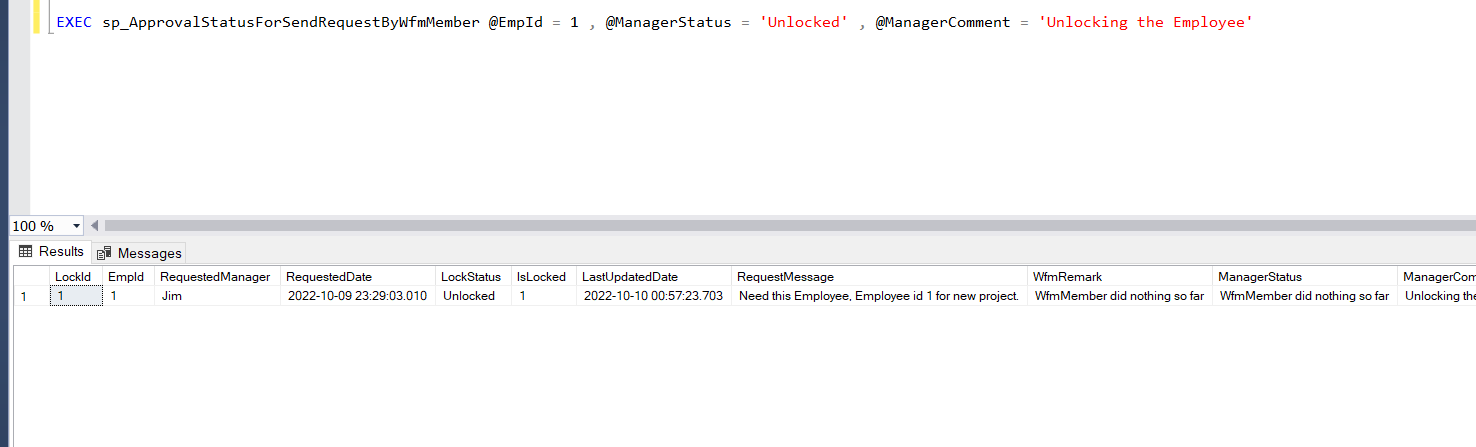
Before update



During Execution :



After Execution (Update Soft Lock) :



This is how my WFM Member Management system works .

Service I use for this Service.

using Wfm.Core;

using Wfm.Core.Abstractions;

using Wfm.Data;

using Wfm.Domain;

namespace Wfm.Service

{

public class ManagerService : IManagerService

{

private readonly LoginRepository loginRepository;

private readonly ManagerRepository managerRepository;

private readonly WfmMemberRepository wfmMemberRepository;

public ManagerService()

{

loginRepository = new LoginRepository();

managerRepository = new ManagerRepository();

wfmMemberRepository = new WfmMemberRepository();

}

public List<UserDetails> GetUserForLogin(LoginCriteria loginCriteria)

{

List<UserDetails> detail = new List<UserDetails>();

detail = loginRepository.GetUserForLogin(loginCriteria);

return detail;

}

public List<EmployeeDetails> GetDetailsForManagerPage()

{

List<EmployeeDetails> detail = new List<EmployeeDetails>();

detail = managerRepository.GetDetailsForManagerPage();

return detail;

}

public void LockStatusUpdateForManager(UpdateApprovalCriteria updateApprovalCriteria)

{

managerRepository.LockStatusUpdateForManager(updateApprovalCriteria);

}

public List<WfmMemberApprovalDetail> GetDetailsForWfmMemberPage(string UserName)

{

List<WfmMemberApprovalDetail> detail = new List<WfmMemberApprovalDetail>();

detail = wfmMemberRepository.GetDetailsForWfmMemberPage(UserName);

return detail;

}

public void LockStatusUpdateForWfmMember(UpdateApprovalCriteria updateApprovalCriteria)

{

wfmMemberRepository.LockStatusUpdateForWfmMember(updateApprovalCriteria);

}

}

}