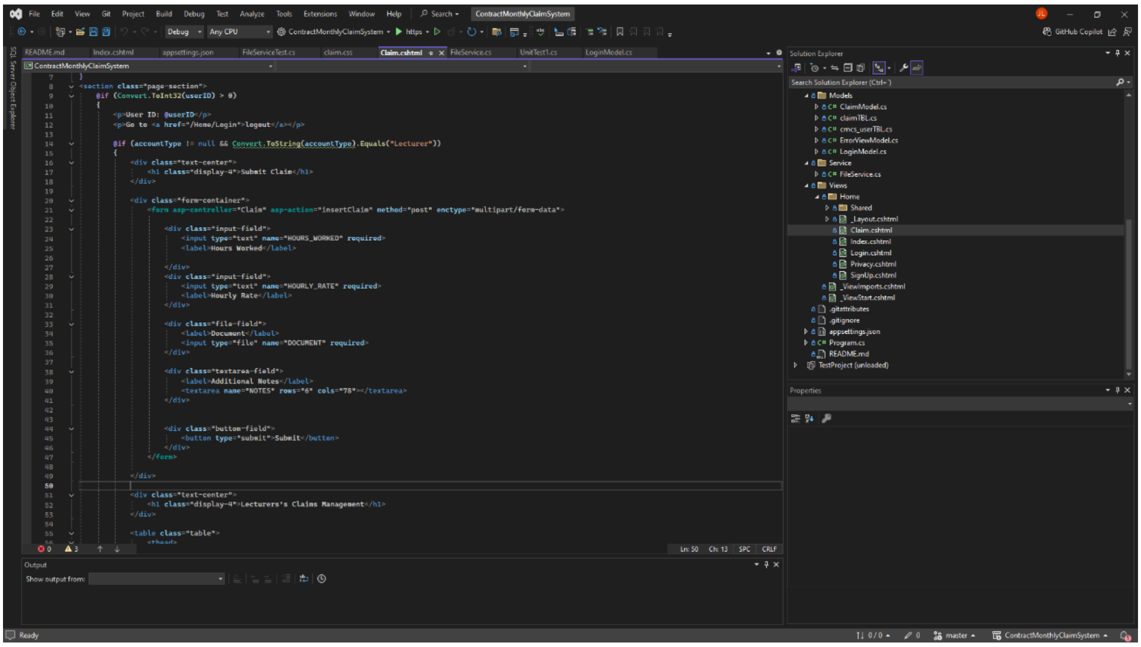
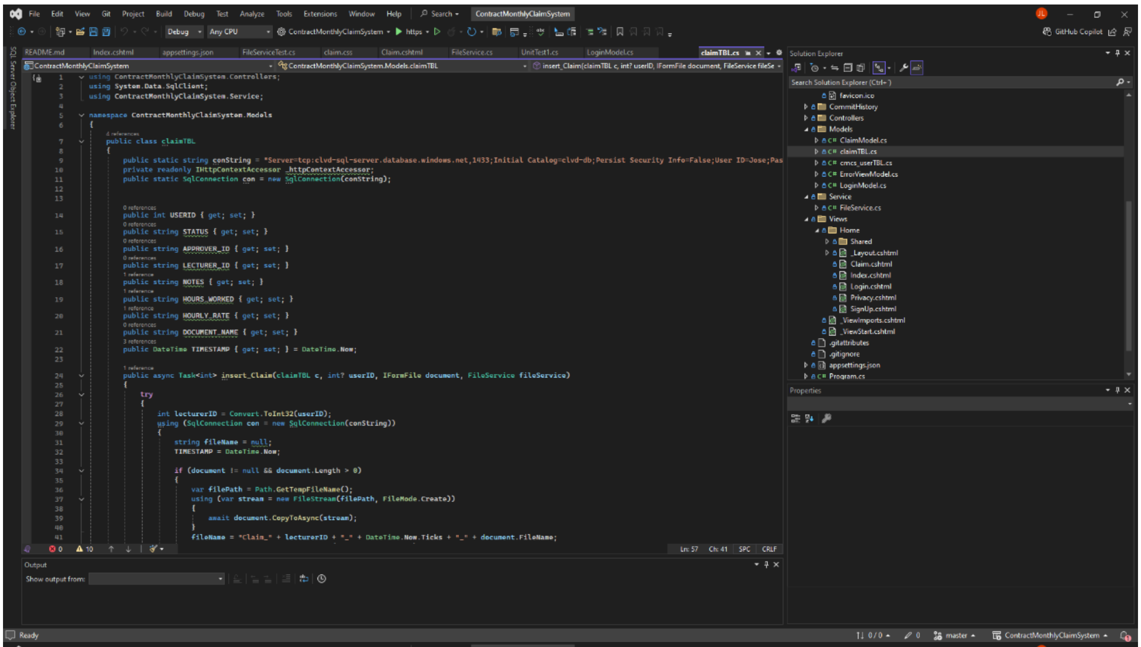
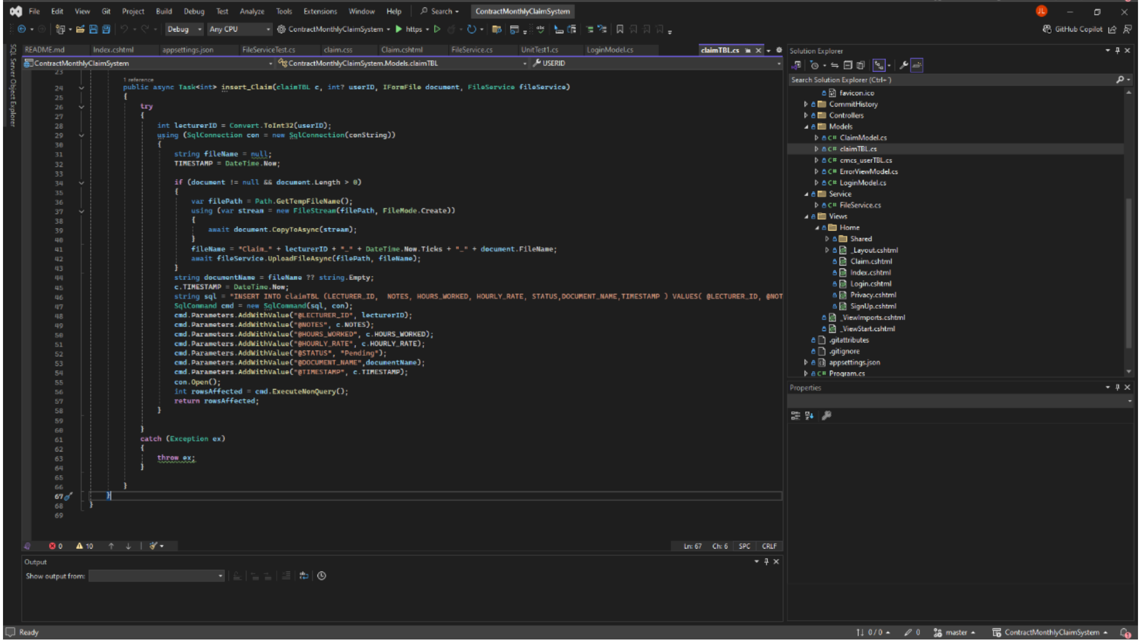
Lecturers Can submit their Claims at any time with a click of a button



```
    }
    <section class="page-section">
        @if (Convert.ToInt32(userID) > 0)
        {
            <p>User ID: @userID</p>
            <p>Go to <a href="/Home/Login">logout</a></p>

            @if (accountType != null && Convert.ToString(accountType).Equals("Lecturer"))
            {
                <div class="text-center">
                    <h1 class="display-4">Submit Claim</h1>
                </div>

                <div class="form-container">
                    <form asp-controller="Claim" asp-action="insertClaim" method="post" enctype="multipart/form-data">

                        <div class="input-field">
                            <input type="text" name="HOURS_WORKED" required>
                            <label>Hours Worked</label>

                        </div>
                        <div class="input-field">
                            <input type="text" name="HOURLY_RATE" required>
                            <label>Hourly Rate</label>
                        </div>

                        <div class="file-field">
                            <label>Document</label>
                            <input type="file" name="DOCUMENT" required>
                        </div>

                        <div class="textarea-field">
                            <label>Additional Notes</label>
                            <textarea name="NOTES" rows="6" cols="78"></textarea>
                        </div>


                        <div class="button-field">
                            <button type="submit">Submit</button>
                        </div>
                    </form>

                </div>

                <div class="text-center">
                    <h1 class="display-4">Lecturers's Claims Management</h1>
                </div>

                <table class="table">
                    <thead>
```

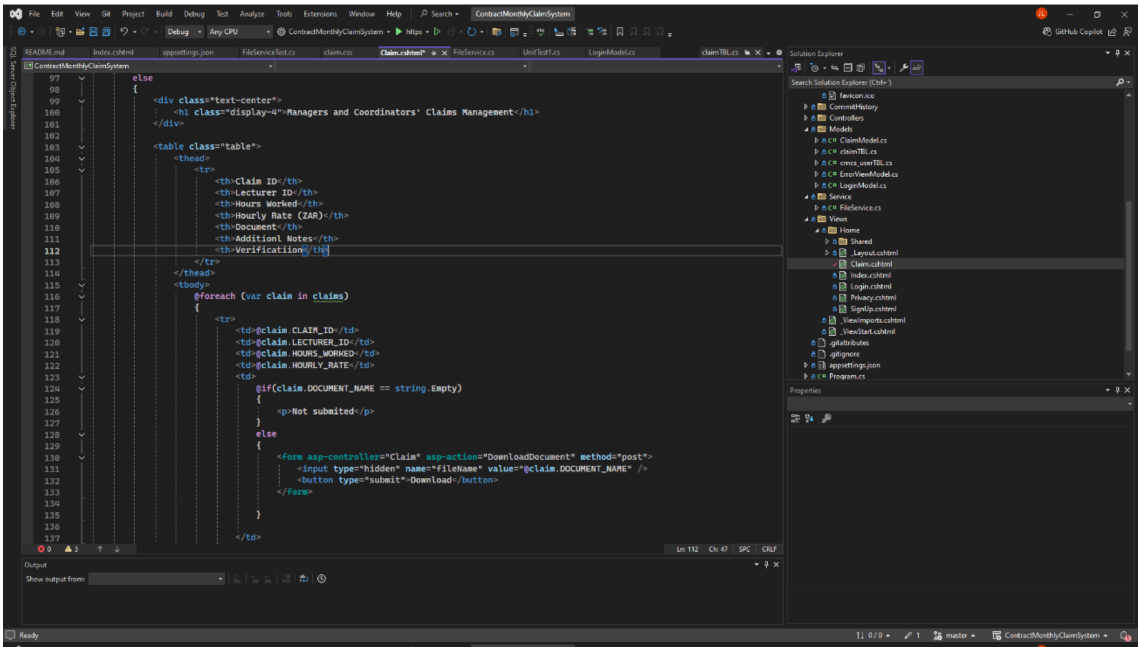


```
using ContractMonthlyClaimSystem.Controllers;
using System.Data.SqlClient;
using ContractMonthlyClaimSystem.Service;

namespace ContractMonthlyClaimSystem.Models
{
    public class claimTBL
    {
        public static string conString = "Server=tcp:clvd-sql-server.database.windows.net,1433;Initial Catalog=clvd-db;Persist Security Info=False;User ID=Jose;Pas
        private readonly IHttpContextAccessor _httpContextAccessor;
        public static SqlConnection con = new SqlConnection(conString);


        public int USERID { get; set; }
        public string STATUS { get; set; }
        public string APPROVER_ID { get; set; }
        public string LECTURER_ID { get; set; }
        public string NOTES { get; set; }
        public string HOURS_WORKED { get; set; }
        public string HOURLY_RATE { get; set; }
        public string DOCUMENT_NAME { get; set; }
        public DateTime TIMESTAMP { get; set; } = DateTime.Now;

        public async Task<int> insert_Claim(claimTBL c, int? userID, IFormFile document, FileService fileService)
        {
            try
            {
                int lecturerID = Convert.ToInt32(userID);
                using (SqlConnection con = new SqlConnection(conString))
                {
                    string fileName = null;
                    TIMESTAMP = DateTime.Now;

                    if (document != null && document.Length > 0)
                    {
                        var filePath = Path.GetTempFileName();
                        using (var stream = new FileStream(filePath, FileMode.Create))
                        {
                            await document.CopyToAsync(stream);
                        }
                        fileName = "Claim_" + lecturerID + "_" + DateTime.Now.Ticks + "_" + document.FileName;
```

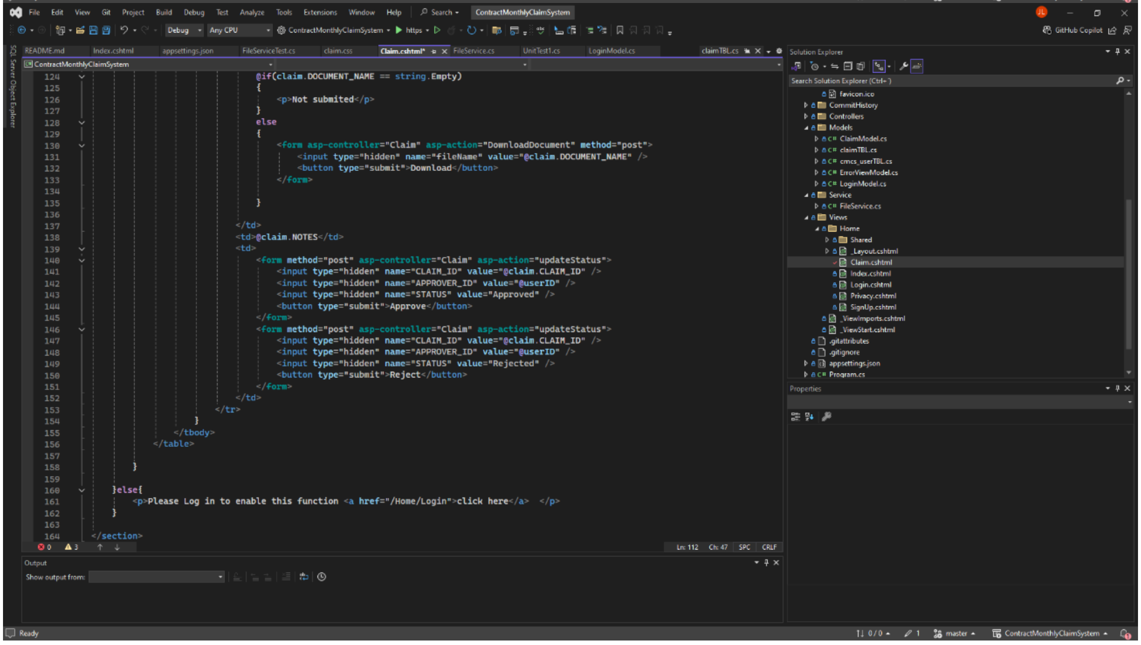


```
        public async Task<int> insert_Claim(claimTBL c, int? userID, IFormFile document, FileService fileService)
        {
            try
            {
                int lecturerID = Convert.ToInt32(userID);
                using (SqlConnection con = new SqlConnection(conString))
                {
                    string fileName = null;
                    TIMESTAMP = DateTime.Now;

                    if (document != null && document.Length > 0)
                    {
                        var filePath = Path.GetTempFileName();
                        using (var stream = new FileStream(filePath, FileMode.Create))
                        {
                            await document.CopyToAsync(stream);
                        }
                        fileName = "Claim_" + lecturerID + "_" + DateTime.Now.Ticks + "_" + document.FileName;
                        await fileService.UploadFileAsync(filePath, fileName);
                    }
                    string documentName = fileName ?? string.Empty;
                    c.TIMESTAMP = DateTime.Now;
                    string sql = "INSERT INTO claimTBL (LECTURER_ID,  NOTES, HOURS_WORKED, HOURLY_RATE, STATUS,DOCUMENT_NAME,TIMESTAMP ) VALUES( @LECTURER_ID, @NOT
                    SqlCommand cmd = new SqlCommand(sql, con);
                    cmd.Parameters.AddWithValue("@LECTURER_ID", lecturerID);
                    cmd.Parameters.AddWithValue("@NOTES", c.NOTES);
                    cmd.Parameters.AddWithValue("@HOURS_WORKED", c.HOURS_WORKED);
                    cmd.Parameters.AddWithValue("@HOURLY_RATE", c.HOURLY_RATE);
                    cmd.Parameters.AddWithValue("@STATUS", "Pending");
                    cmd.Parameters.AddWithValue("@DOCUMENT_NAME",documentName);
                    cmd.Parameters.AddWithValue("@TIMESTAMP", c.TIMESTAMP);
                    con.Open();
                    int rowsAffected = cmd.ExecuteNonQuery();
                    return rowsAffected;
                }

            }
            catch (Exception ex)
            {
                throw ex;
            }

        }
    }
}
```

Programme Coordinators and Academic Managers can easily verify and approve claims



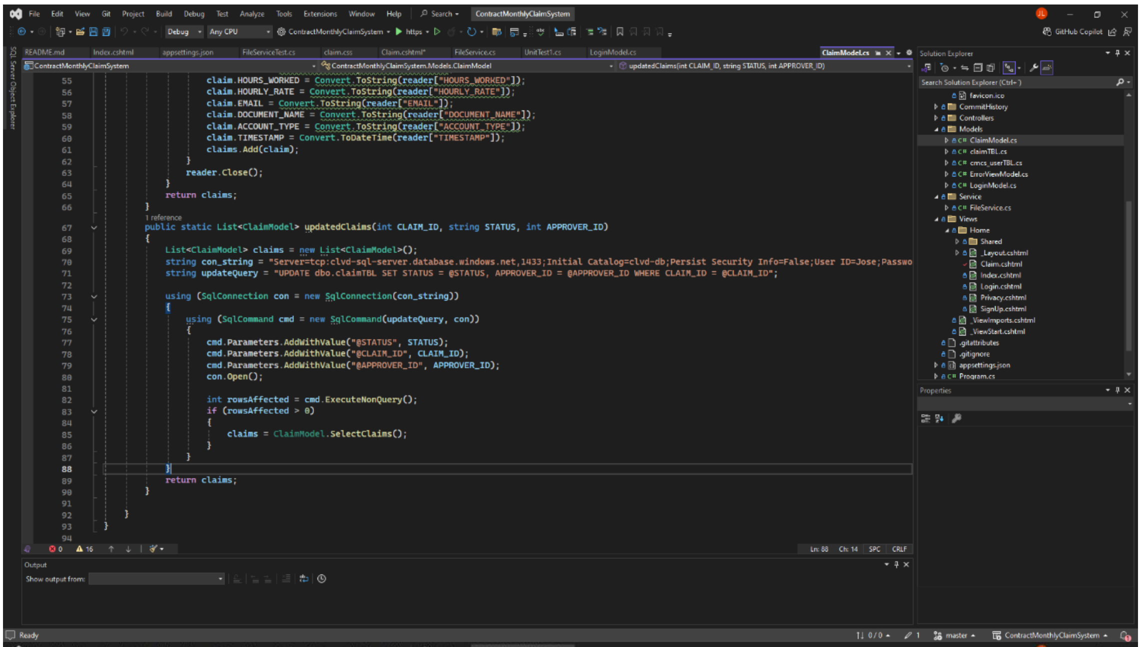
```
        else
        {
            <div class="text-center">
                <h1 class="display-4">Managers and Coordinators' Claims Management</h1>
            </div>

            <table class="table">
                <thead>
                    <tr>
                        <th>Claim ID</th>
                        <th>Lecturer ID</th>
                        <th>Hours Worked</th>
                        <th>Hourly Rate (ZAR)</th>
                        <th>Document</th>
                        <th>Additionl Notes</th>
                        <th>Verificatiion</th>
                    </tr>
                </thead>
                <tbody>
                    @foreach (var claim in claims)
                    {
                        <tr>
                            <td>@claim.CLAIM_ID</td>
                            <td>@claim.LECTURER_ID</td>
                            <td>@claim.HOURS_WORKED</td>
                            <td>@claim.HOURLY_RATE</td>
                            <td>
                                @if(claim.DOCUMENT_NAME == string.Empty)
                                {
                                    <p>Not submited</p>
                                }
                                else
                                {
                                    <form asp-controller="Claim" asp-action="DownloadDocument" method="post">
                                        <input type="hidden" name="fileName" value="@claim.DOCUMENT_NAME" />
                                        <button type="submit">Download</button>
                                    </form>

                                }

                            </td>
```



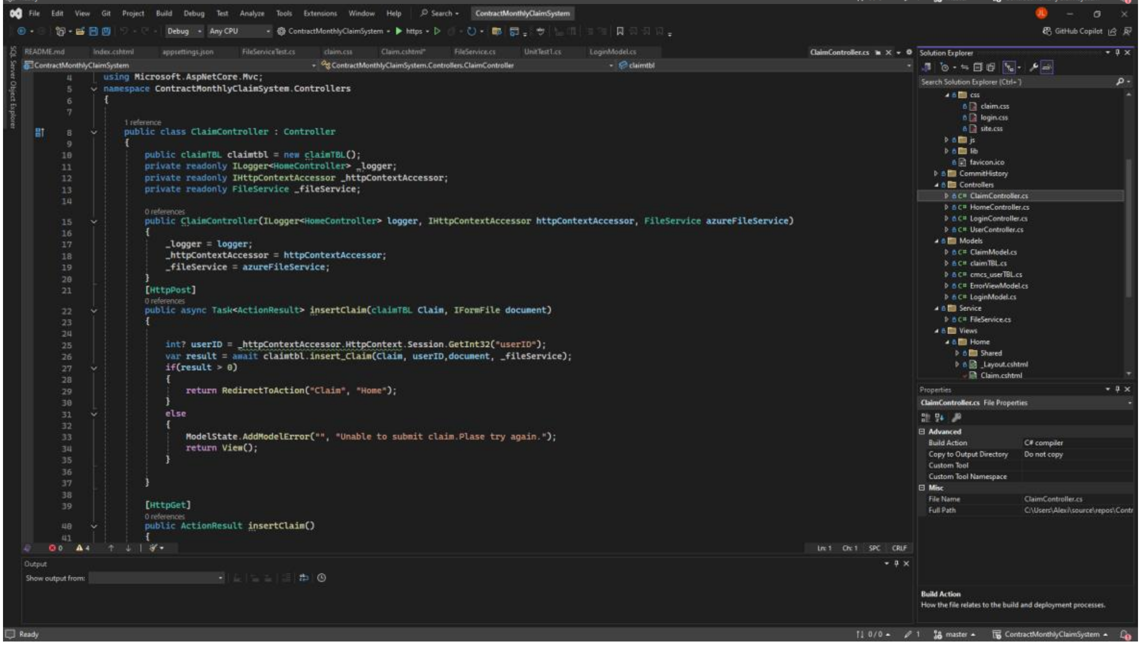
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search ContractMonthlyClaimSystem
Solution Explorer
Properties
Output
Show output from:
Ready
Ln: 112 Ch: 47 SPC CRLF
master

```
                                @if(claim.DOCUMENT_NAME == string.Empty)
                                {
                                    <p>Not submited</p>
                                }
                                else
                                {
                                    <form asp-controller="Claim" asp-action="DownloadDocument" method="post">
                                        <input type="hidden" name="fileName" value="@claim.DOCUMENT_NAME" />
                                        <button type="submit">Download</button>
                                    </form>

                                }

                            </td>
```

```
                            <td>@claim.NOTES</td>
                            <td>
                                <form method="post" asp-controller="Claim" asp-action="updateStatus">
                                    <input type="hidden" name="CLAIM_ID" value="@claim.CLAIM_ID" />
                                    <input type="hidden" name="APPROVER_ID" value="@userID" />
                                    <input type="hidden" name="STATUS" value="Approved" />
                                    <button type="submit">Approve</button>
                                </form>
                                <form method="post" asp-controller="Claim" asp-action="updateStatus">
                                    <input type="hidden" name="CLAIM_ID" value="@claim.CLAIM_ID" />
                                    <input type="hidden" name="APPROVER_ID" value="@userID" />
                                    <input type="hidden" name="STATUS" value="Rejected" />
                                    <button type="submit">Reject</button>
                                </form>
                            </td>
                        </tr>
                    }
                </tbody>
            </table>

        }

    }else{
        <p>Please Log in to enable this function <a href="/Home/Login">click here</a>  </p>
    }

</section>
```



```
                    claim.HOURS_WORKED = Convert.ToString(reader["HOURS_WORKED"]);
                    claim.HOURLY_RATE = Convert.ToString(reader["HOURLY_RATE"]);
                    claim.EMAIL = Convert.ToString(reader["EMAIL"]);
                    claim.DOCUMENT_NAME = Convert.ToString(reader["DOCUMENT_NAME"]);
                    claim.ACCOUNT_TYPE = Convert.ToString(reader["ACCOUNT_TYPE"]);
                    claim.TIMESTAMP = Convert.ToDateTime(reader["TIMESTAMP"]);
                    claims.Add(claim);
                }
                reader.Close();
            }
            return claims;
        }
        public static List<ClaimModel> updatedClaims(int CLAIM_ID, string STATUS, int APPROVER_ID)
        {
            List<ClaimModel> claims = new List<ClaimModel>();
            string con_string = "Server=tcp:clvd-sql-server.database.windows.net,1433;Initial Catalog=clvd-db;Persist Security Info=False;User ID=Jose;Passwo
            string updateQuery = "UPDATE dbo.claimTBL SET STATUS = @STATUS, APPROVER_ID = @APPROVER_ID WHERE CLAIM_ID = @CLAIM_ID";

            using (SqlConnection con = new SqlConnection(con_string))
            {
                using (SqlCommand cmd = new SqlCommand(updateQuery, con))
                {
                    cmd.Parameters.AddWithValue("@STATUS", STATUS);
                    cmd.Parameters.AddWithValue("@CLAIM_ID", CLAIM_ID);
                    cmd.Parameters.AddWithValue("@APPROVER_ID", APPROVER_ID);
                    con.Open();

                    int rowsAffected = cmd.ExecuteNonQuery();
                    if (rowsAffected > 0)
                    {
                        claims = ClaimModel.SelectClaims();
                    }
                }
            }
            return claims;
        }

    }
}
```



```
using Microsoft.AspNetCore.Mvc;
namespace ContractMonthlyClaimSystem.Controllers
{

    public class ClaimController : Controller
    {
        public claimTBL claimtbl = new claimTBL();
        private readonly ILogger<HomeController> _logger;
        private readonly IHttpContextAccessor _httpContextAccessor;
        private readonly FileService _fileService;

        public ClaimController(ILogger<HomeController> logger, IHttpContextAccessor httpContextAccessor, FileService azureFileService)
        {
            _logger = logger;
            _httpContextAccessor = httpContextAccessor;
            _fileService = azureFileService;
        }
        [HttpPost]
        public async Task<ActionResult> insertClaim(claimTBL Claim, IFormFile document)
        {

            int? userID = _httpContextAccessor.HttpContext.Session.GetInt32("userID");
            var result = await claimtbl.insert_Claim(Claim, userID,document, _fileService);
            if(result > 0)
            {
                return RedirectToAction("Claim", "Home");
            }
            else
            {
                ModelState.AddModelError("", "Unable to submit claim.Plase try again.");
                return View();
            }

        }

        [HttpGet]
        public ActionResult insertClaim()
        {
```

ClaimController.cs File Properties

Advanced

Build Action: C# compiler

Copy to Output Directory: Do not copy

Custom Tool

Custom Tool Namespace

Misc

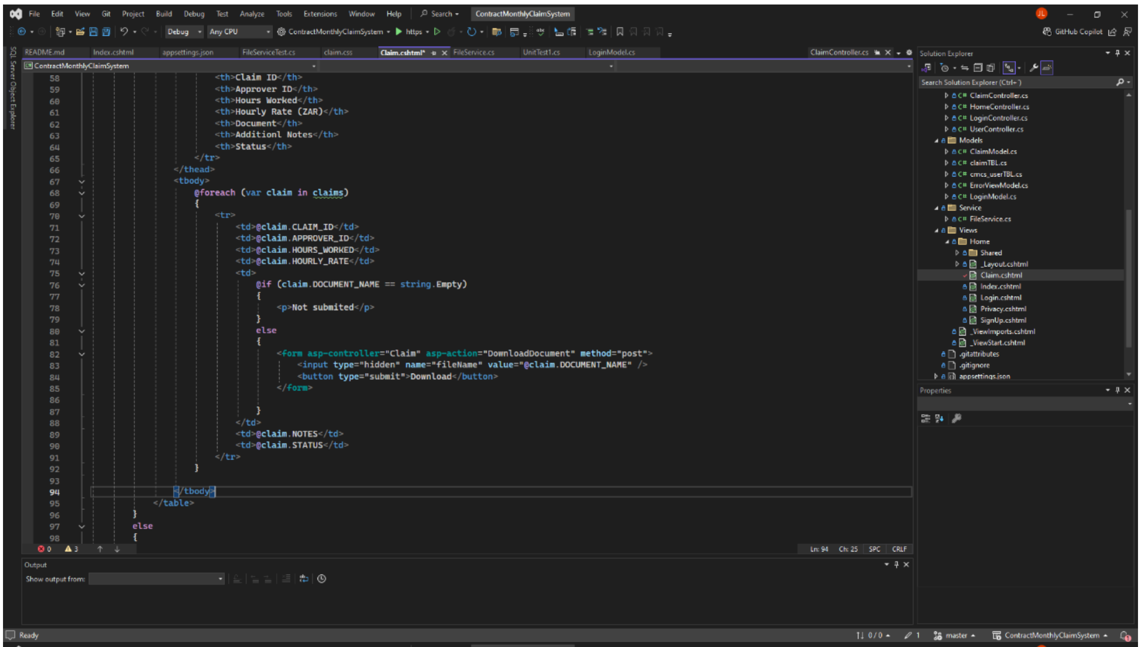
File Name: ClaimController.cs

Full Path: C:\Users\Alex\\source\repos\Contr

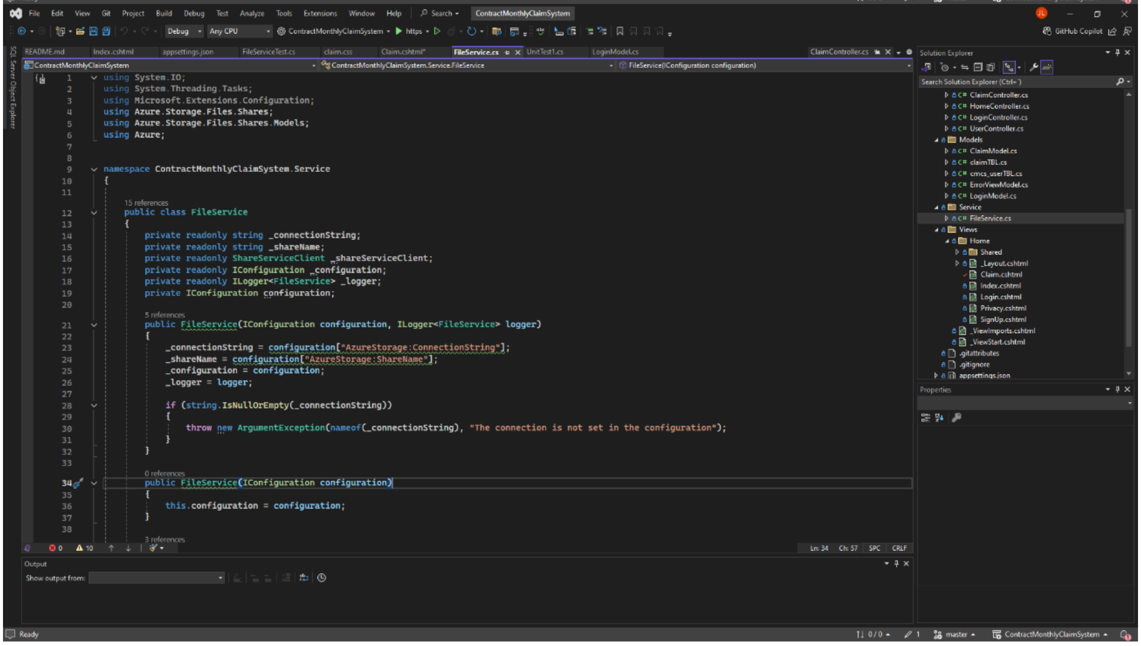
Build Action

How the file relates to the build and deployment processes.

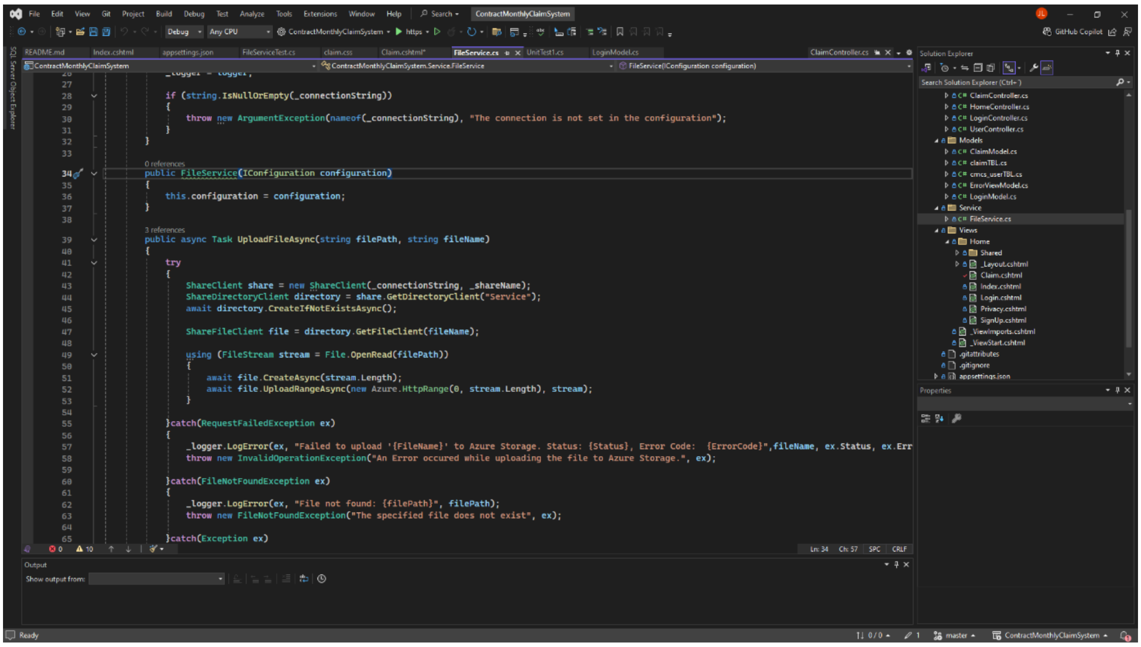
Lecturers can upload supporting Documents for their Claim



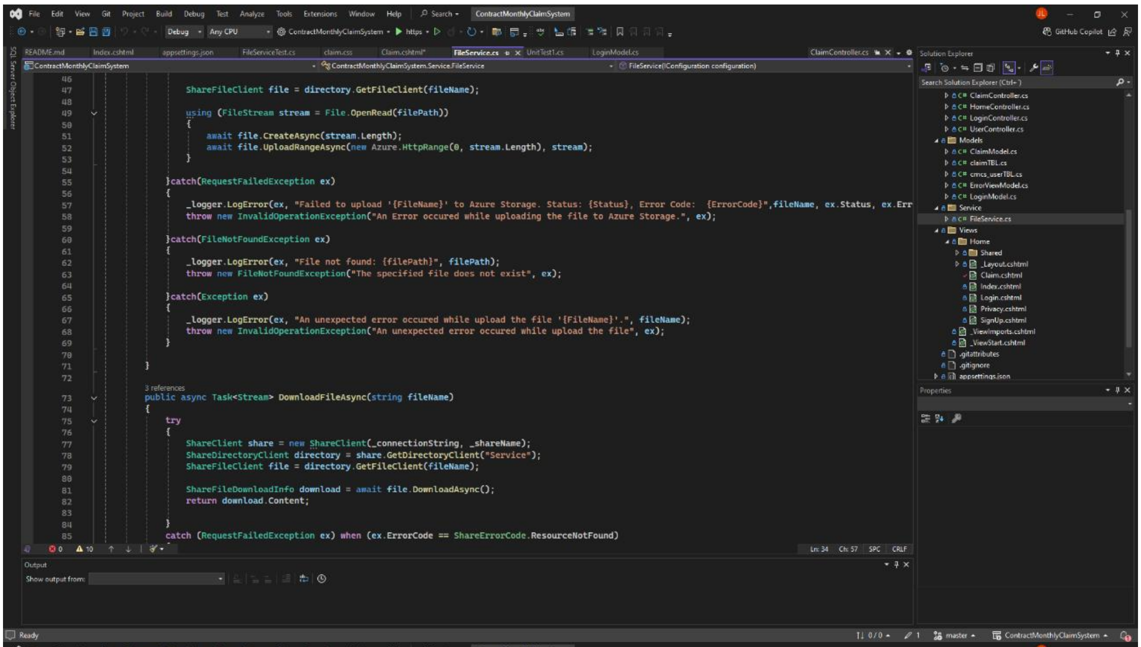
```
                    <th>Claim ID</th>
                    <th>Approver ID</th>
                    <th>Hours Worked</th>
                    <th>Hourly Rate (ZAR)</th>
                    <th>Document</th>
                    <th>Additionl Notes</th>
                    <th>Status</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var claim in claims)
                {
                    <tr>
                        <td>@claim.CLAIM_ID</td>
                        <td>@claim.APPROVER_ID</td>
                        <td>@claim.HOURS_WORKED</td>
                        <td>@claim.HOURLY_RATE</td>
                        <td>
                            @if (claim.DOCUMENT_NAME == string.Empty)
                            {
                                <p>Not submited</p>
                            }
                            else
                            {
                                <form asp-controller="Claim" asp-action="DownloadDocument" method="post">
                                    <input type="hidden" name="fileName" value="@claim.DOCUMENT_NAME" />
                                    <button type="submit">Download</button>
                                </form>

                            }
                        </td>
                        <td>@claim.NOTES</td>
                        <td>@claim.STATUS</td>
                    </tr>
                }

            </tbody>
        </table>
    }
    else
    {
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search ContractMonthlyClaimSystem
Debug Any CPU ContractMonthlyClaimSystem https
GitHub Copilot
README.md Index.cshtml appsettings.json FileServiceTest.cs claim.css Claim.cshtml* FileService.cs UnitTest1.cs LoginModel.cs ClaimController.cs
ContractMonthlyClaimSystem
ContractMonthlyClaimSystem.Service.FileService
FileService(IConfiguration configuration)
Solution Explorer
Search Solution Explorer (Ctrl+')
ClaimController.cs
HomeController.cs
LoginController.cs
UserController.cs
Models
ClaimModel.cs
claimTBL.cs
cmcs_userTBL.cs
ErrorViewModel.cs
LoginModel.cs
Service
FileService.cs
Views
Home
Shared
_Layout.cshtml
Claim.cshtml
Index.cshtml
Login.cshtml
Privacy.cshtml
SignUp.cshtml
_ViewImports.cshtml
_ViewStart.cshtml
.gitattributes
.gitignore
appsettings.json
Properties
Output
Show output from:
Ready
Ln: 34 Ch: 57 SPC CRLF
0 10
0 / 0 1 master ContractMonthlyClaimSystem

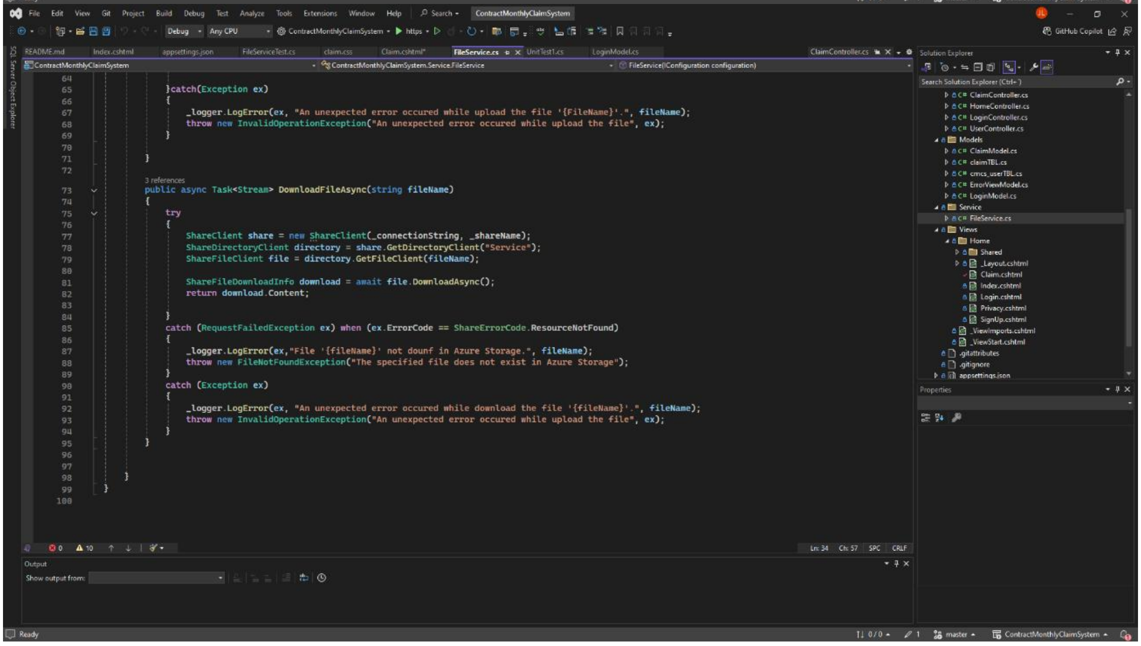
```
using System.IO;
using System.Threading.Tasks;
using Microsoft.Extensions.Configuration;
using Azure.Storage.Files.Shares;
using Azure.Storage.Files.Shares.Models;
using Azure;


namespace ContractMonthlyClaimSystem.Service
{

    15 references
    public class FileService
    {
        private readonly string _connectionString;
        private readonly string _shareName;
        private readonly ShareServiceClient _shareServiceClient;
        private readonly IConfiguration _configuration;
        private readonly ILogger<FileService> _logger;
        private IConfiguration configuration;

        5 references
        public FileService(IConfiguration configuration, ILogger<FileService> logger)
        {
            _connectionString = configuration["AzureStorage:ConnectionString"];
            _shareName = configuration["AzureStorage:ShareName"];
            _configuration = configuration;
            _logger = logger;

            if (string.IsNullOrEmpty(_connectionString))
            {
                throw new ArgumentException(nameof(_connectionString), "The connection is not set in the configuration");
            }
        }

        0 references
        public FileService(IConfiguration configuration)
        {
            this.configuration = configuration;
        }

        3 references
```



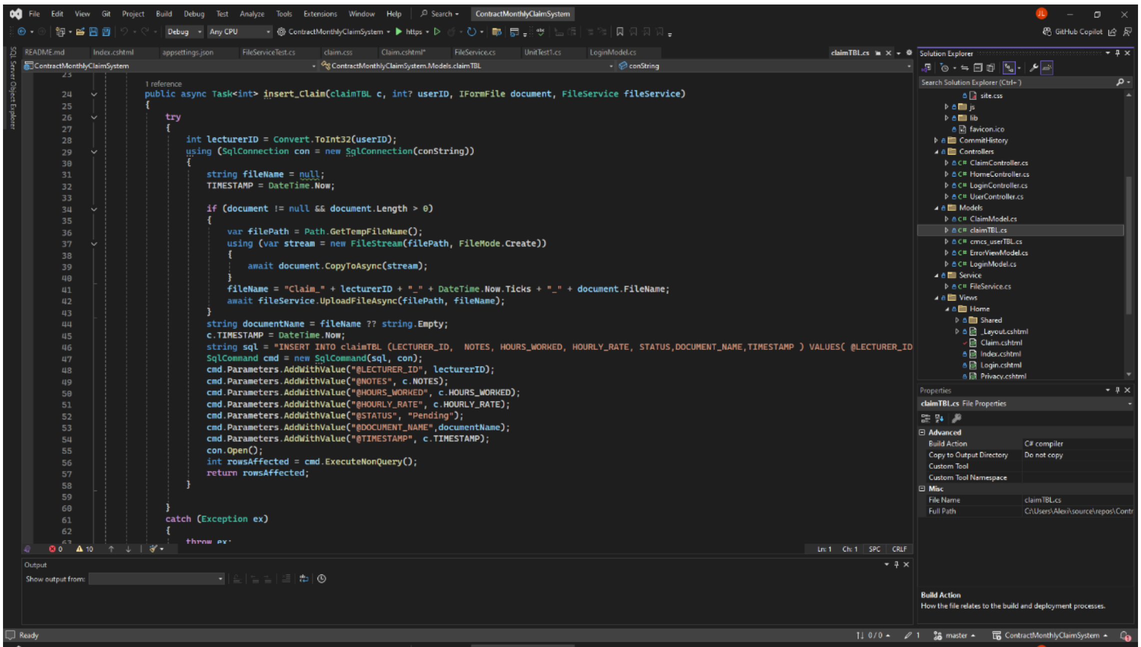
```
            _logger = logger;

            if (string.IsNullOrEmpty(_connectionString))
            {
                throw new ArgumentException(nameof(_connectionString), "The connection is not set in the configuration");
            }
        }

        public FileService(IConfiguration configuration)
        {
            this.configuration = configuration;
        }

        public async Task UploadFileAsync(string filePath, string fileName)
        {
            try
            {
                ShareClient share = new ShareClient(_connectionString, _shareName);
                ShareDirectoryClient directory = share.GetDirectoryClient("Service");
                await directory.CreateIfNotExistsAsync();

                ShareFileClient file = directory.GetFileClient(fileName);

                using (FileStream stream = File.OpenRead(filePath))
                {
                    await file.CreateAsync(stream.Length);
                    await file.UploadRangeAsync(new Azure.HttpRange(0, stream.Length), stream);
                }

            }catch(RequestFailedException ex)
            {
                _logger.LogError(ex, "Failed to upload '{FileName}' to Azure Storage. Status: {Status}, Error Code:  {ErrorCode}",fileName, ex.Status, ex.Err
                throw new InvalidOperationException("An Error occured while uploading the file to Azure Storage.", ex);

            }catch(FileNotFoundException ex)
            {
                _logger.LogError(ex, "File not found: {filePath}", filePath);
                throw new FileNotFoundException("The specified file does not exist", ex);

            }catch(Exception ex)
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search ContractMonthlyClaimSystem
Debug Any CPU ContractMonthlyClaimSystem https
GitHub Copilot
README.md Index.cshtml appsettings.json FileServiceTest.cs claim.css Claim.cshtml* FileService.cs UnitTest1.cs LoginModel.cs ClaimController.cs
ContractMonthlyClaimSystem
ContractMonthlyClaimSystem.Service.FileService
FileService(IConfiguration configuration)
Solution Explorer
Search Solution Explorer (Ctrl+;)
ClaimController.cs
HomeController.cs
LoginController.cs
UserController.cs
Models
ClaimModel.cs
claimTBL.cs
cmcs_userTBL.cs
ErrorViewModel.cs
LoginModel.cs
Service
FileService.cs
Views
Home
Shared
_Layout.cshtml
Claim.cshtml
Index.cshtml
Login.cshtml
Privacy.cshtml
SignUp.cshtml
_ViewImports.cshtml
_ViewStart.cshtml
.gitattributes
.gitignore
appsettings.json
Properties
Ln: 34 Ch: 57 SPC CRLF
Output
Show output from:
Ready
master
ContractMonthlyClaimSystem

```
                ShareFileClient file = directory.GetFileClient(fileName);

                using (FileStream stream = File.OpenRead(filePath))
                {
                    await file.CreateAsync(stream.Length);
                    await file.UploadRangeAsync(new Azure.HttpRange(0, stream.Length), stream);
                }

            }catch(RequestFailedException ex)
            {
                _logger.LogError(ex, "Failed to upload '{FileName}' to Azure Storage. Status: {Status}, Error Code:  {ErrorCode}",fileName, ex.Status, ex.Err
                throw new InvalidOperationException("An Error occured while uploading the file to Azure Storage.", ex);

            }catch(FileNotFoundException ex)
            {
                _logger.LogError(ex, "File not found: {filePath}", filePath);
                throw new FileNotFoundException("The specified file does not exist", ex);

            }catch(Exception ex)
            {
                _logger.LogError(ex, "An unexpected error occured while upload the file '{FileName}'.", fileName);
                throw new InvalidOperationException("An unexpected error occured while upload the file", ex);
            }

        }

        3 references
        public async Task<Stream> DownloadFileAsync(string fileName)
        {
            try
            {
                ShareClient share = new ShareClient(_connectionString, _shareName);
                ShareDirectoryClient directory = share.GetDirectoryClient("Service");
                ShareFileClient file = directory.GetFileClient(fileName);

                ShareFileDownloadInfo download = await file.DownloadAsync();
                return download.Content;

            }
            catch (RequestFailedException ex) when (ex.ErrorCode == ShareErrorCode.ResourceNotFound)
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search ContractMonthlyClaimSystem
Debug Any CPU ContractMonthlyClaimSystem https
GitHub Copilot
README.md Index.cshtml appsettings.json FileServiceTest.cs claim.css Claim.cshtml* FileService.cs UnitTest1.cs LoginModel.cs ClaimController.cs
ContractMonthlyClaimSystem
ContractMonthlyClaimSystem.Service.FileService
FileService(IConfiguration configuration)
Solution Explorer
Search Solution Explorer (Ctrl+;)
ClaimController.cs
HomeController.cs
LoginController.cs
UserController.cs
Models
ClaimModel.cs
claimTBL.cs
cmcs_userTBL.cs
ErrorViewModel.cs
LoginModel.cs
Service
FileService.cs
Views
Home
Shared
_Layout.cshtml
Claim.cshtml
Index.cshtml
Login.cshtml
Privacy.cshtml
SignUp.cshtml
_ViewImports.cshtml
_ViewStart.cshtml
.gitattributes
.gitignore
appsettings.json
Properties
Ln: 34 Ch: 57 SPC CRLF
Output
Show output from:
Ready
master
ContractMonthlyClaimSystem

```
            }catch(Exception ex)
            {
                _logger.LogError(ex, "An unexpected error occured while upload the file '{FileName}'.", fileName);
                throw new InvalidOperationException("An unexpected error occured while upload the file", ex);
            }

        }

        3 references
        public async Task<Stream> DownloadFileAsync(string fileName)
        {
            try
            {
                ShareClient share = new ShareClient(_connectionString, _shareName);
                ShareDirectoryClient directory = share.GetDirectoryClient("Service");
                ShareFileClient file = directory.GetFileClient(fileName);

                ShareFileDownloadInfo download = await file.DownloadAsync();
                return download.Content;

            }
            catch (RequestFailedException ex) when (ex.ErrorCode == ShareErrorCode.ResourceNotFound)
            {
                _logger.LogError(ex,"File '{fileName}' not dounf in Azure Storage.", fileName);
                throw new FileNotFoundException("The specified file does not exist in Azure Storage");
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "An unexpected error occured while download the file '{fileName}'.", fileName);
                throw new InvalidOperationException("An unexpected error occured while upload the file", ex);
            }
        }


    }
}
```



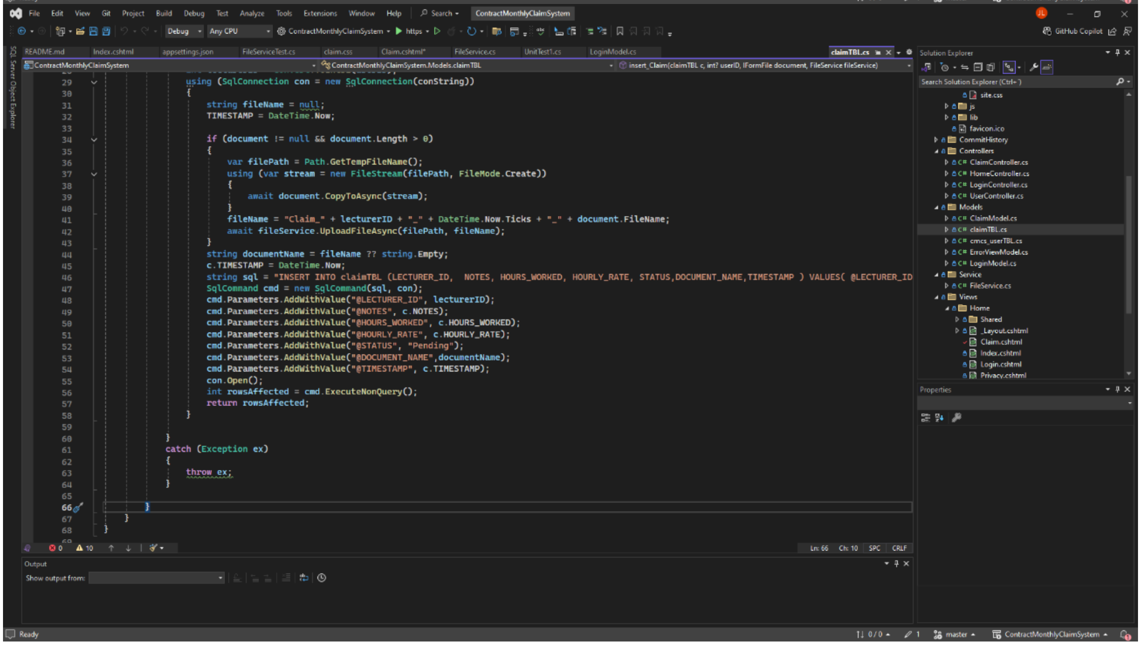
```
1 reference
public async Task<int> insert_Claim(claimTBL c, int? userID, IFormFile document, FileService fileService)
{
    try
    {
        int lecturerID = Convert.ToInt32(userID);
        using (SqlConnection con = new SqlConnection(conString))
        {
            string fileName = null;
            TIMESTAMP = DateTime.Now;

            if (document != null && document.Length > 0)
            {
                var filePath = Path.GetTempFileName();
                using (var stream = new FileStream(filePath, FileMode.Create))
                {
                    await document.CopyToAsync(stream);
                }
                fileName = "Claim_" + lecturerID + "_" + DateTime.Now.Ticks + "_" + document.FileName;
                await fileService.UploadFileAsync(filePath, fileName);
            }
            string documentName = fileName ?? string.Empty;
            c.TIMESTAMP = DateTime.Now;
            string sql = "INSERT INTO claimTBL (LECTURER_ID,  NOTES, HOURS_WORKED, HOURLY_RATE, STATUS,DOCUMENT_NAME,TIMESTAMP ) VALUES( @LECTURER_ID
            SqlCommand cmd = new SqlCommand(sql, con);
            cmd.Parameters.AddWithValue("@LECTURER_ID", lecturerID);
            cmd.Parameters.AddWithValue("@NOTES", c.NOTES);
            cmd.Parameters.AddWithValue("@HOURS_WORKED", c.HOURS_WORKED);
            cmd.Parameters.AddWithValue("@HOURLY_RATE", c.HOURLY_RATE);
            cmd.Parameters.AddWithValue("@STATUS", "Pending");
            cmd.Parameters.AddWithValue("@DOCUMENT_NAME",documentName);
            cmd.Parameters.AddWithValue("@TIMESTAMP", c.TIMESTAMP);
            con.Open();
            int rowsAffected = cmd.ExecuteNonQuery();
            return rowsAffected;
        }

    }
    catch (Exception ex)
    {
        throw ex;
```

claimTBL.cs File Properties

Advanced
Build Action: C# compiler
Copy to Output Directory: Do not copy
Custom Tool:
Custom Tool Namespace:

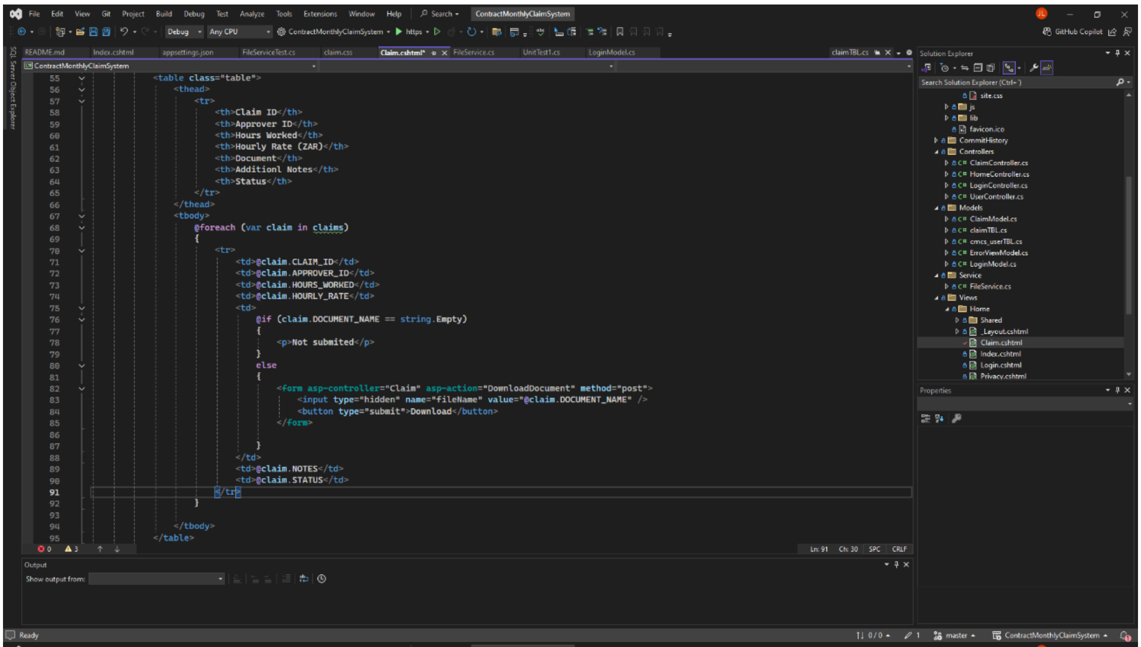
Misc
File Name: claimTBL.cs
Full Path: C:\Users\Alexi\source\repos\Contr

Build Action
How the file relates to the build and deployment processes.

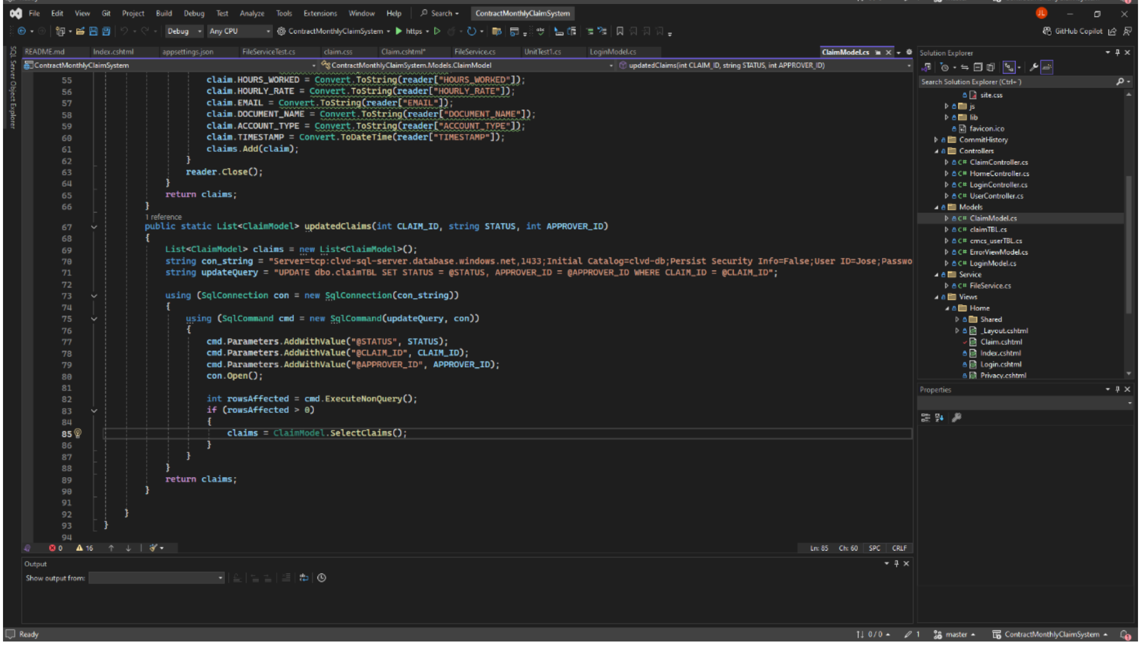


```
        using (SqlConnection con = new SqlConnection(conString))
        {
            string fileName = null;
            TIMESTAMP = DateTime.Now;

            if (document != null && document.Length > 0)
            {
                var filePath = Path.GetTempFileName();
                using (var stream = new FileStream(filePath, FileMode.Create))
                {
                    await document.CopyToAsync(stream);
                }
                fileName = "Claim_" + lecturerID + "_" + DateTime.Now.Ticks + "_" + document.FileName;
                await fileService.UploadFileAsync(filePath, fileName);
            }
            string documentName = fileName ?? string.Empty;
            c.TIMESTAMP = DateTime.Now;
            string sql = "INSERT INTO claimTBL (LECTURER_ID,  NOTES, HOURS_WORKED, HOURLY_RATE, STATUS,DOCUMENT_NAME,TIMESTAMP ) VALUES( @LECTURER_ID
            SqlCommand cmd = new SqlCommand(sql, con);
            cmd.Parameters.AddWithValue("@LECTURER_ID", lecturerID);
            cmd.Parameters.AddWithValue("@NOTES", c.NOTES);
            cmd.Parameters.AddWithValue("@HOURS_WORKED", c.HOURS_WORKED);
            cmd.Parameters.AddWithValue("@HOURLY_RATE", c.HOURLY_RATE);
            cmd.Parameters.AddWithValue("@STATUS", "Pending");
            cmd.Parameters.AddWithValue("@DOCUMENT_NAME",documentName);
            cmd.Parameters.AddWithValue("@TIMESTAMP", c.TIMESTAMP);
            con.Open();
            int rowsAffected = cmd.ExecuteNonQuery();
            return rowsAffected;
        }

    }
    catch (Exception ex)
    {
        throw ex;
    }

}
    }
}
```

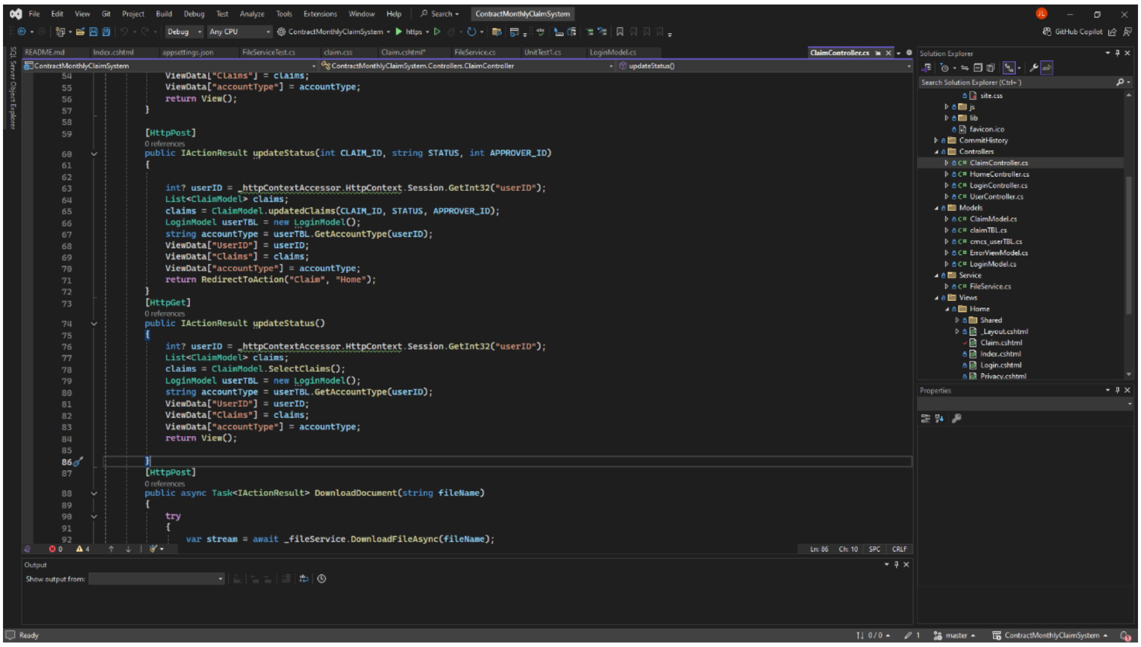
Claim status can be tracked until settled



```
            <table class="table">
                <thead>
                    <tr>
                        <th>Claim ID</th>
                        <th>Approver ID</th>
                        <th>Hours Worked</th>
                        <th>Hourly Rate (ZAR)</th>
                        <th>Document</th>
                        <th>Additionl Notes</th>
                        <th>Status</th>
                    </tr>
                </thead>
                <tbody>
                    @foreach (var claim in claims)
                    {
                        <tr>
                            <td>@claim.CLAIM_ID</td>
                            <td>@claim.APPROVER_ID</td>
                            <td>@claim.HOURS_WORKED</td>
                            <td>@claim.HOURLY_RATE</td>
                            <td>
                                @if (claim.DOCUMENT_NAME == string.Empty)
                                {
                                    <p>Not submited</p>
                                }
                                else
                                {
                                    <form asp-controller="Claim" asp-action="DownloadDocument" method="post">
                                        <input type="hidden" name="fileName" value="@claim.DOCUMENT_NAME" />
                                        <button type="submit">Download</button>
                                    </form>

                                }
                            </td>
                            <td>@claim.NOTES</td>
                            <td>@claim.STATUS</td>
                        </tr>
                    }

                </tbody>
            </table>
```



```
                    claim.HOURS_WORKED = Convert.ToString(reader["HOURS_WORKED"]);
                    claim.HOURLY_RATE = Convert.ToString(reader["HOURLY_RATE"]);
                    claim.EMAIL = Convert.ToString(reader["EMAIL"]);
                    claim.DOCUMENT_NAME = Convert.ToString(reader["DOCUMENT_NAME"]);
                    claim.ACCOUNT_TYPE = Convert.ToString(reader["ACCOUNT_TYPE"]);
                    claim.TIMESTAMP = Convert.ToDateTime(reader["TIMESTAMP"]);
                    claims.Add(claim);
                }
                reader.Close();
            }
            return claims;
        }
        1 reference
        public static List<ClaimModel> updatedClaims(int CLAIM_ID, string STATUS, int APPROVER_ID)
        {
            List<ClaimModel> claims = new List<ClaimModel>();
            string con_string = "Server=tcp:clvd-sql-server.database.windows.net,1433;Initial Catalog=clvd-db;Persist Security Info=False;User ID=Jose;Passwo
            string updateQuery = "UPDATE dbo.claimTBL SET STATUS = @STATUS, APPROVER_ID = @APPROVER_ID WHERE CLAIM_ID = @CLAIM_ID";

            using (SqlConnection con = new SqlConnection(con_string))
            {
                using (SqlCommand cmd = new SqlCommand(updateQuery, con))
                {
                    cmd.Parameters.AddWithValue("@STATUS", STATUS);
                    cmd.Parameters.AddWithValue("@CLAIM_ID", CLAIM_ID);
                    cmd.Parameters.AddWithValue("@APPROVER_ID", APPROVER_ID);
                    con.Open();

                    int rowsAffected = cmd.ExecuteNonQuery();
                    if (rowsAffected > 0)
                    {
                        claims = ClaimModel.SelectClaims();
                    }
                }
            }
            return claims;
        }

    }
}
```



```
                ViewData["Claims"] = claims;
                ViewData["accountType"] = accountType;
                return View();
            }

            [HttpPost]
            public IActionResult updateStatus(int CLAIM_ID, string STATUS, int APPROVER_ID)
            {

                int? userID = _httpContextAccessor.HttpContext.Session.GetInt32("userID");
                List<ClaimModel> claims;
                claims = ClaimModel.updatedClaims(CLAIM_ID, STATUS, APPROVER_ID);
                LoginModel userTBL = new LoginModel();
                string accountType = userTBL.GetAccountType(userID);
                ViewData["UserID"] = userID;
                ViewData["Claims"] = claims;
                ViewData["accountType"] = accountType;
                return RedirectToAction("Claim", "Home");
            }
            [HttpGet]
            public IActionResult updateStatus()
            {
                int? userID = _httpContextAccessor.HttpContext.Session.GetInt32("userID");
                List<ClaimModel> claims;
                claims = ClaimModel.SelectClaims();
                LoginModel userTBL = new LoginModel();
                string accountType = userTBL.GetAccountType(userID);
                ViewData["UserID"] = userID;
                ViewData["Claims"] = claims;
                ViewData["accountType"] = accountType;
                return View();

            }
            [HttpPost]
            public async Task<IActionResult> DownloadDocument(string fileName)
            {
                try
                {
                    var stream = await _fileService.DownloadFileAsync(fileName);
```