MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE
(Autonomous)

NEW PANVEL

PROJECT REPORT ON

# "Online Notes Sharing Portal"

IN PARTIAL FULFILLMENT OF

BACHELOR OF COMPUTER APPLICTION

SEMESTER V 2025-26

PROJECT GUIDE

**Mr. Omkar Sherkhane**

SUBMITTED BY: Paras Chaudhari

ROLL NO: 8112

Mahatma Education Society's

**PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE (Autonomous), New Panvel**

**Re-accredited "A" Grade by NAAC (3rd Cycle)**

**Project Completion Certificate**

**THIS IS TO CERTIFY THAT**

**Paras Vinod Chaudhari**

of **TYBCA** has completed the project titled **"Online Notes Sharing Portal"**

Under our guidance and supervision during the academic year 2025-26 in the department of Computer Application

Course Coordinator | Head of Department | Project Guide

Pillai College of Arts, Commerce & Science (Autonomous) New Panvel

# Introduction

The **Online Notes Sharing Portal** is a web application that enables students and professionals to upload, manage, and share their notes in digital format (mainly PDFs). The core purpose of this project is to build a collaborative platform where users can contribute their notes and at the same time access notes uploaded by others.

Each user must register and log in to the system before accessing or uploading notes. Once logged in, users can:

- Upload notes in PDF format.
- View details of uploaded notes (title, uploader, date, and file).
- Edit or delete their own notes.
- Access notes uploaded by other users.

The project ensures data security and privacy by associating each uploaded note with the logged-in user and restricting edit/delete actions only to the uploader.

this portal acts as a **knowledge-sharing hub** where all authenticated users benefit equally by sharing and accessing academic resources.

# Tools & Techniques Used

- **Programming Language**: C# with ASP.NET Core MVC (Model-View-Controller framework).
- **Backend Database**: MySQL (for storing users, sessions, and notes data).
- **Frontend**: HTML5, CSS3 (custom modern CSS styles), Razor syntax for server-side rendering.
- **File Handling**: Uploaded PDFs are stored in the server's wwwroot/uploads directory with unique GUID filenames to avoid conflicts.
- **Authentication**: ASP.NET Core Session is used to maintain user login states (UserId, UserName).
- **IDE/Editor**: Visual Studio 2022

# Techniques implemented

- **Session Management** to track logged-in users.
- **File Upload & Validation** (restrict to PDF, limit to 20 MB).
- **CRUD Operations**: Create (upload notes), Read (view notes), Update (edit notes), Delete (remove notes).
- **Access Control**: Only uploader can edit/delete their notes.

# Code

### 1.Login.cshtml

```
@{
   ViewData["Title"] = "Login";
}
<h2>Login</h2>
<form asp-action="Login" method="post">
   @Html.AntiForgeryToken()
   <div><label>Email</label><input name="email" /></div>
   <div><label>Password</label><input name="password" type="password"
/></div>
   <button type="submit">Login</button>
</form>
@if (ViewBag.Error != null)
{
   <p style="color:red">@ViewBag.Error</p>
}
```

### 2.Register.cshtml

```
@{
   ViewData["Title"] = "Register";
}
<h2>Register</h2>
<form asp-action="Register" method="post">
   @Html.AntiForgeryToken()
   <div><label>Name</label><input name="name" /></div>
   <div><label>Email</label><input name="email" /></div>
   <div><label>Password</label><input name="password" type="password"
/></div>
   <button type="submit">Register</button>
```

```
</form>
@if (ViewBag.Error != null)
{
    <p style="color:red">@ViewBag.Error</p>
}
```

**3.Index.cshtml**

```
@model IEnumerable<NotesSharing.Models.Note>
@{
    ViewData["Title"] = "Dashboard";
}
<h2>Dashboard</h2>

<h3>Upload New Note</h3>
<form asp-action="Create" method="post" enctype="multipart/form-data">
    @Html.AntiForgeryToken()
    <div><label>Title</label><input name="title" /></div>
    <div><label>PDF</label><input type="file" name="file" accept=".pdf" /></div>
    <button type="submit">Upload</button>
</form>
@if (TempData["Error"] != null)
{
    <p style="color:red">@TempData["Error"]</p>
}

<hr />
<h3>All Notes</h3>
<table>
    <thead><tr><th>Title</th><th>Uploader</th><th>Uploaded
At</th><th>Actions</th></tr></thead>
    <tbody>
```

```
        @foreach (var n in Model)
        {
            <tr>
                <td>@n.Title</td>
                <td>@n.UploaderName</td>
                <td>@n.UploadTime</td>
                <td class="table-actions">
                    <a asp-controller="Notes" asp-action="Details" asp-route-id="@n.Id" class="btn-link details-link">Details</a>
                    <a asp-controller="Notes" asp-action="Edit" asp-route-id="@n.Id" class="btn-link edit-link">Edit</a>
                    <a asp-controller="Notes" asp-action="Delete" asp-route-id="@n.Id" class="btn-link delete-link">Delete</a>
                </td>

            </tr>
        }
    </tbody>
</table>
```

**4.Profile.cshtml**

```
@{
    ViewData["Title"] = "Profile";
    var userName = Context.Session.GetString("UserName");
    var userEmail = Context.Session.GetString("UserEmail"); // If you set email in session
}

<div class="profile-container">
    <h2>My Profile</h2>

    @if (userName != null)
```

```
    {
        <div class="profile-card">
            <p><strong>Username:</strong> @userName</p>
            @if (!string.IsNullOrEmpty(userEmail))
            {
                <p><strong>Email:</strong> @userEmail</p>
            }
            <p><a asp-controller="Account" asp-action="Logout" class="logout-
btn">Logout</a></p>
        </div>
    }
    else
    {
        <p>You are not logged in. Please <a asp-controller="Account" asp-
action="Login">Login</a></p>
    }
</div>


<style>
    .profile-container {
        max-width: 600px;
        margin: 50px auto;
        background: #f9f9f9;
        padding: 25px;
        border-radius: 12px;
        box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    }

        .profile-container h2 {
            text-align: center;
            margin-bottom: 20px;
```

```
            color: #007BFF;
        }

    .profile-card {
        font-size: 18px;
        line-height: 1.6;
        color: #333;
    }

        .profile-card p {
            margin: 12px 0;
        }

    .logout-btn {
        display: inline-block;
        margin-top: 15px;
        padding: 10px 18px;
        background: #007BFF;
        color: #fff;
        text-decoration: none;
        border-radius: 6px;
        transition: 0.3s;
    }

        .logout-btn:hover {
            background: #0056b3;
        }
</style>
```

**5.All.cshtml**

```
@model IEnumerable<NotesSharing.Models.Note>

<h2>All Notes (Shared)</h2>

<table class="table table-bordered table-striped">
    <thead>
        <tr>
            <th>Title</th>
            <th>Uploader</th>
            <th>Uploaded On</th>
            <th>Download</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var note in Model)
        {
            <tr>
                <td>@note.Title</td>
                <td>@note.UploaderName</td>
                <td>@note.UploadTime.ToString("g")</td>
                <td><a href="@note.FilePath" target="_blank" class="btn-link details-
link">  View</a></td>
            </tr>
        }
    </tbody>
</table>
```

**6.Details.cshtml**

```
@model NotesSharing.Models.Note

<h2>Note Details</h2>

<p><strong>Title:</strong> @Model.Title</p>
<p><strong>Uploaded By:</strong> @Model.UploaderName</p>
<p><strong>Uploaded On:</strong> @Model.UploadTime.ToString("g")</p>
<p><a class="btn btn-primary" href="@Model.FilePath" target="_blank">📄 View PDF</a></p>

<a class="btn btn-secondary" asp-action="Index">Back to Dashboard</a>
```

**7.Edit.cshtml**

```
@model NotesSharing.Models.Note

<h2>Edit Note</h2>

<form asp-action="Edit" method="post" enctype="multipart/form-data">
    <input type="hidden" name="id" value="@Model.Id" />
    <div>
        <label>Title</label>
        <input type="text" name="title" value="@Model.Title" required />
    </div>
    <div>
        <label>Replace File (PDF only)</label>
        <input type="file" name="file" />
    </div>
    <button type="submit" class="btn btn-primary">💾 Save Changes</button>
    <a class="btn btn-secondary" asp-action="Index">Cancel</a>
</form>
```

**8.Delete.cshtml**

```
@model NotesSharing.Models.Note

<h2>Delete Note</h2>

<p>Are you sure you want to delete <strong>@Model.Title</strong> uploaded by
<strong>@Model.UploaderName</strong>?</p>

<form asp-action="Delete" method="post">
    <input type="hidden" name="id" value="@Model.Id" />
    <button type="submit" class="btn btn-danger">🗑 Delete</button>
    <a class="btn btn-secondary" asp-action="Index">Cancel</a>
</form>
```

**9._Layout.cshtml**

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>@ViewData["Title"] - NotesSharing</title>
    <link rel="stylesheet" href="~/css/site.css" />
    <style>
        nav {
            display: flex;
            justify-content: space-between;
            align-items: center;
            padding: 10px 20px;
            background: #007BFF;
        }
```

```
        .nav-left a {
            font-weight: bold;
            text-decoration: none;
            color: white;
        }

        .nav-middle {
            flex: 1;
            text-align: center;
            color: white;
            font-weight: bold;
        }

        .nav-right a,
        .nav-right span {
            margin-left: 15px;
            text-decoration: none;
            color: white;
        }

        .nav-right a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <nav>

        <div class="nav-left">
            <a asp-controller="Home" asp-action="Index">NotesSharing</a>
```

```
</div>

<div class="nav-middle">
    @if (Context.Session.GetInt32("UserId") != null)
    {
        <span>Hello, @Context.Session.GetString("UserName")</span>
    }
</div>

<div class="nav-right" id="navLinks">

    @if (Context.Session.GetInt32("UserId") != null)
    {
        <a asp-controller="Account" asp-action="Profile">Profile</a>
        <a asp-controller="Notes" asp-action="Index">Dashboard</a>

            <a asp-controller="Notes" asp-action="All">All Notes</a>

    }
    else
    {
```

```
            <a asp-controller="Account" asp-action="Login">Login</a>
            <a asp-controller="Account" asp-action="Register">Register</a>
        }
    </div>
</nav>

<div class="container">
    @RenderBody()
</div>

<script>
    function toggleMenu() {
        document.getElementById("navLinks").classList.toggle("show");
    }
</script>
</body>
</html>
```

**10.Accountcontroller.cs**

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using MySql.Data.MySqlClient;
using NotesSharing.Models;
using System;
using System.Security.Cryptography;
using System.Text;
using Microsoft.AspNetCore.Http;

namespace NotesSharing.Controllers
{
```

```
public class AccountController : Controller
{

    public IActionResult Profile()
    {
        if (HttpContext.Session.GetInt32("UserId") == null)
        {
            // Redirect to login if not logged in
            return RedirectToAction("Login");
        }

        ViewData["UserName"] = HttpContext.Session.GetString("UserName");
        return View();
    }

    private readonly string _connStr;
    public AccountController(IConfiguration config)
    {
        _connStr = config.GetConnectionString("MySqlConn");
    }

    // GET: /Account/Register
    public IActionResult Register() => View();

    [HttpPost]
    [ValidateAntiForgeryToken]
    public IActionResult Register(string name, string email, string password)
    {
        if (string.IsNullOrWhiteSpace(name) || string.IsNullOrWhiteSpace(email) ||
string.IsNullOrWhiteSpace(password))
        {
```

```
            ViewBag.Error = "Please fill all fields.";
            return View();
        }

        var hashed = Hash(password);

        using var conn = new MySqlConnection(_connStr);
        conn.Open();

        // check duplicate email
        using (var check = new MySqlCommand("SELECT COUNT(*) FROM users
WHERE email=@e", conn))
        {
            check.Parameters.AddWithValue("@e", email);
            var cnt = Convert.ToInt32(check.ExecuteScalar());
            if (cnt > 0)
            {
                ViewBag.Error = "Email already registered.";
                return View();
            }
        }

        using var cmd = new MySqlCommand("INSERT INTO users
(name,email,password) VALUES (@n,@e,@p)", conn);
        cmd.Parameters.AddWithValue("@n", name);
        cmd.Parameters.AddWithValue("@e", email);
        cmd.Parameters.AddWithValue("@p", hashed);
        cmd.ExecuteNonQuery();

        return RedirectToAction("Login");
    }
```

```
        // GET: /Account/Login
        public IActionResult Login() => View();

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Login(string email, string password)
        {
            if (string.IsNullOrWhiteSpace(email) || string.IsNullOrWhiteSpace(password))
            {
                ViewBag.Error = "Enter email and password.";
                return View();
            }

            var hashed = Hash(password);

            using var conn = new MySqlConnection(_connStr);
            conn.Open();
            using var cmd = new MySqlCommand("SELECT id, name FROM users
WHERE email=@e AND password=@p", conn);
            cmd.Parameters.AddWithValue("@e", email);
            cmd.Parameters.AddWithValue("@p", hashed);
            using var reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                int userId = reader.GetInt32("id");
                string name = reader.GetString("name");

                // set session
                HttpContext.Session.SetInt32("UserId", userId);
                HttpContext.Session.SetString("UserName", name);
```

```
            reader.Close();

            // log session in DB and save session id in session
            using var cmd2 = new MySqlCommand("INSERT INTO sessions (user_id, ip_address) VALUES (@u, @ip); SELECT LAST_INSERT_ID();", conn);
            cmd2.Parameters.AddWithValue("@u", userId);
            var ip = HttpContext.Connection.RemoteIpAddress?.ToString() ?? "0.0.0.0";
            cmd2.Parameters.AddWithValue("@ip", ip);
            var lastId = Convert.ToInt32(cmd2.ExecuteScalar());
            HttpContext.Session.SetInt32("SessionId", lastId);

            return RedirectToAction("Index", "Notes");
        }

        ViewBag.Error = "Invalid credentials.";
        return View();
    }

    public IActionResult Logout()
    {
        var sId = HttpContext.Session.GetInt32("SessionId");
        if (sId.HasValue)
        {
            using var conn = new MySqlConnection(_connStr);
            conn.Open();
            using var cmd = new MySqlCommand("UPDATE sessions SET logout_time=NOW() WHERE id=@s", conn);
            cmd.Parameters.AddWithValue("@s", sId.Value);
            cmd.ExecuteNonQuery();
```

```
            }

            HttpContext.Session.Clear();
            return RedirectToAction("Login");
        }

        // simple hash helper
        private string Hash(string input)
        {
            using var sha = SHA256.Create();
            var bs = sha.ComputeHash(Encoding.UTF8.GetBytes(input));
            return Convert.ToBase64String(bs);
        }
    }
}
```

**11.NotesController.cs**

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using MySql.Data.MySqlClient;
using NotesSharing.Models;
using System;
using System.Collections.Generic;
using System.IO;
using Microsoft.AspNetCore.Http;

namespace NotesSharing.Controllers
{
    public class NotesController : Controller
    {
```

```
        private readonly string _connStr;
        private readonly string _uploadFolder;



        // Show all notes from all users
        public IActionResult All()
        {
            if (CurrentUserId() == null) return RedirectToAction("Login", "Account");

            var notes = new List<Note>();
            using var conn = new MySqlConnection(_connStr);
            conn.Open();
            using var cmd = new MySqlCommand(
                "SELECT n.id, n.title, n.file_path, n.upload_time, u.name AS uploader " +
                "FROM notes n JOIN users u ON n.uploaded_by=u.id ORDER BY
n.upload_time DESC", conn);

            using var reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                notes.Add(new Note
                {
                    Id = reader.GetInt32("id"),
                    Title = reader.GetString("title"),
                    FilePath = reader.GetString("file_path"),
                    UploadTime = reader.GetDateTime("upload_time"),
                    UploaderName = reader.GetString("uploader")
                });
            }

            return View(notes);
```

```
    }


    public NotesController(IConfiguration config)
    {
        _connStr = config.GetConnectionString("MySqlConn");
        _uploadFolder = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot",
"uploads");
        if (!Directory.Exists(_uploadFolder))
Directory.CreateDirectory(_uploadFolder);
    }

    private int? CurrentUserId() => HttpContext.Session.GetInt32("UserId");

    // Dashboard (list notes of logged-in user only)
    public IActionResult Index()
    {
        if (CurrentUserId() == null) return RedirectToAction("Login", "Account");

        var notes = new List<Note>();
        using var conn = new MySqlConnection(_connStr);
        conn.Open();

        // ✅ Only get current user's notes
        using var cmd = new MySqlCommand(
            "SELECT n.id, n.title, n.file_path, n.upload_time, u.name AS uploader " +
            "FROM notes n JOIN users u ON n.uploaded_by=u.id " +
            "WHERE n.uploaded_by=@uid " +
            "ORDER BY n.upload_time DESC", conn);
        cmd.Parameters.AddWithValue("@uid", CurrentUserId().Value);
```

```
            using var reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                notes.Add(new Note
                {
                    Id = reader.GetInt32("id"),
                    Title = reader.GetString("title"),
                    FilePath = reader.GetString("file_path"),
                    UploadTime = reader.GetDateTime("upload_time"),
                    UploaderName = reader.GetString("uploader")
                });
            }
            return View(notes);
        }

        // Show Upload form
        public IActionResult Create()
        {
            if (CurrentUserId() == null) return RedirectToAction("Login", "Account");
            return View();
        }

        // Upload Note (PDF)
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Create(string title, IFormFile file)
        {
            if (CurrentUserId() == null) return RedirectToAction("Login", "Account");
            if (file == null || file.Length == 0) { TempData["Error"] = "Select a PDF file.";
return RedirectToAction("Index"); }
            var ext = Path.GetExtension(file.FileName).ToLowerInvariant();
```

```
        if (ext != ".pdf") { TempData["Error"] = "Only PDF allowed."; return
RedirectToAction("Index"); }
        if (file.Length > 20 * 1024 * 1024) { TempData["Error"] = "Max 20 MB.";
return RedirectToAction("Index"); }

        var unique = $"{Guid.NewGuid()}{ext}";
        var savePath = Path.Combine(_uploadFolder, unique);
        using (var fs = new FileStream(savePath, FileMode.Create))
        {
            file.CopyTo(fs);
        }

        var relPath = "/uploads/" + unique;

        using var conn = new MySqlConnection(_connStr);
        conn.Open();
        using var cmd = new MySqlCommand("INSERT INTO notes (title, file_path,
uploaded_by) VALUES (@t, @f, @u)", conn);
        cmd.Parameters.AddWithValue("@t", title);
        cmd.Parameters.AddWithValue("@f", relPath);
        cmd.Parameters.AddWithValue("@u", CurrentUserId().Value);  // ✅ Assign
logged-in user
        cmd.ExecuteNonQuery();

        return RedirectToAction("Index");
    }

    // View Note Details
    public IActionResult Details(int id)
    {
        if (CurrentUserId() == null) return RedirectToAction("Login", "Account");
        Note note = null;
```

```
        using var conn = new MySqlConnection(_connStr);
        conn.Open();
        using var cmd = new MySqlCommand("SELECT n.*, u.name as uploader
FROM notes n JOIN users u ON n.uploaded_by=u.id WHERE n.id=@id", conn);
        cmd.Parameters.AddWithValue("@id", id);
        using var r = cmd.ExecuteReader();
        if (r.Read())
        {
            note = new Note
            {
                Id = r.GetInt32("id"),
                Title = r.GetString("title"),
                FilePath = r.GetString("file_path"),
                UploadTime = r.GetDateTime("upload_time"),
                UploadedBy = r.GetInt32("uploaded_by"),
                UploaderName = r.GetString("uploader")
            };
        }
        if (note == null) return NotFound();
        if (note.UploadedBy != CurrentUserId()) return Unauthorized();
        return View(note);
    }

    // Edit (GET)
    public IActionResult Edit(int id)
    {
        if (CurrentUserId() == null) return RedirectToAction("Login", "Account");
        using var conn = new MySqlConnection(_connStr);
        conn.Open();
        using var cmd = new MySqlCommand("SELECT * FROM notes WHERE
id=@id", conn);
```

```
        cmd.Parameters.AddWithValue("@id", id);
        using var r = cmd.ExecuteReader();
        if (r.Read())
        {
            var note = new Note
            {
                Id = r.GetInt32("id"),
                Title = r.GetString("title"),
                FilePath = r.GetString("file_path"),
                UploadedBy = r.GetInt32("uploaded_by")
            };
            if (note.UploadedBy != CurrentUserId()) return Unauthorized();
            return View(note);
        }
        return NotFound();
    }

    // Edit (POST)
    [HttpPost]
    [ValidateAntiForgeryToken]
    public IActionResult Edit(int id, string title, IFormFile file)
    {
        if (CurrentUserId() == null) return RedirectToAction("Login", "Account");

        using var conn = new MySqlConnection(_connStr);
        conn.Open();

        string oldPath = null;
        int owner = 0;
        using (var g = new MySqlCommand("SELECT file_path, uploaded_by FROM
notes WHERE id=@id", conn))
```

```
        {
            g.Parameters.AddWithValue("@id", id);
            using var r = g.ExecuteReader();
            if (r.Read()) { oldPath = r.GetString("file_path"); owner =
r.GetInt32("uploaded_by"); }
            else return NotFound();
        }

        if (owner != CurrentUserId()) return Unauthorized();

        string newRelPath = oldPath;
        if (file != null && file.Length > 0)
        {
            var ext = Path.GetExtension(file.FileName).ToLowerInvariant();
            if (ext != ".pdf") { TempData["Error"] = "Only PDF allowed."; return
RedirectToAction("Edit", new { id }); }
            var unique = $"{Guid.NewGuid()}{ext}";
            var savePath = Path.Combine(_uploadFolder, unique);
            using (var fs = new FileStream(savePath, FileMode.Create))
file.CopyTo(fs);
            newRelPath = "/uploads/" + unique;

            // delete old file
            var oldPhysical = Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot", oldPath.TrimStart('/'));
            if (System.IO.File.Exists(oldPhysical)) System.IO.File.Delete(oldPhysical);
        }

        using var cmd = new MySqlCommand("UPDATE notes SET title=@t,
file_path=@f WHERE id=@id", conn);
        cmd.Parameters.AddWithValue("@t", title);
        cmd.Parameters.AddWithValue("@f", newRelPath);
```

```
            cmd.Parameters.AddWithValue("@id", id);
            cmd.ExecuteNonQuery();

            return RedirectToAction("Index");
        }

        // Delete (GET confirm)
        public IActionResult Delete(int id)
        {
            if (CurrentUserId() == null) return RedirectToAction("Login", "Account");
            Note note = null;
            using var conn = new MySqlConnection(_connStr);
            conn.Open();
            using var cmd = new MySqlCommand("SELECT n.*, u.name as uploader
FROM notes n JOIN users u ON n.uploaded_by=u.id WHERE n.id=@id", conn);
            cmd.Parameters.AddWithValue("@id", id);
            using var r = cmd.ExecuteReader();
            if (r.Read())
            {
                note = new Note
                {
                    Id = r.GetInt32("id"),
                    Title = r.GetString("title"),
                    FilePath = r.GetString("file_path"),
                    UploadTime = r.GetDateTime("upload_time"),
                    UploaderName = r.GetString("uploader"),
                    UploadedBy = r.GetInt32("uploaded_by")
                };
            }
            if (note == null) return NotFound();
            if (note.UploadedBy != CurrentUserId()) return Unauthorized();
```

```
        return View(note);
    }

    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public IActionResult DeleteConfirmed(int id)
    {
        if (CurrentUserId() == null) return RedirectToAction("Login", "Account");
        using var conn = new MySqlConnection(_connStr);
        conn.Open();

        string path = null;
        int owner = 0;
        using (var g = new MySqlCommand("SELECT file_path, uploaded_by FROM
notes WHERE id=@id", conn))
        {
            g.Parameters.AddWithValue("@id", id);
            using var r = g.ExecuteReader();
            if (r.Read()) { path = r.GetString("file_path"); owner =
r.GetInt32("uploaded_by"); } else return NotFound();
        }
        if (owner != CurrentUserId()) return Unauthorized();

        using var del = new MySqlCommand("DELETE FROM notes WHERE
id=@id", conn);
        del.Parameters.AddWithValue("@id", id);
        del.ExecuteNonQuery();

        var physical = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot",
path.TrimStart('/'));
        if (System.IO.File.Exists(physical)) System.IO.File.Delete(physical);
```

```
            return RedirectToAction("Index");
        }
    }
}
```

**12.HomeController.cs**

```
using System.Diagnostics;
using Microsoft.AspNetCore.Mvc;
using NotesSharing.Models;

namespace NotesSharing.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Privacy()
        {
            return View();
        }
```

```
    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
NoStore = true)]
    public IActionResult Error()
    {
        return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
    }
  }
}
```

**13.Notes.cs**

```
using System;

namespace NotesSharing.Models
{
  public class Note
  {
    public int Id { get; set; }
    public string Title { get; set; }
    public string FilePath { get; set; } // e.g. /uploads/uuid.pdf
    public int UploadedBy { get; set; }
    public DateTime UploadTime { get; set; }
    public string UploaderName { get; set; } // for easy display from join
  }
}
```

**14.user.cs**

```
using System;

namespace NotesSharing.Models
{
   public class User
   {
      public int Id { get; set; }
      public string Name { get; set; }
      public string Email { get; set; }
      public string Password { get; set; } // stored hashed
      public DateTime CreatedAt { get; set; }
   }
}
```

**15.appsetting.json**

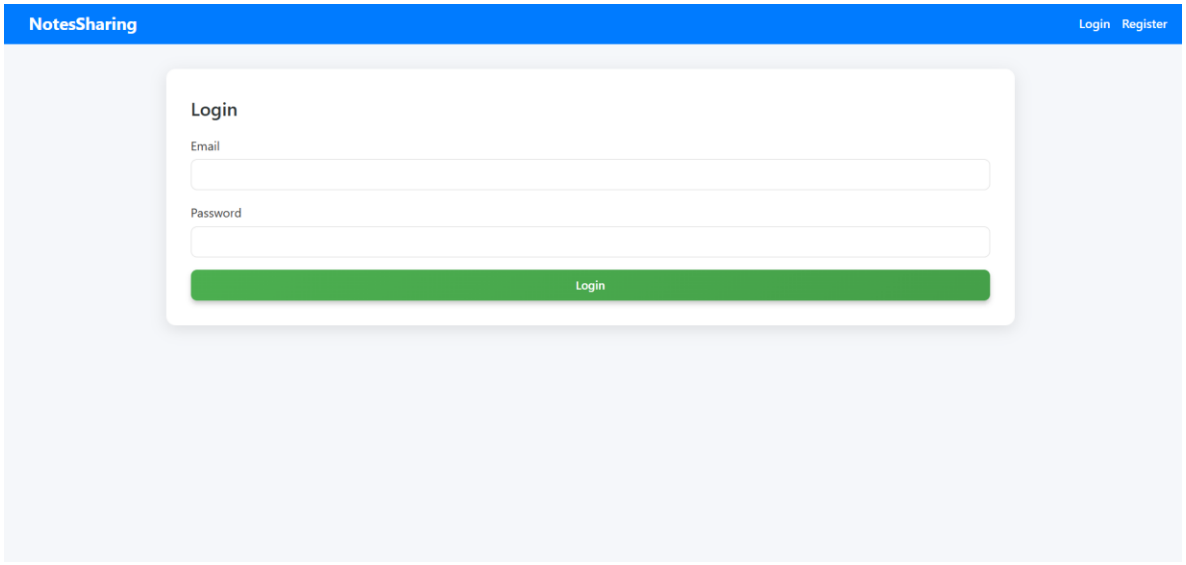
```
{
 "Logging": {
   "LogLevel": {
     "Default": "Information",
     "Microsoft.AspNetCore": "Warning"
   }
 },
 "AllowedHosts": "*",

 "ConnectionStrings": {
   "MySqlConn":
"server=127.0.0.1;port=3306;database=notesdb;user=root;password=;"
 }
}
```

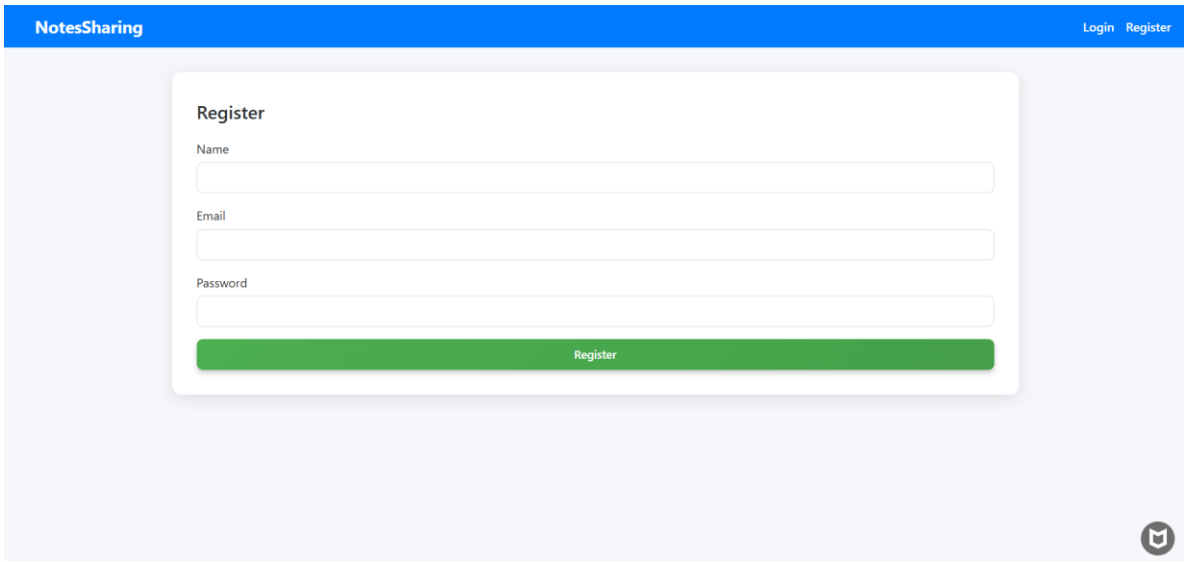
**16.Program.cs**

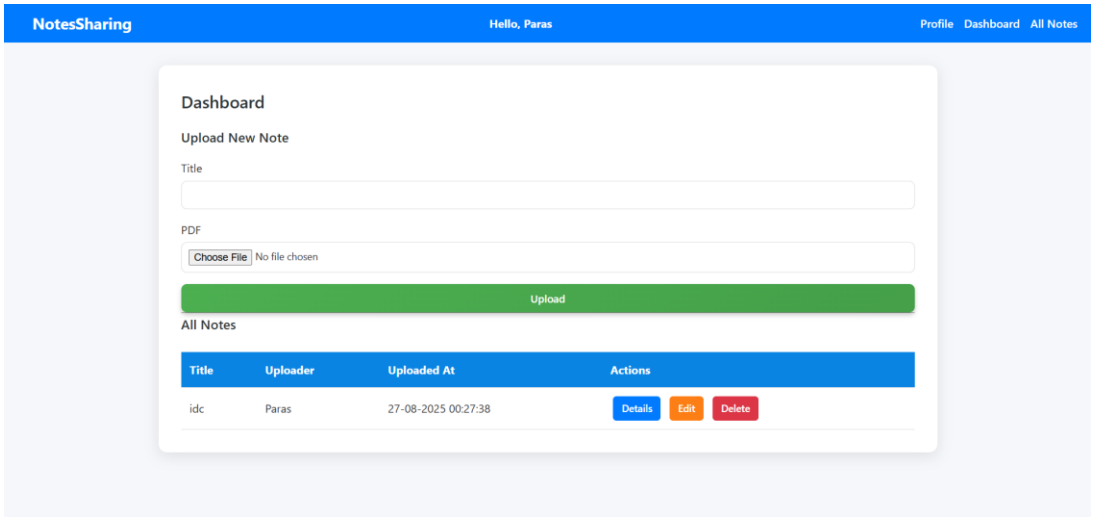
```
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System;

var builder = WebApplication.CreateBuilder(args);

// Add services
builder.Services.AddControllersWithViews();
builder.Services.AddDistributedMemoryCache();
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(60);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});

var app = builder.Build();

if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseStaticFiles();
app.UseRouting();
app.UseSession();
```

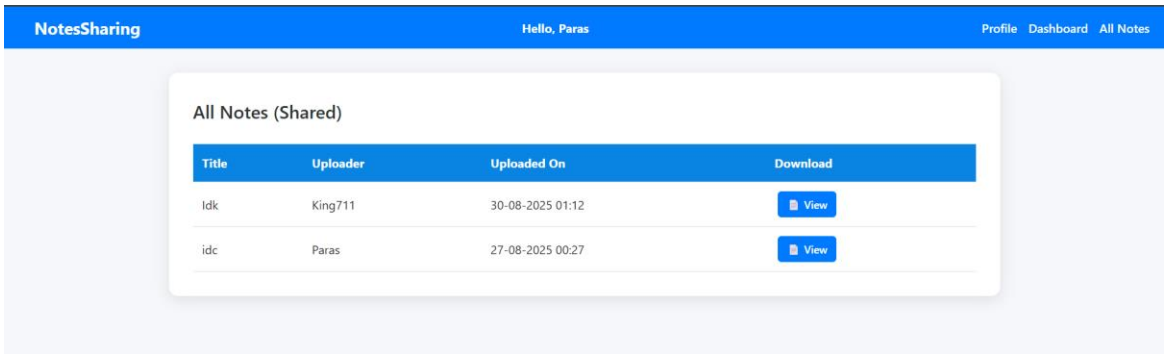
```
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Account}/{action=Login}/{id?}");

app.Run();
```

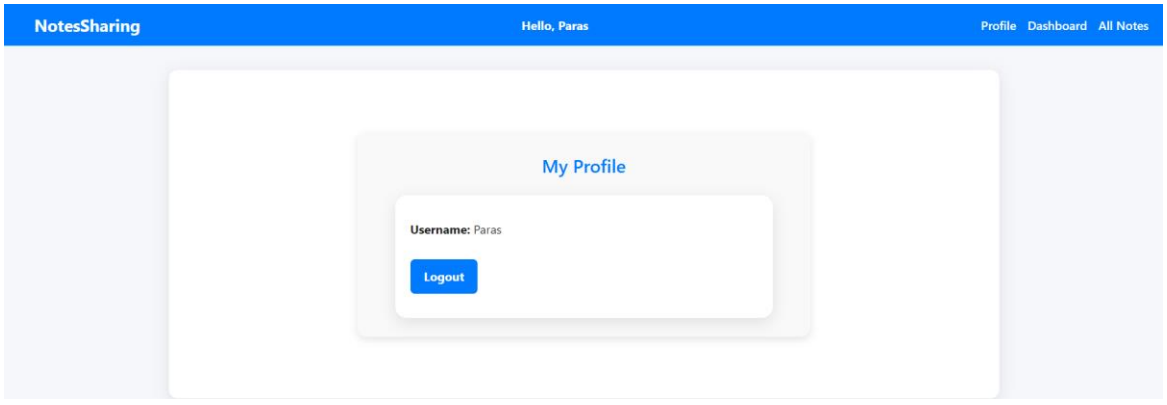
# Output




NotesSharing
Hello, Paras
Profile Dashboard All Notes
Dashboard
Upload New Note
Title
PDF
Choose File No file chosen
Upload
All Notes
Title Uploader Uploaded At Actions
idc Paras 27-08-2025 00:27:38
Details Edit Delete


NotesSharing
Hello, Paras
Profile Dashboard All Notes
All Notes (Shared)
Title Uploader Uploaded On Download
Idk King711 30-08-2025 01:12
View
idc Paras 27-08-2025 00:27
View


NotesSharing
Hello, Paras
Profile Dashboard All Notes
My Profile
Username: Paras
Logout

# Conclusion

The **Online Notes Sharing portal** successfully demonstrates a platform for collaborative learning by allowing users to upload and access study materials easily. It ensures secure login, user-specific operations, and global access to notes for all authenticated users.

**Key achievements:**

- Implemented user authentication and session handling.
- Designed CRUD operations with strict access control for note management.
- Enabled file uploads with validation and storage security.

Overall, this portal is a practical implementation of ASP.NET Core MVC with MySQL and helps in understanding real-world concepts of web development such as authentication, file handling, and database integration.