

[program.cs](http://program.cs)

using Microsoft.AspNetCore.Identity;

using Microsoft.EntityFrameworkCore;

using ProductManagement.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

// Configure EF Core with SQL Server connection string

builder.Services.AddDbContext<ProductContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

// Configure Identity with ApplicationUser and IdentityRole

builder.Services.AddIdentity<ApplicationUser, IdentityRole>(options =>

{

// Password policy configuration

options.Password.RequireDigit = true;

options.Password.RequiredLength = 8;

options.Password.RequireUppercase = true;

options.Password.RequireNonAlphanumeric = true;

})

.AddEntityFrameworkStores<ProductContext>()

.AddDefaultTokenProviders();

// Add MVC services with views

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

// Use appropriate error handling in production

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

// Enforce HTTPS

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

// Enable authentication and authorization middleware

app.UseAuthentication();

app.UseAuthorization();

// Map controller routes

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

[AccountController.cs](http://accountcontroller.cs)

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using ProductManagement.Models.ViewModels;

using ProductManagement.Models;

using System.Threading.Tasks;

public class AccountController : Controller

{

private readonly UserManager<ApplicationUser> \_userManager;

private readonly SignInManager<ApplicationUser> \_signInManager;

private readonly RoleManager<IdentityRole> \_roleManager;

public AccountController(UserManager<ApplicationUser> userManager,

SignInManager<ApplicationUser> signInManager,

RoleManager<IdentityRole> roleManager)

{

\_userManager = userManager;

\_signInManager = signInManager;

\_roleManager = roleManager;

}

[HttpGet]

public IActionResult Register() => View();

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Register(RegisterViewModel model)

{

if (!ModelState.IsValid) return View(model);

var user = new ApplicationUser { UserName = model.Username };

var result = await \_userManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

// Ensure roles exist

if (!await \_roleManager.RoleExistsAsync("Admin"))

await \_roleManager.CreateAsync(new IdentityRole("Admin"));

if (!await \_roleManager.RoleExistsAsync("Manager"))

await \_roleManager.CreateAsync(new IdentityRole("Manager"));

// Assign role

await \_userManager.AddToRoleAsync(user, model.Role);

await \_signInManager.SignInAsync(user, isPersistent: false);

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

}

foreach (var error in result.Errors)

ModelState.AddModelError("", error.Description);

return View(model);

}

[HttpGet]

public IActionResult Login() => View();

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Login(LoginViewModel model)

{

if (!ModelState.IsValid) return View(model);

var result = await \_signInManager.PasswordSignInAsync(model.Username, model.Password, false, false);

if (result.Succeeded)

{

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

}

ModelState.AddModelError("", "Invalid login attempt.");

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Logout()

{

await \_signInManager.SignOutAsync();

return RedirectToAction("Login", "Account");

}

}

[HomeController.cs](http://homecontroller.cs)

using System.Diagnostics;

using Microsoft.AspNetCore.Mvc;

using ProductManagement.Models;

namespace ProductManagement.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();

}

public IActionResult AccessDenied()

{

ViewBag.ErrorMessage = "Access Denied: You do not have permission to delete products.";

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

[ProductController.cs](http://productcontroller.cs)

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Authorization;

using Microsoft.EntityFrameworkCore;

using ProductManagement.Models;

using System.Threading.Tasks;

using System.Linq;

[Authorize(Roles = "Admin,Manager")]

public class ProductController : Controller

{

private readonly ProductContext \_context;

public ProductController(ProductContext context)

{

\_context = context;

}

// GET: /Product

public async Task<IActionResult> Index()

{

var products = await \_context.Products.ToListAsync();

return View(products);

}

// GET: Create (Admin only)

[Authorize(Roles = "Admin")]

public IActionResult Create() => View();

[HttpPost]

[ValidateAntiForgeryToken]

[Authorize(Roles = "Admin")]

public async Task<IActionResult> Create(Product product)

{

if (ModelState.IsValid)

{

\_context.Add(product);

await \_context.SaveChangesAsync();

TempData["Message"] = $"Product \"{product.Name}\" has been successfully created!";

return RedirectToAction(nameof(Index));

}

return View(product);

}

// GET: Edit (Admin, Manager)

[Authorize(Roles = "Admin,Manager")]

public async Task<IActionResult> Edit(int? id)

{

if (id == null) return NotFound();

var product = await \_context.Products.FindAsync(id);

if (product == null) return NotFound();

return View(product);

}

[HttpPost]

[ValidateAntiForgeryToken]

[Authorize(Roles = "Admin,Manager")]

public async Task<IActionResult> Edit(int id, Product product)

{

if (id != product.Id) return NotFound();

if (ModelState.IsValid)

{

try

{

\_context.Update(product);

await \_context.SaveChangesAsync();

TempData["Message"] = $"Product \"{product.Name}\" has been successfully updated!";

return RedirectToAction(nameof(Index));

}

catch (DbUpdateConcurrencyException)

{

if (!\_context.Products.Any(e => e.Id == product.Id)) return NotFound();

else throw;

}

}

return View(product);

}

// GET: Delete (Admin only)

[Authorize(Roles = "Admin")]

public async Task<IActionResult> Delete(int? id)

{

if (id == null) return NotFound();

var product = await \_context.Products.FirstOrDefaultAsync(m => m.Id == id);

if (product == null) return NotFound();

return View(product);

}

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

[Authorize(Roles = "Admin")]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var product = await \_context.Products.FindAsync(id);

if (product != null)

{

\_context.Products.Remove(product);

await \_context.SaveChangesAsync();

TempData["Message"] = $"Product \"{product.Name}\" has been successfully deleted!";

}

return RedirectToAction(nameof(Index));

}

}

[ProductContext.cs](http://productcontext.cs)

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

using ProductManagement.Models;

public class ProductContext : IdentityDbContext<ApplicationUser>

{

public ProductContext(DbContextOptions<ProductContext> options) : base(options) { }

public DbSet<Product> Products { get; set; }

}

[LoginViewModel.cs](http://loginviewmodel.cs)

namespace ProductManagement.Models.ViewModels

{

public class LoginViewModel

{

public string Username { get; set; }

public string Password { get; set; }

}

}

[RegisterViewModel.cs](http://registerviewmodel.cs)

namespace ProductManagement.Models.ViewModels

{

public class RegisterViewModel

{

public string Username { get; set; }

public string Password { get; set; }

public string Role { get; set; }

}

}

[ApplicationUser.cs](http://applicationuser.cs)

using Microsoft.AspNetCore.Identity;

namespace ProductManagement.Models;

public class ApplicationUser : IdentityUser

{

}

[ErrorViewMode.cs](http://errorviewmode.cs)

namespace ProductManagement.Models;

public class ErrorViewModel

{

public string? RequestId { get; set; }

public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

}

[Product.cs](http://product.cs)

using System;

namespace ProductManagement.Models

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

}

}

Views :-

account/login.cshtml

@model ProductManagement.Models.ViewModels.LoginViewModel

@{

ViewData["Title"] = "Login";

}

<h2>Login</h2>

<form asp-action="Login" method="post">

@Html.AntiForgeryToken()

<div asp-validation-summary="All" class="text-danger"></div>

<div>

<label asp-for="Username"></label>

<input asp-for="Username" class="form-control" />

<span asp-validation-for="Username" class="text-danger"></span>

</div>

<div>

<label asp-for="Password"></label>

<input asp-for="Password" class="form-control" type="password" />

<span asp-validation-for="Password" class="text-danger"></span>

</div>

<button type="submit" class="btn btn-primary">Login</button>

</form>

Register.cshtml

@model ProductManagement.Models.ViewModels.RegisterViewModel

@{

ViewData["Title"] = "Register";

}

<h2>Register</h2>

<form asp-action="Register" method="post">

@Html.AntiForgeryToken()

<div asp-validation-summary="All" class="text-danger"></div>

<div>

<label asp-for="Username"></label>

<input asp-for="Username" class="form-control" />

<span asp-validation-for="Username" class="text-danger"></span>

</div>

<div>

<label asp-for="Password"></label>

<input asp-for="Password" class="form-control" type="password" />

<span asp-validation-for="Password" class="text-danger"></span>

</div>

<div>

<label asp-for="Role"></label>

<select asp-for="Role" class="form-control">

<option>Admin</option>

<option>Manager</option>

</select>

<span asp-validation-for="Role" class="text-danger"></span>

</div>

<button type="submit" class="btn btn-primary">Register</button>

</form>

Home/AccessDenied.cshtml

@{

ViewData["Title"] = "Access Denied";

}

<h2>@ViewData["Title"]</h2>

<div class="alert alert-danger">

@ViewBag.ErrorMessage

</div>

product/create.cshtml

@model ProductManagement.Models.Product

@{

ViewData["Title"] = "Create Product";

}

<h2>Create New Product</h2>

<form asp-action="Create" method="post">

@Html.AntiForgeryToken()

<div asp-validation-summary="All" class="text-danger"></div>

<div>

<label asp-for="Name"></label>

<input asp-for="Name" class="form-control" />

<span asp-validation-for="Name" class="text-danger"></span>

</div>

<div>

<label asp-for="Price"></label>

<input asp-for="Price" class="form-control" />

<span asp-validation-for="Price" class="text-danger"></span>

</div>

<button type="submit" class="btn btn-primary">Create</button>

<a asp-action="Index" class="btn btn-secondary">Cancel</a>

</form>

<a asp-controller="Account" asp-action="Logout" class="btn btn-secondary">Logout</a>

Delete.cshtml

@model ProductManagement.Models.Product

@{

ViewData["Title"] = "Delete Product";

}

<h2>Delete Product</h2>

<form asp-action="Delete" method="post">

@Html.AntiForgeryToken()

<input type="hidden" asp-for="Id" />

<div>

<strong>Name:</strong> @Model.Name

</div>

<div>

<strong>Price:</strong> @Model.Price.ToString("C")

</div>

<p>Are you sure you want to delete this product?</p>

<button type="submit" class="btn btn-danger">Delete</button>

<a asp-action="Index" class="btn btn-secondary">Cancel</a>

</form>

Edit.cshtml

@model ProductManagement.Models.Product

@{

ViewData["Title"] = "Edit Product";

}

<h2>Edit Product</h2>

<form asp-action="Edit" method="post">

@Html.AntiForgeryToken()

<input type="hidden" asp-for="Id" />

<div asp-validation-summary="All" class="text-danger"></div>

<div>

<label asp-for="Name"></label>

<input asp-for="Name" class="form-control" />

<span asp-validation-for="Name" class="text-danger"></span>

</div>

<div>

<label asp-for="Price"></label>

<input asp-for="Price" class="form-control" />

<span asp-validation-for="Price" class="text-danger"></span>

</div>

<button type="submit" class="btn btn-primary">Save Changes</button>

<a asp-action="Index" class="btn btn-secondary">Cancel</a>

</form>

<a asp-controller="Account" asp-action="Logout" class="btn btn-secondary">Logout</a>

Index.cshtml

@model IEnumerable<ProductManagement.Models.Product>

<h2>Product List</h2>

@if (TempData["Message"] != null)

{

<div class="alert alert-success">@TempData["Message"]</div>

}

<table class="table">

<thead>

<tr><th>Id</th><th>Name</th><th>Price</th><th>Actions</th></tr>

</thead>

<tbody>

@foreach (var product in Model)

{

<tr>

<td>@product.Id</td>

<td>@product.Name</td>

<td>@product.Price.ToString("C")</td>

<td>

<a asp-action="Edit" asp-route-id="@product.Id">Edit</a>

@if (User.IsInRole("Admin"))

{

<a asp-action="Delete" asp-route-id="@product.Id" class="text-danger">Delete</a>

}

</td>

</tr>

}

</tbody>

</table>

@if (User.IsInRole("Admin"))

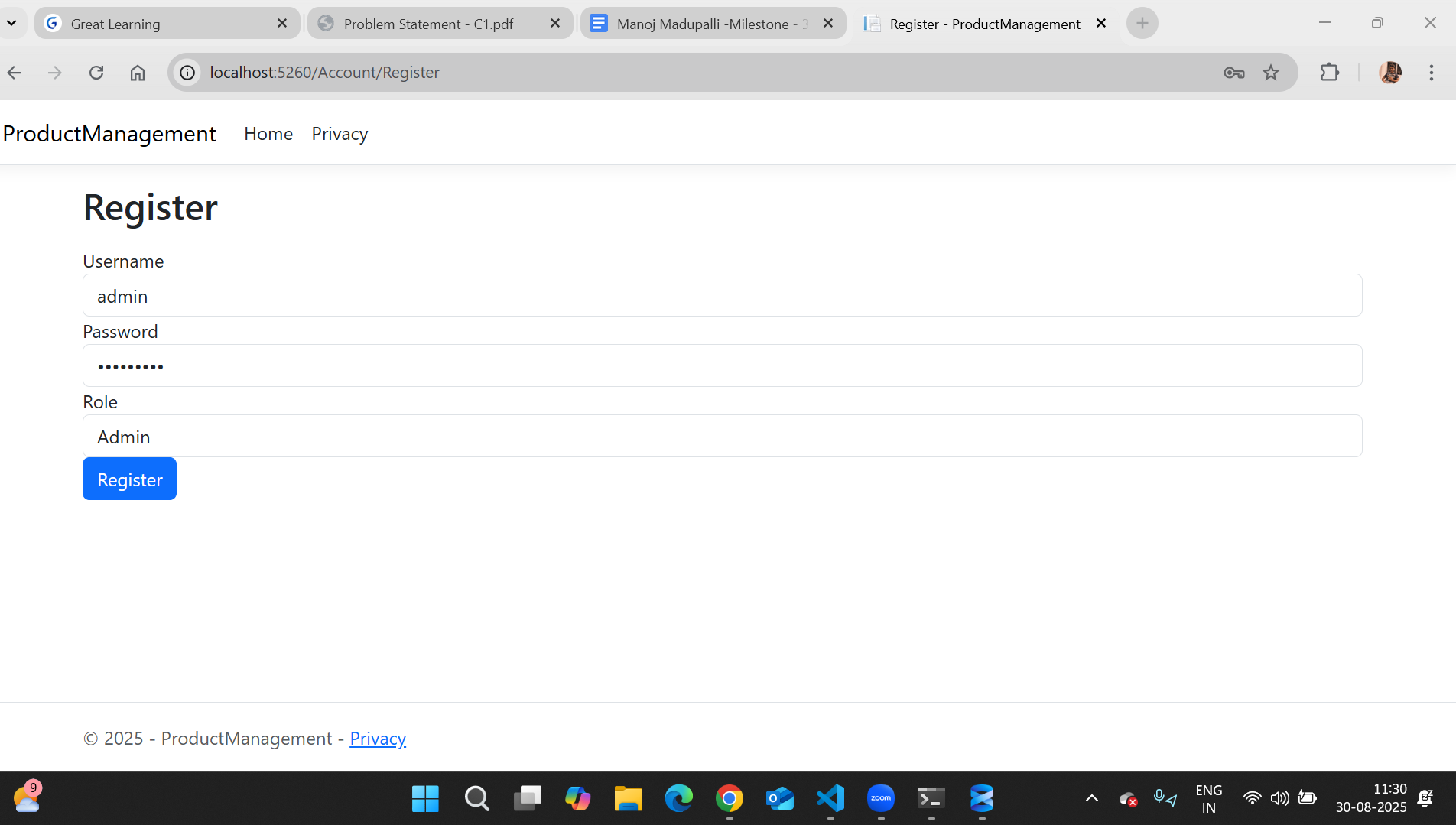
{

<a asp-action="Create" class="btn btn-success">Add New Product</a>

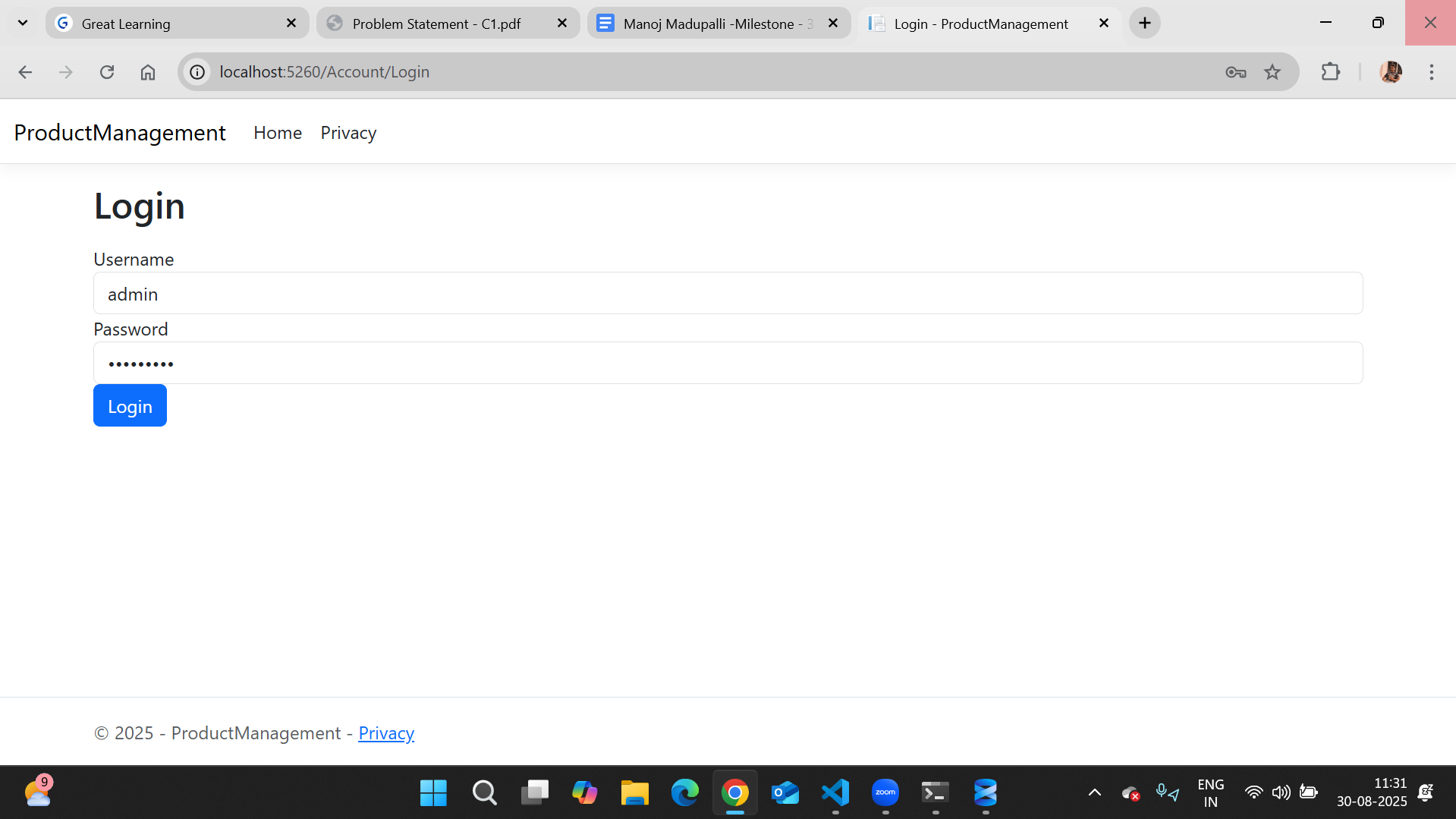
}

Output Screenshots :-

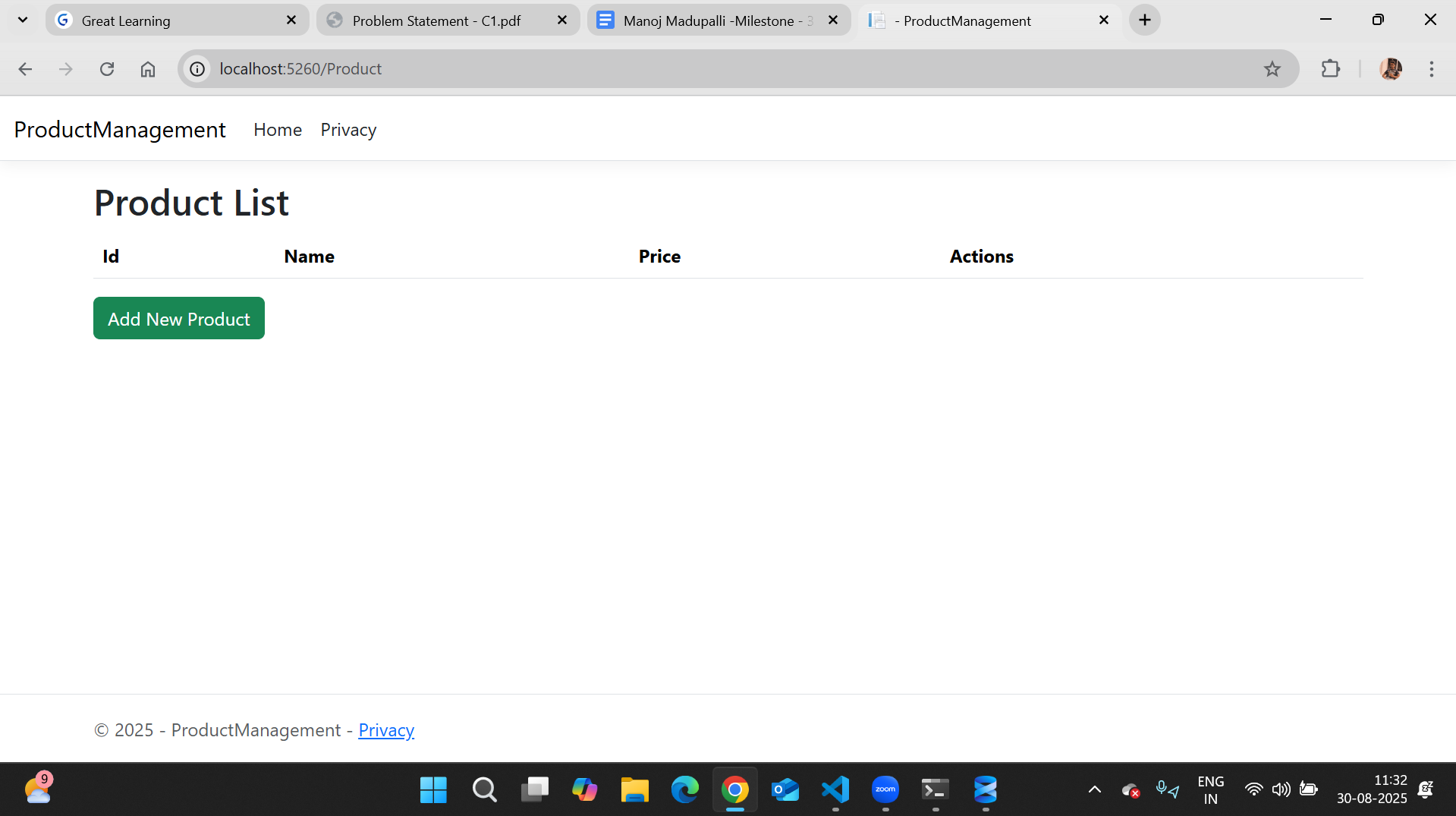
Registering admin :-



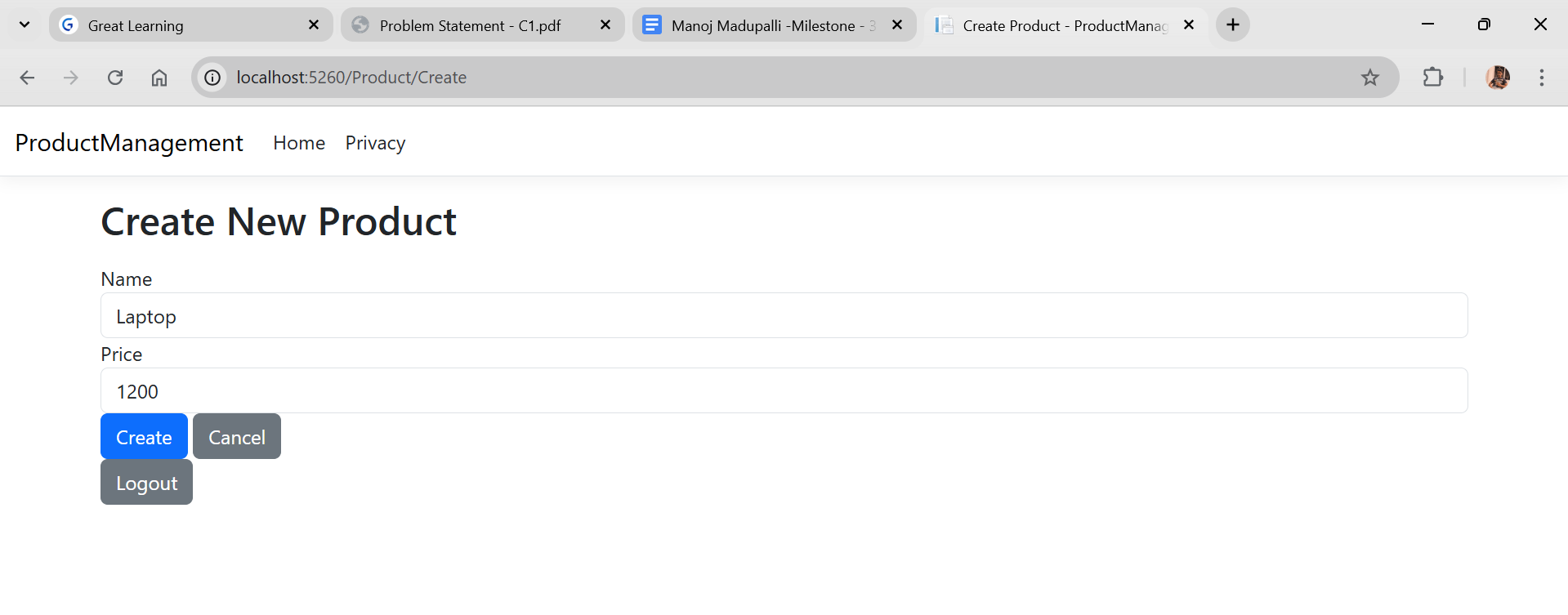
Login page



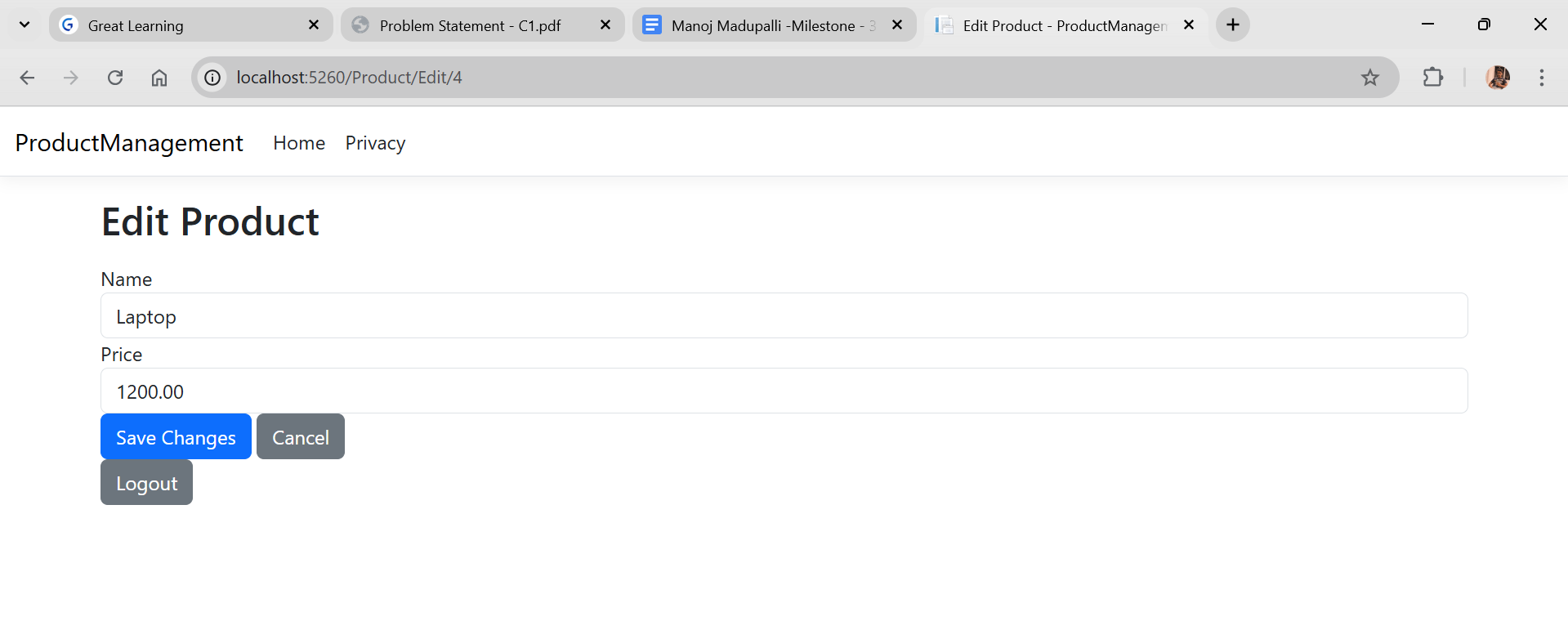
Initial product page



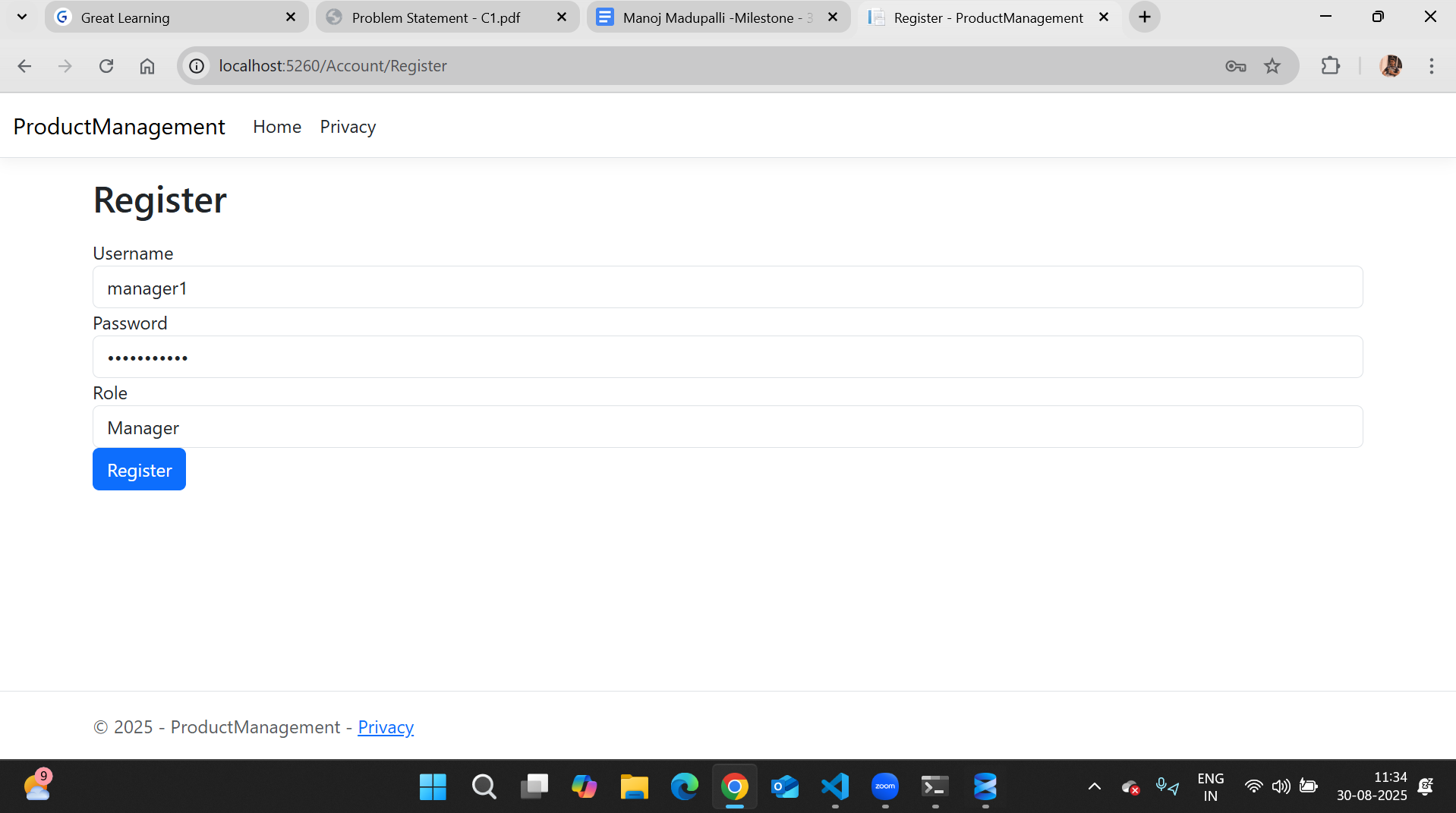
Creating a new product as admin



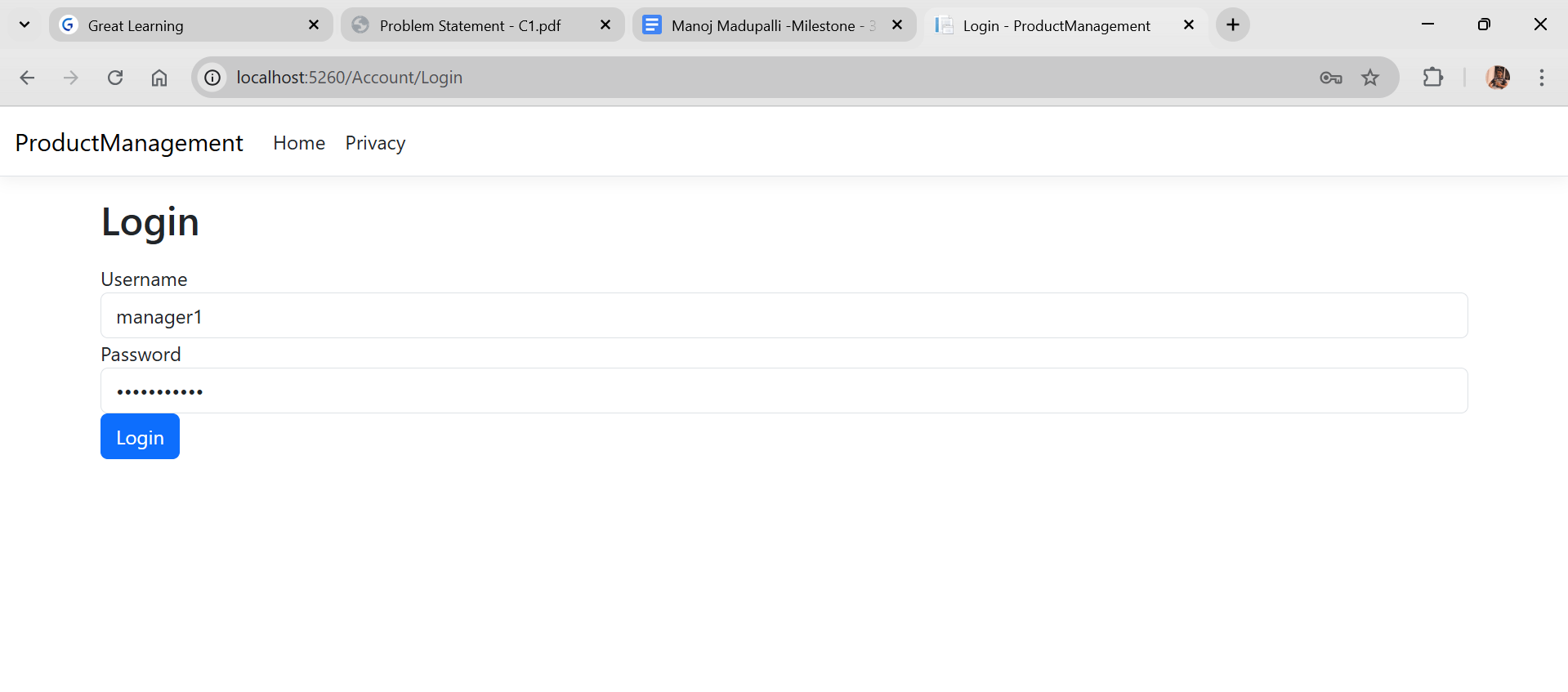
Edit product as admin



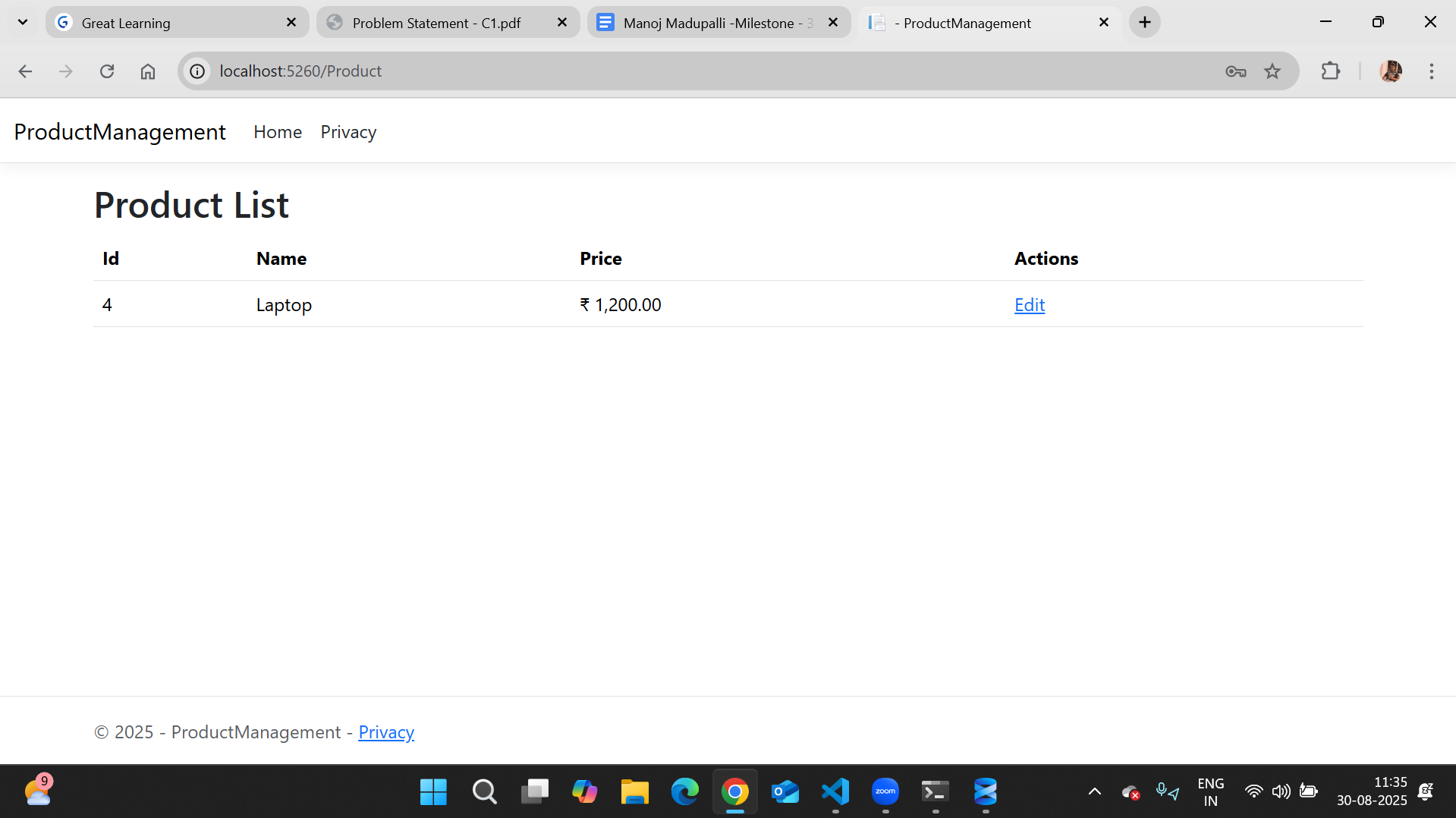
Registering manager :-



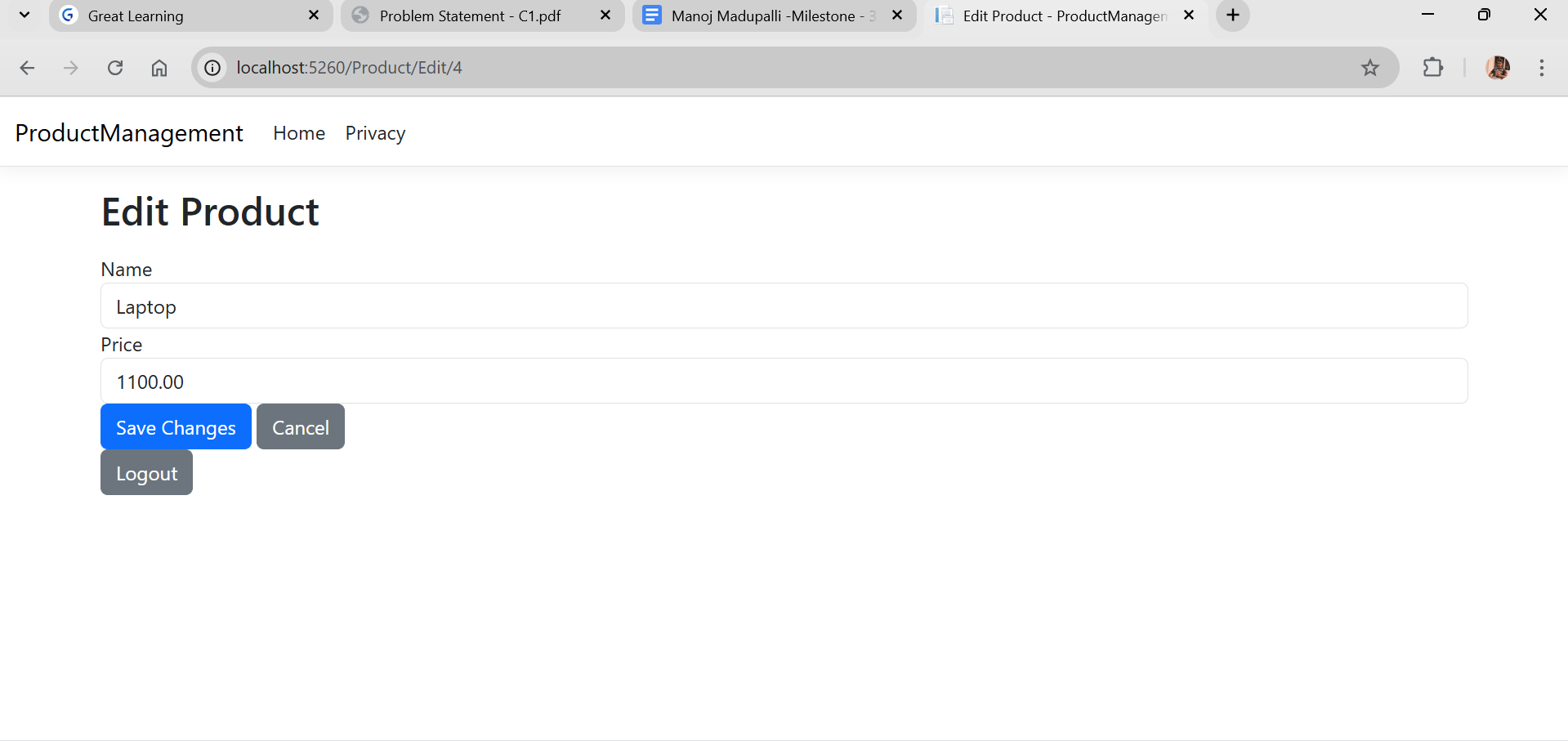
Login as manager :-



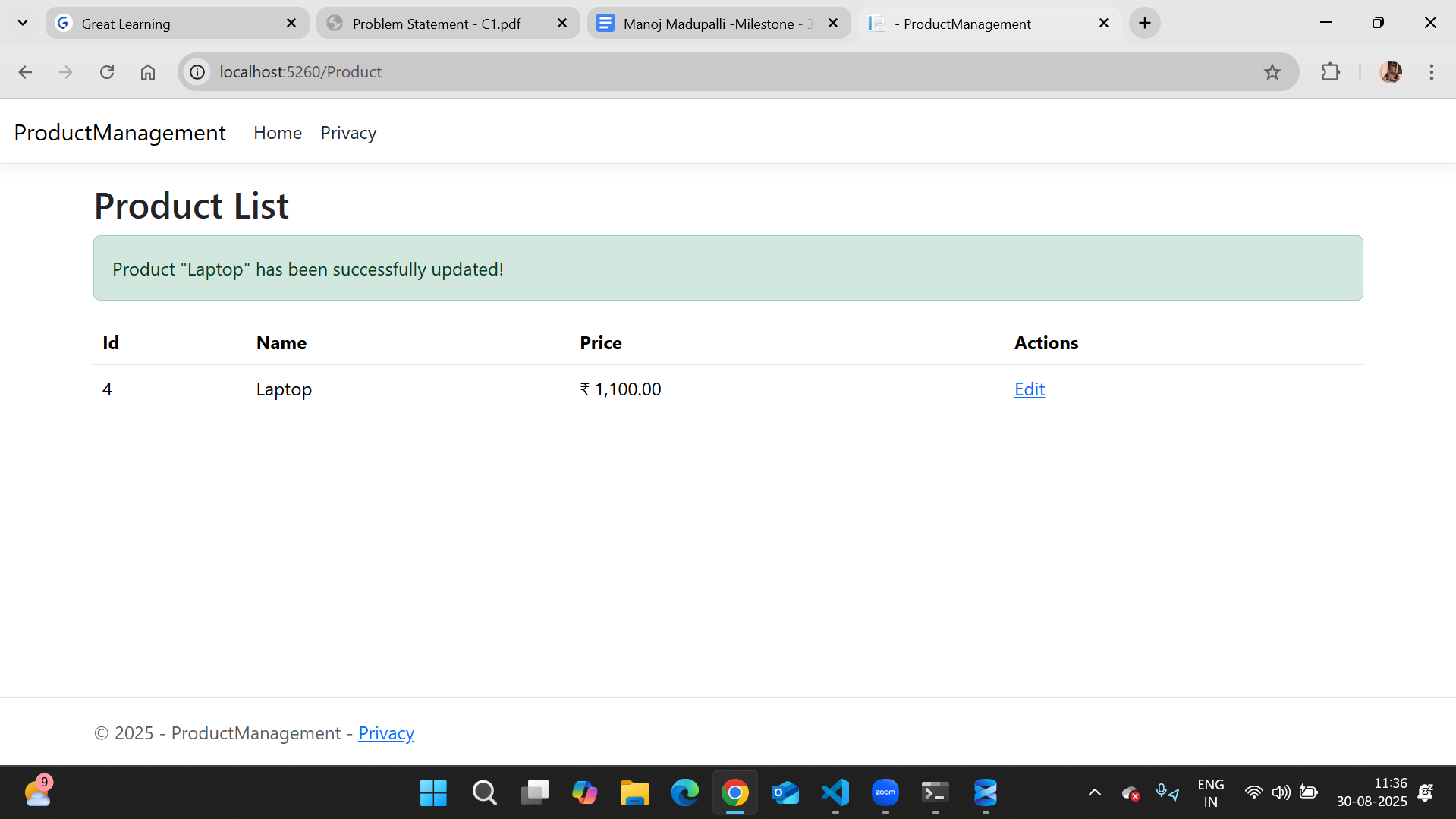
Manager HomePage :-



Edit product as Manager page



Product updated page after changes



Delete product page as admin :-

