PRACTICAL 11

Aim: Develop ASP.NET Core MVC application with following functionalities:
• Create model classes "Product", "Category", "Supplier"
• Use code first approach with entity framework
• Store and retrieve data to/from MySQL database.

Code:

Add Entity Framework Core and Tools

```
>> dotnet add package Pomelo.EntityFrameworkCore.MySql
>> dotnet add package Microsoft.EntityFrameworkCore.Design
>> dotnet add package Microsoft.EntityFrameworkCore
```

Configure Database in 'appsettings.json' :

```
"ConnectionStrings": {
  "DefaultConnection":
"Server=localhost;Database=PRACTICAL11db;User=root;Password=yourPassword;"
 },
```

Program.cs :

```
using Microsoft.EntityFrameworkCore;
using PRACTICAL11.Data;
builder.Services.AddDbContext<AppDbContext>(options =>
  options.UseMySql(builder.Configuration.GetConnectionString("DefaultConnection"),
    new MySqlServerVersion(new Version(8, 0, 26))));
```

Product.cs : (Models / Product.cs)

```
using System.ComponentModel.DataAnnotations;
namespace PRACTICAL11.Models {
  public class Product {
    public int Id { get; set; }
    [Required]
    public required string Name { get; set; }
    public decimal Price { get; set; }
    public int CategoryId { get; set; }
    public Category? Category { get; set; }
    public int SupplierId { get; set; }
    public Supplier? Supplier { get; set; }
  }
}
```

Category.cs : (Models / Category.cs)
```
using System.Collections.Generic;
namespace PRACTICAL11.Models {
   public class Category {
      public int Id { get; set; }
      public required string Name { get; set; }
      public ICollection<Product>? Products { get; set; }
   }
}
```

Supplier.cs : (Models / Supplier.cs)
```
using System.ComponentModel.DataAnnotations;
namespace PRACTICAL11.Models {
   public class Supplier {
      public int Id { get; set; }
      [Required]
      public required string Name { get; set; } = string.Empty;
      [Required]
      public required string Contact { get; set; } = string.Empty;
      public Supplier() { }
   }
}
```

AppDbContext.cs : (Data / AppDbContext.cs)
```
using Microsoft.EntityFrameworkCore;
using PRACTICAL11.Models;
namespace PRACTICAL11.Data {
   public class AppDbContext : DbContext {
      public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }
      public DbSet<Product> Products { get; set; }
      public DbSet<Category> Categories { get; set; }
      public DbSet<Supplier> Suppliers { get; set; }
   }
}
```

AppDbContextFactory.cs : (Data / AppDbContextFactory.cs)
```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Design;
using Microsoft.Extensions.Configuration;
using System.IO;
namespace PRACTICAL11.Data {
   public class AppDbContextFactory : IDesignTimeDbContextFactory<AppDbContext> {
      public AppDbContext CreateDbContext(string[] args) {
         var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();
```

```
            var configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("appsettings.json")
                .Build();
            var connectionString = configuration.GetConnectionString("DefaultConnection");
            if (string.IsNullOrEmpty(connectionString)) {
                throw new InvalidOperationException("Connection string cannot be null or
empty.");
            }
            optionsBuilder.UseMySql(connectionString,
ServerVersion.AutoDetect(connectionString));
            return new AppDbContext(optionsBuilder.Options);
        }
    }
}
```

HomeController.cs : (Controllers / HomeController.cs)

```
using Microsoft.AspNetCore.Mvc;
namespace PRACTICAL11.Controllers {
    public class HomeController : Controller {
        public IActionResult Index() {
            return View();
        }
    }
}
```

ProductController.cs : (Controllers / ProductController.cs)

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using PRACTICAL11.Data;
using PRACTICAL11.Models;
using System.Threading.Tasks;
namespace PRACTICAL11.Controllers {
    public class ProductController : Controller {
        private readonly AppDbContext _context;
        public ProductController(AppDbContext context) {
            _context = context;
        }
        public async Task<IActionResult> Index() {
            var products = await _context.Products
                .Include(p => p.Category)
                .Include(p => p.Supplier)
                .ToListAsync();
            return View(products);
        }
        public IActionResult Create() {
            ViewBag.Categories = _context.Categories.ToList();
            ViewBag.Suppliers = _context.Suppliers.ToList();
            return View();
        }
        [HttpPost]
        public async Task<IActionResult> Create(Product product) {
            if (ModelState.IsValid) {
                _context.Add(product);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            ViewBag.Categories = _context.Categories.ToList();
            ViewBag.Suppliers = _context.Suppliers.ToList();
            return View(product);
        }
    }
}
```

CategoryController.cs : (Controllers / CategoryController.cs)

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using PRACTICAL11.Data;
using PRACTICAL11.Models;
using System.Threading.Tasks;
namespace PRACTICAL11.Controllers {
    public class CategoryController : Controller {
        private readonly AppDbContext _context;
        public CategoryController(AppDbContext context) {
            _context = context;
        }
        public async Task<IActionResult> Index() {
            var categories = await _context.Categories.ToListAsync();
            return View(categories);
        }
        public IActionResult Create() {
            return View();
        }
        [HttpPost]
        public async Task<IActionResult> Create(Category category) {
            if (ModelState.IsValid) {
                _context.Add(category);
                await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
         }
         return View(category);
      }
   }
}
```

SupplierController.cs : (Controllers / SupplierController.cs)

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using PRACTICAL11.Data;
using PRACTICAL11.Models;
using System.Threading.Tasks;
namespace PRACTICAL11.Controllers {
   public class SupplierController : Controller {
      private readonly AppDbContext _context;
      public SupplierController(AppDbContext context) {
         _context = context;
      }
      public async Task<IActionResult> Index() {
         var suppliers = await _context.Suppliers.ToListAsync();
         return View(suppliers);
      }
      public IActionResult Create() {
         var supplier = new Supplier {
            Name = string.Empty,
            Contact = string.Empty
         };
         return View(supplier);
      }
      [HttpPost]
      [ValidateAntiForgeryToken]
      public async Task<IActionResult> Create([Bind("Name,Contact")] Supplier supplier) {
         if (ModelState.IsValid) {
            _context.Add(supplier);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
         }
         return View(supplier);
      }
   }
}
```

Add Migrations and Update Database :

```
>> dotnet ef migrations add InitialCreate
>> dotnet ef database update
```

Index.cshtml : (Views / Home / Index.cshtml)

```
@{
    ViewData["Title"] = "Home";
}
<center><br><h2>Welcome To The Store</h2>
<p><br>
    <a asp-controller="Product" asp-action="Index" class="btn btn-primary">Manage
Products</a>
    <a asp-controller="Supplier" asp-action="Index" class="btn btn-primary">Manage
Suppliers</a>
    <a asp-controller="Category" asp-action="Index" class="btn btn-primary">Manage
Categories</a>
</p></center>
```

Index.cshtml : (Views / Product / Index.cshtml)

```
@model IEnumerable<PRACTICAL11.Models.Product>
@{
    ViewData["Title"] = "Products";
}
<h2>Products</h2>
<a asp-action="Create" class="btn btn-primary">Create New Product</a>
<table class="table">
    <thead>
        <tr>
            <th>Name</th>
            <th>Price</th>
            <th>Category</th>
            <th>Supplier</th>
        </tr>
    </thead>
    <tbody>
        @if (Model != null && Model.Any())
        {
            foreach (var product in Model)
            {
                <tr>
                    <td>@product.Name</td>
                    <td>@product.Price.ToString("C")</td>
                    <td>@(product.Category?.Name ?? "N/A")</td>
                    <td>@(product.Supplier?.Name ?? "N/A")</td>
                </tr>
            }
        }
        else
        {
            <tr>
                <td colspan="4" class="text-center">No products found.</td>
            </tr>
        }
    </tbody>
</table>
@model IEnumerable<PRACTICAL11.Models.Product>
@{
    ViewData["Title"] = "Products";
}
<h2>Products</h2>
<a asp-action="Create" class="btn btn-primary">Create New Product</a>
<table class="table">
    <thead>
        <tr>
            <th>Name</th>
            <th>Price</th>
            <th>Category</th>
            <th>Supplier</th>
        </tr>
    </thead>
    <tbody>
        @if (Model != null && Model.Any()) {
            foreach (var product in Model) {
                <tr>
                    <td>@product.Name</td>
                    <td>@product.Price.ToString("C")</td>
                    <td>@(product.Category?.Name ?? "N/A")</td>
                    <td>@(product.Supplier?.Name ?? "N/A")</td>
                </tr>
            }
        } else {
            <tr>
                <td colspan="4" class="text-center">No products found.</td>
            </tr>
        }
    </tbody>
</table>
```

Index.cshtml : (Views / Category / Index.cshtml)

```
@model IEnumerable<PRACTICAL11.Models.Category>
@{
   ViewData["Title"] = "Categories";
}
<h2>Categories</h2>
<a asp-action="Create" class="btn btn-primary">Create New Category</a>
<table class="table">
   <thead>
      <tr>
         <th>Name</th>
      </tr>
   </thead>
   <tbody>
      @if (Model != null && Model.Any()) {
         foreach (var category in Model) {
            <tr>
               <td>@category.Name</td>
            </tr>
         }
      } else {
         <tr>
            <td colspan="1" class="text-center">No categories found.</td>
         </tr>
      }
   </tbody>
</table>
```

Index.cshtml : (Views / Supplier / Index.cshtml)

```
@model IEnumerable<PRACTICAL11.Models.Supplier>
@{
   ViewData["Title"] = "Suppliers";
}
<h2>Suppliers</h2>
<a asp-action="Create" class="btn btn-primary">Create New Supplier</a>
<table class="table">
   <thead>
      <tr>
         <th>Name</th>
      </tr>
   </thead>
   <tbody>
      @if (Model != null && Model.Any()) {
         foreach (var supplier in Model) {
                <tr>
                    <td>@supplier.Name</td>
                </tr>
            }
        } else {
            <tr>
                <td colspan="2" class="text-center">No suppliers found.</td>
            </tr>
        }
    </tbody>
</table>
```

Create.cshtml : (Views / Product / Create.cshtml)

```
@model PRACTICAL11.Models.Product
@{
    ViewData["Title"] = "Create Product";
}
<h2>Create Product</h2>
<form asp-action="Create" method="post">
    <div class="form-group">
        <label asp-for="Name" class="control-label"></label>
        <input asp-for="Name" class="form-control" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Price" class="control-label"></label>
        <input asp-for="Price" class="form-control" />
        <span asp-validation-for="Price" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="CategoryId" class="control-label">Category</label>
        <select asp-for="CategoryId" class="form-control" asp-items="@(new
SelectList(ViewBag.Categories, "Id", "Name"))"></select>
    </div>
    <div class="form-group">
        <label asp-for="SupplierId" class="control-label">Supplier</label>
        <select asp-for="SupplierId" class="form-control" asp-items="@(new
SelectList(ViewBag.Suppliers, "Id", "Name"))"></select>
    </div>
    <button type="submit" class="btn btn-primary">Create</button>
</form>
<a asp-action="Index">Back to List</a>
```

Create.cshtml : (Views / Category / Create.cshtml)

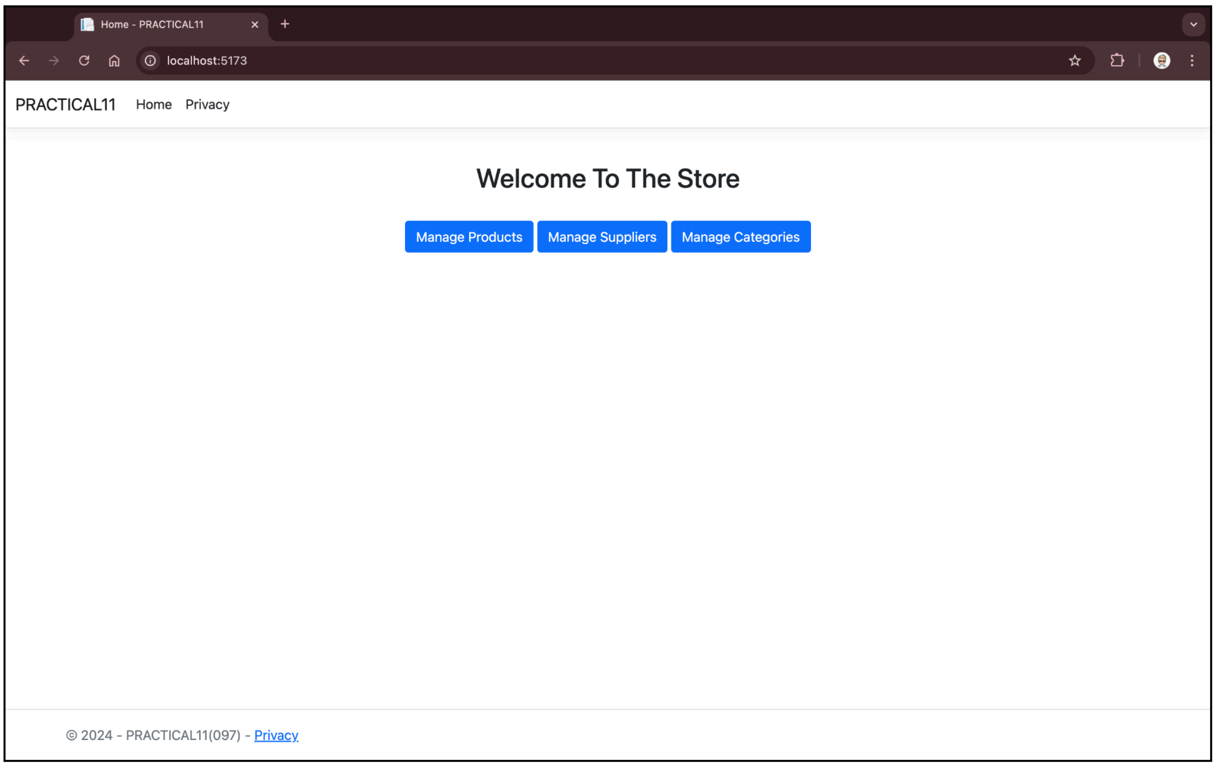
```
@model PRACTICAL11.Models.Category
@{
   ViewData["Title"] = "Create Category";
}
<h2>Create Category</h2>
<form asp-action="Create" method="post">
   <div class="form-group">
      <label asp-for="Name" class="control-label"></label>
      <input asp-for="Name" class="form-control" />
      <span asp-validation-for="Name" class="text-danger"></span>
   </div>
   <button type="submit" class="btn btn-primary">Create</button>
</form>
<a asp-action="Index">Back to List</a>
```

Create.cshtml : (Views / Supplier / Create.cshtml)

```
@model PRACTICAL11.Models.Supplier
<h1>Create Supplier</h1>
<form asp-action="Create">
   <div class="form-group">
      <label asp-for="Name"></label>
      <input asp-for="Name" class="form-control" />
      <span asp-validation-for="Name" class="text-danger"></span>
   </div>
   <div class="form-group">
      <label asp-for="Contact"></label>
      <input asp-for="Contact" class="form-control" />
      <span asp-validation-for="Contact" class="text-danger"></span>
   </div>
   <button type="submit" class="btn btn-primary">Create</button>
</form>
<a asp-action="Index">Back to List</a>
<script>
   const form = document.querySelector('form');
   form.addEventListener('submit', function(event) { });
</script>
```

Enrollment No: 202203103510097

Output:


Home - PRACTICAL11

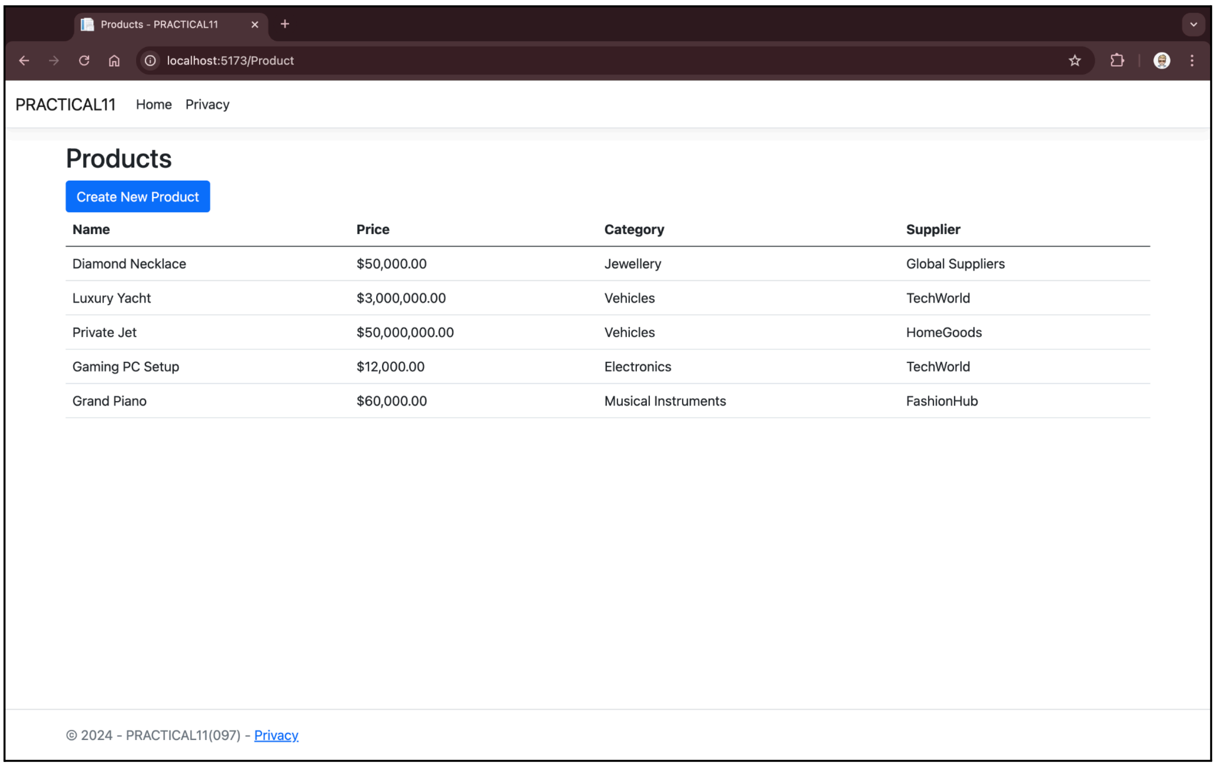
localhost:5173

PRACTICAL11 Home Privacy

Welcome To The Store

Manage Products Manage Suppliers Manage Categories

© 2024 - PRACTICAL11(097) - Privacy


Products - PRACTICAL11

localhost:5173/Product

PRACTICAL11 Home Privacy

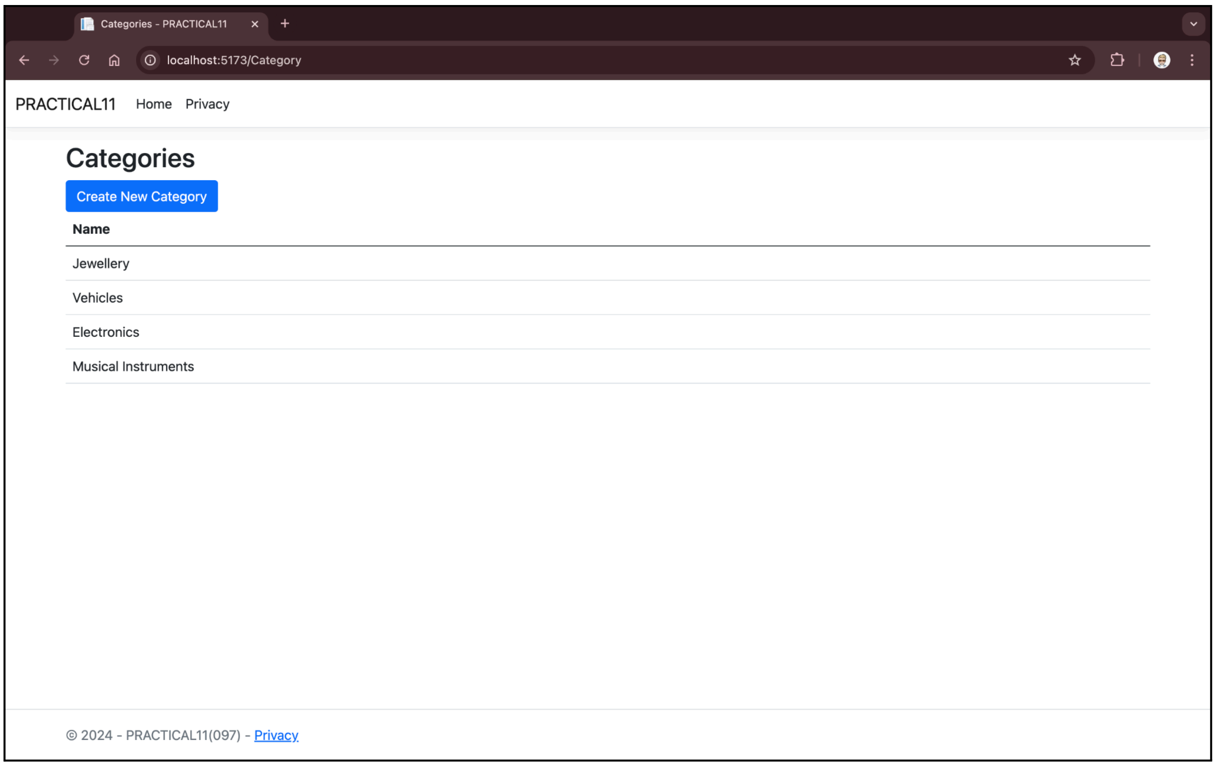
Products

Create New Product

Name	Price	Category	Supplier
Diamond Necklace	$50,000.00	Jewellery	Global Suppliers
Luxury Yacht	$3,000,000.00	Vehicles	TechWorld
Private Jet	$50,000,000.00	Vehicles	HomeGoods
Gaming PC Setup	$12,000.00	Electronics	TechWorld
Grand Piano	$60,000.00	Musical Instruments	FashionHub

© 2024 - PRACTICAL11(097) - Privacy

Enrollment No: 202203103510097


Categories - PRACTICAL11

localhost:5173/Category

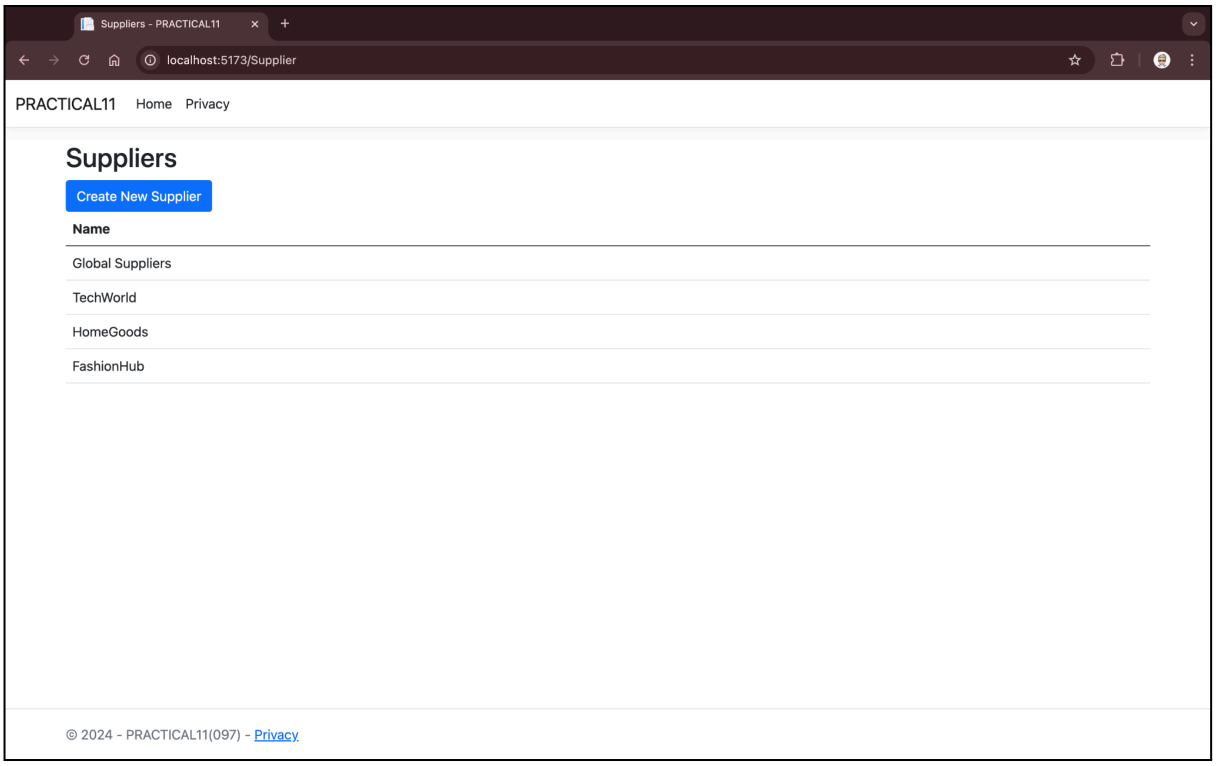
PRACTICAL11 Home Privacy

Categories

Create New Category

Name
Jewellery
Vehicles
Electronics
Musical Instruments

© 2024 - PRACTICAL11(097) - Privacy


Suppliers - PRACTICAL11

localhost:5173/Supplier

PRACTICAL11 Home Privacy

Suppliers

Create New Supplier

Name
Global Suppliers
TechWorld
HomeGoods
FashionHub

© 2024 - PRACTICAL11(097) - Privacy