Appsettings.json

```
"ConnectionStrings": {
    "DevConnection": "server=localhost;user
id=root;password=;database=kct; charset=utf8; default command
timeout=0"
  }
```

Program.cs

```
builder.Services.AddDbContext<DataDbContext>(option =>
{

option.UseMySQL(builder.Configuration.GetConnectionString("DevConn
ection"));
});
```

Models

Users.cs

```
using Microsoft.AspNetCore.Mvc;

namespace AspCoreMVCEF.Models
{
    [BindProperties]
    public class User
    {
        public int id { get; set; }
        public string username { get; set; }
        public string password { get; set; }
        public string email { get; set; }

    }
}
```

DataDbContext.cs

```
using Microsoft.EntityFrameworkCore;

namespace AspCoreMVCEF.Models
{
    public class DataDbContext : DbContext
    {
        public DataDbContext(DbContextOptions options) :
base(options)
        {
        }
        public DbSet<User> Users { get; set; }

    }
}
```

Controllers

UsersController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using AspCoreMVCEF.Models;

namespace AspCoreMVCEF.Controllers
{
    public class UsersController : Controller
    {
        private readonly DataDbContext _context;

        public UsersController(DataDbContext context)
        {
            _context = context;
        }

        // GET: Users
        // This enables the action methods to perform asynchronous
operations,
        // like database calls or web service requests, without
blocking the calling thread
        public async Task<IActionResult> Index()
        {
              return _context.Users != null ?
                          View(await _context.Users.ToListAsync())
:
                          Problem("Entity set
'DataDbContext.Users'  is null.");
        }

        // GET: Users/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Users == null)
            {
                return NotFound();
            }

            var user = await _context.Users
                .FirstOrDefaultAsync(m => m.id == id);
            if (user == null)
            {
                return NotFound();
            }

            return View(user);
        }

        // GET: Users/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Users/Create
        // To protect from overposting attacks, enable the
specific properties you want to bind to.
        // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create(User user)
        {
            if (ModelState.IsValid)
            {
                _context.Add(user);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(user);
        }

        // GET: Users/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null || _context.Users == null)
            {
                return NotFound();
            }

            var user = await _context.Users.FindAsync(id);
            if (user == null)
            {
                return NotFound();
            }
            return View(user);
        }

        // POST: Users/Edit/5
        // To protect from overposting attacks, enable the
specific properties you want to bind to.
        // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, User user)
        {
            if (id != user.id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(user);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!UserExists(user.id))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            return View(user);
        }
        // GET: Users/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null || _context.Users == null)
            {
                return NotFound();
            }

            var user = await _context.Users
                .FirstOrDefaultAsync(m => m.id == id);
            if (user == null)
            {
                return NotFound();
            }

            return View(user);
        }

        // POST: Users/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.Users == null)
            {
                return Problem("Entity set 'DataDbContext.Users'
is null.");
            }
            var user = await _context.Users.FindAsync(id);
            if (user != null)
            {
                _context.Users.Remove(user);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool UserExists(int id)
        {
          return (_context.Users?.Any(e => e.id ==
id)).GetValueOrDefault();
        }
    }
}
```

Views>Users

Create.cshtml

```
@model AspCoreMVCEF.Models.User

@{
    ViewData["Title"] = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Create</h1>

<h4>User</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-
danger"></div>
            <div class="form-group">
                <label asp-for="username" class="control-
label"></label>
                <input asp-for="username" class="form-control" />
                <span asp-validation-for="username" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="password" class="control-
label"></label>
                <input asp-for="password" class="form-control" />
                <span asp-validation-for="password" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="email" class="control-
label"></label>
                <input asp-for="email" class="form-control" />
                <span asp-validation-for="email" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn
btn-primary" />
            </div>
        </form>
    </div>
```

```
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Delete.cshtml

```
@model AspCoreMVCEF.Models.User

@{
    ViewData["Title"] = "Delete";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>User</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.username)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.username)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.password)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.password)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.email)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.email)
        </dd>
    </dl>
    <form asp-action="Delete">
        <input type="hidden" asp-for="id" />
        <input type="submit" value="Delete" class="btn btn-danger"
/> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
```

Details.cshtml

```
@model AspCoreMVCEF.Models.User

@{
    ViewData["Title"] = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Details</h1>

<div>
    <h4>User</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.username)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.username)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.password)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.password)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.email)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.email)
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model?.id">Edit</a> |
    <a asp-action="Index">Back to List</a>
```

```
</div>
```

Edit.cshtml

```
@model AspCoreMVCEF.Models.User

@{
    ViewData["Title"] = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Edit</h1>

<h4>User</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-
danger"></div>
            <input type="hidden" asp-for="id" />
            <div class="form-group">
                <label asp-for="username" class="control-
label"></label>
                <input asp-for="username" class="form-control" />
                <span asp-validation-for="username" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="password" class="control-
label"></label>
                <input asp-for="password" class="form-control" />
                <span asp-validation-for="password" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="email" class="control-
label"></label>
                <input asp-for="email" class="form-control" />
                <span asp-validation-for="email" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-
primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Index.cshtml

```
@model IEnumerable<AspCoreMVCEF.Models.User>

@{
    ViewData["Title"] = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Index</h1>

<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.username)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.password)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.email)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.username)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.password)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.email)
            </td>
            <td>
                <a asp-action="Edit" asp-route-
id="@item.id">Edit</a> |
                <a asp-action="Details" asp-route-
id="@item.id">Details</a> |
                <a asp-action="Delete" asp-route-
id="@item.id">Delete</a>
            </td>
        </tr>
}
    </tbody>
</table>
```