ASP.NET CORE

- **Day – 1**
  - Building a walking skeleton – **2hr**
    - Creating the .Net API Project using the dotnet CLI
    - Setting up VS code to work with C#
    - VS Code tips and troubleshooting
    - Getting to know the API project files
    - Creating our first Entity
    - Introduction to Entity Framework
    - Adding Entity Framework to our project
    - Adding a DbContext class
    - Creating the Connection string
    - Generate the scripts using database first approch
    - Adding a new API Controller
    - Making our code Asynchronous
    - Saving our code into Source control
  - Introduction to building ASP.NET Core Web APIs using .NET6 - **15 minutes**
    - Introduction to building ASP.NET Core Web APIs using .NET8.
    - Setting up development environment.
    - Install .NET SDK and .NET Runtime.
    - Install SQL Server Management Studio.
  - Create New ASP.NET Core Web API & Domain Model – **1hr**
    - Create New ASPNET Core Web API
    - Understand ASP.NET Core Web API and File Structure
    - Understand REST and HTTP Verbs
    - Routing In ASP.NET Core Web API
    - Running and Calling ASP.NET Core Web API
    - Understanding Our Domain (Domain Driven Development)
    - Create Domain Models
    - Adding Entity Framework Core Packages
    - Creating the DBContext Class
    - Adding ConnectionString To The Database In Appsettings.Json
    - Understanding Dependency Injection & Injecting DbContext Into Our Application
    - Run EF Core Migrations

- **Day - 2**
    - Create New Controller - Regions Controller and CRUD Operations / Action Methods - **1hr.**
        - Create New Controller (Regions Controller)
        - Get All Regions Action Method
        - Get Region By Id Action Method
        - DTOs and Domain Models
        - Change Methods To Use DTOs
        - Create Region Action Method
        - Update Region Action Method
        - Delete Region Action Method
    - Asynchronous Programming, Repository Pattern and Automapper - **1hr**
        - Asynchronous Programming and Implement Async Await In our Methods
        - Repository Pattern and Benefits
        - Implement Repository Pattern In ASP.NET Core Web APIs
        - Add Other Methods To Repository
        - Automapper Introduction
        - Implement Automapper
    - Functionality For Walks - Create, Read, Update, Delete for Walks - **1hr**
        - Seeding Data Using Entity Framework Core
        - Creating Walks Controller and Create New Walk Action Method
        - Get All Walks Action Method
        - Navigation Properties in Entity Framework Core
        - Get Walk by Id Action Method
        - Update Walk Action Method
        - Delete Walk Action Method
- **Day - 3**
    - Validations in ASP.NET Core Web API - **15 minutes**
        - Introduction To Model Validations
        - Adding Model Validations to Endpoints
        - Custom Validate Model Attribute
    - Filtering, Sorting, Pagination in ASP.NET Core Web API - **45 minutes**
        - Seeding Data in SQL Server
        - Filtering
        - Sorting
        - Pagination

- Authentication and Authorization - JWT Tokens - **1hr 45 minutes**
  - Authentication Flow
  - Setting Up Authentication - Install Nuget Packages
  - Testing Authentication Without JWT Token
  - Setting Up Identity DbContext
  - Seeding Roles
  - Run EF Core Migrations To Create Identity Database
  - Setting Up Identity
  - Create Auth Controller and Register Action Method
  - Create Login Action Method
  - Create JWT Token in ASP.NET Core Web API
  - Inject TokenRepository and Create Token
  - Role Based Authorization
  - Add Authorization to Swagger

- **Day – 4**
  - Advanced Functionality in ASP.NET Core Web APIs - **30 minutes**
    - Add Logging to Console
    - Add Logging to Text File
    - Error handling
      - Creating an error controller for testing errors
      - Handling server errors
      - Exception handling middleware
      - Validation errors
      - Handling not found.
      - Adding a server error page
    - Global Exception Handling
    - What is Versioning and How to Implement Versioning
    - Implement Versioning by Folder Structure
    - Implement Versioning by Nuget Package - Proffered Approach
    - Fix Versioning in Swagger
  - Consuming our Web API - **1hr**
    - Consuming REST Web APIs
    - Creating ASP.NET MVC Web Application
    - GET - Consume Web API
    - POST - Consume Web API
    - GET single Region - Edit Region Prerequisite
    - PUT - Updating Using Web API

        - DELETE - Deleting a Resource using Web API

- **Day – 5**
    - Identity and role management - **2hr**
        - Setting up the entities.
        - Configuring the DbContext.
        - Configuring the startup class.
        - Refactoring and adding a new migration.
        - Updating the seed method.
        - Updating the account controller.
        - Adding roles to the app.
        - Adding the roles to the JWT token.
        - Adding policy based authorization.
        - Getting the users with roles.
        - Editing user roles.
    - SignalR - **2hr**
        - Adding a presence hub.
        - Authenticating to SignalR.
        - Client side SignalR.
        - Adding a presence tracker.
        - Displaying online presence.
        - Creating a message hub.
        - Adding the send message method to the hub.
        - Adding the hub connection to the message service.
        - Refactoring the message components to use the hub.
        - Sending messages via the hub.
        - Tracking the message groups.
        - Updating the message hub with group tracking.
        - Dealing with UTC date formats.
        - Notifying users when they receive a message.
        - Optimizing the presence.
        - Optimizing the messages.
- **Day – 6**
    - Publishing - **2hr 30 minutes**
        - Preparing the angular app and serving this from the API server
        - Adding a fallback controller
        - Creating an angular production build
        - Changing the DB Server in our app

- Dockerizing our app.
- Updating the config to use database.
- Creating a deployment to fly io.
- Creating the config variables for fly io
- Using github actions
- Fixing the issues with the message thread
- Automating the deployment
- Deploying ASP.NET Web API To Azure (Optional)