# Homework: ASP.NET MVC Introduction

This document defines the homework assignments from the ["ASP.NET MVC" Course @ Software University](https://softuni.bg/courses/asp-net-mvc/). Please submit as homework a single zip / rar / 7z archive holding the solutions (source code) of all below described problems.

## Long-Term Task: Implement a Twitter-Like Web Application

Your task is to implement **Twitter Web application** during the entire course. The homework for all lectures will be parts of the application. Give all best from you, so you can use the application as part of your CV.

* Users
  + Users have followers and list of following users.
  + Every user can follow the others and can be followed by the others.
  + Users may post new tweets.
  + Users can edit their own profiles.
  + Users can delete their tweets.
  + Users can send messages to the other users.
* Tweets
  + Every tweet can be favourite by any user.
  + Every tweet can be retweeted by another user.
  + Every tweet can be reported.
  + Every tweet should have its page (URL).
  + Every tweet can be shared via Facebook.
  + Every user can reply to a tweet by new tweet.
* Users public profiles
  + Each user should have a public profile.
  + Profile have own tweets, following users, followers and favourite tweets.
* Application home page
  + The application home page should list all tweets by all users chronologically.
* User home page
  + The user home page should display all tweets created by the followed users.
* Users have notifications
  + Users receive notifications for retweet, favourite tweet and for new followers.
  + The notification holds content, date and user.

## Prepare for the Application

Your task is to think about the project structure and make empty Visual Studio projects for all application layers.

* Create empty solution with the name of your application (e.g. Twitter).
* Create “**Data**” folder to hold the projects for your **data layer** (e.g. **Data**, **Models**, **Contracts**, **Repositories**, …)
* Create “**Web**” folder to hold the projects for the Web layer (e.g. MVC application, common application, …).
* Create “**Tests**” folder to hold the projects for testing the different layers (e.g. Web Tests, Data Tests, …).
* Create “**External Libraries**” folder where you should put all external **.dll** files.

## Create the Data Layer

Your task is to implement the data model classes for the database. Use Entity Framework Code First.

* Make Class Library projects in the Data folder:
  + Twitter.Models project for all models.
  + Twitter.Data project for the data context, repositories and unit of work.
* Create your data model classes in the Twitter.Models project.
* Create your data context class in the Twitter.Data project.

## \* Implement Repository Pattern and Unit of Work

* Implement the **Repository** pattern over the data layer.
* Implement the **Unit of Work** patternto unify your repositories.

## \* Install Ninject and Map the Interfaces to Implementations

Your task is to install **Ninject** (or any other Inversion of Control container) through **NuGet** package manager and map the interfaces to the implementations. For example, map the ITwitterData interface to your data TwitterData class, and ITwitterDbContext interface to TwitterDbContext.

The recommended version is Ninject.Mvc5. Bind mappings in the RegisterServices method in the NinjectWebCommon class in the App\_Start folder.

## Create the ASP.NET MVC Project

Your task is to create ASP.NET MVC project using the Visual Studio Web application wizard. Move the generated DbContext class in your data layer and the ApplicationUser class in your models. Map everything to work with your new database schema.

Create a BaseController class. It will be inherited by all your controllers. BaseController should have ITwitterData field with protected property only with getter. The BaseController should have only one constructor with one parameter (ITwitterData). Every controller should have two constructors. The first constructor takes ITwitterData and calls the base constructor. The second constructor makes **poor man’s injection** via default constructor.

When you register a new user, your Entity Framework code first classes should populate the database schema in the SQL Server.

## Install Glimpse

Install Glimpse Mvc5 and Glimpse EF6. Play with the extension and investigate what is the benefit of the Glimpse.

## Create Empty Controllers for Your Main Actions

Create empty MVC controllers for the basic actions in your application.

## Play with Controllers and Views

Your task is to write some actions. Play with different ActionResults. Play with different views:

* Play with views
* Play with layouts
* Play with partial views
* Research the concept "display template" and try to use it