# Technical task list

This is a technical guide document to run the Manage Expense project. During reading this document if you encounter any question feel free to ask me through [m.hoseinalizadeh@gmail.com](mailto:m.hoseinalizadeh@gmail.com)

You can find the source code of this project in this path: <https://github.com/mhosseinalizadeh/AspNetTemplate>

## Requirements

1. Microsoft Visual studio 2017 or higher
2. Microsoft SQL Server 2012 or higher
3. .Net Core SDK 2.2.108 for Visual studio 2017 or latest version of .Net Core SDK for Visual studio 2019

## Steps to run

1. Restore the SQL Server database backup name "Database.bak". You can find this file in the root folder of the project, next to the "AspNetTemplate.sln" file or in this URL <https://github.com/mhosseinalizadeh/AspNetTemplate/blob/master/Database.bak>
2. Make sure you installed .Net Core SDK 2.2.108 for Visual studio 2017 or latest version of .Net Core SDK for Visual studio 2019 on your machine.
3. Open "AspNetTemplate.sln" file in Visual studio.
4. Open the "appsettings.json" file and change DefaultConnection value to a proper connection string and save it. Please make sure the connection string points to restored database in step 1.
5. Because we send email to users in some scenarios, we need to have access to the internet on the machine which this project is running on it. (If you have a local Mail server, you can configure MailSettings in section AppSettings in "appsettings.json" file. So after that we don’t need have access to the internet)
6. Press f5 button in Visual studio to run the project.

## Some Technical Notes

This project has MVC and N Layered architecture and. Presentation layer, Application Service Layer and Repository Layer are the main layers of the application.

In presentation layer I used ASP.Net core Razor view. This layer is responsible to render views to the end user.

In Application Service layer we have 2 project (Application Service and Common Service). All the logic of our business implemented in this layer. Business functionalities have been implemented to Application Service Project and general functionalities such as sending email, encryption and caching have been implemented in Common Service project.

For client entities and Data Transfer Objects (DTO) I created a separate project, named Client Entity. And for domain entities which are map to the database I created another project, named Domain Entity.

Repositories have been implemented in Data Access project and are responsible to CRUD operations.

Also I created a Unit Test project, named Test. It has only two unit test currently and code coverage for all solution is not 100 percent. But for the "CheckPassword" method code coverage is 100%.

## Tools and Technologies

* .NET Core as main framework
* Dapper.NET as ORM
* MVC and N Layered architecture
* SQL Server as database
* Serilog as message logging system
* MailKit for sending email
* Cookie Authentication for user authenticating
* .Net Core MemoryCache to implement caching system
* Bootstrap as CSS Framework
* Datatables jQuery Plugin for showing grid
* Using Repository and Unit Of Work pattern
* Try to use SOLID principles(especially Single Responsibility, Open/Closed and Dependency Injection)
* Try to use Clean Code principles such as meaningful name for variables, methods and class names, indenting, smaller methods and …
* Unit testing