# Vakil Uncle Project – Angular

When I am running the app on visual studio it is running the previous original code but when I am running from angular it is using the updated code.

What I feel it is using index.html inside browser folder which is residing in the HOMEPage folder which is at root level which is using app-root component in visual studio , currently strange for me. I fixed the issue in angular project hence it is visible only when I run that but when i run in visual studio then the issue persist.

Seeing this error everytime new components or page loads: Unable to establish SignalR connection.

## Common notes

Calling , chating and loader components are grouped in shared folder.

## Home Page

1. After clicking on run, it lands on Angular index.html 🡪 app-root 🡪 app-loader 🡪

How then it is going to home page. – home page module route is being loaded for path ‘home’ and also for fallback , defined in app-routing.module.ts

1. There is home-page module which is containing many [non-standalone] components in it : advocate details, advocate list,.. till view port.

Where is home-page module is imported, used. How it is coming into picture. Imported inside Routes array not as direct import on the top.

1. In home-page module there is home component which is the actual page, being displayed in the first landed page.

## Lawyer Registration

1. When running on visual studio , it is going to the home page and going to sign up password validation is not working even though I have fixed that but it is working if running on angular one.
2. Registration , otp-verification, login and forgot password is implemented in the auth module.

### Registration flow

1. First we are clicking button submit then it is calling submitRegister() in register component.
2. Which is calling VakilRegister Api which is defined in Authentication controller inside Apiservice folder.
3. That apiservice is getting form data, some specific checks and validations and then it is calling a dataservice method insert advocate defined in DataServices.cs of superadmin model.
4. ADO.NET is used here.

## Communication

WebRTC is used for video call and SignalR for real time communication.

Calling feature is implemented from scratch not using with any api, and the code is written in callingController.

Used Websockets of signalR to implement the chat controller which is written in Chatcontroller

Linq queries are used to filter notifications by dates and user.

## Authentication and Role Authorization

Forms Authentication with role-based authorization is implemented.

Role Authentication class is created by extending role provider and implemented getroleforuser and isuserinrole but in logic they are using Logic service methods inside the method implementation.

[Authorize(Roles = "admin")] is used on the whole SuperAdmin Controller.

Forms authentication is used in checklogin post action method, which is calling check\_login from login services then setting the authentication cookie and if remember me is clicked then expiry time with expiration parameter true and creating the ticket manually and adding in the cookie.

Delegation handler is used to populate requestContext.Principal, because by default form authentication only populates the httpcontext.user but webapi controllers needs requestcontext.principal.

Reason:   
Yes, **Web API** and **MVC** can live in the same ASP.NET project, but inside IIS/ASP.NET Framework they actually run on **two different pipelines**:

* **MVC / WebForms / System.Web pipeline** → works with HttpContext.Current and HttpContext.Current.User.
* **Web API pipeline** → is built on top of HttpMessageHandler and HttpRequestMessage, which use their own abstraction: HttpRequestContext.Principal.
* When you decorate a Web API controller or action with [Authorize], the authorization filter checks RequestContext.Principal, not HttpContext.Current.User.

Even though Web API is running inside the same ASP.NET project:

* MVC requests → handled by System.Web.Mvc → use HttpContext.Current.User.
* API requests → handled by System.Web.Http pipeline → look at RequestContext.Principal.

The cookie is validated by Forms Authentication (part of the System.Web pipeline), which sets HttpContext.Current.User.  
But **Web API does not automatically pick that up**.  
Hence the need for something like FormsAuthDelegatingHandler to bridge the identity between the two worlds.

Both pipelines cooperate, but they don’t automatically share everything. That’s why the user principal needs to be "copied over" if you want Web API controllers to see the authenticated user.

For all keys and connection strings web.config and appsettings files are used.

## Basic workflow of different users

Client flow: registration 🡪 lawyer search 🡪 case creation 🡪 communication 🡪 Payment

Advocate Journey : Registration 🡪 admin approval 🡪 profile setup 🡪 dashboar 🡪 client management and service delivery

Admin process : user verification

## Tables

For tables used at different places, most probably jquery datatable plugin is used.

## Chat Working: