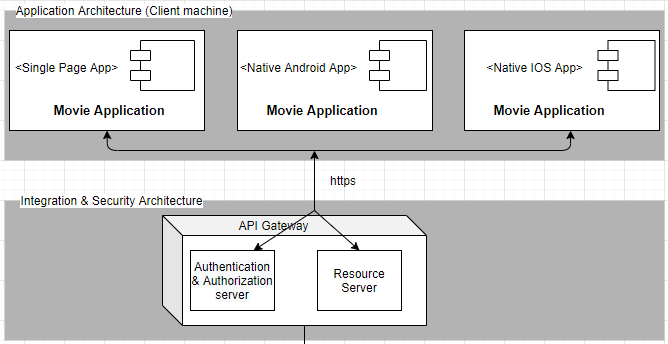
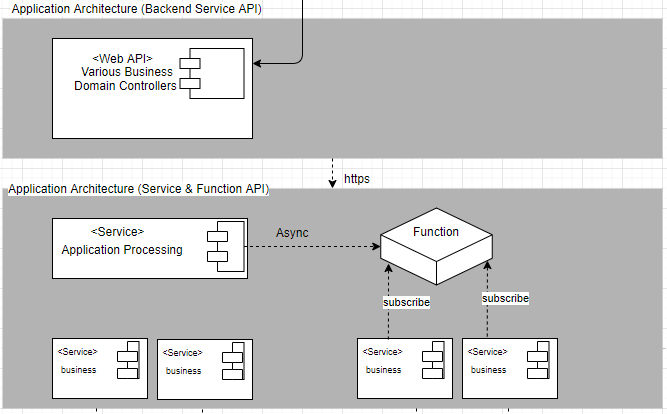
**Task Explanation**

**Project Overview:** This project is just a simple example of online movie site. Here in this project I didn’t introduce any video streaming functionalities. I wanted to give a base backbone of any movie site and based on that a full functional movie site can be built easily. Before giving the explanation of the tasks (asked), I am going to discuss the architecture of the application, flow of the application (how communication can be done between the Front-end and Back-end), and each Front-end and Back-end briefly.

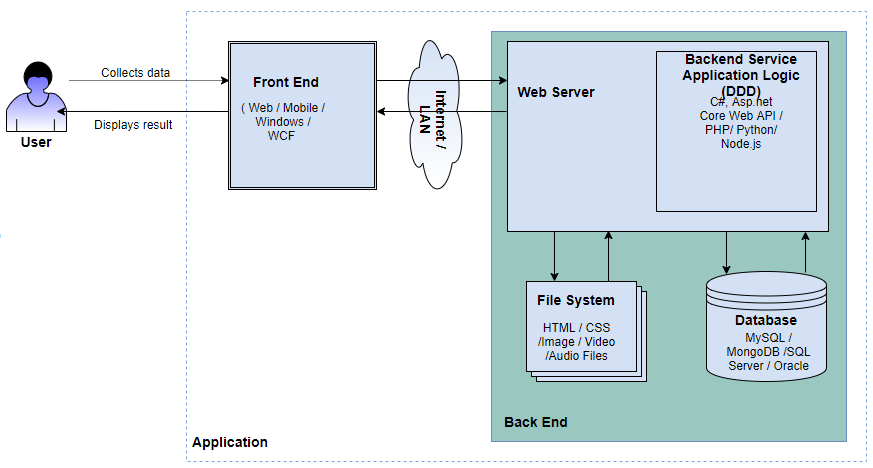
**Project’s Application Architecture:** The application architecture of this project is divided into three; Front-end SPA movie application; Back-end Web API service and database (data architecture). The complete picture of such architecture is depicted bellow.



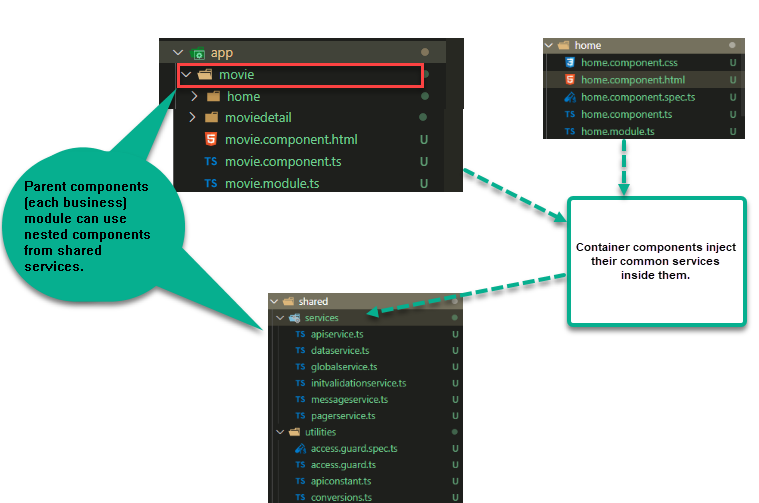




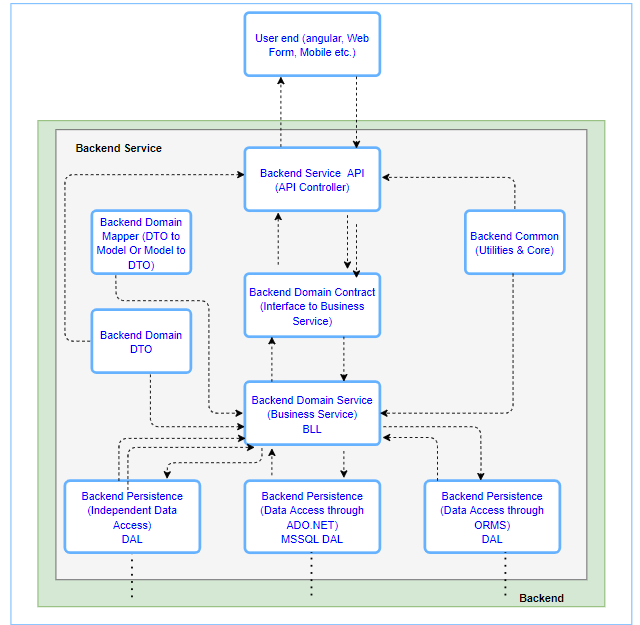
**Overall application’s flow:** The application (Front-end and Back-end) is built based on Domain Driven Design pattern, which is a seed for Microservices architecture. The overall application’s flow is given bellow.



**Front-end Application:**   
The structure of our Front-end application is given below.



**Back-end Service Application (Web API):**   
The structure (Data communication) of our Back-end application is given below. Here I have used ORM Framework Entity Framework .NET core for SQL server. But I have other choices to communicate with the multiple and different databases through traditional ADO.NET objects or only Provider specific ADO.NET objects.



**Task 1 Explanation**

Here in this project, for Authentication I have used JWT, which stands for JSON Web Token. It is just a token format which can be used by OAuth 2.0 protocols. It is open standard and defines a better way for transferring data securely between two entities (client and server). A JWT is digitally signed using a secret key by a token provider or authentication server. A JWT helps the resource server verify the token data using the same secret key, so that we can trust the data. So the approach is given bellow.

Step 1: Web application is requesting token  
The client sends a request (user credential) to the authentication server or API server with the necessary information to prove its identity.

Step 2: Token creation  
The authentication server or API server receives the token request (user credential) and verifies the identity. If it is found valid, a token will be created with the necessary claims, and a JWT token will be sent back to the client application.

Step 3: Client (Front-end Movie web Application) sends token to resource server  
For each request to Resource or the API server, the client needs to include a token in the header and request the resource using its URI.

Step 4: Resource server verifies the token  
I do follow these steps to verify the token:

* Read the token from authentication header.
* Split the header, payload, and signature from token.
* Create signature of received header and payload using the same secret key used when creating the token.
* Check whether both newly created signature and signature received from token are valid.
* If the signatures are the same, the tokens are valid (not altered in the middle) and they provide access to the requested resource.
* If the signatures are different, an unauthorized response will be sent back to the client. (In the middle, if claims are alerted, they will generate a different signature, hence resource access will be restricted.)

**Task 2 Explanation**

Based on Back-end Project (Back-end API)’s response (Boolean value) I am maintaining user’s login or logout status. When first time user logged in, I keep the logged in status as true and assigning to a static variable globally. If user wants to logout then a request sent to Back-end API. API reassigns false value and give the response to the web application to initiate the logout process. In logout function, I am clearing the local storage for saved user information and redirect to login page. User will remain logged in and walk through the different routes in web application, when it is required to logout, the Back-end API will give the clearance. In Front-end, I do use localstorage. But to store list of data in Back-end service during the first call (so that it doesn’t have to communicate with database for every single same request), I love to use Redis cache. But built-in In-memory storage of .NET Core can be used. Here in this project though I haven’t use Redis or In-Memory storage.

**Task 3 Explanation**

I implement a check whether User is logged in or not by checking the global Boolean value (AppUser.AppIsLoggedIn) in every case before executing a certain method.   
  
From the movie web application, when I pass a request (resource, endpoint, http method, http header or authentication header, and data format) to the API server, authentication header is being parsed in the authentication middleware (I have written in startup.cs) and it validates the token. If it is found valid, it will set UIdentity.IsAuthenticated to true. The Authorize attribute added in the controller will check whether the  
request is authenticated. If it is true, the API can be accessed. If UIdentity.IsAuthenticated returns false, a 401 unauthorized error will be returned.