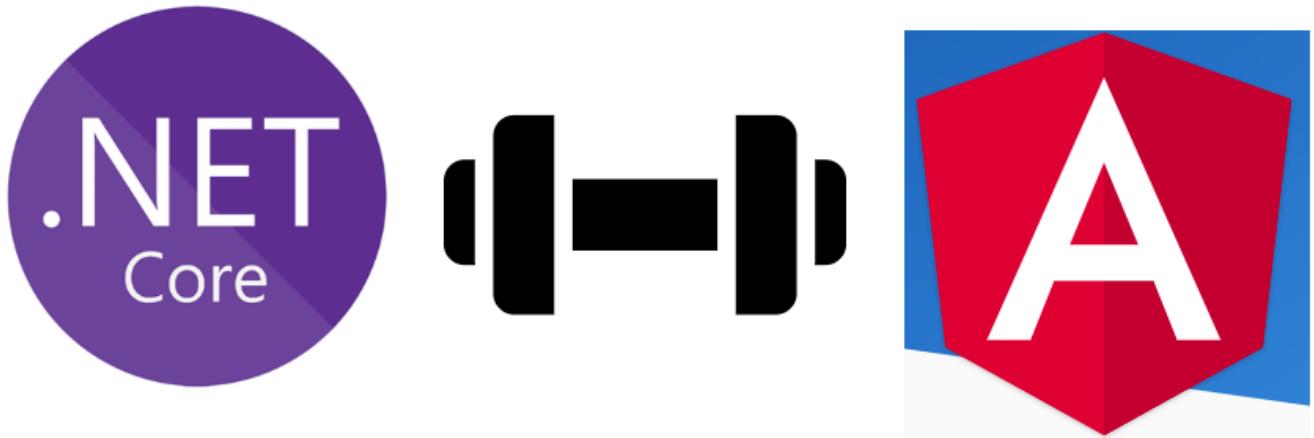


In this article, we are going to learn what this project is about, for whom it is useful and how you can learn from it. In the end, you will get an idea of how angular project and APIS are used in the project and you can create your own project by seeing this project.



This project contains good features which are required in all projects developed in industries nowadays.

Let's see what this project is all about, this project is a basic gym project which has 2 modules in it

1. Admin end
2. User end

### **Admin end**

Let's start with admin end first in this part admin has all rights of applications master such as adding Users, Role, Scheme, Plan, and various reports such as month wise income and year wise income reports, all member reports and also has a renewal report which shows how much renewal are there for period admin choose.

### **User end**

If you see User end where a User is a person who does work of registration of new members and collecting payment of membership. The user has limited access such user can register a new member and renew membership and see payment details of the member along with renewal date.

The project has 3 parts

1. Angular CLI which is on top of node js
2. ASP.NET Core for APIS
3. SQL server for database parts

### **Platform Used**

Angular Version Used

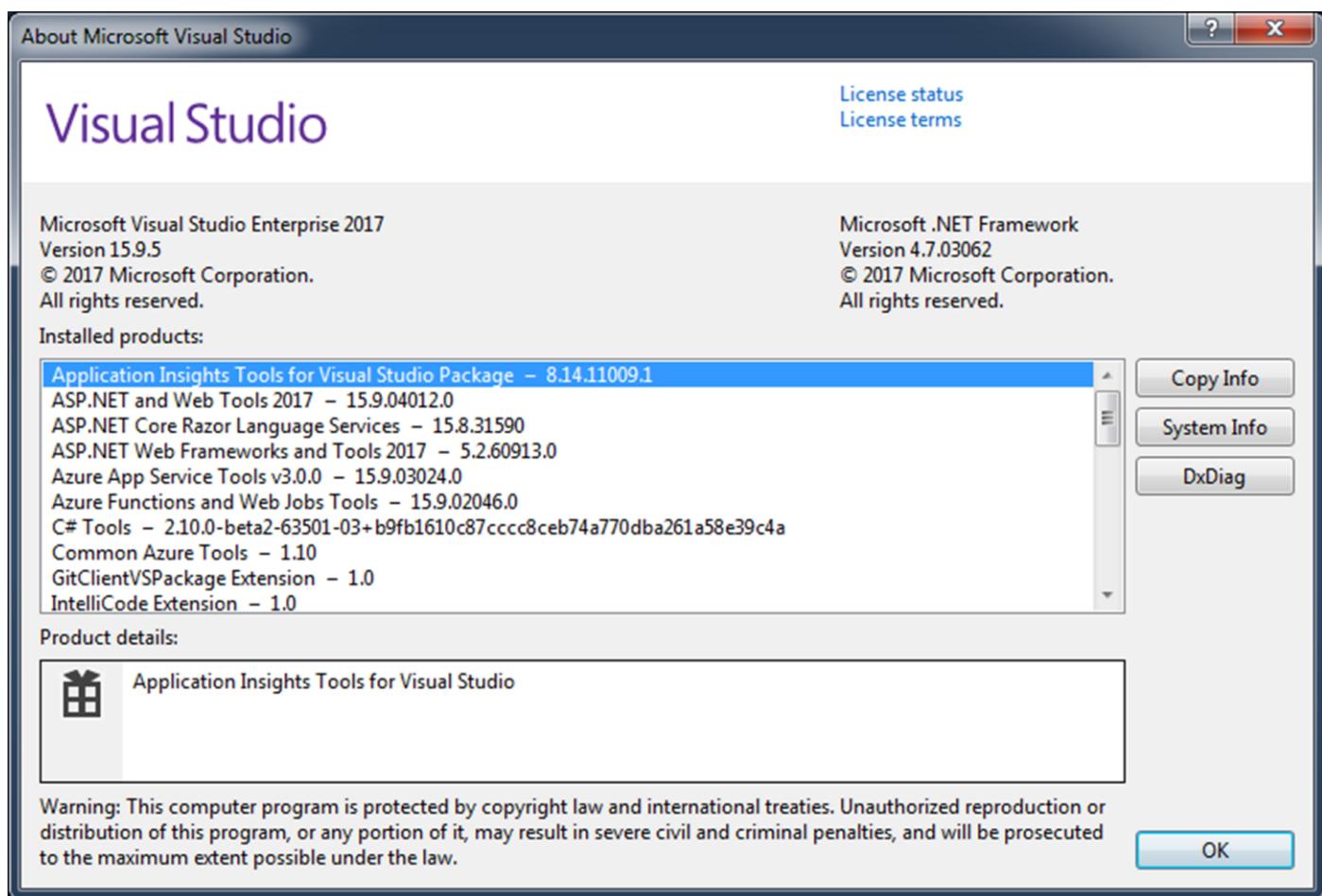


```
Angular CLI: 7.0.6
Node: 8.12.0
OS: win32 x64
Angular: 7.0.4
... animations, common, compiler, compiler-cli, core, forms
... http, language-service, platform-browser
... platform-browser-dynamic, router
```

## Microsoft Visual Studio Community 2017

Link to download Microsoft Visual Studio Community 2017:-

<https://visualstudio.microsoft.com/vs/>



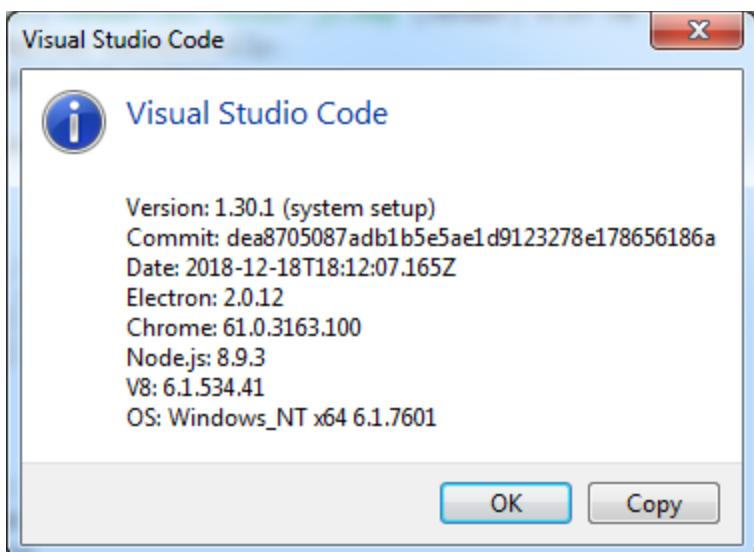
## Microsoft SQL Server 2012

Link to download SQL Server Express: - <https://www.microsoft.com/en-in/download/details.aspx?id=29062>



## Visual Studio Code

Link to download Visual studio code: - <https://code.visualstudio.com/download>



## JWT Token for Authentication of APIS



Image Referenced from: - <https://jwt.io/>

## External packages which are used in .Net Core Project

1. JWT Token for Authentication of APIS
2. Dapper ORM
3. AutoMapper
4. System.Linq.Dynamic.Core

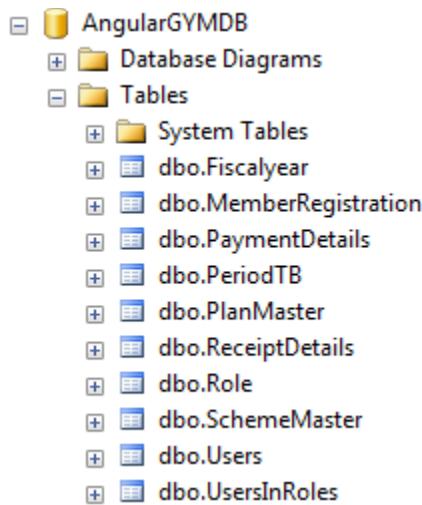
## External packages which are used in Angular Project

1. @angular/material
2. @ngx-bootstrap/datepicker

After completing preparing desk for exploring this project now let's start with database tables first.

## Database

For this application I have used SQL Server database you can download the entire script of this database and use it.



Let's explore tables which are there in this database one by one such that you will have a clear idea which table is used for which purpose.

## Fiscal year

This table contains financial year details.

Column Name	Data Type	Allow Nulls
Fid	int	<input type="checkbox"/>
FiscalyearFromDate	datetime	<input checked="" type="checkbox"/>
FiscalyearToDate	datetime	<input checked="" type="checkbox"/>
Year	varchar(4)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

## Member Details

Whenever new member takes subscription all member details are stored in this database.

## SAI-PC\SQLEXPRESS...emberRegistration X

Column Name	Data Type	Allow Nulls
MemberId	bigint	<input type="checkbox"/>
MemberNo	nvarchar(20)	<input checked="" type="checkbox"/>
MemberFName	nvarchar(100)	<input checked="" type="checkbox"/>
MemberLName	nvarchar(100)	<input checked="" type="checkbox"/>
MemberMName	nvarchar(100)	<input checked="" type="checkbox"/>
DOB	datetime	<input checked="" type="checkbox"/>
Age	nvarchar(10)	<input checked="" type="checkbox"/>
Contactno	nvarchar(10)	<input checked="" type="checkbox"/>
EmailID	nvarchar(30)	<input checked="" type="checkbox"/>
Gender	nvarchar(30)	<input checked="" type="checkbox"/>
PlanID	int	<input checked="" type="checkbox"/>

## Payment Details

Whenever new member takes subscription all Payment details are stored in this database.

## SAI-PC\SQLEXPRES...bo.PaymentDetails X

Column Name	Data Type	Allow Nulls
PaymentID	bigint	<input type="checkbox"/>
PlanID	int	<input checked="" type="checkbox"/>
WorkouttypeID	int	<input checked="" type="checkbox"/>
Paymenttype	nvarchar(50)	<input checked="" type="checkbox"/>
PaymentFromdt	datetime	<input checked="" type="checkbox"/>
PaymentTodt	datetime	<input checked="" type="checkbox"/>
PaymentAmount	numeric(18, 0)	<input checked="" type="checkbox"/>
NextRenwalDate	datetime	<input checked="" type="checkbox"/>
CreateDate	datetime	<input checked="" type="checkbox"/>
Createdby	int	<input checked="" type="checkbox"/>
ModifyDate	datetime	<input checked="" type="checkbox"/>
ModifiedBy	int	<input checked="" type="checkbox"/>

## PeriodTB

This table contains period such as 3 months, 6 months, 1 year.

## SAI-PC\SQLEXPRESS...DB - dbo.PeriodTB X

Column Name	Data Type	Allow Nulls
PeriodID	int	<input type="checkbox"/>
Text	varchar(50)	<input checked="" type="checkbox"/>
Value	varchar(50)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

## SchemeMaster

This table contains Schemes.

Example: - GYM+CARDIO1, GYM

Column Name	Data Type	Allow Nulls
SchemeID	int	<input type="checkbox"/>
SchemeName	nvarchar(50)	<input checked="" type="checkbox"/>
Createdby	int	<input checked="" type="checkbox"/>
Createddate	datetime	<input checked="" type="checkbox"/>
Status	bit	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

## **Plan Master**

This table contains Plans along with Plan amount and Service Tax and for one Scheme there can be many plans.

Example: - plan name such as Quarterly, Half Yearly, Yearly, Men Special Plan, Women Special Plan.

Column Name	Data Type	Allow Nulls
PlanID	int	<input type="checkbox"/>
PlanName	varchar(50)	<input checked="" type="checkbox"/>
PlanAmount	decimal(18, 0)	<input checked="" type="checkbox"/>
ServicetaxAmount	decimal(18, 0)	<input checked="" type="checkbox"/>
ServiceTax	varchar(50)	<input checked="" type="checkbox"/>
CreateDate	datetime	<input checked="" type="checkbox"/>
CreateUserID	int	<input checked="" type="checkbox"/>
ModifyDate	datetime	<input checked="" type="checkbox"/>
ModifyUserID	int	<input checked="" type="checkbox"/>
RecStatus	bit	<input checked="" type="checkbox"/>
SchemeID	int	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

## **Role**

This table contains Roles.

Example: - Admin, User.

Column Name	Data Type	Allow Nulls
RoleId	int	<input type="checkbox"/>
RoleName	nvarchar(256)	<input type="checkbox"/>
Status	bit	<input checked="" type="checkbox"/>

## **UsersInRoles**

This table contains Userid and roleid means a role assigned to the user is stored in this table.

Example: - Admin, User.

Column Name	Data Type	Allow Nulls
UserRoleId	int	<input type="checkbox"/>
UserId	int	<input checked="" type="checkbox"/>
RoleId	int	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

## Users

This table contains Users Login Credentials.

Column Name	Data Type	Allow Nulls
UserId	int	<input type="checkbox"/>
UserName	nvarchar(56)	<input type="checkbox"/>
FullName	nvarchar(200)	<input checked="" type="checkbox"/>
EmailId	nvarchar(200)	<input checked="" type="checkbox"/>
Contactno	nvarchar(10)	<input checked="" type="checkbox"/>
Password	nvarchar(200)	<input checked="" type="checkbox"/>
Createdby	int	<input checked="" type="checkbox"/>
CreatedDate	datetime	<input checked="" type="checkbox"/>
Status	bit	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

After completing with Understanding tables structure let's move forward to understand API application structure.

## API Project Structure

In this part, we are going to have a look into the API project in details. APIs Project Name is WebGYM which is developed in ASP.NET CORE Framework version 2.1.

If we see project structure is a simple and clean repository pattern with asp.net core built-in dependency injection framework.

Let's start from Model class library this class library contains all Models which are used in the application and which are exactly similar to database entities. In the same way, you can see ViewModels these models are used for taking request and display response of APIS. Further, we are having a view into interface Class library which contains all interfaces of application used for losing coupling of application this all interfaces are implemented in concrete Class library this layer directly interacts with your database, in this project have used EntityFrameworkCore and Dapper or m to access the database.

Added Asp.net MVC Core Web Project.

## WebGYM

## Solution 'WebGYM' (5 projects)

### WebGYM

- ▷ Connected Services
- ▷ Dependencies
- ▷ Properties
- ▷ wwwroot
- ▷ Common
- ▷ Controllers
- ▷ Mappings
- ▷ Models
- ▷ Views
- ▷ appsettings.json
- ▷ Program.cs
- ▷ Startup.cs
- ▷ WebGYM.Concrete
- ▷ WebGYM.Interface
- ▷ WebGYM.Models
- ▷ WebGYM.ViewModels

Next, we are going to have a look at WebGYM.Models Class Library.

### WebGYM.Models

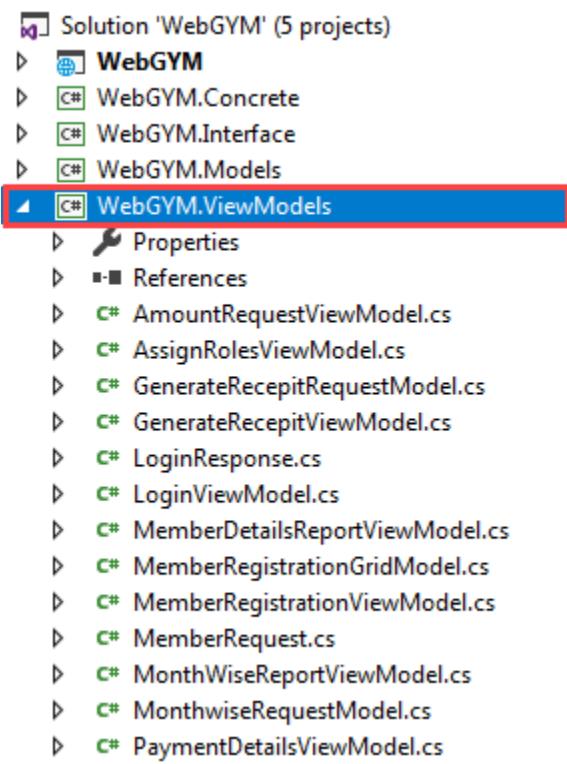
## Solution 'WebGYM' (5 projects)

- ▷ WebGYM
- ▷ WebGYM.Concrete
- ▷ WebGYM.Interface
- ◀ **WebGYM.Models**

- ▷ Properties
- ▷ References
- ▷ DynamicExtensions.cs
- ▷ MemberRegistration.cs
- ▷ PaymentDetails.cs
- ▷ PeriodTB.cs
- ▷ PlanMaster.cs
- ▷ QueryParameters.cs
- ▷ QueryParametersExtensions.cs
- ▷ Role.cs
- ▷ SchemeMaster.cs
- ▷ Users.cs
- ▷ UsersInRoles.cs
- ▷ WebGYM.ViewModels

Next, we are going to have a look at WebGYM.ViewModels Class Library.

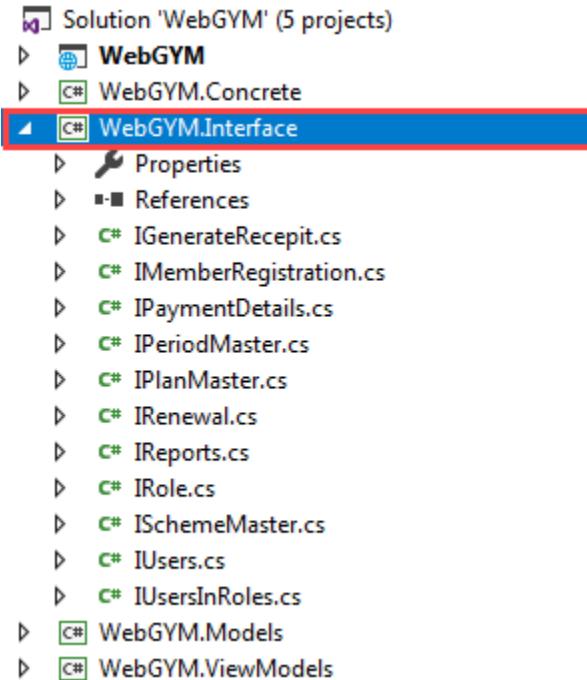
### WebGYM.ViewModels



After having a view of View Models Class library next let's have a view of Interface Class library.

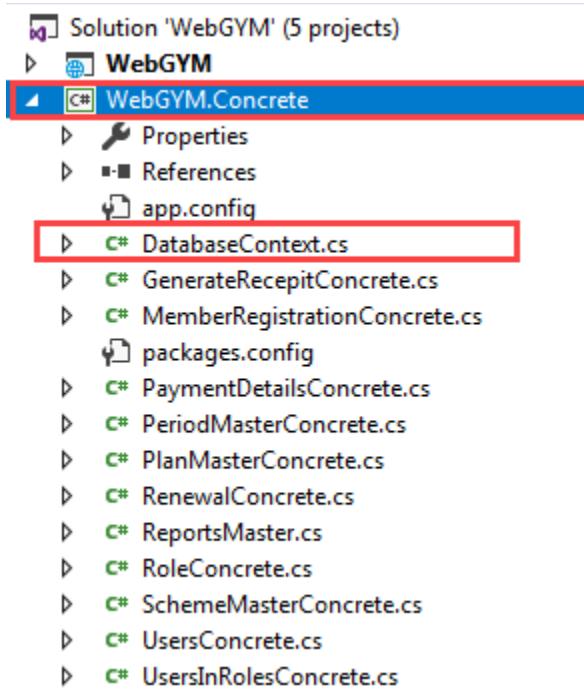
This Class Library contains all Interfaces in which we have declared methods, this interface is going to implement a concrete class (WebGYM.Concrete).

## WebGYM.Interface



## WebGYM.Concrete

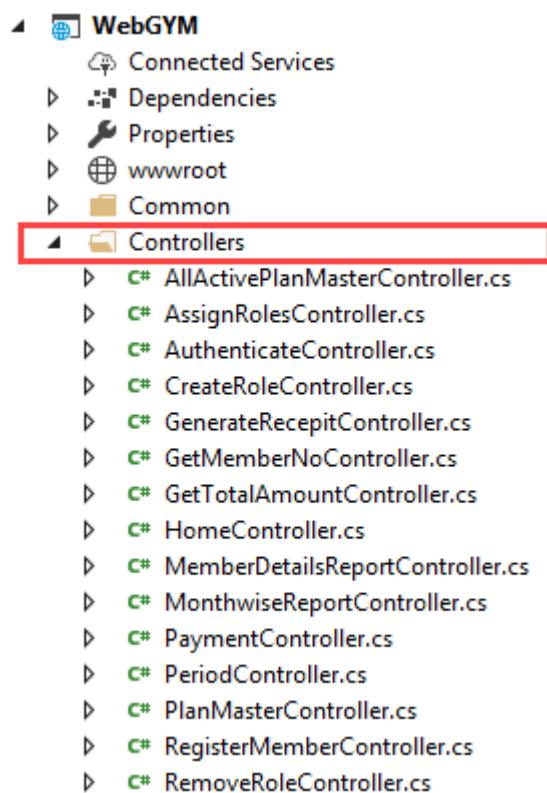
The WebGYM.Concrete Class Library contains all Classes which are using Entity Framework Core ORM for accessing database and it also has DatabaseContext and is the main class for interacting with the database. I have also used Dapper Orm for some database operations.



After Completing View of Class Library Next, we are going to View API Controller Structure.

## Controllers

Controller Folder contains all API Controllers created in this application.



## DbContext (we are using Entity Framework core in this project)

The screenshot shows the Visual Studio IDE with the file 'DatabaseContext.cs' open. The code defines a class 'DbContext' that inherits from 'DbContext'. It contains properties for various database entities: 'SchemeMaster', 'PeriodTb', 'PlanMaster', 'Role', 'MemberRegistration', 'Users', 'UsersInRoles', and 'PaymentDetails'. Each property is annotated with its respective entity type and includes get and set methods.

```
7  using WebGYM.Models;
8  ...
9  namespace WebGYM.Concrete
10 {
11     public class DbContext : DbContext
12     {
13         public DbSet<SchemeMaster> SchemeMaster { get; set; }
14         public DbSet<PeriodTB> PeriodTb { get; set; }
15         public DbSet<PlanMaster> PlanMaster { get; set; }
16         public DbSet<Role> Role { get; set; }
17         public DbSet<MemberRegistration> MemberRegistration { get; set; }
18         public DbSet<Users> Users { get; set; }
19         public DbSet<UsersInRoles> UsersInRoles { get; set; }
20         public DbSet<PaymentDetails> PaymentDetails { get; set; }
21     }
22 }
23 
```

## Appsettings.json file

In the appsettings.json file, we store all application settings in a key-value pair. Here we have stored Connection string of database.

The screenshot shows the Visual Studio IDE with the file 'appsettings.json' open. The JSON object contains several settings: 'AppSettings' (with a secret key), 'Logging' (with a LogLevel of 'Warning'), 'AllowedHosts' (set to '\*'), and 'ConnectionStrings' (containing a connection string for 'DatabaseConnection').

```
1  {
2      "AppSettings": {
3          "Secret": "6XJCIEJ041PQZNWJC4RR"
4      },
5      "Logging": {
6          "LogLevel": {
7              "Default": "Warning"
8          }
9      },
10     "AllowedHosts": "*",
11     "ConnectionStrings": {
12         "DatabaseConnection": "Data Source=SAI-PC\\SQLEXPRESS; UID=sa; Password=Pass$123;Database=AngularGYMDB;"
13     }
14 }
```

## Setting Dependence injection in Startup.cs class

The screenshot shows the Visual Studio IDE with the 'WebGYM - Startup.cs' window active. The code in the startup.cs file is as follows:

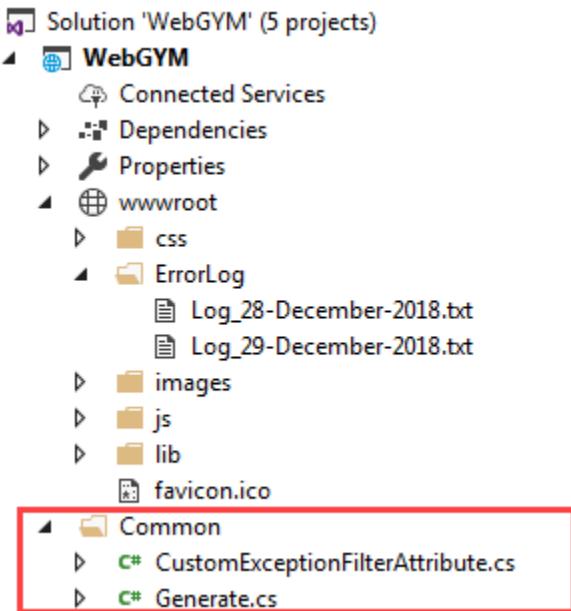
```
73 services.AddSingleton< IConfiguration>(Configuration);
74 services.AddTransient< ISchemeMaster, SchemeMasterConcrete>();
75 services.AddTransient< IPlanMaster, PlanMasterConcrete>();
76 services.AddTransient< IPeriodMaster, PeriodMasterConcrete>();
77 services.AddTransient< IRole, RoleConcrete>();
78 services.AddTransient< IMemberRegistration, MemberRegistrationConcrete>();
79 services.AddTransient< IUsers, UsersConcrete>();
80 services.AddTransient< IUsersInRoles, UsersInRolesConcrete>();
81 services.AddTransient< IPaymentDetails, PaymentDetailsConcrete>();
82 services.AddTransient< IRenewal, RenewalConcrete>();
83 services.AddTransient< IReports, ReportsMaster>();
84 services.AddTransient< IGenerateRecepit, GenerateRecepitConcrete>();
85 services.AddSingleton< IActionContextAccessor, ActionContextAccessor>();
86 services.AddScoped< IUrlHelper>(implementationFactory =>
87 {
88     var actionContext = implementationFactory.GetService< IActionContextAccessor>().ActionContext;
89     return new UrlHelper(actionContext);
90 });
91 });
92 }
```

## Filters and Encryption library

For Authentication of APIs I have Used JSON Web Tokens and for error logging, I have created “**CustomExceptionFilterAttribute**” which is registered as a filter in ConfigureServices Method in start-up Class.

As you can see the error log folder which contains text files where generated errors in the application are stored.

For Encryption of Password, I have used Advanced Encryption Standard (AES) Algorithm.

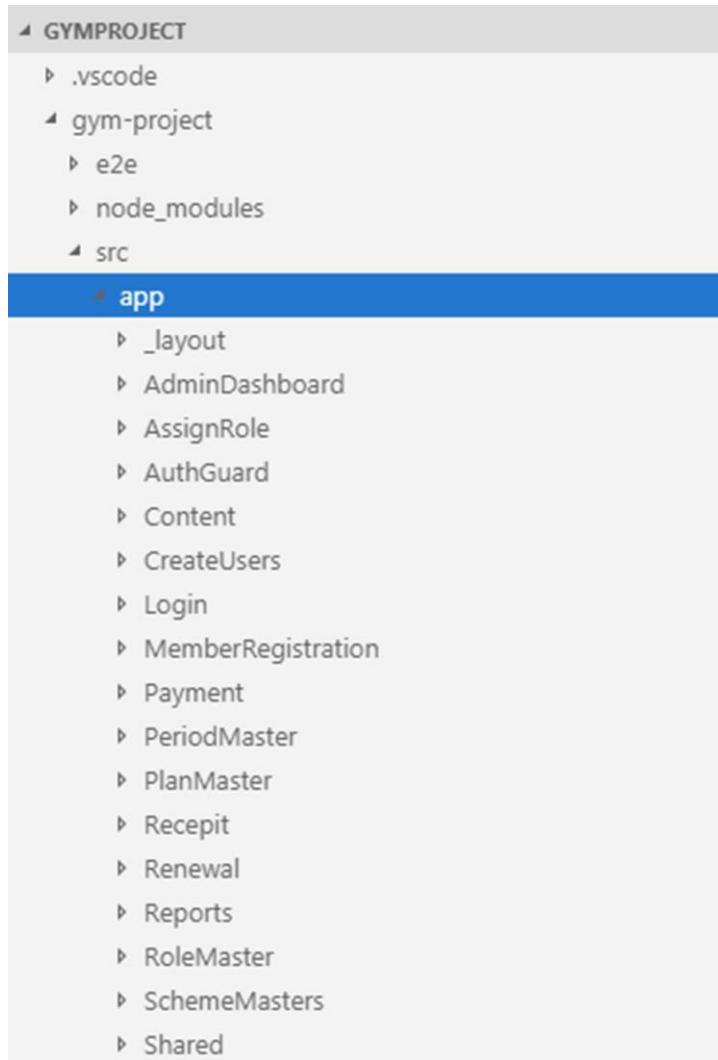


The above shows how the directory and folder structure look, now let's check out Application screens.

After having a view of completed API Application Screens next we are going to view Angular Application Project Structure.

## **Angular Application Project Structure**

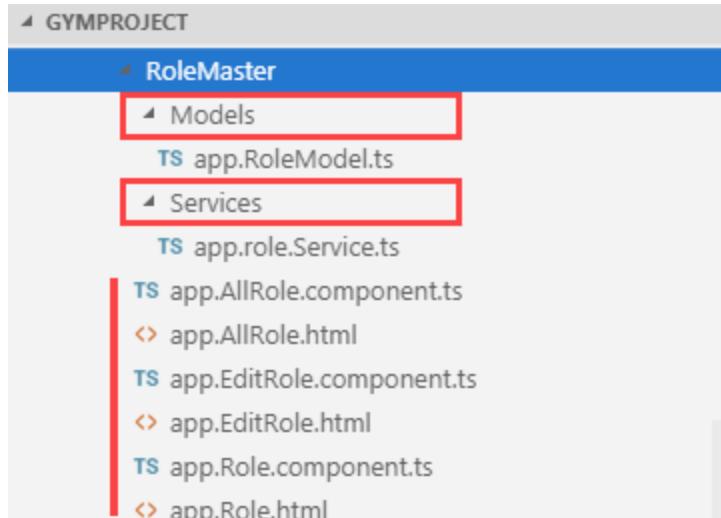
Below is a complete view of Angular Application project structure, if you see below snapshot you will see all folder with proper naming which tells which folder contains which Component in it.



Next let's View some folder to know how we are maintaining Services, Models, Component and Html Views.

Let's Expand RoleMaster Folder if you see int that folder we have 2 folders in it one is Models folder and another is Service Folder.

Model Folder contains Class which is used with Views and API to Post, Get Request and Response.



Let's have look completed flow of role module you will get a good idea how the application is designed.

### app.RoleModel.ts

This is model which is used on role HTML template for model binding.

```
export class RoleModel
{
    public RoleName: string;
    public Status: boolean;
    public RoleId : number;
}
```

### app.Role.component.ts

Role Component which is going to render role temple and going to Role call service to save role data in to database.

```
import { Component } from '@angular/core';
import { RoleModel } from './Models/app.RoleModel';
import { RoleService } from './Services/app.role.Service';
import { Router } from '@angular/router';

@Component({
    templateUrl : './app.Role.html',
    styleUrls: [ '../Content/vendor/bootstrap/css/bootstrap.min.css' ,
        '../Content/vendor/metisMenu/metisMenu.min.css' ,
        '../Content/dist/css/sb-admin-2.css' ,
        '../Content/vendor/font-awesome/css/font-awesome.min.css'
    ]
})

export class RoleComponent
{
    private _roleService;
    RoleModel : RoleModel = new RoleModel();
    output: any;

    constructor(private _Route: Router, roleService :RoleService ){
        this._roleService = roleService;
    }

    onSubmit()
    {
        this._roleService.AddRole(this.RoleModel).subscribe(
            response => {
                this.output = response
                if (this.output.StatusCode == "409") {
                    alert('Role Already Exists');
                }
                else if (this.output.StatusCode == "200") {
                    alert('Role Saved Successfully');
                    this._Route.navigate(['/Role/All']);
                }
                else {
                    alert('Something Went Wrong');
                }
            }
        )
    }
}
```

```
});  
}  
  
}
```

Moving forward you can see the Services folder in this folder we are consuming Web API Services.

## **Role.Service**

In this class, we use HttpClient to call API services and along with that we also send JWT token which is used for authentication. If you see Role.Service class you will find all kind of Http Method get, post, put, delete.

```
import { Injectable } from '@angular/core';  
import { Observable, throwError } from 'rxjs'  
import { catchError, tap } from 'rxjs/operators'  
import { HttpClient, HttpHeaders, HttpResponse } from  
'@angular/common/http';  
import { RoleModel } from '../Models/app.RoleModel';  
import { environment } from 'src/app/Shared/environment';  
@Injectable({  
    providedIn: 'root'  
)  
  
export class RoleService {  
  
    private data: any;  
    private apiUrl = environment.apiEndpoint + "/api/CreateRole/";  
    token: any;  
    username: any;  
  
    constructor(private http: HttpClient) {  
        this.data = JSON.parse(localStorage.getItem('AdminUser'));  
        this.token = this.data.token;  
    }  
  
    public AddRole(rolemodel: RoleModel) {  
        let headers = new HttpHeaders({ 'Content-Type': 'application/json' });  
        headers = headers.append('Authorization', 'Bearer ' + `${this.token}`);  
        return this.http.post<any>(this.apiUrl, rolemodel, { headers: headers })  
            .pipe(  
                catchError(this.handleError)  
            );  
    }  
  
    // Get All Role  
    public GetAllRole() {  
        let headers = new HttpHeaders({ 'Content-Type': 'application/json' });  
        headers = headers.append('Authorization', 'Bearer ' + `${this.token}`);  
        return this.http.get<RoleModel[]>(this.apiUrl, { headers: headers }).pipe(tap(data  
=> data),  
            catchError(this.handleError)  
        );  
    }  
  
    // Get All Role By ID
```

```

public GetRoleById(RoleId) {
    var editUrl = this.apiUrl + '/' + RoleId;
    let headers = new HttpHeaders({ 'Content-Type': 'application/json' });
    headers = headers.append('Authorization', 'Bearer ' + `${this.token}`);
    return this.http.get<RoleModel>(editUrl, { headers: headers }).pipe(tap(data =>
data),
    catchError(this.handleError)
);
}

// Update Role
public UpdateRole(rolemodel: RoleModel) {
    var putUrl = this.apiUrl + '/' + rolemodel.RoleId;
    let headers = new HttpHeaders({ 'Content-Type': 'application/json' });
    headers = headers.append('Authorization', 'Bearer ' + `${this.token}`);
    return this.http.put<any>(putUrl, rolemodel, { headers: headers })
    .pipe(
        catchError(this.handleError)
);
}

public DeleteRole(RoleId) {
    var deleteUrl = this.apiUrl + '/' + RoleId;
    let headers = new HttpHeaders({ 'Content-Type': 'application/json' });
    headers = headers.append('Authorization', 'Bearer ' + `${this.token}`);
    return this.http.delete<any>(deleteUrl, { headers: headers })
    .pipe(
        catchError(this.handleError)
);
}

private handleError(error: HttpErrorResponse) {
    if (error.error instanceof ErrorEvent) {
        // A client-side or network error occurred. Handle it accordingly.
        console.error('An error occurred:', error.error.message);
    } else {
        // The backend returned an unsuccessful response code.
        // The response body may contain clues as to what went wrong,
        console.error(`Backend returned code ${error.status}, ` + `body was:
${error.error}`);
    }
    // return an observable with a user-facing error message
    return throwError('Something bad happened; please try again later.');
};
}

```

After we have viewed Services folder Next let's view Html Templates of Role Module.

## app.Role.html template

app.Role.html Template which is been used as templateUrl in app.Role.component.ts class.

```
<h4>Add Role</h4>
<hr>
<div class="panel panel-default">
  <div class="panel-heading">Add Role</div>
  <div class="panel-body">
    <form #f="ngForm" novalidate (ngSubmit)="onSubmit()">
      <div class="row">
        <div class="col-md-4">
          <label for="name">RoleName</label>
          <input type="text" class="form-control" name="RoleName"
[(ngModel)]="RoleModel.RoleName" maxlength="50"
          #refRoleName="ngModel" id="RoleName" required>
          <div *ngIf="!refRoleName.valid && (refRoleName.dirty || refRoleName.touched)" class="alert alert-danger">
            <div [hidden]="!refRoleName.errors.required">
              RoleName is required
            </div>
          </div>
        <div class="col-md-4">
          <label for="name">Status</label>
          <input type="checkbox" name="Status" [(ngModel)]="RoleModel.Status"
maxlength="50" #refStatus="ngModel"
          id="Status" required>
          <div *ngIf="!refStatus.valid && (refStatus.dirty || refStatus.touched)" class="alert alert-danger">
            <div [hidden]="!refStatus.errors.required">
              Status is required
            </div>
          </div>
        <div class="col-md-4">
          <button type="submit" style="margin-top: 10px"
[disabled]="!f.form.valid" class="btn btn-success">Submit</button>
          <a style="margin-left: 10px; margin-top:7px;" class="btn btn-info"
[routerLink]=["'/Role/All'"]>
            All Roles </a>
          </div>
        </div>
      </div>
    </form>
  </div>
```

## package.json

Package json contains all packages which are used in this project.

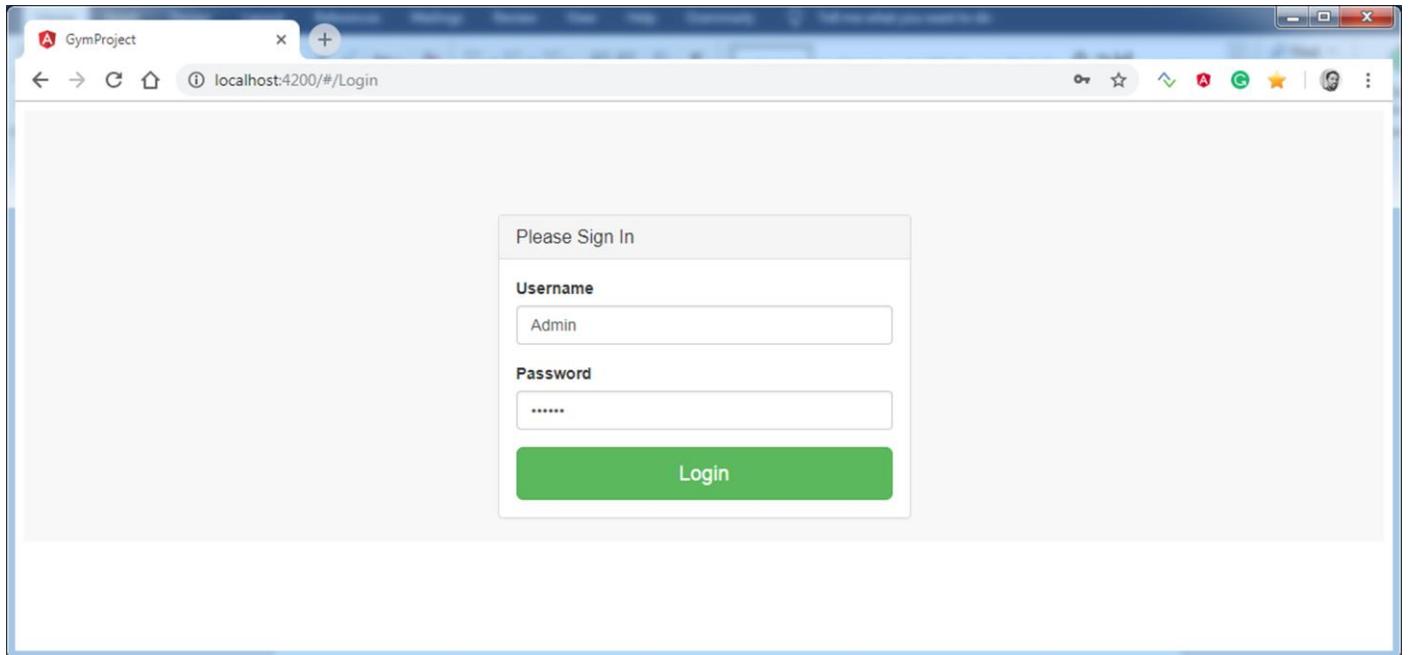
```
{  
  "name": "gym-project",  
  "version": "0.0.0",  
  "scripts": {  
    "ng": "ng",  
    "start": "ng serve",  
    "build": "ng build",  
    "test": "ng test",  
    "lint": "ng lint",  
    "e2e": "ng e2e"  
  },  
  "private": true,  
  "dependencies": {  
    "@angular/animations": "^7.0.4",  
    "@angular/cdk": "^7.1.1",  
    "@angular/common": "~7.0.0",  
    "@angular/compiler": "~7.0.0",  
    "@angular/core": "~7.0.0",  
    "@angular/forms": "~7.0.0",  
    "@angular/http": "~7.0.0",  
    "@angular/material": "^7.1.1",  
    "@angular/platform-browser": "~7.0.0",  
    "@angular/platform-browser-dynamic": "~7.0.0",  
    "@angular/router": "~7.0.0",  
    "core-js": "^2.5.4",  
    "hammerjs": "^2.0.8",  
    "html2canvas": "^1.0.0-alpha.12",  
    "jspdf": "^1.5.2",  
    "ngx-bootstrap": "^3.1.2",  
    "rxjs": "~6.3.3",  
    "xlsx": "^0.14.1",  
    "zone.js": "~0.8.26"  
  },  
  "devDependencies": {  
    "@angular-devkit/build-angular": "~0.10.0",  
    "@angular/cli": "~7.0.6",  
    "@angular/compiler-cli": "~7.0.0",  
    "@angular/language-service": "~7.0.0",  
    "@types/node": "~8.9.4",  
    "@types/jasmine": "~2.8.8",  
    "@types/jasminewd2": "~2.0.3",  
    "codelyzer": "~4.5.0",  
    "jasmine-core": "~2.99.1",  
    "jasmine-spec-reporter": "~4.2.1",  
    "karma": "~3.0.0",  
    "karma-chrome-launcher": "~2.2.0",  
    "karma-coverage-istanbul-reporter": "~2.0.1",  
    "karma-jasmine": "~1.1.2",  
    "karma-jasmine-html-reporter": "^0.2.2",  
    "protractor": "~5.4.0",  
    "ts-node": "~7.0.0",  
    "tslint": "~5.11.0",  
    "typescript": "~3.1.6"  
  }  
}
```

After we have completed understand single module next, we are going to have a look on all admin screens of applications.

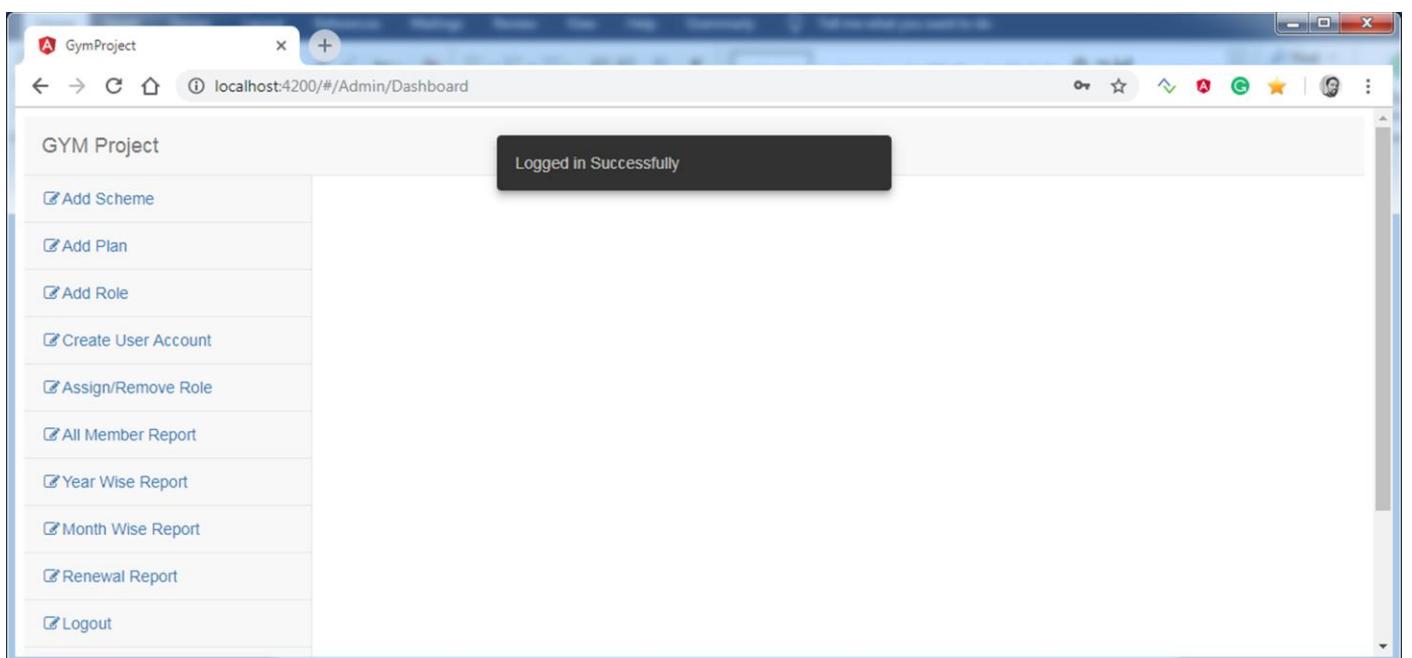
## Admin Application Screens

Login is Common for Admin and User.

We are going to log in to the application with admin credentials.



After logging into application first screen appears is your dashboard.



After login into the application, we are first going to view see Add Scheme.

## Add Scheme

In this part, we are going to Add New Scheme.

The screenshot shows a web browser window titled 'GymProject'. On the left, there is a sidebar menu with the following items:

- Add Scheme
- Add Plan
- Add Role
- Create User Account
- Assign/Remove Role
- All Member Report
- Year Wise Report
- Month Wise Report
- Renewal Report
- Logout

The main content area is titled 'Add Scheme' and contains a form with two fields: 'SchemeName' and 'Status'. Below the form are two buttons: 'Submit' (green) and 'All Scheme' (blue).

## All Scheme

Displays all scheme which is added.

The screenshot shows a web browser window titled 'GymProject'. On the left, there is a sidebar menu with the following items:

- Add Scheme
- Add Plan
- Add Role
- Create User Account
- Assign/Remove Role
- All Member Report
- Year Wise Report
- Month Wise Report
- Renewal Report
- Logout

The main content area is titled 'All Scheme' and contains a table with the following data:

SchemeID	SchemeName	Status	Createddate	Action	Action
6	GYM+CARDIO1	true	2015-07-31T09:32:38.3	<input checked="" type="button"/> Edit	<input type="button"/> Delete
7	GYM	true	2014-04-20T00:00:00	<input checked="" type="button"/> Edit	<input type="button"/> Delete
12	Cardio	true	2018-11-24T16:57:43.38	<input checked="" type="button"/> Edit	<input type="button"/> Delete
18	Test	true	2018-12-25T16:58:29.697	<input checked="" type="button"/> Edit	<input type="button"/> Delete
22	pro	false	2018-12-25T18:12:59.05	<input checked="" type="button"/> Edit	<input type="button"/> Delete

At the bottom, there are pagination controls: 'Items per page: 5' and '1 - 5 of 5'.

## Edit Scheme

In this part, we can just edit the status of the scheme. If you want to change the name then you can delete and add a new scheme.

The screenshot shows a web browser window titled "GymProject" with the URL "localhost:4200/#/Scheme/Edit/6". The left sidebar contains a list of navigation items: "Add Scheme", "Add Plan", "Add Role", "Create User Account", "Assign/Remove Role", "All Member Report", "Year Wise Report", "Month Wise Report", and "Renewal Report". The main content area is titled "Edit Scheme" and contains a form with the following fields: "SchemeName" (input value: "GYM+CARDIO1"), "Status" (checkbox checked), "Submit" button, and "All Scheme" button.

## Add Plan

In this part, we are going to add the plan and while adding we are going to select Period and Scheme also enter Plan amount and required service tax for it.

The screenshot shows a web browser window titled "GymProject" with the URL "localhost:4200/#/Plan/Add". The left sidebar contains a list of navigation items, with "Add Plan" highlighted by a red box. The main content area is titled "Add Plan" and contains a form with the following fields: "PlanName" (input value: "Quarely"), "Period" (dropdown value: "3 Month"), "Scheme" (dropdown value: "GYM+CARDIO1"), "PlanAmount" (input value: "5000"), "ServiceTax" (input value: "230"), "Submit" button, and "All Plans" button.

## All Plan

Displays all Plans which are added.

The screenshot shows a web-based application titled "Gym Project". On the left, there is a sidebar with various menu items: Add Scheme, Add Plan, Add Role, Create User Account, Assign/Remove Role, All Member Report, Year Wise Report, Month Wise Report, Renewal Report, and Logout. The main content area is titled "All Plan" and contains a table with five rows of data. The columns are PlanID, PlanName, SchemeName, Period, TotalAmount, Status, and Action (with Edit and Delete buttons). The data in the table is as follows:

PlanID	PlanName	SchemeName	Period	TotalAmount	Status	Action
1	Quarety	GYM	1 Year	1348	DeActive	<a href="#">Edit</a> <a href="#">Delete</a>
2	Half Yearly	GYM	6 Month	2247	DeActive	<a href="#">Edit</a> <a href="#">Delete</a>
3	Yearly	GYM	1 Year	3932	DeActive	<a href="#">Edit</a> <a href="#">Delete</a>
4	Yearly	GYM+CARDIO1	1 Year	5393	DeActive	<a href="#">Edit</a> <a href="#">Delete</a>
5	Quarety	GYM+CARDIO1	6 Month	2022	DeActive	<a href="#">Edit</a> <a href="#">Delete</a>

At the bottom, there is a pagination control with "Items per page: 5" and "1 - 5 of 11" followed by navigation arrows.

## Edit Plan

In this part, we can Edit Plan which we have added.

The screenshot shows the "Edit Plan" page. The sidebar on the left is identical to the previous screenshot. The main content area is titled "Edit Plan" and contains a form with fields for PlanName, PlanAmount, Period, and ServiceTax. The form is currently populated with the values from the first row of the "All Plan" table. At the bottom of the form are two buttons: "Submit" and "All Plans".

## Add Role

In this part, we are going to add new roles

The screenshot shows a web browser window titled 'GymProject' at the URL 'localhost:4200/#/Role/Add'. On the left, there is a sidebar with various menu items: 'Add Scheme', 'Add Plan', 'Add Role' (which is highlighted), 'Create User Account', 'Assign/Remove Role', 'All Member Report', 'Year Wise Report', 'Month Wise Report', 'Renewal Report', and 'Logout'. The main content area is titled 'Add Role' and contains a form with fields for 'RoleName' and 'Status'. There are 'Submit' and 'All Roles' buttons at the bottom right of the form.

## All Role

Displays all Roles which are added.

The screenshot shows a web browser window titled 'GymProject' at the URL 'localhost:4200/#/Role/All'. The sidebar on the left is identical to the one in the previous screenshot. The main content area is titled 'All Role' and displays a table of roles. The table has columns for 'RoleId', 'RoleName', 'Status', and 'Action'. It shows two entries: '1 Admin Active Edit Delete' and '2 User Active Edit Delete'. At the bottom of the table, there is a pagination control with 'Items per page: 5' and navigation arrows for '1 - 2 of 2'.

## Edit Role

In this part, we are going to edit and update role status.

The screenshot shows a web browser window titled "GymProject" at the URL "localhost:4200/#/Role/Edit/1". On the left, there is a sidebar with various menu items: "Add Scheme", "Add Plan", "Add Role", "Create User Account", "Assign/Remove Role", "All Member Report", "Year Wise Report", "Month Wise Report", "Renewal Report", and "Logout". The "Create User Account" item is highlighted with a red border. The main content area is titled "Edit Role" and contains a form for editing a role. The form has fields for "RoleName" (set to "Admin") and "Status" (checkbox checked). There are "Submit" and "All Roles" buttons at the bottom right of the form.

## Create New User

In this part, we are going to add new User/admin who is going to Use this Application.

The screenshot shows a web browser window titled "GymProject" at the URL "localhost:4200/#/User/Add". On the left, there is a sidebar with menu items: "Add Scheme", "Add Plan", "Add Role", "Create User Account", "Assign/Remove Role", "All Member Report", "Year Wise Report", "Month Wise Report", "Renewal Report", and "Logout". The "Create User Account" item is highlighted with a red border. The main content area is titled "Add System User" and contains a form for adding a new user. The form has fields for "UserName", "FullName", "EmailId", "Contactno", "Password", and "Status" (checkbox). There are "Submit" and "All Users" buttons at the bottom right of the form.

## All Users

Displays all Users which are added.

User Id	User Name	Full Name	Email Id	Contact No	Status	Action	Action
1	Admin	Admin	saihacksoft@gmail.com	9999999999	Active	<input checked="" type="button"/> Edit	<input type="button"/> Delete
2	User	User	User@gmail.com	9999999999	Active	<input checked="" type="button"/> Edit	<input type="button"/> Delete
4	Demo	Demo	Demo@gmail.com	9998887770	Active	<input checked="" type="button"/> Edit	<input type="button"/> Delete
5	superadmin	superadmin	superadmin@gmail.com	9999999999	Active	<input checked="" type="button"/> Edit	<input type="button"/> Delete
6	demox	demox	demox@gg.cpm	9999999999	Active	<input checked="" type="button"/> Edit	<input type="button"/> Delete

## Edit Users

In this part, we can Edit User Details which we have added such as User Email or Phone and Password.

<b>UserName</b>	<b>FullName</b>	<b>EmailId</b>
User	User	User@gmail.com
Contactno	Password	Status <input checked="" type="checkbox"/>
9999999999	*****	<input type="button"/> Submit <input type="button"/> All Users

## Assign / Remove Role

In this part, we are going to assign and remove roles to the user which we have created.

The screenshot shows the 'Assign / Remove Role' page. On the left sidebar, under 'Create User Account', the 'Assign/Remove Role' option is selected and highlighted with a red border. The main area has a title 'Assign / Remove Role'. It contains two dropdown menus: 'User' (set to 'Demo') and 'Role' (set to 'Please select Role'). Below these are three buttons: 'Assign Role' (green), 'Remove Role' (green), and 'All Assign Role' (blue).

## All Assigned Roles

In this part, we are going to assign and remove roles to the user which we have created.

The screenshot shows the 'All AssignRoles' page. On the left sidebar, under 'Create User Account', the 'Assign/Remove Role' option is selected. The main area has a title 'All AssignRoles' and a 'Filter' input field. Below is a table showing assigned roles:

RoleName	UserName
Admin	Admin
User	User
Admin	Admin
User	User

At the bottom, there are pagination controls: 'Items per page: 5', '1 - 4 of 4', and navigation arrows.

## All Member Report

In this part, admin can export all member data for reporting.

MemberNo	Name	Contactno	EmailID	PlanName	SchemeName	JoiningDate	RenewalDate	PaymentAmount
GYMONE/3/2019	two   two	99999999	gmail.com	Yearly	GYM	25/12/2018	25/03/2019	3932
GYMONE/2/2019	one   one	9999999999	one@gmail.com	Quarety	GYM	25/12/2018	25/03/2019	1348
GYMONE/1/2019	Sai   Bag	99999999	demo@gmail.com	Yearly	GYM+CARDIO1	25/12/2018	25/03/2019	5393

### Exported all Member Report in Excel

MemberNo	Name	Contactno	EmailID	PlanName	SchemeName	JoiningDate	RenwalDate	PaymentAmount
GYMONE/3/2019	two   two	99999999	gmail.com	Yearly	GYM	25/12/2018	25/03/2019	3932
GYMONE/2/2019	one   one	9999999999	one@gmail.com	Quarety	GYM	25/12/2018	25/03/2019	1348
GYMONE/1/2019	Sai   Bag	99999999	demo@gmail.com	Yearly	GYM+CARDIO1	25/12/2018	25/03/2019	5393

### Year wise Collection Report

In this part, the admin will select the year and click on the submit button to generate a report and also can export it.

The screenshot shows a web-based application titled "Gym Project". On the left, there is a sidebar with various menu items: "Add Scheme", "Add Plan", "Add Role", "Create User Account", "Assign/Remove Role", "All Member Report", "Year Wise Report" (which is currently selected), "Month Wise Report", "Renewal Report", and "Logout". The main content area is titled "Year Wise Collection Report". It contains a "Report" section with a dropdown menu set to "2018" and buttons for "Submit" and "Export as Excel". Below this is a table showing monthly collection data for the year 2019. The table has columns for CurrentYear, April, May, June, July, August, Sept, Oct, Nov, Decm, Jan, Feb, March, and Total. The data row for 2019 shows values: 0, 0, 0, 0, 0, 0, 0, 0, 0, 14717, 0, 0, 0, and 14717 respectively.

CurrentYear	April	May	June	July	August	Sept	Oct	Nov	Decm	Jan	Feb	March	Total
2019	0	0	0	0	0	0	0	0	14717	0	0	0	14717

### Exported Year Wise Collection Report in Excel

The screenshot shows a Microsoft Excel spreadsheet titled "YearWiseReport.xlsx - Excel". The spreadsheet has a single sheet named "Sheet1". The data is organized into two rows. Row 1 contains column headers: "CurrentYear", "April", "May", "June", "July", "August", "Sept", "Oct", "Nov", "Decm", "Jan", "Feb", "March", and "Total". Row 2 contains data for the year 2019: 0, 0, 0, 0, 0, 0, 0, 0, 0, 14717, 0, 0, 0, and 14717 respectively. The "CurrentYear" header in Row 1 is bolded.

CurrentYear	April	May	June	July	August	Sept	Oct	Nov	Decm	Jan	Feb	March	Total
2019	0	0	0	0	0	0	0	0	14717	0	0	0	14717

### Month-wise Collection Report

In this part, the admin will select Month and click on the submit button to generate and export report.

## Exported Month Wise Collection Report in Excel

	FirstName	MemberNo	LastName	MiddleName	CreateDate	Total	Paymentmonth	PaymentAmount	Username
1	Martin	GYMONE/5/2019	bags	Martin	18/01/2019	1348	January	1348	User
2									
3									
4									
5									
6									

## Renewal Report

In this part admin just need to choose dates from and to and submit to get all renewal Member details.

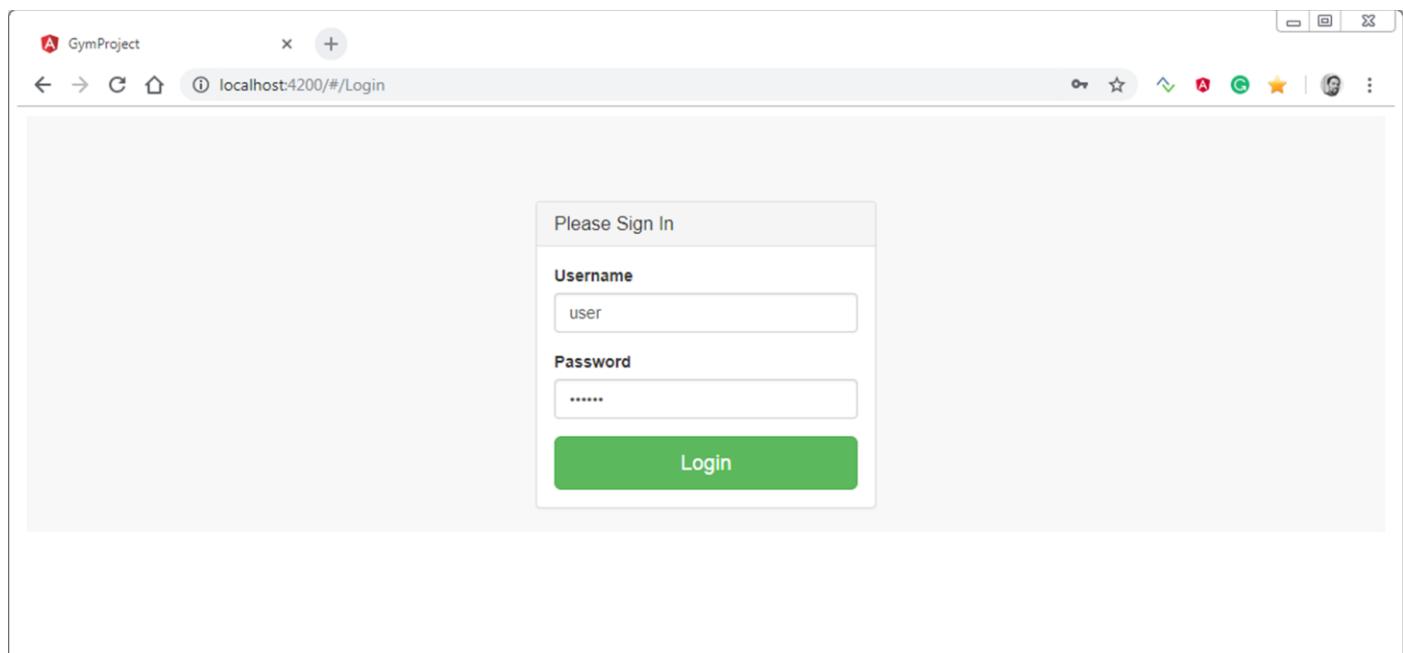
## Exported Renewal Report in Excel

The screenshot shows a Microsoft Excel spreadsheet titled "RenewalReport.xlsx - Excel". The sheet is named "Sheet1". The data is organized in a table with the following columns:

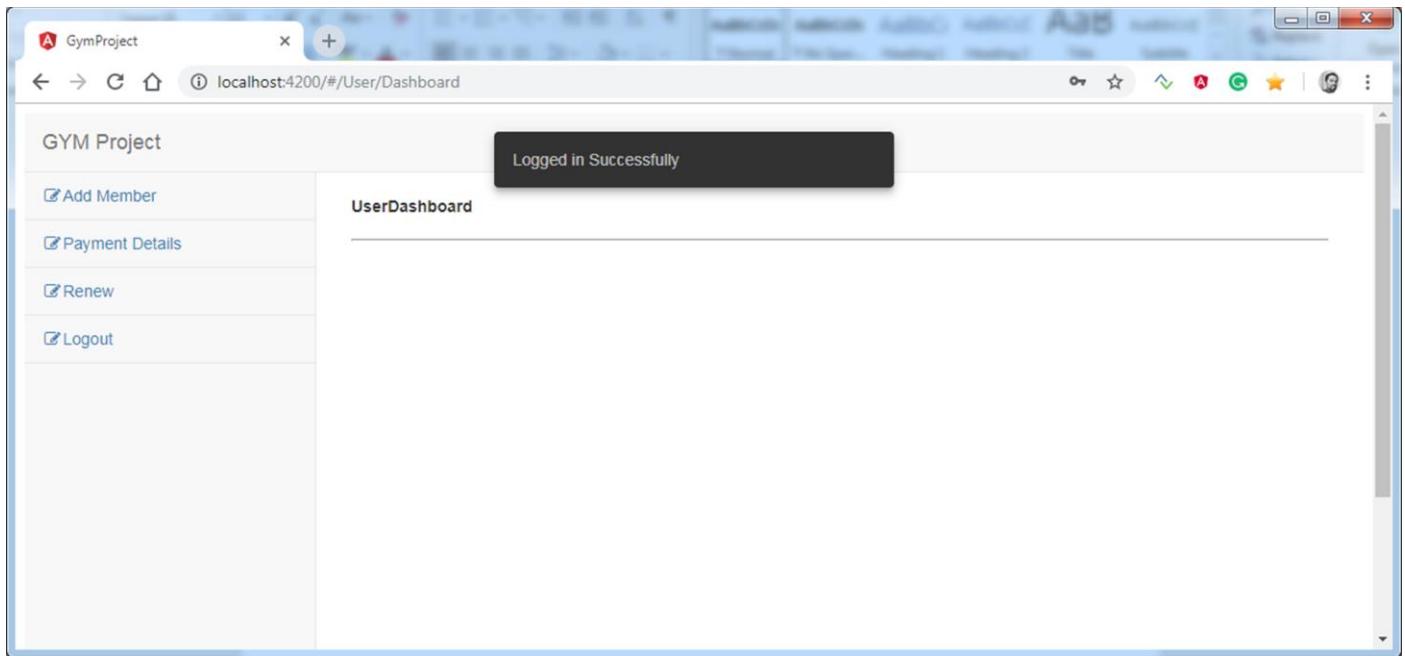
	A	B	C	D	E	F	G	H	I	J
1	Name	Contactno	EmailID	MemberNo	PlanName	SchemeName	JoiningDate	RenwalDate	PaymentAmount	
2	demo   demo	9999999999	demo@gg.com	GYMONE/4/2019	Quarately	GYM+CARDIO1	29/12/2018	28/02/2019	2022	
3										
4										
5										
6										

## User Application Screens

We are going to log in to the application with User credentials.

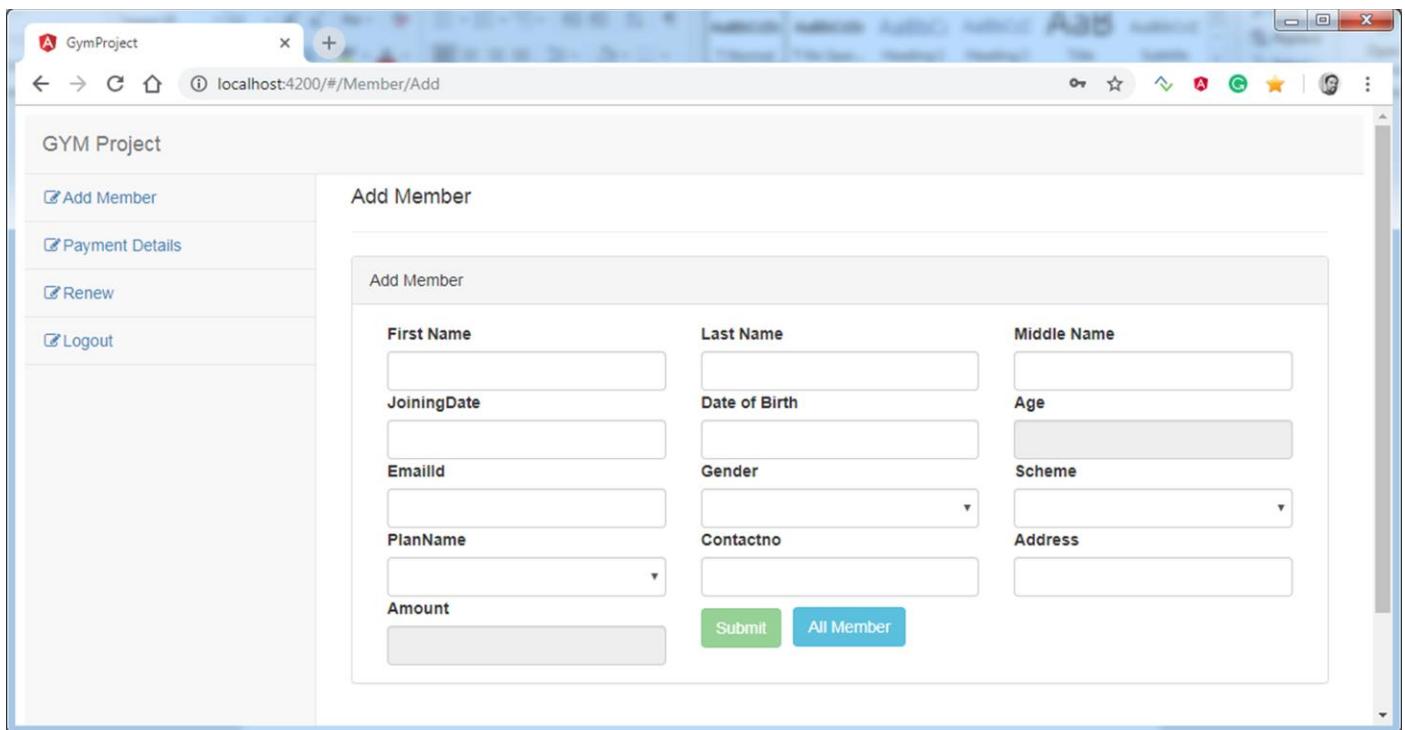


## User Dashboard



## Add Member

From this view, we are going to Register New member and also, we select scheme and plan here according to plan we choose it will get the amount of membership member need to pay.



## All Member

Display all Members who took Membership.

MemberId	MemberNo	MemberName	Contactno	PlanName	SchemeName	JoiningDate	EditAction	DeleteAction
1	GYMONE/1/2019	Sai Bag Bag	999999999	Yearly	GYM+CARDIO1	2018-12-25	<button>Edit</button>	<button>Delete</button>
2	GYMONE/2/2019	one one one	999999999	Quarety	GYM	2018-12-25	<button>Edit</button>	<button>Delete</button>
3	GYMONE/3/2019	two two two	99999999	Yearly	GYM	2018-12-25	<button>Edit</button>	<button>Delete</button>
6	GYMONE/4/2019	demo demo demo	999999999	Quarety	GYM+CARDIO1	2018-12-29	<button>Edit</button>	<button>Delete</button>
7	GYMONE/5/2019	Martin Martin bags	888888888	Quarety	GYM	2019-01-18	<button>Edit</button>	<button>Delete</button>

Items per page: 10 | 1 - 5 of 5 | < >

## Edit Member

In the edit part, we can edit some Member details but we cannot change the plan of members. If something is wrong then we need to delete member again and create a new one.

Edit Member		
<b>First Name</b> Martin	<b>Last Name</b> bags	<b>Middle Name</b> Martin
<b>JoiningDate</b> 2019-01-18	<b>Date of Birth</b> 1992-05-11	<b>Age</b> 26
<b>EmailId</b> Martin@yahoo.in	<b>Gender</b> Male	<b>Scheme</b> GYM
<b>PlanName</b> Quarety	<b>Contactno</b> 888888888	<b>Address</b> Goa
<b>Amount</b> 1348	<b>Submit</b> <b>All Member</b>	

## All Payment Details

The Member who took Membership there all payment details is displayed in this view.

Smo.	MemberNo	MemberName	PlanName	SchemeName	PaymentAmount	NextRenwalDate	PaymentFromdt	PaymentTodt	PrintAction
1	GYMONE/1/2019	Sai Bag Bag	Yearly	GYM+CARDIO1	5393	2019-03-25	2018-12-25	2019-03-25	<input checked="" type="button"/> Receipt
2	GYMONE/2/2019	one one one	Quareltly	GYM	1348	2019-03-25	2018-12-25	2019-03-25	<input checked="" type="button"/> Receipt
3	GYMONE/3/2019	two two two	Yearly	GYM	3932	2019-03-25	2018-12-25	2019-03-25	<input checked="" type="button"/> Receipt
5	GYMONE/4/2019	demo demo demo	Quareltly	GYM+CARDIO1	2022	2019-02-28	2018-12-29	2019-02-28	<input checked="" type="button"/> Receipt
8	GYMONE/4/2019	demo demo demo	Quareltly	GYM+CARDIO1	2022	2019-07-08	2019-05-08	2019-07-08	<input checked="" type="button"/> Receipt
9	GYMONE/5/2019	Martin Martin bags	Quareltly	GYM	1348	2019-04-18	2019-01-18	2019-04-18	<input checked="" type="button"/> Receipt

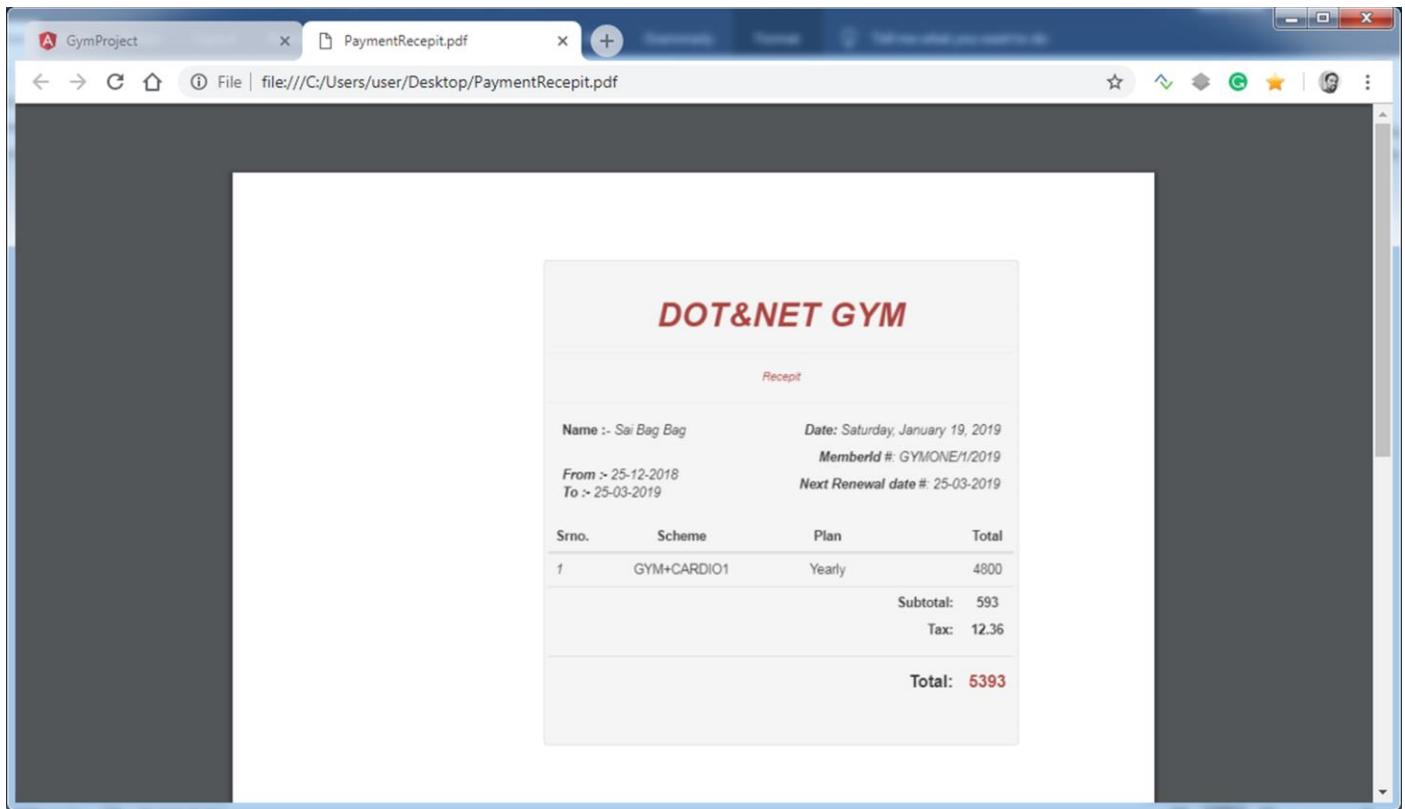
Items per page: 10 | 1 - 6 of 6 | < >

## Receipt

In this part, we need to provide a receipt to a member who took membership. On-grid we have Receipt button just clicking on it will show you a receipt of a particular member. On receipt page, we have a print button to generate pdf.

DOT&NET GYM			
<i>Recepit</i>			
Name :- Sai Bag Bag	Date: Saturday, January 19, 2019		
From :- 25-12-2018	MemberId #: GYMONE/1/2019		
To :- 25-03-2019	Next Renewal date #: 25-03-2019		
Srno.	Scheme	Plan	Total
1	GYM+CARDIO1	Yearly	4800
			Subtotal: 593
			Tax: 12.36
			<b>Total: 5393</b>

## Pdf format of the receipt



## **Renewal of Membership**

In this part, we are going to renew membership of member whose membership is going to expire. The page is simple we just need to search member name which a autocomplete after choosing a name it will get all member details. here we can change Member Scheme and Plan if he wants and we can choose New Membership Date for renewal.

The screenshot shows a web application interface for a gym project. On the left, there's a sidebar with links: 'Add Member', 'Payment Details', 'Renew', and 'Logout'. The main content area is titled 'Renewal' and contains a 'Renewal Process' section. It features a 'MemberName' input field with the value 'sa' and a 'Search' button. A dropdown menu below the input field lists 'Sai Bag Bag' as a suggestion.

After searching Member and click on Member name all member details are displayed.

The screenshot shows a web browser window titled "GymProject" at the URL "localhost:4200/#/Renewal/Renew". The left sidebar contains links for "Add Member", "Payment Details", "Renew", and "Logout". The main content area is titled "Renewal" and "Renewal Process". It includes fields for "MemberName" (with placeholder "Sai Bag Bag"), "Scheme" (selected as "GYM+CARDIO1"), "PlanName" (selected as "Yearly"), "MemberNo" (text "GYMONE/1/2019"), "New Date" (text "2019-03-30"), "Amount" (text "5393"), and a "Renew" button. To the right, there are fields for "Member Name" (placeholder "Sai Bag Bag"), "Renewal Date" (placeholder "2019-03-25T18:39:14.433"), and another "Renew" button.

Choose a date to renew and click on the Renew button to renew membership.

Wow Finally, we have taken completely go through of application and you can download the entire application and database and play with it also try to make this application better contribute to it on GitHub.