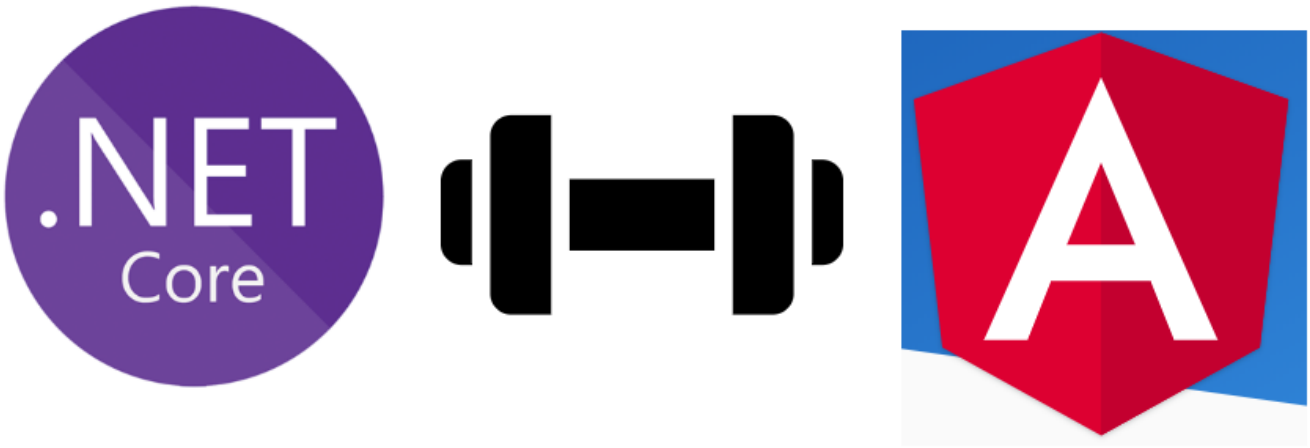
In this article, we are going learn what is this project about for whom it is useful and how you can learn from it. In the end, you will get an idea of how angular project and APIS are used in the project and you can create your own project by seeing this project.



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

Let's see what is this project 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 right 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

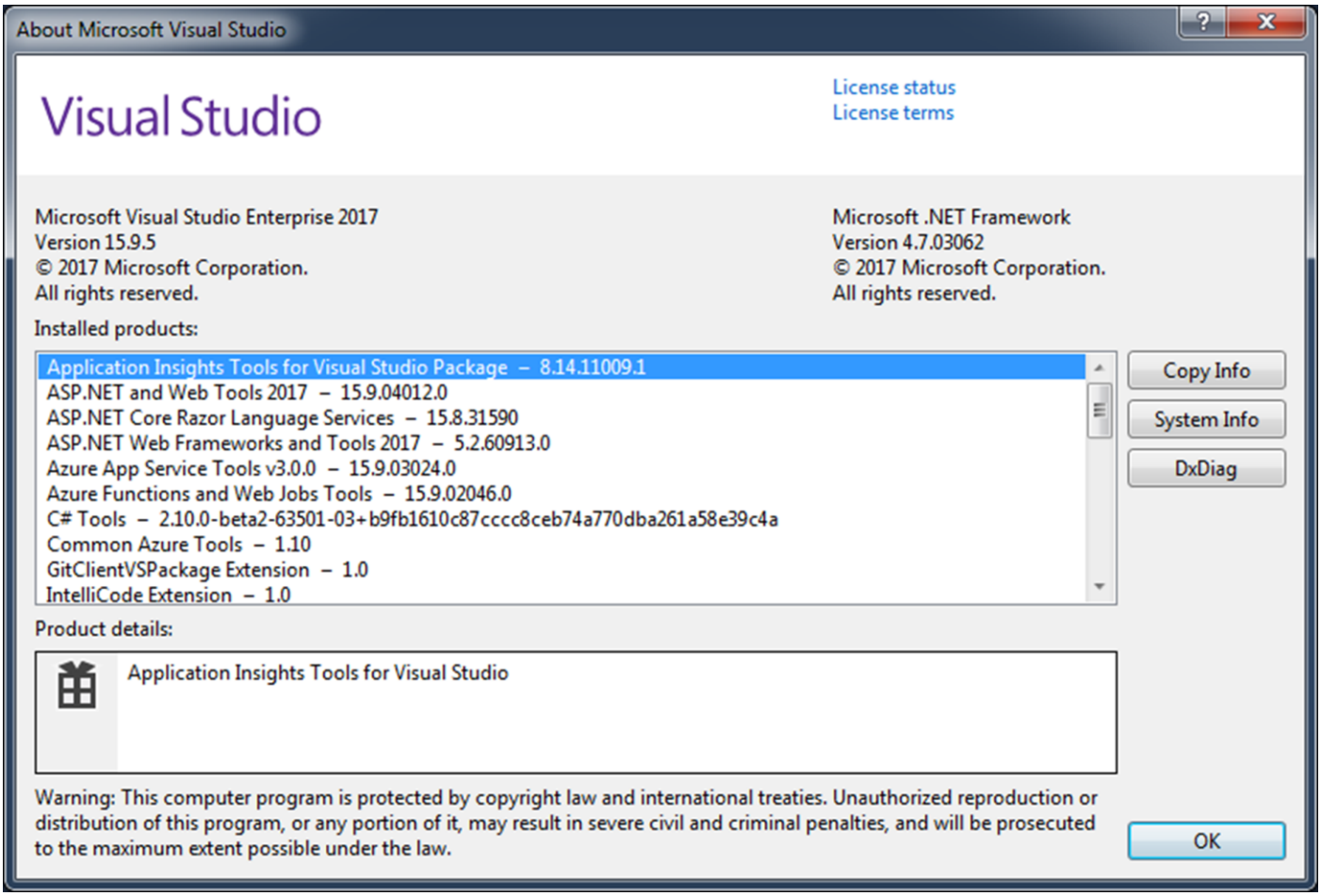
Platform Used

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/

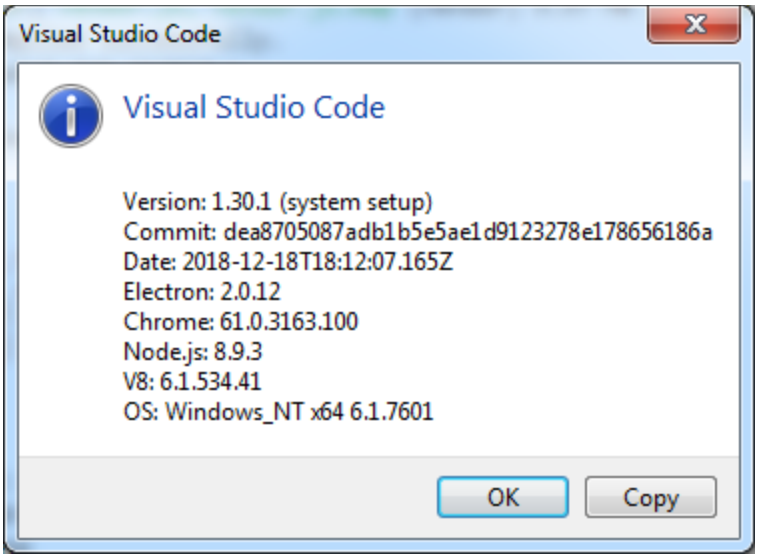


Microsoft SQL Server 2012

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

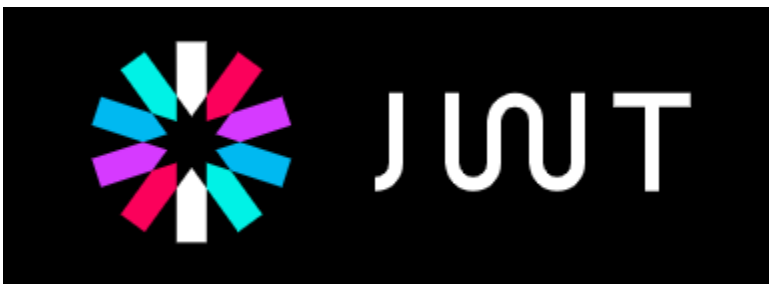


Image Referenced from: - https://jwt.io/

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

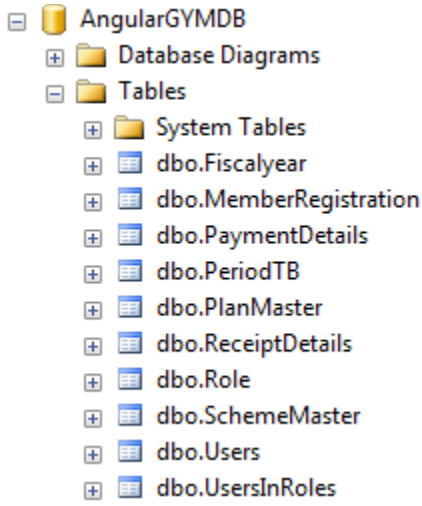
1. @angular/material

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 explorer 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.

SAI-PC\SQLEXPRESS....B - dbo.Fiscalyear

Column Name	Data Type	Allow Nulls
Fid	int	☐
FiscalyearFromDate	datetime	☑
FiscalyearToDate	datetime	☑
Year	varchar(4)	☑
		☐

Member Details

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

SAI-PC\SQLEXPRESS...emberRegistration

Column Name	Data Type	Allow Nulls
MemberId	bigint	☐
MemberNo	nvarchar(20)	☑
MemberFName	nvarchar(100)	☑
MemberLName	nvarchar(100)	☑
MemberMName	nvarchar(100)	☑
DOB	datetime	☑
Age	nvarchar(10)	☑
Contactno	nvarchar(10)	☑
EmailID	nvarchar(30)	☑
Gender	nvarchar(30)	☑
PlanID	int	☑

Payment Details

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

SAI-PC\SQLEXPRES...bo.PaymentDetails

Column Name	Data Type	Allow Nulls
PaymentID	bigint	☐
PlanID	int	☑
WorkouttypeID	int	☑
Paymenttype	nvarchar(50)	☑
PaymentFromdt	datetime	☑
PaymentTodt	datetime	☑
PaymentAmount	numeric(18, 0)	☑
NextRenwalDate	datetime	☑
CreateDate	datetime	☑
Createdby	int	☑
ModifyDate	datetime	☑
ModifiedBy	int	☑

PeriodTB

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

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

Column Name	Data Type	Allow Nulls
PeriodID	int	☐
Text	varchar(50)	☑
Value	varchar(50)	☑
		☐

SchemeMaster

This table contains Schemes.

Example: - GYM+CARDIO1, GYM

SAI-PC\SQLEXPRES...dbo.SchemeMaster

Column Name	Data Type	Allow Nulls
SchemeID	int	☐
SchemeName	nvarchar(50)	☑
Createdby	int	☑
Createddate	datetime	☑
Status	bit	☑
		☐

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.

SAI-PC\SQLEXPRESS...- dbo.PlanMaster

Column Name	Data Type	Allow Nulls
PlanID	int	☐
PlanName	varchar(50)	☑
PlanAmount	decimal(18, 0)	☑
ServicetaxAmount	decimal(18, 0)	☑
ServiceTax	varchar(50)	☑
CreateDate	datetime	☑
CreateUserID	int	☑
ModifyDate	datetime	☑
ModifyUserID	int	☑
RecStatus	bit	☑
SchemeID	int	☑

Role

This table contains Roles.

Example: - Admin, User.

SAI-PC\SQLEXPRES...rGYMDB - dbo.Role

Column Name	Data Type	Allow Nulls
RoleId	int	☐
RoleName	nvarchar(256)	☐
Status	bit	☑
		☐

UsersInRoles

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

Example: - Admin, User.

SAI-PC\SQLEXPRESS...- dbo.UsersInRoles

Column Name	Data Type	Allow Nulls
UserRolesId	int	☐
UserId	int	☑
RoleId	int	☑
		☐

Users

This table contains Users Login Credentials.

SAI-PC\SQLEXPRES...GYMDB - dbo.Users

Column Name	Data Type	Allow Nulls
UserId	int	☐
UserName	nvarchar(56)	☐
FullName	nvarchar(200)	☑
EmailId	nvarchar(200)	☑
Contactno	nvarchar(10)	☑
Password	nvarchar(200)	☑
Createdby	int	☑
CreatedDate	datetime	☑
Status	bit	☑
		☐

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

API Project Structure

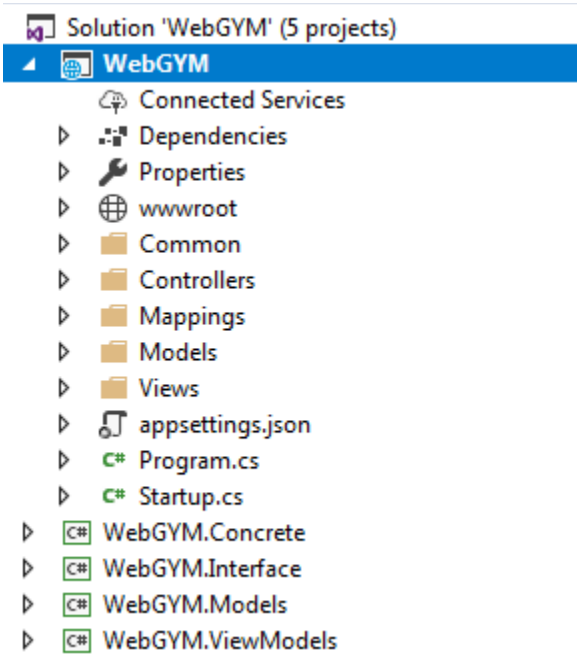
In this part, we are going to have a look into the API project in details. APIs Project Name is WebGYM which is developed in ASP.NET CORE Framework version 2.1.

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 as 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 orm to access the database.

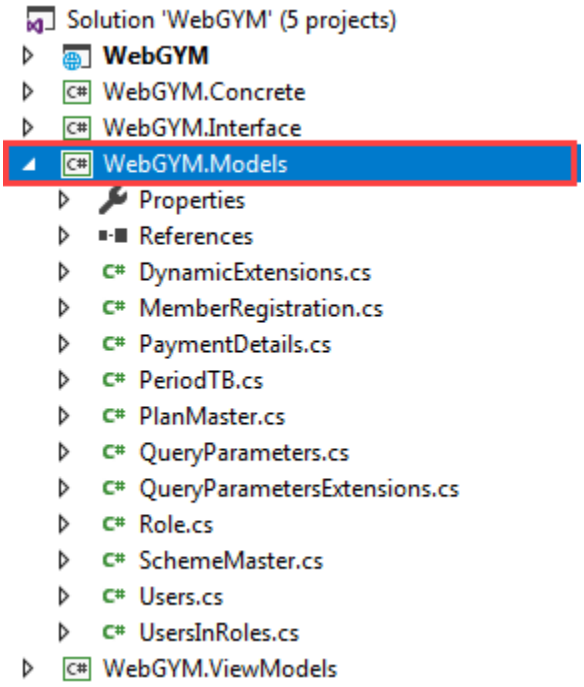
Added Asp.net MVC Core Web Project.

WebGYM



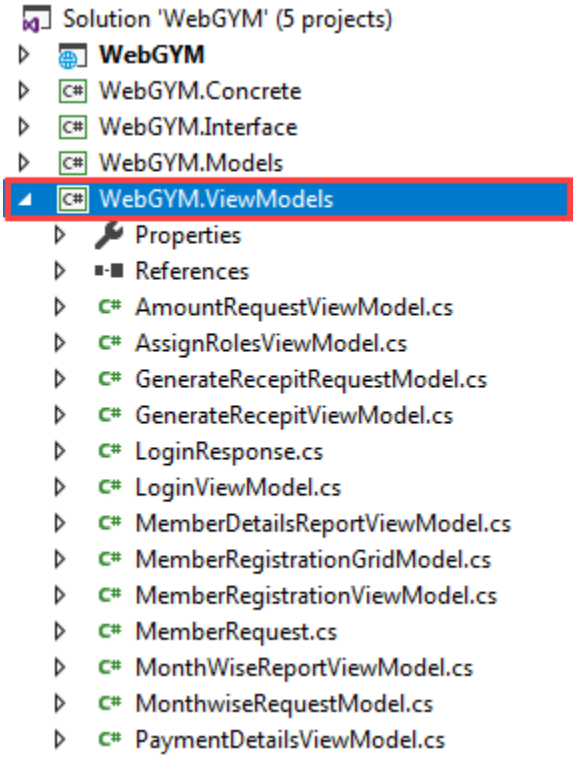
Next, we are going to have a look at WebGYM.Models Class Library.

WebGYM.Models



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

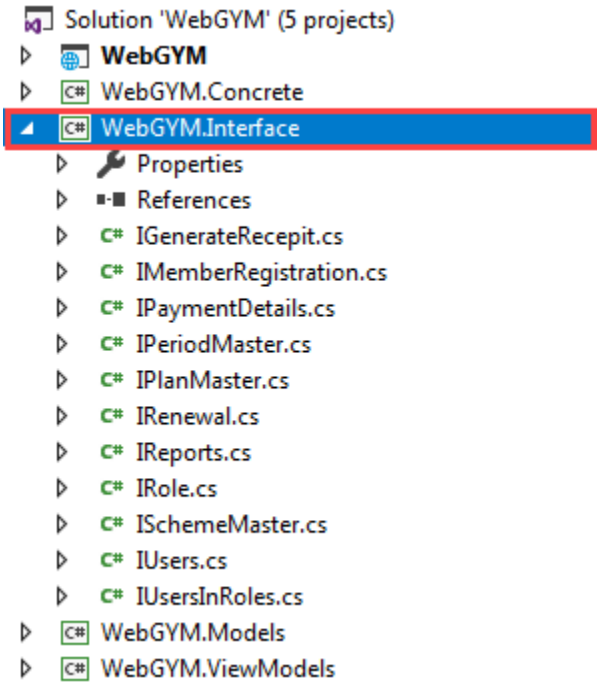
WebGYM.ViewModels



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

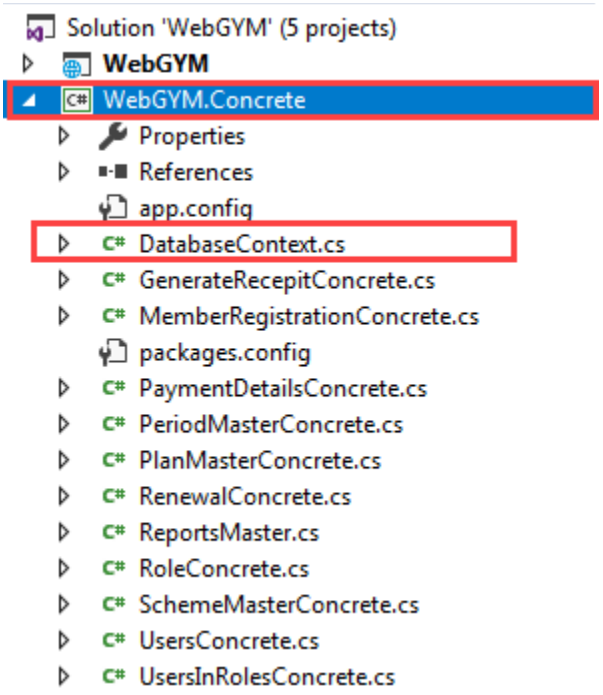
This Class Library contains all Interfaces in which we have declared methods, this interface is going to implement a concrete class (WebGYM.Concrete).

WebGYM.Interface



WebGYM.Concrete

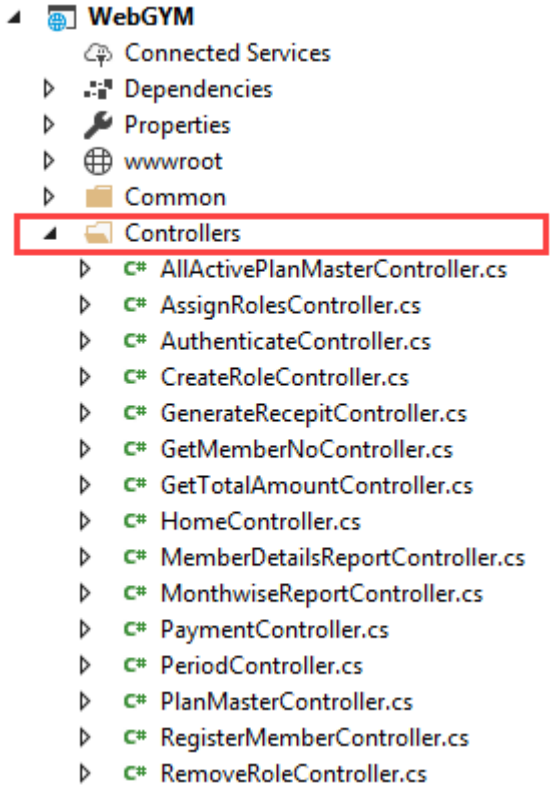
The WebGYM.Concrete Class Library contains all Classes which are using Entity Framework Core ORM for accessing database and it also has DatabaseContext and is the main class for interacting with the database. I have also used Dapper Orm for some database operations.



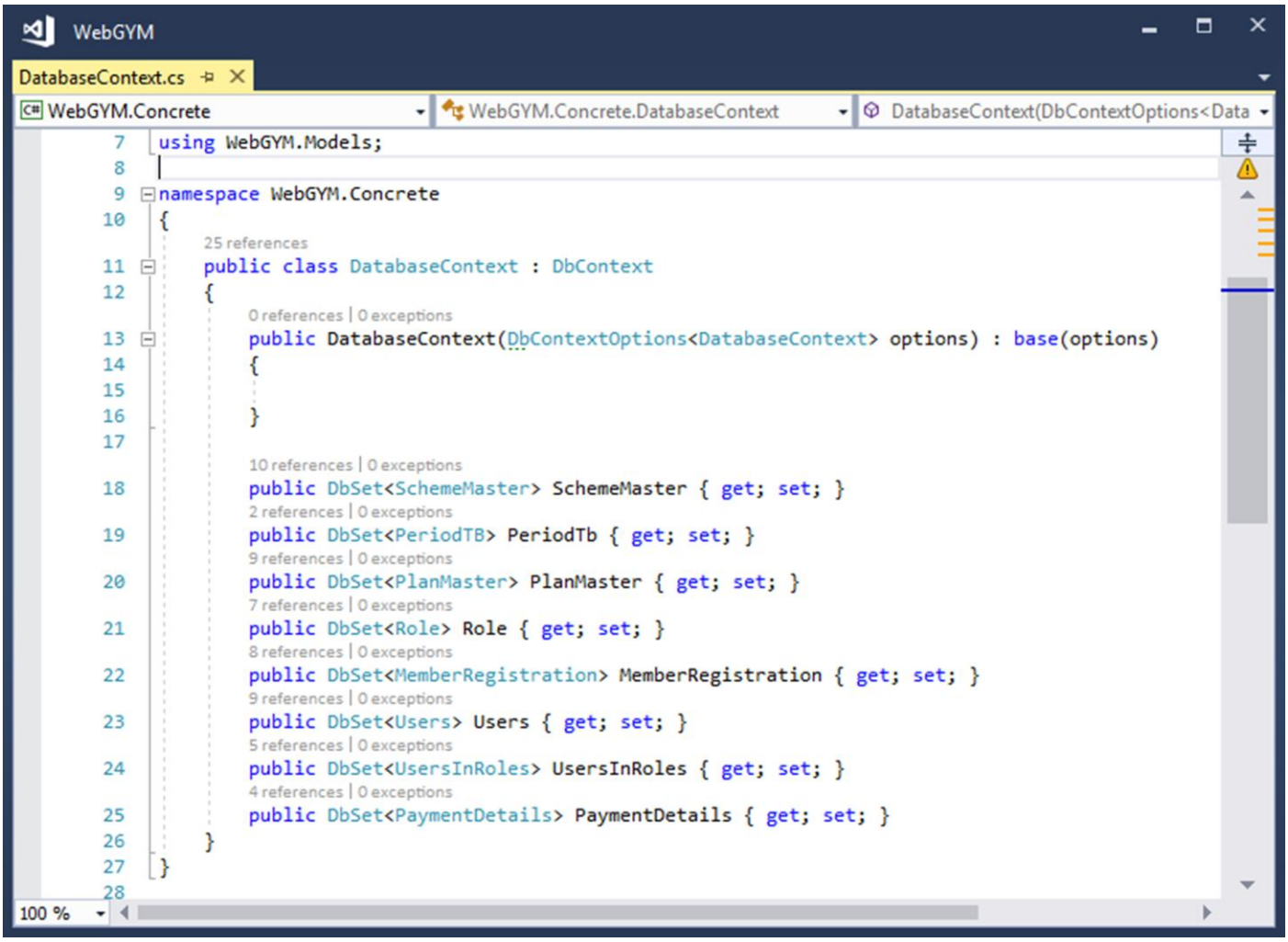
After Completing View of Class Library Next, we are going to View API Controller Structure.

Controllers

Controller Folder contains all API Controllers created in this application.

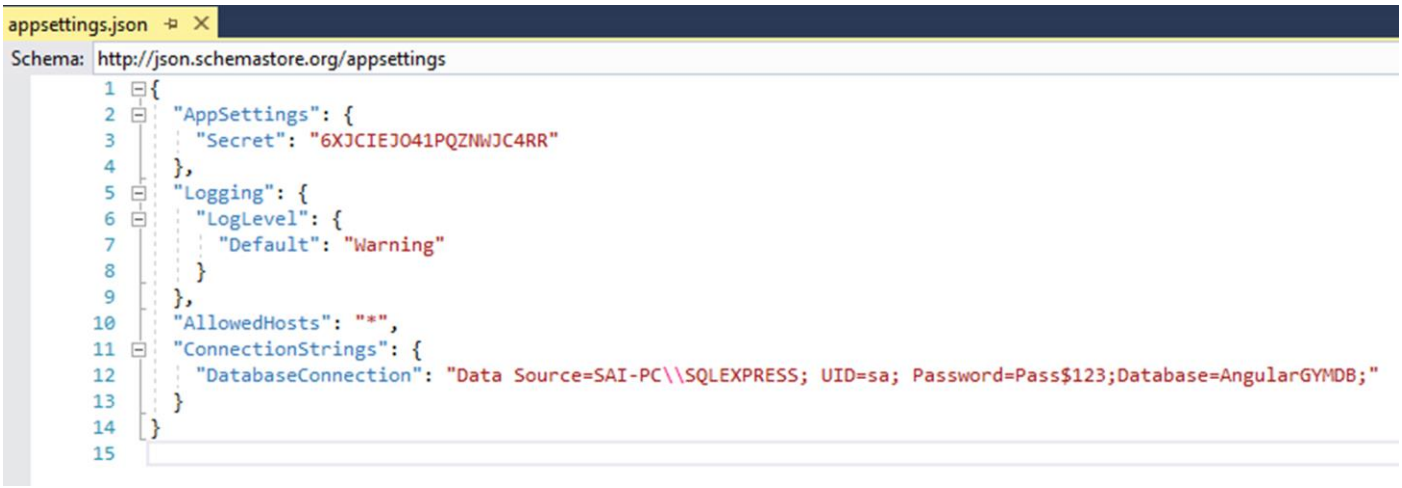


DbContext (we are using Entity Framework core in this project)

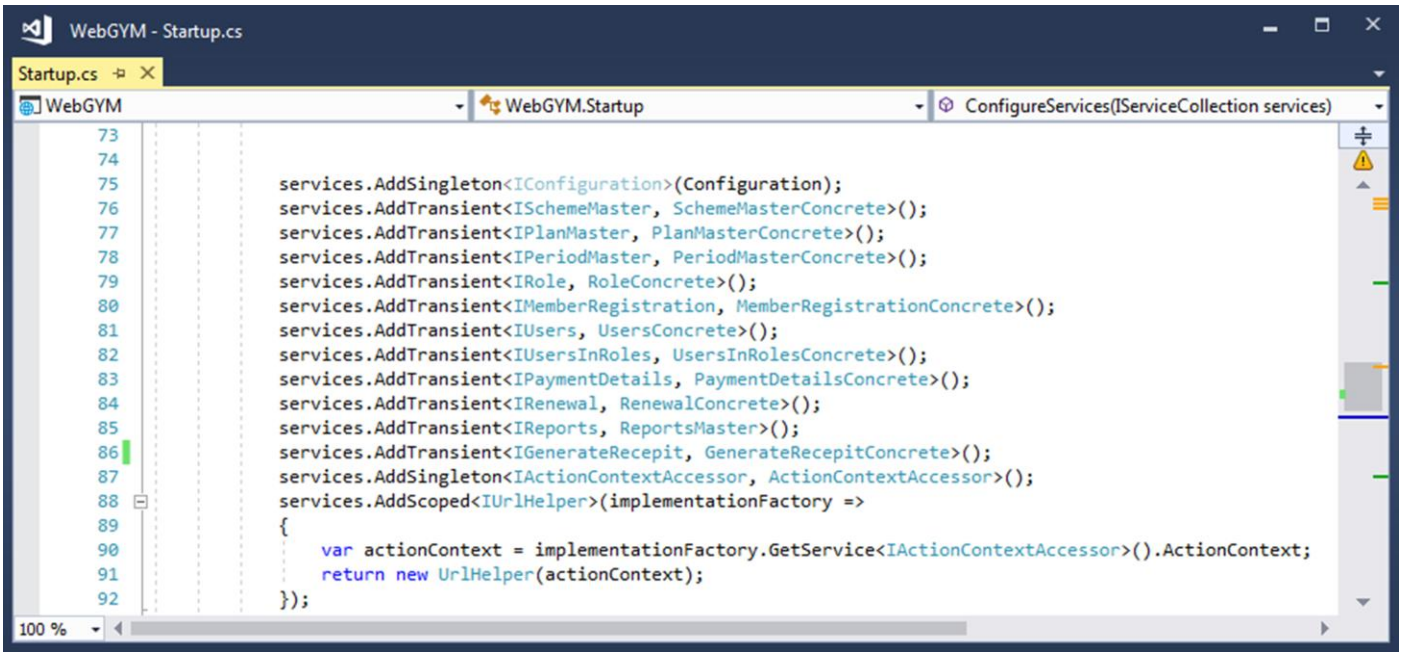


Appsettings.json file

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



Setting Dependence injection in Startup.cs class

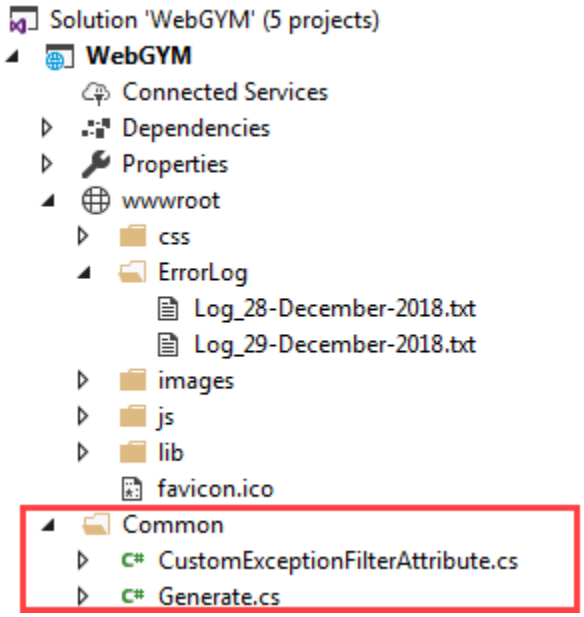


Filters and Encryption library

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.

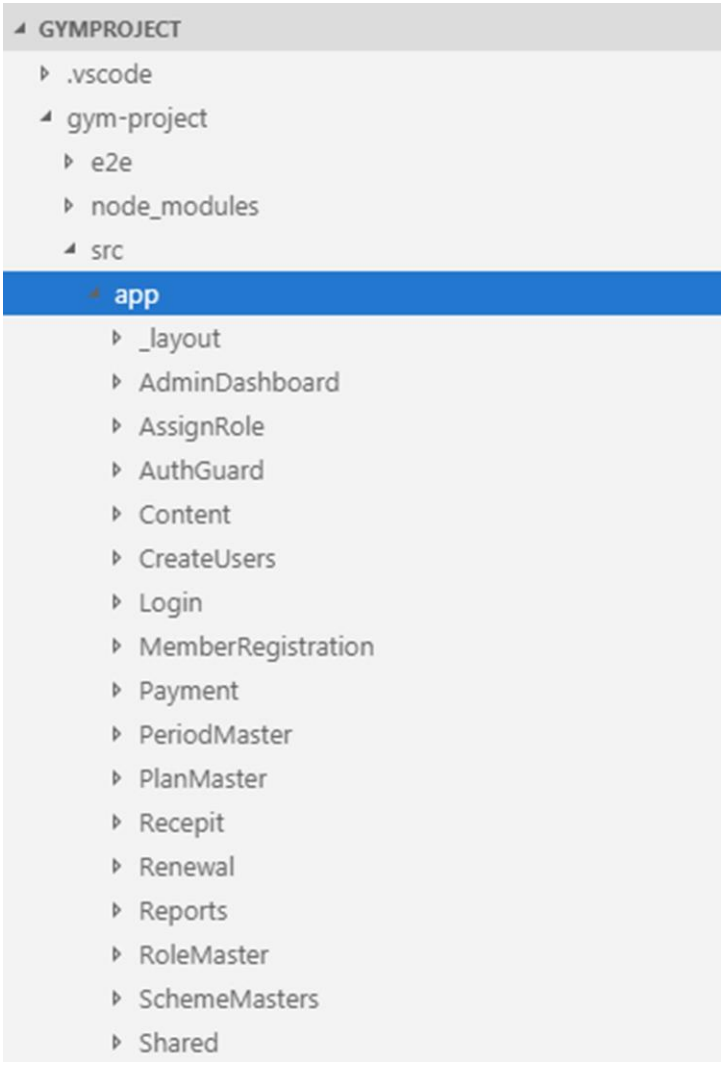


The above shows how the directory and folder structure look, now let's check out Application screens.

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

Angular Application Project Structure

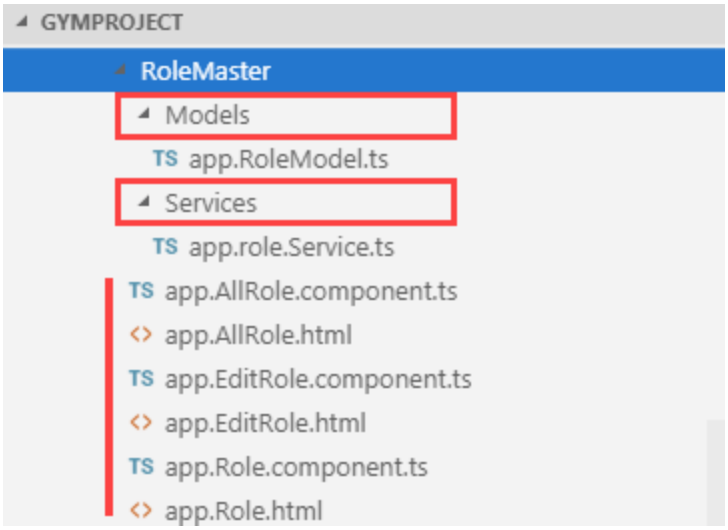
Below is a complete view of Angular Application project structure, if you see below snapshot you will see all folder with proper naming which tells which folder contains which Component in it.



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, HttpErrorResponse, 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>
                <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>
                <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>
        </form>

    </div>
</div>
```

package.json

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

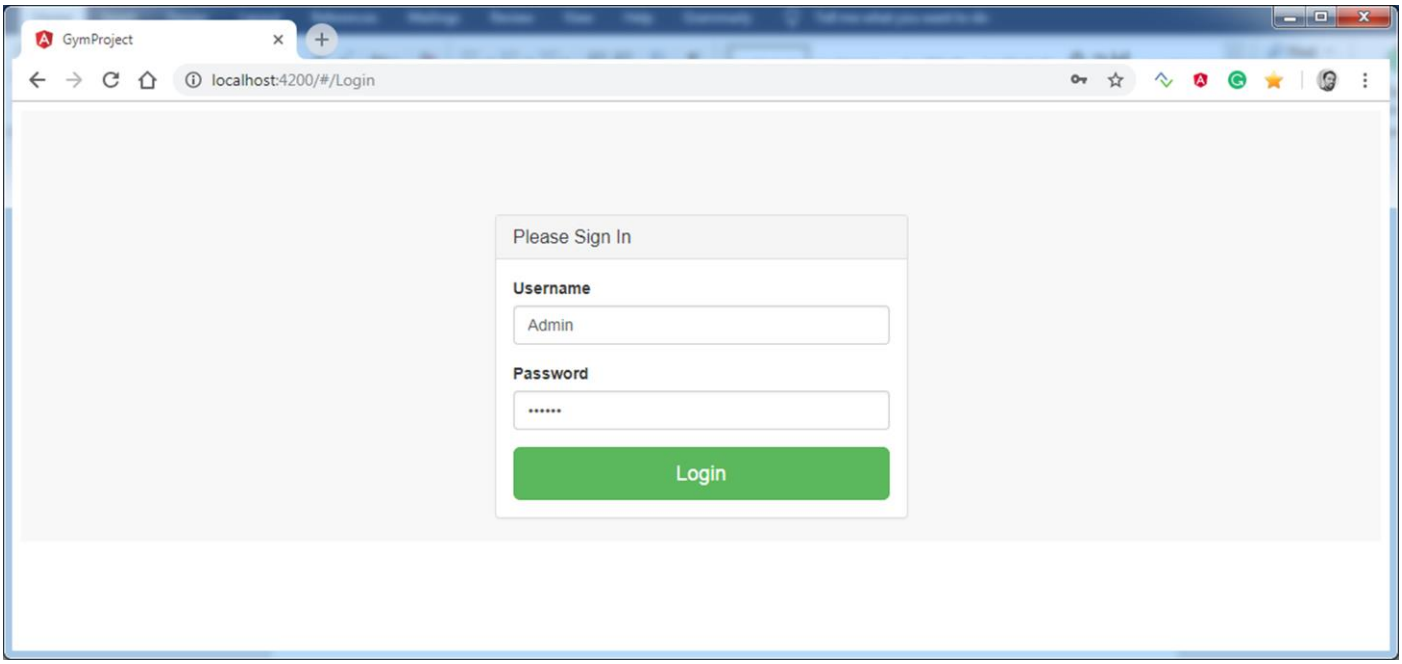
```
{
  "name": "gym-project",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^7.0.4",
    "@angular/cdk": "^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.

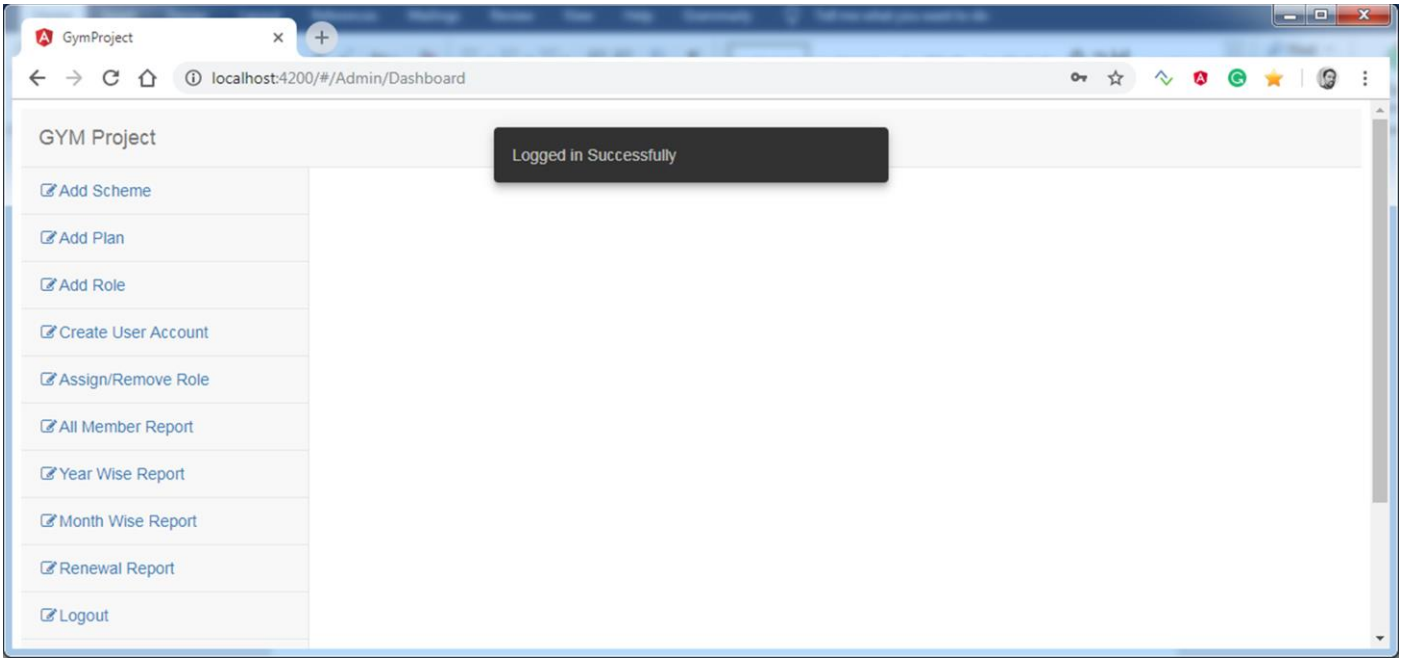
Admin Application Screens

Login is Common for Admin and User.

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



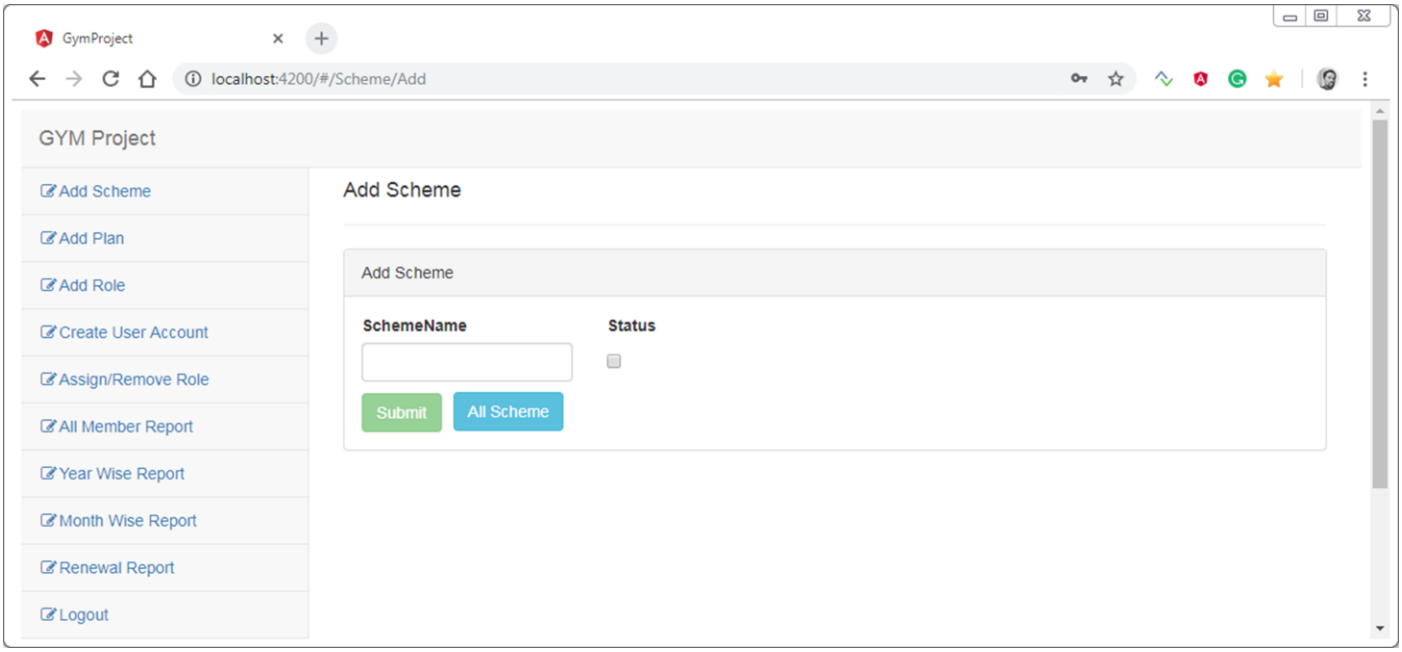
After logging into application first screen appears is your dashboard.



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

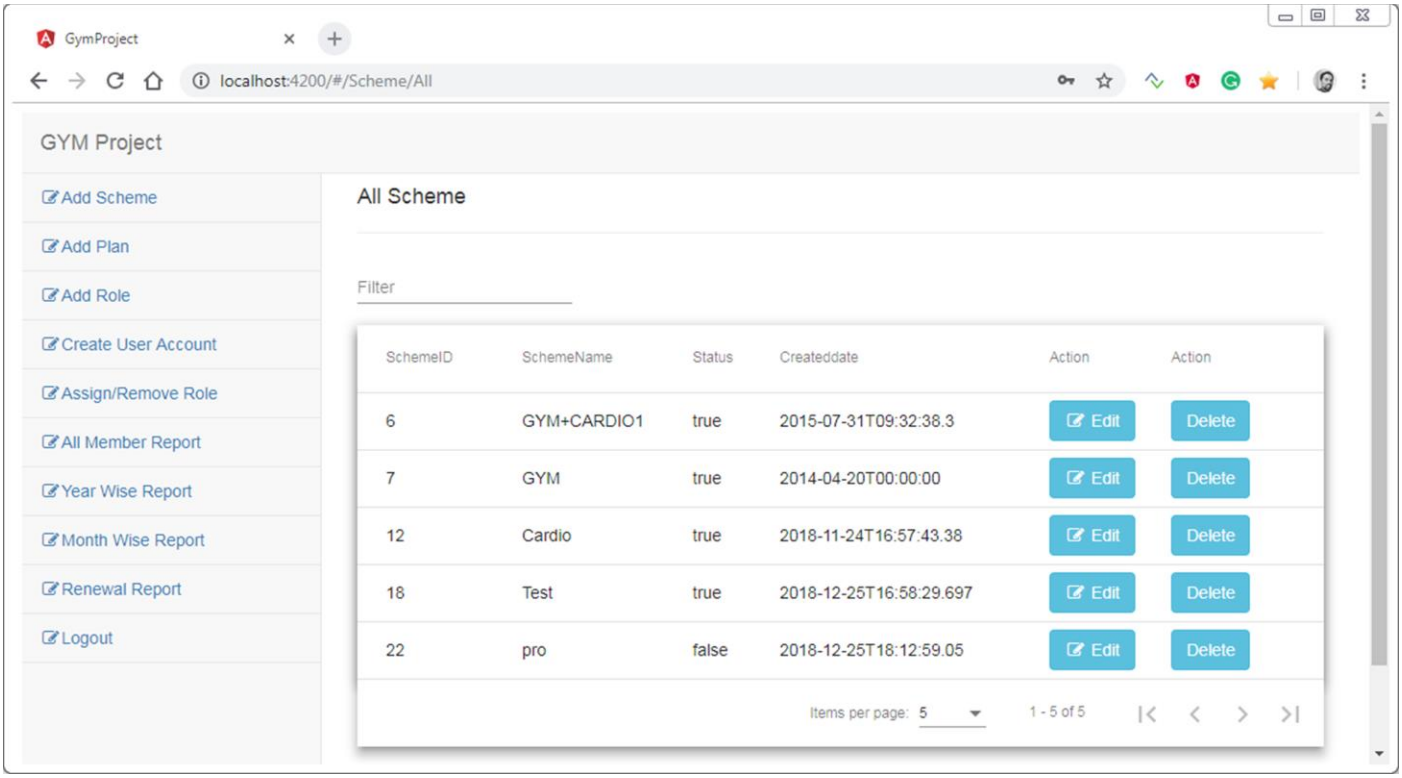
Add Scheme

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



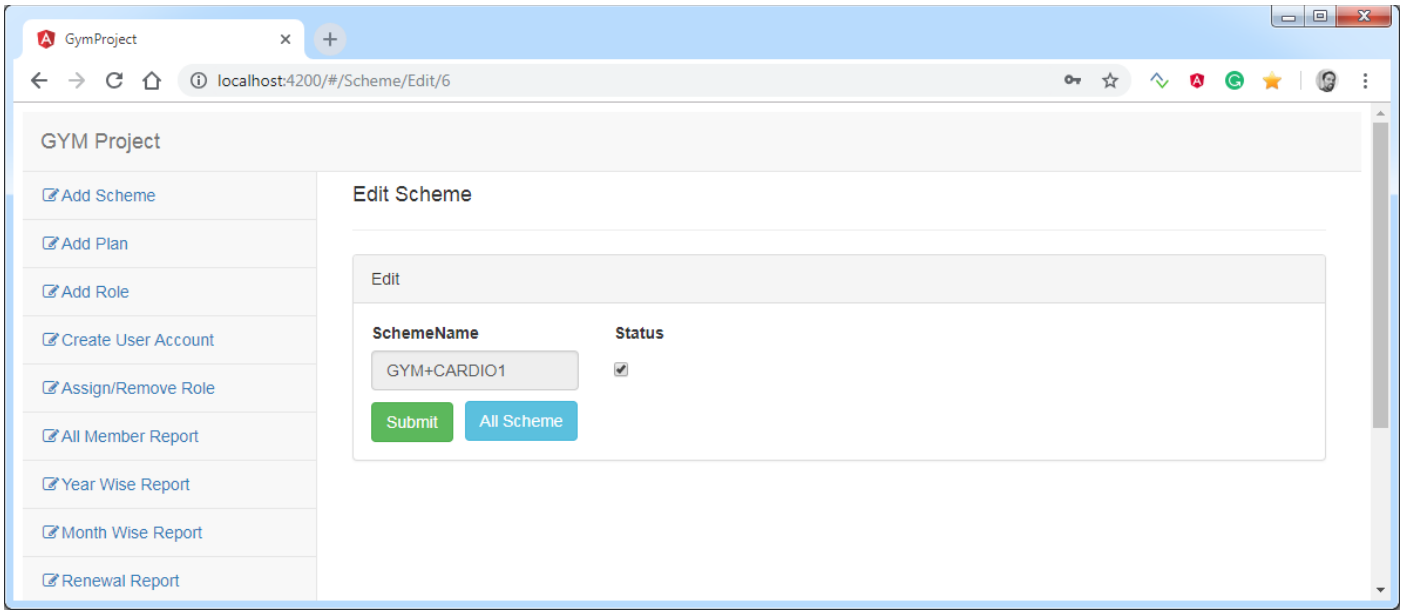
All Scheme

Displays all scheme which is added.



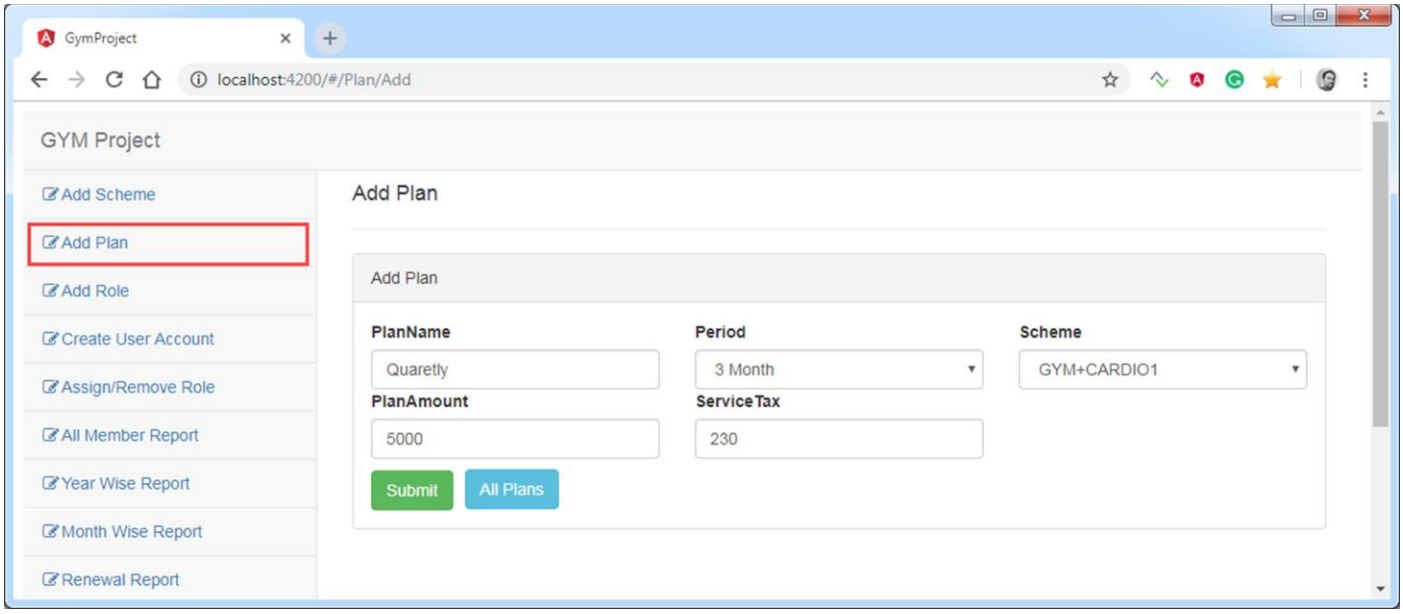
Edit Scheme

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.



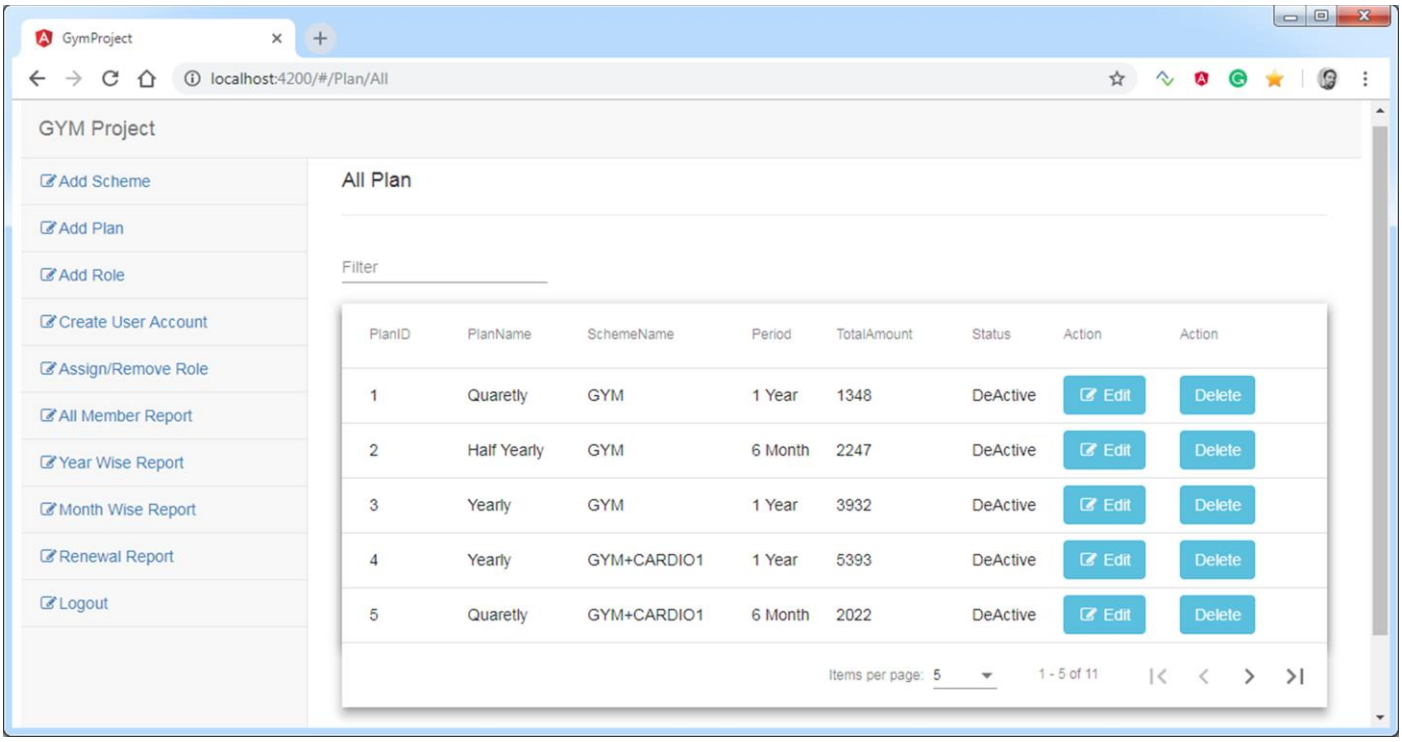
Add Plan

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.



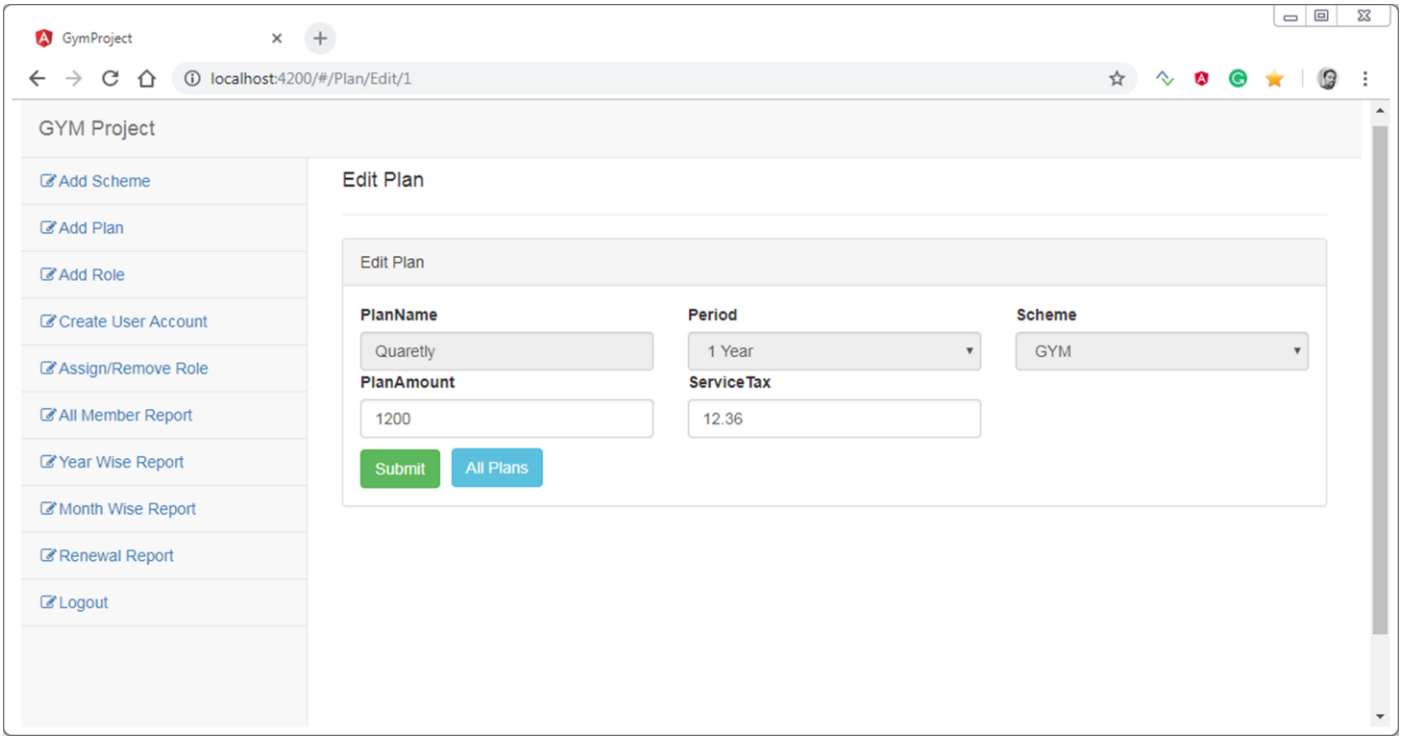
All Plan

Displays all Plans which are added.



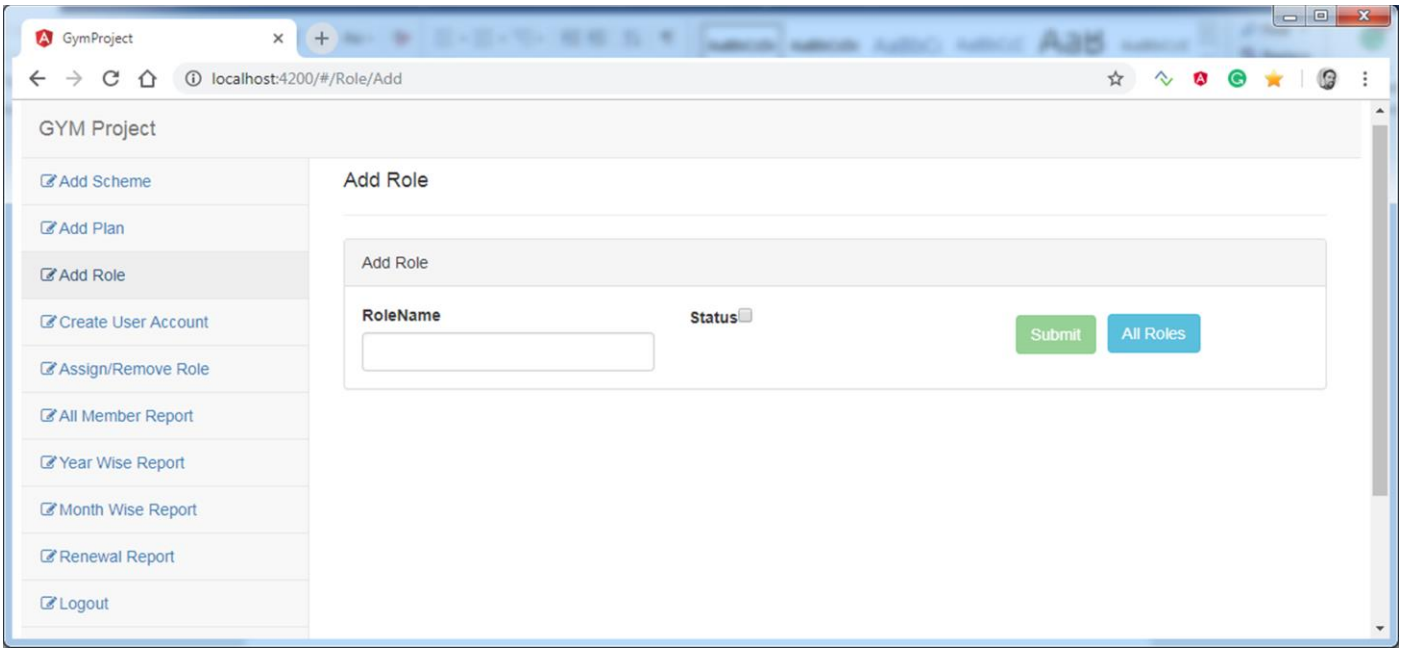
Edit Plan

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



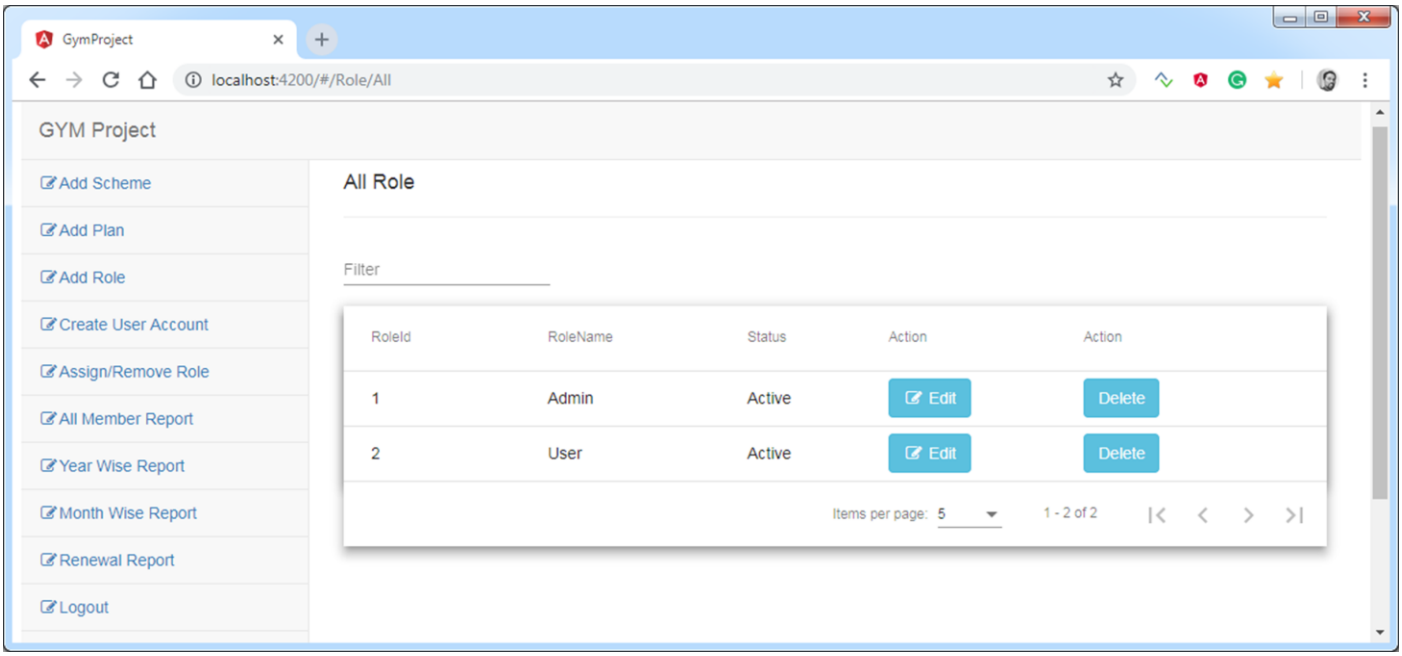
Add Role

In this part, we are going to add new roles



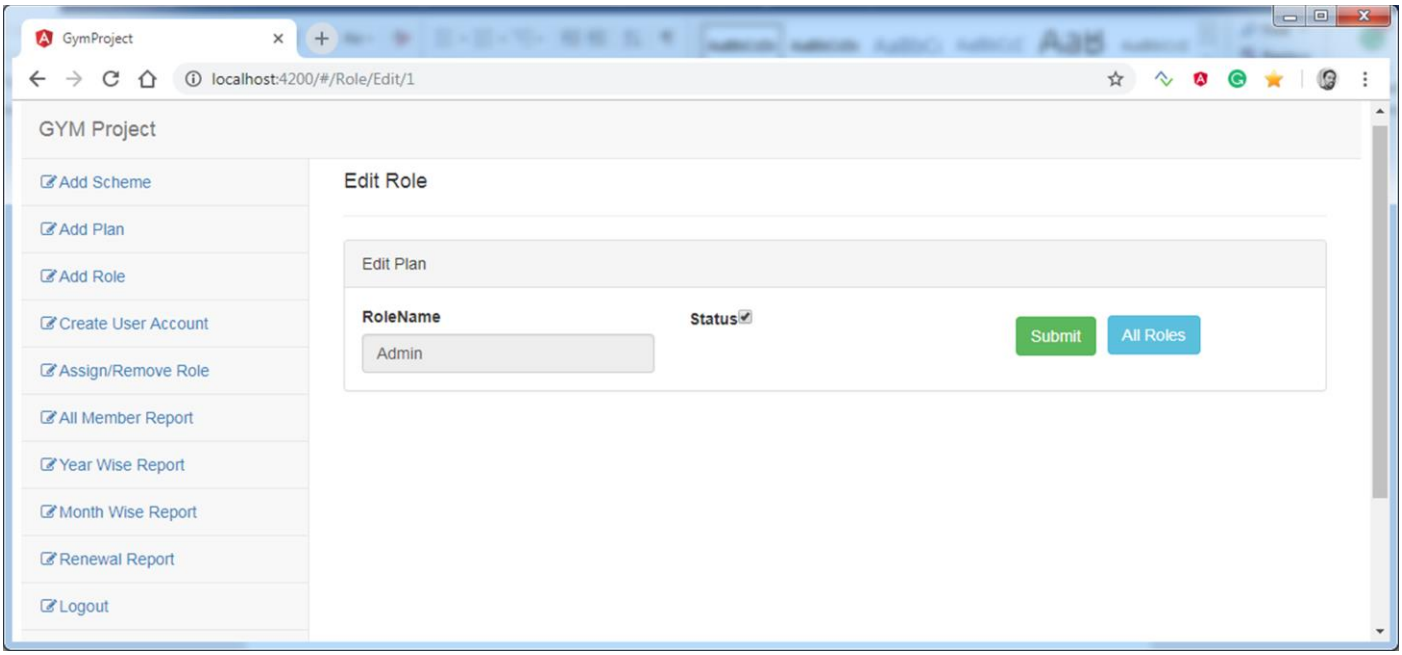
All Role

Displays all Roles which are added.



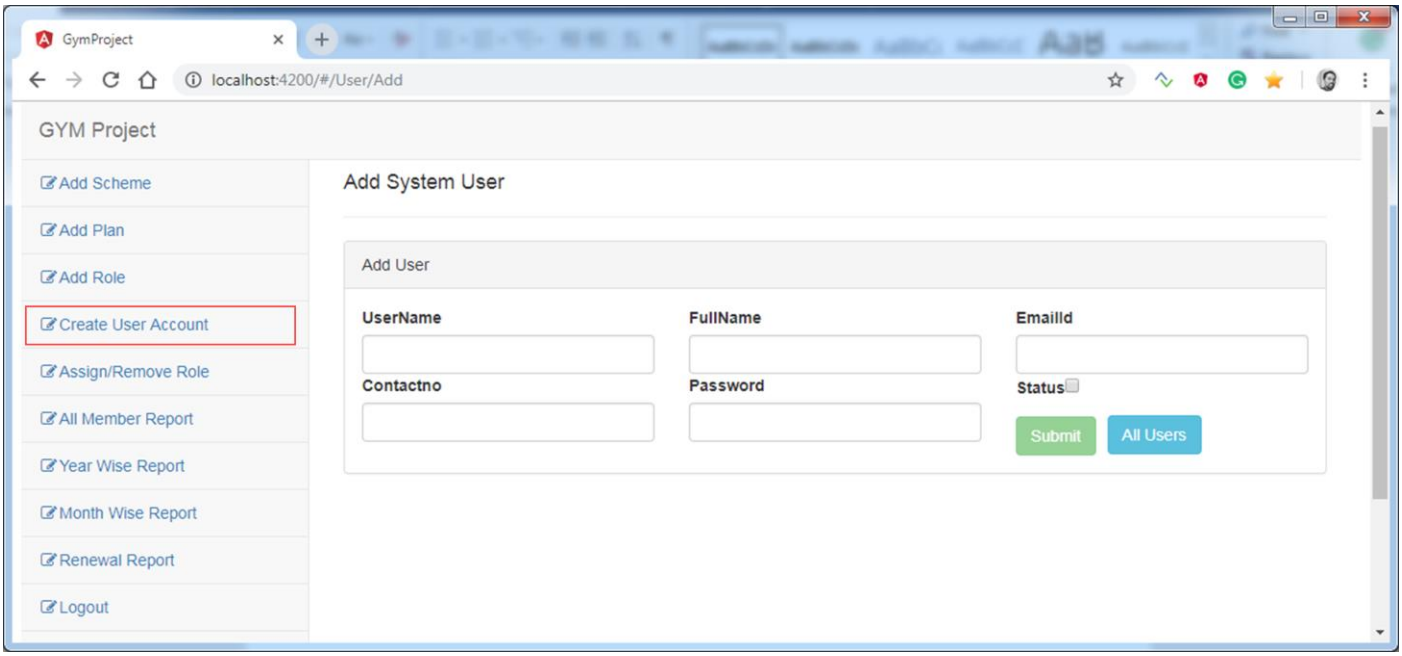
Edit Role

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



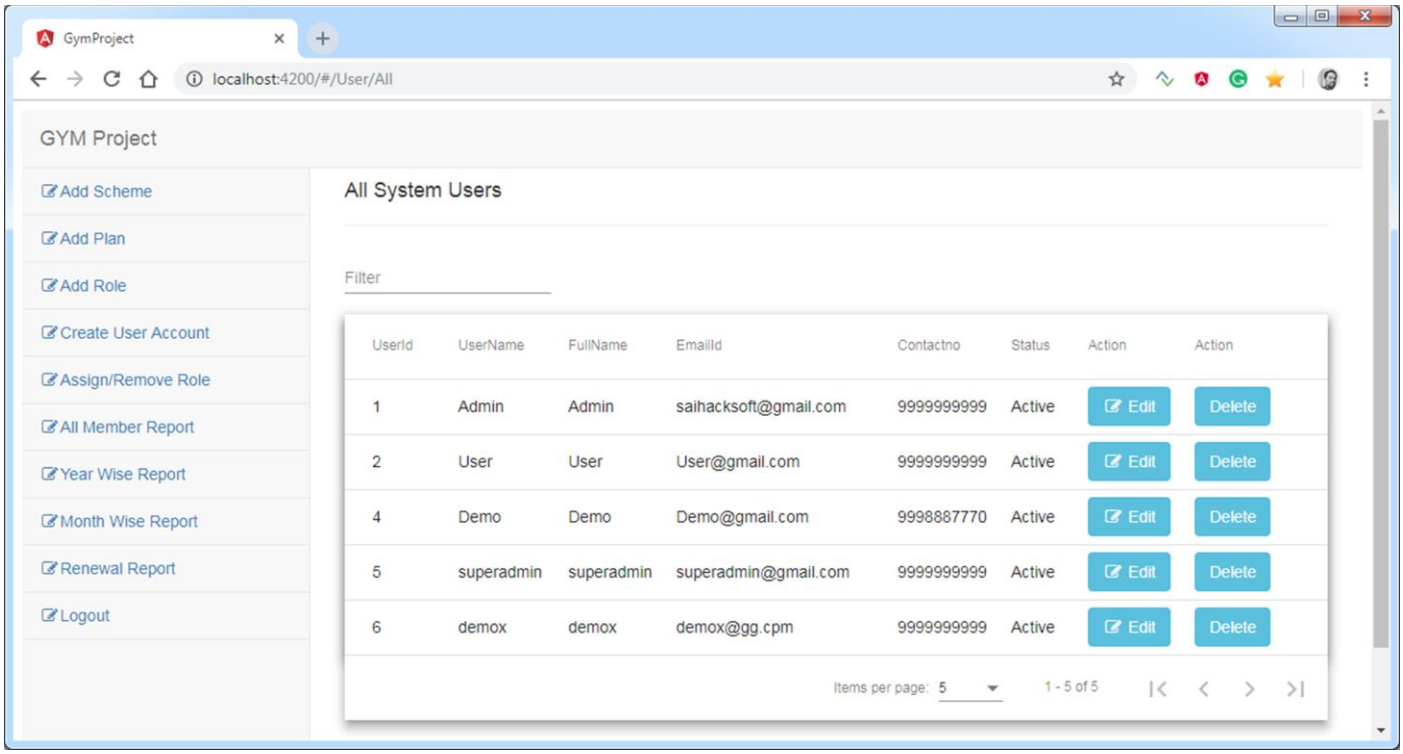
Create New User

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



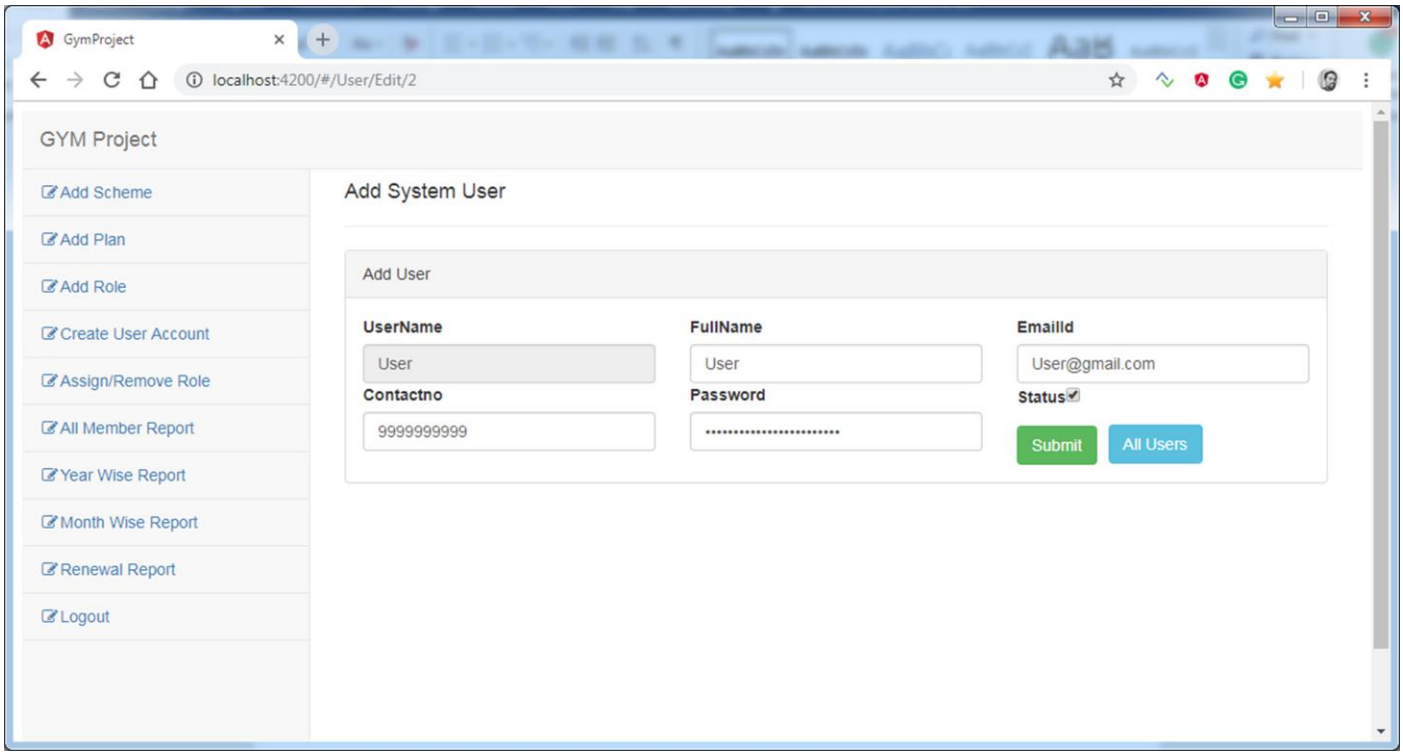
All Users

Displays all Users which are added.



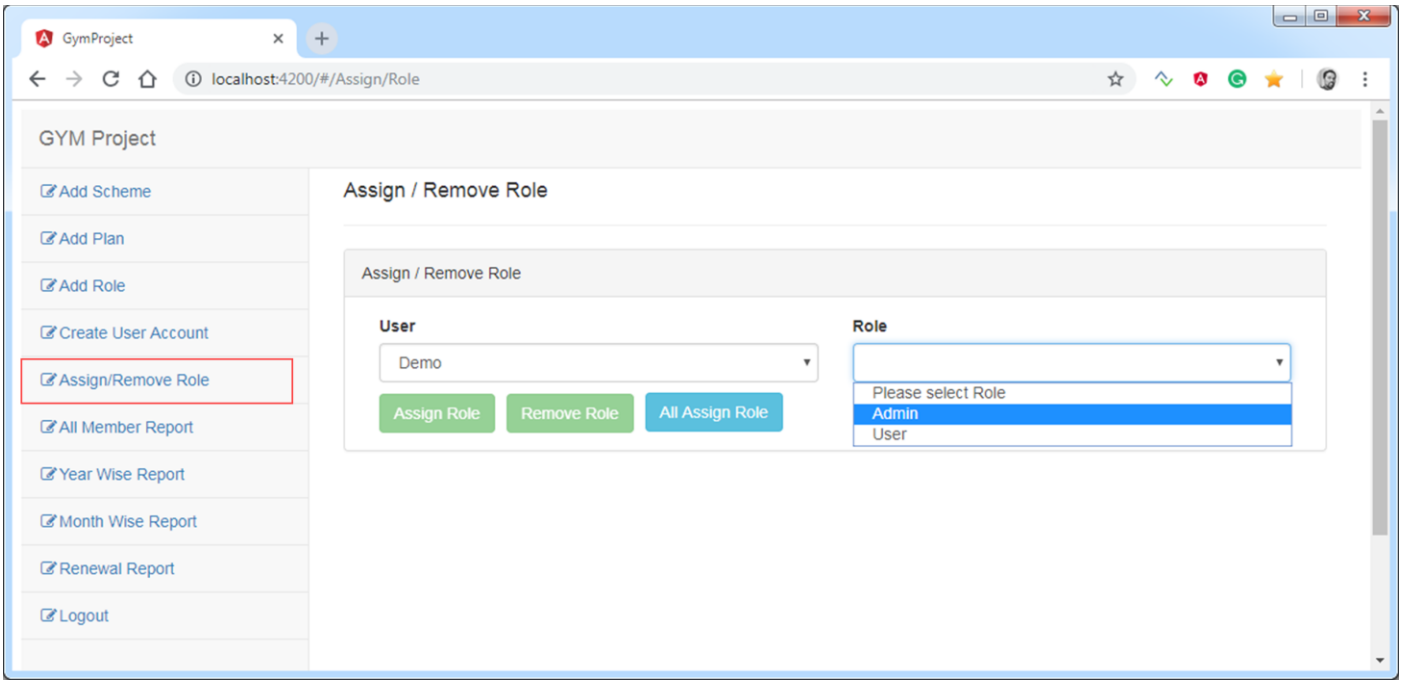
Edit Users

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



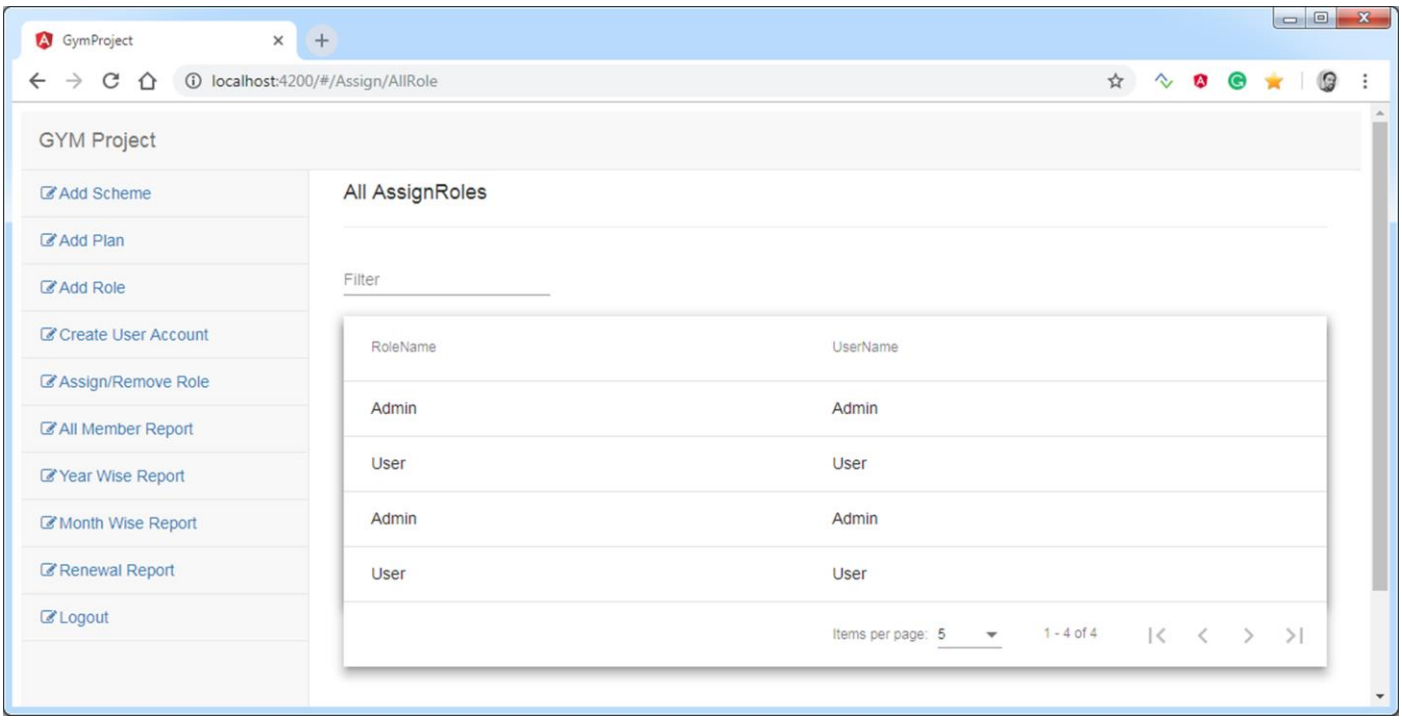
Assign / Remove Role

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



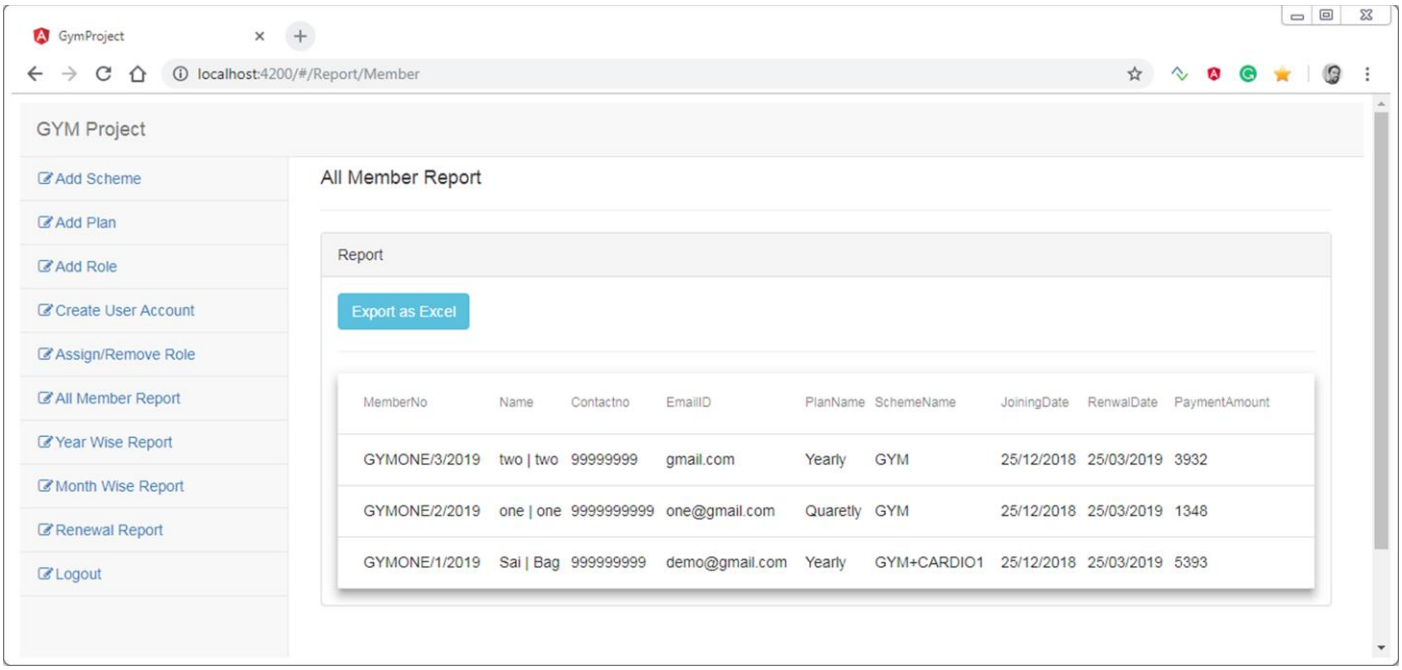
All Assigned Roles

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

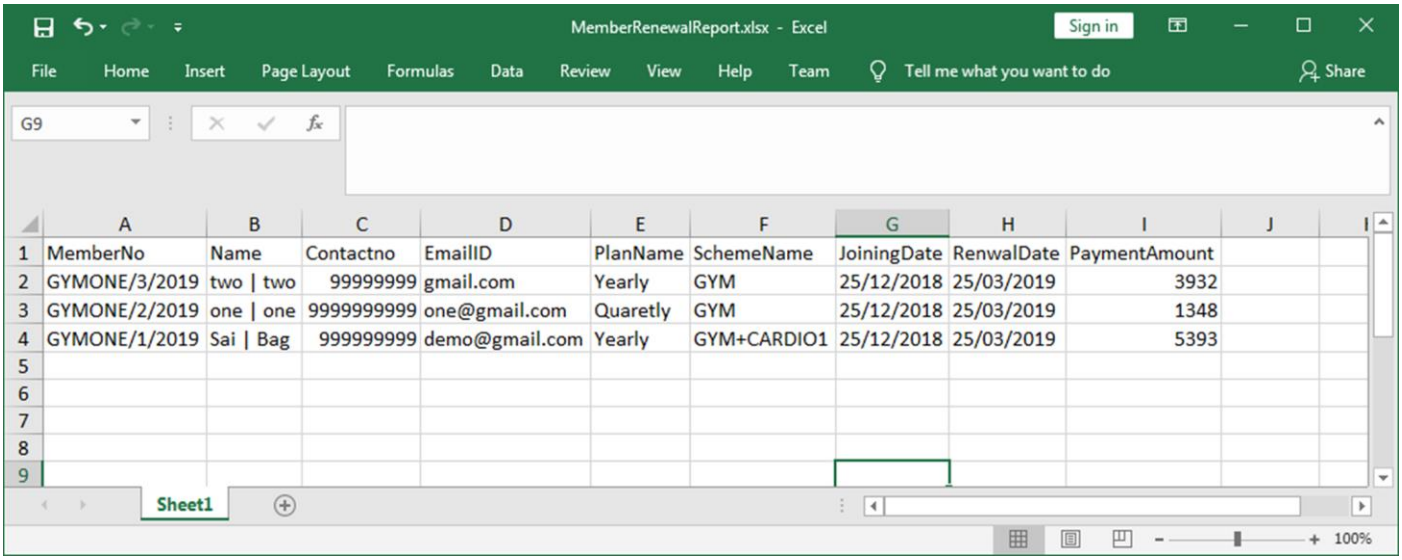


All Member Report

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

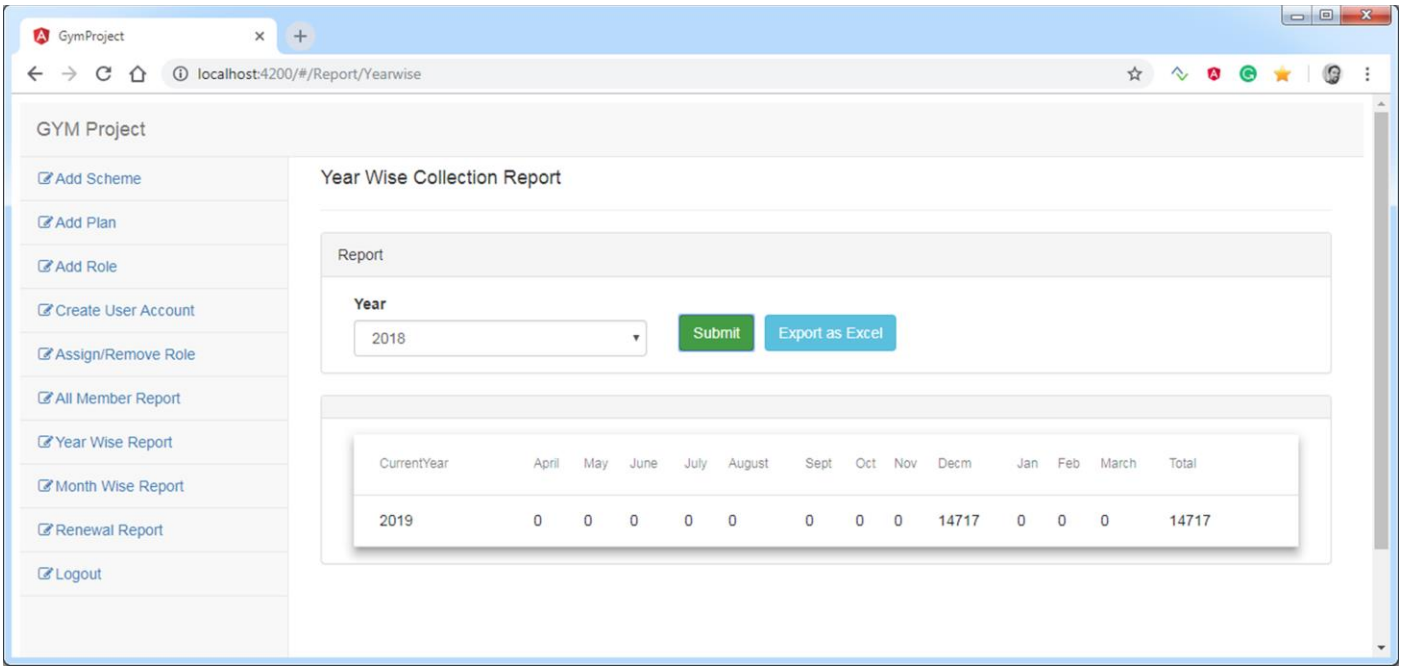


Exported all Member Report in Excel

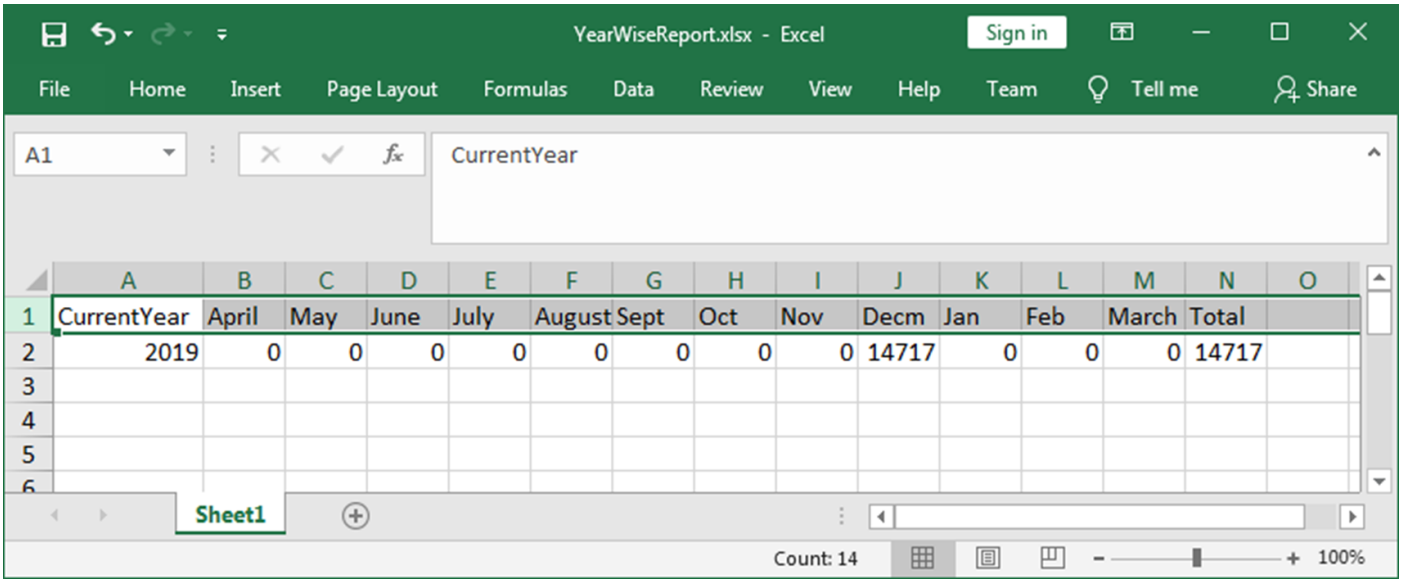


Year wise Collection Report

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

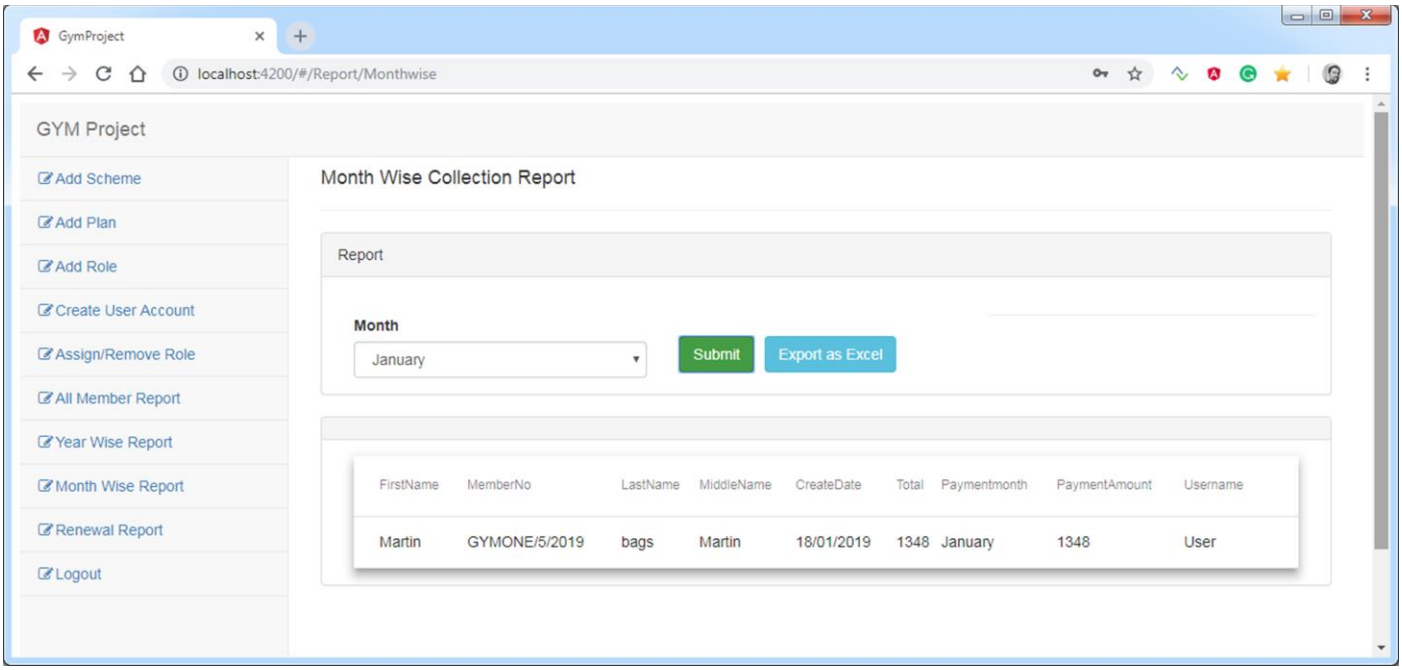


Exported Year Wise Collection Report in Excel

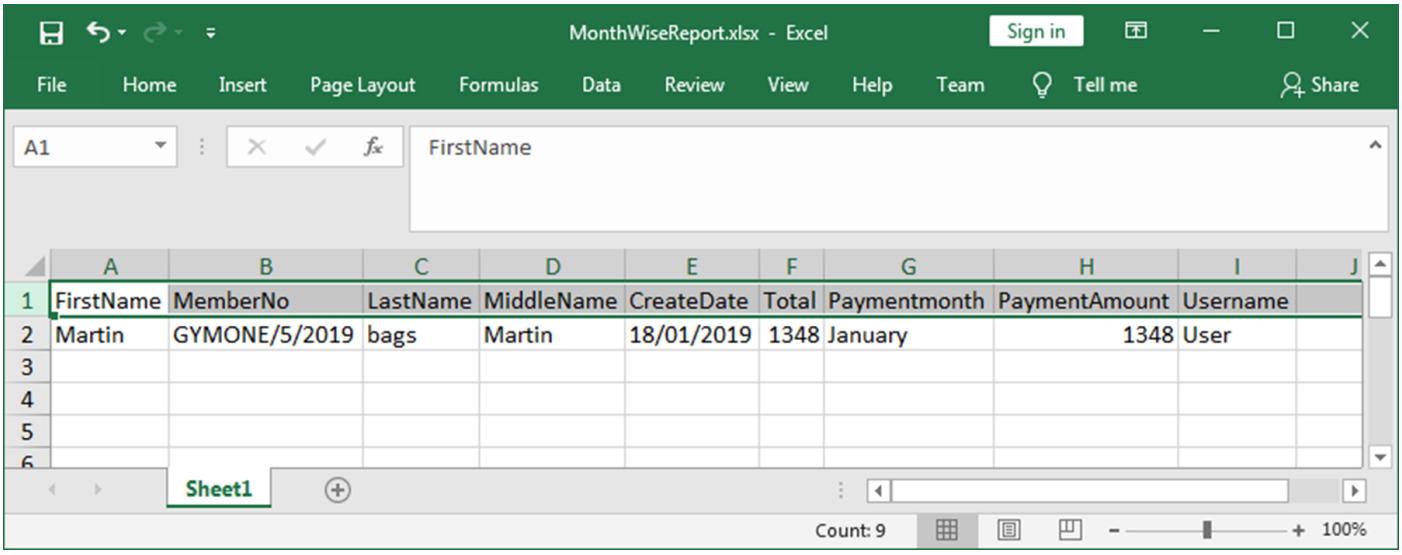


Month-wise Collection Report

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

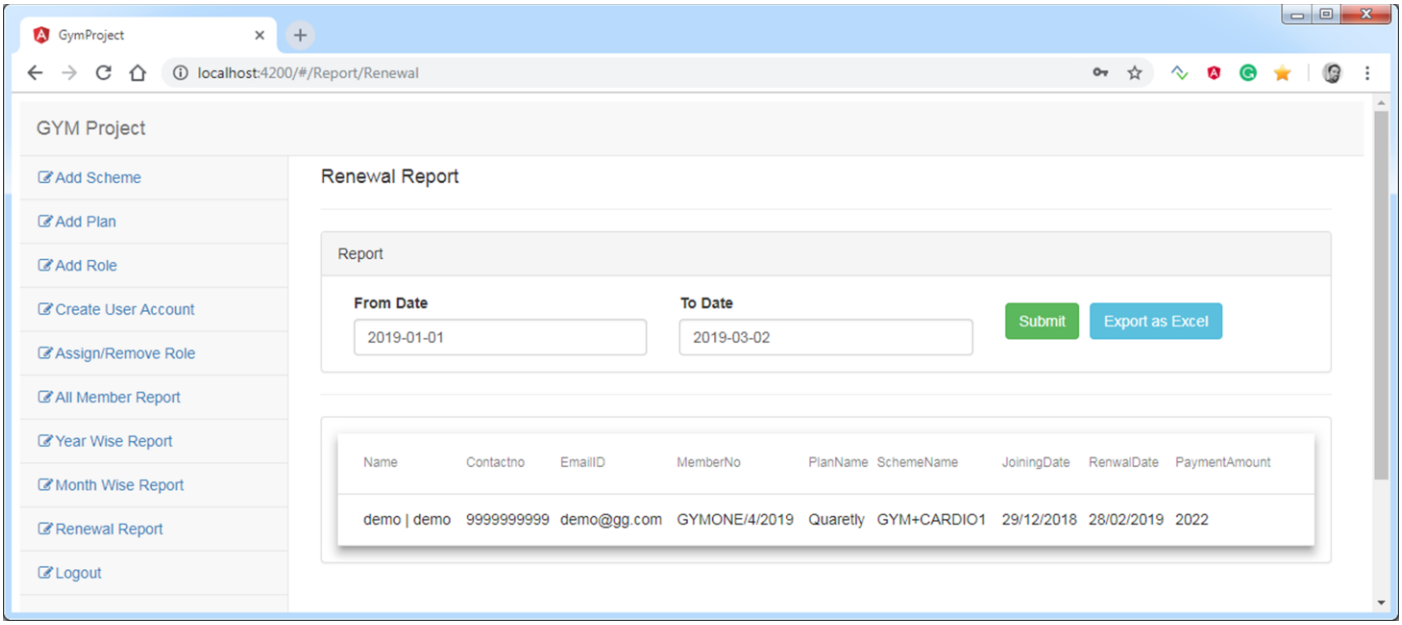


Exported Month Wise Collection Report in Excel

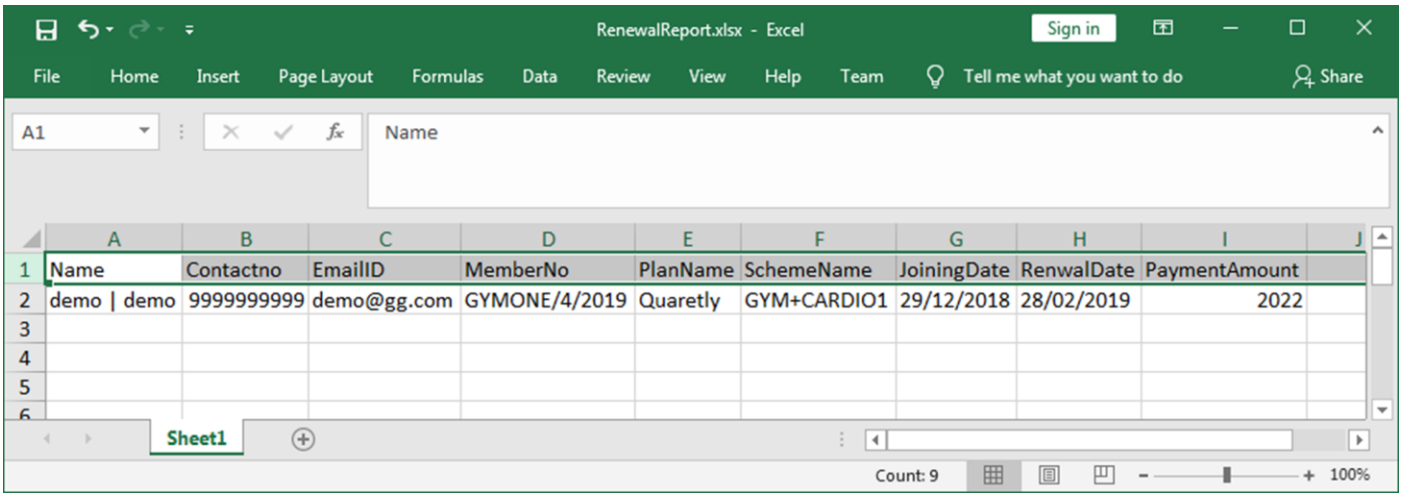


Renewal Report

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



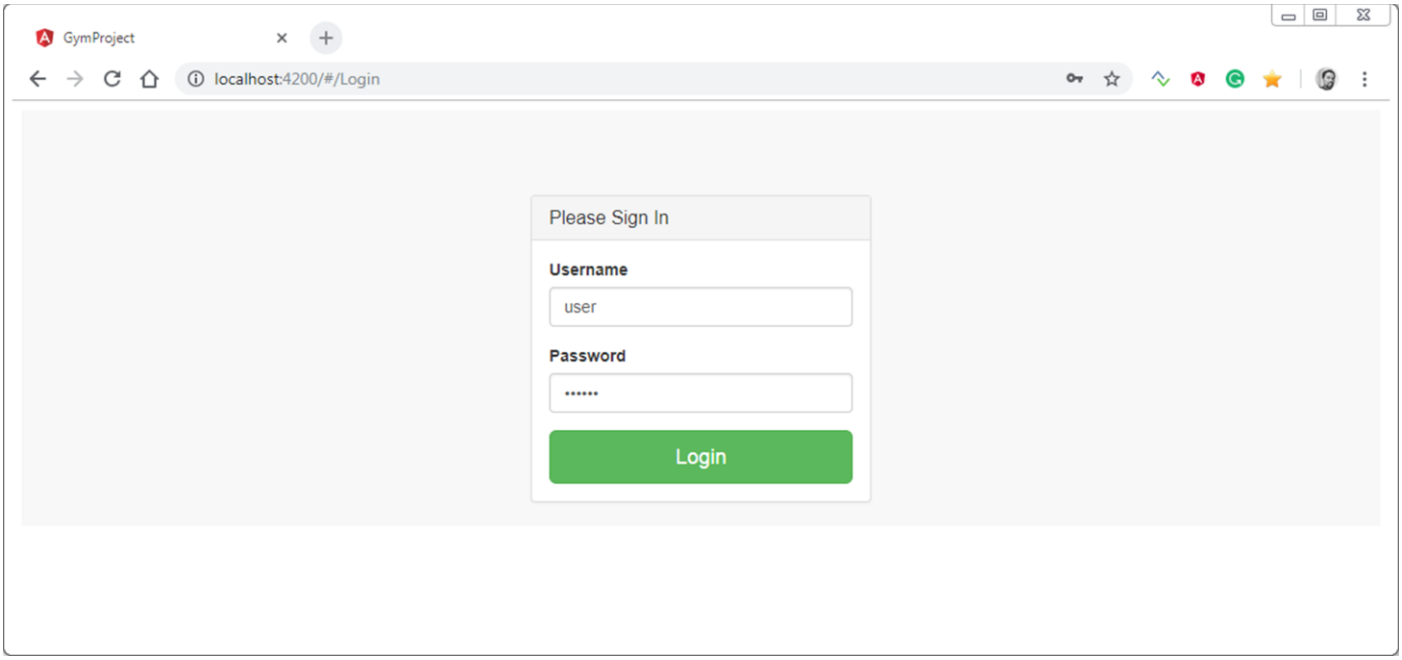
Exported Renewal Report in Excel



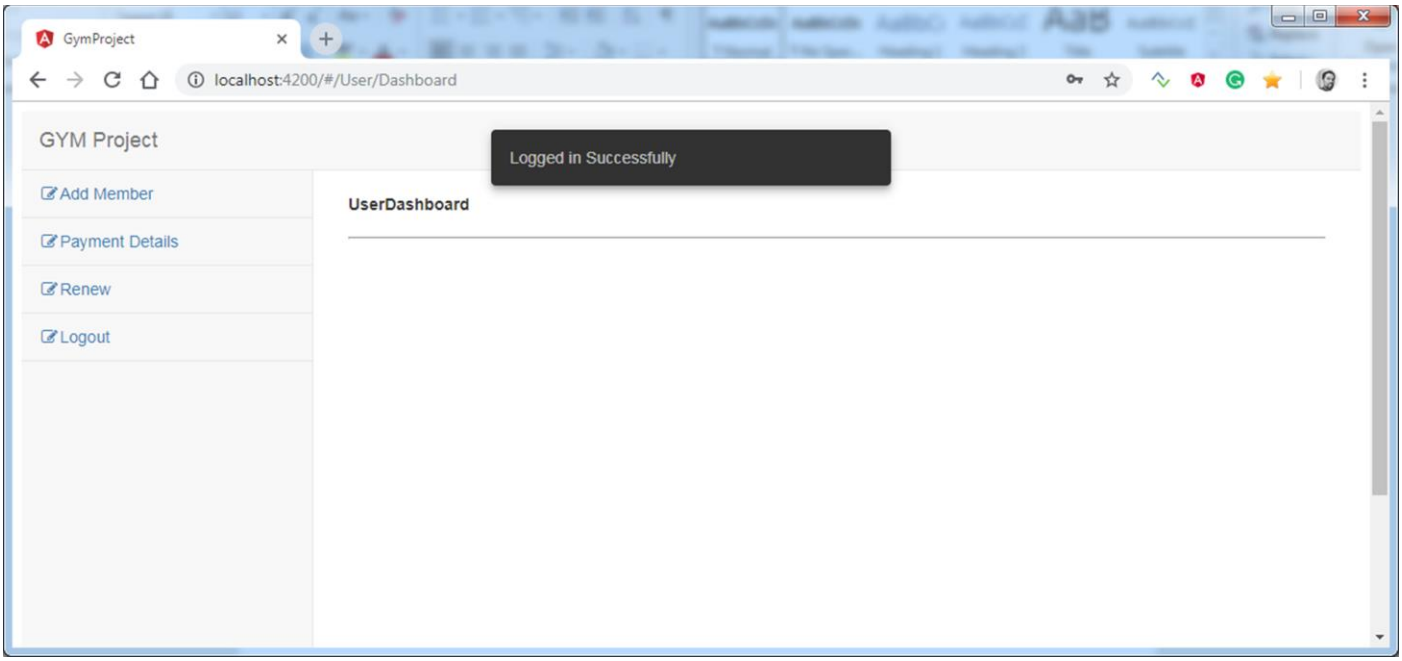
Name	Contactno	EmailID	MemberNo	PlanName	SchemeName	JoiningDate	RenwalDate	PaymentAmount
demo | demo	9999999999	demo@gg.com	GYMONE/4/2019	Quaretly	GYM+CARDIO1	29/12/2018	28/02/2019	2022

User Application Screens

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

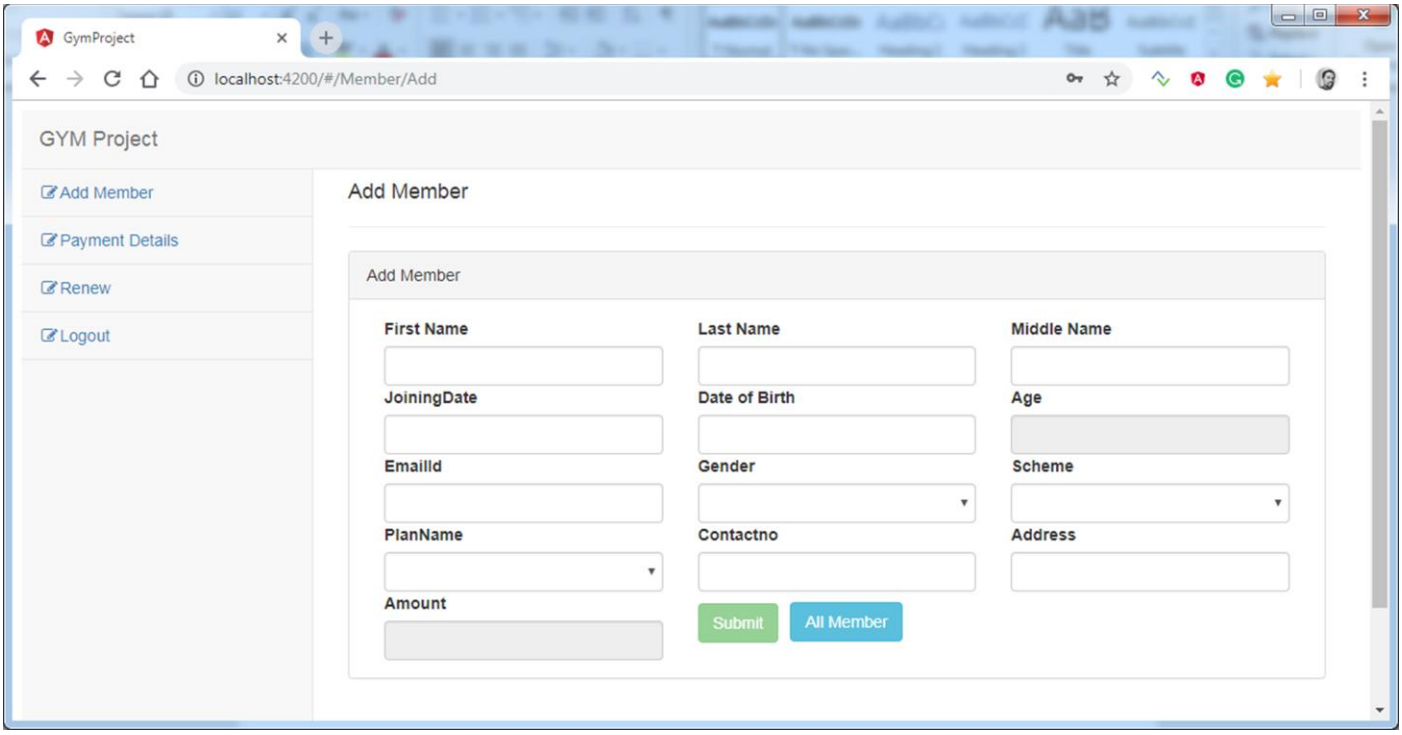


User Dashboard



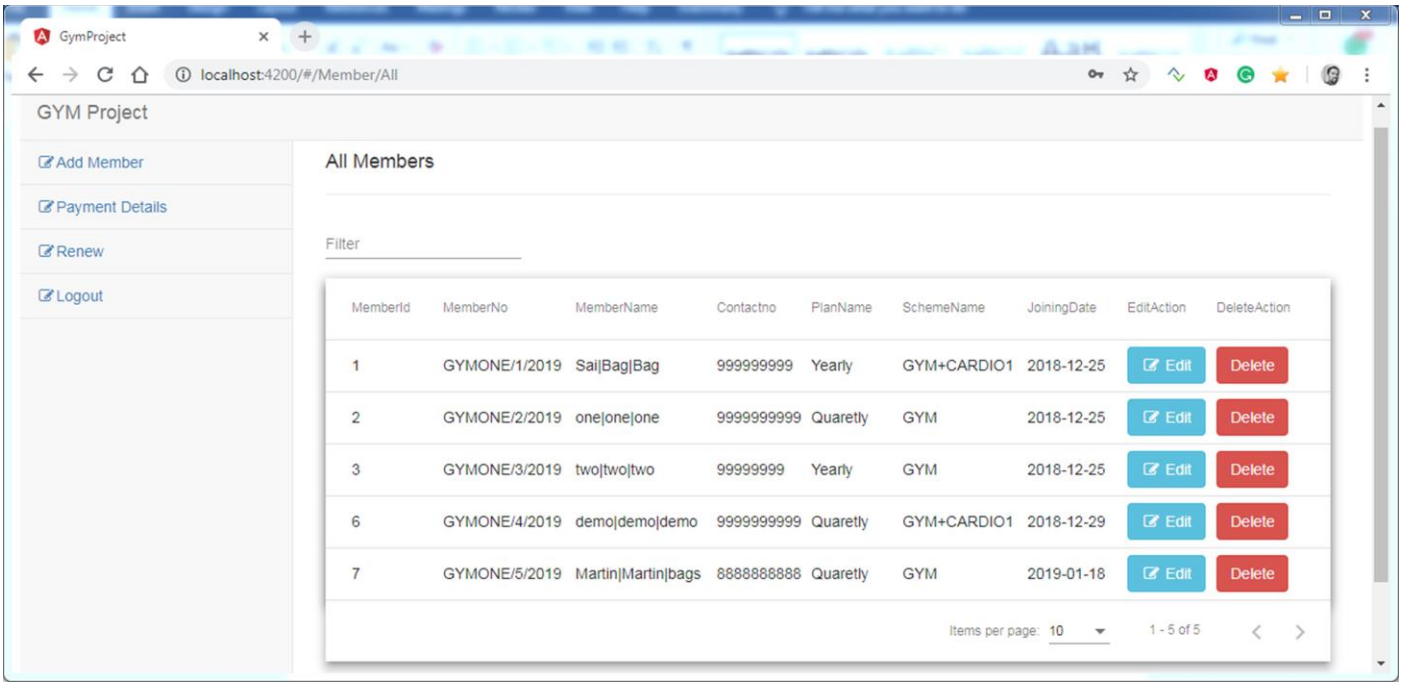
Add Member

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.



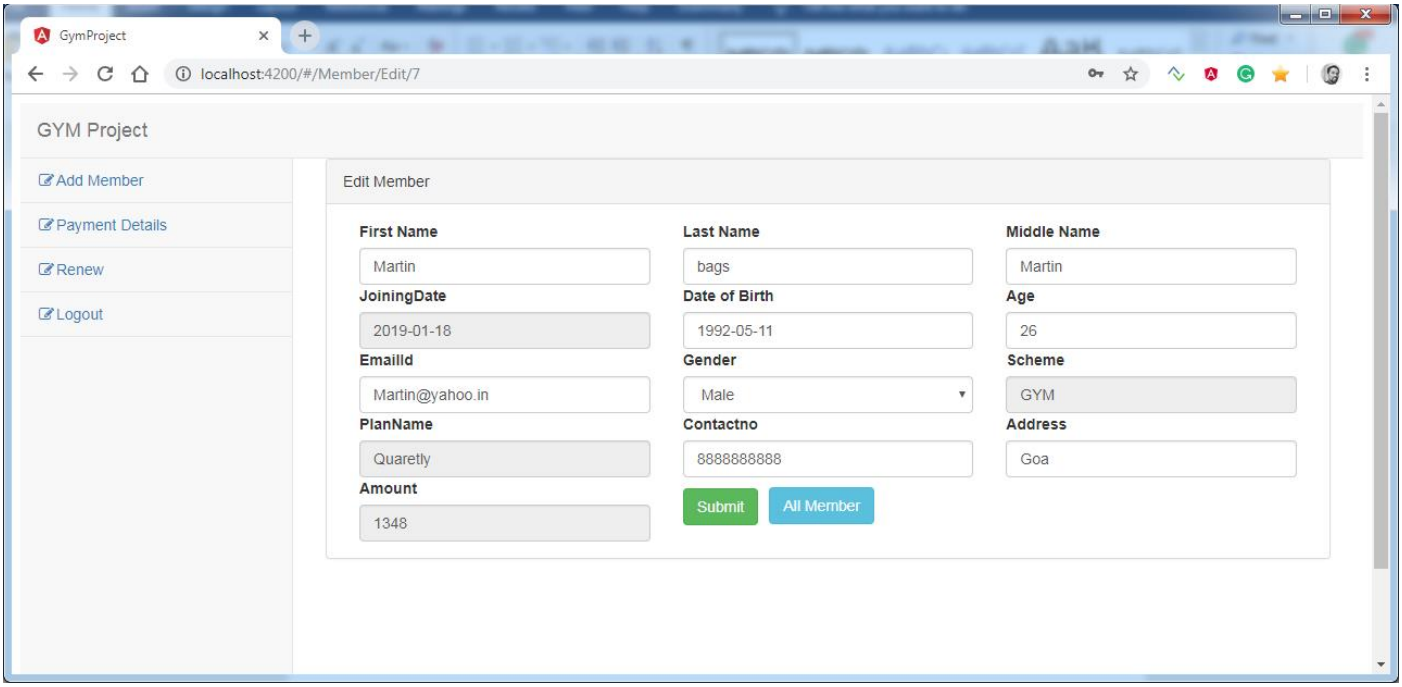
All Member

Display all Members who took Membership.



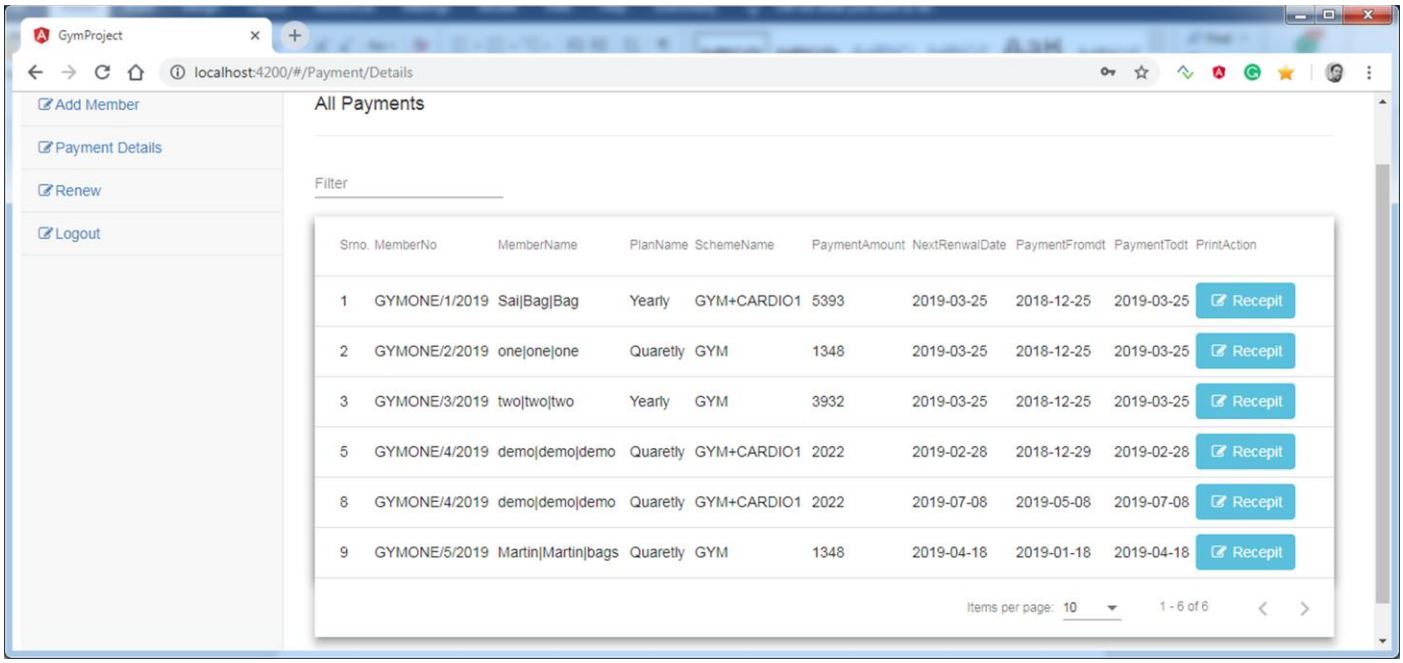
Edit Member

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.



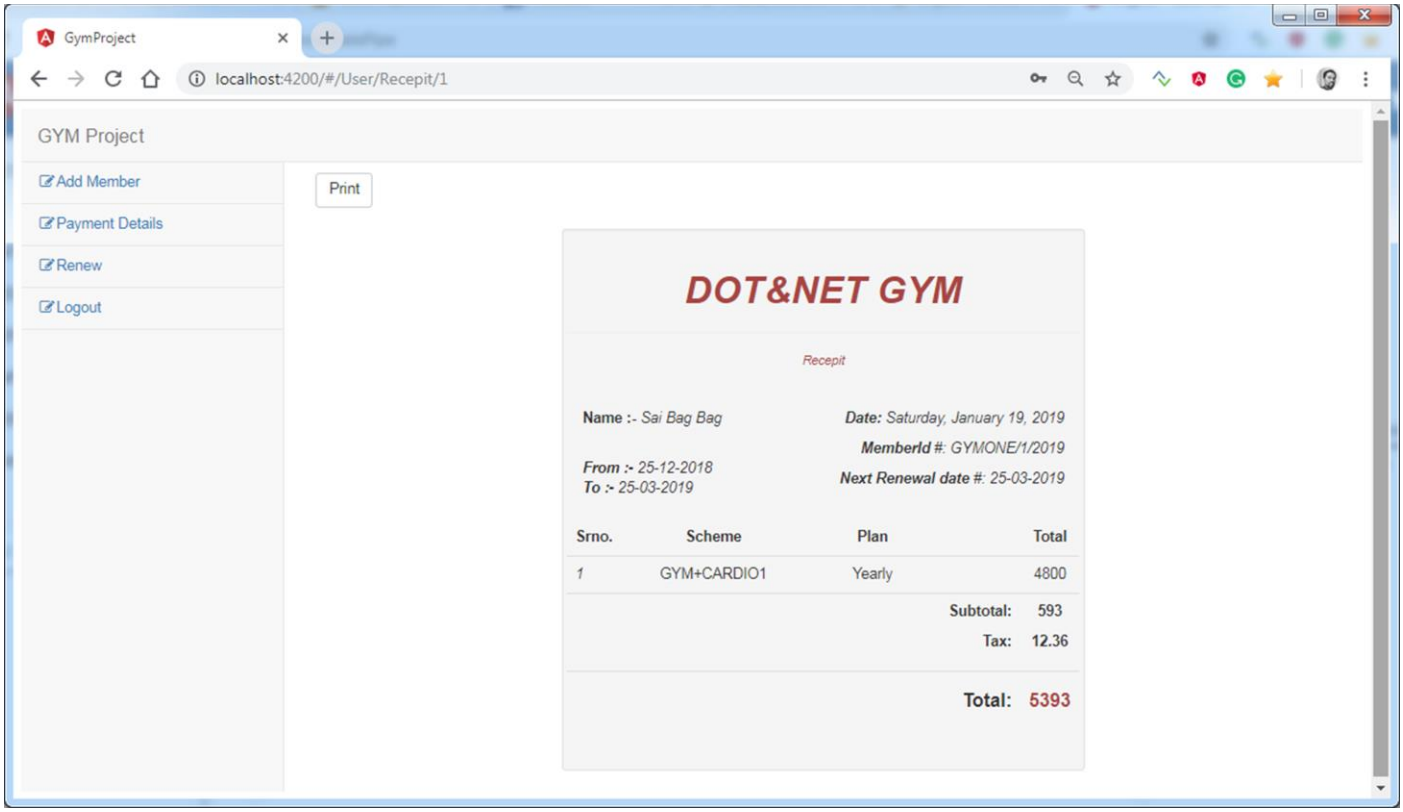
All Payment Details

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

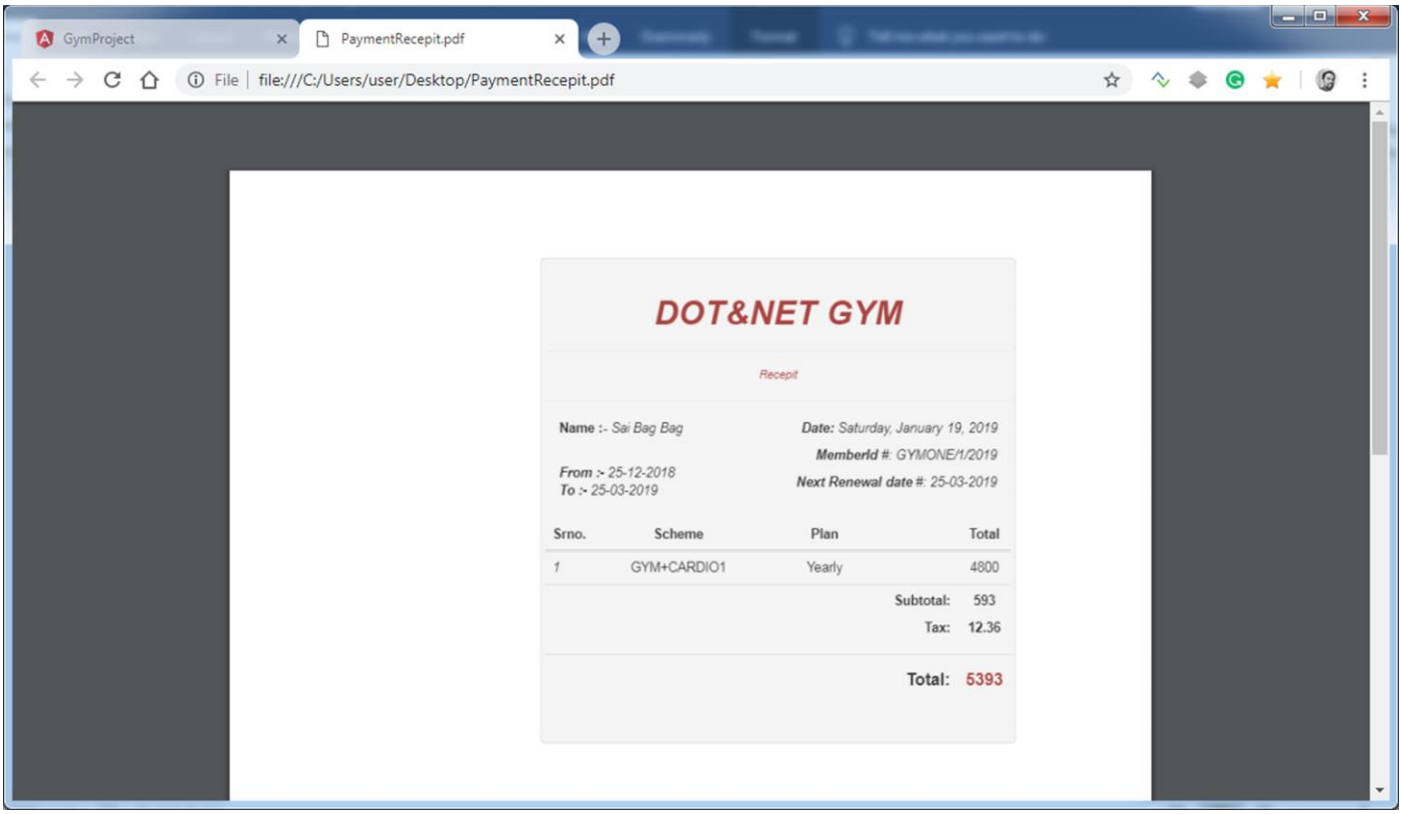


Receipt

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.

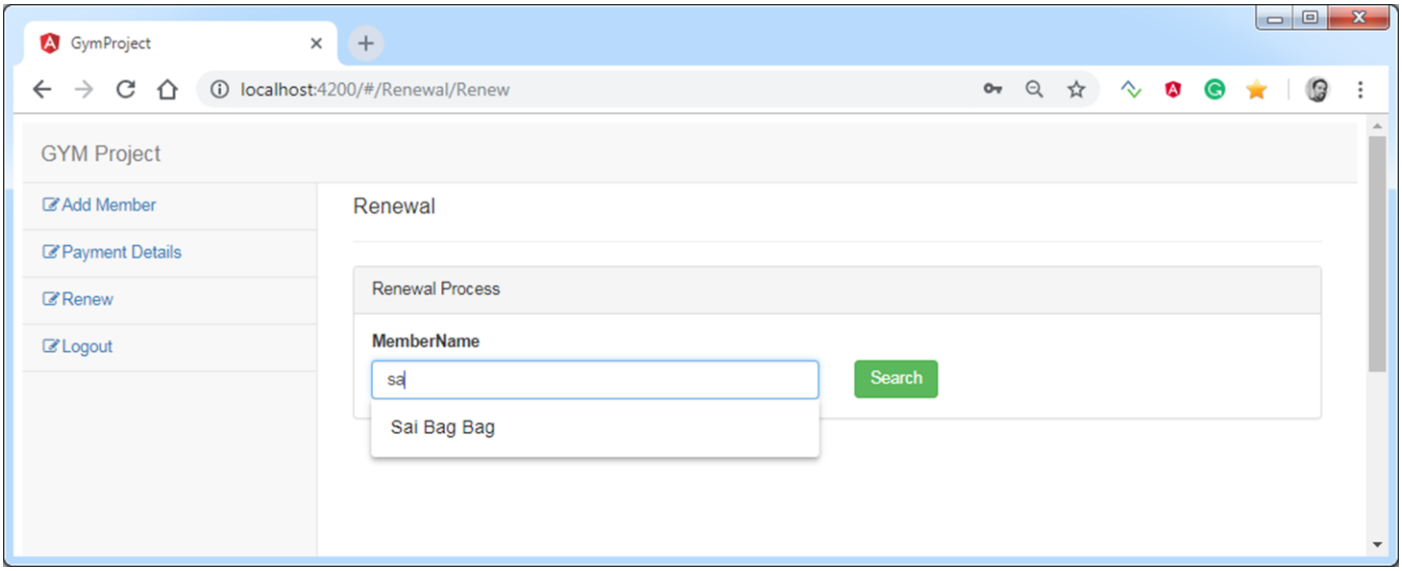


Pdf format of the receipt

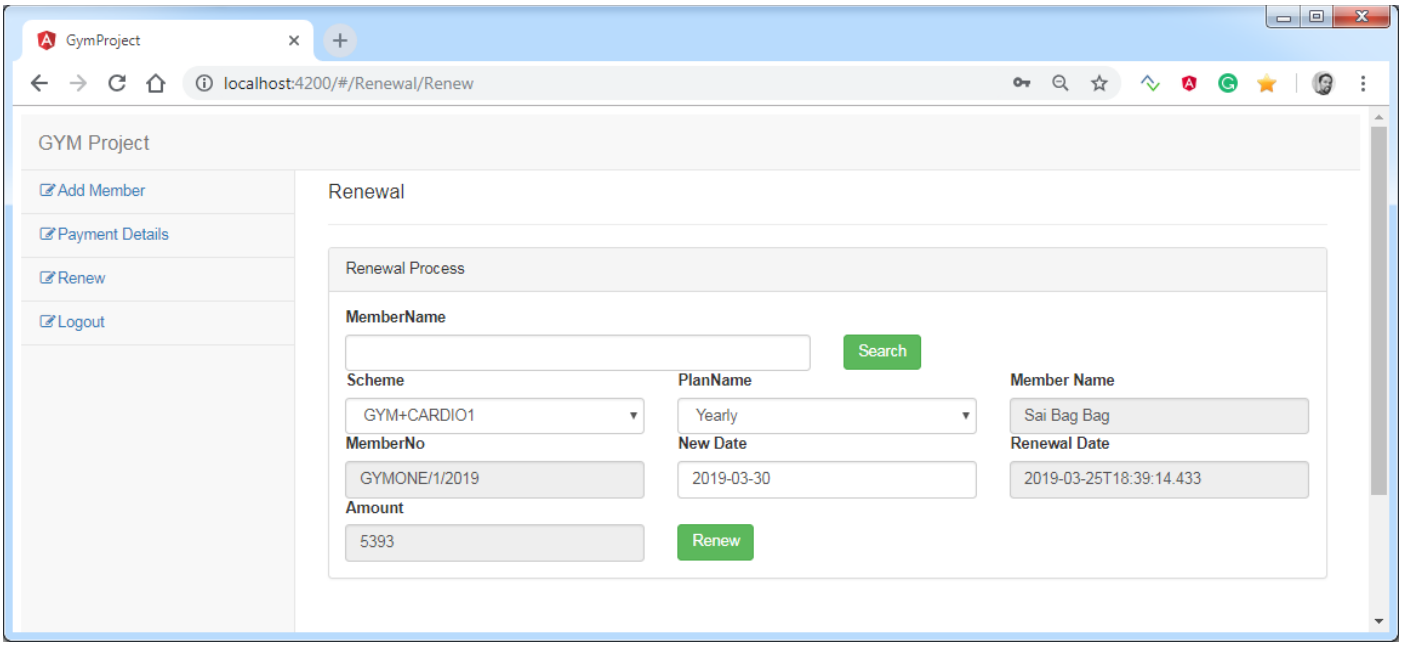


Renewal of Membership

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.



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



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.