2019

Smart Devices Repair Handling System

FINAL TERM PROJECT REPORT

SOUTHEAST MISSOURI STATE UNIVERSITY | CS 575 ADVANCED WEB DEVELOPMENT

Contents Page No.

Smart Device Repair Handling System

Abstract:

The smart device repair handling system is a full-fledged angular 8 application that lets the user register, Login and make new service requests by adding their requirements. At the same time, they can make the payments and an admin can have the privilege to add new dealers to the portal. At the end, the user can provide his feedback for further improvements of the service.

The application uses JWT authentication and is built utilizing C# ASP.NET core Entity Framework, angular and Boot Strap. The application is mobile compatible and can be compressed.

Technologies used:

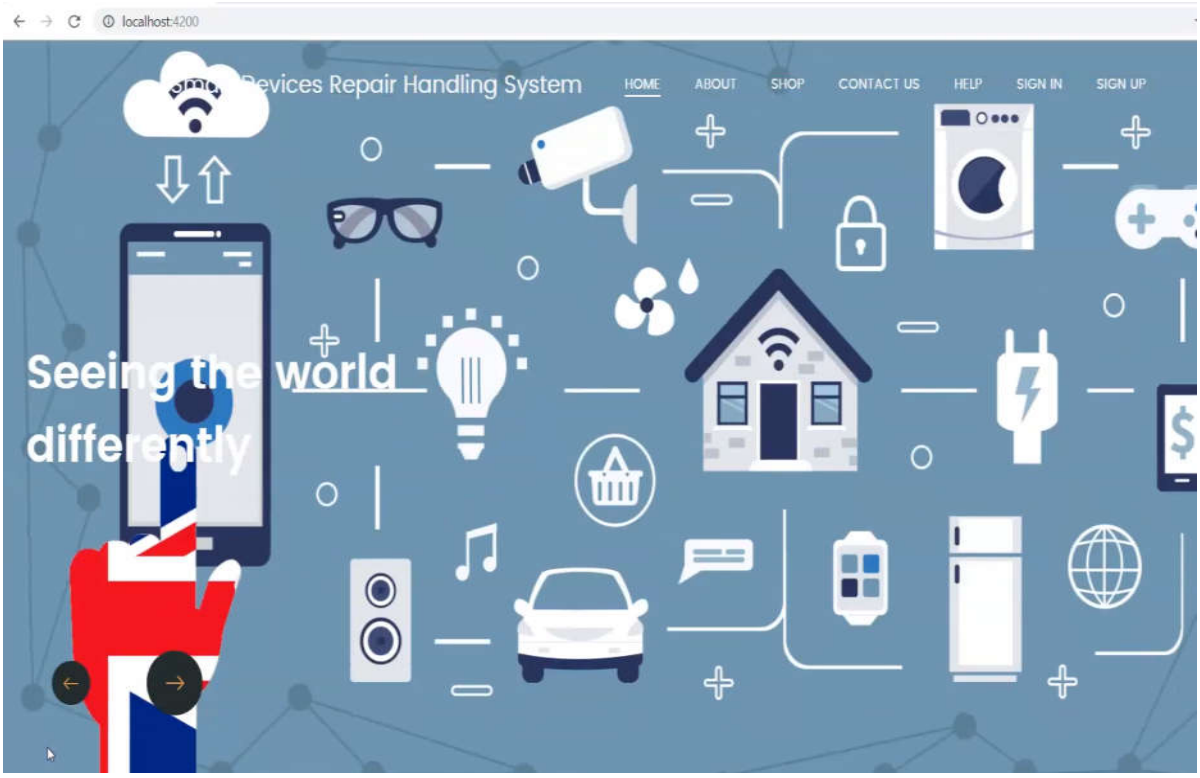
Front end: Angular 7, Bootstrap, JavaScript, HTML 5, CSS 3

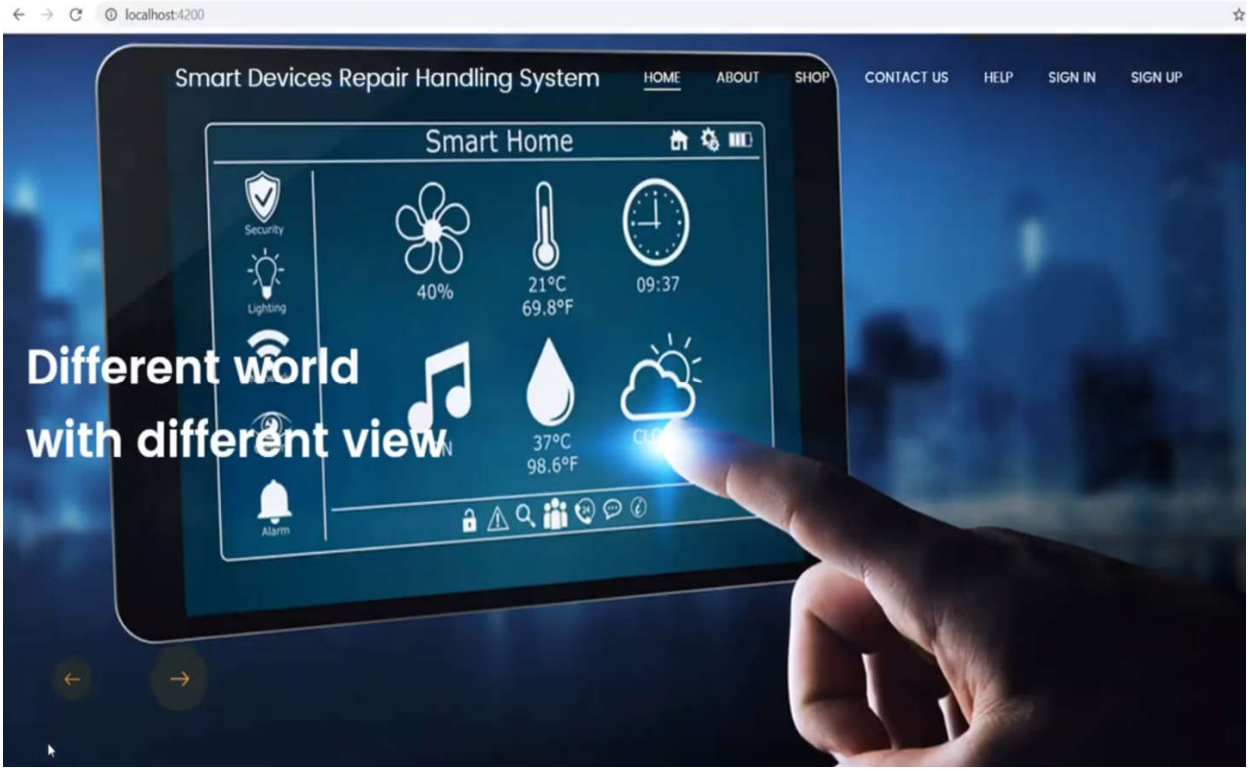
Backend: C# ASP.NET core 2.1.0, Entity Framework 2.1.1

Database: Microsoft SQL server 2008., SQL Server Management Studio

IDE: Visual Studio code, Visual Studio 2019

The application has a carousel and the arrows are much useful in navigating from one page to the other.





Modules:

- **Login using JWT Authentication**
- **Sign Up**
- **Service Request**
- **Feedback**
- **CRUD enabled dealer services**
- **Payment Module in CRUD**
- **Graphical reporting**
- **Database Management**
- **Entity framework logic**

Login using JWT Authentication:

JWT Authentication basically uses the claims, and JSON based Web tokens to authorize the details of the authentic users. If a user once logs in to the system, the details are saved as a session id and once if the user gets back, then their credentials will be validated using the session id, claims and tokens.

Below are the screenshots for JWT authentication.

```
ApplicationDbContext.cs        ResponseExtensions.cs        ApplicationDbContextModelSnapshot.cs
private const string SecretKey = "iNivDmHLpUA223sqsfhqGbMRdRj1PVkH"; // todo: get this
private readonly SymmetricSecurityKey _signingKey = new SymmetricSecurityKey(Encoding.

public Startup(IConfiguration configuration)
{
  Configuration = configuration;
}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the cont
public void ConfigureServices(IServiceCollection services)
{
  // Add framework services.
  services.AddDbContext<ApplicationDbContext>(options =>
      options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"),
          b => b.MigrationsAssembly("SRDHS")));

  services.AddSingleton<IJwtFactory, JwtFactory>();
```

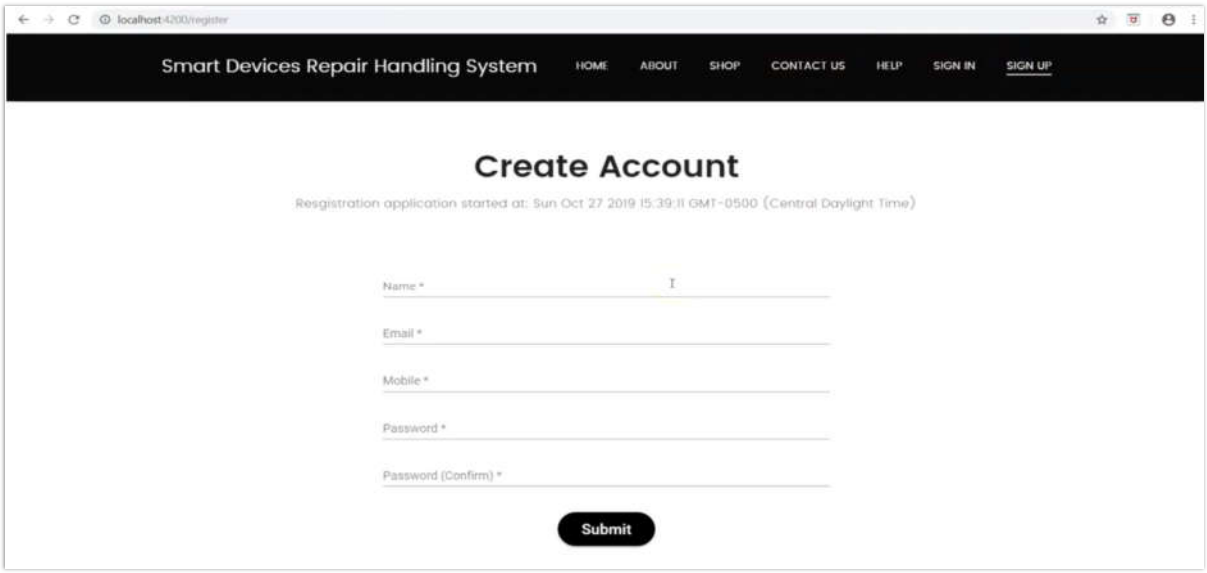
```
File  Edit  View  Tools  Extensions  Window  Help  Search (Ctrl+Q)  AngularASPNETCore2WebApiAuth
Select Startup Item...
Toolbox
Startup.cs*   ApplicationDbContext.cs   ResponseExtensions.cs   ApplicationDbContextModelSnapshot.cs   Customer.cs
1
2  using SRDHSWebApiAuth.Models.Entities;
3  using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
4  using Microsoft.EntityFrameworkCore;
5
6  namespace SRDHSWebApiAuth.Data
7  {
8      public class ApplicationDbContext : IdentityDbContext<AppUser>
9      {
10         public ApplicationDbContext(DbContextOptions options)
11             : base(options)
12         {
13         }
14
15         public DbSet<Customer> Customers { get; set; }
16     }
17 }
18
```

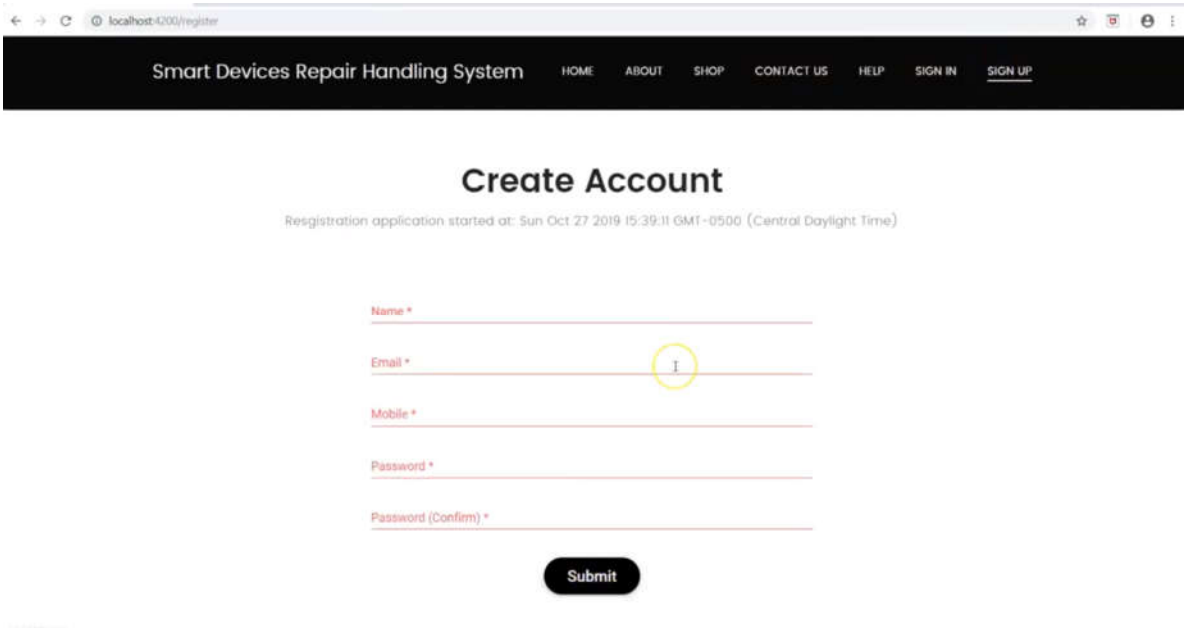
```
options.Audience = jwtAppSettingOptions[nameof(JwtIssuerOptions.Audience)];
options.SigningCredentials = new SigningCredentials(_signingKey, SecurityAlgorithms.HmacSha256);
});

var tokenValidationParameters = new TokenValidationParameters
{
  ValidateIssuer = true,
  ValidIssuer = jwtAppSettingOptions[nameof(JwtIssuerOptions.Issuer)],

  ValidateAudience = true,
  ValidAudience = jwtAppSettingOptions[nameof(JwtIssuerOptions.Audience)],

  ValidateIssuerSigningKey = true,
  IssuerSigningKey = _signingKey,

  RequireExpirationTime = false,
  ValidateLifetime = true,
  ClockSkew = TimeSpan.Zero
};

services.AddAuthentication(options =>
{
  options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
  options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
```

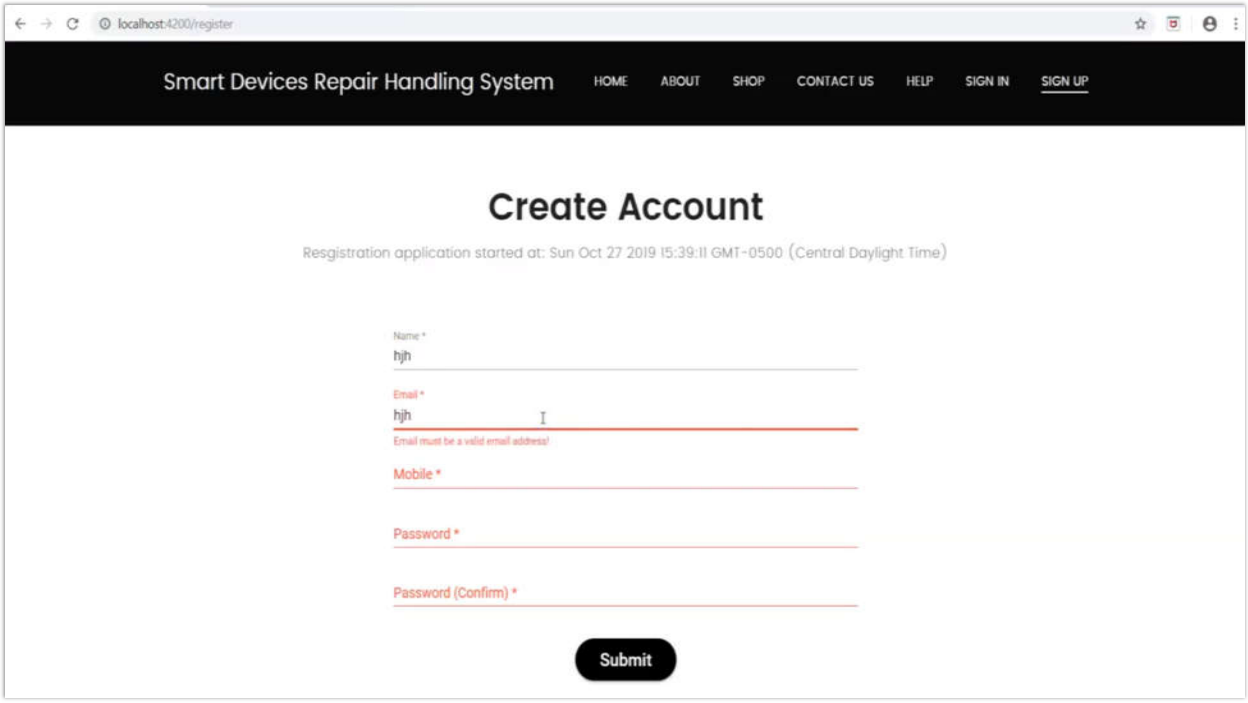
```
{
        new Claim(JwtRegisteredClaimNames.Sub, userName),
        new Claim(JwtRegisteredClaimNames.Jti, await _jwtOptions.JtiGenerator()),
        new Claim(JwtRegisteredClaimNames.Iat, ToUnixEpochDate(_jwtOptions.IssuedAt).ToString(), ClaimValueTypes.Integer64),
        identity.FindFirst(Helpers.Constants.Strings.JwtClaimIdentifiers.Rol),
        identity.FindFirst(Helpers.Constants.Strings.JwtClaimIdentifiers.Id)
    };

    var jwt = new JwtSecurityToken(
        issuer: _jwtOptions.Issuer,
        audience: _jwtOptions.Audience,
        claims: claims,
        notBefore: _jwtOptions.NotBefore,
        expires: _jwtOptions.Expiration,
        signingCredentials: _jwtOptions.SigningCredentials);

    var encodedJwt = new JwtSecurityTokenHandler().WriteToken(jwt);

    return encodedJwt;
}

public ClaimsIdentity GenerateClaimsIdentity(string userName, string id)
{
    return new ClaimsIdentity(new GenericIdentity(userName, "Token"), new[]
    {
        new Claim(Helpers.Constants.Strings.JwtClaimIdentifiers.Id, id),
```

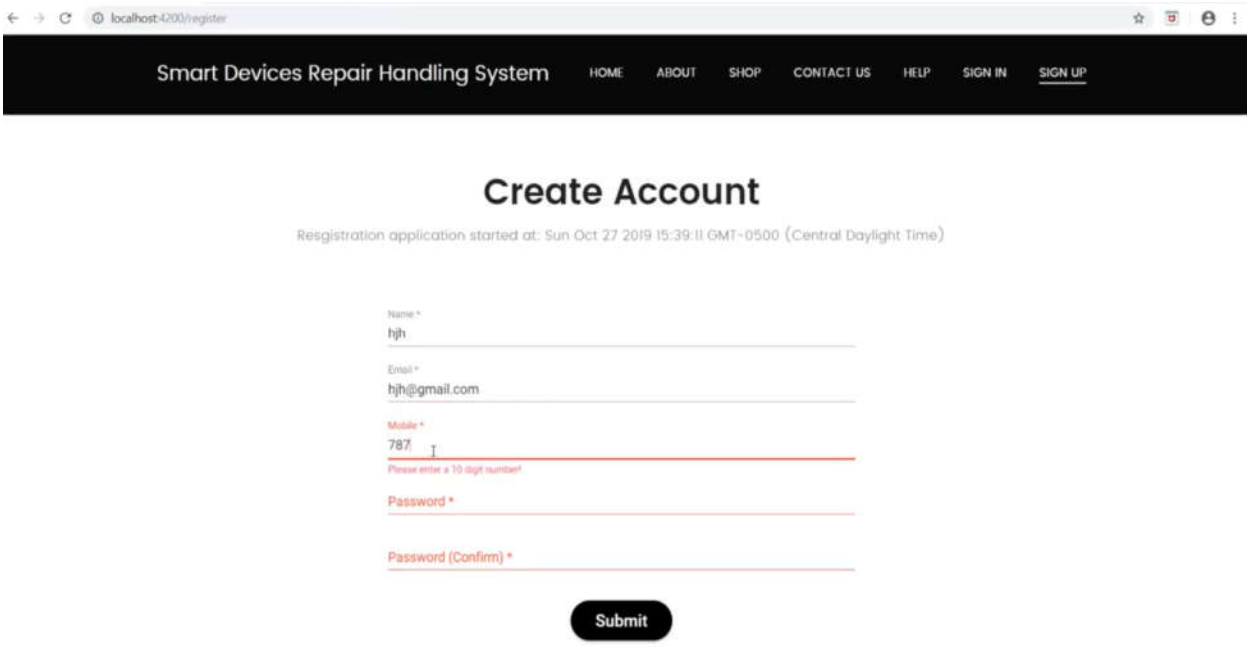
Sign Up:

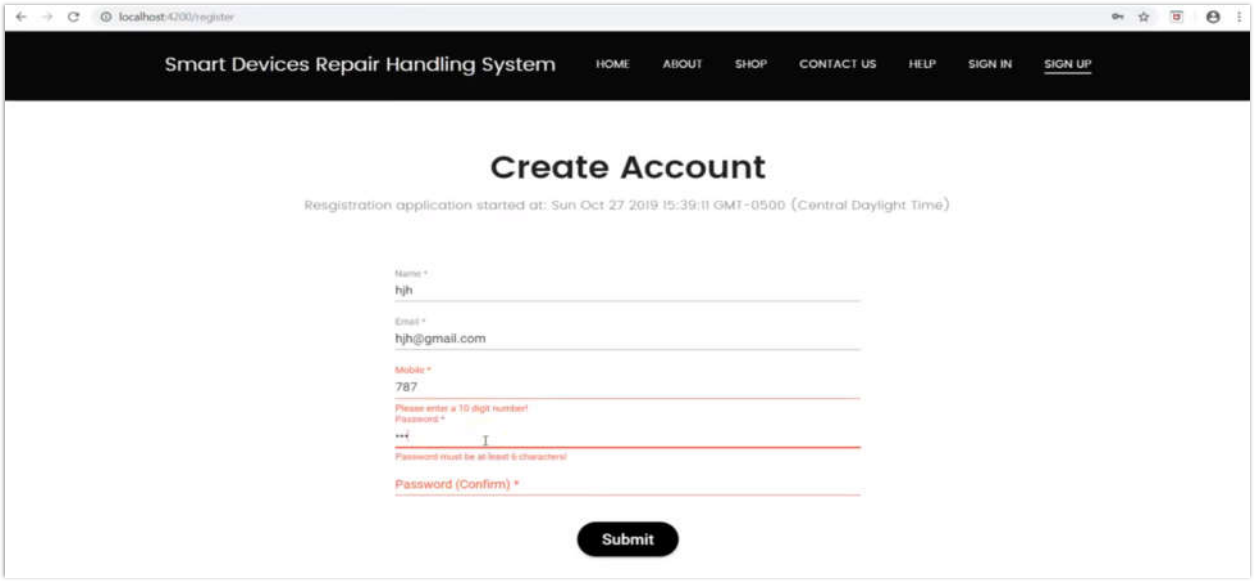
The sign-up page is for the letting the new user signup. The details have validations as in the name should be characters, Email should be of the format name@domain.com, the password type and retype needs to be matched and contact number should be of length 10. Below are the screenshots of the same.

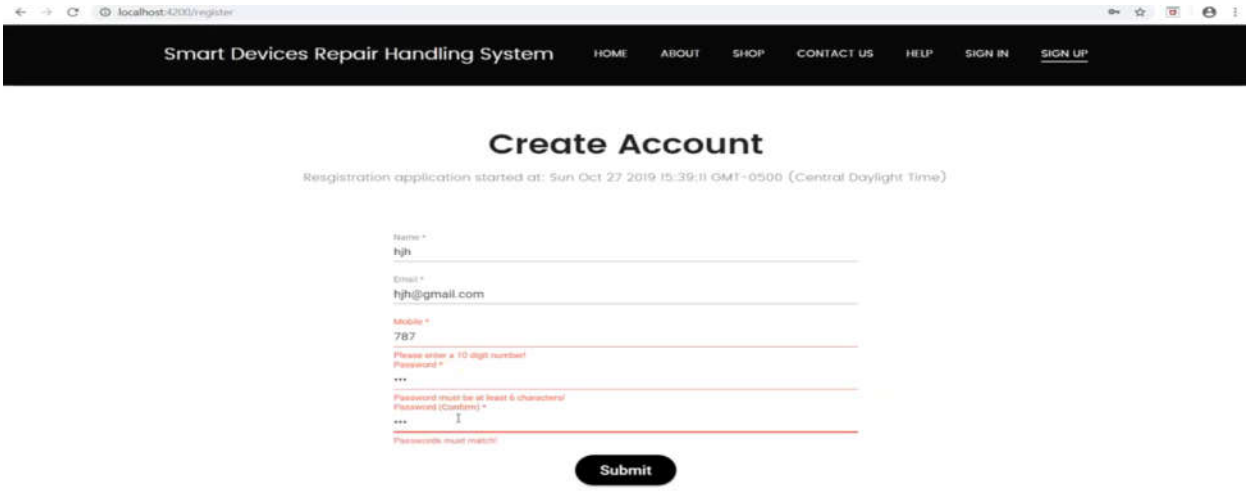





localhost:4200/register
Smart Devices Repair Handling System
HOME
ABOUT
SHOP
CONTACT US
HELP
SIGN IN
SIGN UP
Create Account
Resgistration application started at: Sun Oct 27 2019 15:39:11 GMT-0500 (Central Daylight Time)
Name *
hjh
Email *
hjh
Email must be a valid email address!
Mobile *
Password *
Password (Confirm) *
Submit

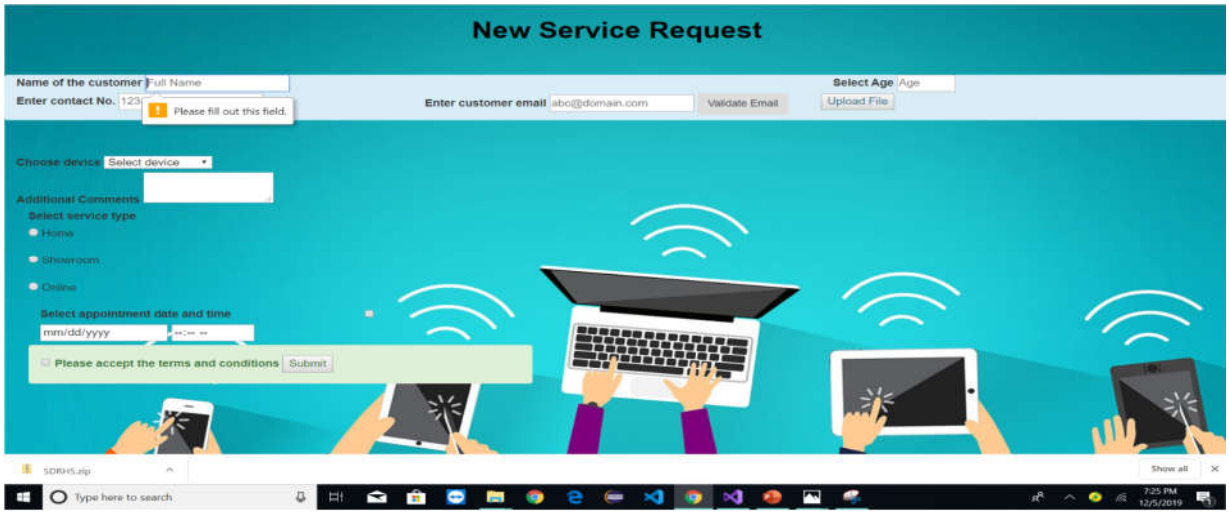

localhost:4200/register
Smart Devices Repair Handling System
HOME
ABOUT
SHOP
CONTACT US
HELP
SIGN IN
SIGN UP
Create Account
Resgistration application started at: Sun Oct 27 2019 15:39:11 GMT-0500 (Central Daylight Time)
Name *
hjh
Email *
hjh@gmail.com
Mobile *
787
Please enter a 10 digit number!
Password *
Password (Confirm) *
Submit





Service Request:

The user can make new service requests from this page. The JavaScript validations are enabled, and the user needs to have all the columns filled and should accept the terms and conditions to be able to make a successful request.

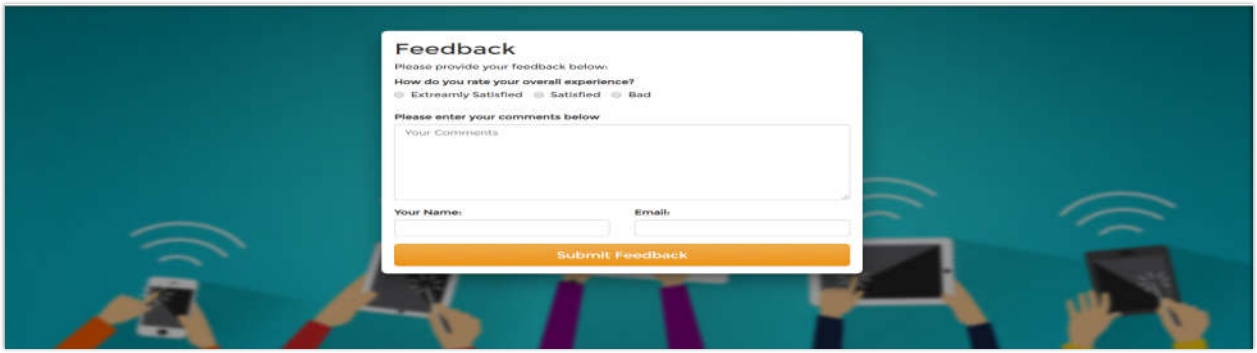

New Service Request
Name of the customer
Full Name
Enter contact No.
Please fill out this field.
Enter customer email
abc@domain.com
Validate Email
Select Age
Age
Upload File
Choose device
Select device
Additional Comments
Select service type
Home
Showroom
Online
Select appointment date and time
mm/dd/yyyy
Please accept the terms and conditions
Submit
Type here to search


This page says
Invalid Email.
OK
Name of the customer
agsjdj
Enter contact No.
121454545
Enter customer email
abc@domain.com
Validate Email
Select Age
Age
Upload File
Choose device
Select device
Additional Comments
Select service type
Home
Showroom
Online
Select appointment date and time
mm/dd/yyyy
Please accept the terms and conditions
Submit


This page says
Request Recorded! Tracking will be sent your email! Our executive will call you for the Confirmation
OK
Name of the customer
agsjdj
Enter contact No.
121454545
Enter customer email
hj@gmail.com
Validate Email
Select Age
10
Upload File
Choose device
Laptop
Additional Comments
Select service type
Home
Showroom
Online
Enter Team Viewer Id
46464
Select appointment date and time
12/27/2019
12:59 AM
Please accept the terms and conditions
Submit

Feedback:

The feedback is a basic static feedback page with radio buttons and submit button and a text box to record the user feelings on the service that he received from the users.



CRUD enabled dealer services:

The Application page built in “Smart Devices Repairing System” is added with new devices and Add-on Detail Categories.

· These connections are made to the backend using C#, ASP.NET and Entity Framework.

· Authentication is done using the database credentials “SA” and password 1234 to the complete database creation using C# thus making a connection to SQL server for storage and retrieval of data.

· The demonstration of the application is that we implemented the CRUD operations such that we can insert the data or new occurrences of data in the front end so that the data is stored at back end. We can modify the existing records by choosing Modify option or We can read and update the data values of the existing records.

· One Major Functionality is “New Release Registration” Where admin can add the releases of the dealers with categories as Type, Launch, Value, Data, Category and Description.

· Further the functionalities of category are again categorized into five types like IoT devices, Living, Home Appliances etc. and the same CURD functionalities are applicable on it.

· Different modifications like changing name, category type can be done, and these changes could be seen from the front end.

· As the “Edit” functionality helps in modifying the original saved data to new data, The “Delete” option is used to completely erase the records which are saved and this Delete functionality is done with the pop-up where we get to decide whether or not to delete the record.

· Delete option is available for the category in record also.

Module screenshots:

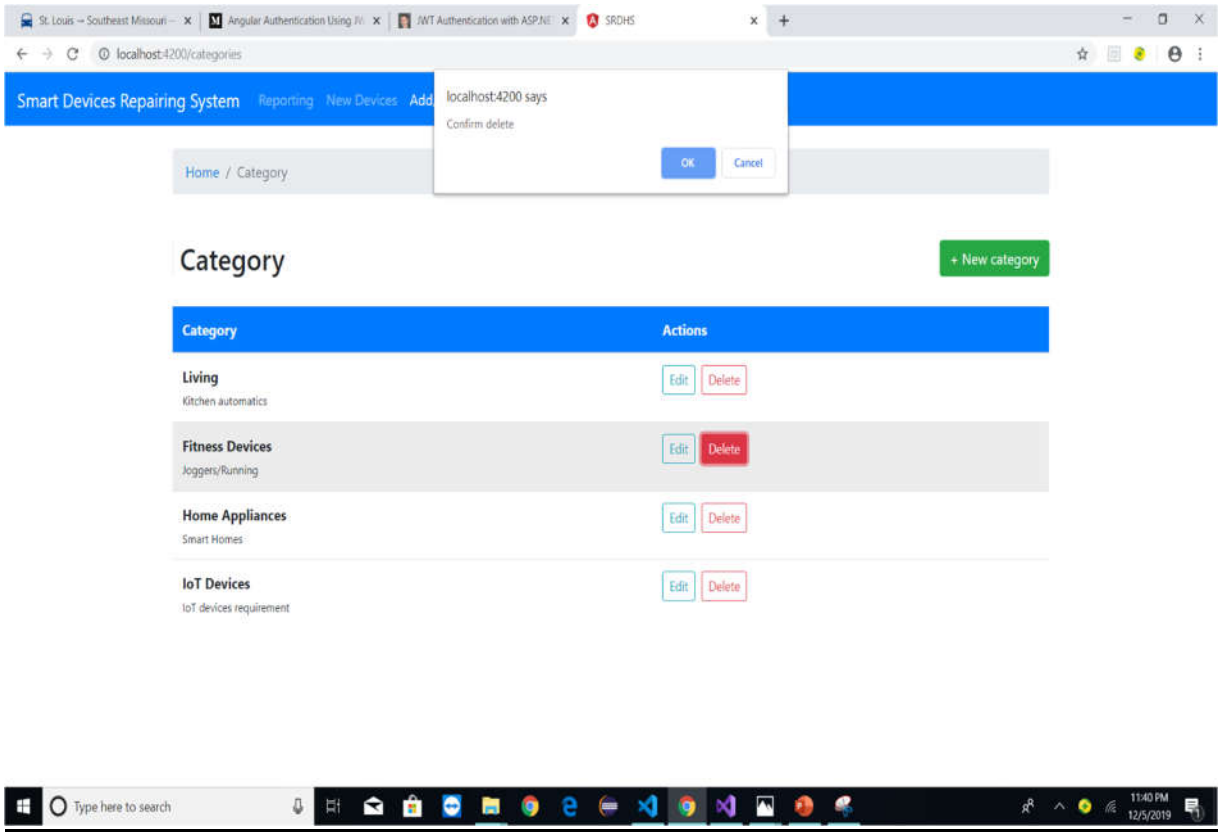
Quick overview:

New releases

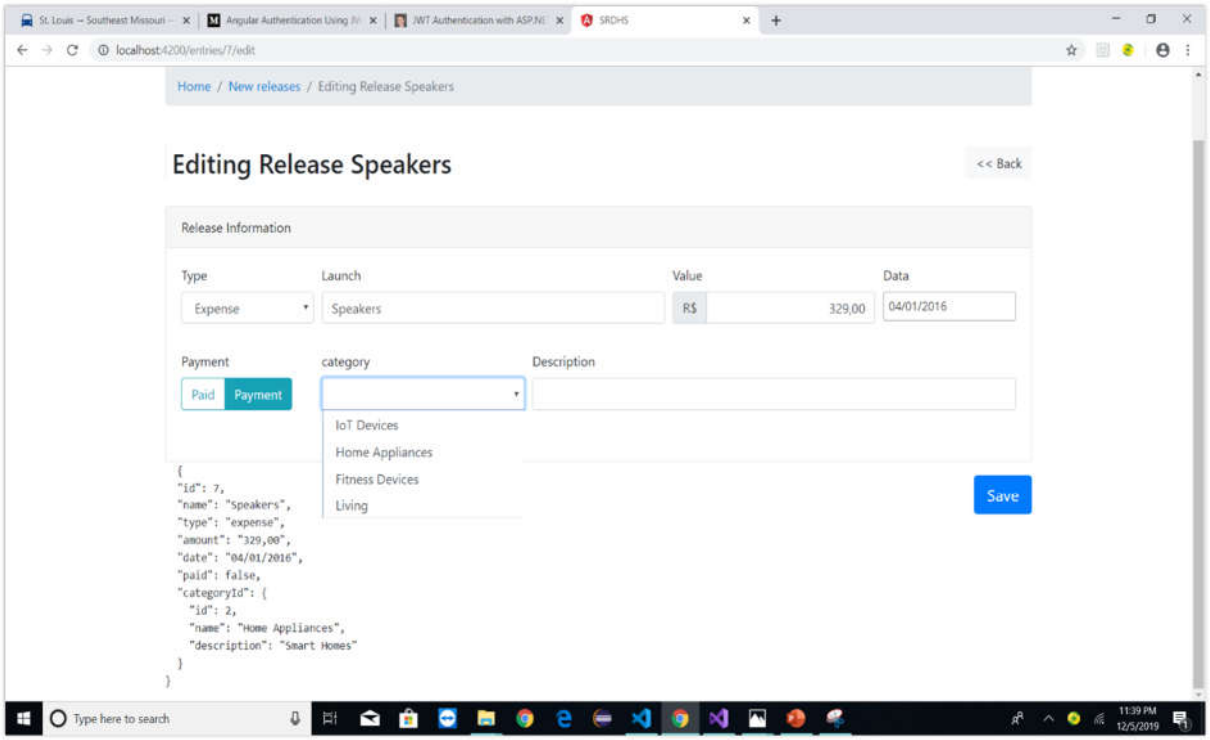
+ New Release

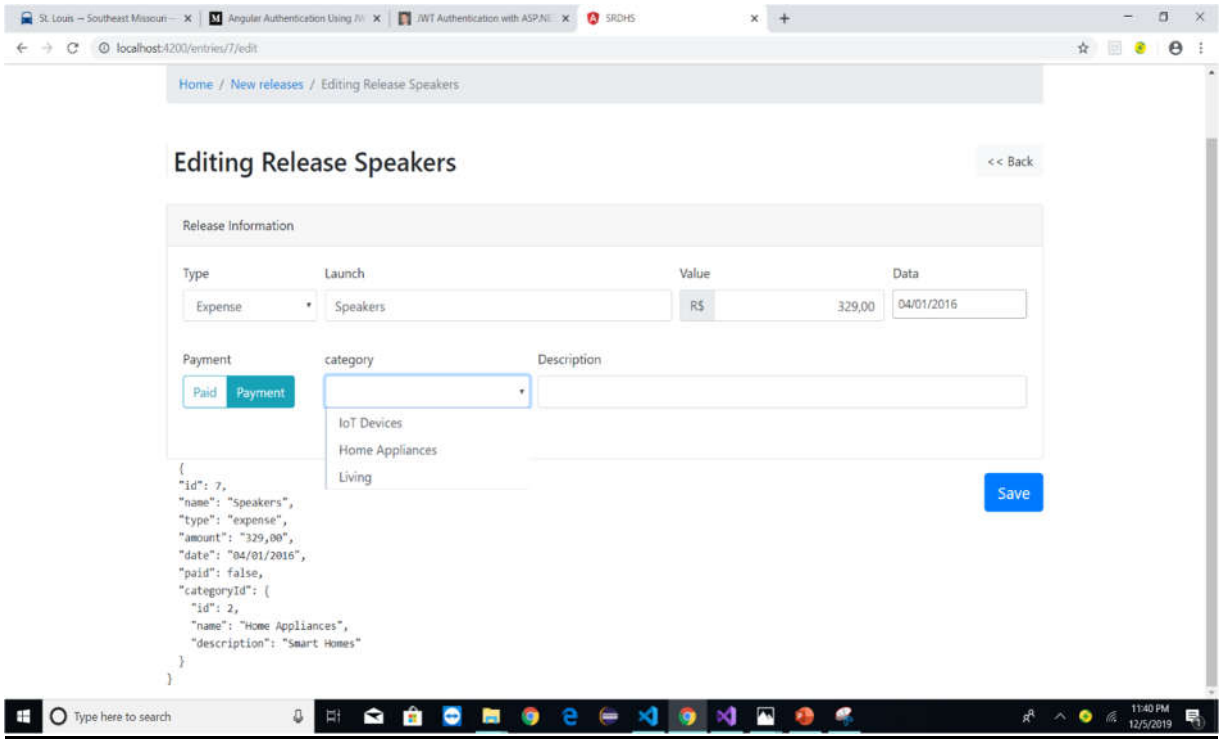
Launch	category	Price($)	Actions
Speakers 04/01/2016	Home Appliances	329,00 Payment	Edit Delete
Joggers 05/31/2017	Fitness Devices	115,00 Payment	Edit Delete
SmartHome Retro fit 12/03/2020	IoT Devices	210,00 Payment	Edit Delete
Smart Bluetooth Trackers 11/20/2019	IoT Devices	160,00 Payment	Edit Delete
Modern kitchen 10/28/2019	Living	3800,53 Payment	Edit Delete
Apple and Samsung watches 09/16/2019	IoT Devices	450,00 Payment	Edit Delete
Fitbit Smart watches 05/31/2019	IoT Devices	1200,00 Payment	Edit Delete

Delete confirmation:



Categories and Edit on categories to reflect on data from the database





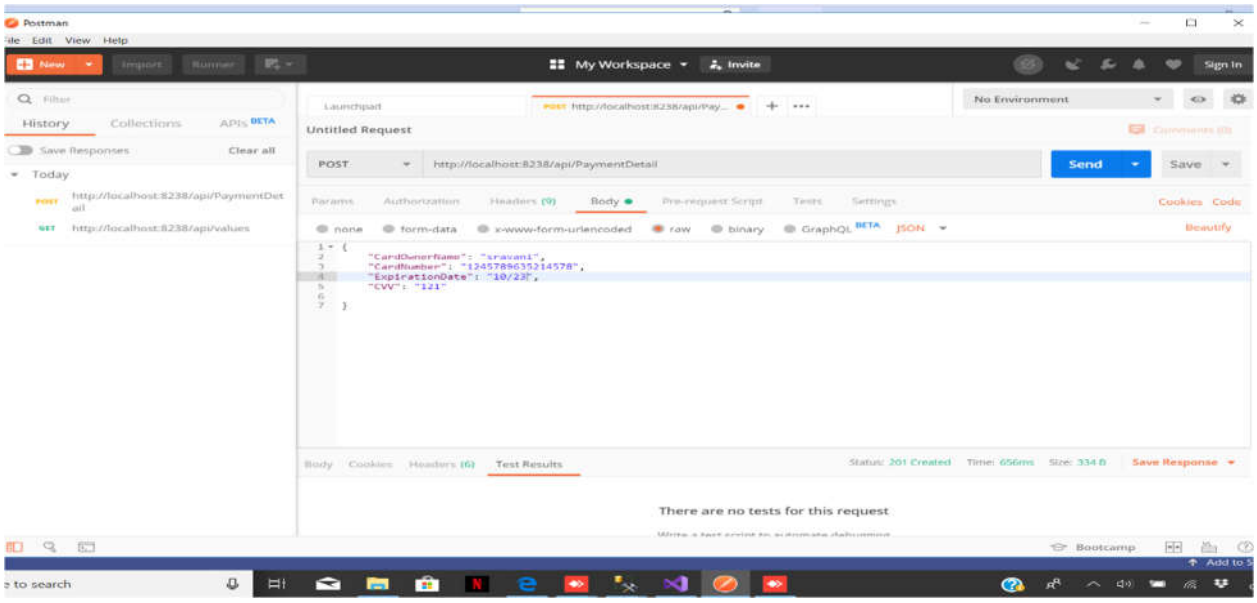
Payment Module in CRUD:

The payment module also uses the Entity framework to perform the CRUD operations, we can add the card details to the database, delete the complete details from the database, update the details from the database and read the details.

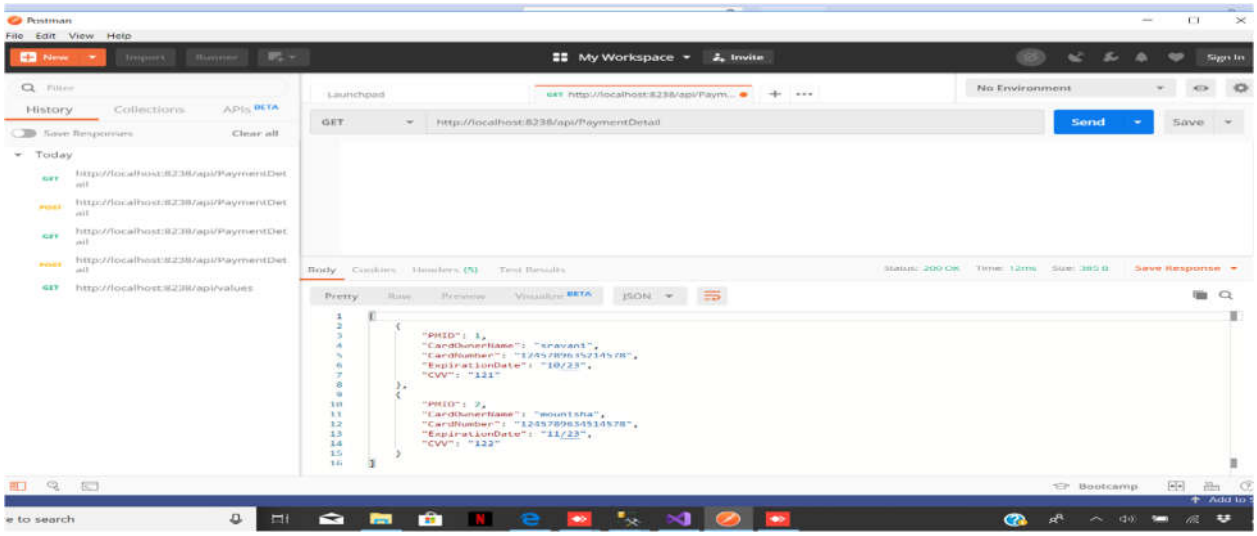
The requests i.e. data can be entered and viewed by performing GET and POST from postman. The screenshots of the same are as follows. For any modifications made on the data from front end, the data gets changed at the backend too.

We have enabled alert boxes for successful operations performed.

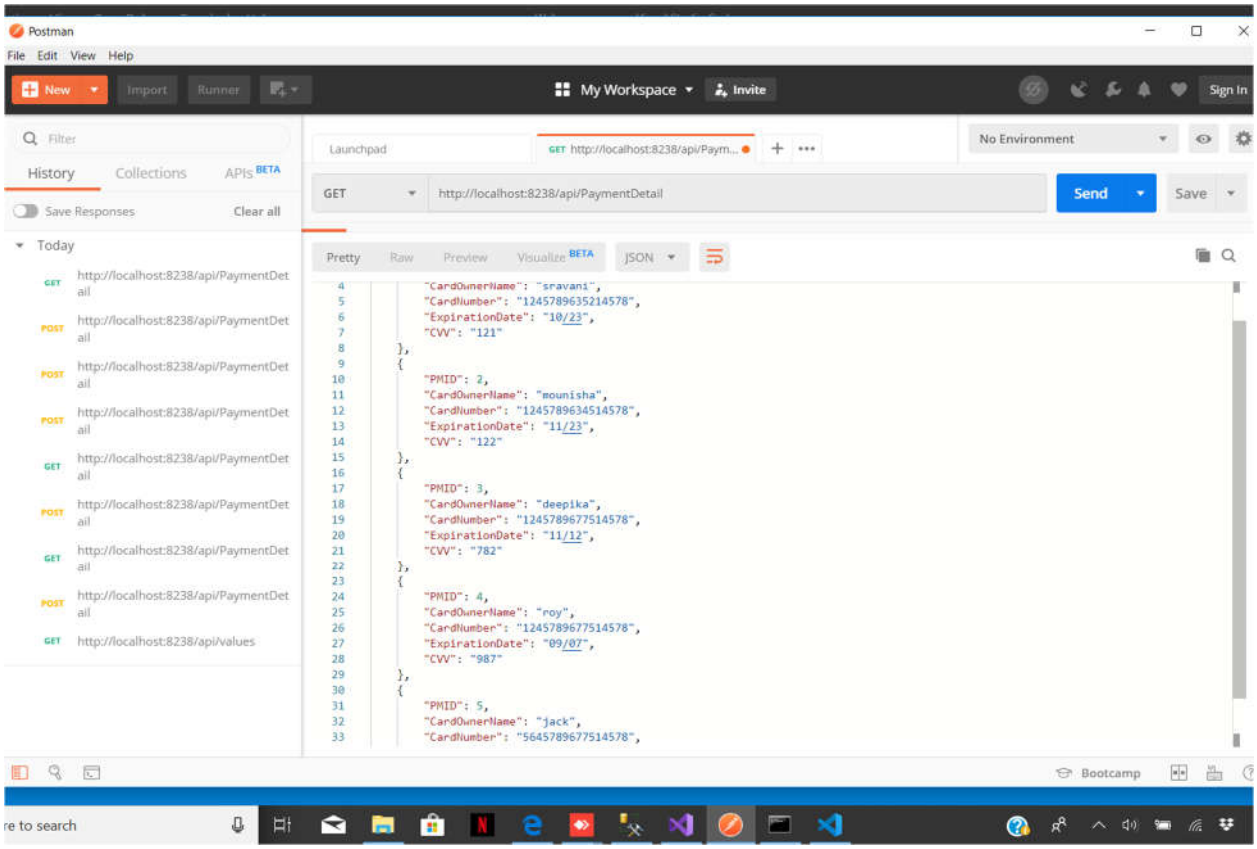
Postman POST method:



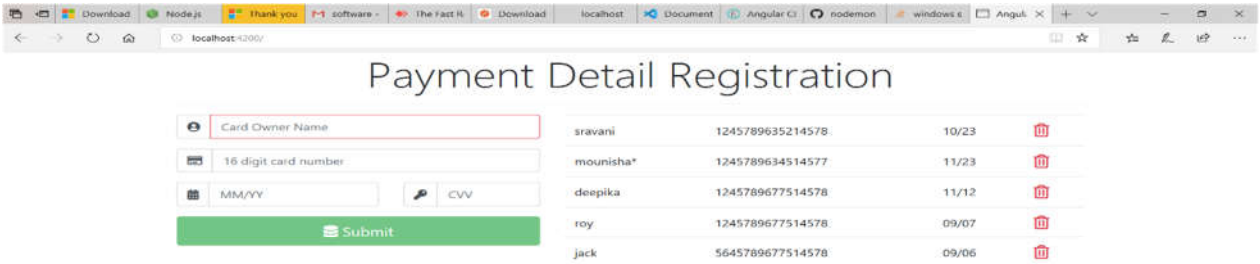
Postman GET Method:



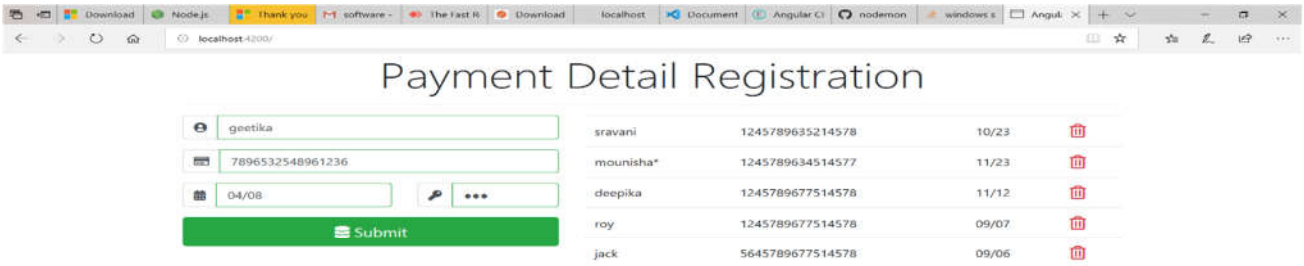
Postman log for GET and POST Methods:



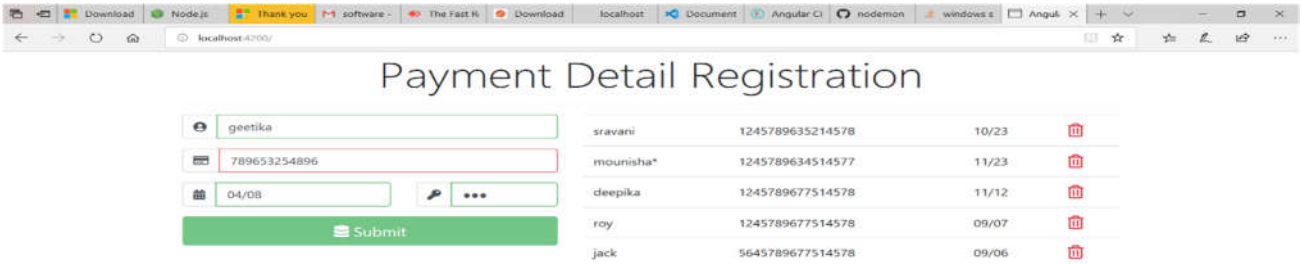
Payment CRUD launch page:



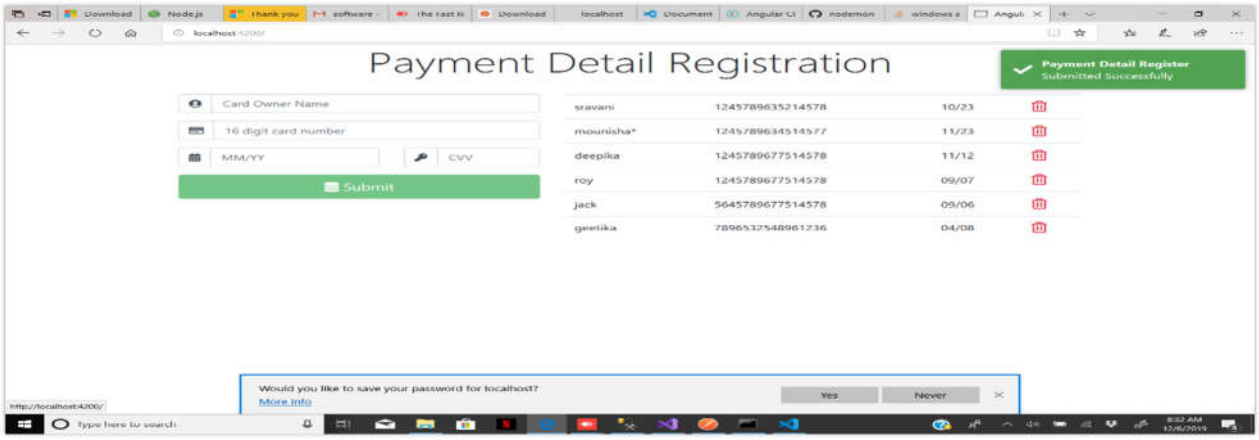
Payment CRUD Create functionality:



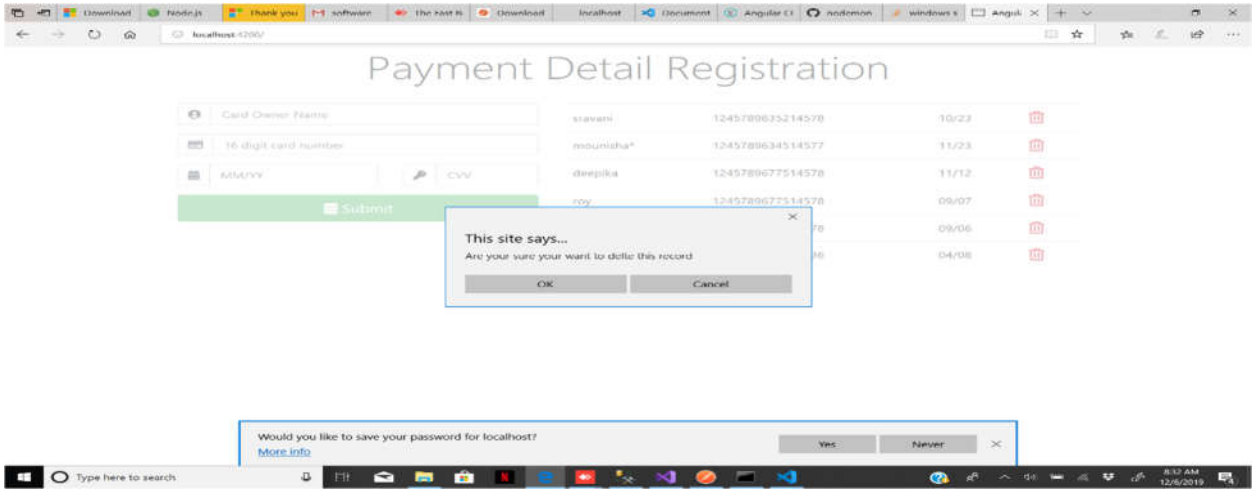
Payment CRUD validation:



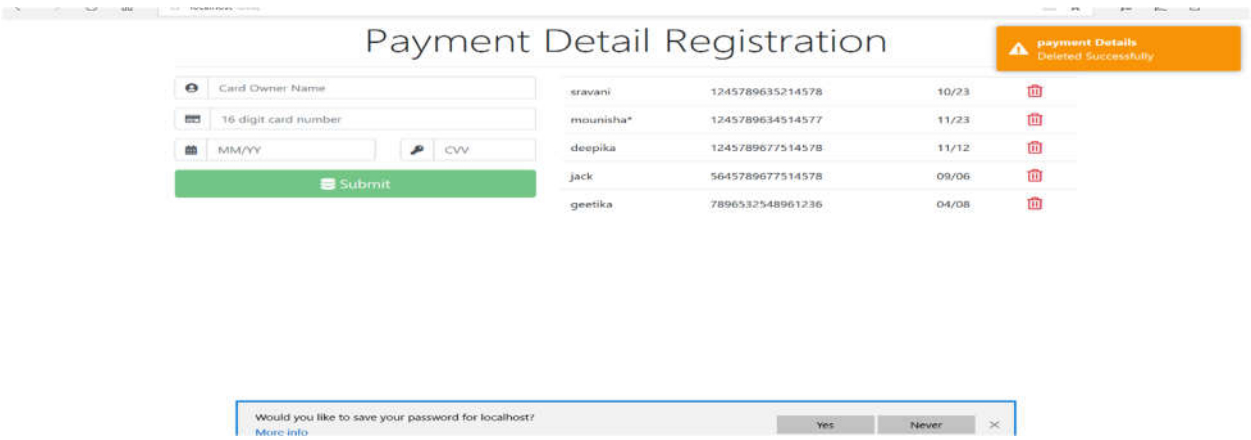
Payment updated successfully:



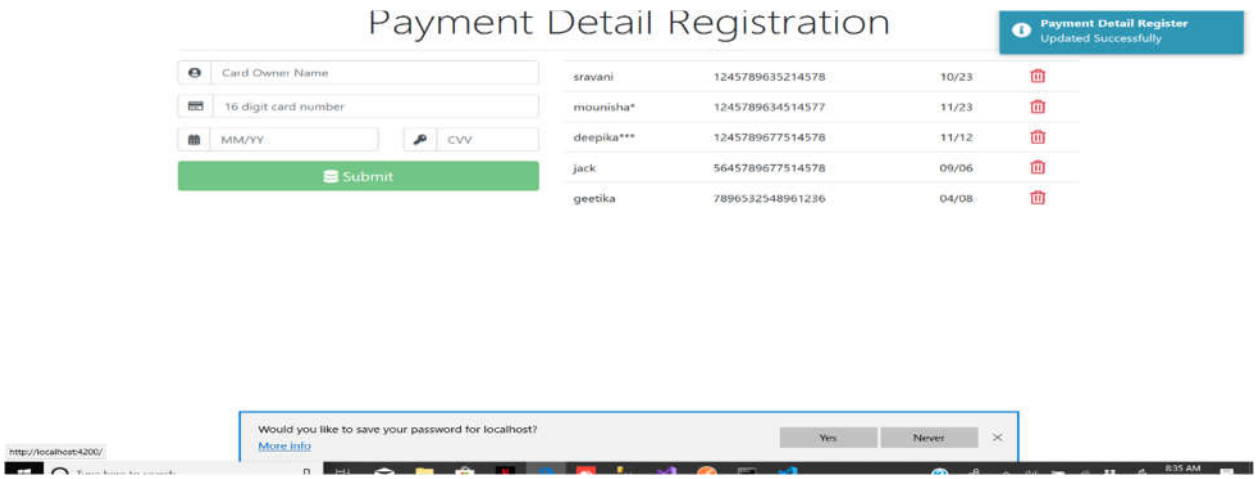
Payment CRUD deletion confirmation:



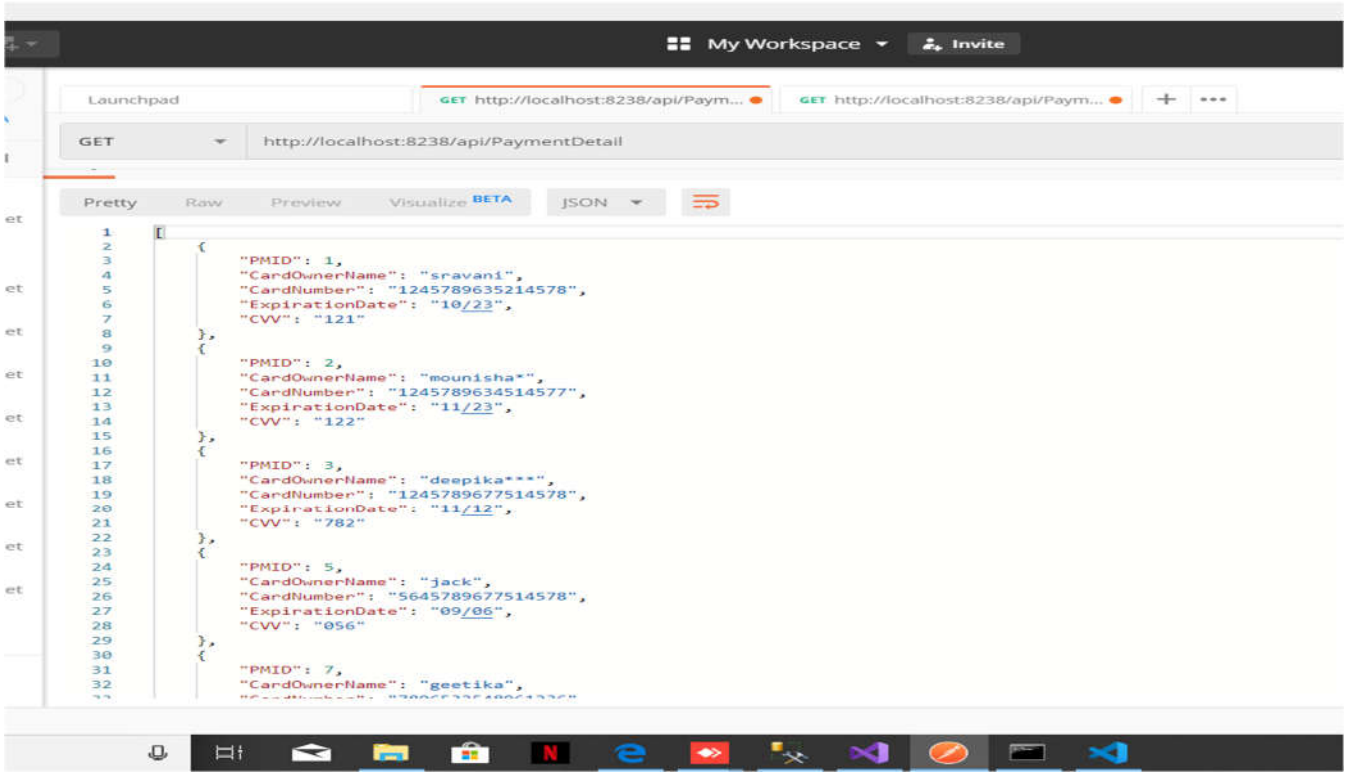
Payment CRUD deletion completion:



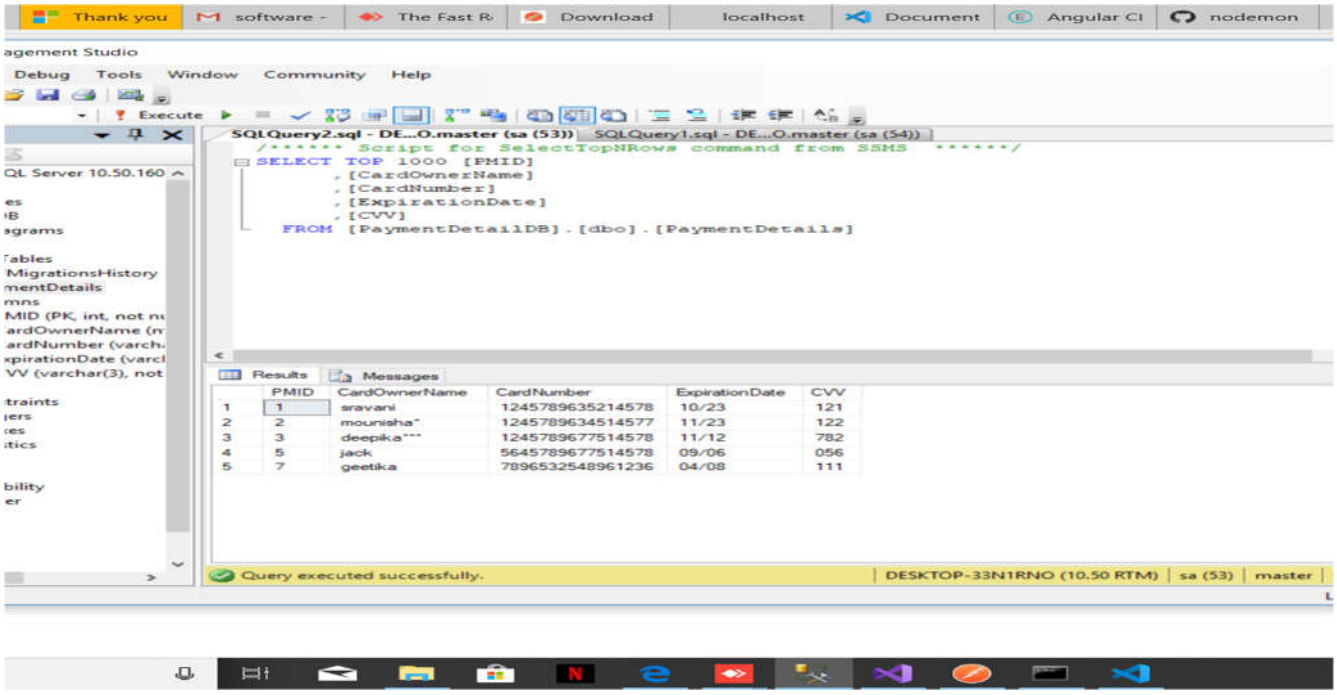
Payment CRUD update functionality:



Postman view after performing CRUD on the payment's component:

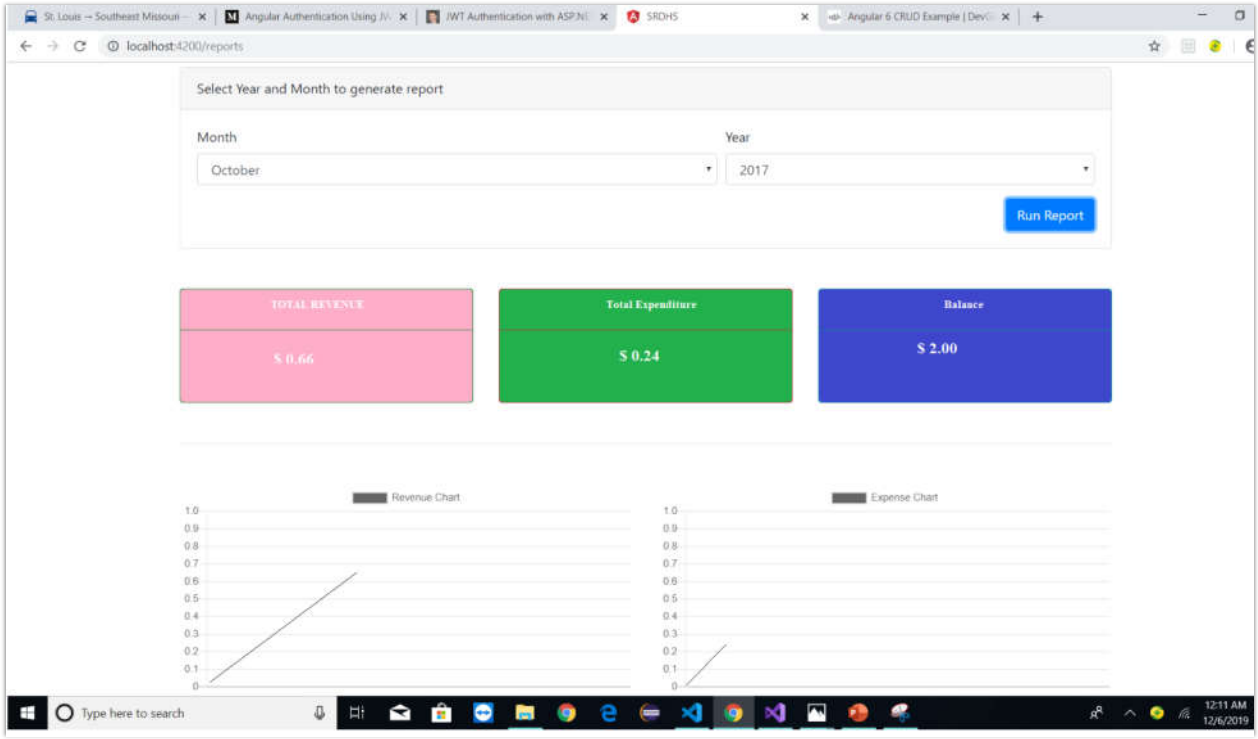


SQL fetch of the data from SQL server 2008:



Graphical Reporting:

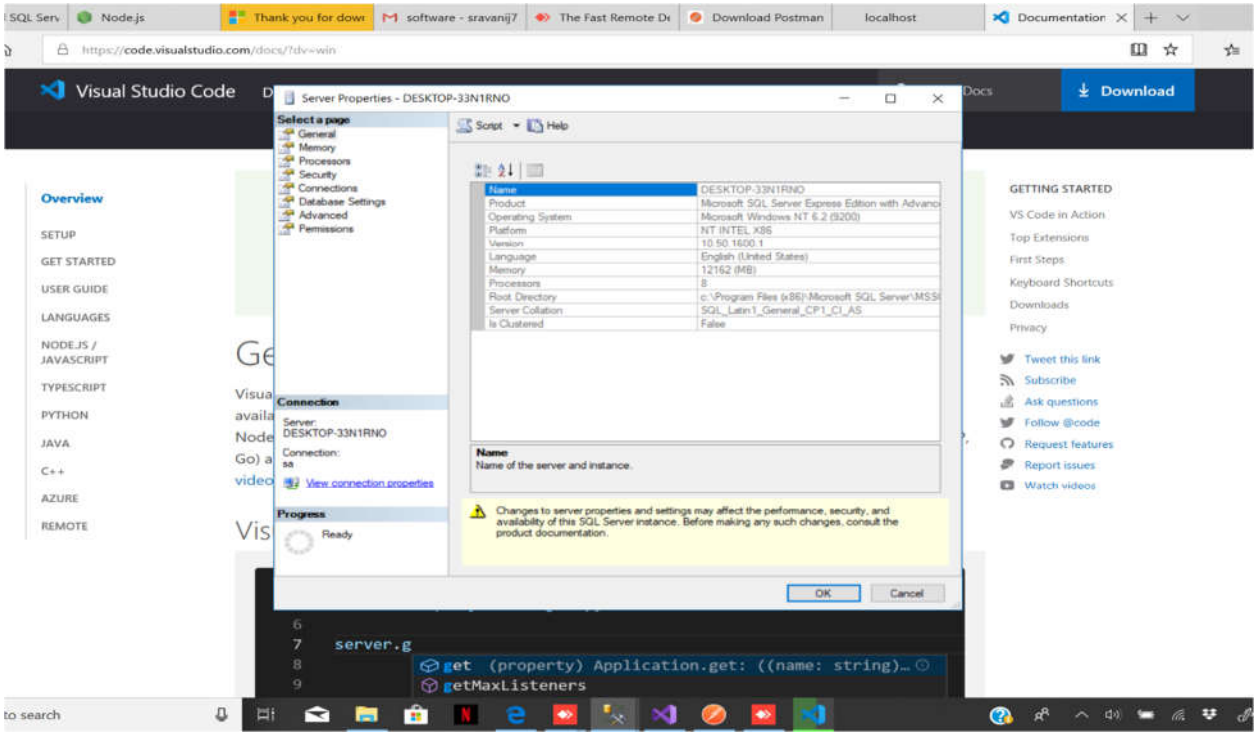
Since we have the data to be projected, we have tried to plot the values of revenue and expenditure over a graph. Please find the screenshot below:



Database Management:

Data management is done to ensure the accessibility and reliability of the data for its users. The data management solution has made processing and validation of the data simpler. We have two main built-in tools for interacting with a SQL Server database platform: SQL Server Management Studio (SSMS) and SQL Server Data Tools (SSDT). For this project, we have used SQL Sever 2008 which provided interface for connecting and working with MS SQL server for our project. Data management and data visualization are easier than ever with new SQL Server. It also helps maintain a single integrated environment for SQL Server Database Engine.

Server Properties:



C# ASP.NET Core Entity Framework:

ASP.NET core Entity framework enabled us in implementing model, view, controller framework.

Accounts Controller for Payments:

```
using System.Threading.Tasks;
using AngularSRDHSWebApiAuth.Data;
using AngularSRDHSWebApiAuth.Helpers;
using AngularSRDHSWebApiAuth.Models.Entities;
using AngularSRDHSWebApiAuth.ViewModels;
using AutoMapper;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;


namespace AngularSRDHSWebApiAuth.Controllers
{
    [Route("api/[controller]")]
    public class AccountsController : Controller
    {
        private readonly ApplicationDbContext _appDbContext;
        private readonly UserManager<AppUser> _userManager;
        private readonly IMapper _mapper;

        public AccountsController(UserManager<AppUser> userManager, IMapper mapper, ApplicationDbContext appDbContext)
        {
            _userManager = userManager;
            _mapper = mapper;
            _appDbContext = appDbContext;
        }

        // POST api/accounts
```

JWT Authorization controller:

```
    [Route("api/[controller]")]
    public class AuthController : Controller
    {
        private readonly UserManager<AppUser> _userManager;
        private readonly IJwtFactory _jwtFactory;
        private readonly JwtIssuerOptions _jwtOptions;

        public AuthController(UserManager<AppUser> userManager, IJwtFactory jwtFactory, IOptions<JwtIssuerOptions> jwtOptions)
        {
            _userManager = userManager;
            _jwtFactory = jwtFactory;
            _jwtOptions = jwtOptions.Value;
        }

        // POST api/auth/login
        [HttpPost("login")]
        public async Task<IActionResult> Post([FromBody]CredentialsViewModel credentials)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var identity = await GetClaimsIdentity(credentials.UserName, credentials.Password);
            if (identity == null)
            {
                return BadRequest(Errors.AddErrorToModelState("login_failure", "Invalid username or password.", ModelState));
            }
```

Users JWT authentication:

```
AccountsController.cs*   Customer.cs   AppUser.cs   IJwtFactory.cs   JwtFactory.cs   AuthController.cs   DashboardController.cs

namespace AngularASPNETCore2WebApiAuth.Controllers
{
  [Authorize(Policy = "ApiUser")]
  [Route("api/[controller]/[action]")]
  public class DashboardController : Controller
  {
    private readonly ClaimsPrincipal _caller;
    private readonly ApplicationDbContext _appDbContext;

    public DashboardController(UserManager<AppUser> userManager, ApplicationDbContext appDbContext, IHttpContextAccessor httpContextAccessor)
    {
      _caller = httpContextAccessor.HttpContext.User;
      _appDbContext = appDbContext;
    }

    // GET api/dashboard/home
    [HttpGet]
    public async Task<IActionResult> Home()
    {
      // retrieve the user info
      //HttpContext.User
      var userId = _caller.Claims.Single(c => c.Type == "id");
      var customer = await _appDbContext.Customers.Include(c => c.Identity).SingleAsync(c => c.Identity.Id == userId.Value);

      return new OkObjectResult(new
      {
        Message = "This is secure API and user data!",
        customer.Identity.FirstName
```

Invalid User authentication logic:

```
Select Startup Item...
AccountsController.cs*   AppUser.cs   IJwtFactory.cs   JwtFactory.cs   AuthController.cs   DashboardController.cs   ExternalAuthController.cs

        UserName = userInfo.Email,
        PictureUrl = userInfo.Picture.Data.Url
      };

      var result = await _userManager.CreateAsync(appUser, Convert.ToBase64String(Guid.NewGuid().ToByteArray()).Substring(0, 8));

      if (!result.Succeeded) return new BadRequestObjectResult(Errors.AddErrorsToModelState(result, ModelState));

      await _appDbContext.Customers.AddAsync(new Customer { IdentityId = appUser.Id, Location = "",Locale = userInfo.Locale,Gender =
      await _appDbContext.SaveChangesAsync();
    }

    // generate the jwt for the local user...
    var localUser = await _userManager.FindByNameAsync(userInfo.Email);

    if (localUser==null)
    {
      return BadRequest(Errors.AddErrorToModelState("login_failure", "Failed to create local user account.", ModelState));
    }

    var jwt = await Tokens.GenerateJwt(_jwtFactory.GenerateClaimsIdentity(localUser.UserName, localUser.Id),
      _jwtFactory, localUser.UserName, _jwtOptions, new JsonSerializerSettings {Formatting = Formatting.Indented});

    return new OkObjectResult(jwt);
  }
 }
}
```

Property file:

```
AccountsController.cs*   AuthController.cs   DashboardController.cs   ExternalAuthController.cs   ResponseExtensions.cs
25                  {
26                      b.Property<string>("Id")
27                          .ValueGeneratedOnAdd();
28
29                      b.Property<int>("AccessFailedCount");
30
31                      b.Property<string>("ConcurrencyStamp")
32                          .IsConcurrencyToken();
33
34                      b.Property<string>("Email")
35                          .HasMaxLength(256);
36
37                      b.Property<bool>("EmailConfirmed");
38
39                      b.Property<long?>("FacebookId");
40
41                      b.Property<string>("FirstName");
42
43                      b.Property<string>("LastName");
44
45                      b.Property<bool>("LockoutEnabled");
46
47                      b.Property<DateTimeOffset?>("LockoutEnd");
48
49                      b.Property<string>("NormalizedEmail")
50                          .HasMaxLength(256);
51
52                      b.Property<string>("NormalizedUserName")
53                          .HasMaxLength(256);
100 %   No issues found
```

DB Context Module logic:

```
File  Edit  View  Tools  Extensions  Window  Help   Search (Ctrl+Q)   AngularASPNETCore2WebApiAuth-master
Select Startup Item...
DashboardController.cs   ExternalAuthController.cs   ResponseExtensions.cs   20180104134211_initial.Designer.cs   ApplicationDbContextModelSnapshot.cs
226                 {
227                     b.HasOne("AngularASPNETCore2WebApiAuth.Models.Entities.AppUser")
228                         .WithMany()
229                         .HasForeignKey("UserId")
230                         .OnDelete(DeleteBehavior.Cascade);
231                 });
232
233             modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>
234                 {
235                     b.HasOne("AngularASPNETCore2WebApiAuth.Models.Entities.AppUser")
236                         .WithMany()
237                         .HasForeignKey("UserId")
238                         .OnDelete(DeleteBehavior.Cascade);
239                 });
240
241             modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>
242                 {
243                     b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole")
244                         .WithMany()
245                         .HasForeignKey("RoleId")
246                         .OnDelete(DeleteBehavior.Cascade);
247
248                     b.HasOne("AngularASPNETCore2WebApiAuth.Models.Entities.AppUser")
249                         .WithMany()
250                         .HasForeignKey("UserId")
251                         .OnDelete(DeleteBehavior.Cascade);
252                 });
253
254             modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>
100 %   No issues found   Ln: 1   Ch: 1
Output
Show output from: Build
```

Issuer logic of JwtIssuerOption:

```
ExternalAuthController.cs   ResponseExtensions.cs   20180104134211_initial.Designer.cs   ApplicationDbContextModelSnapshot.cs   JwtIssuerOptions.cs

        /// <summary>
        /// 4.1.5.  "nbf" (Not Before) Claim - The "nbf" (not before) claim identifies the time before which the JWT MUST NOT be accepted f
        /// </summary>
        public DateTime NotBefore => DateTime.UtcNow;

        /// <summary>
        /// 4.1.6.  "iat" (Issued At) Claim - The "iat" (issued at) claim identifies the time at which the JWT was issued.
        /// </summary>
        public DateTime IssuedAt => DateTime.UtcNow;

        /// <summary>
        /// Set the timespan the token will be valid for (default is 120 min)
        /// </summary>
        public TimeSpan ValidFor { get; set; } = TimeSpan.FromMinutes(120);



        /// <summary>
        /// "jti" (JWT ID) Claim (default ID is a GUID)
        /// </summary>
        public Func<Task<string>> JtiGenerator =>
          () => Task.FromResult(Guid.NewGuid().ToString());

        /// <summary>
        /// The signing key to use when generating tokens.
        /// </summary>
        public SigningCredentials SigningCredentials { get; set; }
    }
}
```

Get and set values:

```
AuditableEntity.cs

using System;
using System.ComponentModel.DataAnnotations;

namespace AspNetCoreSpa.Core.Entities
{
    public class AuditableEntity : IAuditableEntity
    {
        [Required]
        public Guid UserId { get; set; }
        [MaxLength(256)]
        public string CreatedBy { get; set; }
        [MaxLength(256)]
        public string UpdatedBy { get; set; }
        public DateTime UpdatedDate { get; set; }
        public DateTime CreatedDate { get; set; }
    }
}
```

Code connection in angular for database:

```
20191027153340_initial.cs   HomeController.cs   ServicesCollectionExtension.cs   SeedData.cs   IUnitOfWork.cs   HttpUnitOfWork.cs
using System;
using Microsoft.EntityFrameworkCore.Migrations;

namespace AspNetCoreSpa.Web.Migrations
{
    public partial class initial : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Payment",
                columns: table => new
                {
                    Id = table.Column<int>(nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    Name = table.Column<string>(nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Cultures", x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "Customers",
                columns: table => new
                {
                    Id = table.Column<int>(nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    UserId = table.Column<Guid>(nullable: false)
100 %   No issues found
Output
```

Contribution:

6.Mounisha Bhashyam	7.Sravani Boppana	10. Manasa Devidi	15.Deepika Golla	24.Geetika Koneru	35.Nisha Shahi	38.Yeshaswini Sukavasi
14.28%	**14.28%**	**14.28%**	**14.28%**	**14.28%**	**14.28%**	**14.28%**
Angular Integration	Angular Integration	Angular Integration	Angular Integration	JWT Authentication	Login and Sign Up	Angular Integration
Service Request Page	Tracking Page	CRUD Page Layout Dealer	Feedback, CRUD Payment layout in angular	CRUD implementation Dealer	CRUD implementation Payment	JavaScript Validations
BootStrap	Calender and JavaScript	Update Operations	Insert Operations	Backend C# Models, Controllers	Contact Us	Delete Operations
				Graph Generation	Java Script Validations	