Project Source Code

API Code

Model

Model/Medicine.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ABCHealthcare.Model
{
    public class Medicine
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public long Price { get; set; }
        public string Image { get; set; }
        public string Seller { get; set; }
        public string Description { get; set; }
        public int Quantity { get; set; }
        public string Category { get; set; }
    }
}
```

Model/Cart.cs

```
using System.Collections.Generic;
using System.Linq;

namespace ABCHealthcare.Model
{
    public class Cart
    {
        public int Id { get; set; }
        public string BuyerId { get; set; }
        public List<CartItem> Items { get; set; } = new List<CartItem>();
        public void AddItem(Medicine medicine, int quantity)
        {

            if (Items.All(item => item.MedicineId != medicine.Id))
            {
                Items.Add(new CartItem { Medicine = medicine, Quantity =
quantity });
            }
            var existingItem = Items.FirstOrDefault(item => item.MedicineId ==
medicine.Id);
            if (existingItem != null) existingItem.Quantity =
existingItem.Quantity + quantity;
        }

        public void RemoveItem(int medicineId, int quantity)
        {
            var item = Items.FirstOrDefault(item => item.MedicineId ==
medicineId);
            if (item == null) return;
            item.Quantity = item.Quantity - quantity;
            if (item.Quantity == 0) Items.Remove(item);
        }
    }
}
```

Model/CartItem.cs

```
using System.ComponentModel.DataAnnotations.Schema;

namespace ABCHealthcare.Model
{
    [Table("CartItems")]
    public class CartItem
    {
        public int Id { get; set; }
        public int Quantity { get; set; }


        public int MedicineId { get; set; }
        public Medicine Medicine { get; set; }

        public int CartId { get; set; }
        public Cart Cart { get; set; }
    }
}
```

Model/User.cs

```
using Microsoft.AspNetCore.Identity;

namespace ABCHealthcare.Model
{
    public class User : IdentityUser
    {

    }
}
```

DataTransferObjects

DataTransferObjects/CartDto.cs

```
using System.Collections.Generic;

namespace ABCHealthcare.DataTransferObjects
{
    public class CartDto
    {
        public int Id { get; set; }
        public string BuyerId { get; set; }
        public List<CartItemDto> Items { get; set; }
    }
}
```

DataTransferObjects/CartItemDto.cs

```
namespace ABCHealthcare.DataTransferObjects
{
    public class CartItemDto
    {
        public int MedicineId { get; set; }
        public string Name { get; set; }
        public long Price { get; set; }
        public string Image { get; set; }
        public string Seller { get; set; }
        public string Description { get; set; }
        public int Quantity { get; set; }
        public string Category { get; set; }
    }
}
```

DataTransferObjects/UserDto.cs

```
namespace ABCHealthcare.DataTransferObjects
{
    public class UserDto
    {
        public string Email { get; set; }
        public string Token { get; set; }
    }
}
```

DataTransferObjects/LoginDto.cs

```
namespace ABCHealthcare.DataTransferObjects
{
    public class LoginDto
    {
        public string Username { get; set; }
        public string Password { get; set; }
    }
}
```

DataTransferObjects/RegisterDto.cs

```
namespace ABCHealthcare.DataTransferObjects
{
    public class RegisterDto : LoginDto
    {
        public string Email { get; set; }

    }
}
```

Controllers

Controllers/MedicinesController.cs

```
using ABCHealthcare.Model;
using ABCHealthcare.Store;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ABCHealthcare.Controllers
{
    [Route("api/[controller]")]
    [ApiController]

    public class MedicinesController : ControllerBase
    {

        private readonly StoreContext _context;

        public MedicinesController(StoreContext context)
        {
            _context = context;
        }

        [HttpGet]
        public async Task<ActionResult<List<Medicine>>> GetProducts()
        {
            return await _context.Medicines.ToListAsync();
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<Medicine>> GetProduct(int id)
        {
            return await _context.Medicines.FindAsync(id);
        }
    }
}
```

Controllers/CartController.cs

```
using ABCHealthcare.DataTransferObjects;
using ABCHealthcare.Model;
using ABCHealthcare.Store;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System;
using System.Linq;
using System.Threading.Tasks;

namespace ABCHealthcare.Controllers
{
    [Route("api/[controller]")]
    [ApiController]

    public class CartController : ControllerBase
    {

        private readonly StoreContext _context;

        public CartController(StoreContext context)
        {
            _context = context;
        }

        [HttpGet(Name = "GetCart")]
        public async Task<ActionResult<CartDto>> GetCart()
        {
            var cart = await RetrieveCart();

            if (cart == null) return NotFound();
            return MapCartToDto(cart);
        }

        [HttpPost] // api/Cart?medicineId=2&quantity=5
        public async Task<ActionResult<CartDto>> AddItemToCart(int medicineId,
int quantity)
        {
            // get cart
            var cart = await RetrieveCart();

            // if cart not there, create one
            if (cart == null) cart = CreateCart();

            // get medicine
            var medicine = await _context.Medicines.FindAsync(medicineId);
            if (medicine == null) return NotFound(); // this case should not
arrive as we have proper ID for medicines, still adding for safety

            // add item
            cart.AddItem(medicine, quantity);
            // save changes
            var result = await _context.SaveChangesAsync() > 0; // this method
returns an int for number of changes has been made to the DB
            if (result) return CreatedAtRoute("GetCart", MapCartToDto(cart));
            return BadRequest(new ProblemDetails { Title = "Problem saving
item to cart" });
        }


        [HttpDelete]
        public async Task<ActionResult> RemoveCartItem(int medicineId, int
quantity)
        {
            // get cart
            var cart = await RetrieveCart();
            if (cart == null) return NotFound();

            // remove item or reduce quantity
            cart.RemoveItem(medicineId, quantity);

            // save changes
            var result = await _context.SaveChangesAsync() > 0;
            if (result) return Ok();
            return BadRequest(new ProblemDetails { Title = "Problem removing
item from cart" });
        }

        private async Task<Cart> RetrieveCart()
        {
            return await _context.Carts
                .Include(i => i.Items)
                .ThenInclude(m => m.Medicine)
                .FirstOrDefaultAsync(x => x.BuyerId ==
Request.Cookies["buyerId"]);
        }

        private Cart CreateCart()
        {
            var buyerId = Guid.NewGuid().ToString();
            var cookieOptions = new CookieOptions { IsEssential = true,
Expires = DateTime.Now.AddDays(30) };
            Response.Cookies.Append("buyerId", buyerId, cookieOptions);
            var cart = new Cart { BuyerId = buyerId };
            _context.Carts.Add(cart);
            return cart;
        }

        private CartDto MapCartToDto(Cart cart)
        {
            return new CartDto
            {
                Id = cart.Id,
                BuyerId = cart.BuyerId,
                Items = cart.Items.Select(item => new CartItemDto
                {
                    MedicineId = item.MedicineId,
                    Name = item.Medicine.Name,
                    Price = item.Medicine.Price,
                    Image = item.Medicine.Image,
                    Seller = item.Medicine.Seller,
                    Description = item.Medicine.Description,
                    Quantity = item.Quantity,
                    Category = item.Medicine.Category
                }).ToList()
            };
        }
    }
}
```

Controllers/JournalController.cs

```
using ABCHealthcare.DataTransferObjects;
using ABCHealthcare.Model;
using ABCHealthcare.Resources;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using System.Threading.Tasks;

namespace ABCHealthcare.Controllers
{

    [Route("api/[controller]")]
    [ApiController]
    public class JournalController : ControllerBase
    {

        private readonly UserManager<User> _userManager;

        private readonly Token _tokenService;

        public JournalController(UserManager<User> userManager, Token 
tokenService)
        {
            _userManager = userManager;
            _tokenService = tokenService;
        }

        [HttpPost("login")]
        public async Task<ActionResult<UserDto>> Login(LoginDto loginDto)
        {
            var user = await _userManager.FindByNameAsync(loginDto.Username);
            if (user == null || !await _userManager.CheckPasswordAsync(user, 
loginDto.Password)) return Unauthorized();
```

```
            return new UserDto
            {
                Email = user.Email,
                Token = await _tokenService.GenerateToken(user)
            };
        }

        [HttpPost("register")]
        public async Task<ActionResult> Register(RegisterDto registerDto)
        {
            var user = new User { UserName = registerDto.Username, Email =
registerDto.Email };
            var result = await _userManager.CreateAsync(user,
registerDto.Password);
            if (!result.Succeeded)
            {
                return ValidationProblem();
            }

            await _userManager.AddToRoleAsync(user, "User");
            return StatusCode(201);
        }

        [Authorize]
        [HttpGet("currentUser")]
        public async Task<ActionResult<UserDto>> GetCurrentUser()
        {
            var user = await _userManager.FindByNameAsync(User.Identity.Name);

            return new UserDto
            {
                Email = user.Email,
                Token = await _tokenService.GenerateToken(user)
            };
        }
    }
}
```

Store

Store/StoreContext.cs

```
using ABCHealthcare.Model;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ABCHealthcare.Store
{
    public class StoreContext : IdentityDbContext<User>
    {
        public StoreContext(DbContextOptions options) : base(options)
        {
        }

        public DbSet<Medicine> Medicines { get; set; }
        public DbSet<Cart> Carts { get; set; }

        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);

            builder.Entity<IdentityRole>()
                .HasData(
                    new IdentityRole { Name = "User", NormalizedName = "USER"
},
                    new IdentityRole { Name = "Admin", NormalizedName =
"ADMIN" }
                );
        }
    }
}
```

Store/Initializer.cs

```
using ABCHealthcare.Model;
using Microsoft.AspNetCore.Identity;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ABCHealthcare.Store
{
    public static class Initializer
    {
        public static async Task Initialize(StoreContext context,
UserManager<User> userManager)
        {
            if (!userManager.Users.Any())
            {
                var user = new User
                {
                    UserName = "Sahil",
                    Email = "sahil123@test.com"
                };
                await userManager.CreateAsync(user, "Password@12345");
                await userManager.AddToRoleAsync(user, "User");


                var admin = new User
                {
                    UserName = "admin",
                    Email = "admin123@test.com"
                };
                await userManager.CreateAsync(admin, "Password@12345");
                await userManager.AddToRolesAsync(admin, new[] { "User",
"Admin" });

            }

            if (context.Medicines.Any()) return;
            var medicines = new List<Medicine>
            {
                new Medicine
                {
                    Name = "Crocin",
                    Price = 20,
                    Image = "https://5.imimg.com/data5/OU/XS/MY-
53366293/crocin-500x500.jpg",
                    Seller = "XYZ Suppliers",
                    Description = "Crocin Pain Relief provides targeted pain
relief. It provides symptomatic relief from mild to moderate pain e.g from
headache, migraine, toothache.",
                    Quantity = 15,
                    Category = "Painkiller"
                },

                new Medicine
                {
                    Name = "Paracetamol",
                    Price = 18,
                    Image =
"https://5.imimg.com/data5/SELLER/Default/2022/9/IV/UY/CG/75459511/500mg-
paracetamol-tablet-250x250.jpg",
                    Seller = "XYZ Suppliers",
                    Description = "Paracetamol is a common painkiller used to
treat aches and pain. It can also be used to reduce a high temperature.",
                    Quantity = 12,
                    Category = "Painkiller"
                },

                new Medicine
                {
                    Name = "Azithral 500",
                    Price = 25,
                    Image =
"https://newassets.apollo247.com/pub/media/catalog/product/a/z/azi0013_1.jpg",
                    Seller = "ABC Suppliers",
                    Description = "Azithral 500 Tablet is an antibiotic used
to treat various types of bacterial infections of the respiratory tract, ear,
nose, throat, lungs, skin, and eye in adults and children.",
                    Quantity = 5,
                    Category = "Antibiotic"
                },

                new Medicine
                {
                    Name = "Azee 500",
                    Price = 20,
                    Image =
"https://5.imimg.com/data5/SELLER/Default/2022/4/MG/JO/VM/31640038/medzee-500-
tablets-250x250.jpg",
                    Seller = "ABC Suppliers",
                    Description = "Azee 500 Tablet is an antibiotic used to
treat various types of bacterial infections of the respiratory tract, ear,
nose, throat, lungs, skin, and eye in adults and children. It is also
effective in typhoid fever and some sexually transmitted diseases like
gonorrhea.",
                    Quantity = 5,
                    Category = "Antibiotic"
                },

                new Medicine
                {
                    Name = "Avil Injection",
                    Price = 5,
                    Image = "https://5.imimg.com/data5/JH/HI/MG/SELLER-
16645300/avil-inj-250x250.jpg",
                    Seller = "PQR Suppliers",
                    Description = "Avil Injection is an antiallergic
medication. It is used to treats symptoms of allergic conditions caused by
insect bites/stings, certain medicines, or hives (rashes, swelling, etc.).",
                    Quantity = 1,
                    Category = "Injection"
                },

                new Medicine
                {
                    Name = "Dolo 650",
                    Price = 26,
                    Image = "https://5.imimg.com/data5/SU/FN/MY-53366293/dolo-
65-250x250.jpg",
                    Seller = "XYZ Suppliers",
                    Description = "Dolo 650 Tablet helps relieve pain and
fever by blocking the release of certain chemical messengers responsible for
fever and pain. It is used to treat headaches, migraine, nerve pain,
toothache, sore throat, period (menstrual) pains, arthritis, muscle aches, and
the common cold.",
                    Quantity = 15,
                    Category = "Painkiller"
                },

                new Medicine
                {
                    Name = "Cefix 200",
                    Price = 10,
                    Image =
"https://5.imimg.com/data5/SELLER/Default/2021/10/DZ/UE/PQ/63235102/cefix-
cefixime-200mg-tablets-250x250.jpg",
                    Seller = "ABC Suppliers",
                    Description = "Cefix 200 Tablet is an antibiotic belonging
that is used to treat a variety of bacterial infections. It is effective in
infections of the respiratory tract (eg. pneumonia), urinary tract, ear, nasal
sinus, throat, and some sexually transmitted diseases.",
                    Quantity = 10,
                    Category = "Antibiotic "
                },

                new Medicine
                {
                    Name = "Enzoflam",
                    Price = 15,
                    Image =
"https://5.imimg.com/data5/SELLER/Default/2022/5/KS/TE/HE/136261961/n2gesepigq
muphnnwupw-250x250.jpg",
                    Seller = "XYZ Suppliers",
                    Description = "Enzoflam Tablet is a pain-relieving
medicine. It helps in relieving moderate pain and reducing fever. It is used
in various conditions such as muscle ache, back pain, joint pain, menstrual
cramps, and toothache.",
                    Quantity = 10,
                    Category = "Painkiller "
                },

            };

            foreach (var medicine in medicines)
            {
                context.Medicines.Add(medicine);
            }

            context.SaveChanges();
        }
    }
}
```

Migrations

Store/Migrations/InitialCreate.cs

```
using Microsoft.EntityFrameworkCore.Migrations;

namespace ABCHealthcare.Store.Migrations
{
    public partial class InitialCreate : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Medicines",
                columns: table => new
                {
                    Id = table.Column<int>(type: "INTEGER", nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    Name = table.Column<string>(type: "TEXT", nullable: true),
                    Price = table.Column<long>(type: "INTEGER", nullable:
false),
                    Image = table.Column<string>(type: "TEXT", nullable:
true),
                    Seller = table.Column<string>(type: "TEXT", nullable:
true),
                    Description = table.Column<string>(type: "TEXT", nullable:
true),
                    Quantity = table.Column<int>(type: "INTEGER", nullable:
false),
                    Category = table.Column<string>(type: "TEXT", nullable:
true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Medicines", x => x.Id);
                });
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Medicines");
        }
    }
}
```

Store/Migrations/CartEntityAdded.cs

```
using Microsoft.EntityFrameworkCore.Migrations;

namespace ABCHealthcare.Store.Migrations
{
    public partial class CartEntityAdded : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Carts",
                columns: table => new
                {
                    Id = table.Column<int>(type: "INTEGER", nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    BuyerId = table.Column<string>(type: "TEXT", nullable:
true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Carts", x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "CartItems",
                columns: table => new
                {
                    Id = table.Column<int>(type: "INTEGER", nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    Quantity = table.Column<int>(type: "INTEGER", nullable:
false),
                    MedicineId = table.Column<int>(type: "INTEGER", nullable:
false),
                    CartId = table.Column<int>(type: "INTEGER", nullable:
false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_CartItems", x => x.Id);
                    table.ForeignKey(
                        name: "FK_CartItems_Carts_CartId",
                        column: x => x.CartId,
                        principalTable: "Carts",
                        principalColumn: "Id",
                        onDelete: ReferentialAction.Cascade);
                    table.ForeignKey(
                        name: "FK_CartItems_Medicines_MedicineId",
                        column: x => x.MedicineId,
                        principalTable: "Medicines",
                        principalColumn: "Id",
                        onDelete: ReferentialAction.Cascade);
                });
            migrationBuilder.CreateIndex(
                name: "IX_CartItems_CartId",
                table: "CartItems",
                column: "CartId");

            migrationBuilder.CreateIndex(
                name: "IX_CartItems_MedicineId",
                table: "CartItems",
                column: "MedicineId");
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "CartItems");

            migrationBuilder.DropTable(
                name: "Carts");
        }
    }
}
```

Store/Migrations/IdentityAdded.cs

```
using System;
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace ABCHealthcare.Store.Migrations
{
    public partial class IdentityAdded : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "AspNetRoles",
                columns: table => new
                {
                    Id = table.Column<string>(type: "TEXT", nullable: false),
                    Name = table.Column<string>(type: "TEXT", maxLength: 256,
nullable: true),
                    NormalizedName = table.Column<string>(type: "TEXT",
maxLength: 256, nullable: true),
                    ConcurrencyStamp = table.Column<string>(type: "TEXT",
nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_AspNetRoles", x => x.Id);
                });
            migrationBuilder.CreateTable(
                name: "AspNetUsers",
                columns: table => new
                {
                    Id = table.Column<string>(type: "TEXT", nullable: false),
                    UserName = table.Column<string>(type: "TEXT", maxLength:
256, nullable: true),
                    NormalizedUserName = table.Column<string>(type: "TEXT",
maxLength: 256, nullable: true),
                    Email = table.Column<string>(type: "TEXT", maxLength: 256,
nullable: true),
                    NormalizedEmail = table.Column<string>(type: "TEXT",
maxLength: 256, nullable: true),
                    EmailConfirmed = table.Column<bool>(type: "INTEGER",
nullable: false),
                    PasswordHash = table.Column<string>(type: "TEXT",
nullable: true),
                    SecurityStamp = table.Column<string>(type: "TEXT",
nullable: true),
                    ConcurrencyStamp = table.Column<string>(type: "TEXT",
nullable: true),
                    PhoneNumber = table.Column<string>(type: "TEXT", nullable:
true),
                    PhoneNumberConfirmed = table.Column<bool>(type: "INTEGER",
nullable: false),
                    TwoFactorEnabled = table.Column<bool>(type: "INTEGER",
nullable: false),
                    LockoutEnd = table.Column<DateTimeOffset>(type: "TEXT",
nullable: true),
                    LockoutEnabled = table.Column<bool>(type: "INTEGER",
nullable: false),
                    AccessFailedCount = table.Column<int>(type: "INTEGER",
nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_AspNetUsers", x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "AspNetRoleClaims",
                columns: table => new
                {
                    Id = table.Column<int>(type: "INTEGER", nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    RoleId = table.Column<string>(type: "TEXT", nullable:
false),
                    ClaimType = table.Column<string>(type: "TEXT", nullable:
true),
                    ClaimValue = table.Column<string>(type: "TEXT", nullable:
true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_AspNetRoleClaims", x => x.Id);
                    table.ForeignKey(
                        name: "FK_AspNetRoleClaims_AspNetRoles_RoleId",
                        column: x => x.RoleId,
                        principalTable: "AspNetRoles",
                        principalColumn: "Id",
                        onDelete: ReferentialAction.Cascade);
                });

            migrationBuilder.CreateTable(
                name: "AspNetUserClaims",
                columns: table => new
                {
                    Id = table.Column<int>(type: "INTEGER", nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    UserId = table.Column<string>(type: "TEXT", nullable:
false),
                    ClaimType = table.Column<string>(type: "TEXT", nullable:
true),
                    ClaimValue = table.Column<string>(type: "TEXT", nullable:
true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_AspNetUserClaims", x => x.Id);
                    table.ForeignKey(
                        name: "FK_AspNetUserClaims_AspNetUsers_UserId",
                        column: x => x.UserId,
                        principalTable: "AspNetUsers",
                        principalColumn: "Id",
                        onDelete: ReferentialAction.Cascade);
                });

            migrationBuilder.CreateTable(
                name: "AspNetUserLogins",
                columns: table => new
                {
                    LoginProvider = table.Column<string>(type: "TEXT",
nullable: false),
                    ProviderKey = table.Column<string>(type: "TEXT", nullable:
false),
                    ProviderDisplayName = table.Column<string>(type: "TEXT",
nullable: true),
                    UserId = table.Column<string>(type: "TEXT", nullable:
false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_AspNetUserLogins", x => new {
x.LoginProvider, x.ProviderKey });
                    table.ForeignKey(
                        name: "FK_AspNetUserLogins_AspNetUsers_UserId",
                        column: x => x.UserId,
                        principalTable: "AspNetUsers",
                        principalColumn: "Id",
                        onDelete: ReferentialAction.Cascade);
                });

            migrationBuilder.CreateTable(
                name: "AspNetUserRoles",
                columns: table => new
                {
                    UserId = table.Column<string>(type: "TEXT", nullable:
false),
                    RoleId = table.Column<string>(type: "TEXT", nullable:
false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_AspNetUserRoles", x => new {
x.UserId, x.RoleId });
                    table.ForeignKey(
                        name: "FK_AspNetUserRoles_AspNetRoles_RoleId",
                        column: x => x.RoleId,
                        principalTable: "AspNetRoles",
                        principalColumn: "Id",
                        onDelete: ReferentialAction.Cascade);
                    table.ForeignKey(
                        name: "FK_AspNetUserRoles_AspNetUsers_UserId",
                        column: x => x.UserId,
                        principalTable: "AspNetUsers",
                        principalColumn: "Id",
                        onDelete: ReferentialAction.Cascade);
                });

            migrationBuilder.CreateTable(
                name: "AspNetUserTokens",
                columns: table => new
                {
                    UserId = table.Column<string>(type: "TEXT", nullable:
false),
                    LoginProvider = table.Column<string>(type: "TEXT",
nullable: false),
                    Name = table.Column<string>(type: "TEXT", nullable:
false),
                    Value = table.Column<string>(type: "TEXT", nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_AspNetUserTokens", x => new {
x.UserId, x.LoginProvider, x.Name });
                    table.ForeignKey(
                        name: "FK_AspNetUserTokens_AspNetUsers_UserId",
                        column: x => x.UserId,
                        principalTable: "AspNetUsers",
                        principalColumn: "Id",
                        onDelete: ReferentialAction.Cascade);
                });
            migrationBuilder.InsertData(
                table: "AspNetRoles",
                columns: new[] { "Id", "ConcurrencyStamp", "Name",
"NormalizedName" },
                values: new object[] { "7593666d-60b0-4063-ace6-1fb7020ef77e",
"61257d0d-f11a-4164-ba63-22eef15e2b70", "User", "USER" });

            migrationBuilder.InsertData(
                table: "AspNetRoles",
                columns: new[] { "Id", "ConcurrencyStamp", "Name",
"NormalizedName" },
                values: new object[] { "9a1679fd-fa3e-4b19-a7a9-0b64e0f45f75",
"b1946f9d-95bd-4a67-9ae4-cb099fac485e", "Admin", "ADMIN" });

            migrationBuilder.CreateIndex(
                name: "IX_AspNetRoleClaims_RoleId",
                table: "AspNetRoleClaims",
                column: "RoleId");

            migrationBuilder.CreateIndex(
                name: "RoleNameIndex",
                table: "AspNetRoles",
                column: "NormalizedName",
                unique: true);

            migrationBuilder.CreateIndex(
                name: "IX_AspNetUserClaims_UserId",
                table: "AspNetUserClaims",
                column: "UserId");

            migrationBuilder.CreateIndex(
                name: "IX_AspNetUserLogins_UserId",
                table: "AspNetUserLogins",
                column: "UserId");

            migrationBuilder.CreateIndex(
                name: "IX_AspNetUserRoles_RoleId",
                table: "AspNetUserRoles",
                column: "RoleId");

            migrationBuilder.CreateIndex(
                name: "EmailIndex",
                table: "AspNetUsers",
                column: "NormalizedEmail");

            migrationBuilder.CreateIndex(
                name: "UserNameIndex",
                table: "AspNetUsers",
                column: "NormalizedUserName",
                unique: true);
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "AspNetRoleClaims");

            migrationBuilder.DropTable(
                name: "AspNetUserClaims");

            migrationBuilder.DropTable(
                name: "AspNetUserLogins");

            migrationBuilder.DropTable(
                name: "AspNetUserRoles");

            migrationBuilder.DropTable(
                name: "AspNetUserTokens");

            migrationBuilder.DropTable(
                name: "AspNetRoles");

            migrationBuilder.DropTable(
                name: "AspNetUsers");
        }
    }
}
```

Resources

Resources/Token.cs

```
using ABCHealthcare.Model;
using Microsoft.AspNetCore.Identity;
using Microsoft.Extensions.Configuration;
using Microsoft.IdentityModel.Tokens;
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using System.Threading.Tasks;

namespace ABCHealthcare.Resources
{
    public class Token
    {
        private readonly UserManager<User> _userManager;
        private readonly IConfiguration _config;

        public Token(UserManager<User> userManager, IConfiguration config)
        {
            _userManager = userManager;
            _config = config;
        }
        public async Task<string> GenerateToken(User user)
        {
            var claims = new List<Claim>
            {
                new Claim(ClaimTypes.Email,user.Email),
                new Claim(ClaimTypes.Name, user.UserName)
            };

            var roles = await _userManager.GetRolesAsync(user);
            foreach (var role in roles)
            {
                claims.Add(new Claim(ClaimTypes.Role, role));
            }

            var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config["JWTSettings:TokenKey"]));
            var creds = new SigningCredentials(key,
SecurityAlgorithms.HmacSha512);

            var tokenOptions = new JwtSecurityToken(
                issuer: null,
                audience: null,
                claims: claims,
                expires: DateTime.Now.AddDays(14),
                signingCredentials: creds
            );

            return new JwtSecurityTokenHandler().WriteToken(tokenOptions);
        }
    }
}
```

Client - React Code

src/index.tsx

```tsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { BrowserRouter } from 'react-router-dom';
import { StoreProvider } from './Context';

const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <StoreProvider>
        <App />
      </StoreProvider>
    </BrowserRouter>
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

src/App.tsx

```tsx
import Inventory from "./features/inventory/Inventory";
import Navbar from "./Navbar";
import { Container } from "@mui/system";
import { Route, Routes } from "react-router-dom";
import HomePage from "./features/home/HomePage";
import MedicineDetails from "./features/inventory/MedicineDetails";
import CartPage from "./features/cart/CartPage";
import { useStoreContext } from "./Context";
import { useEffect } from "react";
import { getCookie } from "./util";
import agent from "./agent";
import CheckoutPage from "./features/checkout/CheckoutPage";
import Register from "./features/journal/Register";
import Login from "./features/journal/Login";

function App() {

  const {setCart} = useStoreContext();
  useEffect(() => {
    const buyerId = getCookie('buyerId')
    if (buyerId) {
      agent.Cart.get()
        .then(cart => setCart(cart))
        .catch(error => console.log(error));
    }
  }, [setCart])

  return (
    <>
      <Navbar />
      <Container>
        <Routes>
          <Route path="/" element={<Register />} />
          <Route path="/login" element={<Login />} />
          <Route path="/homepage" element={<HomePage />} />
          <Route path="/inventory" element={<Inventory/>} />
          <Route path="/inventory/:id" element={<MedicineDetails/>} />
          <Route path="/cart" element={<CartPage />} />
          <Route path='/checkout' element={<CheckoutPage />} />
        </Routes>
      </Container>
    </>
  )
}

export default App;
```

src/Navbar.tsx

```
import { AppBar, Badge, Box, createTheme, IconButton, List, ListItem,
ThemeProvider, Toolbar, Typography } from "@mui/material";
import LocalHospitalIcon from '@mui/icons-material/LocalHospital';
import { Link, NavLink } from "react-router-dom";
import { ShoppingCart } from "@mui/icons-material";
import { useStoreContext } from "./Context";

const darkTheme = createTheme({
    palette: {
        mode: 'dark',
    },
});
const rightLinks = [
    { title: 'INVENTORY', path: '/inventory' },
]
const navStyles = {
    color: 'inherit', textDecoration: 'none', typography: 'h6',
    '&:hover': {
        color: '#b39ddb'
    },
    '&.active': {
        color: '#81d4fa'
    }
}

export default function Navbar() {

    const { cart } = useStoreContext();
    const medicineCount = cart?.items.reduce((sum, item) => sum +
item.quantity, 0)

    return (
        <ThemeProvider theme={darkTheme}>
            <AppBar position="static" sx={{ mb: 4 }}>
                <Toolbar sx={{ display: 'flex', justifyContent: 'space-
between', alignItems: 'center' }}>

                    <Box display='flex' alignItems='center'>
                        <LocalHospitalIcon fontSize="large" />
                        <Typography variant="h6" marginLeft={1}
component={NavLink} end to='/homepage' sx={navStyles} >
                            ABC Healthcare
                        </Typography>
                    </Box>


                    <Box display='flex' alignItems='center'>
                        <List sx={{ display: 'flex' }}>
                            {rightLinks.map(({ title, path }) => (
                                <ListItem
                                    component={NavLink}
                                    to={path}
                                    key={path}
                                    sx={navStyles}
                                >
                                    {title}
                                </ListItem>
                            ))}
                        </List>

                        <IconButton component={Link} to='/cart' size='large'
sx={navStyles}>
                            <Badge badgeContent={medicineCount}
color='primary'>
                                <ShoppingCart />
                            </Badge>
                        </IconButton>
                    </Box>

                </Toolbar>
            </AppBar>
        </ThemeProvider >
    )
}
```

src/agent.ts

```
import axios, { AxiosResponse } from 'axios';

axios.defaults.baseURL = 'http://localhost:5000/api/';
axios.defaults.withCredentials = true;

const responseBody = (response: AxiosResponse) => response.data;

const requests = {
    get: (url: string) => axios.get(url).then(responseBody),
    post: (url: string, body: {}) => axios.post(url, body).then(responseBody),
    put: (url: string, body: {}) => axios.put(url, body).then(responseBody),
    delete: (url: string) => axios.delete(url).then(responseBody),
}

const Inventory = {
    list: () => requests.get('medicines'),
    details: (id: number) => requests.get(`medicines/${id}`)
}

const Cart = {
    get: () => requests.get('cart'),
    addItem: (medicineId: number, quantity = 1) =>
requests.post(`cart?medicineId=${medicineId}&quantity=${quantity}`, {}),
    removeItem: (medicineId: number, quantity = 1) =>
requests.delete(`cart?medicineId=${medicineId}&quantity=${quantity}`),
}

const Journal = {
    login: (values: any) => requests.post('journal/login', values),
    register: (values: any) => requests.post('journal/register', values),
    currentUser: () => requests.get('journal/currentUser'),
}

const agent = {
    Inventory,
    Cart,
    Journal
}

export default agent;
```

src/util.ts

```
export function getCookie(key: string) {
    const b = document.cookie.match("(^|;)\\s*" + key + "\\s*=\\s*([^;]+)");
    return b ? b.pop() : "";
  }
```

models

models/medicine.ts

```
export interface Medicine {
    id: number;
    name: string;
    price: number;
    image: string;
    seller: string;
    description: string;
    quantity: number;
    category: string;
}
```

models/cart.ts

```
export interface CartItem {
    medicineId: number;
    name: string;
    price: number;
    image: string;
    seller: string;
    description: string;
    quantity: number;
    category: string;
}


export interface Cart {
    id: number;
    buyerId: string;
    items: CartItem[];
}
```

models/user.ts

```
export interface User {
    email: string;
    token: string;
}
```

features / journal

features/journal/Register.tsx

```
import { Paper, Typography, Box, TextField, Container, Button } from
"@mui/material";
import { useForm } from "react-hook-form";
import { Link } from "react-router-dom";
import agent from "../../agent";

export default function Register() {
    const { register, handleSubmit, setError, formState: { errors, isValid } }
= useForm({
        mode: 'all'
    });

    function handleApiErrors(errors: any) {
        if (errors) {
            errors.forEach((error: string) => {
                if (error.includes('Password')) {
                    setError('password', { message: error })
                } else if (error.includes('Email')) {
                    setError('email', { message: error })
                } else if (error.includes('Username')) {
                    setError('username', { message: error })
                }
            });
        }
    }

    return (
        <Container component={Paper} maxWidth="sm" sx={{ display: 'flex',
flexDirection: 'column', alignItems: 'center', p: 4 }}>
            <Typography component="h1" variant="h5">
                Register
            </Typography>
            <Box component="form"
                onSubmit={handleSubmit((data) =>
                    agent.Journal.register(data)
                        .catch(error => handleApiErrors(error))
                )}
                noValidate sx={{ mt: 1 }}
            >
                <TextField
                    margin="normal"
                    fullWidth
                    label="Username"
                    autoFocus
                    {...register('username', { required: 'Username is
required' })}
                    error={!!errors.username}
                />
                <TextField
                    margin="normal"
                    fullWidth
                    label="Email address"
                    {...register('email', {
                        required: 'Email is required',
                        pattern: {
                            value: /^\w+[\w-.]@\w+((-\w+)|(\w)).[a-z]{2,3}$/,
                            message: 'Not a valid email address'
                        }
                    })}
                    error={!!errors.email}
                />
                <TextField
                    margin="normal"
                    fullWidth
                    label="Password"
                    type="password"
                    {...register('password', {
                        required: 'Password is required',
                        pattern: {
                            value: /(?=^.{6,10}$)(?=.*\d)(?=.*[a-z])(?=.*[A-
Z])(?=.*[!@#$%^&amp;*()_+}{&quot;:;'?/&gt;.&lt;,])(?!.*\s).*$/,
                            message: 'Password is not complex enough'
                        }
                    })}
                    error={!!errors.password}
                />
                <Button
                    disabled={!isValid}
                    type="submit"
                    fullWidth
                    variant="contained"
                    sx={{ mt: 3 }}
                >
                    Register
                </Button>
                <Button
                    component={Link}
                    to='/login'
                    fullWidth
                    variant="contained"
                    sx={{ mt: 3, mb: 2, '&:hover': { color: 'white' } }}
                >
                    Log In
                </Button>
            </Box>
        </Container>
    );
}
```

features/journal/Login.tsx

```
import { Paper, Typography, Box, TextField, Container, Button } from
"@mui/material";
import { useForm } from "react-hook-form";
import { Link } from "react-router-dom";

export default function Login() {

    const { register, formState: {errors, isValid}} = useForm({
        mode: 'all'
    })

    return (

        <Container component={Paper} maxWidth="sm" sx={{ display: 'flex',
flexDirection: 'column', alignItems: 'center', p: 4 }}>

            <Typography component="h1" variant="h5">
                Log In
            </Typography>

            <Box component="form" noValidate sx={{ mt: 1 }}>
                <TextField
                    margin="normal"
                    fullWidth
                    label="Username"
                    autoFocus
                    {...register('username', { required: 'Username Required'
})}
                    error={!!errors.username}
                />

                <TextField
                    margin="normal"
                    fullWidth
                    label="Password"
                    type="password"
                    {...register('password', { required: 'Password Required'
})}
                    error={!!errors.username}
                />

                <Button
                    disabled={!isValid}
                    component={Link}
                    to='/homepage'
                    fullWidth
                    variant="contained"
                    sx={{ mt: 3 }}
                >
                    Log In
                </Button>
                <Button
                    component={Link}
                    to='/register'
                    fullWidth
                    variant="contained"
                    sx={{ mt: 3, mb: 2, '&:hover': { color: 'white' }, }}
                >
                    Register
                </Button>

            </Box>

        </Container>
    )
}
```

<u>features / home</u>

features/home/HomePage.tsx

```
import 'bootstrap/dist/css/bootstrap.min.css';

import { Carousel } from 'react-bootstrap';
export default function HomePage() {
    return (
        <Carousel>
            <Carousel.Item>
                <img
                    className="d-block w-100"
                    src="https://i.ibb.co/HG8QDgc/1665652096862.jpg"
                    alt="First slide"
                />
            </Carousel.Item>
            <Carousel.Item>
                <img
                    className="d-block w-100"
                    src="https://i.ibb.co/bRR0p9k/2.png"
                    alt="Second slide"
                />
            </Carousel.Item>
        </Carousel>
    )
}
```

features / inventory

features/inventory/Inventory.tsx

```tsx
import { useState, useEffect } from "react";
import agent from "../../agent";
import { Medicine } from "../../models/medicine";
import MedicineList from "./MedicineList";

export default function Inventory() {

    const [medicines, setMedicines] = useState<Medicine[]>([]);

  useEffect(() => {
    agent.Inventory.list().then(medicines => setMedicines(medicines))
  }, [])

    return (
        <>
            <MedicineList medicines={medicines}/>
        </>
    )
}
```

features/inventory/MedicineList.tsx

```tsx
import { Grid } from "@mui/material";
import { Medicine } from "../../models/medicine";
import MedCard from "./MedCard";

interface Props {
    medicines: Medicine[];
}

export default function MedicineList({medicines}: Props) {
    return (

        <Grid container spacing={4}>
            {medicines.map(medicine => (
                <Grid item xs={3} key={medicine.id}>
                    <MedCard medicine={medicine}/>
                </Grid>
            ))}
        </Grid>

    )
}
```

features/inventory/MedCard.tsx

```
import { Button, Card, CardActions, CardContent, CardMedia, Typography } from
"@mui/material";
import { Link } from "react-router-dom";
import agent from "../../agent";
import { Medicine } from "../../models/medicine";
import { LoadingButton } from '@mui/lab';
import { useState } from "react";
import { useStoreContext } from "../../Context";

interface Props {
    medicine: Medicine;
}

export default function MedCard({ medicine }: Props) {

    const [loading, setLoading] = useState(false);
    const { setCart } = useStoreContext();

    function handleAddItem(medicineId: number) {
        setLoading(true);
        agent.Cart.addItem(medicineId)
            .then(cart => setCart(cart))
            .catch(error => console.log())
            .finally(() => setLoading(false));
    }

    return (
        <Card>
            <CardMedia
                sx={{ height: 180, backgroundSize: 'contain' }}
                image={medicine.image}
            />
            <CardContent sx={{ bgcolor: "#F1F1F1" }}>
                <Typography gutterBottom component="div">
                    <span style={{ fontSize: "25px", fontWeight: "bold"
}}>{medicine.name}</span> <span style={{ float: "right", fontSize: "20px"
}}>₹{medicine.price}</span>
                </Typography>
                <Typography variant="body2" color="text.secondary"
align="justify">
                    {medicine.category}
                </Typography>
            </CardContent>
            <CardActions sx={{ bgcolor: "#F1F1F1" }}>
                <Button component={Link} to={`/inventory/${medicine.id}`}
size="small">View</Button>
                <LoadingButton loading={loading} onClick={() =>
handleAddItem(medicine.id)} size="small">Add to cart</LoadingButton>
            </CardActions>
        </Card>
    )
}
```

features/inventory/MedicineDetails.tsx

```
import { Divider, Grid, Table, TableBody, TableCell, TableContainer, TableRow,
Typography } from "@mui/material";
import { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import agent from "../../agent";
import { useStoreContext } from "../../Context";
import { Medicine } from "../../models/medicine";

export default function MedicineDetails() {

    const {id} = useParams<{id: string}>();
    const [medicine, setMedicine] = useState<Medicine | null>(null);

    const {cart} = useStoreContext();
    const item = cart?.items.find(i => i.medicineId === medicine?.id);

    useEffect(() => {
        agent.Inventory.details(parseInt(id!))
            .then(response => setMedicine(response))
            .catch(error => console.log(error));
    }, [id])

    if (!medicine) return <h3>Product not found!</h3>

    return (
        <Grid container spacing={6}>
            <Grid item xs={6}>
                <img src={medicine.image} alt={medicine.name} style={{width:
'100%'}} />
            </Grid>
            <Grid item xs={6}>
                <Typography variant='h3'>{medicine.name}</Typography>
                <Divider sx={{mb: 2}} />
                <Typography variant='h4'
color='primary'>₹{medicine.price}</Typography>

                <TableContainer>
                    <Table>
                        <TableBody>
                            <TableRow>
                                <TableCell>Name</TableCell>
                                <TableCell>{medicine.name}</TableCell>
                            </TableRow>
                            <TableRow>
                                <TableCell>Description</TableCell>
                                <TableCell><Typography variant='inherit'
align='justify'>{medicine.description}</Typography></TableCell>
                            </TableRow>
                            <TableRow>
                                <TableCell>Category</TableCell>
                                <TableCell>{medicine.category}</TableCell>
                            </TableRow>
                            <TableRow>
                                <TableCell>Seller Information</TableCell>
                                <TableCell>{medicine.seller}</TableCell>
                            </TableRow>
                            <TableRow>
                                <TableCell>Quantity</TableCell>
                                <TableCell>{medicine.quantity}
Tablets</TableCell>
                            </TableRow>
                        </TableBody>
                    </Table>
                </TableContainer>
            </Grid>
        </Grid>
    )
}
```

features / cart

features/cart/CartPage.tsx

```
import { Add, Delete, Remove } from "@mui/icons-material";
import { Button, Divider, Grid, IconButton, Paper, Table, TableBody,
TableCell, TableContainer, TableHead, TableRow, Typography } from
"@mui/material";
import { Link } from "react-router-dom";
import agent from "../../agent";
import { useStoreContext } from "../../Context";

export default function BasketPage() {

    const { cart, setCart, removeItem } = useStoreContext();

    function addQuantity(medicineId: number) {
        agent.Cart.addItem(medicineId)
            .then(cart => setCart(cart))
            .catch(error => console.log(error));
    }

    function reduceQuantity(medicineId: number, quantity = 1) {
        agent.Cart.removeItem(medicineId, quantity)
            .then(() => removeItem(medicineId, quantity))
            .catch(error => console.log(error));
    }

    const total = cart?.items.reduce((sum, item) => sum + (item.quantity *
item.price), 0) ?? 0;

    if (!cart) return <Typography variant='h3'>Your cart is empty</Typography>

    return (
        <>
            <TableContainer component={Paper}>
```

```
                <Table sx={{ minWidth: 650 }}>
                    <TableHead>
                        <TableRow>
                            <TableCell align="center">Medicine</TableCell>
                            <TableCell align="center">Price</TableCell>
                            <TableCell align="center">Quantity</TableCell>
                            <TableCell align="center">Subtotal</TableCell>
                            <TableCell align="center"></TableCell>
                        </TableRow>
                    </TableHead>
                    <TableBody>
                        {cart.items.map(item => (
                            <TableRow
                                key={item.medicineId}
                                sx={{ '&:last-child td, &:last-child th': {
border: 0 } }}
                            >
                                <TableCell align="center" component="th"
scope="row">
                                    {item.name}
                                </TableCell>
                                <TableCell
align="center">₹{item.price}</TableCell>
                                <TableCell align="center">
                                    {item.quantity}
                                    <IconButton onClick={() =>
reduceQuantity(item.medicineId)} color='error'>
                                        <Remove />
                                    </IconButton>
                                    <IconButton onClick={() =>
addQuantity(item.medicineId)} sx={{ color: "green" }}>
                                        <Add />
                                    </IconButton>
                                </TableCell>
                                <TableCell align="center">₹{item.price *
item.quantity}</TableCell>
                                <TableCell align="left">
                                    <IconButton onClick={() =>
reduceQuantity(item.medicineId, item.quantity)} color='error'>
                                        <Delete />
                                    </IconButton>
                                </TableCell>
                            </TableRow>
                        ))}
                    </TableBody>
                </Table>
            </TableContainer>

            <Divider />

            <Grid container>
                <Grid item xs={6} />
                <Grid item xs={6}>
```

```
                    <TableContainer component={Paper} variant={'outlined'}>
                        <Table>
                            <TableBody>
                                <TableRow>
                                    <TableCell colSpan={2}>Total</TableCell>
                                    <TableCell
align="right">₹{total}</TableCell>
                                </TableRow>
                            </TableBody>
                        </Table>
                    </TableContainer>

                    <Divider />

                    <Button component={Link} to='/checkout'
variant='contained' size='large' fullWidth sx={{'&:hover': {color:
'#b39ddb'}}}>
                        Checkout
                    </Button>

                </Grid>
            </Grid>
        </>
    )
}
```

features / checkout

features/checkout/CheckoutPage.tsx

```
import { Divider, Typography } from "@mui/material";
import { useStoreContext } from "../../Context";

export default function CheckoutPage() {

    const { cart } = useStoreContext();
    const total = cart?.items.reduce((sum, item) => sum + (item.quantity *
item.price), 0) ?? 0;
    if (!cart) return <Typography variant='h3'>Your cart is empty</Typography>

    return (
        <div className="container mt-5 mb-5">
            <div className="row d-flex justify-content-center mt-1">
                <div className="col-md-8">
                    <div className="card">

                        <div className="invoice p-5">
                            <img alt="logo" style={{ display: "inline" }}
className="text-left logo p-2" src="https://thumbs.dreamstime.com/b/plus-
195775898.jpg" width="150" />
                            <h1 style={{ display: "inline" }}>Order
Confirmed!</h1>
```

```
                            <Divider sx={{ borderBottomWidth: 5, bgcolor:
"black" }} />

                            <div className="product border-bottom table-
responsive">
                                <table className="table table-borderless">
                                    <tbody>
                                        <tr>
                                            <th style={{ width: "10%" }}></th>
                                            <th style={{ width: "20%" }}>
                                                <h5>Medicines</h5>
                                            </th>
                                            <th style={{ width: "47%" }}>
                                            </th>
                                            <th style={{ width: "23%" }}>
                                                <h5>SubTotal</h5>
                                            </th>
                                        </tr>
                                    </tbody>
                                </table>
                                <table className="table table-borderless">
                                    <tbody>
                                        {cart.items.map(item => (
                                            <tr>
                                                <td width="20%">
                                                    <img alt={item.name}
src={item.image} width="90" />
                                                </td>
                                                <td width="60%">
                                                    <span className="font-
weight-bold"><b>{item.name}</b></span>
                                                    <div className="product-
qty">
                                                        <span className="d-
block">Quantity: {item.quantity}</span>
                                                        <span className="d-
block">Price: ₹{item.price}</span>
                                                    </div>
                                                </td>
                                                <td width="20%">
                                                    <div className="text-
right">
                                                        <span className="font-
weight-bold">₹{item.price * item.quantity}</span>
                                                    </div>
                                                </td>
                                            </tr>
                                        ))}
                                    </tbody>
                                </table>
                            </div>

                            <div className="row d-flex justify-content-end">
                                <div className="col-md-5">
```

```
                                    <table className="table table-borderless">
                                        <tbody className="totals">
                                            <tr>
                                                <td>
                                                    <div className="text-
left">
                                                        <span className="text-
muted">Subtotal</span>
                                                    </div>
                                                </td>
                                                <td>
                                                    <div className="text-
right">
                                                        <span>₹{total}</span>
                                                    </div>
                                                </td>
                                            </tr>
                                            <tr>
                                                <td>
                                                    <div className="text-
left">
                                                        <span className="text-
muted">Shipping Fee</span>
                                                    </div>
                                                </td>
                                                <td>
                                                    <div className="text-
right">
                                                        <span>₹50</span>
                                                    </div>
                                                </td>
                                            </tr>
                                            <tr className="border-top border-
bottom">
                                                <td>
                                                    <div className="text-
left">
                                                        <span className="font-
weight-bold">Total</span>
                                                    </div>
                                                </td>
                                                <td>
                                                    <div className="text-
right">
                                                        <span className="font-
weight-bold">₹{total + 50}</span>
                                                    </div>
                                                </td>
                                            </tr>
                                        </tbody>
                                    </table>
                                </div>
                            </div>
                            <p>You order has been confirmed and will be
shipped in next two days!</p>
                            <p className="font-weight-bold mb-0">Thanks for
shopping with us!</p>
                            <span><b>ABC Healthcare</b></span>

                        </div>
                    </div>
                </div>
            </div>
        </div>
    )
}
```

Testing Code

Features

Features/RegisterUser.feature

```
Feature: RegisterUser

A basic feature to test registration

@tag1
Scenario: Registration of User
    Given I navigate to website
    Then I enter Username Email and Password
    And I click on Register button
```

Features/UserLogin.feature

```
Feature: UserLogin

Logging a user into the website

@tag1
Scenario: Login user into the website
    Given I navigate to the website
    Then I click on Login button as I already have account
    Then I enter Username and Password
    And I click on Login
    Then I should see Homepage of website
```

Features/OrderConfirmation.feature

```
Feature: OrderConfirmation

Confirm one order

@tag1
Scenario: Comfirming an order
    Given I navigate to  website
    Then I click on Login button as I already have an account
    Then I enter an Username and Password
    When I click on Login
    Then I should see homepage of website
    Then I click on Inventory
    And I add some medicines to Cart
    When I click on Cart icon
    Then I should see Cart page
    When I click on Checkout
    Then Order should be confirmed
```

StepDefinitions

StepDefinitions/RegisterUserStepDefinitions.cs

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using System;
using TechTalk.SpecFlow;

namespace AbcHealthcareTesting.StepDefinitions
{
    [Binding]
    public class RegisterUserStepDefinitions
    {

        private String searchKeyword;
        private ChromeDriver chromeDriver;
        public RegisterUserStepDefinitions() => chromeDriver = new 
ChromeDriver("C:\\Users\\sahil.shaikh\\Downloads\\chromedriver_win32");


        [Given(@"I navigate to website")]
        public void GivenINavigateToWebsite()
        {
            chromeDriver.Navigate().GoToUrl("http://localhost:3000/");
            WebDriverWait wait = new WebDriverWait(chromeDriver, 
TimeSpan.FromMilliseconds(9500));
        }

        [Then(@"I enter Username Email and Password")]
        public void ThenIEnterUsernameEmailAndPassword()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver, 
TimeSpan.FromMilliseconds(9500));
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/div
[1]/div/input")).SendKeys("Rahul");
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/div
[2]/div/input")).SendKeys("rahul123@test.com");
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/div
[3]/div/input")).SendKeys("Rahul@12");
        }

        [Then(@"I click on Register button")]
        public void ThenIClickOnRegisterButton()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver, 
TimeSpan.FromMilliseconds(9500));
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/but
ton")).Click();
        }
    }
}
```

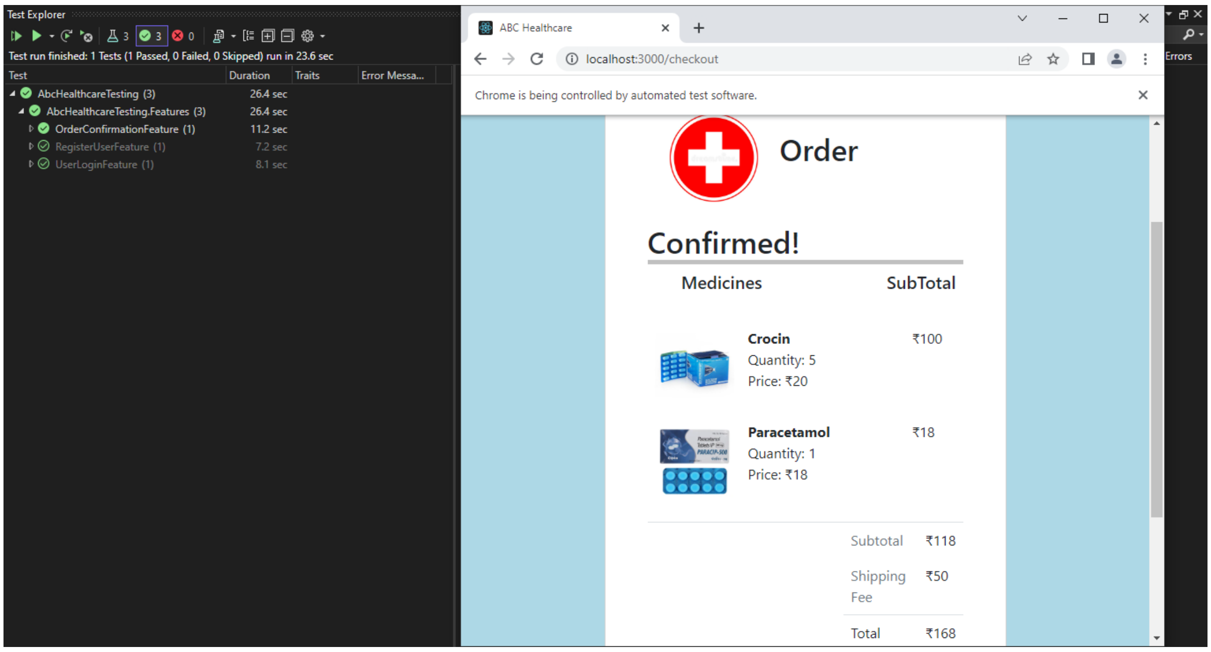
StepDefinitions/UserLoginStepDefinitions.cs

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using System;
using TechTalk.SpecFlow;

namespace AbcHealthcareTesting.StepDefinitions
{
    [Binding]
    public class UserLoginStepDefinitions
    {

        private String searchKeyword;
        private ChromeDriver chromeDriver;
        public UserLoginStepDefinitions() => chromeDriver = new
ChromeDriver("C:\\Users\\sahil.shaikh\\Downloads\\chromedriver_win32");


        [Given(@"I navigate to the website")]
        public void GivenINavigateToTheWebsite()
        {
            chromeDriver.Navigate().GoToUrl("http://localhost:3000/");
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
        }

        [Then(@"I click on Login button as I already have account")]
        public void ThenIClickOnLoginButtonAsIAlreadyHaveAccount()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/a")
).Click();
        }

        [Then(@"I enter Username and Password")]
        public void ThenIEnterUsernameAndPassword()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/div
[1]/div/input")).SendKeys("Rahul");
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/div
[2]/div/input")).SendKeys("Rahul@12");
        }

        [Then(@"I click on Login")]
        public void ThenIClickOnLogin()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
```

```
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/a[1
]")).Click();
        }

        [Then(@"I should see Homepage of website")]
        public void ThenIShouldSeeHomepageOfWebsite()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
        }
    }
}
```

StepDefinitions/OrderConfirmationStepDefinitions.cs

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using System;
using TechTalk.SpecFlow;

namespace AbcHealthcareTesting.StepDefinitions
{
    [Binding]
    public class OrderConfirmationStepDefinitions
    {

        private String searchKeyword;
        private ChromeDriver chromeDriver;
        public OrderConfirmationStepDefinitions() => chromeDriver = new
ChromeDriver("C:\\Users\\sahil.shaikh\\Downloads\\chromedriver_win32");


        [Given(@"I navigate to  website")]
        public void GivenINavigateToWebsite()
        {
            chromeDriver.Navigate().GoToUrl("http://localhost:3000/");
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
        }

        [Then(@"I click on Login button as I already have an account")]
        public void ThenIClickOnLoginButtonAsIAlreadyHaveAnAccount()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/a")
).Click();
        }

        [Then(@"I enter an Username and Password")]
        public void ThenIEnterAnUsernameAndPassword()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/div
[1]/div/input")).SendKeys("Rahul");
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/div
[2]/div/input")).SendKeys("Rahul@12");
        }

        [When(@"I click on Login")]
        public void WhenIClickOnLogin()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/form/a[1
]")).Click();
        }

        [Then(@"I should see homepage of website")]
        public void ThenIShouldSeeHomepageOfWebsite()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
        }

        [Then(@"I click on Inventory")]
        public void ThenIClickOnInventory()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
            chromeDriver.FindElement(By.XPath("/html/body/div/header/div/div[2
]/ul/a")).Click();
        }

        [Then(@"I add some medicines to Cart")]
        public void ThenIAddSomeMedicinesToCart()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
            chromeDriver.Manage().Timeouts().ImplicitWait =
TimeSpan.FromSeconds(10);
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/div[1]/d
iv/div[3]/button")).Click();
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/div[1]/d
iv/div[3]/button")).Click();
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/div[2]/d
iv/div[3]/button")).Click();
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/div[1]/d
iv/div[3]/button")).Click();
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/div[1]/d
iv/div[3]/button")).Click();
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div/div[1]/d
iv/div[3]/button")).Click();
        }
        [When(@"I click on Cart icon")]
        public void WhenIClickOnCartIcon()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
            chromeDriver.FindElement(By.XPath("/html/body/div/header/div/div[2
]/a")).Click();
        }

        [Then(@"I should see Cart page")]
        public void ThenIShouldSeeCartPage()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
        }

        [When(@"I click on Checkout")]
        public void WhenIClickOnCheckout()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
            chromeDriver.FindElement(By.XPath("/html/body/div/div/div[2]/div[2
]/a")).Click();
        }

        [Then(@"Order should be confirmed")]
        public void ThenOrderShouldBeConfirmed()
        {
            WebDriverWait wait = new WebDriverWait(chromeDriver,
TimeSpan.FromMilliseconds(9500));
        }
    }
}
```



```
Test Explorer
3  3  0
Test run finished: 1 Tests (1 Passed, 0 Failed, 0 Skipped) run in 23.6 sec
Test                                        Duration   Traits   Error Messa...
◢ AbcHealthcareTesting (3)                  26.4 sec
  ◢ AbcHealthcareTesting.Features (3)       26.4 sec
    ▷ OrderConfirmationFeature (1)          11.2 sec
    ▷ RegisterUserFeature (1)                7.2 sec
    ▷ UserLoginFeature (1)                   8.1 sec
```

ABC Healthcare

localhost:3000/checkout

Chrome is being controlled by automated test software.

Order

Confirmed!

Medicines	SubTotal
Crocin Quantity: 5 Price: ₹20	₹100
Paracetamol Quantity: 1 Price: ₹18	₹18

Subtotal	₹118
Shipping Fee	₹50
Total	₹168

<u>Fig. 1:</u> All testcases passed