HealthCare System SDK Documentation

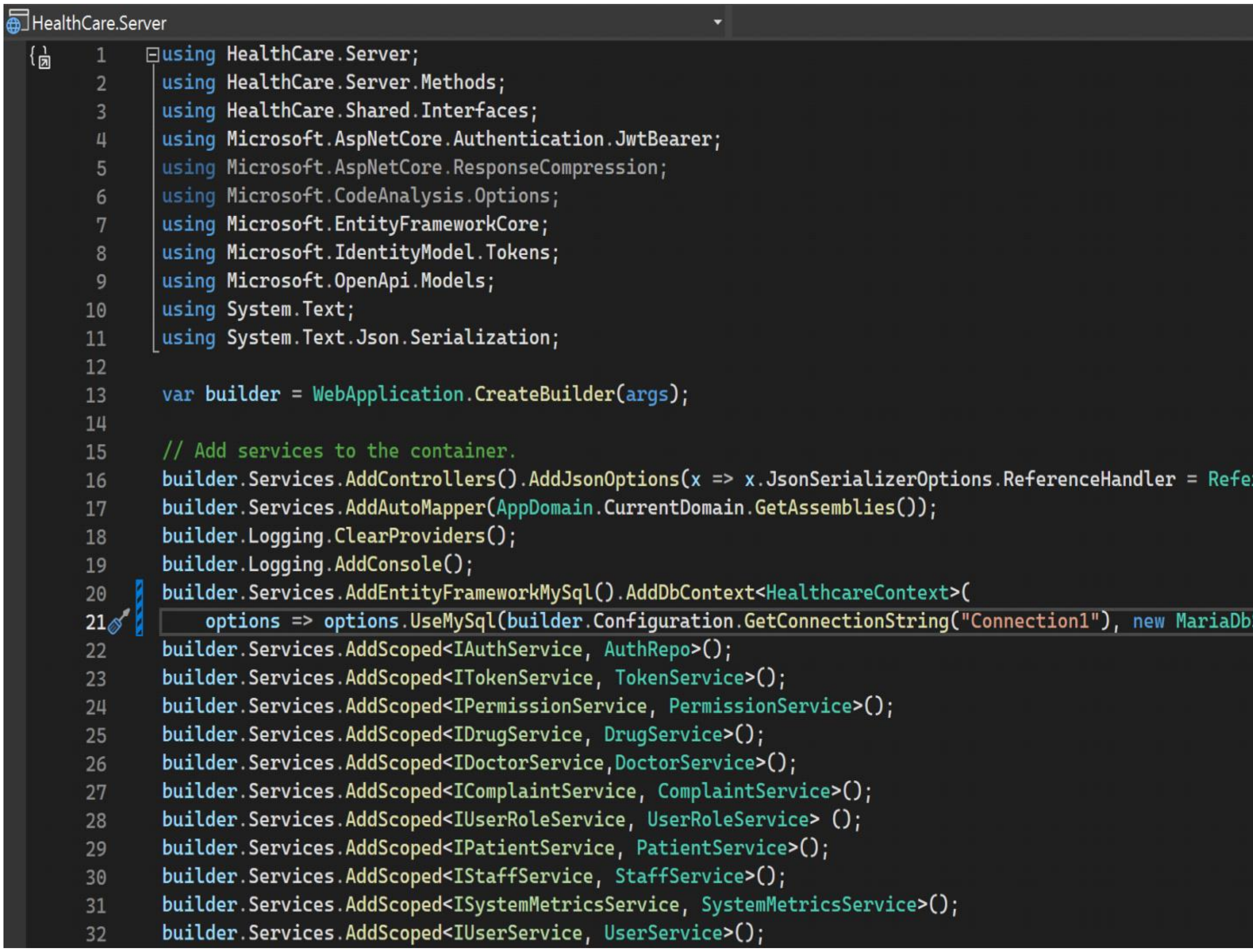
Server Application

The HealthCare.Server.csproj is a blazor .NET web project file that defines project-specific metadata and configuration options. It specifies the project's target framework, nullable settings, and references to required NuGet packages such as AutoMapper, BCrypt, CsvHelper, and Microsoft.EntityFrameworkCore. It also includes project references to two other project files: HealthCare.Client.csproj and HealthCare.Shared.csproj.

Program.cs

The Program.cs file is the entry point of the HealthCare Server Application. The file contains code that configures the application's services, HTTP request pipeline, and endpoints. This documentation will cover the major components of the Program.cs file and how they work.

Add services to the container



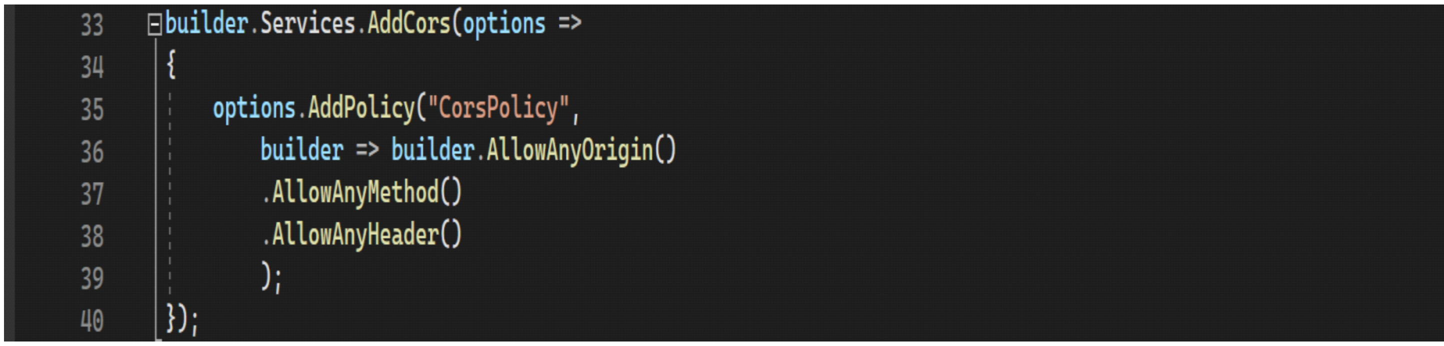
```
using HealthCare.Server;
using HealthCare.Server.Methods;
using HealthCare.Shared.Interfaces;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.ResponseCompression;
using Microsoft.CodeAnalysis.Options;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using System.Text;
using System.Text.Json.Serialization;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers().AddJsonOptions(x => x.JsonSerializerOptions.ReferenceHandler = Refe
builder.Services.AddAutoMapper(AppDomain.CurrentDomain.GetAssemblies());
builder.Logging.ClearProviders();
builder.Logging.AddConsole();
builder.Services.AddEntityFrameworkMySql().AddDbContext<HealthcareContext>(
    options => options.UseMySql(builder.Configuration.GetConnectionString("Connection1"), new MariaDb
builder.Services.AddScoped<IAuthService, AuthRepo>();
builder.Services.AddScoped<ITokenService, TokenService>();
builder.Services.AddScoped<IPermissionService, PermissionService>();
builder.Services.AddScoped<IDrugService, DrugService>();
builder.Services.AddScoped<IDoctorService,DoctorService>();
builder.Services.AddScoped<IComplaintService, ComplaintService>();
builder.Services.AddScoped<IUserRoleService, UserRoleService> ();
builder.Services.AddScoped<IPatientService, PatientService>();
builder.Services.AddScoped<IStaffService, StaffService>();
builder.Services.AddScoped<ISystemMetricsService, SystemMetricsService>();
builder.Services.AddScoped<IUserService, UserService>();
```

The Add services to the container code block adds services to the dependency injection container. The AddControllers method adds the controller services to the container. The AddJsonOptions method configures JSON serialization options for the controllers. The AddAutoMapper method configures AutoMapper for mapping objects. The AddDbContext method configures the HealthcareContext database context using the MySQL database provider. The AddScoped method adds scoped services to the container. Each service is resolved from the container when needed by the application.

Add Cors policy



```
builder.Services.AddCors(options =>
{
    options.AddPolicy("CorsPolicy",
        builder => builder.AllowAnyOrigin()
        .AllowAnyMethod()
        .AllowAnyHeader()
        );
});
```

The Add Cors policy code block adds a Cors policy to the application. The policy allows any origin, any method, and any header to access the application.

Configure Services

The `ConfigureServices` method is called by the runtime at startup to configure the services that the application will use. The following services are configured in this method:

- **AddControllers**: Adds controllers for handling HTTP requests to the service container. The **AddJsonOptions** method is called to configure the serialization settings for JSON responses to ignore cycles to prevent JSON serialization issues.
- **AddAutoMapper**: Configures AutoMapper, which is a mapping library used to map between objects.
- **AddEntityFrameworkMySql**: Adds support for MySQL with Entity Framework Core. The **AddDbContext** method is called to register the application's **HealthcareContext** with the dependency injection container.
- **AddScoped**: Registers scoped services with the dependency injection container. The following services are registered:
    - **IAuthService** with **AuthRepo**
    - **ITokenService** with **TokenService**
    - **IPermissionService** with **PermissionService**
    - **IDrugService** with **DrugService**

    - **IDoctorService** with **DoctorService**
    - **IComplaintService** with **ComplaintService**
    - **IUserRoleService** with **UserRoleService**
    - **IPatientService** with **PatientService**
    - **IStaffService** with **StaffService**
    - **ISystemMetricsService** with **SystemMetricsService**
    - **IUserService** with **UserService**
- **AddCors**: Adds CORS (Cross-Origin Resource Sharing) services to the application's service container. The **AddPolicy** method is called to configure the CORS policy that allows any origin, method, and header.
- **AddAuthentication**: Adds authentication services to the application's service container using JWT bearer authentication. The **RequireHttpsMetadata** property is set to **false** to allow authentication over HTTP, which is not recommended for production environments.
- **AddEndpointsApiExplorer**: Adds API explorer services to the application's service container. This enables the application to generate OpenAPI documentation automatically.
- **AddSwaggerGen**: Adds Swagger services to the application's service container. This enables the application to generate OpenAPI documentation.
- **AddControllersWithViews**: Adds controllers and views for handling HTTP requests to the service container.
- **AddRazorPages**: Adds Razor Pages for handling HTTP requests to the service container.

Configure

The **`Configure`** method is called by the runtime after the **`ConfigureServices`** method to configure the HTTP request pipeline. The following middleware components are configured in this method:

- **UseWebAssemblyDebugging** (in development environment only): Adds support for debugging Blazor WebAssembly applications in the browser.
- **UseSwagger**: Serves the Swagger JSON endpoint to enable Swagger UI to display API documentation.
- **UseSwaggerUI**: Serves the Swagger UI that provides the user interface for exploring and testing the API.
- **UseExceptionHandler**: Configures the middleware to handle exceptions. In case of an exception, the middleware returns an error response.
- **UseHsts**: Adds HTTP Strict Transport Security (HSTS) headers to responses to instruct browsers to only use HTTPS for future requests.

- **UseHttpsRedirection**: Redirects all HTTP requests to HTTPS.
- **UseBlazorFrameworkFiles**: Serves the static files required by the Blazor WebAssembly client application.
- **UseStaticFiles**: Serves static files for the web application.
- **UseRouting**: Adds endpoint routing middleware to the pipeline.
- **UseCors**: Enables CORS for the application.
- **UseAuthentication**: Adds authentication middleware to the pipeline.
- **UseAuthorization**: Adds authorization middleware to the pipeline.
- **MapRazorPages**: Maps Razor Pages requests to the appropriate Razor Page.

The code then defines the startup configuration for the application using the builder object. The builder object creates a new instance of the web application and configures it with a set of services and middleware components.

First, the builder adds controllers to the container using the **`AddControllers()`** method. It also configures the JSON serializer to ignore circular references using the **`AddJsonOptions()`** method.

```
builder.Services.AddControllers().AddJsonOptions(x =>
x.JsonSerializerOptions.ReferenceHandler = ReferenceHandler.IgnoreCycles);
```

Next, the builder configures AutoMapper using the AddAutoMapper() method, and clears the default logging providers using the Logging.ClearProviders() method. It then adds a console logging provider using the Logging.AddConsole() method.

HealthcareContext.cs
The HealthcareContext.cs file contains the implementation of the DbContext class, which represents a session with the database and provides a way to query and save data. Here's a summary of the file:

- **HealthcareContext class:** This is the main class of the file and it derives from the DbContext class. It represents the database context and provides access to the entities and relationships in the database.

- **OnModelCreating method:** This method is used to configure the model that was discovered by convention from the entity types exposed in HealthcareContext. It overrides the base implementation to configure the relationships, indexes, and constraints for the entities.

- **DbSet properties**: The HealthcareContext class defines properties for each entity set in the database. These properties are of type DbSet<TEntity> and are used to query and save instances of the corresponding entity type. Each property represents a table in the database and provides a way to access the rows in that table.

- **Entity classes:** The file also contains several classes that represent the entities in the database. These classes are decorated with the Table attribute to specify the name of the table in the database. Each class defines properties for the columns in the corresponding table.

API Documentation

POST /api/Tokens/authenticate

Authenticates a user and generates an access token.

Request

- Method: **POST**
- Path: **/api/Tokens/authenticate**
- Headers:
- **Content-Type**: **application/json**
- Request body:

Parameter	Type	Required	Description
Username	string	Yes	The user's username.
Password	string	Yes	The user's password.

Response

- Status code: **200 OK**

Response body: A JSON object representing the user's account details, including the access token.
Example:

```
{
  "UserId": 1,
  "Username": "johndoe",
  "RoleId": 2,
  "Role": {
    "RoleId": 2,
    "RoleName": "Doctor",
    "Description": "A doctor user with access to patient records."
  },
  "LastLogin": "2022-01-01T12:00:00Z",
  "AccessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG
9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c"
}
```

- Status code: **400 Bad Request**

- Response body: An error message indicating that the credentials are invalid.

  Example:
  "Invalid credentials"

POST /api/Tokens/register

Creates a new user account.

Request

- Method: **POST**
- Path: **/api/Tokens/register**
- Headers:
- **Content-Type**: **application/json**
- **Authorization**: **Bearer [access token]**
- Request body:

Parameter	Type	Required	Description
Username	string	Yes	The user's username.
Password	string	Yes	The user's password.

Response

- Status code: **200 OK**
- Response body: A message indicating that the user account has been created successfully.
  Example:
  User created successfully
- Status code: **400 Bad Request**
- Response body: An error message indicating that the user account could not be created.
  Example:
  Invalid credentials
- Status code: **400 Bad Request**
- Response body: An error message indicating that a user with the same username already exists. Example:
  User with the same username already exists

- Status code: **401 Unauthorized**
- Response body: An error message indicating that the access token is invalid or has expired.
  Example:
  Invalid access token

Analysis
GET /api/AnalysisController/performance/{a_start}/{a_end}
Endpoint to return analysis on actions by specified doctor within a specified period

Request Parameters

Parameter Name	**Type**	**Required**	**Description**
a_start	**DateTime**	Yes	The start of the specified period for the analysis
a_end	**DateTime**	Yes	The end of the specified period for the analysis

Request Headers

Header	**Type**	**Required**	**Description**
Authorization	**string**	Yes	The authentication token for the user

Response

Status Code	**Response Model**	**Description**
200	**DoctorAnalysis**	The doctor analysis response model containing the analysis data for the specified period
400	**string**	The error message returned when the request is invalid or unauthorized

Response Models

Field Name	Type	Description
AttendanceObjects	**IEnumerable<AttendanceObject>**	The collection of attendance objects for the specified doctor and period
TopPrescribedDrugs	**IEnumerable<PrescribedDrugObject>**	The collection of top prescribed drugs for the specified doctor and period
TopCases	**IEnumerable<FrequentCasesObject>**	The collection of top frequent cases for the specified doctor and period
TotalCases	**int**	The total number of cases for the specified doctor and period

DrugController API Endpoints

POST api/Drug/create
Creates a new drug.
Request Parameters

- **DrugModel**: The object containing the drug details to create.

Response
Returns a string message of the result of the operation.

PUT api/Drug/decomission
Decommissions an existing drug.
Request Parameters

- **string**: The id of the drug to decommission.

Response
Returns a string message of the result of the operation.

PUT api/Drug/reinstate
Reinstates a decommissioned drug.
Request Parameters

- **string**: The id of the drug to reinstate.

Response
Returns a string message of the result of the operation.

PUT api/Drug/markrefill
Marks a drug as needing a refill.
Request Parameters

- **string**: The id of the drug to mark for refill.

Response
Returns a string message of the result of the operation.

PUT api/Drug/restock
Restocks a drug.
Request Parameters

- **StockItem**: The object containing the stock details to restock the drug.

Response
Returns a string message of the result of the operation.

PUT api/Drug/setExpired
Sets a drug as expired.
Request Parameters

- **Expiryitem**: The object containing the drug details and the expiry date to set the drug as expired.

Response
Returns a string message of the result of the operation.

GET api/Drug/list
Gets a list of all drugs.
Response
Returns a list of Drug objects.

PUT api/Drug/update
Updates the details of an existing drug.
Request Parameters

- **DrugModel**: The object containing the updated drug details.

Response
Returns a string message of the result of the operation.

Patient Controller
Endpoint: GET /api/Patient/records/{a_start}/{a_end}
Description: This endpoint returns a list of attendance records within the specified date range.

Authorization: This endpoint requires authentication.

Parameters:

a_start (required): The start date of the range to search for attendance records. Must be in ISO 8601 format (e.g. 2023-04-24T00:00:00.000Z).
a_end (required): The end date of the range to search for attendance records. Must be in ISO 8601 format (e.g. 2023-04-24T23:59:59.999Z).

Response:
200 OK: Returns a list of AttendanceObject objects representing the attendance records within the specified date range.
400 Bad Request: Returns an error message if the authentication fails or if the request parameters are invalid.

Endpoint: GET /api/Patient/records/history/{a_id}
Description: This endpoint returns the medical history of a patient with the specified ID.

Authorization: This endpoint requires authentication.

Parameters:
a_id (required): The ID of the patient to retrieve the medical history for.

Response:
200 OK: Returns a list of MedHistory objects representing the medical history of the specified patient.
400 Bad Request: Returns an error message if the authentication fails or if the request parameters are invalid.

Endpoint: GET /api/Patient/records/profile/{a_id}
Description: This endpoint returns the profile of a patient with the specified ID.

Authorization: This endpoint requires authentication.

Parameters:
a_id (required): The ID of the patient to retrieve the profile for.

Response:
200 OK: Returns a list of Patient objects representing the profile of the specified patient.
400 Bad Request: Returns an error message if the authentication fails or if the request parameters are invalid.

Endpoint: GET /api/Patient/records/patientConsults
Description: This endpoint returns a list of user consultations for the currently authenticated user.

Authorization: This endpoint requires authentication.

Response:
200 OK: Returns a list of UserConsults objects representing the user consultations for the currently authenticated user.
400 Bad Request: Returns an error message if the authentication fails or if the request parameters are invalid.

Endpoint: GET /api/Patient/records/profile/list/{a_id}
Description: This endpoint returns a list of patients matching the specified search criteria.

Authorization: This endpoint requires authentication.

Parameters:
a_id (required): The search criteria to use when looking up patients.

Response:
200 OK: Returns a list of Patient objects representing the patients that match the search criteria.
400 Bad Request: Returns an error message if the authentication fails or if the request parameters are invalid.