PE\_PRN231\_FA24\_TrialTest  
FALL 2024  
Subject: PRN231  
Duration: 90 minutes

**INSTRUCTIONS**

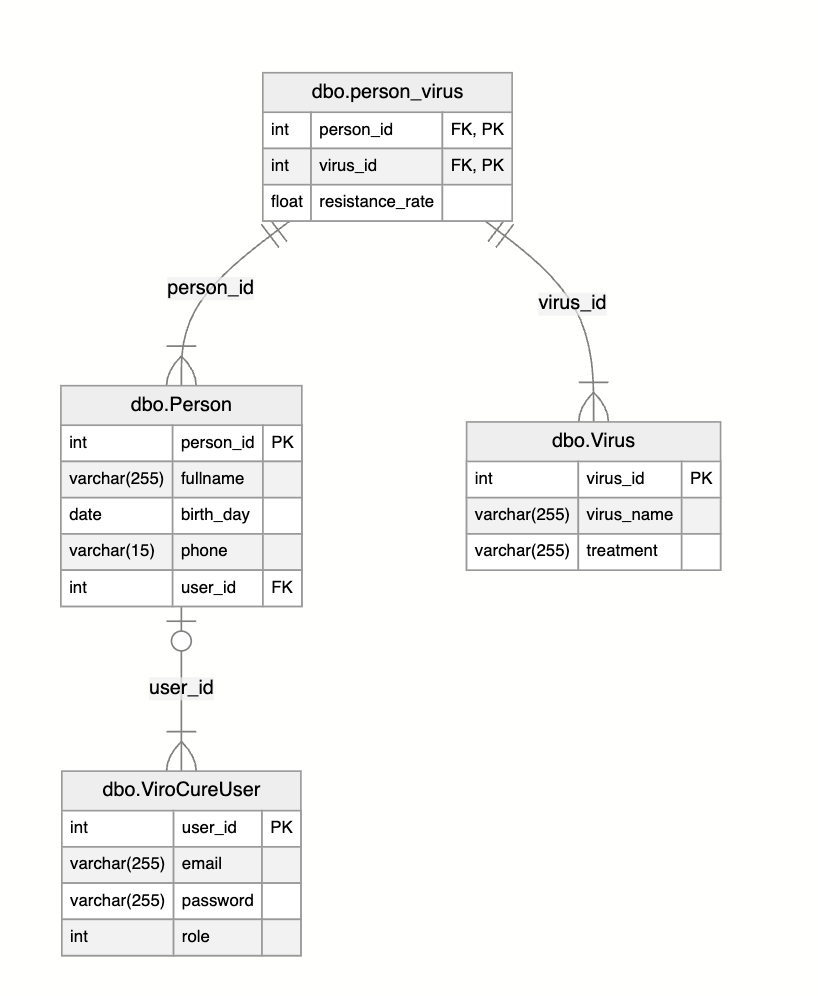
**Please read the instructions carefully before doing the questions.**

* You are **NOT allowed** to use any device to share data with others.
* You must use **Visual Studio 2019 or above, MSSQL Server 2012 or above** for your development tools.

**IMPORTANT – before you start doing your solution, MUST do the following steps:**

1. Create Solution/Project in Visual Studio named **PE\_PRN231\_FA24\_TrialTest\_StudentFullname\_BE** for API, and **PE\_PRN231\_FA24\_TrialTest\_StudentCode\_FE** for Client Application. Set the default Client application for your project as **Login** page**.**
2. Create your MS SQL database named **ViroCureFAL2024DB** by running code in script **ViroCureFAL2024DB.sql.**
3. *You are not allowed to connect directly to a database from ASP.NET Core Web API, every database connection must be used with Repository and Data Access Objects.*
4. *The database connection string must get from the appsettings.json file.* ***In the case your code connects direct to the database from ASP.NET Core Web API or hard coded the connection string, you will get 0 point.***
5. *Please check before you submit your work, no syntax error(s) in the submitted PE.* ***If there are syntax errors or compilation errors in your PE program, you will not pass the PE requirements, the point will be 0.***
6. ***Your work will be considered invalid (0 point) if your code inserts stuff that is unrelated to the test.***

Implement a part of **ViroCure** Application. The below Figure is a part of **ViroCureFAL2024DB** database.



Note that: *Member role: Administrator = 1; Patients= 2; Doctor=3 .* A patient can have many types of viruses. Each virus will have different treatment drugs and drug resistance rates in each patient's body.

**Part 01 (7.0 points)**. Create an API using **ASP.NET Core Web API** with Entity Framework Core. A MS SQL Server database will be created to persist the data and it will be used for reading and managing data.

1. Check authentication/authorization with the ASP.NET Core Web API with JSON Web Token (JWT)

* ***Doctor role will have a permission to all actions (CRUD and search).***
* ***Administrator role will have only permission to search information.***
* ***Patients role have the right to view their information.***

*Note that the permission for Authentication feature is using UserEmail and UserPassword.*

1. You must use RESTful API to implement the ASP.NET Core Web API. CORS is using in this case.

**Endpoint:** POST /api/login

**HTTP Method:** POST

**Function:** Authenticates the user using their email and password. On successful authentication, the API returns a token (JWT or session token) that can be used for further authenticated requests.

**Request Body (JSON):**

|  |
| --- |
| {  "email": "user@example.com",  "password": "yourpassword"  } |

**Validations:**

**Email:** Must be a valid email address format.

**Password**: Should meet security requirements (e.g., minimum length, complexity), but this check is typically handled during user registration.

**Success Response:**

**Response Code**: 200 OK

**Response Body** (JSON):

|  |
| --- |
| {  "message": "Login successful",  "token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c",  "user": {  "id": 1,  "email": "user@example.com",  "role": "admin"  }  } |

**Error Response:**

**Response Code**: 401 Unauthorized (for incorrect email/password)

**Response Body** (JSON):

{ "error": "Invalid email or password" }

**Create (Add a person and the viruses they are infected with)**

**Endpoint:** POST /api/person

**HTTP Method**: POST

**Function**: Adds a new person and, if applicable, the viruses they are infected with.

**Request Body** (JSON):

|  |
| --- |
| {  “personID”: 1,  "fullName": "John Doe",  "birthDay": "1990-05-15",  "phone": "1234567890",  "viruses": [  {  "virusName": "COVID-19",  "resistanceRate": 0.2  },  {  "virusName": "Influenza",  "resistanceRate": 0.0  }  ]  } |

**Response**: 201 Created

|  |
| --- |
| {  "personId": 1,  "message": "Person and viruses added successfully"  } |

**Read (Retrieve person details and associated viruses)**

**Endpoint**: GET /api/person/{id}

**HTTP Method:** GET

**Function**: Retrieves details of a person, including any viruses they are infected with.

**URL Parameters:**

id (ID of the person to retrieve)

**Response:** 200 OK

|  |
| --- |
| {  "personId": 1,  "fullName": "John Doe",  "birthDay": "1990-05-15",  "phone": "1234567890",  "viruses":  [  {  "virusName": "COVID-19",  "resistanceRate": 0.2  },  {  "virusName": "Influenza",  "resistanceRate": 0.0  }  ]  } |

**Endpoint (Retrieve all persons and their viruses):** GET /api/persons

**Response**: 200 OK

|  |
| --- |
| [  {  "personId": 1,  "fullName": "John Doe",  "birthDay": "1990-05-15",  "phone": "1234567890",  "viruses":  [  {  "virusName": "COVID-19",  "resistanceRate": 0.2  }  ]  },  {  "personId": 2,  "fullName": "Jane Smith",  "birthDay": "1985-10-22",  "phone": "0987654321",  "viruses": []  }  ] |

**Update (Update person details and their viruses)**

**Endpoint**: PUT /api/person/{id}

**HTTP Method**: PUT

**Function**: Updates the details of a person, including their associated viruses.

**URL Parameters:**

id (ID of the person to update)

**Request Body** (JSON):

|  |
| --- |
| {  "fullName": "Jonathan Doe",  "birthDay": "1990-05-15",  "phone": "1234567890",  "viruses": [  {  "virusName": "COVID-19",  "resistanceRate": 0.5  },  {  "virusName": "Influenza",  "resistanceRate": 0.1  }  ]  } |

**Response**: 200 OK

|  |
| --- |
| { "message": "Person and viruses updated successfully" } |

**Delete (Delete a person and their associated viruses)**

**Endpoint:** DELETE /api/person/{id}

**HTTP Method:** DELETE

**Function:** Deletes a person and their relationship with any viruses they are infected with.

**URL Parameters:**

id (ID of the person to delete)

**Response:** 200 OK

{ "message": "Person and related viruses deleted successfully" }

* Implement CRUD actions with **Person** table. Add new item with the r*equirements:*
  + All fields are required (**Add a person and the viruses they are infected with)**
  + Value for Fullname includes a-z, A-Z, space, @, # and digit 0-9. Each word of the Fullname must begin with the capital letter.
  + Value for Birthday < 01-01-2007.
  + Resistance Rate: Must be between 0 and 1.
  + Phone Number: Should match a specific pattern, e.g., +84989xxxxxx (international format). The exact format can vary, but it should ensure the phone number is valid.

**Invalid cases**

**Response Code**: 400 Bad Request

**Response Body ex:**

{ "error": "Phone number must be in the format +84989xxxxxx" }

{ "error": "Each word of the Fullname must begin with the capital letter" }

{ "error": "Value for Birthday < 01-01-2007" }

{ "error": "Resistance Rate: Must be between 0 and 1" }

**Part 02 (3.0 points).**

Student can choose one of these options for client application

* *ASP.NET Core Web App with MVC*
* *ASP.NET Core Razor Pages*
* *Windows Forms*
* *Windows Presentation Foundation*

1. Authentication function

If user with a *Doctor role* *or Patients role* logs in successfully, save this information to a temporary parameter.

* *All CRUD actions are required with Doctor role*
* *Search information is required with Administrator**role or Doctor role.*
* *Patients role have the right to view their information.*

In the case login unsuccessfully, display *“You are not allowed to access this function!”*

1. Check if login successfully, search all items with 2 conditions: Achivements or Nominatio (using relative search)
2. Check if login successfully, delete the selected item with the confirmation then update the list of items.
3. Check if login successfully, add new item with the r*equirements:*
   * *The new item will display at the top of the list.*
   * All fields are required (**Add a person and the viruses they are infected with)**
   * Value for Fullname includes a-z, A-Z, space, @, # and digit 0-9. Each word of the Fullname must begin with the capital letter.
   * Value for Birthday < 01-01-2007.
   * Resistance Rate: Must be between 0 and 1.
   * Phone Number: Should match a specific pattern, e.g., +84989xxxxxx (international format). The exact format can vary, but it should ensure the phone number is valid.
4. Check if login successfully, update existing item with the same r*equirements* with add function.