Blood Bank Management REST API Assignment Description

Assignment Overview

This assignment involves creating a RESTful API for a Blood Bank Management system using **C#** and **ASP.NET Core**. You will use a single model for all data interactions and perform operations using an in-memory list as the database. The project should include CRUD operations, pagination, and search functionality. You'll demonstrate your API through **Swagger** and **Postman** screenshots, and update your project on **GitHub**.

Project Requirements

You are required to create an API with **one model** and **one controller** that manages all operations for the Blood Bank system. The API should cover CRUD operations, support pagination, and include search functionalities.

Model Description

Use a single model named BloodBankEntry with the following attributes:

- **Id**: A unique identifier for the entry (auto-generated).
- **DonorName**: Name of the donor.
- **Age**: Donor's age.
- **BloodType**: Blood group of the donor (e.g., A+, O-, B+).
- **ContactInfo**: Contact details (phone number or email).
- **Quantity**: Quantity of blood donated (in ml).
- **CollectionDate**: Date when the blood was collected.
- **ExpirationDate**: Expiration date for the blood unit.
- **Status**: Status of the blood entry (e.g., "Available", "Requested", "Expired").

API Requirements

Develop the following features using a single controller:

1. **CRUD Operations**
   - **Create (POST /api/bloodbank)**: Add a new entry to the blood bank list. The input should include donor details, blood type, quantity, and collection/expiration dates.
   - **Read (GET /api/bloodbank)**: Retrieve all entries in the blood bank list.
   - **Read (GET /api/bloodbank/{id})**: Retrieve a specific blood entry by its Id.
   - **Update (PUT /api/bloodbank/{id})**: Update an existing entry (e.g., modify donor details or update blood status).
   - **Delete (DELETE /api/bloodbank/{id})**: Remove an entry from the list based on its Id.
2. **Pagination**
   - **GET /api/bloodbank?page={pageNumber}&size={pageSize}**: Retrieve a paginated list of blood bank entries. The response should show entries based on page number and page size parameters.
3. **Search Functionality**
   - **GET /api/bloodbank/search?bloodType={bloodType}**: Search for blood bank entries based on blood type.
   - **GET /api/bloodbank/search?status={status}**: Search for blood bank entries by status (e.g., "Available", "Requested").
   - **GET /api/bloodbank/search?donorName={donorName}**: Search for donors by name.

Implementation Instructions

1. **Setup**:
    - Create an ASP.NET Core project.
    - Use a single BloodBankEntry model and a single controller (BloodBankController).
    - Store data in an in-memory list (List<BloodBankEntry>).
    - Implement CRUD operations, pagination, and search in the controller.
2. **Swagger Integration**:
    - Add **Swagger** support to your project for easy API documentation and testing.
    - Ensure all endpoints are well-documented in Swagger.
3. **Testing**:
    - Use **Postman** to manually test each API endpoint.
    - Capture screenshots of both **Swagger** and **Postman** interactions showing successful API calls and responses.
4. **Code Quality**:
    - Write clean, maintainable code with appropriate comments.
    - Ensure proper error handling (e.g., return 404 for non-existent entries, 400 for bad input).
5. **GitHub**:
    - Push your entire project to a **public GitHub repository**.
    - Include a README.md file explaining how to run the project, test the endpoints, and a summary of the API.

Deliverables

- **Swagger Screenshots**: Provide screenshots of Swagger UI showcasing all the available endpoints with successful requests and responses.
- **Postman Screenshots**: Include screenshots of API testing done via Postman for each endpoint.
- **GitHub**: Upload your project to a GitHub repository and include the following:
    - README.md with project overview, setup instructions, and endpoint descriptions.
    - Sample JSON payloads used for testing the API in Swagger and Postman.
    - Screenshots folder containing Swagger and Postman screenshots.

Bonus

- **Sorting**: Implement sorting capabilities for GET /api/bloodbank (e.g., sort by BloodType or CollectionDate).
- **Filtering**: Allow multiple search parameters to be used simultaneously (e.g., bloodType and status).

Evaluation Criteria

- **API Functionality**: Proper implementation of CRUD operations, pagination, and search.
- **Code Quality**: Clean and structured code with good practices.
- **Swagger & Postman Testing**: Demonstrated by screenshots showcasing the working API.
- **Documentation**: Clear README.md and detailed API endpoint descriptions.
- **GitHub Usage**: Proper usage of GitHub for code management and documentation.

This assignment will help you understand how to develop, test, and document REST APIs efficiently using **C# and ASP.NET Core**. Good luck!