# Exam Task for Full-stack Developer

## Part 1: Database

1. Create a database based on SQLite with two tables: "Users" and "InsurancePolicies".
2. The "Users" table should have the following fields:
   * ID (unique user identifier)
   * Name (string)
   * Email (string)
3. The "InsurancePolicies" table should have the following fields:
   * ID (unique policy identifier)
   * PolicyNumber (string)
   * InsuranceAmount (number)
   * StartDate (date)
   * EndDate (date)
   * UserID (reference to the "Users" table)

## Part 2: Web API

1. Create an ASP.NET Core Web API project.
2. Include the following classes for database interaction and request handling:
   * User (user model) with fields ID, Name, Email.
   * InsurancePolicy (insurance policy model) with fields ID, Policy Number, Insurance Amount, Start Date, End Date, User ID.
   * AppDbContext (database context) for interacting with the SQLite database, including DbSet for the User and InsurancePolicymodels.
   * IUserRepository and IInsurancePolicyRepository interfaces with methods for CRUD operations on the respective models.
   * UserRepository and InsurancePolicyRepository implementations of the IUserRepository and IInsurancePolicyRepository interfaces, using AppDbContext to execute database queries.
   * API controllers (UsersController and InsurancePoliciesController) for handling HTTP requests and invoking repository methods.
3. Use dependency injection to link controllers and repositories.
4. Implement the following HTTP methods in the controllers to handle requests:
   * GET methods to retrieve a list of all users and insurance policies, as well as a specific user and their insurance policies by ID.
   * POST methods to create a new user and insurance policy.
   * PUT methods to update user and insurance policy information by ID.
   * DELETE methods to delete a user and insurance policy by ID.
5. Use Entity Framework Core for database interaction.

**Part 3: Client-side in Angular**

1. Create an Angular application to interact with the Web API.
2. Implement the following components:
   * UserListComponent: A component to display a list of users. Includes:
     + List of users with the ability to select a specific user to view their insurance policies.
     + Buttons to add a new user and update the list of users.
     + Functionality to edit and delete users.
   * UserDetailsComponent: A component to display information about a selected user and their insurance policies. Includes:
     + User information (name, email) and their insurance policies (policy number, insurance amount, start and end dates).
     + Buttons to add a new insurance policy, edit, and delete user's insurance policies.
     + Functionality to filter insurance policies by start date.
   * InsurancePolicyFormComponent: A component for creating and editing insurance policies. Includes:
     + Form to input insurance policy details (policy number, insurance amount, start and end dates).
     + Form data validation.
     + Buttons to save changes and cancel the operation.
3. Use services to interact with the Web API from components.
4. Implement routing for navigation between components (e.g., user list and user details).
5. Style components to enhance the user interface.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note:

* Additional classes, components, and functionality can be added as needed.
* Evaluation will be based on the completeness of implementation, code quality, use of Angular patterns, and adherence to the task requirements.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Please let me know if you have any further questions or requests regarding this task.