Cymulate Full Stack Engineer

Task Description:
You are required to build a phishing simulation and awareness web application using the following technologies:

- **Backend:** .NET server for handling the phishing simulation and NestJS server for Phishing attempts management.
- **Frontend:** React for the web application.
- **DevOps (Bonus):** Docker for containerization.
- **Database:** MongoDB for storing user information and phishing attempts.

Detailed Requirements:

1. **Phishing Simulation (.NET server):**
    - Build a **.NET** server to manage phishing simulations.
    - Create a **POST /phishing/send** endpoint to send phishing emails to a provided email address.
    - Use **any** email library to send the phishing emails.
    - The phishing email should contain a link that, when clicked, will notify your server that the user has clicked the phishing test.
    - Update the phishing attempt status in the DB when the link is clicked.
2. **Phishing attempts Management (NestJS server):**
    - Implement user registration and login functionality using **JWT** for authentication.
    - Craete route that will retrieve all the phishing attempts (which will be displayed in the client)
    - Create route that will get a phising attempt to sent by the client. This route should communicate with the Phishing simulation server for sending email
3. **Frontend with React:**
    - Develop a React web application to interface with both backends.
    - **Login and Registration Page**: Build pages to allow admin users to register and log in using the JWT-based authentication.
    - **Phishing Simulation Page**:
        - Allow users to input an email address and trigger a phishing attempt via the frontend.
        - Display a table of all phishing attempts, including the recipient email, the email content, and the status.
        - The attempt status should be updated and synced in realtime. You can choose any approproate solution.
    - The frontend should communicate with the phishing attempts management server.
4. **Docker and DevOps (Bonus):**
    - Create Dockerfiles for both servers and the React frontend.
    - Ensure that the application can be deployed using Docker Compose.

    - Document the steps to set up and run the application using Docker Compose.

Notes:

- Docker solution is a Bonus.
- Use TypeScript and NOT JavaScript.
- Use NestJS and not Express.

Submission:

- A GitHub repository with the complete source code.
- A README file with instructions on how to build and run the application using Docker Compose.
- Any additional documentation or notes that will help in understanding your approach and implementation.