Sky World Limited - Software Engineering Pre-Interview Task

The task will involve creating a simple Survey application. The application will allow users to respond to survey questions and view a list of all responses to the survey questions.

The task will be split into 3 components:

1. Databases

Using a Relational Database Management System, either **MySQL** , **Oracle** , **Microsoft SQL Server** or **Postgres**:

1. Design an **Entity Relationship Diagram (ERD)** for the database for the application.
2. Implement the database, name it **sky_survey_db**.

2. REST API

Using your preferred language, create a **REST API** that connects to your database.

The API should have the following endpoints:

1. To fetch list of questions
    - url: `/api/questions`
    - method: **GET**
    - response:

```
<!-- Questions List Response -->
<questions>
    <question name="full_name" type="short_text" required="yes">
        <text>What is your full name?</text>
        <description>[Surname] [First Name] [Other Names]</description>
    </question>
    <question name="email_address" type="email" required="yes">
        <text>What is your email address?</text>
        <description/>
    </question>
    <question name="description" type="long_text" required="yes">
        <text>Tell us a bit more about yourself</text>
        <description/>
    </question>
    <question name="gender" type="choice" required="yes">
        <text>What is your gender?</text>
        <description/>
        <options multiple="no">
            <option value="MALE">Male</option>
            <option value="FEMALE">Female</option>
            <option value="OTHER">Other</option>
        </options>
    </question>
    <question name="programming_stack" type="choice" required="yes">
        <text>What programming stack are you familiar with?</text>
        <description>You can select multiple</description>
        <options multiple="yes">
            <option value="REACT">React JS</option>
            <option value="ANGULAR">Angular JS</option>
            <option value="VUE">Vue JS</option>
            <option value="SQL">SQL</option>
            <option value="POSTGRES">Postgres</option>
            <option value="MYSQL">MySQL</option>
            <option value="MSSQL">Microsoft SQL Server</option>
            <option value="Java">Java</option>
            <option value="PHP">PHP</option>
            <option value="GO">Go</option>
            <option value="RUST">Rust</option>
        </options>
    </question>
    <question name="certificates" type="file" required="yes">
        <text>Upload any of your certificates?</text>
        <description>You can upload multiple (.pdf)</description>
        <file_properties format=".pdf" max_file_size="1" max_file_size_unit="mb" multiple="yes"/>
    </question>
</questions>
```

2. To submit responses to the questions
    - url: `/api/questions/responses`
    - method: **PUT**

- response:

```
<question_response>
    <full_name>Jane Doe</full_name>
    <email_address>janedoe@gmail.com</email_address>
    <description>I am an experienced FrontEnd Engineer with over 6 years experience.</description>
    <gender>MALE</gender>
    <programming_stack>REACT,VUE</programming_stack>
    <certificates>
        <certificate>Adobe Certification 19-08-2023.pdf</certificate>
        <certificate>Figma Fundamentals 19-08-2023.pdf</certificate>
    </certificates>
    <date_responded>2023-09-23 12:30:12</date_responded>
</question_response>
```

The API should support uploading of files through use of **form-data**.

3. To fetch submitted responses to the questions
    - url: `/api/questions/responses`
    - method: **GET**
    - response:

```
<question_responses current_page="1" last_page="1" page_size="10" total_count="2">
    <question_response>
        <response_id>1</response_id>
        <full_name>John Doe</full_name>
        <email_address>johndoe@gmail.com</email_address>
        <description>I am an experienced FullStack Engineer with over 2 years experience.</description>
        <gender>MALE</gender>
        <programming_stack>REACT,JAVA,SQL,POSTGRES</programming_stack>
        <certificates>
            <certificate id="1">Oracle Java Certification 19-08-2023.pdf</certificate>
            <certificate id="2">Oracle SQL Certification 19-08-2023.pdf</certificate>
        </certificates>
        <date_responded>2023-09-21 12:30:12</date_responded>
    </question_response>
    <question_response>
        <response_id>2</response_id>
        <full_name>Jane Doe</full_name>
        <email_address>janedoe@gmail.com</email_address>
        <description>I am an experienced FrontEnd Engineer with over 6 years experience.</description>
        <gender>MALE</gender>
        <programming_stack>REACT,VUE</programming_stack>
        <certificates>
            <certificate id="3">Adobe Certification 19-08-2023.pdf</certificate>
            <certificate id="4">Figma Fundamentals 19-08-2023.pdf</certificate>
        </certificates>
        <date_responded>2023-09-23 12:30:12</date_responded>
    </question_response>
</question_responses>
```

The API should support:

- pagination of the records.
- filtering of the responses based on **email_address**

4. To download a certificate by providing the **id** of the certificate as a URL Parameter
    - url: `/api/questions/responses/certificates/{id}`
    - method: **GET**

Provide a **Postman Collection** documenting the endpoints above with their saved responses

3. User Interface (Mobile or Web)

Create a User Interface for the application.

- For **mobile developers**, use your preferred mobile development languages or framework i.e. **Android**, **Flutter** or **React Native**.
- For **web & backend developers**, use your preferred web development languages or framework.

The User Interface should have two pages:

1. **Survey Form**
2. **Survey Responses**

1. Survey Form

The page will have the form through which users can respond to the questions.

Requirements

1. The form should be a stepped form;with question as a step.
2. The list of questions should be fetched by making a request to the **Endpoint 1** in the **REST API** section above
3. The form should have **Next** and **Previous** button to navigate through each question.
4. On the first question, the **Previous** button should be hidden
5. For questions with *required* - **yes**, ensure the user provides a response before proceeding to the next question
6. The final step should have a preview of all the collected data and a **Submit** button to submit the collected data.
7. On clicking the Submit button, the responses should be submitted to the database via the **Endpoint 2** in the **REST API** section above
8. Use the appropriate form input for each question i.e. *long_text* => *textarea*

2. Survey Responses

The page will be used to show the submitted responses to the questions.

Requirements

1. Fetch the list of submitted responses using **Endpoint 3** in the **REST API** section above
2. The list should be paginated
3. You should be able to filter the responses using the **email_address**
4. You should be able to download the certificates using the **Endpoint 4** in the **REST API** section above

Add a navigation to be able to switch between the two pages.

Task guidelines

As an applicant you are expected to do all the components of the application:

1. Database
2. REST API
3. User Interface i.e. either **Web** or **Mobile**

Submission guidelines are as follows:

1. Create **public** GitHub repositories to version your source code:
    - `simple-survey-client` to store your *user interface* code.
    - `simple-survey-api` to store your:
        1. *ERD Diagram*
        2. *database SQL file*
        3. *REST API code*
        4. *Postman Collection*. each repository should have a `README.md` documenting:
    - how to set up and run your application on a local machine
    - the deployment process (*OPTIONAL but more points to those who can*).
2. Deploy your application and provide a public URL to access it. For the **mobile developers**, generate an **APK** of your application and have it on the `simple-survey-client` GitHub Repository. (*OPTIONAL but more points to those who can*)
3. Timeline to complete the project is **2 weeks**. Submit the task by **Monday 13th November 2023**. Submit by sending an email to recruitments@skyworld.co.ke with the following attached:
    1. Link to the `simple-survey-client` GitHub Repository
    2. Link to the `simple-survey-api` GitHub Repository
    3. Link to the public deployed web application for **web & backend developers**. (*OPTIONAL but more points to those who can*).
    - **Ensure the above attached are fully functional**.
4. Only shortlisted applicants will be contacted for the interviews.

NOTE: You will be expected to demonstrate your application during the interview process

Be innovative and creative. All the best.