Front-End Test: Login Interface

Introduction

Your task is to create an integrated version of the **Qencode Login UI** using **React**, based on the **design from Figma** and the **Authentication API** specification provided below.

Objectives

Design Link: Figma Design

- Implement the Figma Design as accurately as possible using React components.
- Write JavaScript code within the React framework to handle form submission, input validation, and interactions with the provided API endpoints.

Tasks

1. **Create Components**

- Create Reusable React components for the following pages according to the Figma Design.
  - Login
  - Forgot password
  - Reset password
- Ensure that the components are responsive and work on both mobile and desktop views.

2. **Design Implementation**

- Use modern CSS techniques to achieve the layout.
- Pay attention to details such as font sizes, padding, and margins to match the design.

3. **Functional Implementation**

- Use React state and props effectively to handle form data and validations.
- Implement client-side validation:
  - The email should be a valid email format.
  - The password fields should be at least 8 characters long.
- Simulate the "Forgot your password?" flow as indicated in the screenshots.

4. **API Integration**

- Integrate the forms with the provided API endpoints using `fetch` or `axios`.

- Handle different API response scenarios, such as successful login, login failure, password reset success, and password reset failure.

5. **Error Handling**

- Implement error handling that informs users of any input mistakes or API request issues.
- Utilize conditional rendering in React to show or hide error messages as needed.

6. **State Management**

- You may use React's built-in state management or incorporate any state management library of your choice if necessary.

Authentication API

API Endpoint	https://auth-qa.qencode.com
API Version	v1
API References	https://auth-qa.qencode.com/v1/auth-api-references
Content type	application/json
Headers	Content-Type

Method Name	Path	API References
Login/signup Section		
Login	/v1/auth/login	https://auth-qa.qencode.com/v1/auth-api-references#tag/login/operation/login_v1_auth_login_post
Authentication Section		
Access Token	/v1/auth/access-token	https://auth-qa.qencode.com/v1/auth-api-references#tag/auth/operation/access_token_v1_auth_access_token_post
Refresh Token	/v1/auth/refresh-token	https://auth-qa.qencode.com/v1/auth-api-references#tag/auth/operation/refresh_token_v1_auth_refresh_token_post
Password Section		
Password Reset	/v1/auth/password-reset	https://auth-qa.qencode.com/v1/auth-api-references#tag/password/operation/password_reset_v1_auth_password_reset_post
Set new Password	/v1/auth/password-set	https://auth-qa.qencode.com/v1/auth-api-references#tag/password/operation/password_set_v1_auth_password_set_post

Login

Method	Arguments/Body	Returns
POST	email: stringpassword: string	error: int = 0detail: stringtimestamp: float = CURRENT_TIMESTAMPaccess_token: stringrefresh_token: stringtoken_expire: TIMESTAMPrefresh_token_expire: TIMESTAMP

Verify Access Token

Method	Arguments/Body	**Headers**	Returns	HTTP Status Code	Description
POST	null	Authorization: Bearer *access_token*	null	200 or 401	Possible use case. Checking the validity of the token signature. Checking the integrity of the payload (whether the payload has been modified). Can be used to validate the token when opening sensitive pages. (such as billing or user_settings)

Password Reset

Method	Arguments/Body	Returns
POST	email: "string",	error: int = 0detail: "Please check your email {0} to complete the password reset process."timestamp: float = CURRENT_TIMESTAMP

Set new Password

Method	Arguments/Body	Returns
POST	token: "string"secret: “string“password: “string“	error: int = 0detail: "Password reset successfully"timestamp: float = CURRENT_TIMESTAMP

Submission

Submit your completed assignment by pushing to a GitHub repository and providing us with the link. Please also deploy your code to netlify.com and provide us the link for testing.

Deliverables

- A link to a GitHub repository containing your project.
- A `README.md` file with setup instructions and any scripts to run your application.
- A Netlify link for testing

Code Quality

- Organize your project structure and files in a clean and logical manner.
- Write well-documented, maintainable code.
- Follow best practices for React development.

Evaluation Criteria

- The accuracy of the design implementation compared to the Figma mockups.
- The functionality of the React components and form validation.

- Proper state management and API integration.
- Responsiveness and aesthetic alignment with the design.
- Code quality, including project structure, coding standards, and documentation.