- # Frontend Design

1. API Service Integration (useAxiosPrivate Hook):

Function: Manages Axios instance with interceptors for handling authorization tokens.
Request Intercept: Attaches the access token from the authentication context to request headers.
Response Intercept: Handles token expiration. On a 403 error, it retries the request with a refreshed access token.
Dependencies: Uses custom hooks useRefreshToken for refreshing tokens and useAuth for accessing the authentication context.

2. Authentication Hooks:

useAuth: Provides authentication context and state management.
useRefreshToken: Sends a request to refresh the access token using the stored refresh token. Updates the authentication state with the new access token.
useLogout: Sends a request to revoke the refresh token, clearing the authentication state upon success.

3. Member Management API:

Endpoints:
getAllMembers: Fetches all members, with optional sorting by birthday.
getSortedMembers: Fetches members sorted by date birthdays.
createMember: Creates a new member record.
deleteMember: Deletes a member by ID.
getMemberById: Fetches a member by ID.
editMember: Updates a member by ID.
getMembersWithBirthdaysToday: Fetches members whose birthdays are today.

4. Custom Axios Instance (axiosPrivate):

Configuration: Base Axios instance configured to include credentials in requests.

- ## Backend Design

1. User Authentication and Authorization:

Token Generation:
Access Token: Short-lived token (15 minutes).
Refresh Token: Long-lived token (7 days).
Registration: Validates and hashes the user password, creates user with associated role.
Login: Validates user credentials, generates tokens, stores refresh token in a cookie.
Refresh Token: Validates the refresh token, issues a new access token and refresh token.
Logout: Revokes the refresh token by clearing it from the user record.

2. Role-Based Access Control:

Middleware: Verifies JWT and extracts user roles.
Role Verification: Ensures certain endpoints are only accessible by users with specific roles (e.g., admin).

3. Member Management:

CRUD Operations:

Create: Validates member age and required fields before creation.
Read: Retrieves members, supports sorting by birthday.
Update: Updates member details by ID.
Delete: Deletes a member by ID.
Birthday Logic: Filters members to find those with birthdays today and sorts by date birthdays.

- ## Database Models (Using Sequelize):

User Model: Stores user credentials, role ID, and refresh token.

Role Model: Stores role names (e.g., admin, user).

Member Model: Stores member details (first name, last name, birth date, country, city).

API Routes:

Authentication Routes:

/register: Handles user registration.

/login: Handles user login.

/refresh: Handles token refresh.

/revoke: Handles token revocation (logout).

Member Routes:

/members: Handles CRUD operations for members.

/members/birthday: Retrieves members with birthdays today.

/members/sorted: Retrieves members sorted by upcoming birthdays.

Security Considerations:

JWT Secret Management: Securely manage JWT secrets using environment variables.

Token Expiry: Short expiry for access tokens and periodic refresh using refresh tokens.

HTTP-Only Cookies: Store refresh tokens in HTTP-only cookies to mitigate XSS attacks.

Password Hashing: Use bcrypt for securely hashing user passwords.

Authorization Middleware: Ensure protected routes are accessible only to authenticated users with appropriate roles.