Login Chat Portfolio

Overview

Using Google OAuth, JWT login to create a chat feature.

User Stories and Estimation

*	User Story	Groomed Estimation (hour)
1	Development Environment	1
2	UI definition figma (login, chat) key feature - after logging in display the user's name, login fail - modal	5
3	Investigation for google Oauth - response definition (success, fail)	3
4	Error code definition	2
5	Component design ● Component structure ○ Business Component: A component dedicated to handling business logic. ○ ex) Communicating with third-party APIs to retrieve and process data. ○ Business Component ■ Props ● apiEndpoint: string (URL to request API) ■ State Variables ● data: ApiData[] | null (Data fetched from the API) ● loading: boolean (Loading state of the data) ● error: string | null (Error message) ■ Functions ● fetchData: async (An asynchronous function to request data and update state) ■ Usage	6

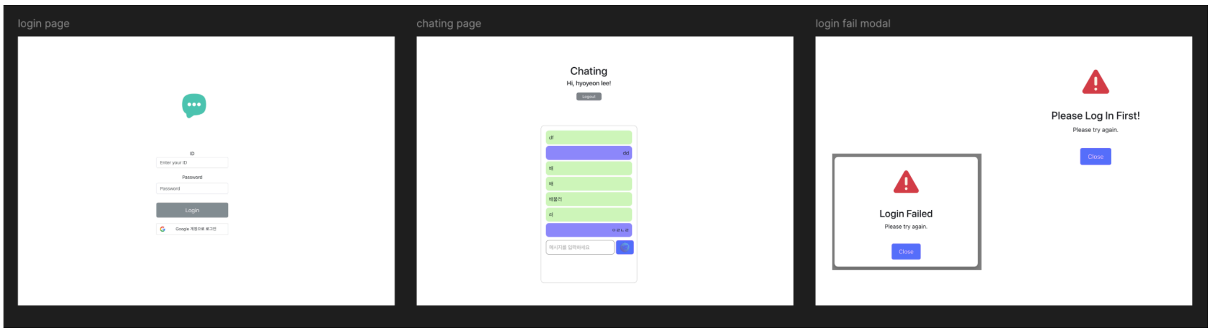
	● <BusinessComponent apiEndpoint="https://api.example.com/data" /> ○ Presentational Component - for rendering ex) modal, button ○ ex) Button Component ■ Props ● label: string ● onClick: () => void ● variant: 'primary' | 'secondary' ● disabled: boolean ■ Usage ● <Button (reference button Component) ● label="Login"/ "send chat" ● onClick={handleSubmit} (when clicked move to handleSubmit) ● variant="primary" ● /> ○ Modal Component(login Failed) ■ label: “Login Failed” ■ onClose ={handleCloseModal} ■ show = {showModal} ● Managing login State ○ useEffect triggers when the component mounts, updates, or unmounts, based on the dependencies. ○ need to track user activity (ex. mouse movements, clicks) and use a timer to log out after 10 minutes of inactivity. ● Security ○ Manage token: After 10 minutes of inactivity, the token in the browser's cookies should be expired and deleted. ○ error msg “You have been logged out due to inactivity for more than 10 minutes.”	
6	Dev sequence	2
7	Run the application and write commands from git clone to something like npm run dev	2

User story 1 - Development Environment

IDE- vscode
node - v22.2.0
npm - 10.7.0
javascript - ES6 or later
react - 18.3.1 (npm show react version)
Google OAuth - 2.0

User story 2 - UI definition

https://www.figma.com/design/fOgDIFpK5vBIhDZMa4uZLy/Login-Chat-Portfolio?node-id=0-1&node-type=canvas&t=jjPMy6nuThFBYJEc-0

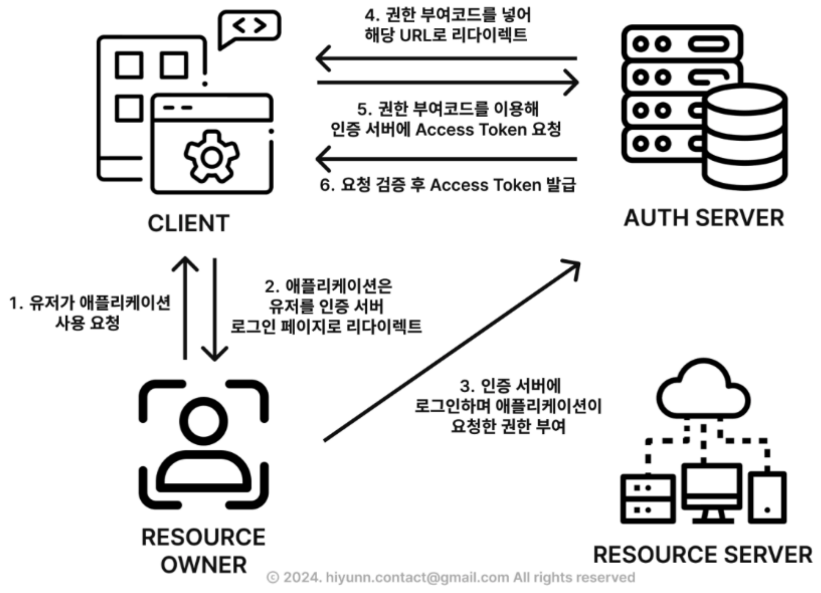


User story 3 - Investigation for google Oauth

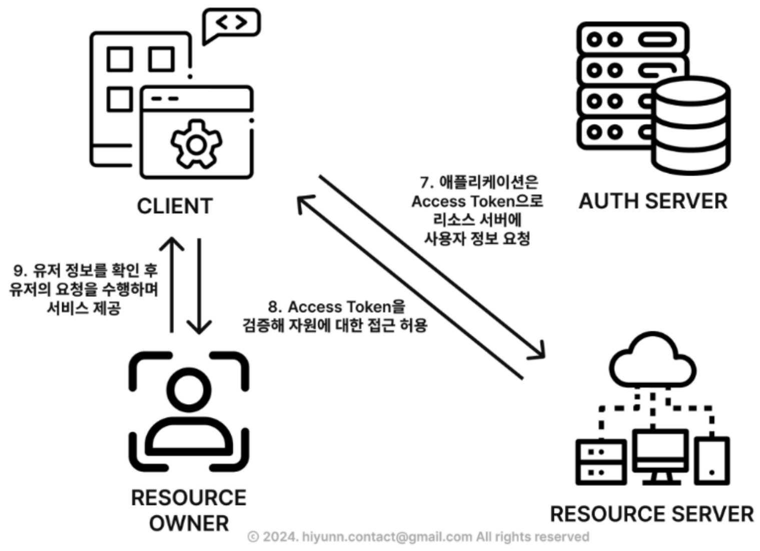
(request할때 어떤것들이 필요한지, response는 어떤식으로 오는지)
OAuth 2.0 allows users to share specific data with an application while keeping their usernames, passwords, and other information private.

Flow Chart

RESOURCE OWNER: The person who wants to use the application, such as user.
CLIENT: The application that needs the user's access permissions to provide services and features, represents goldie-front
AUTH SERVER: The server that manages access for the application, Google auth server
RESOURCE SERVER: The server that holds the user's information, represents Google resource server, not goldie-back


4. 권한 부여코드를 넣어
해당 URL로 리다이렉트
5. 권한 부여코드를 이용해
인증 서버에 Access Token 요청
6. 요청 검증 후 Access Token 발급
CLIENT
AUTH SERVER
1. 유저가 애플리케이션
사용 요청
2. 애플리케이션은
유저를 인증 서버
로그인 페이지로 리다이렉트
3. 인증 서버에
로그인하며 애플리케이션이
요청한 권한 부여
RESOURCE
OWNER
RESOURCE SERVER
© 2024. hiyunn.contact@gmail.com All rights reserved

1. 유저가 애플리케이션 사용요청: When the user wants to use the application
2. 애플리케이션은 유저를 인증 서버 로그인 페이지로 리다이렉트: the application redirects the user to the authentication server's login page.
3. 인증 서버에 로그인하여 애플리케이션이 요청한 권한 부여: The user logs in on that page and grants permissions t o the application.
4. 권한 부여코드를 넣어 해당 **URL**로 리다이렉트**:** The authentication server then includes an authorization code in the URL and redirects the page.
5. 권한 부여코드를 이용해 인증 서버에 **Access Token** 발급: The application extracts the authorization code from the URL and requests an Access
Token from the authentication server.
6. 요청 검증 후 **Access Token** 발급: The authentication server verifies the request and, if valid, issues an Access Token.


CLIENT
AUTH SERVER
7. 애플리케이션은 Access Token으로 리소스 서버에 사용자 정보 요청
8. Access Token을 검증해 자원에 대한 접근 허용
9. 유저 정보를 확인 후 유저의 요청을 수행하며 서비스 제공
RESOURCE OWNER
RESOURCE SERVER
© 2024. hiyunn.contact@gmail.com All rights reserved

7. 애플리케이션은 **Access Token**으로 리소스 서버에 사용자 정보 요청: The application uses the Access Token to request resources, such as user information, from the resource server.
8. **Access Token**을 검증해 자원에 대한 접근 허용: The resource server verifies the Access Token and, if valid, allows access.
9. 유저 정보를 확인 후 유저의 요청을 수행하며 서비스 제공**:** The application performs the verified user's requests and provides the service.

—-- 노트 —---

사용자 ➔ 애플리케이션 ➔ 구글 로그인 ➔ 구글 인증 서버 ➔ 애플리케이션 ➔ 구글 리소스 서버 ➔ 애플리케이션 ➔ 사용자

RESOURCE OWNER (리소스 소유자**)**: 애플리케이션을 사용하려는 사람, 즉 사용자를 의미합니다. 예를 들어, 이 서비스나 기능에 접근하고자 하는 일반 사용자입니다.
CLIENT (클라이언트**)**: 사용자로부터 접근 권한을 받아 서비스를 제공하려는 애플리케이션을 뜻합니다. 이 경우에는 goldie-front 애플리케이션이 클라이언트 역할을 합니다.
AUTH SERVER (인증 서버**)**: 애플리케이션을 위한 접근 권한을 관리하는 서버입니다. 이 예에서는 Google 인증 서버가 그 역할을 하며, 인증 서버는 사용자가 애플리케이션에 접근할 수 있도록 인증을 처리합니다.
RESOURCE SERVER (리소스 서버**)**: 사용자 정보를 보유하고 있는 서버입니다. 여기에서는 Google 리소스 서버가 리소스 서버 역할을 하며, 사용자의 정보를 저장하고 있습니다. 중요한 점은 이 서버가 goldie-back이 아닌, Google의 리소스 서버라는 것입니다.

구글 인증 서버는 사용자가 애플리케이션에 안전하게 로그인하도록 지원하는 동시에, 애플리케이션이 필요한 권한만 부여되도록 보장합니다.
 구글 리소스 서버는 구글 인증 서버로부터 발급된 Access Token이 유효할 때만 사용자 정보와 같은 자원에 대한 접근을 허용하여 데이터를 보호합니다.
—-------------------—---

Parameters

Parameters	**Description**	*
client_id	The client ID for your application.	Required
redirect_uri	Determines where the API server redirects the user after the user completes the authorization flow.	Required
response_type	Determines whether the Google OAuth 2.0 endpoint returns an authorization code. code for web server applications.	Required
scope	your application could access on the user's behalf.	Required
access_type	Whether to refresh the token when the user is not in the browser	Recommended
state	Specifies any string value that your application uses to maintain state between your authorization request and the authorization server's response.	Recommended

Reference

https://developers.google.com/identity/protocols/oauth2/web-server
React OAuth2 | Google - https://www.npmjs.com/package/@react-oauth/google

User story 4 - Error code definition

Error code	Error title	Error description
E-001	admin_policy_enforced	Occurs when a Google Workspace administrator has made it possible for a Google Account to grant permission to some or all of the requested scopes.
E-002	disallowed_useragent	open the authorization link in a standard browser instead of an embedded browser.
E-003	org_internal	The OAuth client ID in this request belongs

		to a project that restricts access to Google Accounts within a specific Google Cloud Organization.
E-004	invalid_client	The OAuth client secret is incorrect.
E-005	invalid_grant	When refreshing an access token or requesting incremental authorization, the token might be expired or invalidated.
E-006	redirect_uri_mismatch	The redirect_uri provided in the authorization request doesn't match any authorized redirect URIs for the OAuth client ID.
E-007	invalid_request	The request is invalid due to improper formatting, missing parameters, or unsupported authorization methods—please verify your OAuth integration follows Google's guidelines.

User story 5 - Component Design

- Component Structure
  - Business Component - for business logic ex) 서드파티 API와 통신 후 데이터를 취득
  - Presentational Component - for the render ex) modal, button

- State Management
  - When to use useEffect
    - Start a 10-minute timer as soon as the component is rendered after a successful login.
- Security
  - (토큰) Manage token: After 10 minutes of inactivity, the token in the browser's cookies should be expired and deleted. (refresh-token)
  - Error msg "You have been logged out due to inactivity for more than 10 minutes."

User story 6 - Dev Sequence

1. google login
   a. Determine the endpoints for using the Google API on the client side and properly implement the Google login button.
   b. GCP settings
   c. Access Token is properly stored in the cookies, (you can use document.cookie in the browser's console.)

    d. redirect chatting page

2. chatting page
3. modal - fail

ex) server.

```
router.post('/', async function(req, res, next) {
 //const redirectURL = 'http://127.0.0.1:3000/oauth';
 const redirectURL = 'http://talkwithyou.online:3000/oauth';

 const oAuth2Client = new OAuth2Client(
   process.env.CLIENT_ID,
   process.env.CLIENT_SECRET,
     redirectURL
   );

   // Generate the url that will be used for the consent dialog.
   const authorizeUrl = oAuth2Client.generateAuthUrl({
     access_type: 'offline',
     scope: 'https://www.googleapis.com/auth/userinfo.profile  openid ',
     prompt: 'consent'
   });

   res.json({url:authorizeUrl})

});
```

how to remove node_modules
git rm -r --cached . git add . git commit -m "remove node_modules" git push

User story 7 - How to run login chat

Node.js version 22.2.0

Frontend start: **npm start**

    Compiled successfully! Local: http://localhost:3000

Server start: **npm install**

```
➜  login-chat git:(login_page) ✗ cd server
➜  server git:(login_page) ✗ node index.js
Server running on http://localhost:3001
```

http://hyoyeon.s3-website.us-east-2.amazonaws.com/