OAuth Social Login

A PROJECT REPORT

Submitted by

Abhishek Kumar – 23BCS13197
Saurabh Joshi - 23BCS13165
Kartikay Bisht - 23BCS11163
Dipen Rana - 23BCS11287

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University
October, 2024

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Introduction to Project:

This project focuses on implementing secure OAuth2-based social login functionality in a MERN (MongoDB, Express, React, Node.js) web application. Modern applications require smooth and secure authentication to ensure a seamless user experience. Traditional password-based authentication has several drawbacks, including password fatigue, security risks, and poor user convenience. Social login addresses these issues by allowing users to authenticate using trusted identity providers such as Google and Facebook.

The project uses the OAuth 2.0 Authorization Code Flow with PKCE (Proof Key for Code Exchange) to ensure security across both web and mobile browsers. User identities obtained from external providers are mapped to local user records stored in MongoDB, supporting both first-time user provisioning and returning-user login. A secure session mechanism is implemented using JWT tokens stored in httpOnly cookies, preventing client-side tampering.

Key Features:

- **Sign in with Google and Facebook**
- **Secure OAuth2 Authorization Code + PKCE flow**
- **Automatic user account creation on first login**
- **Account linking/unlinking of multiple providers**
- **JWT-based session stored in httpOnly cookies**
- **Profile synchronization (name, avatar) with user consent**
- **Client-side React UI with redirect handling**
- **Robust security measures (CORS, CSRF, rate limiting)**
- **This project ensures both security and convenience, making authentication smooth while protecting user data.**

1.2 Identification of Problem:

Many web applications rely on traditional username-password login mechanisms. However, these systems have several challenges:

- Users must create and remember multiple passwords.
- Password reuse increases the risk of credential theft.
- Developers must manage password storage securely.
- Login friction often leads to poor user engagement and drop-offs.
- Cross-platform authentication is difficult to implement securely.

To overcome these challenges, the application integrates OAuth2 social login, allowing users to authenticate using existing trusted identities securely and effortlessly.

2. BACKGROUND STUDY

2.1. Existing Tool Limitations

Existing applications widely use Google and Facebook login. However, many open-source or low-quality implementations:

- Use insecure flows (implicit or direct token sharing).
- Store tokens in localStorage, making them vulnerable to XSS attacks.
- Do not support account linking, causing duplicate user accounts.
- Lack structured session management and logout handling.

Commercial Auth platforms like Firebase/Auth0 provide solutions but are:

- Costly at scale
- Less configurable
- Not suitable for academic/custom backend integrations

Thus, a secure custom implementation is required.

2.2. Problem Definition:

This project aims to solve the problem of secure and seamless authentication by:

- Implementing OAuth2 Authorization Code + PKCE flow
- Validating state and nonce to prevent replay attacks
- Ensuring user identity consistency across multiple logins
- Providing secure session cookies to prevent token theft
- Mapping social identities to stable local user profiles

2.3. Goals / Objectives:

1. Allow users to securely log in using Google and Facebook.
2. Automatically create or link user profiles based on identity data.
3. Maintain secure, tamper-proof JWT sessions using httpOnly cookies.
4. Provide a React frontend that handles redirects and login state.
5. Implement user account settings to link/unlink social providers.
6. Apply security measures:
    - Strict CORS
    - CSRF token protection
    - Rate limiting
    - Safe storage of provider secrets

3. DESIGN FLOW

3.1. Evaluation & Selection of Specifications/ Features:

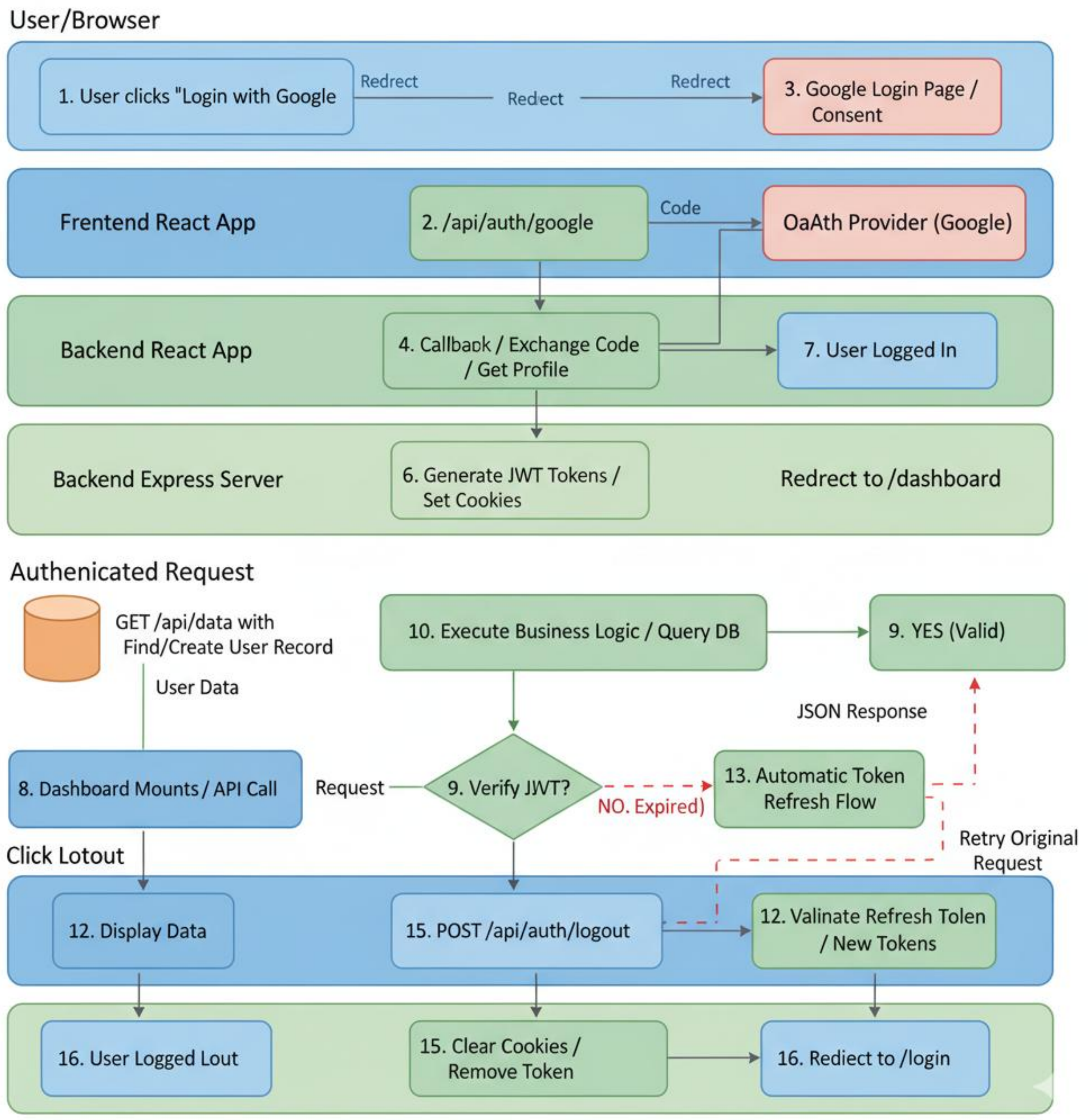
The following key specifications were chosen:

- OAuth Providers: Google and Facebook
- Auth Flow: Authorization Code Flow + PKCE
- Database: MongoDB for persistent identity records
- Session: JWT access token in httpOnly cookies
- Frontend: React with provider login buttons
- Security: CSRF protection, strict CORS, input validation

3.2. Analysis of Features and Finalization Subject to Constraints:

Requirement	**Considerations**	**Final Choice**
Security Level	Must mitigate phishing and token theft	Used PKCE + httpOnly cookies
User Convenience	Reduce friction	Social login buttons + auto provisioning
Maintainability	Adaptable to more providers	Modular provider strategy
Privacy	Minimal PII storage	Storing only name, email, avatar (optional)

3.3. Design Flow

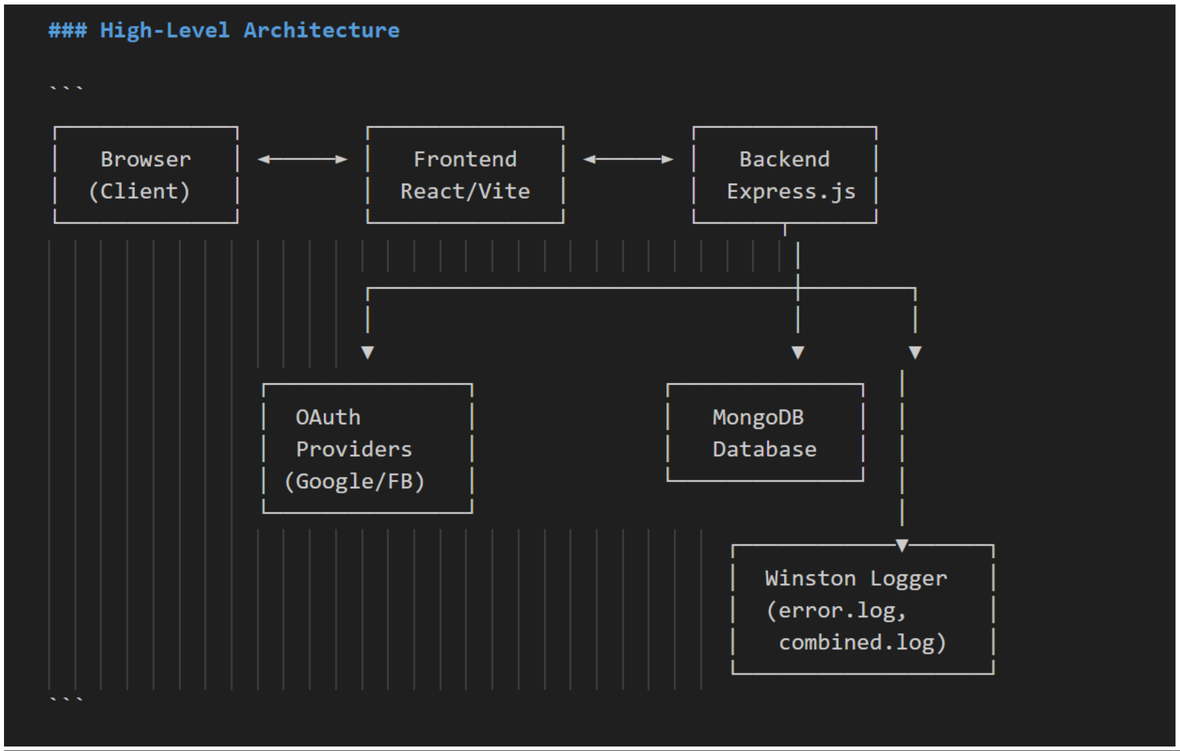

User/Browser
1. User clicks "Login with Google
Redrect
Redect
Redrect
3. Google Login Page / Consent
Frentend React App
2. /api/auth/google
Code
OaAth Provider (Google)
Backend React App
4. Callbaok / Exchange Code / Get Profile
7. User Logged In
Backend Express Server
6. Generate JWT Tokens / Set Cookies
Redrect to /dashboard
Authenicated Request
GET /api/data with Find/Create User Record
User Data
10. Execute Business Logic / Query DB
9. YES (Valid)
JSON Response
8. Dashboard Mounts / API Call
Request
9. Verify JWT?
NO. Expired)
13. Automatic Token Refresh Flow
Retry Original Request
Click Lotout
12. Display Data
15. POST /api/auth/logout
12. Valinate Refresh Tolen / New Tokens
16. User Logged Lout
15. Clear Cookies / Remove Token
16. Rediect to /login

4. RESULT ANALYSIS AND VALIDATION

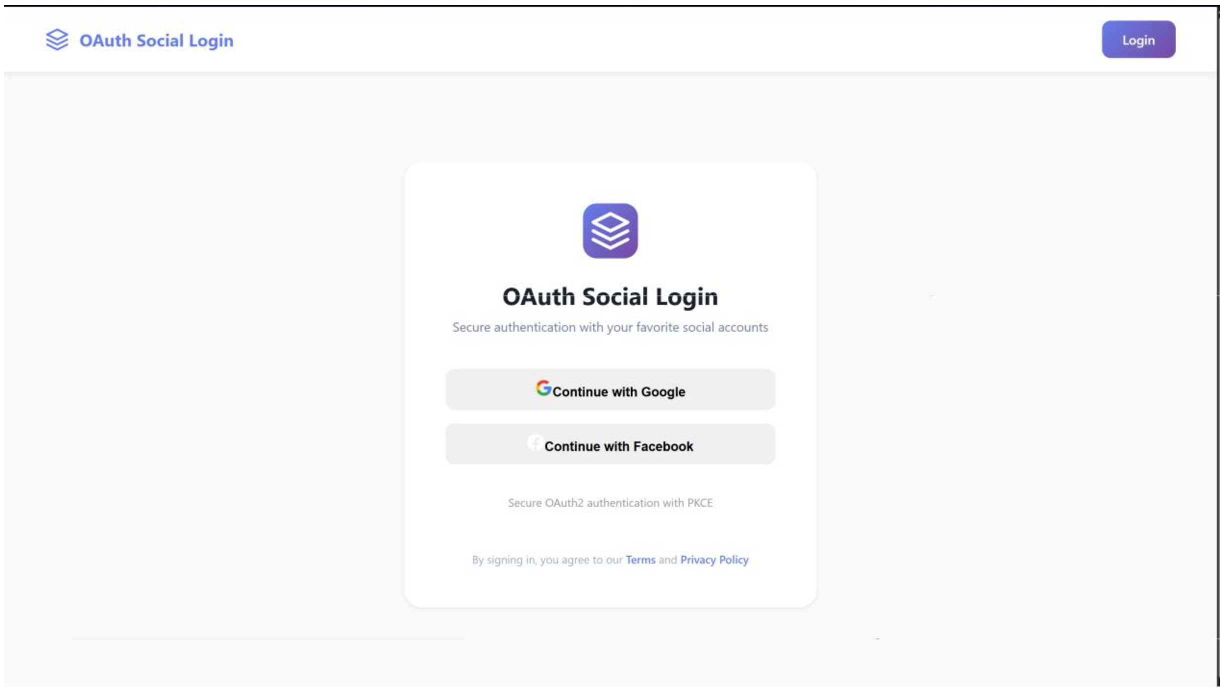
4.1 Implementation of Solution:

- The backend was built using Node.js + Express.
- MongoDB stores user accounts, linked provider identities, and roles.
- On first login, a new user account is created and stored.
- On subsequent logins, the existing user is retrieved and authenticated.
- The React frontend shows "Continue with Google/Facebook" buttons.
- JWT-based sessions allow secure navigation without re-login.
- A settings page allows the user to connect or disconnect social accounts.
- Logout clears server-side session data and cookies.

System Architecture :


### High-Level Architecture
Browser (Client)
Frontend React/Vite
Backend Express.js
OAuth Providers (Google/FB)
MongoDB Database
Winston Logger (error.log, combined.log)

Output User Screen :


OAuth Social Login

Login

OAuth Social Login

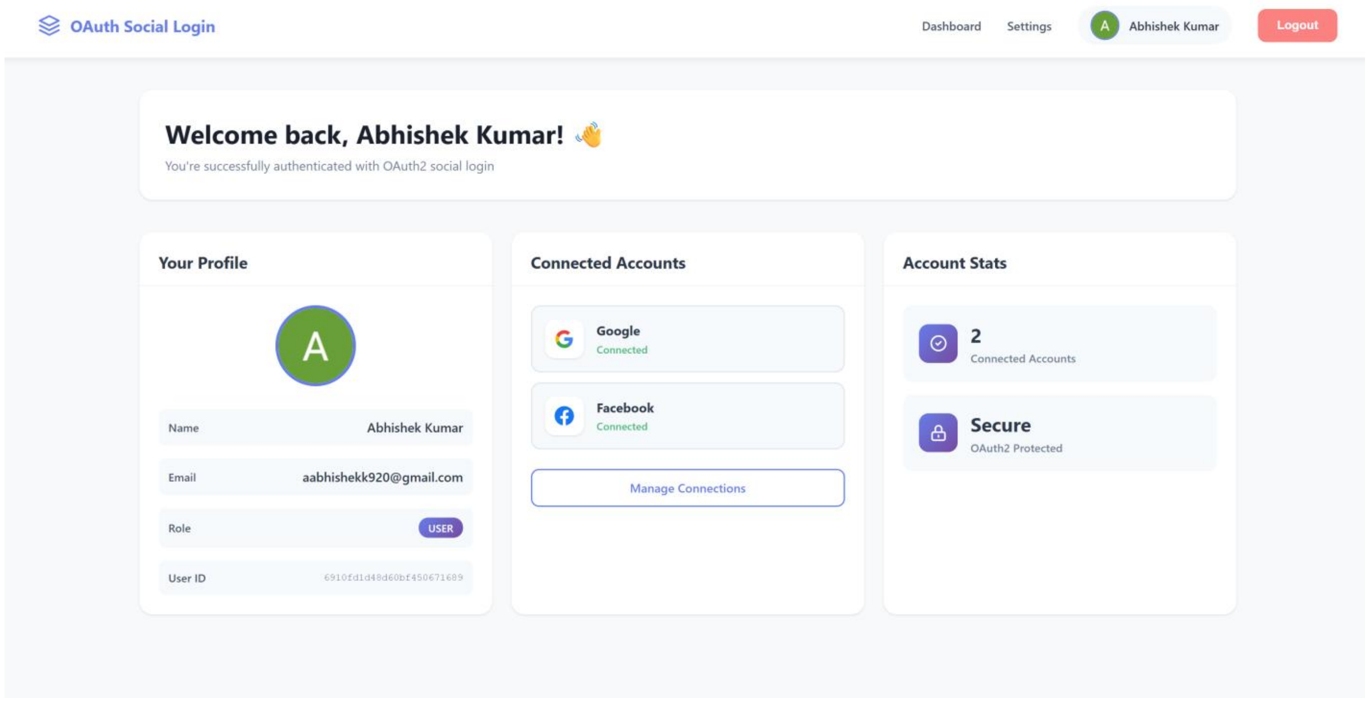
Secure authentication with your favorite social accounts

Continue with Google

Continue with Facebook

Secure OAuth2 authentication with PKCE

By signing in, you agree to our Terms and Privacy Policy


OAuth Social Login

Dashboard Settings A Abhishek Kumar Logout

Welcome back, Abhishek Kumar! 👋

You're successfully authenticated with OAuth2 social login

Your Profile

A

Name	Abhishek Kumar
Email	aabhishekk920@gmail.com
Role	USER
User ID	6910fd1d48d60bf450671689

Connected Accounts

Google
Connected

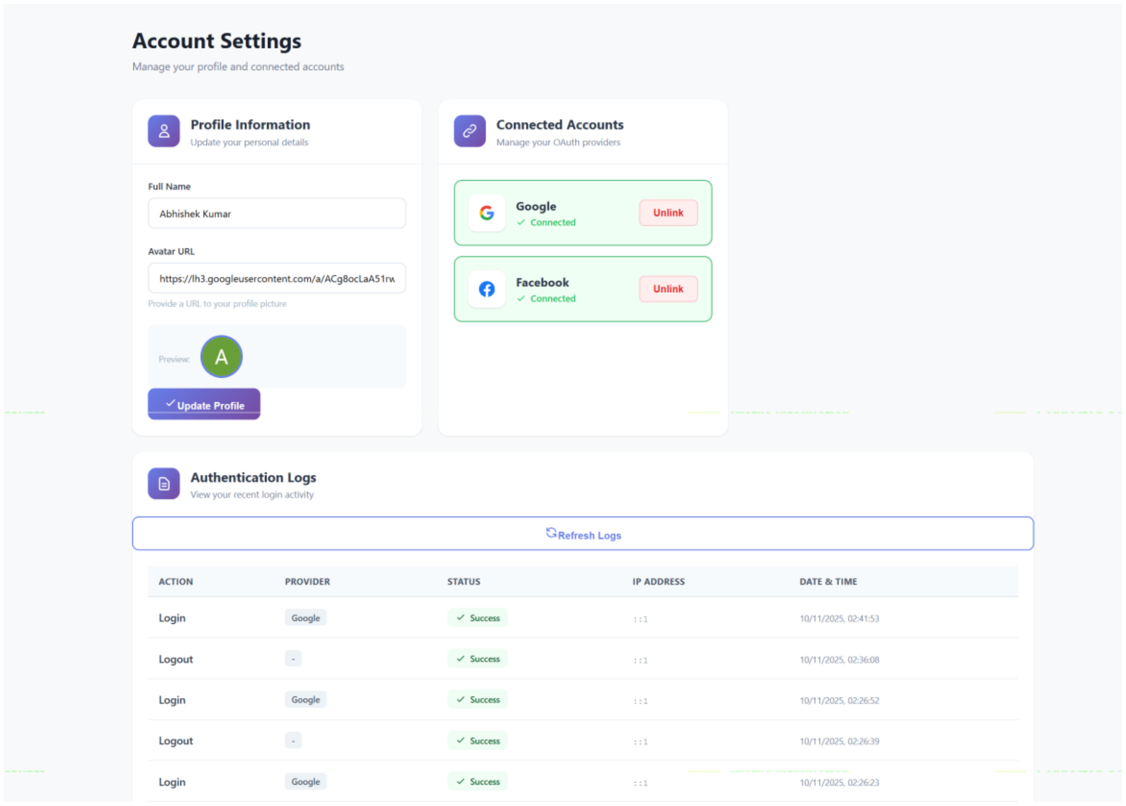
Facebook
Connected

Manage Connections

Account Stats

2
Connected Accounts

Secure
OAuth2 Protected



Account Settings

Manage your profile and connected accounts

Profile Information

Update your personal details

Full Name	Abhishek Kumar
Avatar URL	https://lh3.googleusercontent.com/a/ACg8ocLaA51rw

Provide a URL to your profile picture

Connected Accounts

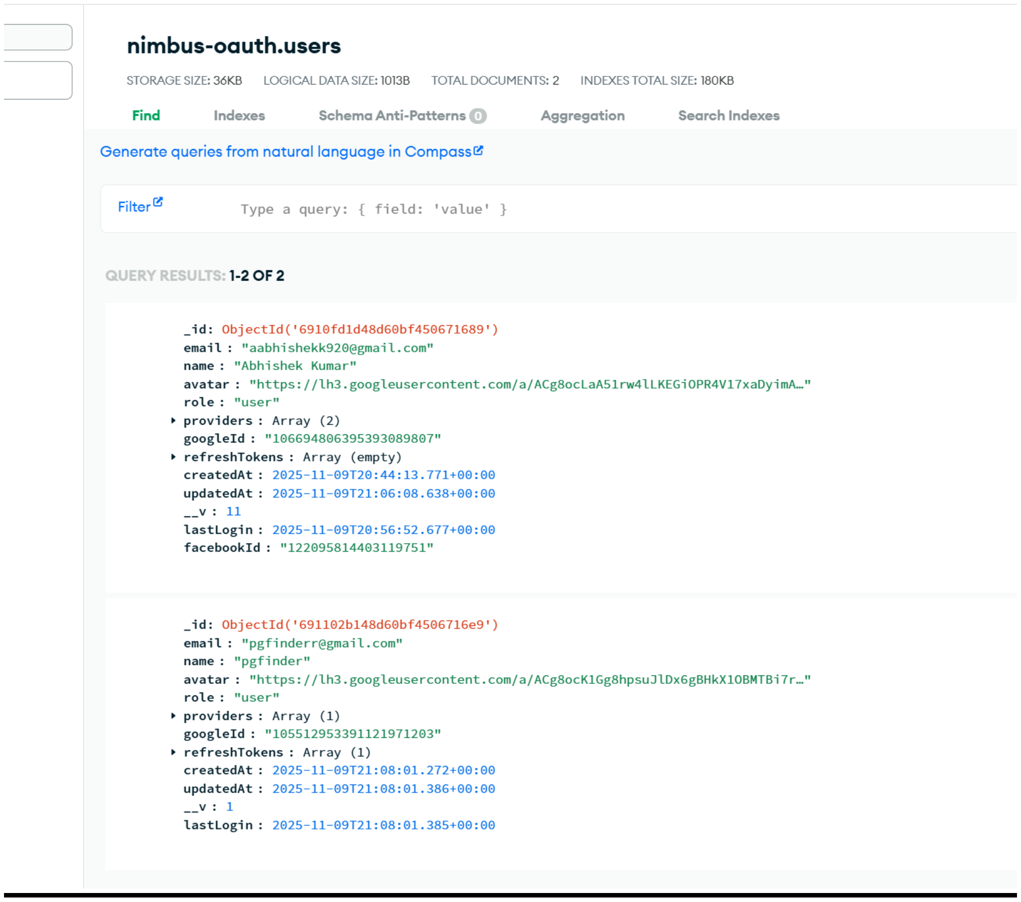
Manage your OAuth providers

Google	✓ Connected	Unlink
Facebook	✓ Connected	Unlink

Authentication Logs

View your recent login activity

ACTION	PROVIDER	STATUS	IP ADDRESS	DATE & TIME
Login	Google	✓ Success	::1	10/11/2025, 02:41:53
Logout	-	✓ Success	::1	10/11/2025, 02:36:08
Login	Google	✓ Success	::1	10/11/2025, 02:26:52
Logout	-	✓ Success	::1	10/11/2025, 02:26:39
Login	Google	✓ Success	::1	10/11/2025, 02:26:23



nimbus-oauth.users

STORAGE SIZE: 36KB   LOGICAL DATA SIZE: 1013B   TOTAL DOCUMENTS: 2   INDEXES TOTAL SIZE: 180KB

Type a query: { field: 'value' }

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('6910fd1d48d60bf450671689')
email : "aabhishekk920@gmail.com"
name : "Abhishek Kumar"
avatar : "https://lh3.googleusercontent.com/a/ACg8ocLaA51rw4lLKEGiOPR4V17xaDyimA…"
role : "user"
providers : Array (2)
googleId : "106694806395393089807"
refreshTokens : Array (empty)
createdAt : 2025-11-09T20:44:13.771+00:00
updatedAt : 2025-11-09T21:06:08.638+00:00
__v : 11
lastLogin : 2025-11-09T20:56:52.677+00:00
facebookId : "122095814403119751"
```

```
_id: ObjectId('691102b148d60bf4506716e9')
email : "pgfinderr@gmail.com"
name : "pgfinder"
avatar : "https://lh3.googleusercontent.com/a/ACg8ocK1Gg8hpsuJlDx6gBHkX1OBMTBi7r…"
role : "user"
providers : Array (1)
googleId : "105512953391121971203"
refreshTokens : Array (1)
createdAt : 2025-11-09T21:08:01.272+00:00
updatedAt : 2025-11-09T21:08:01.386+00:00
__v : 1
lastLogin : 2025-11-09T21:08:01.385+00:00
```

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion:

The project successfully provides a secure and user-friendly authentication system using OAuth2 with Google and Facebook. It ensures strong security through PKCE, validated tokens, encrypted cookies, and proper provider configuration. The result is a seamless login experience that enhances user trust and simplifies access management.

5.2 Future Work:

Future enhancements may include:

- Support for more providers (GitHub, LinkedIn, Twitter)
- Adding Refresh Token Rotation for improved session management
- Multi-factor authentication (MFA)
- Admin dashboard for managing user roles and permissions
- Enabling mobile apps using the same authentication backend