# Integration Hiring Assessment — OAuth2 Based Multi-Provider Integration

**Audience:** Backend Engineer Candidates

**Difficulty:** Easy-Medium (Practical, Design-focused)

**Expected Time:** 2–3 hours

---

## Purpose of This Assessment

This assessment is designed to evaluate how you think and build as a **backend integration engineer**.

We are not testing only OAuth syntax. We are evaluating:

- Real-world OAuth onboarding experience
- Integration design for long-term extensibility
- Engineering judgment and trade-offs
- Code clarity and ownership

In production, integrations evolve continuously. Your solution should reflect that mindset.

---

## Problem Statement

Build a small integration framework that:

1. Performs OAuth2 Authorization Code flow with an OAuth provider.
2. Stores and manages tokens.
3. Calls a secured API using the access token.
4. Is **designed to support multiple OAuth providers in the future**.

This should not be a single-provider, one-off script.

---

## OAuth Provider for This Assessment

You may use either:

- OAuth.com demo authorization server
- OR any free OAuth provider (GitHub, Google, Slack, etc.)

The OAuth.com demo server behaves similarly to the OAuth Playground and is suitable for testing.

Example OAuth.com demo server:

```
None
Authorization URL:
https://authorization-server.com/authorize

Token URL:
https://authorization-server.com/token
```

---

## Client Registration Requirement

You **must register your own OAuth client** and generate:

- client_id
- client_secret
- redirect_uri

These must be supplied via configuration or environment variables.

Hard-coding secrets in source code is discouraged.

This step is intentionally part of the assessment.

---

## Required OAuth Flow

Your solution must implement:

1. Authorization Code flow
2. Redirect handling
3. Token exchange
4. Token storage
5. Token refresh when expired

6. API call using access token

---

## Core Design Requirement

Your architecture must allow new OAuth providers to be added with minimal code change.

Examples of future providers:

- GitHub
- Google
- Salesforce
- HubSpot
- Custom OAuth servers

We are explicitly evaluating your **extensibility design**.

Avoid designs where OAuth logic is tightly coupled to business logic.

---

## Expected Design Characteristics

We expect to see:

- Provider abstraction layer
- Config-driven provider definitions
- Clear separation of concerns
- Reusable OAuth client logic

Conceptually:

```
None
OAuthProvider
OAuthClient
TokenStore
ApiClient
```

Exact structure is up to you.

---

## Functional Requirements

Your solution should:

- Generate authorization URL
- Accept callback with authorization code
- Exchange for token
- Store token securely (in memory, file, or DB)
- Detect expiry
- Refresh token automatically
- Call a secured API
- Retry and Rate Limit Management [Good to Have]
- Log meaningful output

UI is not required. CLI or a simple HTTP server is sufficient.

---

## Deliverables

Please submit:

1. Source code
2. README including:
    - Setup instructions
    - OAuth flow explanation
    - Architecture explanation
    - How to add a new provider
3. Sample output or screenshots / video demo.
4. Any assumptions or limitations

---

## Allowed Languages

- Java (Preferred)
- Node.js
- Python
- Golang

---

## AI Usage Policy

You **may use AI tools** for:

- Learning OAuth concepts
- Debugging
- Syntax assistance

However: The final solution must reflect your own understanding and design.

During interview discussion, you will be asked to explain:

- Why you designed it this way
- How OAuth works in your solution
- How you would extend it

Pure copy-paste solutions will be easily identified.

AI is a tool — not a substitute for engineering ownership.

---

## What We Are Not Looking For

- UI polish
- Framework complexity
- Over-engineering

We care about **clarity, correctness, and design thinking**.

---

## Submission Format

- GitHub repository preferred

Note: Put following empty commits
“START” - when you start this assignment.
“END” - when you are done with your work and are ready to submit.
Anything before “START” commit and after “END” commit won’t be considered for evaluation.

-> Include a detailed README file explaining everything needed to run the project.

***You need to complete this assignment in 3 days from the time you receive this.***

**Once you are done please share your submission (GitHub repository link) over the mail to Ravi, Sahil and Rahul (Emails mentioned in the Contact section below).**

---

## Final Note from Engineering

Real integrations are messy, inconsistent, and constantly evolving.

We value engineers who:

- Design for change
- Think in abstractions
- Understand protocols deeply
- Write maintainable integration code

Treat this as a real production integration, not as a coding exercise.

---

## Contact

If you have any questions or need clarifications regarding this assessment, please feel free to reach out to:

- **Ravi Sadhwani (ravi@revsure.ai)**
- **Sahil Gupta (sahil@revsure.ai)**
- **Rahul Kumar (rahul.kumar@revsure.ai)**

We will be happy to assist you.