An Architectural Blueprint for a Cloud-Native, Dual-Identity Management System

I. Executive Summary & Strategic Recommendation

This report outlines a strategic architectural blueprint for the "POWERFUL MOVES" platform. The platform's success hinges on solving a complex, dual-identity challenge:

1. **Business-to-Consumer (B2C):** A public, scalable, and low-friction pathway for individual "students" to self-discover and onboard.
2. **Business-to-Business (B2B):** A secure, controlled, multi-tenant model for "classrooms" or "organizations" to be managed by a "teacher" (a delegated tenant administrator).
3. **Microservice Security:** A secure, unified method for propagating a user's identity and permissions across a distributed stack of services, including "PMOVES.AI" and "CATACLYSM STUDIOS INC."

Summary of Findings

The analysis of the identity market reveals a fundamental "Build vs. Buy" trade-off, with further bifurcation within each category.

- **"Buy" (Identity-as-a-Service - IDaaS):** This path offers rapid development, guaranteed security compliance, and enterprise-grade features out-of-the-box.
  - **Traditional IDaaS (e.g., Auth0):** These platforms are mature and feature-rich, with excellent developer-facing tools. However, their pricing models, typically based on Monthly Active Users (MAUs), create significant financial risk and unpredictable costs for platforms with a large B2C "student" user base.[1]
  - **B2B-Native IDaaS (e.g., WorkOS, Frontegg):** A new category of IDaaS purpose-built for B2B multi-tenancy. These solutions offer B2B-centric pricing (e.g.,

    per-tenant connection, not per-user) and features like pre-built administrative portals, which directly address the "classroom" use case.[3]

- **"Build" (Open-Source Software - OSS):** This path offers maximum control and eliminates vendor lock-in but carries substantial, often-underestimated, Total Cost of Ownership (TCO).
    - **Traditional OSS (e.g., Keycloak):** A powerful and comprehensive solution, but it is notoriously complex to configure for multi-tenancy, has a large infrastructure footprint, and requires deep, specialized engineering expertise for 24/7/365 management.[5] The TCO for a self-hosted Keycloak cluster can exceed $200,000 over three years.[7]
    - **Modern OSS (e.g., Zitadel):** A promising, cloud-native alternative built from the ground up for B2B multi-tenancy. It is API-first and uniquely provides B2B features like a delegated admin console out-of-the-box, making it a superior "Build" option.[9]

Primary Strategic Recommendation

A **"Buy" strategy centered on a B2B-Native IDaaS provider, specifically WorkOS,** is the recommended architectural path.

This recommendation is based on a critical financial and strategic alignment:

1. **De-risks B2C Growth:** WorkOS's pricing model for User Management (the B2C "student" component) is free for the first 1,000,000 MAUs.[4] This eliminates the "per-user trap" of other IDaaS vendors, allowing the platform's student base to grow without incurring punitive authentication costs.
2. **Aligns B2B Costs:** Its B2B features (like Single Sign-On and SCIM provisioning) are priced on a predictable, per-connection (per-"classroom") basis, not per-user.[4] This aligns cost directly with B2B revenue.
3. **Frees Resources:** This strategy allows the POWERFUL MOVES development team to focus on its core value proposition—the educational platform and AI services—rather than becoming experts in identity infrastructure, security patching, and high-availability IAM.

Secondary Recommendation (Hybrid Architecture)

An alternative, albeit more complex, strategy is a hybrid-cloud model. This would involve using a low-cost, high-scale B2C-focused provider like **Firebase Authentication** for the public "student" sign-up flow.[1] For the B2B "classroom" functionality, the platform would then use a dedicated B2B solution like **Auth0 Organizations**.[14]

This approach requires federating the two systems, using Auth0's Custom Database Connection feature to "migrate" Firebase users into Auth0's user store upon their first login to a B2B "classroom".[15] While technically feasible, this adds significant architectural and user-journey complexity.

High-Level Implementation Roadmap

A phased implementation is recommended:

1. **Phase 1: Foundation:** Select the IDaaS platform. Implement the core security protocols (OIDC with PKCE) and secure the first API endpoint with JSON Web Token (JWT) validation.
2. **Phase 2: B2C Launch:** Enable public self-service sign-up for "students," configure social logins, and launch the B2C offering.
3. **Phase 3: B2B Launch:** Implement the multi-tenant "Organization" (classroom) model, build the "teacher" (delegated admin) portal using the IDaaS APIs, and implement tenant-aware Role-Based Access Control (RBAC).
4. **Phase 4: Enterprise Readiness:** Build and expose a SCIM 2.0 provisioning endpoint to enable automated user lifecycle management for larger "classroom" clients.

II. Architectural Foundation: Deconstructing Your Dual-Identity Model

The platform's requirements demand two distinct identity architectures—Customer Identity and Access Management (CIAM) and B2B Identity and Access Management (B2B IAM)—to operate simultaneously. Understanding their conflicting drivers is the first step to designing a unified solution.

A. Defining the Two User Journeys: CIAM vs. B2B IAM

Journey 1: The B2C "Student" (CIAM)
This journey defines the platform's public-facing, consumer-oriented identity model. It is designed for individual "students" who discover the service and sign up on their own.

- **Function:** Self-service discovery, registration, and profile management.
- **Key Drivers:** This model is optimized for growth and conversion. The primary goals are to reduce friction, create a seamless user experience (UX), and scale to potentially millions of users.[17] Key features include easy-to-use social logins (e.g., Google, Facebook), passwordless options, and user-friendly account recovery processes.[17]

Journey 2: The B2B "Classroom" (Multi-Tenant B2B IAM)
This journey defines the platform's organizational identity model. It is designed for groups, such as "classrooms," schools, or corporate clients, where a single entity (the "teacher" or administrator) manages a discrete set of users.

- **Function:** Admin-led user provisioning, delegated administration, and secure resource isolation.
- **Key Drivers:** This model is optimized for security, control, and B2B integration. The "teacher" acts as a *tenant administrator*, with the authority to add, remove, and manage users *only* within their own "classroom".[20] This model requires features like delegated user management, federated authentication via Single Sign-On (SSO), and granular, policy-based authorization.[18]

B. Key Differences and Why It Matters

Failing to distinguish between these two models is a common architectural error. A platform that treats a B2B user like a B2C user (or vice versa) will fail at both security and user experience. The core differences are summarized below.

Table 1: CIAM (Student) vs. B2B IAM (Classroom) Comparison

Feature	**B2C/CIAM (Your "Students")**	**B2B/Multi-Tenant IAM (Your "Classrooms")**	**Insight & Implication**
Primary Goal	User Acquisition, Conversion, UX [18]	Control, Security, Delegation [19]	The platform must balance a frictionless public experience with strict organizational

			controls.
User Scale	Millions of individual users [17]	Thousands of users *per-tenant* [17]	The B2C component must be architected for massive scale at low cost.
Onboarding	Self-service sign-up, social logins [19]	Admin-provisioned, federated (SSO), email invites [20]	The platform requires two distinct user-creation paths: a public signup page and a private "invite to classroom" function.
Administration	User manages their own profile.	Delegated Admin ("Teacher") manages their users.[20]	This is the core B2B feature. The "teacher" *must* have a dedicated portal to manage their "classroom" roster, roles, and settings.
Identity & Data	User owns their identity; governed by privacy laws (e.g., GDPR).[17]	Organization ("Classroom") owns the user's corporate identity.[17]	This legal and data-ownership distinction is critical for data governance, privacy, and B2B contracts.

C. Foundational Architectural Principles

The chosen identity solution must be built upon modern principles to support the platform's distributed nature.

- **Cloud-Native & Microservices:** The platform ("PMOVES.AI," "CATACLYSM STUDIOS INC") is a distributed system of microservices.[23] A traditional, monolithic IAM system will create a bottleneck. The identity solution must be decentralized, API-first, and align with cloud-native design patterns that support loosely coupled services.[23]
- **Zero Trust:** No service or user is trusted by default. The security model must shift from a "trusted network" perimeter to one of continuous verification.[24] This means *every* API request to *every* microservice (e.g., to "PMOVES.AI") must be independently authenticated and authorized.
- **API-First:** All identity functions—user creation, login, permission changes—must be controllable via a robust API, not just a graphical UI. This enables the automation required for administrative tasks, provisioning, and integrating the "teacher's" IT background into the system's management.

III. Core Identity Protocols: The Lingua Franca of Secure Propagation

The query's request to "propagate credentials...in a secure way" translates directly to a technical requirement for a stateless, token-based authentication architecture. In a distributed system, passwords are never "propagated." Instead, they are exchanged *once* for a portable, verifiable identity document.

A. "Propagating Credentials" as Token-Based Authentication

In a monolithic application, a user's session is stored on the server. In a microservice "stack," there is no single server; the user's request may be handled by multiple, independent services.

1. Sending a user's password with every API request is a catastrophic security vulnerability.
2. Instead, the user authenticates *once* to a central **Authorization Server** (the Identity Provider or IdP).
3. The IdP, upon verifying the user's credentials, issues a **JSON Web Token (JWT)**. A JWT is a secure, digitally-signed "passport" that contains claims about the user (e.g., their user ID, their role, and their "classroom" ID).[25]
4. The "propagation" is the act of the client application (web or mobile) including this JWT

in the Authorization: Bearer header of every subsequent API request it makes to the platform's microservices.[25]
5. Each microservice (e.g., "PMOVES.AI") receives the request, inspects the JWT, and *independently* verifies its digital signature using the IdP's public key. This validation is done locally and instantly, confirming the user's identity without ever needing to contact a central database.[25] This is the essence of a secure, stateless, and scalable Zero Trust architecture.

B. OAuth 2.0 vs. OpenID Connect (OIDC)

These two protocols are the foundation of modern, token-based security and are often confused.

- **OAuth 2.0 (The Authorization Protocol):** OAuth 2.0 is a framework for *delegating access*. It is not an authentication protocol.[27] It allows a user to grant one application permission to access their resources in another application (e.g., "Allow this app to access your Google Calendar"). It defines the flow for obtaining an **Access Token**, which represents a set of permissions (scopes).[28]
- **OpenID Connect (OIDC) (The Authentication Protocol):** OIDC is a thin identity layer built *on top of* OAuth 2.0.[29] It answers the question "Who is this user?".[27] When a client application uses OIDC, it receives two tokens:
  1. An **Access Token** (from OAuth 2.0), for accessing resources (e.g., the "PMOVES.AI" API).
  2. An **ID Token** (from OIDC), which is a JWT containing verifiable information about the user's identity (e.g., user ID, email).[28]

The POWERFUL MOVES platform requires **OIDC** for all user logins, as it provides both the authentication (ID Token) and authorization (Access Token) needed to secure the stack.[32]

C. SAML vs. OIDC

- **SAML (Security Assertion Markup Language):** An older, XML-based federation standard.[34] It is robust and heavily used in enterprise and government sectors but is verbose, complex to implement, and poorly suited for modern single-page applications (SPAs) and mobile apps.[35]
- **OIDC:** The modern, lightweight standard that uses JSON and JWTs.[35] It is API-friendly, easier for developers, and designed from the ground up for web, mobile, and SPA

clients.[34]

Recommendation: The platform's primary authentication protocol *must* be **OIDC**. SAML support should only be offered as an *inbound* B2B feature, allowing "classroom" tenants (e.g., universities) to federate with their existing SAML-based enterprise IdPs.[34]

D. The Correct, Secure Flow: OAuth 2.0 Grant Types

An "OAuth 2.0 grant type" is the specific "flow" used to get a token. Choosing the wrong flow creates severe vulnerabilities.

- **Deprecated Flow (Do Not Use):** The "Implicit" grant type. This flow exposed the Access Token directly in the browser's URL, making it vulnerable to theft. It is now deprecated and must not be used.[38]
- **Modern Standard (Mandatory):** The **Authorization Code Flow with PKCE** (Proof Key for Code Exchange).[39] This is the industry-standard flow for all public clients (web and mobile).
  - **How it works:**
    1. The React app (client) initiates the login and generates a secret "code verifier."
    2. It redirects the user to the IdP's login page.
    3. The user authenticates. The IdP redirects the user back to the React app with a temporary, one-time authorization code.[41] This code is safe to be in the URL.
    4. The React app's backend *securely* exchanges this code (along with its secret "code verifier") with the IdP's token endpoint.
    5. The IdP validates the code and verifier, and only then returns the valuable ID and Access Tokens (JWTs).[40]
  - **Why it's Secure:** The valuable JWTs are *never* exposed in the browser history. They are only transmitted over a secure, direct, server-to-server channel, mitigating interception risks. This flow is the non-negotiable standard for the platform.[40]

IV. Pattern Analysis: Architecting for "Classrooms" (B2B Multi-Tenancy)

The "classroom" requirement defines the platform as a **multi-tenant** application. A "tenant" is a group of users (the "classroom") who share access to the same application instance but whose data and configuration are logically isolated from all other tenants.[42]

A. The Central Dilemma: Identity Isolation Models

The first architectural decision is how to isolate one tenant's (classroom's) data from another's.

- **Model 1: Physical Isolation (Silo Model)**
    - **Concept:** This model provisions a completely separate, dedicated infrastructure stack (e.g., a new database, new application instances) for *each* tenant.[45]
    - **Pros:** Provides the maximum possible isolation and security.[45] A breach in one tenant's silo cannot affect another.
    - **Cons:** Extremely high operational cost, complex provisioning, and does not scale efficiently. This model is not viable for a high-growth, cloud-native "school."
- **Model 2: Logical Isolation (Pool Model)**
    - **Concept:** All tenants share the *same* infrastructure (the same database, the same application APIs).[45] Data isolation is enforced in software.
    - **How it works:** Every data table in the database (e.g., documents, user_profiles) has a tenant_id (or classroom_id) column. Every query executed by the application *must* include a WHERE tenant_id =? clause.
    - **Pros:** Far lower cost, highly efficient, and infinitely scalable.[43] This is the standard model for all modern SaaS.
    - **Cons:** Carries significant security risk. A single programming error (e.g., a developer forgetting to add the tenant_id filter) can lead to a catastrophic cross-tenant data leak.

Recommendation: The **Pool Model** is the only viable path for this platform. The inherent risk is mitigated by *not* relying on application-level developers to remember the tenant_id. Instead, security is enforced centrally by the identity layer and propagated via the token.

B. The Second Dilemma: Authorization Strategy

Given a Pooled Model, the application must know *which tenant* a user's request belongs to.

- **Strategy 1: Application-Based Authorization (Tenant-Agnostic Token)**
    - **Concept:** The IdP issues a "dumb" token containing only a user_id. When a request hits an API, the API must take this user_id, perform a database lookup to find which

tenant(s) the user belongs to, and *then* process the request.[46]
    - **Pros:** Simpler IdP configuration.
    - **Cons:** This is an inefficient and unsecure antipattern. It makes every API call "chattier" (requires an extra database lookup), places the full burden of tenant-logic on the application, and violates the Zero Trust principle (the token itself is not a complete, verifiable credential).
- **Strategy 2: Token-Based Authorization (Tenant-Aware Token)**
    - **Concept:** The IdP is "smart." When a user logs in to a specific "classroom," the IdP issues a JWT that is *tenant-aware*. The token's payload (claims) explicitly contains the tenant_id and the user's role *within that tenant*.[46]
    - **Example JWT Payload:** { "sub": "user_abc", "classroom_id": "cls_123", "roles": ["student"] }
    - **Pros:** This is the modern, secure, and efficient standard.
        1. **Fast & Stateless:** The "PMOVES.AI" microservice can authorize the request *instantly* just by reading the token. No database lookup is needed.[47]
        2. **Secure:** The token is a self-contained, cryptographically-signed "passport" that asserts the user's identity *and* their context within a specific tenant.
    - **Cons:** Requires a more sophisticated IdP capable of handling multi-tenancy logic.

Recommendation: The platform *must* use **Token-Based Authorization**. This architecture centralizes tenant security logic within the IdP and produces self-contained tokens that can be safely "propagated" and verified by any microservice in the stack.

V. Solution Deep Dive: Managed IDaaS (Identity-as-a-Service) - The "Buy" Path

The "Buy" path involves subscribing to a third-party Identity-as-a-Service (IDaaS) provider. This is the fastest, most secure, and most capital-efficient path for launching a new platform, as it offloads the immense complexity of building, securing, and maintaining an identity system.

A. Platform Evaluation: General Purpose

These are the large, established players that serve both B2C and B2B use cases.

1. Auth0 (by Okta)

Auth0 has long been a developer-favorite for its flexibility and excellent documentation.[48]

- **B2B Model - "Organizations":** Auth0 provides a feature called **Organizations** that is *specifically designed* for the "classroom" use case.[14] An "Organization" is a tenant. This feature allows each "classroom" to have:
  - Its own isolated set of users.[52]
  - Its own federated identity providers (e.g., a university's SSO).[14]
  - Custom, per-organization roles (e.g., "teacher," "student").[51]
  - Lightweight branding for its login page.14
    This model is a perfect functional fit for the platform's dual B2C/B2B needs.54
- **Pros:** Extremely powerful, flexible, and provides a clear path for implementing both B2C and B2B user journeys.[56]
- **Cons: Prohibitive Cost.** Auth0's pricing is its primary drawback. It is based almost entirely on **Monthly Active Users (MAUs)**.[2] This model is dangerous for a platform with a large, free B2C "student" base. The platform could see its auth bill skyrocket long before it achieves monetization.[1] Pricing is notoriously complex and subject to "cliffs" that force users into expensive enterprise plans.[2]

2. AWS Cognito

Cognito is the native IDaaS within the Amazon Web Services ecosystem.

- **B2B Model:** Cognito does *not* have an elegant, built-in "Organizations" feature like Auth0. It forces developers into one of two suboptimal multi-tenant patterns [45]:
  1. **Pool-per-Tenant (Silo Model):** Provisioning a new, separate "User Pool" for every "classroom".[45] This provides strong isolation and allows for per-tenant customization (e.g., different password policies or MFA rules).[63] However, it is operationally complex to manage and quickly runs into AWS service quotas.[61]
  2. **Single-Pool (Pool Model):** Placing all users from all "classrooms" into one giant User Pool. Tenant isolation is "faked" by assigning users to "Cognito Groups" or using custom attributes (e.g., custom:classroom_id).[45] This is simpler to manage but provides very weak isolation and places the entire burden of enforcing tenant boundaries on the application, not the IdP.[45]
- **Pros:** Very inexpensive at high B2C scale, making it a viable alternative to Firebase for the

B2C MAU problem.[1] It has deep integration with other AWS services.[50]
- **Cons:** It is widely considered developer-unfriendly, clunky to configure, and lacking the advanced, modern B2B features provided by competitors.[65] The B2B multi-tenancy patterns are workarounds, not first-class features.

B. Platform Evaluation: B2B-Native Specialists

The complexity and cost-prohibitive models of general-purpose IDaaS have created a new market for platforms *purpose-built* for B2B SaaS, which directly maps to the "classroom" model.[3]

1. WorkOS

WorkOS is an API-first platform designed to make an application "enterprise-ready" by providing B2B features as-a-service.[66]

- **B2B Model:** "Organizations" are a first-class architectural primitive, designed to support B2B SSO and SCIM (Directory Sync) provisioning.[66]
- **The Critical Differentiator (Pricing):** WorkOS's pricing model is a hybrid that seems purpose-built for the "POWERFUL MOVES" use case [4]:
  1. **B2C (User Management):** Its "AuthKit" product, which handles B2C login, social auth, and user management, is **free for the first 1,000,000 MAUs**.[4] This completely solves the B2C cost-scaling problem.
  2. **B2B (SSO & Directory Sync):** Its B2B features are priced **per-connection (per-tenant)**, *not* per-user.[4] This provides a highly predictable, linear cost model that scales with the number of B2B clients, not their individual users.
- **Cons:** WorkOS is an API-first "building block." It does *not* provide a pre-built UI for the "teacher's" (delegated admin) portal. This portal must be custom-built by the development team using WorkOS's APIs.[66]

2. Frontegg

Frontegg is another B2B-native platform, but its value proposition is different.

- **The Critical Differentiator (Features):** Frontegg's primary feature is its **pre-built, self-service Admin Portal**.[3] This is a complete UI widget that can be embedded into the application. It provides "teachers" with a ready-made portal to manage their "classroom" users, invite new students, configure their own SSO, and view audit logs.[3]
- **Pros:** Massively accelerates B2B time-to-market by providing the critical UI components that WorkOS and Auth0 require to be custom-built. It is purpose-built for B2B hierarchies.[69]
- **Cons:** Its pricing model is less transparent and appears to be based on a combination of tenants and users, which does not solve the B2C MAU cost problem as cleanly as WorkOS.[70]

C. IDaaS B2B Feature Matrix

Table 2: B2B IDaaS Contender Comparison

Feature	**Auth0**	**WorkOS**	**Frontegg**
B2B Multi-Tenancy Model	**Auth0 Organizations** [14]	**WorkOS Organizations** [67]	**Organizations / Hierarchies** [3]
Pre-Built Delegated Admin Portal	**No.** Must be custom-built with Management APIs.	**No.** Must be custom-built with APIs.[66]	**Yes.** Core feature. A pre-built, embeddable UI.[3]
Built-in SCIM Provisioning	Yes (Enterprise-gated feature).	**Yes.** Core B2B feature.[67]	**Yes.** Core B2B feature.[3]
B2C (Student) Pricing	**MAU-Based.** (High cost at scale).[2]	**Free up to 1,000,000 MAUs.**[4]	MAU-Based. (Free up to 7,500 MAUs).[72]
B2B (Classroom) Pricing	**MAU-Based.** (Unpredictable).[59]	**Per-Connection.** (Predictable).[4]	Per-Tenant / Per-User.[70]

VI. Solution Deep Dive: Open-Source & Self-Hosted - The "Build" Path

The "Build" path involves deploying, managing, and securing a "free" open-source IAM solution. This path offers maximum control but trades subscription fees for high operational and labor costs.

A. Overview & The Total Cost of Ownership (TCO) Fallacy

"Free" open source is a "fallacy"; the software license is free, but the TCO is not.[8] The hidden costs are substantial:

1. **Infrastructure:** Running a highly-available, redundant IAM cluster (3+ VMs, clustered database, reverse proxy) 24/7/365 carries significant cloud hosting costs.[73]
2. **Labor (Expertise):** This is the largest cost. The platform requires senior DevOps/SREs and security engineers to manage, patch, scale, and monitor the IAM system. This is a highly specialized skill set.[7]
3. **Labor (Custom Development):** Many B2B features (like a delegated admin portal) are not included and must be custom-built by the development team, adding to the engineering burden.[6]

A 3-year TCO for a self-hosted Keycloak deployment is estimated to be over **$200,000**, far exceeding the cost of a managed solution.[7]

B. Platform Evaluation: Open-Source

1. Keycloak (Red Hat)

Keycloak is the most dominant, mature, and feature-complete open-source IAM solution.[75]

- **B2B Model (Old): Realm-per-Tenant.** The traditional pattern was to provision a new Keycloak "Realm" for every "classroom".[78]
  - *Pros:* Provides strong isolation and high customization (e.g., per-tenant themes, authentication flows).[80]
  - *Cons:* **Does not scale.** Keycloak's performance severely degrades beyond a few hundred realms, making this pattern unusable for a large-scale SaaS.[79]
- **B2B Model (New): Single-Realm "Organizations".** In recent versions, Keycloak has added native support for **Organizations**, inspired by the Phase Two extension.[81] This allows for scalable multi-tenancy *within* a single realm, where users can be members of different organizations.[83] This is the modern, recommended pattern for Keycloak multi-tenancy.[84]
- **Cons:** Keycloak is a "beast." It is a complex Java monolith with a high memory footprint [5] and is notorious for its difficult documentation and steep learning curve.[6]

2. Ory Stack (Kratos, Keto, Hydra)

The Ory stack is a set of "headless," API-first, cloud-native microservices written in Go: Kratos (identity/login), Keto (authorization/permissions), and Hydra (OAuth/OIDC server).[86]

- **B2B Model:** This is the most critical insight for the Ory stack. The open-source Ory components **do not support B2B multi-tenancy out of the box**.[88] The project's founder has publicly stated, "Ory Kratos will never support multi-tenancy out of the box".[89]
- **Implication:** Multi-tenancy (e.g., tenant-specific SSO, tenant isolation) is a feature of their *paid cloud service*, Ory Network.[90] To use the open-source Ory stack for a multi-tenant application, the development team must *build all the multi-tenancy logic themselves*—this includes managing tenant/user associations, tenant-based authorization policies in Keto, and invitation flows.[89] This is a massive, complex undertaking.
- **Recommendation:** The Ory stack is not a good fit. It is for expert teams who want to build an identity system from its most basic primitives.

3. Zitadel

Zitadel is a modern, cloud-native open-source IAM platform built in Go. It is a direct competitor to Keycloak, designed to solve its complexities.[11]

- **B2B Model:** Zitadel was **built for B2B multi-tenancy from day one**.[9] "Organizations" are a core architectural primitive, not an add-on.[10]

- **Pros:**
  1. **Built-in Delegated Admin:** Zitadel provides a **pre-built, self-service "Console"** for tenants.[21] This allows "teachers" to log in and self-manage their "classroom" users, roles, and SSO settings—a feature that must be custom-built for Keycloak or Ory.[96]
  2. **API-First:** As a modern Go-based application, it is lightweight, API-first, and cloud-native.[9]
  3. **Hybrid Model:** It is fully open-source, but the company also offers a managed cloud service (ZITADEL Cloud).[96] This provides an "off-ramp" from self-hosting, reducing long-term risk.

C. Open-Source IAM Comparison

Table 3: Open-Source "Build" Contender Comparison

Feature	**Keycloak**	**Ory (Open-Source)**	**Zitadel**
Primary Architecture	Java Monolith	Go Microservices (Headless)	Go Microservices (Full Stack)
Native B2B Multi-Tenancy	**Yes** (New "Organizations" feature) [81]	**No.** Must be custom-built by user.[89]	**Yes.** Core architectural primitive.[10]
Pre-Built Delegated Admin Portal	**No.** Must be custom-built.	**No.** (Ory is headless).	**Yes.** Core feature ("Console").[95]
Ease of Use / TCO	High TCO. Complex, "heavyweight".[6]	Very High TCO. "Expert-only," massive custom dev.[92]	Lower TCO. "Cloud-Native," B2B-ready.[11]

VII. Price & Total Cost of Ownership (TCO) Analysis

The financial model for the platform's identity solution is as critical as its technical architecture. A mismatched pricing model will penalize growth.

A. The "Per-User" Trap: MAU-Based Pricing

- **Model:** This model is used by Auth0 [59], Cognito [1], and most traditional IDaaS providers. The platform pays a fee for every user who logs in at least once per month (Monthly Active User).[2]
- **The B2C Trap:** This model is non-viable for a B2C platform that relies on a large, free "student" user base.
  - *Example Scenario:* The platform grows to 100,000 B2C MAUs.
  - *Cognito (New Pricing):* ~$495/month.[1]
  - *Firebase:* ~$275/month.[1]
  - *Auth0 (B2C):* ~$2,000/month (based on B2C Pro pricing).[1]
- **Implication:** Auth0's MAU-based model is ~4x-7x more expensive than its competitors at this scale. It creates a financial "growth penalty" where user success is punished with runaway costs.[2]

B. The B2B-Native Model: Connection-Based Pricing

- **Model:** This hybrid model, pioneered by WorkOS, fundamentally changes the calculation.[4]
  1. **B2C (User Management):** Free for the first 1,000,000 MAUs.[4]
  2. **B2B (SSO/SCIM):** Priced *per-connection* (per-tenant/classroom).[4]
- **The Financial Alignment:** This model *aligns* with the business. It makes the B2C "student" acquisition (a marketing cost) free. It ties the B2B auth cost (a COGS) directly to the number of B2B clients.
- *Example Scenario:* 100,000 B2C MAUs + 50 B2B "Classrooms" (Connections).
  - **WorkOS Cost: $0** (for the 100k MAUs) + (50 connections * $80/ea) = **$4,000/month**.[4]
  - **Auth0 (B2B Plan) Cost:** Auth0's B2B plans do not have public pricing for 100,000 MAUs.[58] This level requires an "Enterprise" contract, which would be *significantly* more expensive than the WorkOS cost, as it would be based on the 100,000 MAUs *in addition* to B2B features.

C. The "Free" Model: Self-Hosted TCO

- **Model:** The cost is 0% licensing and 100% operational (OpEx) and labor (CapEx).
- **Implication:** As detailed in Section VI-A, this "free" model is a multi-year, six-figure investment in infrastructure and specialized engineering talent.[7] This is a strategic error for a startup whose core competency is education and AI, not IAM infrastructure management.

VIII. Provisioning Strategy: Automating the "Classroom" Lifecycle

The query's request to "provision users" refers to user lifecycle management—how users are created, updated, and deleted. This must be automated to be scalable.

A. The Standard: SCIM (System for Cross-domain Identity Management)

- **What it is:** SCIM is an open, REST API protocol (defined in RFCs 7643 and 7644) that standardizes the *automation* of user provisioning.[99]
- **How it works:** It defines a "push" model. The Identity Provider (IdP), such as a B2B customer's Okta or Microsoft Entra ID, acts as the client. The "POWERFUL MOVES" platform acts as the **SCIM Service Provider** (server).[101]
  - **Create:** When a "teacher" at a B2B client adds a new employee to the "Powerful Moves" app in their Okta dashboard, Okta automatically sends an HTTP POST to the platform's /Users endpoint.[102]
  - **Update:** When a user's role changes, Okta sends an HTTP PATCH request.[102]
  - **Delete:** When that employee is fired, Okta sends an HTTP PATCH (to set active: false) or HTTP DELETE request. The user's access is *instantly* and *automatically* revoked.[102]
- **Why it's necessary:** This is a non-negotiable, table-stakes feature for selling to any medium or large enterprise ("classroom"). They demand automated provisioning and de-provisioning for security and compliance.[101]

B. The Alternative: JIT (Just-in-Time) Provisioning

- **What it is:** JIT provisioning is a "pull" model. A user account is created *on the fly* during their first successful SSO login (via OIDC or SAML).[107]
- **Pros:** Simpler to implement than a full SCIM server. It only requires handling the creation logic.[107]
- **Cons:** JIT only handles **creation**. It *cannot* handle **de-provisioning.**[105] If an employee is fired, their JIT-created account still exists and remains active in the platform until manually removed—a massive security hole.

C. Recommendation: A Hybrid Provisioning Model

The platform requires both provisioning strategies for its two different user journeys:

1. **Use JIT Provisioning for B2C "Students":** When a new student signs up for the first time using "Sign in with Google," this is a JIT flow. Their account is created "just-in-time."
2. **Use SCIM Provisioning for B2B "Classrooms":** All B2B tenants must be supported with a full SCIM 2.0 API. This provides the secure, automated lifecycle management that enterprise clients demand.[109]

IX. Implementation Blueprint: Securing the "PMOVES.AI" Stack

This section provides a concrete technical guide for the development team, outlining the implementation of the recommended architecture.

A. Securing the Frontend (React SPA)

1. **Integration:** The development team will install the official React SDK for the chosen IdP

(e.g., @auth0/auth0-react [56], or an equivalent for WorkOS/Zitadel).

2. **Configuration:** The entire React application will be wrapped in the Auth0Provider (or equivalent). This provider is configured with the domain, clientId, and authorizationParams (including redirect_uri and the audience of the API).[56]
3. **Login:** A login component will call the loginWithRedirect() function from the SDK's hook.[57] This will trigger the secure **OIDC Authorization Code + PKCE flow**.[40]
4. **Route Protection:** Application routes (e.g., /dashboard) will be protected by wrapping them in a component that checks the isAuthenticated flag from the SDK's hook.[57]
5. **API Calls:** To call a protected backend API (like "PMOVES.AI"), the component will use the SDK's getAccessTokenSilently() hook. This function retrieves the cached JWT (Access Token) or silently fetches a new one. This token is then added to the Authorization: Bearer <token> header of the fetch or axios request.[56]

B. Securing the Backend APIs (e.g., Node.js / Spring Boot)

The API backend *must* validate the incoming JWT on *every single request* to a protected endpoint.

- **Implementation (Node.js/Express):**
  1. Use a middleware library like express-oauth2-jwt-bearer.
  2. The middleware will be configured with the IdP's issuer URL and the API's audience.[111]
  3. On each request, the middleware will:
     a. Extract the Bearer token from the header.112
     b. Download the IdP's public signing keys (JWKS) from its .well-known/jwks.json endpoint (and cache them).111
     c. Locally verify the token's signature, issuer (iss), audience (aud), and expiration (exp).111
  4. If valid, it will attach the token's decoded payload (the claims) to the req.auth object for use by the route handler.[112]
- **Implementation (Spring Boot):**
  1. Add the spring-boot-starter-oauth2-resourceserver dependency.[113]
  2. In application.properties, configure the spring.security.oauth2.resourceserver.jwt.issuer-uri to point to the IdP's issuer URL.[114]
  3. Spring Security will *automatically* handle all JWT validation (JWKS download, signature/claim verification).[115]
  4. Endpoints can then be secured using method annotations (@PreAuthorize) or a SecurityFilterChain bean.

C. The "Propagation" Pattern: Securing Service-to-Service Communication

A common scenario: a user calls the API Gateway, which then calls PMOVES.AI, which in turn needs to call CATACLYSM_STUDIOS *on behalf of that user*.

- **Antipattern (Do Not Do):** Do not pass the user's original external JWT from service to service. This "token-forwarding" is brittle and insecure.[117]
- **Correct Pattern (M2M with Impersonation):**
  1. The API Gateway validates the user's external JWT. It now *trusts* the user's identity (e.g., user_id: "abc").
  2. The Gateway calls PMOVES.AI. It *does not* send the user's token. Instead, it authenticates itself to PMOVES.AI using a **Machine-to-Machine (M2M)** token (via the Client Credentials grant).[118] It passes the user's identity (user_id: "abc") as part of the request body or a secure header.
  3. PMOVES.AI now needs to call CATACLYSM_STUDIOS. It repeats the process, authenticating itself using *its* M2M token.
- **Alternative (Token Exchange):** An advanced pattern where the API Gateway *exchanges* the user's external JWT for a new, *internally-signed* JWT that contains the validated claims.[26] This internal token is then "propagated" between services.[119] This is more complex but creates a clean internal trust boundary.

D. The "Provisioning" Pattern: Building Your SCIM 2.0 Server

To support B2B "classrooms," the platform must expose a SCIM 2.0-compliant server endpoint.

- **Minimum Endpoints:** The development team must implement the following REST endpoints [120]:
  - POST /scim/v2/Users: Creates a new user. Receives a SCIM User JSON object.[100] Must create the user in the platform's database and return a 201 Created with the full user object.
  - GET /scim/v2/Users/{id}: Retrieves a specific user by their ID.
  - PATCH /scim/v2/Users/{id}: Partially updates a user. This is critical for de-provisioning (e.g., receiving { "active": false }).[104]
  - GET /scim/v2/Users: Lists/queries users, with support for filtering (e.g., ?filter=userName eq "user@example.com").[104]
- **Authentication:** This API is *not* public. It must be protected. The B2B customer's IdP

(e.g., Okta) will authenticate to this endpoint using a long-lived OAuth Bearer Token or API Key that the platform provides.[104]

- **Node.js Tools:** Libraries like scimmy can provide a framework to implement these SCIM-compliant endpoints, simplifying parsing and schema adherence.[123]

E. The "Classroom" Pattern: Tenant-Aware Access Control (RBAC)

Authorization logic must be multi-tenant-aware to prevent data leaks.

- **Bad Logic:** if (user.role == "admin") { // DANGEROUS: This is a global admin }
- **Good Logic (Multi-Tenant RBAC):** if (user.tenant_id == resource.tenant_id && user.roles.includes("teacher")) { // Safe: This user is a teacher *for this specific resource's classroom* }.[124]
- **Implementation:** This logic is simplified by the Token-Based Authorization strategy (Section IV-B). The IdP (Auth0, Zitadel, WorkOS) will be configured to assign roles *per-organization* ("classroom").[124] The resulting JWT will contain claims like:
  JSON
  ```json
  {
   "sub": "user_123",
   "org_id": "classroom_A_uuid",
   "roles": ["teacher"]
  }
  ```

  The API microservice simply reads these claims. The logic becomes clean, stateless, and secure.

F. The Security Foundation (Non-Negotiable)

These are baseline security practices.

- **1. Credential Storage (Hashing):**
  - Passwords must *never* be stored. Only a cryptographic hash shall be stored.[126]
  - **Do not use** MD5 or SHA1/SHA256 for password hashing. They are "fast" hashes designed for data integrity, not password security, and are vulnerable to brute-force attacks.[126]
  - **Must use: Argon2id** (the modern, memory-hard standard and winner of the Password Hashing Competition).[128]

    - **Acceptable alternative: bcrypt** (the long-standing, battle-tested standard).[128]
    - These algorithms are *deliberately slow* and automatically handle "salting," which mitigates rainbow table attacks.[128]
- **2. Data Encryption:**
    - **Data In-Transit:** All network communication *must* be encrypted using **TLS 1.2+ (HTTPS)**. This applies to the public-facing website, all API calls, and—critically—all *internal* service-to-service communication between microservices.[130]
    - **Data At-Rest:** All sensitive user data (PII, credentials, etc.) stored in databases or object storage *must* be encrypted using a strong symmetric algorithm like **AES-256**.[132]
- **3. Multi-Factor Authentication (MFA):**
    - The platform must offer MFA as a security feature.[134]
    - **Prioritize:** Time-based One-Time Passwords (TOTP) (e.g., Google Authenticator) and **FIDO2/WebAuthn** (Passkeys, YubiKey).[134]
    - **Avoid (if possible):** SMS and Email MFA. These methods are vulnerable to interception and phishing.[134]
    - All recommended IDaaS platforms provide robust, out-of-the-box MFA capabilities.[136]

X. Final Architectural Recommendation and Phased Roadmap

A. Final Recommendation

Based on the exhaustive analysis of the platform's dual B2C/B2B requirements, the security demands of a microservice architecture, and the critical need for a financially-sustainable cost model, the following recommendation is made:

Adopt a **"Buy" strategy centered on the WorkOS IDaaS platform.**

This recommendation provides the optimal balance of speed-to-market, security, and financial viability.

- **Solves the B2C Problem:** The "User Management" product is free for the first 1,000,000 MAUs.[4] This de-risks B2C growth and allows the platform to scale its "student" base without incurring punitive authentication costs.

- **Solves the B2B Problem:** The B2B features (SSO, SCIM) are priced on a predictable, per-connection (per-"classroom") model.[4] This aligns costs directly with B2B-generated revenue.
- **Strategic Focus:** This approach allows the POWERFUL MOVES team to focus on its core intellectual property (education, AI) while WorkOS handles the complex, non-differentiating "plumbing" of identity, compliance, and enterprise integrations.
- **The Trade-off:** The development team's effort will be focused on building the (required) custom "Teacher Admin Portal" using WorkOS's modern APIs, which is a superior use of resources than managing the security and infrastructure of a self-hosted "Build" solution.

B. Alternative Recommendation (High-Control)

If absolute control over infrastructure and data, and the avoidance of all vendor lock-in, are the primary business drivers—and the organization is prepared to invest in a dedicated DevOps/SRE team—the following "Build" strategy is recommended:

Adopt the **Zitadel Open-Source Platform (Self-Hosted).**

Zitadel is the superior "Build" choice because it is the only open-source solution that is both cloud-native and purpose-built for B2B multi-tenancy.[11] Its inclusion of a **pre-built, self-service delegated admin console** [95] solves the single largest development burden of the B2B "classroom" model, dramatically accelerating the "Build" timeline compared to Keycloak or Ory.

C. Phased Implementation Roadmap

The following phased roadmap is recommended, regardless of the chosen platform.

- **Phase 1: Foundation (Weeks 1-4)**
  1. **Action:** Finalize the "Build" (Zitadel) vs. "Buy" (WorkOS) decision and procure the service.
  2. **Action:** Configure the first "organization" (the platform's own internal admin tenant).
  3. **Action:** Implement the core OIDC Authorization Code + PKCE flow in the React frontend [56] and the JWT validation in a backend API.[112]
  4. **Milestone:** A single administrator can log in and securely access one protected API endpoint.
- **Phase 2: B2C Launch (Weeks 5-8)**

   1. **Action:** Enable public, self-service sign-up and configure the B2C user journey.
   2. **Action:** Configure social login providers (e.g., Google) for frictionless onboarding.
   3. **Action:** Implement the JIT (Just-in-Time) provisioning flow for these new "student" users.
   4. **Milestone:** The public-facing "school" is live. Individual students can find the site, sign up, and use the B2C features.
- **Phase 3: B2B Launch (Weeks 9-16)**
   1. **Action:** Implement the "Organization" (classroom) model.
   2. **Action:** Build the "Teacher" (Delegated Admin) portal. This UI will use the IdP's APIs to allow a "teacher" to:
      - Invite "students" to their "classroom."
      - Remove "students" from their "classroom."
      - Assign roles (e.g., "student," "co-teacher") *within their classroom*.
   3. **Action:** Implement tenant-aware RBAC in the backend APIs, using the org_id and roles claims from the JWT.
   4. **Milestone:** A "teacher" can be sold a "classroom" license, manage their roster, and their students' data is fully isolated from all other classrooms.
- **Phase 4: Enterprise Readiness (Weeks 17+)**
   1. **Action:** Build the SCIM 2.0 server endpoints (/Users, /Groups) to support automated provisioning.[120]
   2. **Action:** Configure and test inbound B2B SSO, allowing "classroom" clients to log in using their own corporate IdP (via SAML or OIDC).[46]
   3. **Action:** Enforce and roll out MFA options (TOTP, WebAuthn) for all users to enhance platform security.[134]
   4. **Milestone:** The platform is "Enterprise-Ready" and can be sold to large organizations that require automated provisioning and SSO.

Works cited

1. Auth Pricing Wars: Cognito vs Auth0 vs Firebase vs Supabase ..., accessed November 14, 2025, https://zuplo.com/learning-center/api-authentication-pricing
2. Auth0 Pricing Explained (And Why Startups Call It a Growth Penalty) | SSOJet - Enterprise SSO & Identity Solutions, accessed November 14, 2025, https://ssojet.com/blog/auth0-pricing-growth-penalty
3. Compare Frontegg with Leading Solutions | Find the Best Fit, accessed November 14, 2025, https://frontegg.com/compare
4. Pricing — WorkOS, accessed November 14, 2025, https://workos.com/pricing
5. Replacing Keycloak with Ory in k3s. · ory kratos · Discussion #3965 - GitHub, accessed November 14, 2025, https://github.com/ory/kratos/discussions/3965
6. Keycloak or Ory stack : r/devops - Reddit, accessed November 14, 2025, https://www.reddit.com/r/devops/comments/17izuna/keycloak_or_ory_stack/
7. How much does Keycloak cost? - Sirius Open Source, accessed November 14, 2025, https://www.siriusopensource.com/en-us/blog/how-much-does-keycloak-cost

8. Keycloak Price and the Hidden Costs of Free Enterprise Auth Platforms - FusionAuth, accessed November 14, 2025, https://fusionauth.io/blog/migrate-from-keycloak
9. Beyond Authentication: Why Modern Applications Still Need Identity Infrastructure - ZITADEL, accessed November 14, 2025, https://zitadel.com/blog/need-for-identity-infrastructure
10. Simplify Your SaaS: Multi-Tenancy and Delegated Access Management with ZITADEL Organizations - YouTube, accessed November 14, 2025, https://www.youtube.com/watch?v=Cx_WgyY4TOo
11. Zitadel vs Keycloak: Which Is the Future of Cloud-Native Identity in 2025? - House Of FOSS, accessed November 14, 2025, https://www.houseoffoss.com/post/zitadel-vs-keycloak-which-is-the-future-of-cloud-native-identity-in-2025
12. WorkOS Pricing: How It Works and Compares to Datawiza, accessed November 14, 2025, https://www.datawiza.com/blog/industry/workos-pricing-how-it-works-and-compares-to-datawiza/
13. Firebase Authentication, accessed November 14, 2025, https://firebase.google.com/docs/auth
14. Multi-Tenant Applications Best Practices - Auth0, accessed November 14, 2025, https://auth0.com/docs/get-started/auth0-overview/create-tenants/multi-tenant-apps-best-practices
15. Migrating Google Firebase Users Into Auth0, accessed November 14, 2025, https://community.auth0.com/t/migrating-google-firebase-users-into-auth0/186825
16. How to migrate firebase to auth0?, accessed November 14, 2025, https://community.auth0.com/t/how-to-migrate-firebase-to-auth0/72007
17. CIAM VS IAM: Understanding the 9 Key Differences - Infisign, accessed November 14, 2025, https://www.infisign.ai/blog/ciam-vs-iam-understanding-the-key-differences
18. CIAM vs IAM - What You Need to Know - Thales, accessed November 14, 2025, https://cpl.thalesgroup.com/blog/access-management/ciam-vs-iam
19. CIAM vs. Workforce IAM: Key Differences and Considerations - Avatier, accessed November 14, 2025, https://www.avatier.com/blog/ciam-vs-workforce
20. B2B Identity and Access Management - Thales Group, accessed November 14, 2025, https://cpl.thalesgroup.com/access-management/b2b-identity
21. Features - ZITADEL, accessed November 14, 2025, https://zitadel.com/features
22. What is Azure Active Directory B2C? | Microsoft Learn, accessed November 14, 2025, https://learn.microsoft.com/en-us/azure/active-directory-b2c/overview
23. IAM for Microservices & API Security | SecurePass - eMudhra, accessed November 14, 2025, https://emudhra.com/en/blog/iam-for-microservices-api-security-securepass
24. Architecting cloud-native IAM: A microservices-based approach to modern identity management, accessed November 14, 2025, https://journalwjaets.com/sites/default/files/fulltext_pdf/WJAETS-2025-0915.pdf

25. Architecture Patterns in Microservices: Security with JWT - Paradigma Digital, accessed November 14, 2025, https://en.paradigmadigital.com/dev/architecture-patterns-microservices-security-jwt/
26. Authentication and authorization in a microservice architecture: Part 3 - implementing authorization using JWT-based access tokens, accessed November 14, 2025, https://microservices.io/post/architecture/2025/07/22/microservices-authn-authz-part-3-jwt-authorization.html
27. OAuth2 vs OpenID Connect: Auth Protocols Explained - Talent500, accessed November 14, 2025, https://talent500.com/blog/oauth2-vs-openid-connect-auth-protocols/
28. OAuth 2.0 and OpenID Connect overview - Okta Developer, accessed November 14, 2025, https://developer.okta.com/docs/concepts/oauth-openid/
29. How OpenID Connect (OIDC) Works - Ping Identity, accessed November 14, 2025, https://www.pingidentity.com/en/openid-connect.html
30. OAuth 2.0 vs OpenID Connect: Understanding the Differences and Use Cases - Medium, accessed November 14, 2025, https://medium.com/@beno.dev/oauth-2-0-vs-openid-connect-understanding-the-differences-and-use-cases-aea34541589c
31. What Is OpenID Connect (OIDC)? | Microsoft Security, accessed November 14, 2025, https://www.microsoft.com/en-us/security/business/security-101/what-is-openid-connect-oidc
32. How OpenID Connect Works - OpenID Foundation, accessed November 14, 2025, https://openid.net/developers/how-connect-works/
33. What Is OpenID Connect (OIDC)? How Does OIDC Authentication Work? - Fortinet, accessed November 14, 2025, https://www.fortinet.com/resources/cyberglossary/oidc
34. The Difference Between SAML vs OIDC - StrongDM, accessed November 14, 2025, https://www.strongdm.com/blog/oidc-vs-saml
35. OIDC vs. SAML: Understanding the Differences and Upgrading to Modern Authentication, accessed November 14, 2025, https://www.beyondidentity.com/resource/oidc-vs-saml-understanding-the-differences
36. SAML vs OIDC: All You Need to Know - OneLogin, accessed November 14, 2025, https://www.onelogin.com/learn/oidc-vs-saml
37. Single Sign-on VS OIDC based Sign-on - Microsoft Q&A, accessed November 14, 2025, https://learn.microsoft.com/en-us/answers/questions/5540794/single-sign-on-vs-oidc-based-sign-on
38. Understanding OAuth 2.0 Grant Types - A Quick Guide - FusionAuth, accessed November 14, 2025, https://fusionauth.io/blog/understanding-oauth2-grant-types
39. What's the right OAuth 2.0 flow for a mobile app - Stack Overflow, accessed

November 14, 2025, https://stackoverflow.com/questions/17427707/whats-the-right-oauth-2-0-flow-for-a-mobile-app
40. Microsoft identity platform and OAuth 2.0 authorization code flow, accessed November 14, 2025, https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth2-auth-code-flow
41. What is the OAuth 2.0 Authorization Code Grant Type? - Okta Developer, accessed November 14, 2025, https://developer.okta.com/blog/2018/04/10/oauth-authorization-code-grant-type
42. Identity Platform multi-tenancy | Google Cloud Documentation, accessed November 14, 2025, https://docs.cloud.google.com/identity-platform/docs/multi-tenancy
43. Multi-Tenant vs. Single-Tenant IDaaS Solutions - FusionAuth, accessed November 14, 2025, https://fusionauth.io/articles/identity-basics/multi-tenancy-vs-single-tenant-idaas-solutions
44. Multi-Tenant Application. Software Architecture | Update on… | by Sudheer Sandu - Medium, accessed November 14, 2025, https://medium.com/@sudheer.sandu/multi-tenant-application-68c11cc68929
45. SaaS authentication: Identity management with Amazon Cognito ..., accessed November 14, 2025, https://aws.amazon.com/blogs/security/saas-authentication-identity-management-with-amazon-cognito-user-pools/
46. Architectural Approaches for Identity in Multitenant Solutions - Microsoft Learn, accessed November 14, 2025, https://learn.microsoft.com/en-us/azure/architecture/guide/multitenant/approaches/identity
47. How to Manage Multi-tenancy in an IAM System | Curity, accessed November 14, 2025, https://curity.io/blog/manage-multi-tenancy-in-iam-system/
48. 11 Best IDaaS Providers for Reliable Access and Security in 2025 - Infisign, accessed November 14, 2025, https://www.infisign.ai/blog/top-10-idaas-providers-for-2024-a-comprehensive-review
49. The Best Identity and Access Management Providers for 2025 - Solutions Review, accessed November 14, 2025, https://solutionsreview.com/identity-management/best-identity-and-access-management-providers/
50. Auth0 vs Cognito vs Okta vs Firebase vs Userfront Comparison, accessed November 14, 2025, https://userfront.com/blog/auth-landscape
51. Understand How Auth0 Organizations Work, accessed November 14, 2025, https://auth0.com/docs/manage-users/organizations/organizations-overview
52. Add Auth0 Organizations to Your B2B Blazor Web App, accessed November 14, 2025,

https://auth0.com/blog/auth0-organizations-for-b2b-saas-blazor-web-apps/
53. Multiple Organization Architecture - Auth0, accessed November 14, 2025, https://auth0.com/docs/get-started/architecture-scenarios/multiple-organization-architecture
54. User Management Platform Comparison for React: Clerk vs Auth0 vs Firebase (2025), accessed November 14, 2025, https://clerk.com/articles/user-management-platform-comparison-react-clerk-auth0-firebase
55. Authentication (B2B) - Auth0, accessed November 14, 2025, https://auth0.com/docs/get-started/architecture-scenarios/business-to-business/authentication
56. Auth0 React SDK Quickstarts: Add Login to Your React Application, accessed November 14, 2025, https://auth0.com/docs/quickstart/spa/react
57. The Complete Guide to React User Authentication | Auth0, accessed November 14, 2025, https://auth0.com/blog/complete-guide-to-react-user-authentication/
58. Pricing - Auth0, accessed November 14, 2025, https://auth0.com/pricing
59. 2025 Auth0's latest pricing explained and the best Auth0 alternatives - Logto blog, accessed November 14, 2025, https://blog.logto.io/auth0-pricing-explain
60. Auth0 increases price by 300% : r/webdev - Reddit, accessed November 14, 2025, https://www.reddit.com/r/webdev/comments/18d6hcd/auth0_increases_price_by_300/
61. Multi-Tenancy with Cognito: one big user pool or one pool per tenant? - Stack Overflow, accessed November 14, 2025, https://stackoverflow.com/questions/66143647/multi-tenancy-with-cognito-one-big-user-pool-or-one-pool-per-tenant
62. User-pool multi-tenancy best practices - Amazon Cognito, accessed November 14, 2025, https://docs.aws.amazon.com/cognito/latest/developerguide/bp_user-pool-based-multi-tenancy.html
63. Cognito - User pool per tenant, or one user pool for all tenants : r/aws - Reddit, accessed November 14, 2025, https://www.reddit.com/r/aws/comments/gog02b/cognito_user_pool_per_tenant_or_one_user_pool_for/
64. How to use OAuth 2.0 in Amazon Cognito: Learn about the different OAuth 2.0 grants | AWS Security Blog, accessed November 14, 2025, https://aws.amazon.com/blogs/security/how-to-use-oauth-2-0-in-amazon-cognito-learn-about-the-different-oauth-2-0-grants/
65. Does anyone have experience with Auth0 and Cognito prices for thousands of users? What are the best options for an Authorization Service today regarding quality, security, and cost? : r/node - Reddit, accessed November 14, 2025, https://www.reddit.com/r/node/comments/1ddf9xu/does_anyone_have_experience_with_auth0_and/
66. WorkOS vs. Auth0 vs. Frontegg: Which is best?, accessed November 14, 2025, https://workos.com/blog/workos-vs-auth0-vs-frontegg

67. Startups — WorkOS, accessed November 14, 2025, https://workos.com/startups
68. WorkOS vs Clerk, accessed November 14, 2025, https://workos.com/compare/clerk
69. Frontegg vs Auth0 | Auth0 Alternatives, accessed November 14, 2025, https://frontegg.com/frontegg-vs-auth0
70. Frontegg Review 2025: Key Features, Pricing & Alternatives - Infisign, accessed November 14, 2025, https://www.infisign.ai/reviews/frontegg
71. AWS Marketplace: Frontegg, accessed November 14, 2025, https://aws.amazon.com/marketplace/pp/prodview-vizsb6lbsyh2q
72. Frontegg Pricing | CIAM & AgentLink Plans, accessed November 14, 2025, https://frontegg.com/pricing
73. What Is The Cost Of Self Hosting Keycloak? - Skycloak, accessed November 14, 2025, https://skycloak.io/blog/what-is-the-cost-of-self-hosting-keycloak/
74. Keycloak Guide 2024: Pricing, Features, & Limitations - SuperTokens, accessed November 14, 2025, https://supertokens.com/blog/keycloak-pricing
75. Ory vs Keycloak vs SuperTokens, accessed November 14, 2025, https://supertokens.com/blog/ory-vs-keycloak-vs-supertokens
76. Keycloak, accessed November 14, 2025, https://www.keycloak.org/
77. Top Six Open Source Alternatives to Auth0 - Cerbos, accessed November 14, 2025, https://www.cerbos.dev/blog/auth0-alternatives
78. Keycloak multi-tenant architecture - Cloud-IAM, accessed November 14, 2025, https://www.cloud-iam.com/post/keycloak-multi-tenancy/
79. Multi-tenant architectures in Keycloak (realms vs clients vs new organizations) - Reddit, accessed November 14, 2025, https://www.reddit.com/r/KeyCloak/comments/1nka29r/multitenant_architectures_in_keycloak_realms_vs/
80. The Ultimate Best Guide on Keycloak Multi-Tenancy (Part 1) - Skycloak, accessed November 14, 2025, https://skycloak.io/blog/the-ultimate-best-guide-on-keycloak-multi-tenancy-part-1/
81. Server Administration Guide - Keycloak, accessed November 14, 2025, https://www.keycloak.org/docs/latest/server_admin/index.html
82. Keycloak multi tenancy, realms, IdPs best practice - Reddit, accessed November 14, 2025, https://www.reddit.com/r/KeyCloak/comments/1l38shg/keycloak_multi_tenancy_realms_idps_best_practice/
83. Understanding Multi-Tenancy Options in Keycloak - Phase Two, accessed November 14, 2025, https://phasetwo.io/blog/multi-tenancy-options-keycloak/
84. Using Keycloak for Multi-Tenancy With One Realm | by Dale Bingham - Medium, accessed November 14, 2025, https://medium.com/swlh/using-keycloak-for-multi-tenancy-with-one-realm-7be81583ed7b
85. Can multi-tenancy in Keycloak be done within a single realm? - Stack Overflow, accessed November 14, 2025, https://stackoverflow.com/questions/56684168/can-multi-tenancy-in-keycloak-b

e-done-within-a-single-realm
86. Comparisons of identity management solutions | by Hugo - Medium, accessed November 14, 2025, https://sendoh-daten.medium.com/comparisons-of-auth-solutions-e2edbcd9bcfd
87. Compare Auth0 vs. Ory in 2025, accessed November 14, 2025, https://slashdot.org/software/comparison/Auth0-vs-Ory/
88. The Top 7 Ory Kratos Alternatives - Descope, accessed November 14, 2025, https://www.descope.com/blog/post/ory-kratos-alternatives
89. Add support of multiple tenants · Issue #3129 · ory/kratos - GitHub, accessed November 14, 2025, https://github.com/ory/kratos/issues/3129
90. Multi-tenancy at scale · ory kratos · Discussion #2403 - GitHub, accessed November 14, 2025, https://github.com/ory/kratos/discussions/2403
91. Is someone able to link me to what `B2B Multi Tenancy` means Ory Community #ory-network, accessed November 14, 2025, https://archive.ory.sh/t/9626105/is-someone-able-to-link-me-to-what-b2b-multi-tenancy-means-w
92. Hi everyone I m working on a multi tenant SaaS application a Ory Community #ory-selfhosting, accessed November 14, 2025, https://archive.ory.sh/t/29836523/hi-everyone-i-m-working-on-a-multi-tenant-saas-application-a
93. Everything you need to know about Multitenancy - ZITADEL, accessed November 14, 2025, https://zitadel.com/blog/multitenancy
94. ZITADEL - Identity infrastructure, simplified for you. - GitHub, accessed November 14, 2025, https://github.com/zitadel/zitadel
95. Identity Management Use Cases - B2B, Multi-Tenant & API Security ..., accessed November 14, 2025, https://zitadel.com/usecases
96. What makes ZITADEL a great Keycloak alternative, accessed November 14, 2025, https://zitadel-dev.vercel.app/blog/zitadel-vs-keycloak
97. Authentication and authorization in multi-tenancy B2B scenarios | ZITADEL Docs, accessed November 14, 2025, https://zitadel.com/docs/guides/solution-scenarios/b2b
98. ZITADEL - Identity Infrastructure, Simplified, accessed November 14, 2025, https://zitadel.com/
99. What Is SCIM? Meaning and Integration | Microsoft Security, accessed November 14, 2025, https://www.microsoft.com/en-us/security/business/security-101/what-is-scim
100. SCIM: System for Cross-domain Identity Management, accessed November 14, 2025, https://scim.cloud/
101. What Is SCIM Provisioning? How It Works, Benefits, and More | StrongDM, accessed November 14, 2025, https://www.strongdm.com/blog/scim-provisioning
102. Understanding SCIM | Okta Developer, accessed November 14, 2025, https://developer.okta.com/docs/concepts/scim/
103. SCIM Provisioning: A Complete Guide | Zluri, accessed November 14, 2025, https://www.zluri.com/blog/scim-provisioning

104. Tutorial: Develop and plan provisioning for a SCIM endpoint in Microsoft Entra ID, accessed November 14, 2025, https://learn.microsoft.com/en-us/entra/identity/app-provisioning/use-scim-to-provision-users-and-groups
105. SCIM vs JIT: key differences explained - WorkOS, accessed November 14, 2025, https://workos.com/blog/scim-vs-jit-what-s-the-difference
106. Discover SCIM: Simplifying User Provisioning & Identity, accessed November 14, 2025, https://www.pingidentity.com/en/resources/identity-fundamentals/authentication-authorization-protocols/scim.html
107. SCIM vs JIT Provisioning: What is The Difference? | Zluri, accessed November 14, 2025, https://www.zluri.com/blog/scim-vs-jit
108. Difference Between JIT and SCIM Provisioning - JumpCloud, accessed November 14, 2025, https://jumpcloud.com/blog/jit-scim-provisioning-comparison
109. Just In Time vs SCIM - A-Team Chronicles, accessed November 14, 2025, https://www.ateam-oracle.com/post/just-in-time-vs-scim
110. Securing Your React Application with Auth0 | by Aditya Kulkarni | Medium, accessed November 14, 2025, https://medium.com/@kulkarniaditya1997/securing-your-react-application-with-auth0-71b40247728d
111. Control access to HTTP APIs with JWT authorizers in API Gateway - AWS Documentation, accessed November 14, 2025, https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-jwt-authorizer.html
112. OpenID Connect Client with Node.js Express | Curity Identity Server, accessed November 14, 2025, https://curity.io/resources/learn/oidc-node-express/
113. Securing Microservices in Spring Boot | by Nagarjun (Arjun) Nagesh - Medium, accessed November 14, 2025, https://medium.com/@nagarjun_nagesh/securing-microservices-in-spring-boot-b430f2ba1fd6
114. Spring Boot 3.0 - JWT Authentication with Spring Security using MySQL Database, accessed November 14, 2025, https://www.geeksforgeeks.org/springboot/spring-boot-3-0-jwt-authentication-with-spring-security-using-mysql-database/
115. Securing Microservices with Spring Security: Implementing JWT - DEV Community, accessed November 14, 2025, https://dev.to/ayshriv/securing-microservices-with-spring-security-implementing-jwt-38m6
116. Securing Microservices with JWT, Spring Security, and Fault Tolerance Using Spring Cloud Gateway | by A cup of JAVA coffee with NeeSri | Medium, accessed November 14, 2025, https://neesri.medium.com/securing-microservices-with-jwt-spring-security-and-fault-tolerance-using-spring-cloud-gateway-8e5cdb59cd7b
117. Propagating user context between microservices secured with M2M JWT

tokens, accessed November 14, 2025, https://security.stackexchange.com/questions/242881/propagating-user-context-between-microservices-secured-with-m2m-jwt-tokens
118. Securing our Microservices by Authentication and Authorization with JWT, Refresh Tokens and RBAC | by Jose Luis Navarro | Medium, accessed November 14, 2025, https://medium.com/@jose-luis-navarro/securing-our-microservices-through-authentication-and-authorization-with-jwt-refresh-tokens-and-f4439810fce7
119. Secure JWT Propagation in Microservices with Spring Cloud - Edstem Technologies, accessed November 14, 2025, https://www.edstem.com/blog/jwt-tokens-microservice-architectures-spring-cloud/
120. How to build a SCIM 2.0 endpoint: Developer's guide | Scalekit Blog, accessed November 14, 2025, https://www.scalekit.com/blog/build-scim-endpoint
121. Implementing a SCIM API for Your Application: A Comprehensive Guide | SSOJet - Enterprise SSO & Identity Solutions, accessed November 14, 2025, https://ssojet.com/blog/implementing-a-scim-api-for-your-application-a-comprehensive-guide
122. Build your SCIM API service - Okta Developer, accessed November 14, 2025, https://developer.okta.com/docs/guides/scim-provisioning-integration-prepare/main/
123. scimmyjs/scimmy: SCIM m(ade eas)y - SCIM 2.0 library for NodeJS - GitHub, accessed November 14, 2025, https://github.com/scimmyjs/scimmy
124. Multi-tenant Role-based Access Control (RBAC) - Aserto, accessed November 14, 2025, https://www.aserto.com/use-cases/multi-tenant-saas-rbac
125. Example 2: Multi-tenant access control and user-defined RBAC with OPA and Rego, accessed November 14, 2025, https://docs.aws.amazon.com/prescriptive-guidance/latest/saas-multitenant-api-access-authorization/opa-rbac-examples.html
126. c# - Best practice of Hashing passwords - Stack Overflow, accessed November 14, 2025, https://stackoverflow.com/questions/20186354/best-practice-of-hashing-passwords
127. What's the best practice to encrypt password? : r/Hosting - Reddit, accessed November 14, 2025, https://www.reddit.com/r/Hosting/comments/1ovu6ja/whats_the_best_practice_to_encrypt_password/
128. Password Storage - OWASP Cheat Sheet Series, accessed November 14, 2025, https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
129. Argon2 vs bcrypt vs. scrypt: which hashing algorithm is right for you? - Stytch, accessed November 14, 2025, https://stytch.com/blog/argon2-vs-bcrypt-vs-scrypt/
130. Data-at-Rest vs. Data-in-Transit Encryption Explained - Serverion, accessed

November 14, 2025, https://www.serverion.com/uncategorized/data-at-rest-vs-data-in-transit-encryption-explained/

131. Encryption for data-in-transit - Microsoft Service Assurance, accessed November 14, 2025, https://learn.microsoft.com/en-us/compliance/assurance/assurance-encryption-in-transit
132. Data At Rest vs In Data Transit: Encryption & Security Guide - FileCloud, accessed November 14, 2025, https://www.filecloud.com/blog/data-at-rest-vs-transit/
133. Encrypting Data-at-Rest and Data-in-Transit - Logical Separation on AWS, accessed November 14, 2025, https://docs.aws.amazon.com/whitepapers/latest/logical-separation/encrypting-data-at-rest-and--in-transit.html
134. Best Practices for Multi-factor Authentication (MFA) - Corvus Insurance, accessed November 14, 2025, https://www.corvusinsurance.com/blog/multi-factor-authentication-guide
135. AWS Multi-factor authentication in IAM - AWS Identity and Access Management, accessed November 14, 2025, https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa.html
136. Set up multifactor authentication for users - Microsoft 365 admin, accessed November 14, 2025, https://learn.microsoft.com/en-us/microsoft-365/admin/security-and-compliance/set-up-multi-factor-authentication?view=o365-worldwide
137. Deployment considerations for Microsoft Entra multifactor authentication, accessed November 14, 2025, https://learn.microsoft.com/en-us/entra/identity/authentication/howto-mfa-getstarted