AdvanDEB Platform

Development Plan

Phase 0: User Management & Authentication

Version 3.0 - Capability-Based Architecture

AdvanDEB Development Team

December 12, 2025

Abstract

This document presents the comprehensive development plan for the AdvanDEB Platform, an integrated system for biological knowledge management and individual-based modeling (IBM). The platform consists of two main components: the Knowledge Builder for knowledge ingestion and management, and the Modeling Assistant for IBM-based modeling support. This report focuses on Phase 0 implementation: establishing platform-wide authentication, user management, and authorization infrastructure using a simplified capability-based role model.

Contents

1 Executive Summary

1.1 Project Overview

The AdvanDEB Platform is a comprehensive system designed to support biological research through knowledge management and individual-based modeling. The platform integrates two core components:

- **Knowledge Builder (KB)**: A FastAPI + Vue.js system for ingesting, processing, and managing biological knowledge from literature, web sources, and structured data
- **Modeling Assistant (MA)**: A specialized component for knowledge retrieval and reasoning to support individual-based modeling workflows

1.2 Current Development Phase

The project is currently in **Phase 0: User Management & Authentication**. This foundational phase establishes the security, access control, and collaboration infrastructure required for all future platform capabilities.

1.3 Key Architectural Decision: v3.0 Simplification

Following initial design iterations (v1.0 and v2.0), the architecture has been simplified to a capability-based model:

- **From**: 6 distinct roles (Administrator, Knowledge Curator, Knowledge Reviewer, Agent Operator, Data Analyst, Knowledge Explorator)
- **To**: 3 base roles (Administrator, Knowledge Curator, Knowledge Explorator) + 3 optional capabilities (Agent Access, Analytics Access, Reviewer Status)

This simplification eliminates role redundancy while maintaining all functional requirements through a more flexible permission model.

2 System Architecture

2.1 Platform Components

2.1.1 advandeb-shared-utils

A Python package providing shared authentication and authorization utilities, eliminating code duplication between backend components. Contains:

- JWT token generation and validation
- API key management
- Permission checking logic
- User models (Pydantic)
- Audit logging utilities
- Google OAuth integration helpers

2.1.2 Knowledge Builder

Primary Authentication Provider for the platform. Hosts:

- Google OAuth 2.0 endpoints
- User management interface
- Role and capability approval workflows
- Knowledge ingestion and processing
- AI agent framework for automated extraction
- Knowledge graph construction

2.1.3 Modeling Assistant

Shares authentication infrastructure with Knowledge Builder:

- Same JWT tokens valid across both components
- Same user database (MongoDB)
- Knowledge retrieval from KB data
- Scenario building interface
- Model assembly and simulation support

2.2 Authentication Architecture

2.2.1 Single Sign-On (SSO)

Users authenticate once via Google OAuth 2.0 and receive JWT tokens valid across the entire platform. The same credentials provide access to both Knowledge Builder and Modeling Assistant without separate authentication.

2.2.2 Authentication Methods

1. **Google OAuth 2.0**: Primary method for web UI users
2. **API Keys**: For programmatic access, with capability-based scopes
3. **JWT Tokens**: Short-lived access tokens (1 hour) + refresh tokens (30 days)

2.2.3 Platform-Wide User Database

Single MongoDB database stores all user-related data:

- `users` collection - User profiles with base_role and capabilities
- `capability_requests` collection - Base role and capability approval workflows
- `api_keys` collection - API keys valid across entire platform
- `audit_logs` collection - Complete audit trail for all components

3 Role & Permission Model v3.0

3.1 Base Roles

3.1.1 Administrator

Purpose: System-level authority and platform configuration
Permissions:

- Full system access across all components
- User management (approve roles and capabilities)
- System configuration
- Day Zero knowledge seeding
- Override any review decision
- Access all audit logs

3.1.2 Knowledge Curator

Purpose: Content creator and domain expert
Base Permissions:

- Upload documents (single and batch)
- Create facts and stylized facts
- Build knowledge graphs
- Create scenarios and models (in MA)
- Edit own contributions
- View published knowledge

Optional Capabilities (must be requested and approved):

- **Agent Access**: Run AI agents, use custom tools, view agent logs
- **Analytics Access**: Advanced queries, bulk export, API key generation
- **Reviewer Status**: Review queue access, approve/reject knowledge, quality control

3.1.3 Knowledge Explorator

Purpose: Read-only user for browsing knowledge
Permissions:

- Browse and search published knowledge
- View knowledge graphs
- View published models and scenarios (in MA)

- Create private annotations
- Export limited datasets (for personal use)
- Save search queries

3.2 Capability Request Workflow

3.2.1 New User - Base Role Request

1. User signs in with Google
2. Status: `pending_approval`, base_role: `null`
3. User fills role request form (choose Curator or Explorator)
4. Provides affiliation, research area, justification
5. Administrator reviews and approves/rejects
6. User receives email notification
7. Access granted based on approved base role

3.2.2 Existing Curator - Capability Request

1. Curator logs in with base access
2. Profile page shows "Request additional capabilities"
3. User selects desired capabilities:
   - Agent Access
   - Analytics Access
   - Reviewer Status
4. Provides justification for each capability
5. Administrator reviews
6. Capabilities added to user profile
7. User gains new permissions immediately

3.3 Permission Resolution

Permissions are computed based on:

$$\text{User Permissions} = \text{Base Role Permissions} \cup \text{Capability Permissions} \tag{1}$$

Examples:

- Curator (base only): Can create/edit knowledge
- Curator + Agent Access: Can create knowledge AND run agents
- Curator + Analytics Access + Reviewer Status: Can create, export, AND review
- Administrator: Has all permissions regardless of capabilities

4 Current Phase: Phase 0 Development Plan

4.1 Phase Overview

Goal: Establish complete authentication, authorization, and user management infrastructure for the entire AdvanDEB platform.

Deliverable: Fully authenticated platform with 3 base roles + 3 capabilities, unified SSO across KB and MA, Google OAuth integration, API keys, knowledge review workflow, and Day Zero seeding capability.

4.2 Implementation Plan

4.2.1 Stage 1: Foundation

Focus: Backend authentication system

Tasks:

1. Create `advandeb-shared-utils` repository and package structure
2. Implement JWT token generation and validation
3. Implement Google OAuth 2.0 client
4. Create User, CapabilityRequest, APIKey, AuditLog models (Pydantic)
5. Implement permission checking functions (has_base_role, has_capability)
6. Set up MongoDB connection utilities
7. Create audit logging functions
8. Write unit tests for auth utilities
9. Set up CI/CD for shared package

Deliverable: Functional `advandeb-shared-utils` package ready for integration

4.2.2 Stage 2: User Management Backend

Focus: Knowledge Builder backend integration

Tasks:

1. Add `advandeb-shared-utils` dependency to KB backend
2. Create `/auth` router (login, callback, logout, refresh)
3. Create `/users` router (profile, update)
4. Create `/capability-requests` router (create, list, approve/reject)
5. Create `/api-keys` router (generate, list, revoke)
6. Implement AuthMiddleware for all existing routes
7. Implement RateLimiter middleware

8. Implement AuditLogger middleware
9. Create UserService, RoleService, APIKeyService
10. Set up MongoDB collections: users, capability_requests, api_keys, audit_logs
11. Configure Google OAuth credentials
12. Implement email notification system

Deliverable: Fully functional backend authentication and user management

4.2.3 Stage 3: Frontend Integration

Focus: Knowledge Builder frontend
Tasks:

1. Create Login View with Google OAuth button
2. Create Profile View (display user info, base role, capabilities)
3. Create Role Request View (for new users)
4. Create Capability Request View (for existing curators)
5. Create Administrator Dashboard (user list, pending requests)
6. Create API Key Management View (generate, view, revoke)
7. Implement Auth Store (Pinia/Vuex) with token management
8. Add Axios interceptors for JWT token injection
9. Add automatic token refresh logic
10. Update all existing views with permission-based rendering
11. Add error handling for 401/403 responses
12. Implement "Request Access" prompts for insufficient permissions

Deliverable: Complete authenticated frontend with SSO

4.2.4 Stage 4: Review Workflow

Focus: Knowledge validation system
Tasks:

1. Add `status` field to all knowledge entities (facts, stylized_facts, graphs, documents)
2. Implement status transitions: draft → pending_review → published/rejected/changes_requested
3. Create `/reviews` router (queue, approve, reject, request-changes)
4. Create ReviewService with business logic
5. Create Review Queue View (for users with Reviewer Status capability)

6. Add status badges to knowledge list views
7. Implement reviewer assignment logic
8. Add review history tracking
9. Create reviewer dashboard with statistics
10. Add email notifications for review status changes
11. Prevent self-review (users cannot review own contributions)

Deliverable: Functional peer review system for knowledge quality control

4.2.5 Stage 5: Day Zero & Migration

Focus: Initial knowledge seeding and data migration
Tasks:

1. Create Day Zero batch ingestion workflow
2. Add `is_day_zero` flag to knowledge entities
3. Implement admin-only Day Zero creation endpoints
4. Add "Foundational Knowledge" badges in UI
5. Migrate existing 1,300 PDFs from /papers directory
6. Create migration script for legacy data
7. Add attribution metadata to migrated content
8. Auto-approve Day Zero content (skip review)
9. Create Day Zero management dashboard
10. Add bulk tagging for Day Zero content
11. Test and validate all migrated data

Deliverable: Platform seeded with foundational knowledge, legacy data migrated

4.2.6 Stage 6: MA Integration & Polish

Focus: Modeling Assistant authentication and final testing
Tasks:

1. Add `advandeb-shared-utils` dependency to MA backend
2. Implement authentication middleware in MA using shared library
3. Add JWT token validation to all MA routes
4. Implement permission checks for MA-specific operations (create scenarios, run simulations)

5. Update MA frontend to use shared Auth Store
6. Test cross-component authentication (KB → MA with same token)
7. Add component field to audit logs ("knowledge_builder" vs "modeling_assistant")
8. Create comprehensive integration tests
9. Perform security audit (token expiration, permission boundaries, rate limiting)
10. Load testing (authenticate 100+ concurrent users)
11. Write user documentation (authentication guide, capability request guide)
12. Write administrator documentation (user management, approval workflows)
13. Write developer documentation (adding new permissions, extending capabilities)
14. Final bug fixes and polish

Deliverable: Unified platform with complete authentication across KB and MA

5 Technical Implementation Details

5.1 Database Schema

5.1.1 users Collection

```
{
  "_id": ObjectId,
  "google_id": string,          // Unique
  "email": string,
  "name": string,
  "picture_url": string,
  "base_role": string,          // "administrator", "knowledge_curator",
                                //  "knowledge_explorator"
  "capabilities": [string],     // ["agent_access", "analytics_access",
                                //  "reviewer_status"]
  "status": string,             // "active", "suspended", "pending_approval"
  "created_at": datetime,
  "updated_at": datetime,
  "last_login": datetime,
  "login_count": int,
  "metadata": {
    "affiliation": string,
    "research_area": string,
    "orcid": string
  }
}
```

5.1.2 capability_requests Collection

```
{
  "_id": ObjectId,
  "user_id": ObjectId,
  "request_type": string,       // "base_role" or "capability"

  // For base role requests
  "requested_base_role": string,
  "current_base_role": string,

  // For capability requests
  "requested_capabilities": [string],
  "current_capabilities": [string],

  "justification": string,
  "form_data": dict,
  "status": string,             // "pending", "approved", "rejected"
  "created_at": datetime,
  "reviewed_by": ObjectId,
  "reviewed_at": datetime,
  "review_notes": string
}
```

5.1.3 api_keys Collection

```
{
  "_id": ObjectId,
  "user_id": ObjectId,
  "key_hash": string,           // SHA-256 of plain key
  "key_prefix": string,         // "advk_abc12345"
  "name": string,
  "scopes": [string],           // Auto-assigned based on user's
                                // base_role + capabilities
  "status": string,             // "active", "revoked", "expired"
  "created_at": datetime,
  "expires_at": datetime,
  "last_used_at": datetime,
  "rate_limit": {
    "requests_per_minute": int,
    "requests_per_day": int
  }
}
```

5.1.4 audit_logs Collection

```
{
  "_id": ObjectId,
  "user_id": ObjectId,
```

```
  "action": string,              // "create_fact", "approve_knowledge", etc.
  "resource_type": string,       // "fact", "document", "scenario", etc.
  "resource_id": ObjectId,
  "component": string,           // "knowledge_builder" or "modeling_assistant"
  "details": dict,
  "ip_address": string,
  "user_agent": string,
  "auth_method": string,         // "jwt", "api_key"
  "timestamp": datetime
}
```

5.2 API Endpoints

5.2.1 Authentication Routes

- `GET /auth/login` - Redirect to Google OAuth
- `GET /auth/callback` - OAuth callback handler
- `POST /auth/logout` - Invalidate tokens
- `POST /auth/refresh` - Refresh access token

5.2.2 User Management Routes

- `GET /users/me` - Get current user profile
- `PATCH /users/me` - Update profile
- `GET /users` - List users (admin only)
- `GET /users/:id` - Get user details (admin only)

5.2.3 Capability Request Routes

- `POST /capability-requests` - Create new request
- `GET /capability-requests` - List own requests
- `GET /capability-requests/pending` - List pending (admin only)
- `POST /capability-requests/:id/approve` - Approve (admin only)
- `POST /capability-requests/:id/reject` - Reject (admin only)

5.2.4 API Key Routes

- `POST /api-keys` - Generate new key
- `GET /api-keys` - List own keys
- `DELETE /api-keys/:id` - Revoke key

5.2.5 Review Routes

- `GET /reviews/queue` - Pending review items (requires Reviewer Status)
- `POST /reviews/:id/approve` - Approve knowledge
- `POST /reviews/:id/reject` - Reject knowledge
- `POST /reviews/:id/request-changes` - Request changes

5.3 Security Considerations

5.3.1 Token Security

- JWT secret key: 256-bit random value stored in environment
- Access tokens: Short-lived (1 hour) to limit exposure window
- Refresh tokens: Longer-lived (30 days) but stored securely
- Token rotation on refresh to prevent replay attacks
- JTI (JWT ID) for token revocation capability

5.3.2 API Key Security

- SHA-256 hashing before storage (plain key never stored)
- Prefix-based identification (`advk_...`) for quick lookup
- Rate limiting: 200-500 requests/minute depending on capabilities
- Automatic expiration (30-90 days)
- IP whitelist support (optional)

5.3.3 Rate Limiting

- Per-user rate limits based on capabilities
- Redis-backed token bucket algorithm
- Limits: Explorator (100 req/min), Curator (200 req/min), Curator+Analytics (500 req/min)
- Daily limits for bulk operations

5.3.4 Audit Logging

- All write operations logged
- All authentication events logged
- All permission changes logged
- Immutable logs (append-only)
- Retention: 2 years minimum

6 Testing Strategy

6.1 Unit Tests

- All functions in `advandeb-shared-utils`
- JWT generation and validation
- Permission checking logic
- API key hashing and validation
- Target coverage: 90%+

6.2 Integration Tests

- Complete authentication flows (OAuth, JWT, API keys)
- Cross-component authentication (KB → MA)
- Permission enforcement on all protected routes
- Capability request workflow
- Review workflow

6.3 End-to-End Tests

- User registration through approval
- Curator requests capabilities
- Knowledge creation and review
- Cross-component access (create in KB, view in MA)

6.4 Security Testing

- Attempt access without authentication
- Attempt privilege escalation
- Token expiration validation
- Rate limiting enforcement
- SQL injection attempts (though using MongoDB)
- XSS vulnerability testing

7 Deployment Plan

7.1 Development Environment

- Local MongoDB instance
- Local Ollama for LLM functionality
- Google OAuth test credentials
- Mock email service

7.2 Staging Environment

- Hosted MongoDB (Atlas)
- Deployed backend (Docker containers)
- Deployed frontend (Nginx)
- Real Google OAuth credentials (test domain)
- Real email service (SendGrid/Mailgun)

7.3 Production Environment

- Production MongoDB cluster (replica set)
- Redis cluster for rate limiting
- Load-balanced backend servers
- CDN for frontend assets
- SSL/TLS certificates
- Backup and disaster recovery
- Monitoring and alerting

8 Risk Assessment

8.1 Technical Risks

8.1.1 Risk: Complex Permission Logic

Probability: Medium
Impact: High
Mitigation:

- Comprehensive unit tests for permission functions
- Permission matrix documentation
- Code review focus on authorization code

8.1.2 Risk: OAuth Integration Issues

Probability: Low
Impact: High
Mitigation:

- Early integration testing
- Fallback to email/password authentication if needed
- Well-documented OAuth configuration

8.1.3 Risk: Performance Bottlenecks

Probability: Medium
Impact: Medium
Mitigation:

- Load testing during final stages
- Redis caching for frequently accessed data
- Database indexing on user lookups

8.2 Project Risks

8.2.1 Risk: Scope Creep

Probability: Medium
Impact: High
Mitigation:

- Strict adherence to Phase 0 scope
- Feature requests deferred to Phase 1
- Weekly progress reviews

8.2.2 Risk: Project Delays

Probability: Medium
Impact: Medium
Mitigation:

- Regular milestone tracking
- Early identification of blockers
- Flexible resource allocation

9 Success Criteria

Phase 0 will be considered complete when:

1. All authentication methods functional (Google OAuth, JWT, API keys)
2. All 3 base roles + 3 capabilities implemented and tested
3. Capability request workflow functional for both base roles and capabilities
4. Knowledge review workflow operational
5. Existing 1,300 PDFs migrated with Day Zero attribution
6. Cross-component authentication verified (KB $\leftrightarrow$ MA)
7. Complete audit logging operational
8. Security audit passed
9. User documentation complete
10. Administrator documentation complete
11. All unit, integration, and E2E tests passing
12. System handles 100+ concurrent users

10 Next Phases

Following successful completion of Phase 0:

10.1 Phase 1: Knowledge Builder Stabilization

- Harden knowledge CRUD operations with permissions
- Stabilize agent framework with user attribution
- Add comprehensive test coverage
- Implement CI/CD pipeline
- Performance optimization

10.2 Phase 2: Modeling Assistant Prototype

- Finalize MA knowledge integration contracts
- Implement scenario builder with permissions
- Create model assembly interface
- Validate end-to-end flow: knowledge $\rightarrow$ modeling

10.3 Phase 3: Enhanced Integration & UX

- Advanced search tailored for modeling use cases
- Visual knowledge exploration in MA
- Collaboration features (shared scenarios)
- Multi-user contribution tracking

10.4 Phase 4: Extensions & Plugins

- Plugin mechanism for custom tools
- Support for additional modeling paradigms
- Advanced collaboration (workspaces, teams)
- Trust and reputation system

11 Conclusion

Phase 0 represents the critical foundation for the AdvanDEB Platform, establishing security, access control, and collaboration infrastructure. The simplified v3.0 capability-based architecture provides flexibility while reducing complexity compared to previous designs.

Upon completion, the platform will have a robust, production-ready authentication system that scales to support future phases.

The capability-based model allows users to grow their access as their needs evolve, supporting the platform's goal of fostering collaboration while maintaining quality control through the review workflow.

Document Version: 3.0
Date: December 12, 2025
Status: Ready for Implementation
Contact: AdvanDEB Development Team