

Project 4: Complete Authentication & Authorization System

Objective (Why?)	2
Core Requirements (Must-have)	3
Milestone 1: Core Authentication Foundation	4
Milestone 2: Authentication System & JWT Implementation	5
Milestone 3: Role-Based Access Control (RBAC) Implementation	5
Milestone 4: Frontend Authorization & Admin Features	6
Milestone 5: Social Authentication Integration	6
Stretch Goals (Nice-to-have)	7
Deliverables	8
Testing Scenarios	9

Authentication Components (User Identity Verification):

- Standard Login: Username/password authentication with secure validation
- Google SSO: OAuth 2.0 integration with Google Sign-In for social authentication
- Password Security: bcrypt hashing with 12+ rounds for secure password storage
- JWT Token Management: Secure token generation, validation, and refresh mechanisms

- Session Management: User session tracking with secure token storage
- Multi-Provider Support: Facebook and LinkedIn OAuth integration

Authorization Components (Access Control & Permissions):

- User and Admin Roles: Two-tier role system (user/admin) as requested
- Role-Based Access Control (RBAC): Complete RBAC implementation with role validation
- Backend Authorization: Admin-only API endpoints with proper 403 responses
- Frontend Authorization: Conditional UI rendering based on user roles
- Admin Dashboard: Separate administrative interface with management features
- Privilege Protection: Prevention of unauthorized role elevation attacks

Objective (Why?)

Build a complete user authentication and authorization system with registration, login, role-based access control (RBAC), and protected routes in 6 days. This is your transition from Streamlit to modern full-stack development with React and FastAPI. You will practice:

- Authentication & Security: Password hashing, JWT tokens, Google SSO, protected routes
- Authorization Systems: Role-Based Access Control (RBAC) with user/admin roles and conditional UI
- Full-Stack Architecture: FastAPI backend + React frontend separation
- Database Management: PostgreSQL with user data, sessions, roles, and social accounts
- Modern Frontend: React + Vite + Tailwind CSS development
- Social Authentication: OAuth 2.0 integration with Google, Facebook, LinkedIn

Core Requirements (Must-have)

Layer	Authentication Requirements	Authorization Requirements
Backend	FastAPI + PostgreSQL <ul style="list-style-type: none"> • User registration endpoint with validation • Login endpoint with JWT token generation • Password hashing using bcrypt • Protected routes requiring authentication • OAuth 2.0 integration for Google SSO • JWT tokens containing user information 	Role-Based Access Control <ul style="list-style-type: none"> • User roles (user/admin) in database • JWT tokens containing role claims • Admin-only API endpoints with proper authorization • Role-based permissions and middleware • Privilege escalation prevention • Social login with role assignment
Frontend	React + Vite + Tailwind CSS <ul style="list-style-type: none"> • Registration form with validation • Login form with error handling • Google Sign-In button and OAuth flow • JWT token management and storage • Protected route system • User profile management 	Role-Based UI & Navigation <ul style="list-style-type: none"> • Conditional UI rendering based on roles • Admin dashboard accessible only to admin users • Role-based navigation and menu systems • Admin-only features and components • User role display and management

Database	PostgreSQL Schema <ul style="list-style-type: none"> • Users table with authentication data • Password storage (hashed with bcrypt) • Social accounts table for OAuth linking • User sessions/token management • Database migrations for schema evolution 	Role & Permission Storage <ul style="list-style-type: none"> • Role column in users table (user/admin) • Role-based data access patterns • Proper indexing for role-based queries • Role inheritance for social accounts • Session tracking with role information
Security	Authentication Security <ul style="list-style-type: none"> • Password hashing (bcrypt) • JWT token authentication • OAuth 2.0 state validation and CSRF protection • Input validation and sanitization • Secure cookie and session handling 	Authorization Security <ul style="list-style-type: none"> • Role validation in JWT tokens • Protected API endpoints with role checks • Prevention of privilege escalation attacks • Proper 403 Forbidden responses • Frontend authorization with security

Milestone 1: Core Authentication Foundation

Deliverables:

- PostgreSQL database setup with user schema design
- FastAPI backend with SQLAlchemy models and migrations
- React + Vite frontend with Tailwind CSS styling
- Basic API endpoints for user registration and authentication
- Environment configuration and security setup
- JWT token generation and validation system

Review Requirements (Must Pass to Proceed):

- Security Review: Database security, environment setup, input validation
- Architecture Review: Full-stack architecture and component separation
- Code Quality Review: Clean code organization and documentation

Milestone 2: Authentication System & JWT Implementation

Deliverables:

- Complete JWT authentication system with token management
- Password hashing with bcrypt and security best practices
- Protected API routes with middleware implementation
- React authentication context and route protection
- Login/registration forms with comprehensive validation
- Protected dashboard with user profile functionality

Review Requirements (Must Pass to Proceed):

- Security Review: Authentication security, JWT implementation, password handling
- Code Quality Review: Clean authentication patterns and error handling
- Performance Review: Efficient authentication flows and token management

Milestone 3: Role-Based Access Control (RBAC) Implementation

Deliverables:

- Database migration adding role column to users table
- Updated user models with role support (admin, user roles)
- JWT tokens enhanced with role claims
- Role-protected API endpoints (admin-only routes)
- Backend authorization middleware and decorators
- Admin user seeding mechanism for testing

Review Requirements (Must Pass to Proceed):

- Security Review: Authorization security, role validation, privilege escalation prevention

- Architecture Review: Clean RBAC implementation and scalability
- Code Quality Review: Maintainable authorization patterns

Milestone 4: Frontend Authorization & Admin Features

Deliverables:

- Updated React authentication context with role management
- Conditional UI rendering based on user roles
- Admin dashboard accessible only to admin users
- Role-based navigation and menu systems
- Admin-only features and components
- Comprehensive role-based testing scenarios

Review Requirements (Must Pass to Proceed):

- Security Review: Frontend authorization security, UI access control
- User Experience Review: Seamless role-based user experience
- Code Quality Review: Clean conditional rendering and state management

Milestone 5: Social Authentication Integration

Deliverables:

- OAuth 2.0 implementation for third-party authentication
- Integration with specific providers:
 - Google Sign-In implementation
 - Facebook Login integration
 - LinkedIn OAuth authentication
- Social account database models and migrations
- Account linking and profile data synchronization
- Social login buttons in frontend UI
- User profile merging strategy with role preservation

Review Requirements (Must Pass to Proceed):

- Security Review: OAuth implementation, state validation, CSRF protection

- Code Quality Review: Clean provider integration patterns
- User Experience Review: Seamless social login flows and error handling

Milestone 6: Production Readiness & Advanced Features

Deliverables:

- Claude-like dashboard interface with role-appropriate features
- Advanced user management for admin users
- Session management and logout functionality
- Comprehensive error handling and user feedback
- Production deployment preparation and documentation
- Role-based analytics and monitoring capabilities

Review Requirements (Must Pass for Project Completion):

- Architecture Review: Complete full-stack architecture assessment
- Security Review: Production security assessment and penetration testing
- Code Quality Review: Production-ready code quality standards
- Performance Review: Frontend/backend performance optimization
- User Experience Review: Complete user journey testing for all roles

Milestone Progression Rules:

- Cannot advance to next milestone without passing all review requirements
- Flexible timing allows for learning at individual pace
- Quality gates ensure each milestone meets professional standards
- Mentor support available for concept clarification and review failures

Technical Specifications

Stretch Goals (Nice-to-have)

- Advanced RBAC: Implement permission-based control with granular permissions
- Email Verification: Send verification emails for new registrations
- Password Reset: Forgot password functionality with email links
- Two-Factor Authentication: TOTP-based 2FA with role preservation
- User Management UI: Admin interface for user and role management
- Audit Logging: Track role changes and administrative actions
- Dynamic Role Updates: Real-time role changes without re-authentication
- Role Hierarchies: Implement role inheritance (e.g., admin > moderator > user)
- Multi-tenant Support: Organization-based role management

Deliverables

1. GitHub Repository Link (public or invite @mentor)
2. Live Demo with working authentication and authorization
3. COMPLETE_AUTH_DEMO.md - Include:
 - Screenshots of registration and login flows
 - Screenshots of role-based dashboard differences (admin vs user)
 - Screenshots of admin-only features and access controls
 - Screenshots of social login integration
 - Database schema documentation with role relationships
 - API testing examples showing role-based access control
4. Technical_Learnings.md - Include:
 - Authentication vs Authorization concepts learned
 - RBAC implementation insights
 - Security considerations and best practices
 - Social authentication integration challenges

Authentication Security

- Password Security: Bcrypt hashing with minimum 12 rounds
- JWT Security: Secure token generation, expiration, and validation

- Input Validation: Comprehensive server-side validation
- SQL Injection Prevention: Parameterized queries with SQLAlchemy

Authorization Security

- Role Validation: Secure role checking in JWT tokens
- Privilege Escalation Prevention: No unauthorized role elevation
- Access Control: Proper 403 responses for unauthorized access
- Session Management: Secure role-based session handling

General Security

- CORS Configuration: Proper CORS settings for frontend
- Environment Variables: All sensitive data in .env files
- HTTPS Ready: Code prepared for SSL deployment
- Rate Limiting: Protection against brute force attacks

Testing Scenarios

Authentication Testing

- Valid and invalid user registration scenarios
- Login with various credential combinations
- JWT token generation, validation, and expiration
- Social login flows for all providers

Authorization Testing

- Role Assignment: New users get default 'user' role
- Admin Access: Admin users can access protected endpoints
- User Restrictions: Regular users cannot access admin features
- Role Persistence: Roles persist across login sessions
- Privilege Escalation: Attempts to elevate privileges are blocked

Frontend Authorization Testing

- Conditional Rendering: Admin features hidden from regular users
- Route Protection: Admin routes inaccessible to regular users
- Navigation: Role-appropriate menu items and links
- Error Handling: Graceful handling of unauthorized access attempts

Integration Testing

- End-to-end Authentication: Complete user journey from registration to dashboard
- Role-based Workflows: Different user experiences based on roles
- Social Login Integration: Complete OAuth flows with role assignment
- Database Consistency: Role changes reflected across all systems

Quick Start Resources

- FastAPI Security: <https://fastapi.tiangolo.com/tutorial/security/>
- RBAC Concepts:
<https://auth0.com/intro-to-iam/what-is-role-based-access-control-rbac/>
- JWT Best Practices:
<https://auth0.com/blog/a-look-at-the-latest-draft-for-jwt-bcp/>
- SQLAlchemy Migrations:
<https://alembic.sqlalchemy.org/en/latest/tutorial.html>
- React Role-Based Components:
<https://www.robinwieruch.de/react-role-based-authorization/>
- OAuth 2.0 Security: <https://oauth.net/2/>