Module 1: Core Authentication & User Management System

Cloud Document Sharing Platform

Project: Cloud Document Sharing Platform
Module: 1 of 5
Developer: Aryan Srivastav (23BCS10135)

1. Overview & Objectives

Module 1 establishes the foundational security layer for the Cloud Document Sharing Platform, implementing robust user authentication and basic user management capabilities. This module serves as the cornerstone for all subsequent features, ensuring that only authorized users can access the system while providing a seamless user experience.

Primary Objectives:

- **Secure Authentication:** Implement JWT-based authentication with Spring Security
- **User Registration:** Create comprehensive user onboarding with validation
- **Session Management:** Handle secure token storage and refresh mechanisms
- **Role Foundation:** Establish basic user roles for future access control
- **Security Framework:** Set up CORS, CSRF protection, and input validation

Success Criteria:

- Users can successfully register and login
- JWT tokens are properly generated and validated
- Protected routes work correctly on frontend
- Password security meets industry standards
- Database schema supports future scalability

---

2. Technical Architecture & Research

2.1 Backend Architecture (Spring Boot)

Core Dependencies Research:

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-api</artifactId>
        <version>0.12.3</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
</dependencies>Preformatted Text
```

Security Configuration Strategy:

- **BCrypt Password Encoding:** Industry-standard hashing with configurable strength
- **JWT Token Strategy:** Access tokens (15min) + Refresh tokens (7 days)
- **CORS Configuration:** Specific origin allowlisting for Next.js frontend
- **Rate Limiting:** Implement login attempt limitations to prevent brute force

Database Schema Design:

```
CREATE TABLE users (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    role ENUM('USER', 'ADMIN') DEFAULT 'USER',
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    last_login TIMESTAMP NULL
);

CREATE TABLE refresh_tokens (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    token VARCHAR(255) UNIQUE NOT NULL,
    user_id BIGINT NOT NULL,
    expires_at TIMESTAMP NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
)
)
```

2.2 Frontend Architecture (Next.js)

Authentication State Management:

- **Context API:** Global authentication state management
- **Local Storage Strategy:** Secure token storage with httpOnly considerations
- **Automatic Token Refresh:** Background token renewal before expiration
- **Route Protection:** Higher-order components for protected routes

Key Frontend Components Research:

```
javascript
// Core authentication components to develop

├── components/
│   ├── auth/
│   │   ├── LoginForm.jsx
│   │   ├── RegisterForm.jsx
│   │   ├── ProtectedRoute.jsx
│   │   └── AuthProvider.jsx
│   ├── layout/
│   │   ├── Navbar.jsx
│   │   ├── Sidebar.jsx
│   │   └── Layout.jsx
│   └── ui/
│       ├── Input.jsx
│       ├── Button.jsx
│       └── Alert.jsx
```

Form Validation Strategy:

- **Client-Side:** React Hook Form with Zod schema validation
- **Server-Side:** Spring Validation annotations with custom validators
- **Real-time Feedback:** Immediate validation feedback for better UX
- **Error Handling:** Centralized error management with user-friendly messages

3. Implementation Strategy & Research Findings

Backend Development Focus

Project Setup & Security Configuration

- Initialize Spring Boot project with required dependencies
- Configure Spring Security with JWT authentication
- Set up MySQL database connection and basic entity mapping
- Research and implement password encryption strategies

User Management APIs

- Develop user registration endpoint with comprehensive validation
- Implement login endpoint with JWT token generation
- Create token refresh mechanism for seamless user experience
- Build user profile endpoints for basic CRUD operations

Research Insights:

- **JWT vs Session:** JWT chosen for stateless authentication, better for microservices
- **Password Policy:** Minimum 8 characters, mixed case, numbers, special characters
- **Database Indexing:** Email field indexed for faster login queries
- **Security Headers:** Implement security headers (HSTS, X-Frame-Options, CSP)

Frontend Development Focus

Authentication UI Components

- Create responsive login and registration forms
- Implement client-side validation with real-time feedback
- Design consistent UI components following design system
- Set up routing structure with protected route middleware

State Management & Integration

- Implement authentication context for global state management
- Create API service layer for backend communication
- Handle token storage and automatic refresh logic

- Build comprehensive error handling system

Research Findings:

- **Next.js 14 Features:** App router for better performance and SEO
- **Styling Strategy:** Tailwind CSS for rapid development and consistency
- **Form Libraries:** React Hook Form offers better performance than Formik
- **API Integration:** Axios with interceptors for token management

Security & Testing Implementation

Security Implementation

- Implement rate limiting for login attempts
- Add CSRF protection and secure headers
- Create comprehensive input sanitization
- Set up logging for security events

Testing & Documentation

- Write unit tests for authentication logic
- Create integration tests for API endpoints
- Develop comprehensive API documentation
- Conduct security testing and vulnerability assessment

---

4. Risk Assessment & Mitigation Strategies

Technical Risks:

High Priority:

- **Token Security:** Risk of token theft through XSS attacks
    - *Mitigation:* Implement httpOnly cookies, CSP headers, input sanitization
- **Password Security:** Weak password policies leading to breaches
    - *Mitigation:* Enforce strong password requirements, implement account lockout

Medium Priority:

- **Database Security:** SQL injection and data exposure risks
    - *Mitigation:* Use parameterized queries, implement database encryption

- **Session Management:** Token expiration and refresh token security
    - *Mitigation:* Implement secure token rotation, monitor suspicious activity

Implementation Challenges:

- **CORS Configuration:** Complex setup for Next.js and Spring Boot integration
- **Token Storage:** Balancing security and user experience for token management
- **Validation Synchronization:** Keeping client and server validation in sync

---

5. Success Metrics & Testing Strategy

Functional Requirements Testing:

- **Registration Flow:** Users can create accounts with proper validation
- **Login Process:** Successful authentication with token generation
- **Route Protection:** Unauthorized users cannot access protected resources
- **Token Management:** Automatic refresh works without user intervention

Performance Benchmarks:

- **Login Response Time:** < 500ms for authentication requests
- **Database Query Performance:** < 100ms for user lookup operations
- **Frontend Load Time:** < 2s for initial page load
- **Memory Usage:** Efficient state management without memory leaks

Security Validation:

- **Password Strength:** Enforce and validate strong password policies
- **Token Expiration:** Verify proper token lifecycle management
- **Input Validation:** Test against common injection attacks
- **Rate Limiting:** Verify brute force protection works effectively

---

6. Next Steps & Module 2 Preparation

Module 1 Deliverables:

- Fully functional authentication system with JWT implementation

- User registration and login with comprehensive validation
- Protected route system for frontend navigation
- Basic user profile management capabilities
- Comprehensive documentation and testing suite

Preparation for Module 2:

- **Database Schema:** Ensure user table can support file ownership relationships
- **Permission Framework:** Design role system to support future access controls
- **API Structure:** Create consistent API patterns for file management endpoints
- **Security Foundation:** Establish authentication middleware for file operations

Research Areas for Module 2:

- **AWS S3 Integration:** Investigate presigned URLs for secure file uploads
- **File Validation:** Research server-side file type and size validation
- **Upload Progress:** Explore real-time upload progress tracking mechanisms
- **Error Handling:** Design robust error handling for file operations

This foundational module ensures that all subsequent development builds upon a secure, scalable authentication system that meets enterprise security standards while providing an intuitive user experience.