Authentication System - Software Requirements Specification (SRS)

Document Information

- **Version**: 1.0
- **Date**: September 2025
- **Document Type**: Software Requirements Specification
- **System**: User Authentication System

Table of Contents

---

1. Introduction

1.1 Purpose

This document specifies the requirements for a comprehensive user authentication system that provides secure login, logout, and password recovery functionalities. The system will be implemented using React for the frontend and Spring Boot for the backend services.

1.2 Scope

The authentication system will handle:

- User registration and account creation

- Secure user login with JWT token management
- User logout and session termination
- Password reset functionality via email
- Account verification and activation
- Basic user profile management

1.3 Definitions and Acronyms

- **JWT**: JSON Web Token
- **SRS**: Software Requirements Specification
- **API**: Application Programming Interface
- **UI/UX**: User Interface/User Experience
- **RBAC**: Role-Based Access Control

---

2. Overall Description

2.1 Product Perspective

The authentication system serves as a foundational security layer for web applications, providing centralized user management and access control capabilities.

2.2 Product Features

- Secure user registration with email verification
- Multi-factor authentication support
- Password strength validation
- Account lockout protection
- Session management
- Password recovery via email
- User profile management
- Role-based access control

2.3 User Classes

- **End Users**: Individuals who register and use the system
- **Administrators**: System administrators with elevated privileges
- **System Integrators**: Developers integrating with the authentication APIs

3. System Features

3.1 User Registration

Description: Allow new users to create accounts with email verification.

Priority: High

Functional Requirements:

- Users must provide email, password, first name, and last name
- Email addresses must be unique
- Password must meet complexity requirements
- Account activation via email verification link
- Registration form validation

3.2 User Login

Description: Authenticate users and provide secure access tokens.

Priority: High

Functional Requirements:

- Login with email/username and password
- Generate JWT tokens upon successful authentication
- Support "Remember Me" functionality
- Account lockout after failed attempts
- Login attempt logging

3.3 User Logout

Description: Securely terminate user sessions.

Priority: High

Functional Requirements:

- Invalidate JWT tokens
- Clear client-side session data
- Redirect to login page

- Optional logout from all devices

3.4 Password Recovery

Description: Allow users to reset forgotten passwords.

Priority: High

Functional Requirements:

- Password reset request via email
- Secure reset token generation
- Time-limited reset links
- New password validation
- Reset confirmation

---

4. Use Cases

4.1 User Registration Use Case

Use Case ID: UC001 **Use Case Name**: User Registration **Actor**: New User **Precondition**: User has valid email address **Postcondition**: User account created and verification email sent

Main Flow:

1. User navigates to registration page
2. User fills registration form (email, password, first name, last name)
3. System validates input data
4. System checks email uniqueness
5. System creates user account (inactive status)
6. System sends verification email
7. User clicks verification link
8. System activates account
9. User redirected to login page

Alternative Flows:

- A1: Email already exists - Display error message
- A2: Password doesn't meet criteria - Display validation errors

- A3: Verification email not received - Resend verification option

4.2 User Login Use Case

Use Case ID: UC002 **Use Case Name**: User Login **Actor**: Registered User **Precondition**: User has active account **Postcondition**: User authenticated and JWT token issued

Main Flow:

1. User navigates to login page
2. User enters email and password
3. System validates credentials
4. System checks account status (active, locked)
5. System generates JWT token
6. System returns token and user profile
7. User redirected to dashboard

Alternative Flows:

- A1: Invalid credentials - Display error, increment failed attempts
- A2: Account locked - Display lockout message
- A3: Account inactive - Display activation required message

4.3 Password Recovery Use Case

Use Case ID: UC003 **Use Case Name**: Password Recovery **Actor**: Registered User **Precondition**: User has registered email **Postcondition**: Password successfully reset

Main Flow:

1. User clicks "Forgot Password" link
2. User enters email address
3. System validates email exists
4. System generates reset token
5. System sends password reset email
6. User clicks reset link
7. User enters new password
8. System validates new password
9. System updates password

10. User redirected to login with success message

Alternative Flows:

- A1: Email not found - Display generic message (security)
- A2: Reset token expired - Request new reset
- A3: Invalid new password - Display validation errors

4.4 User Logout Use Case

Use Case ID: UC004 **Use Case Name**: User Logout **Actor**: Authenticated User **Precondition**: User is logged in **Postcondition**: User session terminated

Main Flow:

1. User clicks logout button
2. System invalidates JWT token
3. System clears client-side session data
4. User redirected to login page
5. System logs logout event

---

5. Non-Functional Requirements

5.1 Performance Requirements

- Login response time: < 2 seconds
- Registration process: < 5 seconds
- Password reset email delivery: < 1 minute
- System should support 1000+ concurrent users

5.2 Security Requirements

- Passwords hashed using bcrypt (minimum 12 rounds)
- JWT tokens with 24-hour expiration
- HTTPS encryption for all communications
- SQL injection prevention
- XSS protection
- CSRF protection

5.3 Availability Requirements

- System uptime: 99.9%
- Database backup every 24 hours
- Disaster recovery plan

5.4 Usability Requirements

- Responsive design for mobile and desktop
- Accessibility compliance (WCAG 2.1)
- Multi-language support ready
- Clear error messages and validation

---

6. Technical Architecture

6.1 Frontend Technology Stack

- **Framework**: React 18+
- **State Management**: Redux Toolkit or Context API
- **Routing**: React Router
- **HTTP Client**: Axios
- **UI Library**: Material-UI or Tailwind CSS
- **Form Handling**: Formik or React Hook Form
- **Validation**: Yup or Joi

6.2 Backend Technology Stack

- **Framework**: Spring Boot 3.x
- **Security**: Spring Security 6.x
- **Database**: PostgreSQL or MySQL
- **ORM**: Spring Data JPA
- **Authentication**: JWT
- **Email Service**: Spring Mail with SMTP
- **Validation**: Bean Validation (JSR-303)
- **Documentation**: OpenAPI 3 (Swagger)

6.3 Infrastructure

- **Database**: PostgreSQL 15+
- **Cache**: Redis (for session management)
- **Email Service**: SMTP server or cloud service (SendGrid, AWS SES)
- **Logging**: SLF4J with Logback
- **Monitoring**: Actuator endpoints

---

7. Database Schema

7.1 Users Table

```sql
CREATE TABLE users (
    id BIGSERIAL 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,
    is_active BOOLEAN DEFAULT FALSE,
    is_locked BOOLEAN DEFAULT FALSE,
    failed_login_attempts INTEGER DEFAULT 0,
    last_login_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    email_verified_at TIMESTAMP,
    password_changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

7.2 Roles Table

```sql
CREATE TABLE roles (
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(50) UNIQUE NOT NULL,
    description VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

7.3 User Roles Table

```sql
CREATE TABLE user_roles (
    user_id BIGINT REFERENCES users(id) ON DELETE CASCADE,
    role_id BIGINT REFERENCES roles(id) ON DELETE CASCADE,
    assigned_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id, role_id)
);
```

7.4 Password Reset Tokens Table

```sql
CREATE TABLE password_reset_tokens (
    id BIGSERIAL PRIMARY KEY,
    user_id BIGINT REFERENCES users(id) ON DELETE CASCADE,
    token VARCHAR(255) UNIQUE NOT NULL,
    expires_at TIMESTAMP NOT NULL,
    used BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

7.5 Email Verification Tokens Table

```sql
CREATE TABLE email_verification_tokens (
    id BIGSERIAL PRIMARY KEY,
    user_id BIGINT REFERENCES users(id) ON DELETE CASCADE,
    token VARCHAR(255) UNIQUE NOT NULL,
    expires_at TIMESTAMP NOT NULL,
    verified BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

7.6 User Sessions Table

```sql
```

```
CREATE TABLE user_sessions (
    id BIGSERIAL PRIMARY KEY,
    user_id BIGINT REFERENCES users(id) ON DELETE CASCADE,
    jwt_token_id VARCHAR(255) UNIQUE NOT NULL,
    device_info VARCHAR(500),
    ip_address INET,
    expires_at TIMESTAMP NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    last_accessed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

7.7 Audit Logs Table

sql

```
CREATE TABLE audit_logs (
    id BIGSERIAL PRIMARY KEY,
    user_id BIGINT REFERENCES users(id) ON DELETE SET NULL,
    action VARCHAR(100) NOT NULL,
    details JSONB,
    ip_address INET,
    user_agent VARCHAR(500),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

7.8 Database Indexes

sql

```
-- Performance indexes
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_active ON users(is_active);
CREATE INDEX idx_password_reset_tokens_token ON password_reset_tokens(token);
CREATE INDEX idx_password_reset_tokens_expires ON password_reset_tokens(expires_at);
CREATE INDEX idx_email_verification_tokens_token ON email_verification_tokens(token);
CREATE INDEX idx_user_sessions_user_id ON user_sessions(user_id);
CREATE INDEX idx_user_sessions_expires ON user_sessions(expires_at);
CREATE INDEX idx_audit_logs_user_id ON audit_logs(user_id);
CREATE INDEX idx_audit_logs_created_at ON audit_logs(created_at);
```

---

8. API Specifications

8.1 Authentication Endpoints

POST /api/auth/register

Description: Register a new user account

```
json

Request Body:
{
  "email": "user@example.com",
  "password": "SecurePassword123!",
  "firstName": "John",
  "lastName": "Doe"
}

Success Response (201):
{
  "message": "Registration successful. Please check your email for verification.",
  "userId": 12345
}

Error Response (400):
{
  "error": "Validation failed",
  "details": [
    {
      "field": "email",
      "message": "Email already exists"
    }
  ]
}
```

POST /api/auth/login

Description: Authenticate user and return JWT token

```
json
```

Request Body:

```
{
  "email": "user@example.com",
  "password": "SecurePassword123!"
}
```

Success Response (200):

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refreshToken": "refresh_token_string",
  "user": {
    "id": 12345,
    "email": "user@example.com",
    "firstName": "John",
    "lastName": "Doe",
    "roles": ["USER"]
  },
  "expiresIn": 86400
}
```

Error Response (401):

```
{
  "error": "Invalid credentials"
}
```

POST /api/auth/logout

Description: Invalidate user session

json

Headers:
Authorization: Bearer {jwt_token}

Success Response (200):

```
{
  "message": "Logged out successfully"
}
```

POST /api/auth/refresh

Description: Refresh JWT token

```
json

Request Body:
{
  "refreshToken": "refresh_token_string"
}

Success Response (200):
{
  "token": "new_jwt_token",
  "refreshToken": "new_refresh_token",
  "expiresIn": 86400
}
```

8.2 Password Recovery Endpoints

POST /api/auth/forgot-password

Description: Request password reset

```
json

Request Body:
{
  "email": "user@example.com"
}

Success Response (200):
{
  "message": "If the email exists, a password reset link has been sent."
}
```

POST /api/auth/reset-password

Description: Reset password with token

```
json
```

Request Body:

```
{
  "token": "reset_token_string",
  "newPassword": "NewSecurePassword123!"
}
```

Success Response (200):

```
{
  "message": "Password reset successful"
}
```

Error Response (400):

```
{
  "error": "Invalid or expired reset token"
}
```

8.3 Account Management Endpoints

GET /api/auth/verify-email/{token}

Description: Verify email address

json

Success Response (200):

```
{
  "message": "Email verified successfully"
}
```

GET /api/user/profile

Description: Get user profile

json

```
Headers:
Authorization: Bearer {jwt_token}

Success Response (200):
{
  "id": 12345,
  "email": "user@example.com",
  "firstName": "John",
  "lastName": "Doe",
  "emailVerified": true,
  "createdAt": "2025-01-01T00:00:00Z",
  "roles": ["USER"]
}
```

---

9. User Interface Requirements

9.1 Registration Page

Components Required:

- Registration form with fields: email, password, confirm password, first name, last name
- Real-time password strength indicator
- Email format validation
- Terms of service checkbox
- Submit button with loading state
- Link to login page
- Success message component

Validation Rules:

- Email: Valid format, not already registered
- Password: Minimum 8 characters, mixed case, numbers, special characters
- Names: Required, minimum 2 characters

9.2 Login Page

Components Required:

- Login form with email and password fields

- "Remember me" checkbox
- Submit button with loading state
- "Forgot password" link
- Link to registration page
- Error message display
- Social login options (future enhancement)

9.3 Password Recovery Pages

Forgot Password Page:

- Email input field
- Submit button
- Back to login link
- Success/error message display

Reset Password Page:

- New password field
- Confirm password field
- Password strength indicator
- Submit button
- Success message and login redirect

9.4 Dashboard/Profile Page

Components Required:

- User profile information display
- Edit profile functionality
- Change password option
- Active sessions list
- Logout button
- Account settings menu

---

10. Security Requirements

10.1 Password Security

- Minimum 8 characters length
- Must contain uppercase, lowercase, number, and special character
- Hash passwords using bcrypt with salt rounds ≥ 12
- Prevent password reuse (last 5 passwords)
- Force password change after 90 days (configurable)

10.2 Session Management

- JWT tokens with 24-hour expiration
- Refresh token rotation
- Secure HTTP-only cookies for sensitive data
- Session invalidation on logout
- Concurrent session limits per user

10.3 Account Protection

- Account lockout after 5 failed login attempts
- Progressive delays for subsequent failed attempts
- CAPTCHA after multiple failed attempts
- Email notification for suspicious activities
- IP-based rate limiting

10.4 Data Protection

- All API communications over HTTPS
- Input validation and sanitization
- SQL injection prevention using parameterized queries
- XSS protection with Content Security Policy
- CSRF tokens for state-changing operations

10.5 Email Security

- Time-limited tokens (24 hours for reset, 48 hours for verification)
- One-time use tokens

- Secure random token generation
- Email rate limiting to prevent spam

---

11. Testing Requirements

11.1 Unit Testing

- Backend service methods (90%+ coverage)
- Frontend component testing
- Validation logic testing
- Security utility function testing

11.2 Integration Testing

- API endpoint testing
- Database operation testing
- Email service integration testing
- Authentication flow testing

11.3 Security Testing

- Penetration testing for common vulnerabilities
- JWT token validation testing
- SQL injection testing
- XSS protection testing
- Rate limiting testing

11.4 User Acceptance Testing

- Registration flow testing
- Login/logout functionality
- Password recovery process
- Cross-browser compatibility
- Mobile responsiveness

---

12. Deployment Requirements

12.1 Environment Configuration

- Development, staging, and production environments
- Environment-specific configuration files
- Secret management for API keys and passwords
- Database migration scripts

12.2 Monitoring and Logging

- Application performance monitoring
- Error tracking and alerting
- Audit log retention policy
- Security event monitoring
- System health checks

Appendices

Appendix A: Error Codes

- AUTH001: Invalid credentials
- AUTH002: Account locked
- AUTH003: Email not verified
- AUTH004: Token expired
- AUTH005: Token invalid
- AUTH006: Rate limit exceeded

Appendix B: Configuration Parameters

- JWT expiration time: 24 hours (configurable)
- Refresh token expiration: 30 days
- Account lockout threshold: 5 attempts
- Password reset token validity: 24 hours
- Email verification token validity: 48 hours

Appendix C: Third-Party Dependencies

Frontend:

- React, React Router, Axios, Material-UI/Tailwind
- Form libraries, validation libraries

Backend:

- Spring Boot, Spring Security, Spring Data JPA
- JWT libraries, email libraries, validation libraries