NEOX Infinity App - Authentication Module Specification

Complete Authentication & Authorization System

Version: 3.0
Last Updated: 2025-10-28
Status: Implementation Ready
Module: Authentication & Authorization

Table of Contents

---

1. Overview

1.1 Purpose

The Authentication module provides secure user authentication and authorization for the NEOX Infinity App. It handles login, registration, password management, multi-factor authentication, and single sign-on integration.

1.2 Key Features

- Email/password authentication
- Social login (Google, Microsoft)
- Two-factor authentication (2FA)
- Password reset flow
- Email verification
- Session management
- Remember me functionality
- Account lockout protection
- SSO integration (SAML, OAuth 2.0)

1.3 User Roles

- **Global Admin**: Full system access across all tenants
- **Tenant Admin**: Full access within their tenant
- **Manager**: User management within their department
- **User**: Standard access to assigned modules

---

2. Authentication Flow

2.1 Standard Login Flow

```
1. User navigates to /login
2. User enters email + password
3. System validates credentials
4. If 2FA enabled → redirect to /login/verify
5. If 2FA not enabled → create session, redirect to dashboard
6. Store JWT token in httpOnly cookie
7. Store refresh token
```

2.2 SSO Flow (OAuth 2.0)

```
1. User clicks "Sign in with Google/Microsoft"
2. Redirect to OAuth provider
3. User authenticates with provider
4. Provider redirects back with authorization code
5. Exchange code for access token
6. Fetch user profile from provider
7. Check if user exists in system
8. If exists → create session
9. If not exists → create account (if auto-provisioning enabled)
10. Redirect to dashboard
```

2.3 Password Reset Flow

```
1. User clicks "Forgot Password"
2. User enters email
3. System sends reset link via email (valid for 1 hour)
4. User clicks link in email
5. User redirected to /reset-password?token=xxx
6. User enters new password (twice)
7. System validates token and updates password
8. User redirected to login with success message
```

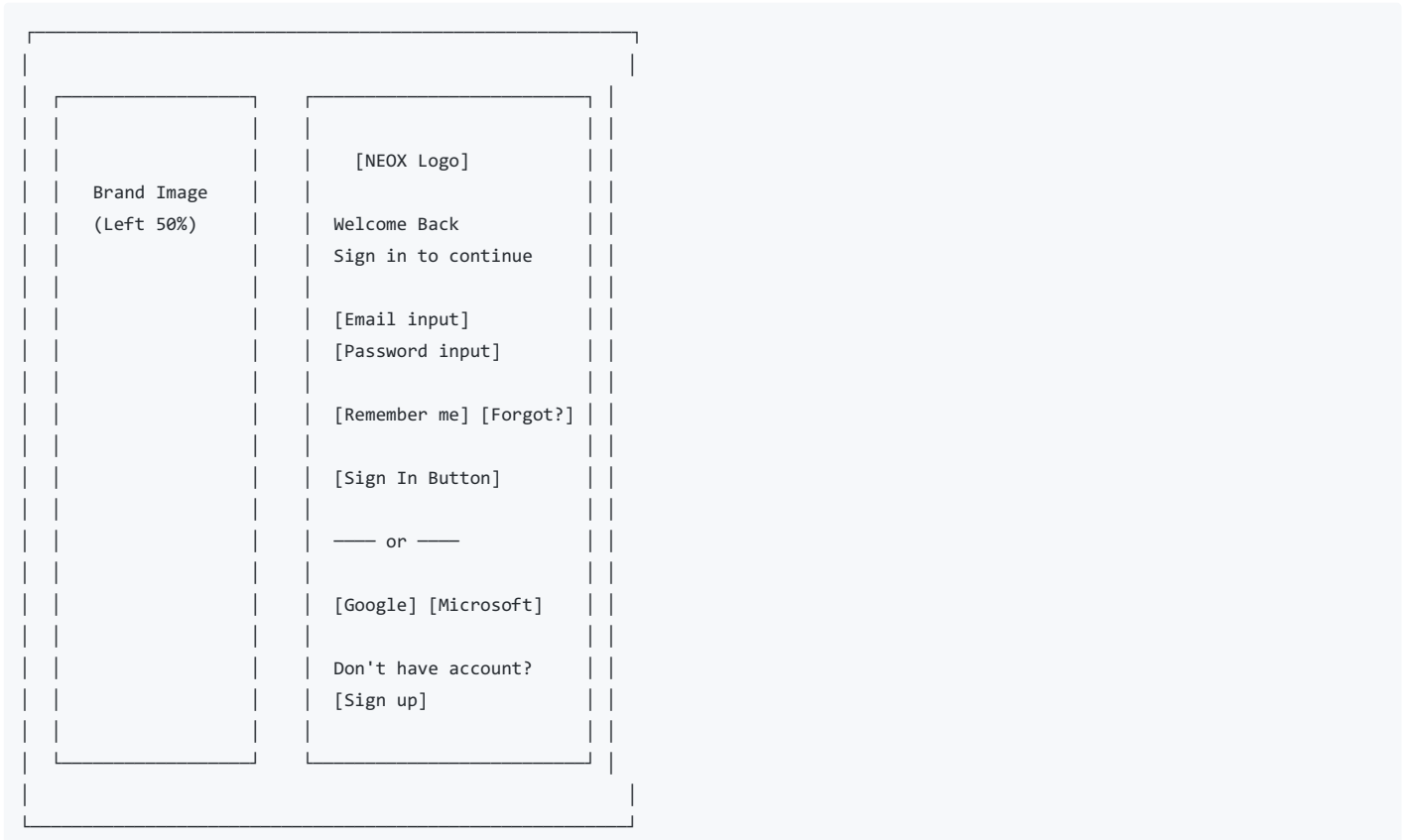
---

3. Screen Specifications

3.1 Login Screen

Route: `/login`
Access: Public (unauthenticated users only)
Redirect: Authenticated users → `/dashboard`

3.1.1 Layout Structure



3.1.2 Component Specifications

Container:

```
<div className="min-h-screen flex">
  {/* Left side - Brand */}
  <div className="hidden lg:flex lg:w-1/2 bg-primary-600 items-center justify-center p-12">
    {/* Brand content */}
  </div>

  {/* Right side - Login Form */}
  <div className="w-full lg:w-1/2 flex items-center justify-center p-8 bg-gray-50">
    {/* Login form */}
  </div>
</div>
```

Logo:

```
<div className="flex justify-center mb-8">
  <img
    src="/logo-primary.svg"
    alt="NEOX Infinity"
    className="h-12"
  />
</div>
```

Heading:

```
<div className="text-center mb-8">
  <h1 className="text-3xl font-bold text-gray-900 mb-2">
    Welcome Back
  </h1>
  <p className="text-sm text-gray-600">
    Sign in to your account to continue
  </p>
</div>
```

Email Input:

```
<div className="space-y-1 mb-4">
  <label
    htmlFor="email"
    className="block text-sm font-medium text-gray-700"
  >
    Email Address
  </label>
  <input
    id="email"
    type="email"
    name="email"
    autoComplete="email"
    required
    value={email}
    onChange={(e) => setEmail(e.target.value)}
    className="
      w-full px-3 py-2
      text-sm text-gray-900
      placeholder:text-gray-400
      bg-white
      border border-gray-300
      rounded-md
      focus:outline-none focus:ring-2 focus:ring-primary-500 focus:border-transparent
      disabled:bg-gray-100 disabled:cursor-not-allowed
    "
    placeholder="you@company.com"
    aria-invalid={errors.email ? 'true' : 'false'}
    aria-describedby={errors.email ? 'email-error' : undefined}
  />
  {errors.email && (
    <p id="email-error" className="text-xs text-error-600 flex items-center gap-1 mt-1">
      <AlertCircle size={12} />
      {errors.email}
    </p>
  )}
</div>
```

Validation Rules - Email:

- Required: "Email is required"
- Format: Must match email regex `/^[^\s@]+@[^\s@]+\.[^\s@]+$/`
- Invalid format: "Please enter a valid email address"
- Max length: 255 characters

Password Input:

```
<div className="space-y-1 mb-4">
  <label
    htmlFor="password"
    className="block text-sm font-medium text-gray-700"
  >
    Password
  </label>
  <div className="relative">
    <input
      id="password"
      type={showPassword ? 'text' : 'password'}
      name="password"
      autoComplete="current-password"
      required
      value={password}
      onChange={(e) => setPassword(e.target.value)}
      className="
        w-full px-3 py-2 pr-10
        text-sm text-gray-900
        placeholder:text-gray-400
        bg-white
        border border-gray-300
        rounded-md
        focus:outline-none focus:ring-2 focus:ring-primary-500 focus:border-transparent
      "
      placeholder="Enter your password"
      aria-invalid={errors.password ? 'true' : 'false'}
      aria-describedby={errors.password ? 'password-error' : undefined}
    />
    <button
      type="button"
      onClick={() => setShowPassword(!showPassword)}
      className="absolute right-3 top-1/2 -translate-y-1/2 text-gray-400 hover:text-gray-600"
      aria-label={showPassword ? 'Hide password' : 'Show password'}
    >
      {showPassword ? <EyeOff size={16} /> : <Eye size={16} />}
    </button>
  </div>
  {errors.password && (
    <p id="password-error" className="text-xs text-error-600 flex items-center gap-1 mt-1">
      <AlertCircle size={12} />
      {errors.password}
    </p>
  )}
</div>
```

Validation Rules - Password:

- Required: "Password is required"
- Min length: 8 characters (enforced on registration, not login)
- No max length validation on login (allow existing passwords)

Remember Me & Forgot Password:

```
<div className="flex items-center justify-between mb-6">
  <label className="flex items-center gap-2 cursor-pointer">
    <input
      type="checkbox"
      checked={rememberMe}
      onChange={(e) => setRememberMe(e.target.checked)}
      className="
        w-4 h-4
        text-primary-600
        bg-white
        border-gray-300
        rounded
        focus:ring-2 focus:ring-primary-500 focus:ring-offset-0
      "
    />
    <span className="text-sm text-gray-700">Remember me</span>
  </label>

  <a
    href="/forgot-password"
    className="text-sm text-primary-600 hover:text-primary-700 font-medium"
  >
    Forgot password?
  </a>
</div>
```

Sign In Button:

```
<button
  type="submit"
  disabled={isLoading}
  className="
    w-full px-4 py-2.5
    bg-primary-600 hover:bg-primary-700 active:bg-primary-800
    text-white font-medium text-sm
    rounded-md
    shadow-sm hover:shadow-md
    transition-all duration-200
    disabled:opacity-50 disabled:cursor-not-allowed
    focus:outline-none focus:ring-2 focus:ring-primary-500 focus:ring-offset-2
    flex items-center justify-center gap-2
  "
>
  {isLoading ? (
    <>
      <Loader size={16} className="animate-spin" />
      <span>Signing in...</span>
    </>
  ) : (
    <span>Sign In</span>
  )}
</button>
```

Divider:

```
<div className="relative my-6">
  <div className="absolute inset-0 flex items-center">
    <div className="w-full border-t border-gray-300"></div>
  </div>
  <div className="relative flex justify-center text-sm">
    <span className="px-4 bg-gray-50 text-gray-500">or continue with</span>
  </div>
</div>
```

Social Login Buttons:

```
<div className="grid grid-cols-2 gap-3">
  {/* Google */}
  <button
    type="button"
    onClick={handleGoogleLogin}
    className="
      flex items-center justify-center gap-2
      px-4 py-2.5
      bg-white hover:bg-gray-50 active:bg-gray-100
      text-gray-700 font-medium text-sm
      border border-gray-300
      rounded-md
      shadow-sm
      transition-all duration-200
      focus:outline-none focus:ring-2 focus:ring-primary-500 focus:ring-offset-2
    "
  >
    <svg className="w-5 h-5" viewBox="0 0 24 24">
      <path fill="#4285F4" d="M22.56 12.25c0-.78-.07-1.53-.2-2.25H12v4.26h5.92c-.26 1.37-1.04 2.53-2.21 3.31v2.77h3.57c2.08-1.92 3.2
      <path fill="#34A853" d="M12 23c2.97 0 5.46-.98 7.28-2.66l-3.57-2.77c-.98.66-2.23 1.06-3.71 1.06-2.86 0-5.29-1.93-6.16-4.53H2.1
      <path fill="#FBBC05" d="M5.84 14.09c-.22-.66-.35-1.36-.35-2.09s.13-1.43.35-2.09V7.07H2.18C1.43 8.55 1 10.22 1 12s.43 3.45 1.18
      <path fill="#EA4335" d="M12 5.38c1.62 0 3.06.56 4.21 1.64l3.15-3.15C17.45 2.09 14.97 1 12 1 7.7 1 3.99 3.47 2.18 7.07l3.66 2.8
    </svg>
    <span>Google</span>
  </button>

  {/* Microsoft */}
  <button
    type="button"
    onClick={handleMicrosoftLogin}
    className="
      flex items-center justify-center gap-2
      px-4 py-2.5
      bg-white hover:bg-gray-50 active:bg-gray-100
      text-gray-700 font-medium text-sm
      border border-gray-300
      rounded-md
      shadow-sm
      transition-all duration-200
      focus:outline-none focus:ring-2 focus:ring-primary-500 focus:ring-offset-2
    "
  >
    <svg className="w-5 h-5" viewBox="0 0 23 23">
      <path fill="#f3f3f3" d="M0 0h23v23H0z"/>
      <path fill="#f35325" d="M1 1h10v10H1z"/>
      <path fill="#81bc06" d="M12 1h10v10H12z"/>
      <path fill="#05a6f0" d="M1 12h10v10H1z"/>
      <path fill="#ffba08" d="M12 12h10v10H12z"/>
    </svg>
    <span>Microsoft</span>
  </button>
</div>
```

Sign Up Link:

```
<p className="mt-6 text-center text-sm text-gray-600">
  Don't have an account?{' '}
  <a
    href="/register"
    className="font-medium text-primary-600 hover:text-primary-700"
  >
    Sign up
  </a>
</p>
```

3.1.3 States & Interactions

Initial State:

- All fields empty
- Sign In button enabled
- No error messages
- Remember me unchecked by default

Loading State:

```
// During API call
- Sign In button disabled
- Button shows spinner + "Signing in..."
- All inputs disabled
- Social login buttons disabled
```

Error States:

Invalid Credentials (401):

```
<div className="mb-4 p-4 bg-error-50 border border-error-200 rounded-lg flex items-start gap-3">
  <XCircle className="text-error-600 flex-shrink-0" size={20} />
  <div>
    <h4 className="text-sm font-medium text-error-900">Invalid credentials</h4>
    <p className="mt-1 text-sm text-error-700">
      The email or password you entered is incorrect. Please try again.
    </p>
  </div>
</div>
```

Account Locked (423):

```
<div className="mb-4 p-4 bg-warning-50 border border-warning-200 rounded-lg flex items-start gap-3">
  <AlertTriangle className="text-warning-600 flex-shrink-0" size={20} />
  <div>
    <h4 className="text-sm font-medium text-warning-900">Account locked</h4>
    <p className="mt-1 text-sm text-warning-700">
      Your account has been locked due to multiple failed login attempts.
      Please try again in 15 minutes or <a href="/forgot-password" className="underline">reset your password</a>.
    </p>
  </div>
</div>
```

Email Not Verified (403):

```
<div className="mb-4 p-4 bg-info-50 border border-info-200 rounded-lg flex items-start gap-3">
  <Info className="text-info-600 flex-shrink-0" size={20} />
  <div>
    <h4 className="text-sm font-medium text-info-900">Email not verified</h4>
    <p className="mt-1 text-sm text-info-700">
      Please verify your email address to continue. Didn't receive the email?
      <button className="underline ml-1" onClick={resendVerification}>Resend verification email</button>
    </p>
  </div>
</div>
```

Network Error:

```
<div className="mb-4 p-4 bg-error-50 border border-error-200 rounded-lg flex items-start gap-3">
  <XCircle className="text-error-600 flex-shrink-0" size={20} />
  <div>
    <h4 className="text-sm font-medium text-error-900">Connection error</h4>
    <p className="mt-1 text-sm text-error-700">
      Unable to connect to the server. Please check your internet connection and try again.
    </p>
  </div>
</div>
```

Success State:

```
// After successful login, before redirect
- Show success toast (auto-dismiss in 2 seconds)
- Redirect to /dashboard (or returnUrl if present)

<div className="fixed top-4 right-4 z-50 p-4 bg-success-50 border border-success-200 rounded-lg shadow-lg flex items-center gap-3 an
  <CheckCircle className="text-success-600" size={20} />
  <div>
    <h4 className="text-sm font-medium text-success-900">Login successful</h4>
    <p className="text-sm text-success-700">Redirecting to dashboard...</p>
  </div>
</div>
```

3.1.4 Form Validation

Client-Side Validation:

```
const validateLogin = (email: string, password: string) => {
  const errors: { email?: string; password?: string } = {};

  // Email validation
  if (!email) {
    errors.email = 'Email is required';
  } else if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {
    errors.email = 'Please enter a valid email address';
  } else if (email.length > 255) {
    errors.email = 'Email must be less than 255 characters';
  }

  // Password validation
  if (!password) {
    errors.password = 'Password is required';
  }

  return errors;
};
```

Validation Timing:

- On blur: Validate individual field
- On submit: Validate all fields
- On change (after first blur): Real-time validation

3.1.5 Responsive Behavior

Mobile (< 768px):

- Hide left brand panel
- Full-width login form
- Reduce padding: `p-4` instead of `p-8`
- Stack social login buttons vertically

```
<div className="grid grid-cols-1 md:grid-cols-2 gap-3">
  {/* Social buttons */}
</div>
```

Tablet (768px - 1023px):

- Show left brand panel (50% width)
- Maintain two-column layout

Desktop (≥ 1024px):

- Full two-column layout
- Maximum form width: 400px

3.1.6 Accessibility

ARIA Labels:

```
<form
  onSubmit={handleSubmit}
  aria-label="Sign in form"
  noValidate
>
  {/* Form fields */}
</form>
```

Keyboard Navigation:

- Tab order: Email → Password → Remember me → Forgot password → Sign In → Google → Microsoft → Sign up
- Enter key submits form from any input field
- Escape key clears form (optional)

Screen Reader Announcements:

```
<div
  role="status"
  aria-live="polite"
  aria-atomic="true"
  className="sr-only"
>
  {errorMessage}
</div>
```

Focus Management:

- On page load: Focus email input
- On error: Focus first invalid field
- On success: No focus change (redirecting)

3.2 Registration Screen

Route: `/register`
Access: Public (unauthenticated users only)

3.2.1 Layout Structure

```
┌──────────────────────────────────────┐
│                                      │
│          [NEOX Logo]                 │
│                                      │
│      Create Your Account             │
│      Get started with NEOX           │
│                                      │
│      [First Name]  [Last Name]       │
│      [Email]                         │
│      [Password]                      │
│      [Confirm Password]              │
│                                      │
│      [Company Name]                  │
│      [Company Size dropdown]         │
│                                      │
│      [✓] I agree to Terms of Service │
│                                      │
│      [Create Account Button]         │
│                                      │
│      ──── or ────                    │
│                                      │
│      [Google]  [Microsoft]           │
│                                      │
│      Already have an account?        │
│      [Sign in]                       │
│                                      │
└──────────────────────────────────────┘
```

3.2.2 Component Specifications

Form Container:

```
<div className="min-h-screen flex items-center justify-center p-4 bg-gray-50">
  <div className="w-full max-w-md">
    {/* Logo */}
    <div className="flex justify-center mb-8">
      <img src="/logo-primary.svg" alt="NEOX Infinity" className="h-12" />
    </div>

    {/* Form Card */}
    <div className="bg-white border border-gray-200 rounded-lg shadow-sm p-8">
      {/* Form content */}
    </div>
  </div>
</div>
```

Heading:

```
<div className="text-center mb-8">
  <h1 className="text-3xl font-bold text-gray-900 mb-2">
    Create Your Account
  </h1>
  <p className="text-sm text-gray-600">
    Get started with NEOX Infinity in seconds
  </p>
</div>
```

Name Fields (Side by Side):

```
<div className="grid grid-cols-2 gap-4 mb-4">
  {/* First Name */}
  <div className="space-y-1">
    <label htmlFor="firstName" className="block text-sm font-medium text-gray-700">
      First Name
    </label>
    <input
      id="firstName"
      type="text"
      name="firstName"
      autoComplete="given-name"
      required
      value={firstName}
      onChange={(e) => setFirstName(e.target.value)}
      className="w-full px-3 py-2 text-sm border border-gray-300 rounded-md focus:ring-2 focus:ring-primary-500"
      placeholder="John"
    />
    {errors.firstName && (
      <p className="text-xs text-error-600">{errors.firstName}</p>
    )}
  </div>

  {/* Last Name */}
  <div className="space-y-1">
    <label htmlFor="lastName" className="block text-sm font-medium text-gray-700">
      Last Name
    </label>
    <input
      id="lastName"
      type="text"
      name="lastName"
      autoComplete="family-name"
      required
      value={lastName}
      onChange={(e) => setLastName(e.target.value)}
      className="w-full px-3 py-2 text-sm border border-gray-300 rounded-md focus:ring-2 focus:ring-primary-500"
      placeholder="Doe"
    />
    {errors.lastName && (
      <p className="text-xs text-error-600">{errors.lastName}</p>
    )}
  </div>
</div>
```

Validation Rules - Name Fields:

- Required: "First name is required" / "Last name is required"
- Min length: 2 characters
- Max length: 50 characters
- Pattern: Letters, spaces, hyphens, apostrophes only
- Error: "Name can only contain letters, spaces, hyphens, and apostrophes"

Email Input:

```
<div className="space-y-1 mb-4">
  <label htmlFor="email" className="block text-sm font-medium text-gray-700">
    Email Address
  </label>
  <input
    id="email"
    type="email"
    name="email"
    autoComplete="email"
    required
    value={email}
    onChange={(e) => setEmail(e.target.value)}
    onBlur={checkEmailAvailability}
    className="w-full px-3 py-2 text-sm border border-gray-300 rounded-md focus:ring-2 focus:ring-primary-500"
    placeholder="you@company.com"
  />
  {checkingEmail && (
    <p className="text-xs text-gray-500 flex items-center gap-1">
      <Loader size={12} className="animate-spin" />
      Checking availability...
    </p>
  )}
  {errors.email && (
    <p className="text-xs text-error-600 flex items-center gap-1">
      <AlertCircle size={12} />
      {errors.email}
    </p>
  )}
  {emailAvailable && !errors.email && email && (
    <p className="text-xs text-success-600 flex items-center gap-1">
      <CheckCircle size={12} />
      Email is available
    </p>
  )}
</div>
```

Validation Rules - Email:

- Required: "Email is required"
- Format: Valid email format
- Max length: 255 characters
- Uniqueness check (async): "This email is already registered"
- Debounce check: 500ms after user stops typing

Password Input with Strength Indicator:

```
<div className="space-y-1 mb-4">
  <label htmlFor="password" className="block text-sm font-medium text-gray-700">
    Password
  </label>
  <div className="relative">
    <input
      id="password"
      type={showPassword ? 'text' : 'password'}
      name="password"
      autoComplete="new-password"
      required
      value={password}
      onChange={(e) => {
        setPassword(e.target.value);
        calculatePasswordStrength(e.target.value);
      }}
      className="w-full px-3 py-2 pr-10 text-sm border border-gray-300 rounded-md focus:ring-2 focus:ring-primary-500"
      placeholder="Create a strong password"
    />
    <button
      type="button"
```

```
      onClick={() => setShowPassword(!showPassword)}
      className="absolute right-3 top-1/2 -translate-y-1/2 text-gray-400 hover:text-gray-600"
    >
      {showPassword ? <EyeOff size={16} /> : <Eye size={16} />}
    </button>
  </div>

  {/* Password Strength Indicator */}
  {password && (
    <div className="space-y-2">
      <div className="flex gap-1">
        <div className={`h-1 flex-1 rounded ${
          passwordStrength >= 1 ?
            passwordStrength === 1 ? 'bg-error-500' :
            passwordStrength === 2 ? 'bg-warning-500' :
            passwordStrength === 3 ? 'bg-success-400' :
            'bg-success-600'
          : 'bg-gray-200'
        }`}></div>
        <div className={`h-1 flex-1 rounded ${passwordStrength >= 2 ? 'bg-warning-500' : 'bg-gray-200'}`}></div>
        <div className={`h-1 flex-1 rounded ${passwordStrength >= 3 ? 'bg-success-400' : 'bg-gray-200'}`}></div>
        <div className={`h-1 flex-1 rounded ${passwordStrength >= 4 ? 'bg-success-600' : 'bg-gray-200'}`}></div>
      </div>
      <p className="text-xs text-gray-600">
        Password strength: {
          passwordStrength === 1 ? 'Weak' :
          passwordStrength === 2 ? 'Fair' :
          passwordStrength === 3 ? 'Good' :
          passwordStrength === 4 ? 'Strong' : ''
        }
      </p>
    </div>
  )}

  {/* Password Requirements */}
  <div className="mt-2 space-y-1">
    <p className="text-xs text-gray-600">Password must contain:</p>
    <div className="space-y-1">
      {[
        { met: password.length >= 8, text: 'At least 8 characters' },
        { met: /[A-Z]/.test(password), text: 'One uppercase letter' },
        { met: /[a-z]/.test(password), text: 'One lowercase letter' },
        { met: /\d/.test(password), text: 'One number' },
        { met: /[!@#$%^&*]/.test(password), text: 'One special character (!@#$%^&*)' },
      ].map((req, idx) => (
        <div key={idx} className="flex items-center gap-2">
          {req.met ? (
            <CheckCircle size={14} className="text-success-600" />
          ) : (
            <XCircle size={14} className="text-gray-300" />
          )}
          <span className={`text-xs ${req.met ? 'text-success-700' : 'text-gray-500'}`}>
            {req.text}
          </span>
        </div>
      ))}
    </div>
  </div>
</div>
```

Password Strength Calculation:

```
const calculatePasswordStrength = (pwd: string): number => {
  let strength = 0;

  if (pwd.length >= 8) strength++;
  if (pwd.length >= 12) strength++;
  if (/[A-Z]/.test(pwd) && /[a-z]/.test(pwd)) strength++;
  if (/\d/.test(pwd)) strength++;
  if (/[!@#$%^&*(),.?":{}|<>]/.test(pwd)) strength++;

  return Math.min(strength, 4);
};
```

Validation Rules - Password:

- Required: "Password is required"
- Min length: 8 characters
- Must contain uppercase letter
- Must contain lowercase letter
- Must contain number
- Must contain special character
- Max length: 128 characters

Confirm Password Input:

```
<div className="space-y-1 mb-4">
  <label htmlFor="confirmPassword" className="block text-sm font-medium text-gray-700">
    Confirm Password
  </label>
  <div className="relative">
    <input
      id="confirmPassword"
      type={showConfirmPassword ? 'text' : 'password'}
      name="confirmPassword"
      autoComplete="new-password"
      required
      value={confirmPassword}
      onChange={(e) => setConfirmPassword(e.target.value)}
      className="w-full px-3 py-2 pr-10 text-sm border border-gray-300 rounded-md focus:ring-2 focus:ring-primary-500"
      placeholder="Re-enter your password"
    />
    <button
      type="button"
      onClick={() => setShowConfirmPassword(!showConfirmPassword)}
      className="absolute right-3 top-1/2 -translate-y-1/2 text-gray-400 hover:text-gray-600"
    >
      {showConfirmPassword ? <EyeOff size={16} /> : <Eye size={16} />}
    </button>
  </div>
  {errors.confirmPassword && (
    <p className="text-xs text-error-600 flex items-center gap-1">
      <AlertCircle size={12} />
      {errors.confirmPassword}
    </p>
  )}
  {confirmPassword && password === confirmPassword && (
    <p className="text-xs text-success-600 flex items-center gap-1">
      <CheckCircle size={12} />
      Passwords match
    </p>
  )}
</div>
```

Validation Rules - Confirm Password:

- Required: "Please confirm your password"
- Must match password: "Passwords do not match"

Company Name Input:

```
<div className="space-y-1 mb-4">
  <label htmlFor="companyName" className="block text-sm font-medium text-gray-700">
    Company Name
  </label>
  <input
    id="companyName"
    type="text"
    name="companyName"
    autoComplete="organization"
    required
    value={companyName}
    onChange={(e) => setCompanyName(e.target.value)}
    className="w-full px-3 py-2 text-sm border border-gray-300 rounded-md focus:ring-2 focus:ring-primary-500"
    placeholder="Acme Corporation"
  />
  {errors.companyName && (
    <p className="text-xs text-error-600">{errors.companyName}</p>
  )}
</div>
```

Validation Rules - Company Name:

- Required: "Company name is required"
- Min length: 2 characters
- Max length: 100 characters

Company Size Dropdown:

```
<div className="space-y-1 mb-6">
  <label htmlFor="companySize" className="block text-sm font-medium text-gray-700">
    Company Size
  </label>
  <select
    id="companySize"
    name="companySize"
    required
    value={companySize}
    onChange={(e) => setCompanySize(e.target.value)}
    className="w-full px-3 py-2 text-sm border border-gray-300 rounded-md focus:ring-2 focus:ring-primary-500 bg-white"
  >
    <option value="">Select company size</option>
    <option value="1-10">1-10 employees</option>
    <option value="11-50">11-50 employees</option>
    <option value="51-200">51-200 employees</option>
    <option value="201-500">201-500 employees</option>
    <option value="501-1000">501-1000 employees</option>
    <option value="1000+">1000+ employees</option>
  </select>
  {errors.companySize && (
    <p className="text-xs text-error-600">{errors.companySize}</p>
  )}
</div>
```

Terms of Service Checkbox:

```
<div className="mb-6">
  <label className="flex items-start gap-3 cursor-pointer">
    <input
      type="checkbox"
      checked={agreedToTerms}
      onChange={(e) => setAgreedToTerms(e.target.checked)}
      className="mt-0.5 w-4 h-4 text-primary-600 border-gray-300 rounded focus:ring-2 focus:ring-primary-500"
      required
    />
    <span className="text-sm text-gray-700">
      I agree to the{' '}
      <a href="/terms" target="_blank" className="text-primary-600 hover:text-primary-700 underline">
        Terms of Service
      </a>
      {' '}and{' '}
      <a href="/privacy" target="_blank" className="text-primary-600 hover:text-primary-700 underline">
        Privacy Policy
      </a>
    </span>
  </label>
  {errors.agreedToTerms && (
    <p className="text-xs text-error-600 mt-1">{errors.agreedToTerms}</p>
  )}
</div>
```

Create Account Button:

```
<button
  type="submit"
  disabled={isLoading || !agreedToTerms}
  className="
    w-full px-4 py-2.5
    bg-primary-600 hover:bg-primary-700 active:bg-primary-800
    text-white font-medium text-sm
    rounded-md
    shadow-sm hover:shadow-md
    transition-all duration-200
    disabled:opacity-50 disabled:cursor-not-allowed
    focus:outline-none focus:ring-2 focus:ring-primary-500 focus:ring-offset-2
    flex items-center justify-center gap-2
  "
>
  {isLoading ? (
    <>
      <Loader size={16} className="animate-spin" />
      <span>Creating account...</span>
    </>
  ) : (
    <span>Create Account</span>
  )}
</button>
```

Sign In Link:

```
<p className="mt-6 text-center text-sm text-gray-600">
  Already have an account?{' '}
  <a href="/login" className="font-medium text-primary-600 hover:text-primary-700">
    Sign in
  </a>
</p>
```

3.2.3 Success Flow

After successful registration:

1. **Show Success Message:**

```
<div className="p-4 bg-success-50 border border-success-200 rounded-lg flex items-start gap-3 mb-4">
  <CheckCircle className="text-success-600 flex-shrink-0" size={20} />
  <div>
    <h4 className="text-sm font-medium text-success-900">Account created successfully!</h4>
    <p className="mt-1 text-sm text-success-700">
      We've sent a verification email to {email}. Please check your inbox and click the verification link.
    </p>
  </div>
</div>
```

2. **Show Next Steps:**

```
<div className="mt-4 space-y-3">
  <button
    onClick={goToLogin}
    className="w-full btn-primary"
  >
    Go to Login
  </button>
  <button
    onClick={resendVerification}
    className="w-full btn-secondary"
  >
    Resend Verification Email
  </button>
</div>
```

3.3 Forgot Password Screen

Route: `/forgot-password`
Access: Public

3.3.1 Layout

```
┌─────────────────────────────────────┐
│                                     │
│          [NEOX Logo]                │
│                                     │
│      Forgot Your Password?          │
│      Enter your email to reset      │
│                                     │
│      [Email input]                  │
│                                     │
│      [Send Reset Link Button]       │
│                                     │
│      [← Back to Login]              │
│                                     │
└─────────────────────────────────────┘
```

3.3.2 Component Specifications

```
<div className="min-h-screen flex items-center justify-center p-4 bg-gray-50">
  <div className="w-full max-w-md">
    <div className="flex justify-center mb-8">
      <img src="/logo-primary.svg" alt="NEOX Infinity" className="h-12" />
    </div>

    <div className="bg-white border border-gray-200 rounded-lg shadow-sm p-8">
      <div className="text-center mb-6">
        <div className="inline-flex items-center justify-center w-16 h-16 bg-primary-100 rounded-full mb-4">
          <Key className="text-primary-600" size={32} />
        </div>
        <h1 className="text-2xl font-bold text-gray-900 mb-2">
          Forgot Your Password?
        </h1>
        <p className="text-sm text-gray-600">
          No worries! Enter your email and we'll send you reset instructions.
        </p>
      </div>

      <form onSubmit={handleSubmit}>
        <div className="space-y-1 mb-6">
          <label htmlFor="email" className="block text-sm font-medium text-gray-700">
            Email Address
          </label>
          <input
            id="email"
            type="email"
            required
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            className="w-full px-3 py-2 text-sm border border-gray-300 rounded-md focus:ring-2 focus:ring-primary-500"
            placeholder="you@company.com"
          />
        </div>

        <button
          type="submit"
          disabled={isLoading}
          className="w-full btn-primary mb-4"
        >
          {isLoading ? (
            <>
              <Loader size={16} className="animate-spin mr-2" />
              Sending...
            </>
          ) : (
            'Send Reset Link'
          )}
        </button>

        <a
          href="/login"
          className="flex items-center justify-center gap-2 text-sm text-gray-600 hover:text-gray-900"
        >
          <ArrowLeft size={16} />
          Back to Login
        </a>
      </form>
    </div>
  </div>
</div>
```

3.3.3 Success State

After sending reset link:

```
<div className="text-center">
  <div className="inline-flex items-center justify-center w-16 h-16 bg-success-100 rounded-full mb-4">
    <Mail className="text-success-600" size={32} />
  </div>
  <h2 className="text-xl font-bold text-gray-900 mb-2">
    Check Your Email
  </h2>
  <p className="text-sm text-gray-600 mb-6">
    If an account exists for {email}, you will receive password reset instructions shortly.
  </p>
  <p className="text-xs text-gray-500 mb-6">
    Didn't receive the email? Check your spam folder or{' '}
    <button onClick={resendEmail} className="text-primary-600 hover:text-primary-700 underline">
      resend the link
    </button>
  </p>
  <a href="/login" className="btn-primary inline-flex items-center gap-2">
    <ArrowLeft size={16} />
    Back to Login
  </a>
</div>
```

3.4 Reset Password Screen

Route: `/reset-password?token=xxx`
Access: Public (with valid token)

3.4.1 Layout

```
┌─────────────────────────────────────┐
│                                     │
│          [NEOX Logo]                │
│                                     │
│      Reset Your Password            │
│      Enter your new password        │
│                                     │
│      [New Password input]           │
│      [Confirm Password input]       │
│                                     │
│      [Password requirements]        │
│                                     │
│      [Reset Password Button]        │
│                                     │
└─────────────────────────────────────┘
```

3.4.2 Component Specifications

```
<div className="min-h-screen flex items-center justify-center p-4 bg-gray-50">
  <div className="w-full max-w-md">
    <div className="flex justify-center mb-8">
      <img src="/logo-primary.svg" alt="NEOX Infinity" className="h-12" />
    </div>

    <div className="bg-white border border-gray-200 rounded-lg shadow-sm p-8">
      <div className="text-center mb-6">
        <div className="inline-flex items-center justify-center w-16 h-16 bg-primary-100 rounded-full mb-4">
          <Lock className="text-primary-600" size={32} />
        </div>
        <h1 className="text-2xl font-bold text-gray-900 mb-2">
          Reset Your Password
        </h1>
```

```
      <p className="text-sm text-gray-600">
        Please enter your new password below
      </p>
    </div>

    <form onSubmit={handleSubmit}>
      {/* New Password */}
      <div className="space-y-1 mb-4">
        <label htmlFor="password" className="block text-sm font-medium text-gray-700">
          New Password
        </label>
        <div className="relative">
          <input
            id="password"
            type={showPassword ? 'text' : 'password'}
            required
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            className="w-full px-3 py-2 pr-10 text-sm border border-gray-300 rounded-md focus:ring-2 focus:ring-primary-500"
            placeholder="Enter new password"
          />
          <button
            type="button"
            onClick={() => setShowPassword(!showPassword)}
            className="absolute right-3 top-1/2 -translate-y-1/2 text-gray-400 hover:text-gray-600"
          >
            {showPassword ? <EyeOff size={16} /> : <Eye size={16} />}
          </button>
        </div>
      </div>

      {/* Confirm Password */}
      <div className="space-y-1 mb-4">
        <label htmlFor="confirmPassword" className="block text-sm font-medium text-gray-700">
          Confirm New Password
        </label>
        <input
          id="confirmPassword"
          type="password"
          required
          value={confirmPassword}
          onChange={(e) => setConfirmPassword(e.target.value)}
          className="w-full px-3 py-2 text-sm border border-gray-300 rounded-md focus:ring-2 focus:ring-primary-500"
          placeholder="Confirm new password"
        />
      </div>

      {/* Password Requirements (same as registration) */}

      <button
        type="submit"
        disabled={isLoading}
        className="w-full btn-primary mt-6"
      >
        {isLoading ? (
          <>
            <Loader size={16} className="animate-spin mr-2" />
            Resetting...
          </>
        ) : (
          'Reset Password'
        )}
      </button>
    </form>
  </div>
```

```
  </div>
</div>
```

3.4.3 Token Validation

On page load, validate token:

```
useEffect(() => {
  const validateToken = async () => {
    const token = new URLSearchParams(window.location.search).get('token');

    if (!token) {
      setError('Invalid or missing reset token');
      return;
    }

    try {
      const response = await fetch(`/api/auth/validate-reset-token`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ token }),
      });

      if (!response.ok) {
        setError('This reset link has expired or is invalid');
      }
    } catch (error) {
      setError('Unable to validate reset token');
    }
  };

  validateToken();
}, []);
```

Invalid Token State:

```
<div className="text-center">
  <div className="inline-flex items-center justify-center w-16 h-16 bg-error-100 rounded-full mb-4">
    <XCircle className="text-error-600" size={32} />
  </div>
  <h2 className="text-xl font-bold text-gray-900 mb-2">
    Invalid Reset Link
  </h2>
  <p className="text-sm text-gray-600 mb-6">
    This password reset link has expired or is invalid. Please request a new one.
  </p>
  <a href="/forgot-password" className="btn-primary inline-block">
    Request New Link
  </a>
</div>
```

3.4.4 Success State

After successful password reset:

```
<div className="text-center">
  <div className="inline-flex items-center justify-center w-16 h-16 bg-success-100 rounded-full mb-4">
    <CheckCircle className="text-success-600" size={32} />
  </div>
  <h2 className="text-xl font-bold text-gray-900 mb-2">
    Password Reset Successful!
  </h2>
  <p className="text-sm text-gray-600 mb-6">
    Your password has been reset successfully. You can now sign in with your new password.
  </p>
  <a href="/login" className="btn-primary inline-block">
    Go to Login
  </a>
</div>
```

3.5 Two-Factor Authentication (2FA) Screen

Route: `/login/verify`
Access: Authenticated session with pending 2FA

3.5.1 Layout

```
┌─────────────────────────────────────┐
│                                     │
│          [NEOX Logo]                │
│                                     │
│      Two-Factor Authentication      │
│      Enter the 6-digit code         │
│                                     │
│      [Code Input] [Code] [Code]     │
│      [Code] [Code] [Code]           │
│                                     │
│      Code sent to ••••@email.com    │
│                                     │
│      [Verify Button]                │
│                                     │
│      Didn't receive code?           │
│      [Resend Code]                  │
│                                     │
│      [Use backup code instead]      │
│                                     │
└─────────────────────────────────────┘
```

3.5.2 Component Specifications

6-Digit Code Input:

```
<div className="flex gap-2 justify-center mb-6">
  {[0, 1, 2, 3, 4, 5].map((index) => (
    <input
      key={index}
      ref={(el) => (inputRefs.current[index] = el)}
      type="text"
      inputMode="numeric"
      maxLength={1}
      value={code[index] || ''}
      onChange={(e) => handleCodeChange(index, e.target.value)}
      onKeyDown={(e) => handleKeyDown(index, e)}
      onPaste={(e) => handlePaste(e)}
      className="
        w-12 h-14
        text-center text-2xl font-semibold
        border-2 border-gray-300
        rounded-lg
        focus:border-primary-500 focus:ring-2 focus:ring-primary-500
        transition-colors
      "
      aria-label={`Digit ${index + 1}`}
    />
  ))}
</div>
```

Code Input Logic:

```
const handleCodeChange = (index: number, value: string) => {
  // Only allow digits
  if (!/^\d*$/.test(value)) return;

  const newCode = [...code];
  newCode[index] = value;
  setCode(newCode);

  // Auto-focus next input
  if (value && index < 5) {
    inputRefs.current[index + 1]?.focus();
  }

  // Auto-submit when all 6 digits entered
  if (newCode.every(digit => digit) && newCode.length === 6) {
    handleVerify(newCode.join(''));
  }
};

const handleKeyDown = (index: number, e: React.KeyboardEvent) => {
  if (e.key === 'Backspace' && !code[index] && index > 0) {
    inputRefs.current[index - 1]?.focus();
  }
};

const handlePaste = (e: React.ClipboardEvent) => {
  e.preventDefault();
  const pastedData = e.clipboardData.getData('text').replace(/\D/g, '').slice(0, 6);

  if (pastedData.length === 6) {
    setCode(pastedData.split(''));
    handleVerify(pastedData);
  }
};
```

Resend Code:

```
<div className="text-center mt-6">
  <p className="text-sm text-gray-600 mb-2">
    Didn't receive the code?
  </p>
  {canResend ? (
    <button
      onClick={handleResend}
      className="text-sm text-primary-600 hover:text-primary-700 font-medium"
    >
      Resend Code
    </button>
  ) : (
    <p className="text-sm text-gray-500">
      Resend available in {countdown}s
    </p>
  )}
</div>
```

3.5.3 Backup Codes

```
<button
  onClick={() => setShowBackupCodeInput(true)}
  className="mt-4 text-sm text-gray-600 hover:text-gray-900"
>
  Use backup code instead
</button>

{showBackupCodeInput && (
  <div className="mt-4">
    <input
      type="text"
      placeholder="Enter 8-digit backup code"
      className="w-full px-3 py-2 text-sm border border-gray-300 rounded-md"
      maxLength={8}
      value={backupCode}
      onChange={(e) => setBackupCode(e.target.value)}
    />
    <button
      onClick={handleVerifyBackupCode}
      className="w-full btn-primary mt-2"
    >
      Verify Backup Code
    </button>
  </div>
)}
```

3.6 Email Verification Screen

Route: `/verify-email?token=xxx`
Access: Public

3.6.1 Auto-Verification

On page load:

```
useEffect(() => {
  const verifyEmail = async () => {
    const token = new URLSearchParams(window.location.search).get('token');

    if (!token) {
      setStatus('error');
      setMessage('Invalid verification link');
      return;
    }

    setStatus('verifying');

    try {
      const response = await fetch(`/api/auth/verify-email`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ token }),
      });

      if (response.ok) {
        setStatus('success');
        setTimeout(() => {
          window.location.href = '/login?verified=true';
        }, 3000);
      } else {
        setStatus('error');
        setMessage('Verification link has expired');
      }
    } catch (error) {
      setStatus('error');
      setMessage('Unable to verify email');
    }
  };

  verifyEmail();
}, []);
```

3.6.2 UI States

Verifying State:

```
<div className="text-center">
  <Loader size={48} className="animate-spin text-primary-600 mx-auto mb-4" />
  <h2 className="text-xl font-bold text-gray-900 mb-2">
    Verifying Your Email...
  </h2>
  <p className="text-sm text-gray-600">
    Please wait while we verify your email address.
  </p>
</div>
```

Success State:

```
<div className="text-center">
  <div className="inline-flex items-center justify-center w-16 h-16 bg-success-100 rounded-full mb-4">
    <CheckCircle className="text-success-600" size={32} />
  </div>
  <h2 className="text-xl font-bold text-gray-900 mb-2">
    Email Verified!
  </h2>
  <p className="text-sm text-gray-600 mb-6">
    Your email has been verified successfully. Redirecting to login...
  </p>
  <div className="flex justify-center">
    <Loader size={20} className="animate-spin text-primary-600" />
  </div>
</div>
```

Error State:

```
<div className="text-center">
  <div className="inline-flex items-center justify-center w-16 h-16 bg-error-100 rounded-full mb-4">
    <XCircle className="text-error-600" size={32} />
  </div>
  <h2 className="text-xl font-bold text-gray-900 mb-2">
    Verification Failed
  </h2>
  <p className="text-sm text-gray-600 mb-6">
    {message}
  </p>
  <button onClick={resendVerification} className="btn-primary">
    Resend Verification Email
  </button>
</div>
```

3.7 SSO Callback Screen

Route: `/auth/callback?provider=google&code=xxx`
Access: Public

This is an intermediate screen that processes OAuth callbacks.

3.7.1 Processing State

```
<div className="min-h-screen flex items-center justify-center bg-gray-50">
  <div className="text-center">
    <Loader size={48} className="animate-spin text-primary-600 mx-auto mb-4" />
    <h2 className="text-xl font-bold text-gray-900 mb-2">
      Signing you in...
    </h2>
    <p className="text-sm text-gray-600">
      Completing authentication with {provider}
    </p>
  </div>
</div>
```

3.7.2 Error Handling

If SSO fails:

```
<div className="min-h-screen flex items-center justify-center p-4 bg-gray-50">
  <div className="w-full max-w-md bg-white border border-gray-200 rounded-lg shadow-sm p-8">
    <div className="text-center">
      <div className="inline-flex items-center justify-center w-16 h-16 bg-error-100 rounded-full mb-4">
        <XCircle className="text-error-600" size={32} />
      </div>
      <h2 className="text-xl font-bold text-gray-900 mb-2">
        Authentication Failed
      </h2>
      <p className="text-sm text-gray-600 mb-6">
        {errorMessage}
      </p>
      <a href="/login" className="btn-primary inline-block">
        Back to Login
      </a>
    </div>
  </div>
</div>
```

3.8 Session Expired Screen

Route: `/session-expired`
Access: Public

Shown when user's session expires.

```
<div className="min-h-screen flex items-center justify-center p-4 bg-gray-50">
  <div className="w-full max-w-md bg-white border border-gray-200 rounded-lg shadow-sm p-8">
    <div className="text-center">
      <div className="inline-flex items-center justify-center w-16 h-16 bg-warning-100 rounded-full mb-4">
        <Clock className="text-warning-600" size={32} />
      </div>
      <h2 className="text-xl font-bold text-gray-900 mb-2">
        Session Expired
      </h2>
      <p className="text-sm text-gray-600 mb-6">
        Your session has expired due to inactivity. Please sign in again to continue.
      </p>
      <a href="/login" className="btn-primary inline-block w-full">
        Sign In Again
      </a>
    </div>
  </div>
</div>
```

---

4. API Endpoints

4.1 POST /api/auth/login

Request:

```
{
  "email": "user@example.com",
  "password": "SecurePass123!",
  "rememberMe": true
}
```

Success Response (200):

```
{
  "success": true,
  "user": {
    "id": "usr_123456",
    "email": "user@example.com",
    "firstName": "John",
    "lastName": "Doe",
    "role": "admin",
    "tenantId": "tenant_123",
    "twoFactorEnabled": false
  },
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refreshToken": "rt_abc123...",
  "expiresIn": 3600
}
```

Success Response with 2FA Required (200):

```
{
  "success": true,
  "requiresTwoFactor": true,
  "sessionId": "sess_temp_123",
  "message": "Please enter your 2FA code"
}
```

Error Responses:

401 Unauthorized - Invalid Credentials:

```
{
  "success": false,
  "error": "INVALID_CREDENTIALS",
  "message": "Invalid email or password"
}
```

403 Forbidden - Email Not Verified:

```
{
  "success": false,
  "error": "EMAIL_NOT_VERIFIED",
  "message": "Please verify your email address before logging in"
}
```

423 Locked - Account Locked:

```
{
  "success": false,
  "error": "ACCOUNT_LOCKED",
  "message": "Account locked due to multiple failed login attempts",
  "lockedUntil": "2025-10-28T02:30:00Z"
}
```

4.2 POST /api/auth/register

Request:

```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "john@example.com",
  "password": "SecurePass123!",
  "companyName": "Acme Corp",
  "companySize": "11-50"
}
```

Success Response (201):

```
{
  "success": true,
  "user": {
    "id": "usr_123456",
    "email": "john@example.com",
    "firstName": "John",
    "lastName": "Doe",
    "emailVerified": false
  },
  "message": "Account created successfully. Please check your email to verify your account."
}
```

Error Responses:

409 Conflict - Email Already Exists:

```
{
  "success": false,
  "error": "EMAIL_EXISTS",
  "message": "An account with this email already exists"
}
```

400 Bad Request - Validation Error:

```
{
  "success": false,
  "error": "VALIDATION_ERROR",
  "message": "Invalid input data",
  "errors": {
    "email": "Invalid email format",
    "password": "Password must be at least 8 characters"
  }
}
```

4.3 POST /api/auth/forgot-password

Request:

```
{
  "email": "user@example.com"
}
```

Success Response (200):

```
{
  "success": true,
  "message": "If an account exists with this email, you will receive password reset instructions"
}
```

Note: Always return success to prevent email enumeration attacks.

4.4 POST /api/auth/reset-password

Request:

```
{
  "token": "reset_token_abc123",
  "newPassword": "NewSecurePass123!"
}
```

Success Response (200):

```
{
  "success": true,
  "message": "Password reset successfully"
}
```

Error Responses:

400 Bad Request - Invalid/Expired Token:

```
{
  "success": false,
  "error": "INVALID_TOKEN",
  "message": "This reset link has expired or is invalid"
}
```

4.5 POST /api/auth/verify-2fa

Request:

```
{
  "sessionId": "sess_temp_123",
  "code": "123456"
}
```

Success Response (200):

```
{
  "success": true,
  "user": {
    "id": "usr_123456",
    "email": "user@example.com",
    "firstName": "John",
    "lastName": "Doe"
  },
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refreshToken": "rt_abc123..."
}
```

Error Responses:

401 Unauthorized - Invalid Code:

```
{
  "success": false,
  "error": "INVALID_CODE",
  "message": "Invalid verification code",
  "attemptsRemaining": 2
}
```

4.6 POST /api/auth/verify-email

Request:

```
{
  "token": "verify_token_abc123"
}
```

Success Response (200):

```
{
  "success": true,
  "message": "Email verified successfully"
}
```

4.7 POST /api/auth/refresh-token

Request:

```
{
  "refreshToken": "rt_abc123..."
}
```

Success Response (200):

```
{
  "success": true,
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "expiresIn": 3600
}
```

4.8 POST /api/auth/logout

Request:

```
{
  "refreshToken": "rt_abc123..."
}
```

Success Response (200):

```
{
  "success": true,
  "message": "Logged out successfully"
}
```

4.9 GET /api/auth/oauth/:provider

Initiates OAuth flow. Redirects to provider's authorization URL.

Supported Providers:

- `google`
- `microsoft`

Query Parameters:

- `returnUrl` (optional): URL to redirect to after successful authentication

---

4.10 GET /api/auth/oauth/:provider/callback

Handles OAuth callback from provider.

Query Parameters:

- `code` : Authorization code from provider
- `state` : State parameter for CSRF protection

Success: Redirects to dashboard with session cookie set
Error: Redirects to `/login?error=oauth_failed`

---

5. Security Requirements

5.1 Password Requirements

- Minimum 8 characters
- At least one uppercase letter
- At least one lowercase letter
- At least one number
- At least one special character (!@#$%^&*)
- Maximum 128 characters
- Not in common passwords list (top 10,000)

5.2 Account Lockout

- Lock account after 5 failed login attempts
- Lock duration: 15 minutes
- Reset counter after successful login
- Send email notification on account lock

5.3 Session Management

- Access token expiry: 1 hour
- Refresh token expiry: 7 days (30 days if "Remember me" checked)
- Refresh token rotation on use
- Maximum 5 active sessions per user
- Invalidate all sessions on password change

5.4 Rate Limiting

Login Endpoint:

- 5 requests per minute per IP
- 10 requests per hour per email

Registration Endpoint:

- 3 requests per hour per IP

Password Reset Endpoint:

- 3 requests per hour per email
- 10 requests per hour per IP

5.5 Token Security

JWT Access Token:

```
{
  "sub": "usr_123456",
  "email": "user@example.com",
  "role": "admin",
  "tenantId": "tenant_123",
  "iat": 1698765432,
  "exp": 1698769032
}
```

Refresh Token:

- Random 64-byte hex string
- Stored in database with user association
- Marked as revoked after use (rotation)
- httpOnly cookie (preferred) or response body

5.6 CSRF Protection

- Use SameSite=Lax for cookies

- Include CSRF token in state parameter for OAuth
- Validate referer header

5.7 SSL/TLS

- All authentication endpoints must use HTTPS
- Enforce HSTS header
- Use secure, httpOnly cookies

---

6. Error Handling

6.1 Error Codes

Code	HTTP Status	Description
INVALID_CREDENTIALS	401	Wrong email/password
EMAIL_NOT_VERIFIED	403	Email not verified
ACCOUNT_LOCKED	423	Too many failed attempts
ACCOUNT_DISABLED	403	Account deactivated
EMAIL_EXISTS	409	Email already registered
INVALID_TOKEN	400	Invalid/expired reset token
INVALID_CODE	401	Invalid 2FA code
SESSION_EXPIRED	401	Session expired
RATE_LIMIT_EXCEEDED	429	Too many requests
VALIDATION_ERROR	400	Input validation failed
OAUTH_ERROR	400	OAuth flow failed

6.2 Error Display Guidelines

- Show specific errors for validation (field-level)
- Show generic errors for authentication failures (prevent enumeration)
- Auto-dismiss success toasts after 3 seconds
- Keep error messages visible until user corrects issue
- Provide actionable next steps in error messages

---

7. Testing Scenarios

7.1 Login Tests

Positive Tests:

- ☐ Login with valid credentials
- ☐ Login with "Remember me" checked
- ☐ Login redirects to returnUrl if present
- ☐ Login with 2FA enabled flows to verification
- ☐ Login via Google OAuth
- ☐ Login via Microsoft OAuth

Negative Tests:

- ☐ Login with wrong password shows error
- ☐ Login with non-existent email shows error
- ☐ Login with unverified email shows verification prompt
- ☐ Login exceeding rate limit shows error
- ☐ Login with locked account shows lock message
- ☐ Login with empty fields shows validation errors

Edge Cases:

- Email with leading/trailing spaces (should be trimmed)
- Case-insensitive email matching
- Password with special characters
- Very long email/password (max length validation)

7.2 Registration Tests

Positive Tests:

- ☐ Register with all required fields
- ☐ Receive verification email after registration
- ☐ Password strength indicator updates correctly

Negative Tests:

- ☐ Register with existing email shows error
- ☐ Register with weak password shows errors
- ☐ Register with mismatched passwords shows error
- ☐ Register without agreeing to terms disabled button
- ☐ Register exceeding rate limit shows error

Edge Cases:

- Email availability check debouncing
- Password requirements all combinations
- Company name with special characters

7.3 Password Reset Tests

Positive Tests:

- ☐ Request reset link with valid email
- ☐ Reset password with valid token
- ☐ Expired token shows appropriate error

Negative Tests:

- ☐ Reset with invalid token shows error
- ☐ Reset with weak password shows errors
- ☐ Used token cannot be reused

7.4 2FA Tests

Positive Tests:

- ☐ Verify with correct 6-digit code
- ☐ Verify with backup code
- ☐ Auto-submit when 6 digits entered
- ☐ Paste 6-digit code works

Negative Tests:

- ☐ Verify with wrong code shows error
- ☐ Verify with expired code shows error
- ☐ Exceeding attempts locks out temporarily

7.5 Session Management Tests

Positive Tests:

- ☐ Access token refreshes automatically
- ☐ Logout invalidates tokens
- ☐ Session expires after inactivity

Negative Tests:

- ☐ Expired access token redirects to login
- ☐ Invalid refresh token logs out user
- ☐ Concurrent logins on different devices

---

8. Implementation Checklist

Frontend Tasks

- [ ] Implement Login screen with all states
- [ ] Implement Registration screen with validation
- [ ] Implement Forgot Password flow
- [ ] Implement Reset Password flow
- [ ] Implement 2FA verification screen

- [ ] Implement Email verification screen
- [ ] Implement SSO OAuth flows (Google, Microsoft)
- [ ] Implement session management (auto-refresh tokens)
- [ ] Implement protected route wrapper
- [ ] Add form validation with real-time feedback
- [ ] Add password strength indicator
- [ ] Add email availability check
- [ ] Implement "Remember me" functionality
- [ ] Add rate limiting feedback
- [ ] Add proper error handling and display
- [ ] Implement accessibility (ARIA labels, keyboard nav)
- [ ] Add loading states for all async operations
- [ ] Write unit tests for components
- [ ] Write E2E tests for auth flows

Backend Tasks

- [ ] Implement user registration endpoint
- [ ] Implement login endpoint with validation
- [ ] Implement JWT token generation/validation
- [ ] Implement refresh token rotation
- [ ] Implement password hashing (bcrypt)
- [ ] Implement forgot password flow
- [ ] Implement email verification
- [ ] Implement 2FA setup and verification
- [ ] Implement OAuth integration (Google, Microsoft)
- [ ] Implement account lockout mechanism
- [ ] Implement rate limiting
- [ ] Implement session management
- [ ] Set up email service (SendGrid/AWS SES)
- [ ] Implement password reset token expiry
- [ ] Implement CSRF protection
- [ ] Add security headers (HSTS, CSP)
- [ ] Set up Redis for session storage
- [ ] Write API tests for all endpoints
- [ ] Set up monitoring and alerting

End of Authentication Module Specification

This authentication system provides enterprise-grade security while maintaining excellent user experience. All screens, states, validations, and error cases are fully specified for implementation.