



COLLEGE CODE : 9604

COLLEGE NAME : C.S.I INSTITUTE OF TECHNOLOGY

EPARTMENT : INFORMATION TECHNOLOGY

STUDENT NM- ID :

DC9FC86CB5D24391DA4CD5AFB42A1

ROLL NO : 960423205019

DATE : 22-9-2025

SUBMITTED BY,

NAME : S.

SANJEEVIKUMAR

MOBILE NO:



Phase 2 - Solution Design & Architecture: User Authentication System

Tech Stack Selection:

A robust authentication system can be implemented using a modern tech stack. Common choices include:

- **Frontend:** React.js for quick UI development, enabling secure user input and seamless interaction with APIs.
- **Backend:** Node.js (Express) to facilitate RESTful API development, session management, and integration with authentication libraries such as Passport.js or JWT.
- **Database:** MongoDB or PostgreSQL for storing user credentials, session tokens, and audit logs in encrypted formats.
- **Security Utilities:** bcrypt (password hashing), Argon2/PBKDF2 (advanced hashing), and secure random token generators.
- **Additional Libraries:** Helmet for securing HTTP headers, CORS middleware, and rate limiting for brute-force prevention.

This tech stack supports scalability, resilience, and industry-standard secure authentication, while remaining flexible for app needs.

UI Structure & API Schema Design:

UI Structure:

The UI is designed for secure, intuitive flows:

- **Login/Register Screen:** Collects user credentials (username/email, password) and optionally prompts for OTP/MFA.
- **Forgot Password:** Guides users through password reset steps.



- **Multi-Factor Authentication Screen:** For users who have MFA enabled, prompts for OTP, push notification response, or biometric verification.
- **Error Handling:** Displays clear feedback for failed attempts, locked accounts, and malicious activity.

API Schema Design:

RESTful API endpoints handle authentication processes:

Endpoint	Method	Description
/api/user/register	POST	Registers new users, validates data, stores securely.
/api/user/authenticate	POST	Authenticates credentials and issues JWT/session token.
/api/user/forgot-password	POST	Initiates password reset flow (email token/OTP).
/api/user/update-password	PUT	Allows password change after validation.
/api/session/validate	GET	Validates active user session, checks token expiry.
/api/session/logout	POST	Ends user session, revokes token.

Authentication uses encrypted (HTTPS) requests, JWT tokens for session persistence, and secure cookies for browser sessions.

Data Handling Approach:

Secure Credential Storage:

- All passwords and sensitive data are hashed with salt using bcrypt or Argon2 before database storage.
- Session tokens (JWTs) contain only non-sensitive claims, signed with secure keys.
- Multi-factor authentication secrets are stored encrypted and never exposed in transit.



Authentication Flow:

- 1. User provides credentials via login/register.
- 2. Backend validates credentials against hashed records.
- 3. Session token/JWT is issued if successful, with expiry time and device context.
- 4. On every API call, token validation ensures session integrity and device legitimacy.
- 5. For critical actions, secondary MFA verification is performed (OTP, push, etc.).

Database Design:

Table	Fields Included
Users	user_id,username/email,password_hash,created_at.
Credentials	credential_id,user_id,password_hash,last_login.
Sessions	session_id,user_id,login_time,last_activity,token_expiry.
Tokens	token_id,user_id,token_value,expiration_time.
PasswordReset	request_id,user_id,token_value,expiration_time.

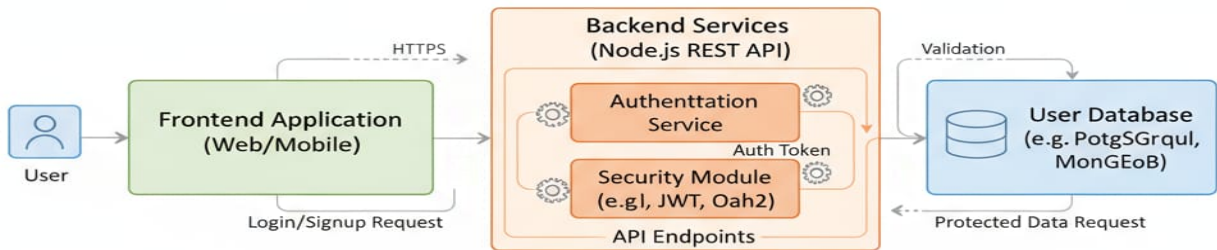
The database is designed for quick retrieval, audit logging, and resistance to brute-force and enumeration attacks.

Component/Module Diagram:

The authentication system follows a modular structure:

- **User Management Module:** Handles registration, profile updates, and password reset requests. Communicates with the credential storage and audit logger.
- **Authentication Module:** Validates login credentials, performs MFA, issues tokens, and logs every authentication event. Integrates with token generation and session management.
- **Session Management Module:** Tracks sessions, validates tokens, and expires sessions based on inactivity or explicit logout.

- **Authorization Module:** Checks user roles and grants or restricts access to resources, working in tandem with RBAC (Role-Based Access Control) tables.
- **Audit & Logging Module:** Keeps immutable records of authentication operations, failed attempts, and suspicious activity.



Legend

 User & Data Storage

 Client

 Orange Backend Logic
Dark Orage for Specfic Backend Modules



Basic Flow Diagram:

User Authentication Flow:

1. User enters their credentials on the login page (username/email, password).
2. Credentials submitted via API are hashed and compared against the stored hashes.
3. Upon success, the system issues a JWT or session token and optionally invokes MFA for critical resources.
4. The client stores the token and presents it in API requests for protected resources.
5. Any suspicious or failed login triggers adaptive checks (IP, device, time) and may challenge with step-up authentication (OTP, push notification).

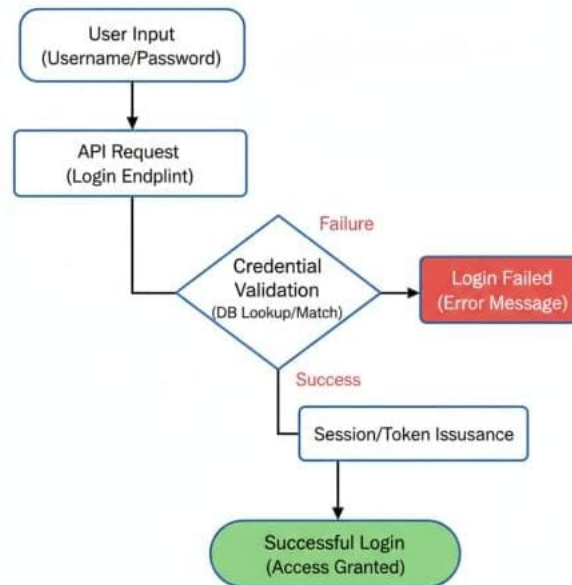
Alternate Flows:

- Failed Authentication: Error returned, and retry options are shown.
- Locked Account: User receives confirmation and instructions to unlock/verify identity.
- Password Reset: Secure flow via email/OTP initiated.

Security Practices

- Strong password policies: Minimum length, complexity requirements, regular enforced changes.
- Salted hashing for passwords: Prevents reversal and protects against rainbow table attacks.
- Multi-factor authentication: Push notifications, OTP, or biometrics on critical flows.
- Role-based access control: Granular permissions for accounts and audit mechanisms.
- Secure session handling: Session timeouts, device binding, and immediate token revocation on logout or suspicious activity.

User Login Authentication Flow



MULTI-FACTOR AUTHENTICATION (MFA) FLOW

