 

COLLEGE CODE :9222 COLLEGE NAME:

THENI KAMMAVAR SANGAM COLLEGE OF TECHNOLOGY DEPARTMENT: CSE

STUDENT NM ID : 510B91AABA82C80AC3C4B7CBA9B2C28C

ROLL NO:23CS036/922223104035

DATE:03-10-2025

COMPLETED THE PROJECT NAMED AS PHASE 3 TECHNOLOGY PROJECT NAME:

TOKEN REFRESH AND EXPIRY SUBMITTED BY

NAME:PARAMESHWARAPANDIAN.S

MOBILE NO: 9894252096

**Project Title**

**Token Refresh and Expiry**

**Abstract**

This project focuses on implementing a secure authentication mechanism using **JSON Web Tokens (JWTs)** with access token expiry and refresh token logic. The system ensures that user sessions remain secure, tokens are renewed seamlessly, and unauthorized access is prevented. The project highlights **data protection, token lifecycle management, and secure persistence**, which are essential in modern web applications.

**Objectives**

* To implement **JWT-based authentication** with token expiry.
* To design a **secure token refresh mechanism**.
* To prevent **unauthorized access** using token verification middleware.
* To persist and manage refresh tokens safely.
* To test the system for **reliability, scalability, and security**.

**Modules**

**1. Project Setup and Environment Finalization**

* Select Tech Stack (Java/Spring Boot or Node.js/Express).
* Configure environment (JWT secret keys, expiry time, refresh logic).
* Install necessary libraries.

**2. Core Features Implementation: Logic and Protection**

* Generate access tokens with expiry time.
* Implement refresh token flow.
* Secure APIs with middleware/interceptors.
* Apply role-based authorization.

**3. Data Storage and Persistence**

* Store refresh tokens securely in a database (MySQL/Redis).
* Hash or encrypt tokens before storing.
* Maintain token logs (issued/revoked/expired).

**4. Testing Core Features and Quality Assurance**

* Unit and integration tests for token expiry and refresh logic.
* Test API protection with expired/invalid tokens.
* Perform load and security testing.

**5. Version Control (GitHub)**

* Use GitHub repository for code and version control.
* Maintain branch structure (main, dev, feature).
* Use GitHub Actions for CI/CD automation.
* Add README.md with setup guide and architecture diagram.

**Proposed System Architecture**

* **Frontend:** React / Angular / Vue (optional).
* **Backend:** Node.js (Express) / Java (Spring Boot) / Python (FastAPI).
* **Database:** MySQL / MongoDB / Redis (for token storage).
* **Authentication:** JWT (Access Token + Refresh Token).

**Expected Outcomes**

* Secure API access with token-based authentication.
* Automatic session renewal using refresh tokens.
* Prevention of unauthorized or expired token usage.
* A scalable and secure authentication flow for modern applications.

**Tools & Technologies**

* Programming Language: Java / Node.js / Python
* Framework: Spring Boot / Express.js / FastAPI
* Database: MySQL / MongoDB / Redis
* Version Control: GitHub
* Testing: JUnit / Mocha / PyTest

**Conclusion**

This project demonstrates the importance of secure token lifecycle management in authentication systems. By implementing **access token expiry** and **refresh token renewal**, the system ensures both **security** and **usability**, making it highly relevant for real-world web and mobile applications.