Refresh Token + Logout + Token Revocation (Enterprise JWT Flow)

This document provides the exact file-wise changes required to implement Access Token + Refresh Token system with Logout (revocation) support in Spring Boot.

File / Module	What to Implement	Why We Do It
application.yml	Add jwt.secret, jwt.access-expiration, jwt.refresh-expiration.	Because we need separate expiry for access token (short) and refresh token (long).
entity/RefreshToken.java	Create RefreshToken entity with fields: token, expiryDate, user, revoked.	Because refresh tokens must be stored in DB for logout and revocation support.
repository/RefreshTokenRepository.java	Create repository with findByToken() and deleteByUserId().	Because we need to fetch refresh token and manage token records in DB.
dto/AuthResponseDto.java	Update DTO to return accessToken + refreshToken + email.	Because login response must return both tokens in enterprise JWT flow.
dto/RefreshTokenRequestDto.java	Create DTO with refreshToken field.	Because refresh/logout APIs require refresh token input in request body.
security/jwt/JwtService.java	Update JwtService to generateAccessToken() and generateRefreshToken() separately.	Because access and refresh tokens must have different expiry durations.
service/services/RefreshTokenService.java	Create interface: createRefreshToken(), verifyExpiration(), revokeToken().	Because refresh token logic should be separated into its own service layer.
service/serviceImpl/RefreshTokenServiceImpl.java	Implement refresh token creation, expiry validation, and revocation.	Because this manages refresh token lifecycle like real enterprise projects.
service/serviceImpl/AuthServiceImpl.java	Update login(): generate access token + create refresh token from RefreshTokenService.	Because login should return short-lived access token and long-lived refresh token.
controller/AuthController.java	Add POST /refresh API using RefreshTokenRequestDto.	Because user should get new access token without logging in again.
service/services/AuthService.java	Add method: AuthResponseDto refreshAccessToken(String refreshToken).	Because controller should call service interface for refresh functionality.
service/serviceImpl/AuthServiceImpl.java	Implement refreshAccessToken(): validate refresh token, generate new access token.	Because refresh token is used to issue a new access token when old one expires.
controller/AuthController.java	Add POST /logout API.	Because logout must invalidate refresh token to prevent future token refresh.

service/services/AuthService.java	Add method: void logout(String refreshToken).	Because logout should be exposed through service interface.
service/serviceImpl/AuthServiceImpl.java	Implement logout(): call refreshTokenService.revokeToken(refreshToken).	Because revoking refresh token ensures user session is terminated securely.

Final Flow:

• Login → returns Access Token + Refresh Token
• Access Token expires quickly (example: 15 minutes)
• Refresh Token expires later (example: 7 days)
• /refresh API generates new Access Token using Refresh Token
• /logout API revokes refresh token so refresh cannot be used again