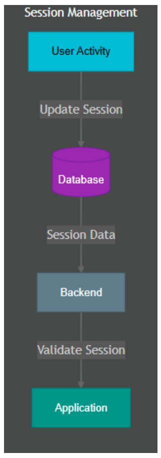
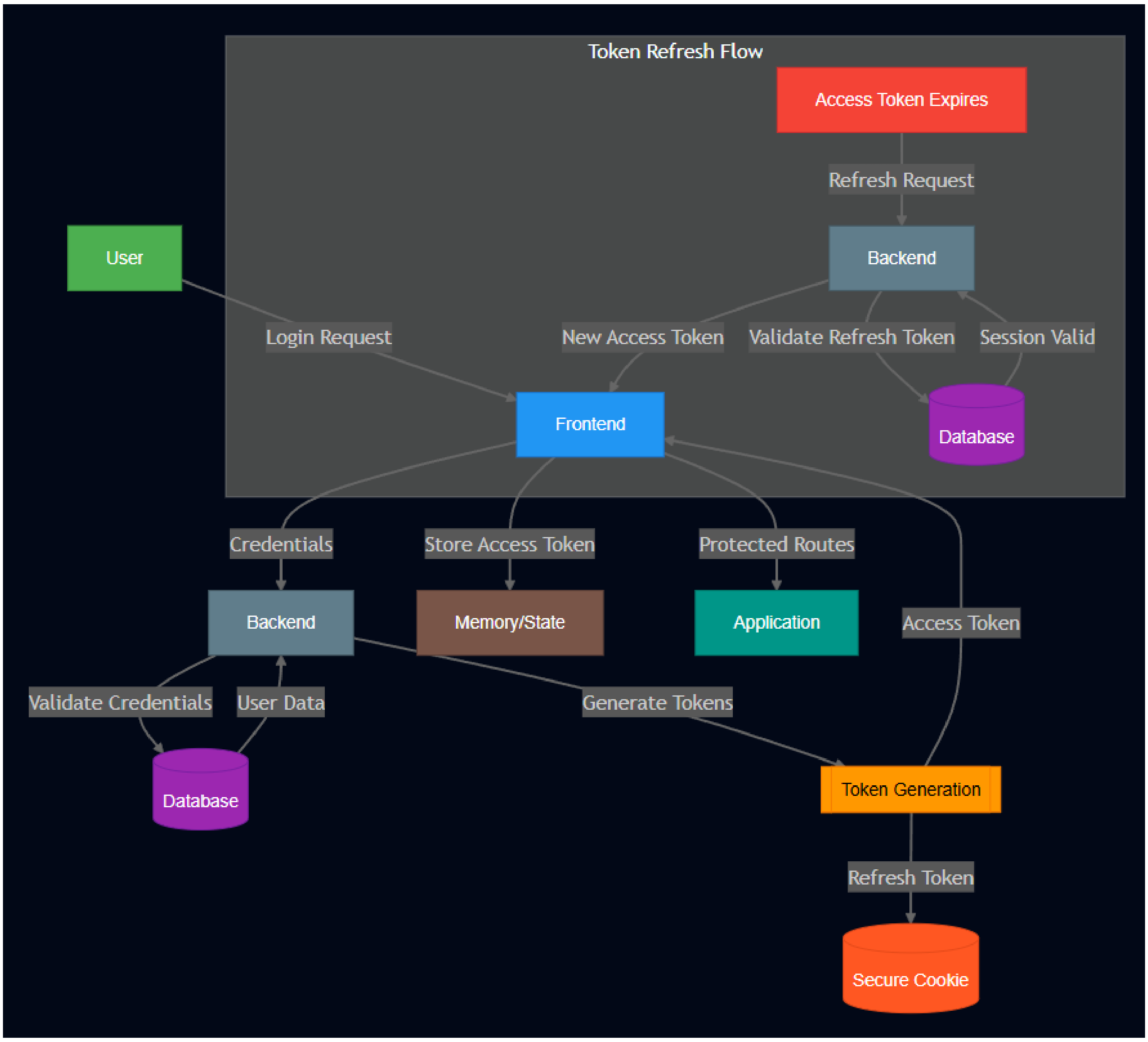
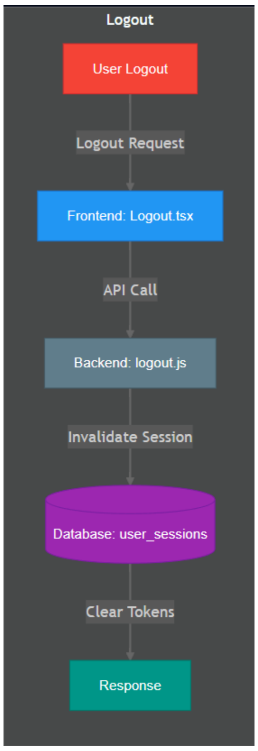
Authentication System Overview

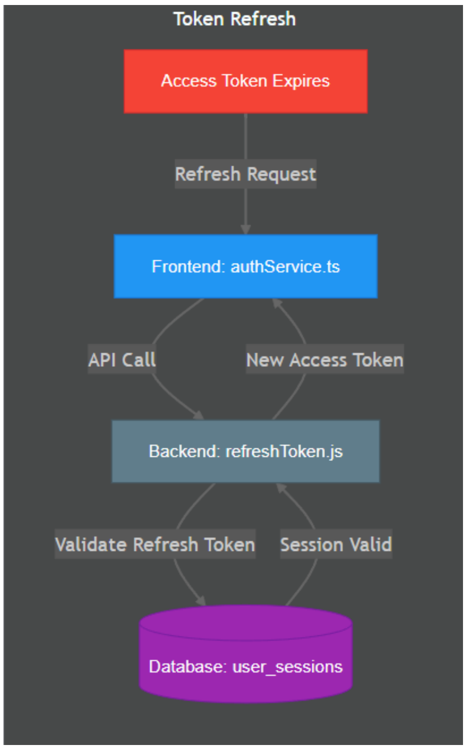
High level view

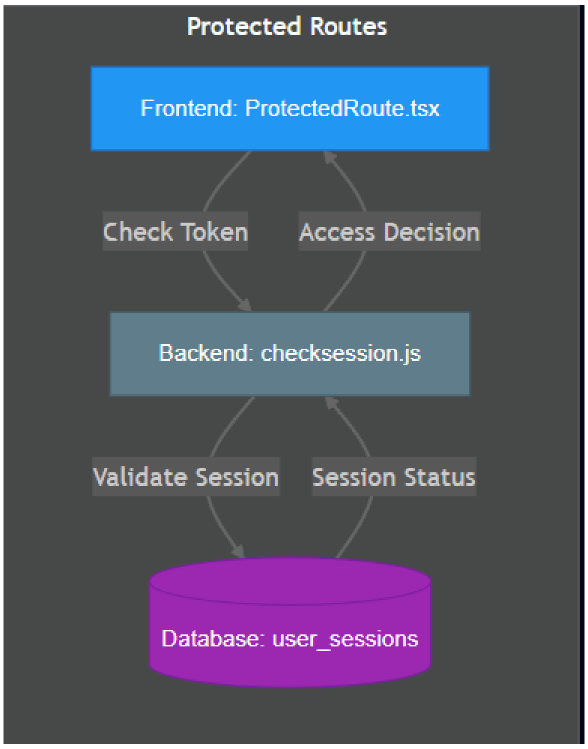


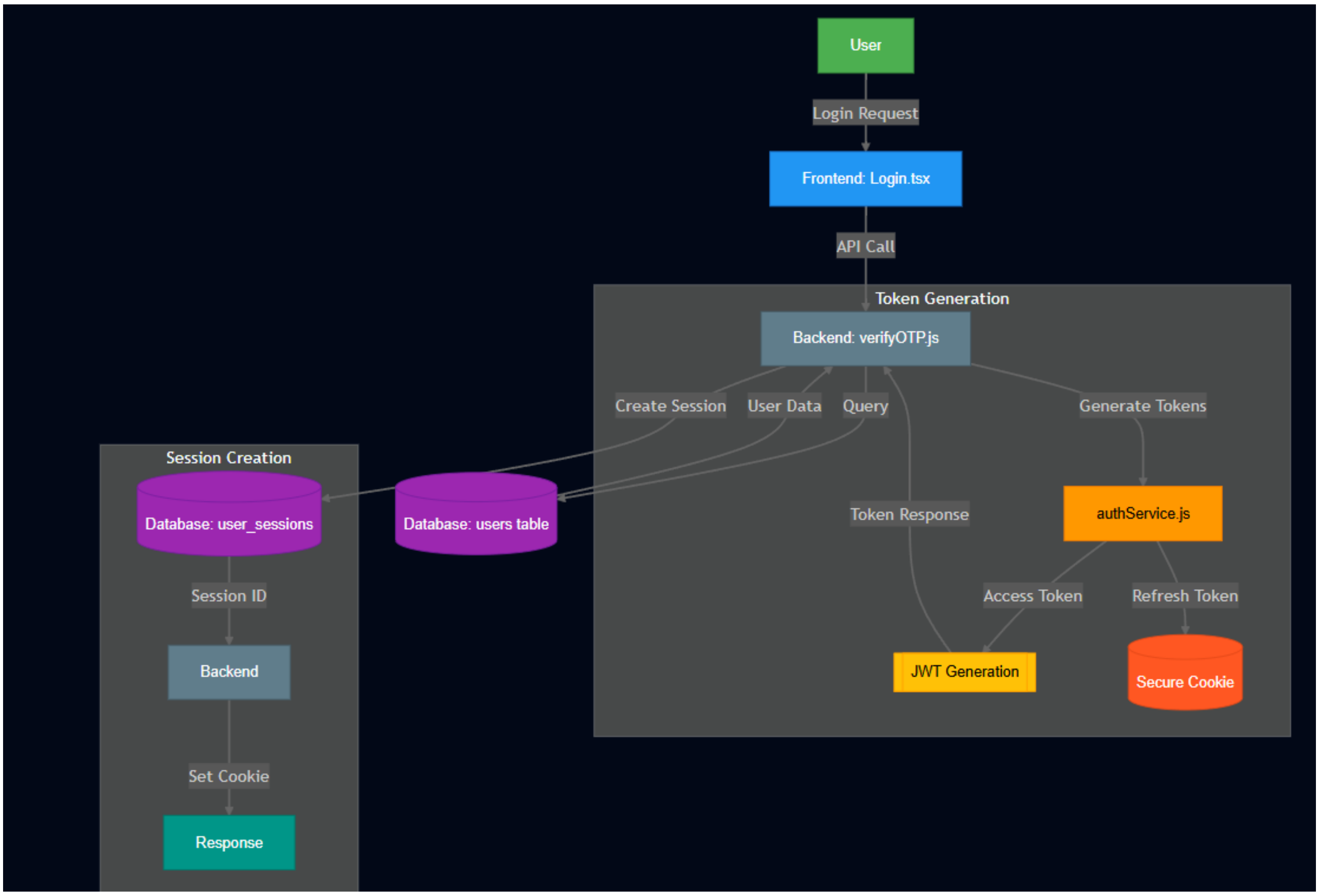


Low level View

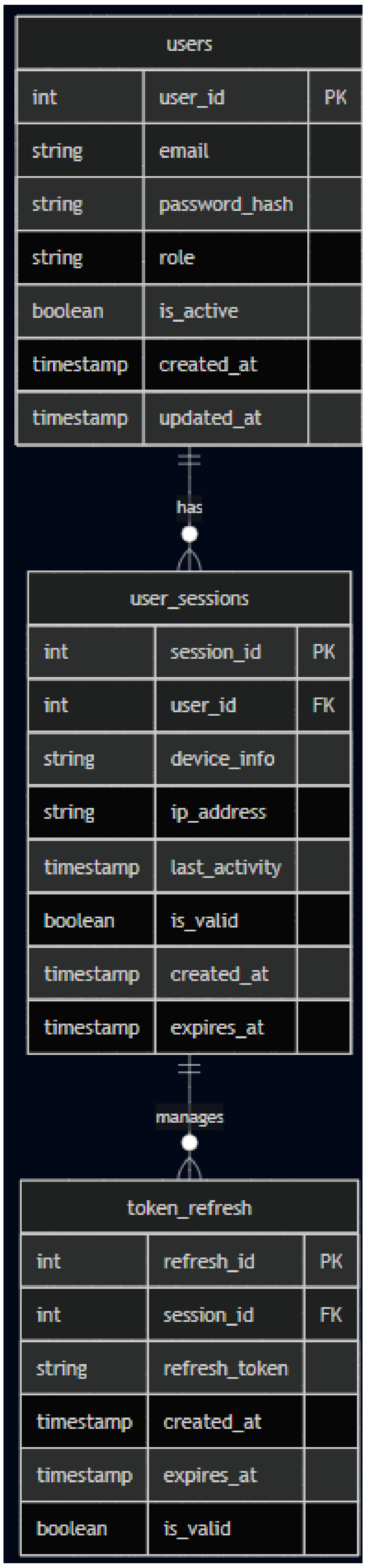




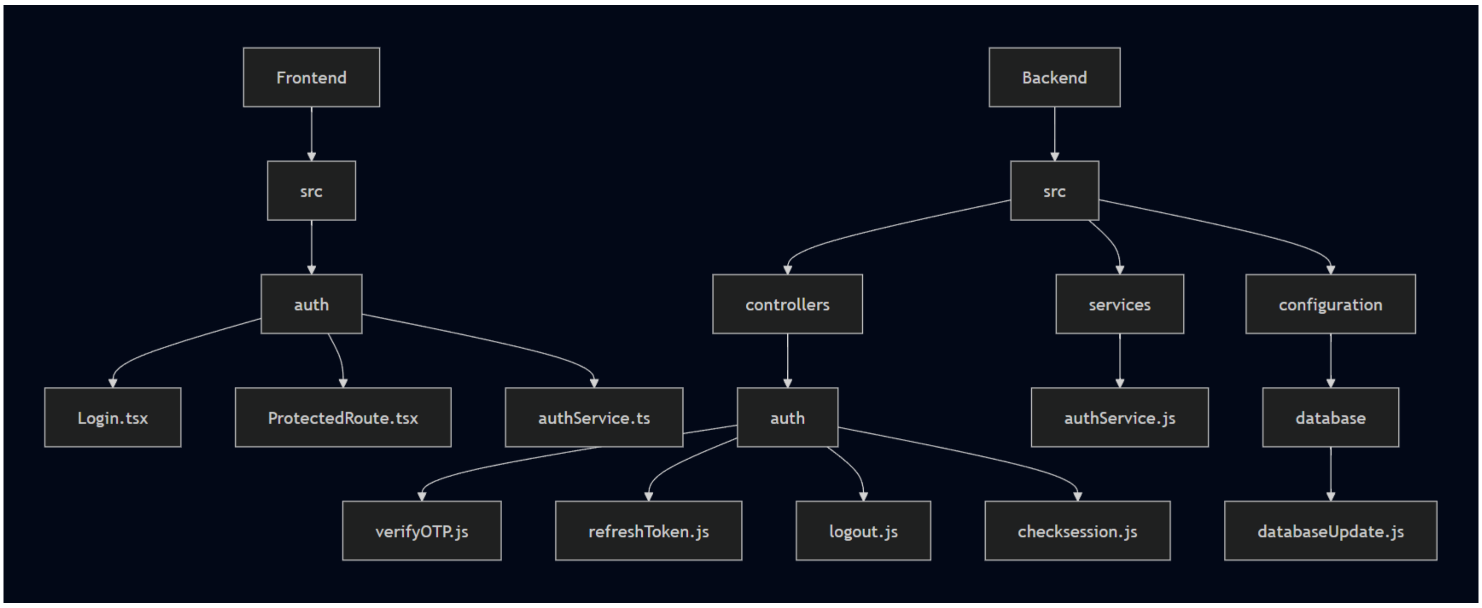




Database ER Diagram



Files Structure and Dependencies



1. System Architecture

The authentication system in SndaramPortal uses a dual-token approach:

- Access Token (short-lived)

- Refresh Token (long-lived)

Key Components:

1. Frontend(React + TypeScript)

   - Handles token storage and management

   - Implements protected routes

   - Manages authentication state

2. Backend (Node.js + Express)

   - Generates and validates tokens

   - Handles token refresh

   - Manages user sessions

2. Token Flow

Initial Authentication Flow:

1. User submits credentials

2. Server validates credentials

3. Server generates:

   - Access Token (JWT)

   - Refresh Token

4. Tokens are sent to client in cookies

5. Client stores tokens securely in cookies

Token Refresh Flow:

1. Access token expires

2. Client sends refresh token

3. Server validates refresh token

4. Server issues new access token via cookies

5. Client updates stored tokens in cookies

3. Security Measures

The system implements several security measures:

- Token expiration
- Secure cookie storage
- Token rotation
- Session tracking
- Input sanitization
- XSS prevention

Implementation Details

1. Backend Implementation

Key Files:

1. verifyOTP.js
   - Handles OTP verification
   - Generates initial tokens
   - Sets up user session

2. refreshToken.js
   - Manages token refresh
   - Validates refresh tokens
   - Issues new access tokens

3. authService.js
   - Core authentication logic
   - Token generation
   - Token validation

2. Frontend Implementation

Key Components:

1. ProtectedRoute.tsx
   - Guards authenticated routes
   - Handles token validation
   - Manages redirects

2. authService.ts
   - Manages token storage
   - Handles token refresh
   - Provides authentication utilities

Token Storage:

- Access Token: Memory/State
- Refresh Token: Secure HTTP-only cookie

3. Session Management

Database Tables:

1. user_sessions
   - Tracks active sessions
   - Stores session metadata
   - Manages session expiration

2. users
   - Stores user credentials
   - Manages user roles
   - Tracks user status

Security Implementation

1. Token Security

Access Token:

- Short expiration (5 minutes)
- Contains minimal user data
- Signed with JWT_SECRET

Refresh Token:

- Long expiration (15 min)
- Stored in HTTP-only cookies
- Signed with REFRESH_TOKEN_SECRET

2. Session Security

Session Tracking:

- Unique session IDs
- Device information tracking
- IP address logging
- Last activity timestamp

Session Invalidation:

- Manual logout
- Token expiration
- Concurrent session limits
- Suspicious activity detection

Error Handling

1. Token Errors

Common Scenarios:

1. Expired Access Token
   - Automatic refresh attempt
   - Redirect to login if refresh fails

2. Invalid Refresh Token
   - Clear all tokens
   - Force logout
   - Redirect to login

3. Token Tampering
   - Immediate invalidation
   - Security logging
   - User notification

2. Session Errors

Handling:

- Session timeout
- Concurrent login conflicts
- Device changes
- IP address changes