Bezpieczeństwo aplikacji mobilnych

Laboratorium 1

Bezpieczne przechowywanie danych i logowanie

Cel:

1. Zademonstrować ryzyko przechowywania tokena „na płasko”.
2. Przenieść token do bezpiecznego magazynu.
3. Dodać weryfikację biometryczną po powrocie aplikacji na pierwszy plan.
4. Dodać odświeżanie sesji po 401 bez zapętleń.

1) Przygotowanie środowiska

1. **Zainstaluj wymagane narzędzia**

- Node.js 18+ (sprawdź: `node -v`)
- Yarn lub npm (sprawdź: `yarn -v` lub `npm -v`)
- Expo CLI (sprawdź: `npx expo --version`)
- Android Studio + emulator **lub** iOS Simulator (macOS)

2. **Utwórz projekt Expo (blank)**

- W katalogu roboczym: `npx create-expo-app@latest –template blank`
- Wejdź do folderu projektu i uruchom: `npx expo start`
- Uruchom na emulatorze.

3. **Przygotuj prosty backend do obsługi logowania**
   Wybierz jedną z opcji:

- **Reqres** (publiczny endpoint do POST /api/login)
- **MockAPI / Beeceptor / Hoppscotch Mock** (utwórz krótkiego mocka zwracającego `{ token: "…" }`)
- **Lokalny mock**: małym serwerem (np. `json-server`) — jeśli wolisz offline.

Warunek akceptacji: Aplikacja startuje w emulatorze, a wybrany endpoint zwraca token w odpowiedzi na poprawne dane.

2) Anty-przykład: ryzykowne przechowywanie

Cel: pokazać, że token zapisany w zwykłej pamięci da się odczytać z systemu plików aplikacji.

1. **Dodaj prosty ekran logowania**

- Formularz z e-mailem i hasłem.
- Po „Zaloguj” — wykonaj żądanie do wybranego endpointu.
- Po sukcesie — zapisz **tymczasowo** token w *nieszyfrowanym* magazynie (np. AsyncStorage) i przejdź do ekranu „Home”.

2. **Uruchom aplikację i zaloguj się** (użyj danych akceptowanych przez Twój mock/Reqres).

3. **Android — znajdź zapisany token w emulatorze**

- Otwórz **Android Studio → View → Tool Windows → Device File Explorer**.
- Rozwiń: data/data/<twój.package.name>/databases/ i files/ (zależnie od magazynu).
- Jeśli użyłeś AsyncStorage: poszukaj bazy **RKStorage** (to SQLite).
- Zapisz screen albo wypisz gdzie znalazłeś dowód (nazwa pliku/bazy, ścieżka).

Warunek akceptacji: Masz dowód (zrzut/ścieżkę), że w nieszyfrowanym magazynie znajduje się token.

3) Migracja: bezpieczne przechowywanie

Cel: przenieść token do bezpiecznego magazynu systemowego (Keychain/Keystore).

1. **Zainstaluj bezpieczny storage**

- Dodaj bibliotekę bezpiecznego magazynu (np. Secure Store/Encrypted Storage).
- Wykonaj instalację zależności i rebuild, jeśli wymagane (na Expo zazwyczaj wystarczy odświeżenie bundla).

2. **Zmień logikę zapisu/odczytu**

- Zastąp zapis tokena do nieszyfrowanego magazynu zapisem do **bezpiecznego**.
- Przy starcie aplikacji: czytaj token z bezpiecznego magazynu, jeśli istnieje → przejdź do „Home”, inaczej pokaż „Login”.

3. **Zweryfikuj efekty**

- Powtórz próbę z **Device File Explorer** (Android) lub kontenerem (iOS).
- Sprawdź, że w poprzednim miejscu (np. RKStorage) token **już się nie pojawia**.

4. **Obsłuż wylogowanie**

- Dodaj akcję „Wyloguj” usuwającą token z bezpiecznego magazynu i czyszczącą stan.

Warunek akceptacji: Token nie występuje w nieszyfrowanym magazynie, logika startu i wylogowania działa poprawnie.

4) Biometria przy powrocie na pierwszy plan

Cel: zabezpieczyć dostęp do ekranu po wznowieniu aplikacji.

1. **Sprawdź wsparcie biometrii**

- Zainstaluj bibliotekę do biometrii (np. Local Authentication).
- Dodaj prosty „guard”: po **powrocie aplikacji do foreground** — wywołuj weryfikację biometryczną **jeśli** użytkownik jest zalogowany.

2. **Android Emulator — konfiguracja odcisku**

- Emulator → **More… (trzy kropki) → Fingerprint** → Dodaj odcisk.
- Przetestuj wyzwolenie biometrii: **home → powrót do aplikacji**.

3. **Ścieżka alternatywna**

- Jeśli urządzenie nie wspiera biometrii lub użytkownik anulował: pokaż PIN/hasło lub cofnij do ekranu logowania.

Warunek akceptacji: Po wznowieniu aplikacji, zalogowany użytkownik musi pozytywnie przejść weryfikację (biometria/PIN), inaczej dostęp jest blokowany.

5) Odświeżanie sesji po 401

Cel: dodać bezpieczny mechanizm „refresh token flow”.

1. **Warstwa sieciowa z interceptorami**

- Dodaj globalną obsługę odpowiedzi: gdy API zwróci **401** i masz *refresh token*, wykonaj **pojedynczą** próbę odświeżenia.
- Zabezpiecz się przed **pętlą** (flaga „w trakcie odświeżania”, kolejka żądań oczekujących, pojedyncza próba na cykl).
- Gdy odświeżenie się nie powiedzie: **wyloguj użytkownika** i wyczyść bezpieczny storage.

Warunek akceptacji: 401 uruchamia **pojedyncze** odświeżenie i bezpieczny powrót do działania; w przeciwnym razie aplikacja wylogowuje.