Phase 4: Feature-Implementierung & Architektur

Zielsetzung

In Phase 4 modularisieren wir die Services, verbessern das State Management, führen eine Clean Architecture ein, erstellen wiederverwendbare UI-Komponenten und etablieren automatisierte Tests in der CI/CD-Pipeline. Jeder Schritt enthält genaue Hinweise zu Pfaden, Dateien und notwendigen Anpassungen.

1. Modularisierung der Services

a) Erstelle für jeden Service ein eigenes Modul:

- `lib/services/auth/auth_service.dart`
  Implementierung:
  - Deklariere abstrakte Klasse `AuthService` im gleichen Ordner.
  - Lege `AuthServiceImpl` in `auth_service.dart` an.
  - Passe `pubspec.yaml` an:
    ```
    flutter:
    assets/services/auth/
    ```
- `lib/services/device/device_service.dart`
  Implementierung: Analog zu Auth, inkl. Methoden `getAllDevices()`, `addDevice()`, `removeDevice()`.
- `lib/services/training/training_service.dart`
  Implementierung: Methoden `startSession()`, `endSession()`, `getSessions()`.
- `lib/services/user/user_service.dart`
  Implementierung: Methoden `getProfile()`, `updateProfile()`.
- `lib/services/affiliate/affiliate_service.dart`
  Implementierung: Falls benötigt, z. B. `getAffiliateLinks()`.

b) **Registrierung in `main.dart`**
Anpassung:

- Importiere alle Services.
- Instanziere einmalig vor `runApp(...)`:
  ```
  AuthService auth = AuthServiceImpl();
  DeviceService device = DeviceServiceImpl();
  // usw.
  ```

2. State Management verbessern

a) Wähle `flutter_bloc` (ohne DI-Framework):

- Füge in `pubspec.yaml` ein:
  ```
  dependencies:
    flutter_bloc: ^8.1.0
  ```

b) Erstelle Bloc-Ordnerstruktur:

- `lib/blocs/auth/`
  `auth_bloc.dart`, `auth_event.dart`, `auth_state.dart`
- Analog: `lib/blocs/device/`, `lib/blocs/training/`, etc.

c) **Implementierung AuthBloc:**

- `AuthEvent`: `LoginRequested`, `LogoutRequested`
- `AuthState`: `AuthInitial`, `AuthLoading`, `AuthAuthenticated`, `AuthError`
- In `auth_bloc.dart`: mapEventToState ruft `AuthService` auf

d) **Verwendung in Widgets:**

- In `lib/screens/login_screen.dart`:

```
BlocProvider(
  create: (_) => AuthBloc(authService),
  child: LoginForm(),
),
```

3. Clean Architecture einführen

a) Erstelle Ordnerstruktur:

```
lib/
  domain/
    usecases/
      get_all_devices.dart
      // weitere UseCases
    models/
      device_model.dart
      // weitere Domain-Modelle
  data/
    repositories/
      device_repository_impl.dart
    sources/
      firestore_device_source.dart
  presentation/
    widgets/
      device_list_widget.dart
    screens/
      dashboard_screen.dart
    blocs/  // siehe Punkt 2
```

b) **UseCase-Beispiel:** `GetAllDevicesUseCase`

- Pfad: `lib/domain/usecases/get_all_devices.dart`
- Code:

```
class GetAllDevicesUseCase {
  final DeviceRepository _repo;
  GetAllDevicesUseCase(this._repo);
  Future<List<Device>> call() => _repo.fetchAll();
}
```

c) **Repository-Implementierung:**

- `lib/data/repositories/device_repository_impl.dart`
- Implementiert `DeviceRepository` und verwendet `FirestoreDeviceSource`

4. Wiederverwendbare UI-Komponenten

a) Lege Ordner `lib/presentation/widgets/common/` an.

b) Erstelle `device_card.dart`:

- Widget mit Parametersatz `Device device`
- Nutzt `Card`, `ListTile` etc.

c) Erstelle `session_overview.dart` mit Semantics-Labels:

- Achte auf `Semantics(label: "...")` für Screen-Reader.

d) Passe in `dashboard_screen.dart` ein:

```
children: [
  DeviceListWidget(),
  SessionOverviewWidget(),
],
```

5. Qualitätssicherung & CI/CD

a) Schreibe erste Unit-Tests:

- Pfad: `test/services/auth_service_test.dart`
- Teste Methoden von `AuthServiceImpl` mit Mocks.

b) Schreibe Widget-Tests:

- Pfad: `test/widgets/login_screen_test.dart`
- Nutze `pumpWidget` und `pumpAndSettle()`.

c) CI/CD-Anpassung:

- GitHub Actions `.github/workflows/flutter.yml` erweitern:

```
- name: Run tests
  run: flutter test
```

- Stelle sicher, dass Tests bei jedem PR automatisch laufen.

Hinweis: Passe bei allen neuen Dateien unbedingt die Importe in `analysis_options.yaml` und deinem `analysis_options.yaml` an, um Lint-Regeln und Namenskonventionen einzuhalten.