Sprint 0: Kickoff & Setup – detaillierte Roadmap

Zeitraum: 4. – 8. Aug 2025 (5 Arbeitstage, Europe/Amsterdam)

Hauptziel: Ein solides Fundament schaffen, indem die Entwicklungsinfrastruktur aufgebaut, Sicherheitsstandards definiert, Basistests geschrieben und eine klare Projektdokumentation erstellt werden. Nach diesem Sprint ist das Team bereit, mit der eigentlichen Funktionsentwicklung in Sprint 1 zu beginnen.
Teamannahme: Mindestens zwei Frontend Entwickler, ein Backend Entwickler/DevOps, ein QA Engineer und ein Product Owner stehen zur Verfügung.
Ergebnisse: Funktionierende CI/CD Pipeline, definierte Code-Review Richtlinien, sichere Firestore Regeln, laufende Basistests und ein gepflegtes README sowie Datenmodell-Dokument.

1. Tag 1 (Mo 4. Aug 2025) – Infrastruktur & Basiskonfiguration

a. Projektstruktur & Repositories

- Aktuellen Zustand der agent-Branch lokal clonen; prüfen, ob die App mit `flutter run` startet.
- `.gitignore` prüfen: Sicherstellen, dass keine Secrets, .env-Dateien, `build/`, `ios/Pods/` etc. eingecheckt werden.

b. CI/CD Pipeline via GitHub Actions (Grundgerüst)

- Neue Workflow Datei `.github/workflows/ci.yml` anlegen.
- Trigger: `on: [push, pull_request]` für alle Branches außer `main` und `production`.
- **Job 1 – Lint & Analyse:**
  - `actions/checkout@v3` zum Code Check out.
  - Flutter via `subosito/flutter-action@v2` (stable channel) installieren.
  - `flutter pub get` ausführen, `flutter format -set-exit-if-changed .`, `flutter analyze`.
- **Job 2 – Unit- & Widget Tests:**
  - Nach dem Lint Job: `flutter test` ausführen.
  - Bei Fehlern CI Status auf "failedßetzen.
- **Job 3 – Build Check (optional):**
  - `flutter build apk -debug` ausführen, um sicherzustellen, dass der Build prinzipiell funktioniert.
- **Pub Pakete mit `actions/cache` cachen** (Key: OS/Flutter-Version/`pubspec.lock`).
- **Job 4 – Code Coverage:**
  - `flutter test -coverage` ausführen; Coverage Report als Artifact hochladen.
- **Secrets & Deploy Stubs:**
  - Secrets wie `FIREBASE_SERVICE_ACCOUNT` später im Repo hinterlegen; in diesem Sprint nur Stub Step "Deploy placeholder".

c. Code-Review Richtlinien einrichten

- `PULL_REQUEST_TEMPLATE.md` in `.github/` anlegen mit Checkliste (Unit Tests geschrieben, Lint bestanden, Screenshots aktualisiert).
- `CODEOWNERS` Datei definieren: Z.B. Änderungen an `lib/features/admin/*` müssen vom Backend Developer, an `lib/features/ui/*` vom UI-Lead reviewed werden.
- Dokument "Development Guidelinesänlegen:
  - Branching Modell (Feature-Branches von agent, Pull Requests nach Review).
  - Commit Konvention (z. B. Conventional Commits).
  - Peer Review Prozess: Mindestens zwei Reviews bei komplexen Features.

2. Tag 2 (Di 5. Aug 2025) – Firestore-Sicherheitskonzept entwerfen

a. Analyse der aktuellen Datenstrukturen

Die Firestore Datenstruktur grob skizzieren:

```
gyms/{gymId}
  - devices/{deviceId}
  - users/{userId}
    - rank/{stats}
    - completedChallenges/{challengeId}
  - feedback/{feedbackId}
  - surveys/{surveyId}/answers/{answerId}
  - challenges/weekly/items/{challengeId}
  - challenges/monthly/items/{challengeId}
  - products/{productId}          // für Affiliate
  - orders/{orderId}              // für Affiliate

users/{userId} (Global User Collection)
  - profile (Name, gymCode, avatar, roles)
  - muscles/{region}
  - badges/{badgeId}
```

Für jeden Pfad festlegen:

- Leserechte?
- Schreibrechte?
- Rollen? (z. B. `role == 'admin'` im `auth.token` für `gyms/gymId`)

b. Konzeption der Multi-Tenancy Rules

- Gym-Isolation: Nutzer dürfen nur auf Daten ihres `gymId`s zugreifen (`request.auth.token.gym_id == gymId`).
- Rollenbasierte Rechte:
  - Admin: darf Geräte anlegen, Umfragen beenden, Branding ändern, Challenges anlegen.
  - User: darf eigene Sessions/Feedback schreiben, Umfragen beantworten.
- User-Subcollection Schutz: Zugriff nur auf eigene Dokumente (`request.auth.uid == userId`).
- Beispiel Regel (Pseudo):

```
match /gyms/{gymId}/devices/{deviceId} {
  allow read: if request.auth != null &&
               request.auth.token.gym_id == gymId;
  allow write: if request.auth != null &&
                request.auth.token.gym_id == gymId &&
                request.auth.token.role == 'admin';
}
```

- Regeln mit dem Team diskutieren und finalisieren.

c. Umsetzung in Code & Deployment

- `firestore.rules` im Projekt ersetzen; development-Regeln entfernen.
- Kopie als `firestore-dev.rules` für lokale Tests ohne Restriktionen.
- Deployment per Firebase CLI: `firebase deploy -only firestore:rules` (zunächst in Test Instanz; Emulator zum Testen verwenden).

3. Tag 3 (Mi 6. Aug 2025) – Basistests entwickeln

a. Unit Tests für Provider

- **DeviceProvider**
  - `loadDevice` testen: Firestore Mock (z. B. `cloud_firestore_mocks`), Device Dokument simulieren; prüfen, ob `device` und `lastSessionSets` gesetzt werden.
  - `saveWorkoutSession` testen: Prüfen, dass Log Dokument erstellt und XP erhöht wird.
- **SurveyProvider**
  - `createSurvey`, `submitAnswer`, `getResults` testen; Optionen-Votes mit Mock Firebase zählen.
- **FeedbackProvider**
  - `submitFeedback`, `loadFeedback`, `markDone` testen.
- **BrandingProvider**
  - Testen, dass Branding korrekt geladen wird; Fehlerbehandlung bei fehlendem Dokument.

b. Widget Tests (Smoke Tests)

- Kleines Widget, das z. B. `HomeScreen` oder `ReportScreenNew` in einen Provider-Wrapper einbettet (`tester.pumpWidget(...)`).
- Erwartung: `DeviceUsageChart` erscheint; bei leerem Dataset wird das Beispiel-Dataset genutzt.

c. Testautomatisierung

- Tests lokal via `flutter test` laufen lassen und über CI ausführen.
- Coverage Report generieren; Ziel: > 30 % Abdeckung.

4. Tag 4 (Do 7. Aug 2025) – Dokumentation & Navigation

a. Datenmodell-Dokumentation

- Datei `docs/data_model.md` erstellen: Jede Firestore Collection und Felder erklären, inkl. Typen, Index Anforderungen und Beziehungen.
- Platzhalter für kommende Features (Affiliate, Lizenzierung) anlegen.

b. Navigationsstruktur dokumentieren

- Datei `docs/navigation.md` erstellen: Routen aus `AppRouter` tabellarisch auflisten (Pfad, Ziel-Screen, Parameter).
- Tab-Bar Aufbau ("Gym", "Profil", "Report", "Muskelgruppen", "Admin", "Rank", "Affiliate", "Pläne") dokumentieren.

c. README & Developer Guide

- README aktualisieren:
  - Kurzbeschreibung des Projekts (Pitch Kernpunkte).
  - Voraussetzungen (Flutter-Version, Dart-Version, Node-Version für Cloud Functions).
  - Setup Anleitung: Firebase CLI installieren, `firebase login`, `flutter pub get`, Emulator starten.
  - Hinweise zu Konfigurationsdateien (`.env`, `firebase_options.dart`).
  - Leitfaden für neue Entwickler (Branch anlegen, Pull Request, Tests schreiben).

5. Tag 5 (Fr 8. Aug 2025) – Review & Sprint Abschluss

a. Firestore Regeln testen (Emulator)

- Firebase Emulator mit Firestore starten (`firebase emulators:start -only firestore`).
- Beispiel Anfragen mittels `curl` oder `@firebase/rules-unit-testing` ausführen:
  - Admin legt ein Gerät im eigenen Gym an → erlaubt.
  - User ohne Admin-Rolle legt Gerät an → abgelehnt.
  - User von Gym A liest Gerät von Gym B → abgelehnt.

b. Code Review & Aufgaben für Sprint 1

- Internes Meeting: Alle Änderungen gemeinsam reviewen; Einhaltung der Development Guidelines prüfen.
- Offene Fragen klären; Aufgaben für Sprint 1 definieren (Feedback-Workflow, Benachrichtigungen etc.).

c. Sprint Retro & Lessons Learned

- Was lief gut? (z. B. reibungsloser CI Setup)
- Was lief schlecht? (z. B. Probleme bei Firestore Mocks)
- Verbessern: z. B. Pair Programming früher beginnen.

d. Backlog Pflege

- Alle noch offenen oder neuen Aufgaben (fehlende Tests, Dokumentationsergänzungen) ins Product Backlog eintragen.
- Sprint 1 Plan bestätigen: Wer übernimmt Feedback-Workflow, Benachrichtigungen etc.

6. Erwartetes Ergebnis am Ende von Sprint 0

- **CI/CD Pipeline:** Lint, Tests und Build werden bei jedem Push automatisch ausgeführt; Fehlermeldungen erscheinen im Pull Request.
- **Code-Review Prozess:** Klare PR Vorlage und `CODEOWNERS` Datei vorhanden; Peer Review etabliert.
- **Sichere Firestore Regeln:** Daten sind pro Gym isoliert, Rollen werden berücksichtigt; Tests decken Grundszenarien ab.
- **Basistest Suite:** Erste Unit- und Widget Tests für Kernprovider laufen; Coverage Report erzeugt.
- **Dokumentation:** Datenmodell und Navigationsstruktur sind dokumentiert; README enthält Setup- und Contributing Hinweise.
- **Team Alignment:** Alle Beteiligten kennen Projektaufbau, Richtlinien und Aufgaben für Sprint 1.

Mit dem Abschluss dieses Sprints ist das Team bestens vorbereitet, um in Sprint 1 mit der Entwicklung des Feedback- und Survey-Workflows fortzufahren.