Sprint 1: Feedback- und Umfrage-Workflow – detaillierte Roadmap

Zeitraum: 11.–15. Aug 2025 (5 Arbeitstage)

Ziel: Den Feedback- und Umfrage-Workflow abschließen, sodass Nutzer Feedback mit Anhängen und Kategorien einreichen können, Studio-Admins Benachrichtigungen erhalten, Umfragen automatisch erinnern und die Report-Seite die Anzahl offener Feedbacks/Umfragen per Badge anzeigt. Danach kann nahtlos mit Sprint 2 (Offline-Sync & NFC-Verbesserungen) begonnen werden.

1. Voraussetzungen und Kontext

- **Sprint 0 Abschluss:** CI/CD-Pipeline, Firestore-Regeln, Basistests und Dokumentation sind vorhanden. Das Repository enthält bereits einen Feedback-Provider und Survey-Provider (inkl. Basisscreens).
- **Team:** Mindestens zwei Frontend-Entwickler, ein Backend-Entwickler (für Cloud Functions-/FCM), ein QA-Engineer.
- **Systeme:** Firebase Cloud Messaging (FCM) konfiguriert, Firebase Cloud Functions nutzbar, lokales Testen mit Emulator möglich.

2. Umfangreiche Aufgabenplanung (Tag-für-Tag)

Tag 1 (Mo 11. Aug 2025) – Anforderungsanalyse & Schema-Erweiterung

Detail-Workshop zu Requirements (AM):

- Sammeln der genauen Anforderungen von Product Owner/Studio-Admins: Welche Feedback-Kategorien sollen existieren? (z. B. „Geräteproblem“, „Sauberkeit“, „Service“).
- Welche Medientypen sollen als Anhang erlaubt sein? (Fotos, evtl. kurze Videos) – aufgrund Bandbreite zunächst auf Bilder begrenzen.
- Definieren, welche Rollen Benachrichtigungen erhalten (Admins aller Gyms vs. nur des betreffenden Gyms).

Datenmodell-Anpassung (Nachmittag):

- Feedback-Einträge erweitern:
  - In `lib/features/feedback/models/feedback_entry.dart` ein Feld `category` (String) und ein Feld `attachmentUrl` (String?) ergänzen.
  - Firestore-Sammlung `gyms/{gymId}/feedback/{feedbackId}` um diese Felder erweitern.
- Client-Modell anpassen:
  - `FeedbackProvider.submitFeedback` so erweitern, dass vor dem Schreiben ein optionaler Upload (siehe unten) stattfindet und dann `category` und `attachmentUrl` mitgespeichert werden.
- Storage-Pfad definieren: Bilder sollen in Cloud Storage unter `feedback/{gymId}/{feedbackId}.jpg` abgelegt werden.

Upload-Strategie (Nachmittag):

- Einbindung von `image_picker` oder `file_picker` in Flutter zur Auswahl eines Fotos im FeedbackButton-Dialog.
- Implementieren eines `uploadAttachment`-Helper, der das ausgewählte Bild in Firebase Storage hochlädt und die Download-URL zurückliefert.
- Optional: Bild komprimieren (z. B. mittels `flutter_image_compress`), um Speicher zu sparen.

UI-Update – Feedback-Dialog:

- Im FeedbackButton-Dialog Kategorie-Drop-down (`DropdownButtonFormField`) mit vordefinierten Kategorien einfügen.
- Button „Foto hinzufügen“ integrieren, der den Bild-Picker öffnet und ein Thumbnail anzeigt.
- Bei Klick auf „Senden“:
  - Foto (falls vorhanden) hochladen und URL merken.
  - `submitFeedback` mit `category` und `attachmentUrl` aufrufen.
- Fehler-Handling und Ladeindikator im Dialog (Disabled-State während Upload).

Tag 2 (Di 12. Aug 2025) – Benachrichtigungen via FCM/E-Mail

Cloud-Function für Feedback-Benachrichtigungen:

- In `functions/index.js` (oder TypeScript-Äquivalent) einen neuen Trigger `onCreate` auf `gyms/{gymId}/feedl` schreiben.
- Logik: Lese `gymId` und `category`, erzeuge eine Benachrichtigungsnachricht („Neues Feedback im Gym X in Kategorie Y“).
- Rufe FCM-API an: Ziel sind alle Geräte mit `role == 'admin'` und `gym_id == gymId` (Topics oder Token-Liste).
- Optional: Fallback-E-Mail über SendGrid/Mailgun, wenn FCM-Token nicht vorhanden.

Cloud-Function für Survey-Benachrichtigungen:

- `onCreate`-Trigger auf `gyms/{gymId}/surveys/{surveyId}`: sendet Push „Neue Umfrage verfügbar“.
- `onUpdate`-Trigger: prüft, wenn Feld `status` auf „abgeschlossen“ springt → Benachrichtigung „Umfrage abgeschlossen. Ergebnisse verfügbar“.

Firebase Cloud Messaging im Client einrichten:

- Sicherstellen, dass `firebase_messaging` im Flutter-Projekt installiert und initialisiert ist (in `main.dart`).
- Registrieren auf Topic `admins_{gymId}` für Admin-Benutzer; `members_{gymId}` optional für Survey-Hinweise.
- Umgang mit Benachrichtigungen im Vorder- und Hintergrund: Lokale Anzeige via `flutter_local_notificati`
- QA-Engineer testet Benachrichtigungswege mit Emulator und echten Geräten; Dokumentation der erforderlichen APNs-/FCM-Keys (iOS/Android).

Tag 3 (Mi 13. Aug 2025) – Erinnerungslogik im Survey-Provider

Reminder-Konzept:

- Ziel: Nutzer, die die Umfrage noch nicht beantwortet haben, sollen nach x Tagen eine Erinnerung bekommen.
- Entscheidung: FCM-Benachrichtigung durch Cloud-Function vs. lokale Notification im Client.
- Für die Pilotphase: Cloud-Function `surveyReminders` einmal täglich per `functions.pubsub.schedule('ever 24 hours')` ausführen.

Implementierung der Cloud-Function:

- Iteriert über alle offenen Surveys (`status == 'open'`) in jedem Gym.
- Für jede Umfrage: liest `answers`-Subcollection, bildet Liste der `userIds` mit Antwort.
- Fragt per Firestore die `gyms/{gymId}/users` (oder separate Users-Sammlung) ab, wählt alle, die noch keine Antwort gegeben haben.
- Sendet FCM-Reminder an diese Nutzer („Du hast noch nicht an der Umfrage XYZ teilgenommen“).
- Optional: Nur wenn Umfrage älter als definierte Frist (z. B. 3 Tage) und vor Ablauf der Umfrage.

Anpassung des SurveyProviders im Flutter-Client:

- Neuer boolescher Wert `hasVoted` pro Survey im Client speichern.
- Beim Laden der Surveys (bereits vorhanden in `listen`) gleichzeitig `currentUserId` mit den Antworten abgleichen, damit die UI weiß, ob der aktuelle User bereits abgestimmt hat.
- In der Survey-Liste ein Badge „Abgestimmt“ oder „Offen“ anzeigen.

Tag 4 (Do 14. Aug 2025) – Widget-Tests & Qualitätssicherung

Widget-Tests für Feedback-Dialog:

- Testfall 1: Öffnen des Feedback-Dialogs → Kategorie-Drop-down und Foto-Button sichtbar.
- Testfall 2: Auswahl einer Kategorie und Hinzufügen eines Bildes (Mock-Picker) → Thumbnail erscheint.
- Testfall 3: „Senden“ ohne Kategorie/Bild → Fehlermeldung bzw. notwendige Validierung.
- Testfall 4: Erfolgreiches Senden → Provider-Methode wird aufgerufen und Snackbar „Feedback gesendet“ angezeigt.

Widget-Tests für Survey-Screens:

- `SurveyOverviewScreen`: Prüfen, dass offene vs. abgeschlossene Umfragen in den beiden Tabs korrekt gerendert werden (verwende Mock-Provider).
- `SurveyVoteScreen`: Auswahl einer Option aktiviert den Absenden-Button; nach Absenden zeigt die Seite „Danke für deine Teilnahme!“
- **SurveyReminder-Badge:** Falls `hasVoted == false`, wird „Abstimmen“-Badge oder farbliche Markierung angezeigt.

CI-Integration:

- Alle neuen Widget-Tests müssen lokal bestehen und in GitHub Actions ohne Flaky-Fehler laufen.
- QA-Engineer fügt Test-IDs (`Key()` Widgets) dort hinzu, wo das Testing sonst schwer ist.

Tag 5 (Fr 15. Aug 2025) – Report-Seite & Abschluss

Badge-Darstellung in der Report-Seite:

- Im `ReportScreenNew` (line 30ff) die Anzahl offener Feedback-Einträge (`openEntries.length`) und offener Umfragen (`openSurveys.length`) als Badge visualisieren.
- Beispiel: neben dem `ListTile`-Titel eine farbige `CircleAvatar` mit Zahl einfügen.
- Falls 0, Badge ausblenden oder dezent grauen Hinweis „Kein offenes Feedback“ beibehalten.

Finale Code-Review & Refactoring:

- Gemeinsamer Review aller Feedback- und Survey-Änderungen; Prüfung auf Einhaltung der Coding-Guidelines.
- Sicherstellen, dass neu eingeführte Felder (`category`, `attachmentUrl`) rückwärtskompatibel sind und keine bestehenden Daten brechen.
- Tests für die Cloud Functions lokal im Emulator durchführen und Logs überprüfen.

Dokumentations-Update:

- Ergänzung der Datenmodell-Dokumentation um `category`, `attachmentUrl` und Reminder-Funktionen.
- Anleitung für Studio-Admins erstellen, wie man Push-Benachrichtigungen testet und worauf bei Feedback-Kategorien zu achten ist.

Sprint-Review & Übergabe in Sprint 2:

- Showcase: Feedback mit Foto und Kategorie senden → Admin erhält Push & Badge erscheint im Report.
- Umfrage erstellen → Nutzer erhält Push; erinnert nach N Tagen, wenn noch nicht abgestimmt.
- QA-Bericht: Liste der erkannten Bugs & gefixten Issues.
- Backlog-Aufgaben für Sprint 2 sammeln (z. B. noch fehlende Bilderskalierung, Sonderfälle in der Erinnerungslogik).

3. Auslieferungs- und Qualitätskriterien (Definition of Done)

- **Feedback:** Nutzer können ein Foto und eine Kategorie im Feedback-Dialog auswählen; Daten landen in Firestore und Storage; Admin-Benachrichtigung trifft ein.
- **Umfragen:** Ersteller erhält E-Mail/Push bei Erstellung und Abschluss; Reminder werden zu definierten Zeiten an nicht abstimmende Nutzer gesendet.
- **Report-Seite:** Zeigt in der Liste „Feedback“ und „Umfragen ansehen“ die Anzahl offener Einträge als Badge.

- **Tests:** Neue Unit- und Widget-Tests sind geschrieben und laufen grün in CI; Functions sind mit Emulator getestet.
- **Dokumentation:** Datenmodell, Ablauf der Benachrichtigungen und Reminder sind aktualisiert; README ergänzt.
- **Code-Qualität:** Lint-Regeln bestehen, Pull-Requests sind reviewed und freigegeben.

4. Nächste Schritte

Mit Abschluss von Sprint 1 ist die Feedback- und Umfrage-Funktionalität robust und benutzerfreundlich. Das Team kann anschließend in Sprint 2 (18.–22. Aug 2025) den Fokus auf die Stabilisierung des Offline-Sync und der NFC-Interaktionen legen, da das Fundament für Kommunikation und QA mit Nutzern gesichert ist.