Brew Metrics - Developer Handoff & Onboarding Guide (v0.3.2)

Date: April 18, 2025 (Simulated Date from Logs)
Current Version: Approx. 0.3.2 (Enhanced Batch Creation, Feedback Aggregation Logic, `isActive` Toggle Implemented)
Handing Off From: Previous AI Developer Session
Handing Off To: New AI Developer

1. Introduction & Project Overview

(Adapted from initial Onboarding Guide)

- **Project Name:** Brew Metrics
- **Website:** www.brewmetrics.xyz (Placeholder)
- **Repository:** `brewmetics-app` (Assumed GitHub: `https://github.com/your-org/brewmetics-app.git` - Replace `your-org` if known)
- **Purpose & Vision:** To create a web-based tool for microbreweries to easily collect and analyze batch-specific customer feedback via web forms. The tool aims to simplify feedback collection using a persistent QR code approach and provide actionable insights through data aggregation. A key differentiator is incorporating patron education ("Flavor School") into the feedback process.
- **Current State:** The application allows brewers to manage detailed batch information (including an active/inactive status), submit detailed flavor feedback via a public form, and view aggregated feedback summaries on their dashboard. Core technical integrations (Auth, Firestore Client SDK, Hosting) are validated. Cloud Functions exist but are minimally used by the current client implementation and have known issues.

2. Technical Stack & Setup

- **Frontend:** HTML, CSS, Vanilla JavaScript (ES Modules)
- **Backend:** Firebase Cloud Functions (Node.js) - *Currently only*
- **Database:** Firestore (NoSQL)
- **Authentication:** Firebase Authentication (Email/Password)
- **Hosting:** Firebase Hosting
- **Storage:** Firebase Storage *(Planned/Available, not yet used)*
- **Development Environment: Firebase Emulator Suite (MANDATORY)**
  - **Services Used:** Auth, Firestore, Hosting, Functions
  - **Command to Start:** `firebase emulators:start` (run from project root)

    - **Accessing App:** `http://localhost:5000` (or configured Hosting port)
    - **Accessing Emulator UI:** `http://localhost:4000` (or configured UI port)
-

3. Repository & Project Association

- **Repository:** `brewmetics-app` (See Project Overview)
- **Firebase Project:** Associated via `.firebaserc`. Assumed ID: `brewmetics-app` (Verify with `firebase use`). Login using `firebase login`.

4. Key Files & Code Structure

```
brewmetics-app/
├── functions/              # Firebase Cloud Functions
│   ├── index.js          # Functions: createUserRecord (Buggy), others likely unused by current client
│   ├── package.json      # Function dependencies
│   └── ...
├── public/               # Firebase Hosting root
│   ├── assets/
│   │   ├── css/
│   │   │   └── styles.css   # Main stylesheet
│   │   └── js/
│   │       ├── app.js      # *** Core Dashboard Logic *** (Batch CRUD, isActive toggle, Feedback
Aggregation/Display, Auth Listener)
│   │       ├── auth.js     # Auth functions (signup, login, logout) - Called by login/signup pages
│   │       ├── feedback-form.js # Logic for the public feedback page (dynamic form build, submission)
│   │       ├── firebase-config.js # *** Firebase SDK Initialization *** (Connects to PROD or EMULATORS)
- Check config!
│   │       └── flavor-data.js # Static definitions for flavor categories/descriptors/tips
│   ├── brewery/
│   │   └── dashboard.html # Brewery's private dashboard page
│   ├── feedback/
│   │   └── index.html    # Public feedback form page
│   ├── index.html        # Entry point (redirects based on auth state via app.js)
│   ├── login.html        # Login page (uses auth.js)
│   └── signup.html        # Signup page (Implied, uses auth.js)
│   # --- NOTE: Select-batch page files do NOT exist yet ---
│   # ├── select-batch/
│   # │   └── index.html    # (To be created)
│   # └── assets/js/select-batch.js # (To be created)
│
├── firestore.rules        # Firestore security rules (Current Secure Version)
├── firebase.json          # Firebase project/emulator configuration
└── .firebaserc          # Firebase project association
└── README.md             # (Assumed)
└── ...                  # Other config files (.gitignore, etc.)
```

content_copy

download

Use code with caution.

5. Current Functionality (What's Working)

- **Authentication:** Signup, Login, Logout using Email/Password. Redirects handled via `auth.js` calls and `app.js` `onAuthStateChanged` listener.
- **Batch Creation (Enhanced):** Dashboard form allows creating batches with Name (required), `isActive` (defaults true), Style, ABV, IBU, Description, Brewer's Notes, Incentive Text (all optional). Data saved correctly to Firestore.
- **Batch Listing (Enhanced):** Dashboard lists batches for the logged-in brewer, displaying Name, Created Date, Style, ABV, IBU, Description, Active Status ("Active for Feedback" / "Inactive"), the specific Feedback URL, and an Activate/Deactivate toggle button.
- **Batch Activation Toggle:** The Activate/Deactivate button on the dashboard successfully updates the `isActive` boolean field for the batch in Firestore and refreshes the list display.
- **Feedback Form:** Public page accessible via URL (`?batchId=...`). Loads Batch Name. Dynamically generates UI for all flavor categories/descriptors from `flavor-data.js`.
- **Feedback Submission:** Form saves all ratings (`overallRating`, nested `flavorSelections` map), `batchId`, `breweryId`, `timestamp`, and optional `comment` to Firestore.
- **Feedback Aggregation Logic:** `app.js` successfully fetches all feedback for a batch and calculates aggregate data (count, avg overall, avg per flavor descriptor) and collects comments when "View Feedback" is clicked.
- **Feedback Display (Partial - Emulator Issue):** The *rendering logic* in `app.js` to display the aggregated feedback summary is implemented. **However, this display currently fails in the emulator due to the known Firestore Emulator Read Rule issue (See Known Issues #1).** It works correctly if the security rule is temporarily relaxed.
- **Incentive Display:** Post-feedback submission screen (`feedback/index.html`) shows `incentiveText` if configured for the batch.

6. Data Model (Firestore)

- **collection:**
  - Doc ID: Auto-generated

- Fields: `batchName` (string), `breweryId` (string), `creationDate` (timestamp), `isActive` (boolean), `style` (string, opt), `abv` (number, opt), `ibu` (number, opt), `description` (string, opt), `brewersNotes` (string, opt), `incentiveText` (string, opt)

-
- **collection:**
  - Doc ID: Auto-generated
  - Fields: `batchId` (string), `breweryId` (string), `timestamp` (timestamp), `overallRating` (number), `comment` (string, opt), `flavorSelections` (map: `{ categoryId: { descriptorId: rating (number) } }`)
-
- **collection:** (Created by `createUserRecord` function)
  - Doc ID: Firebase Auth User ID (uid)
  - Fields: `email` (string), `createdAt` (timestamp)
-

7. Security Rules (`firestore.rules`)

(Current secure version that causes emulator read issue but is correct for production)

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /breweries/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }
    match /batches/{batchId} {
      allow read: if true;
      allow create: if request.auth != null && request.resource.data.breweryId == request.auth.uid;
      allow update, delete: if request.auth != null && resource.data.breweryId == request.auth.uid;
    }
    match /feedback/{feedbackId} {
      allow create: if true;
      allow read: if request.auth != null && resource.data.breweryId == request.auth.uid; // Secure rule
      allow update, delete: if false;
    }
    match /test_collection/{testDocId} { // Optional
      allow create: if request.auth != null;
    }
  }
}
```

content_copy

download

Use code with caution.Firestore

8. API Keys / Credentials / Environment Variables

- **:** Contains Firebase project configuration (apiKey, authDomain, projectId, etc.). **MUST BE PROTECTED.** Ensure these are configured correctly for the **Emulator** during local development. For deployment, replace with production keys using environment variables or a secure build process. **DO NOT COMMIT PRODUCTION KEYS.**

9. Development Environment Setup

1. **Prerequisites:** Node.js (v14+), npm (v6+), Firebase CLI (`npm install -g firebase-tools`)
2. **Clone Repository:** `git clone ...` & `cd brewmetics-app`
3. **Install Dependencies:** `cd functions && npm install && cd ..`
4. **Firebase Login & Project:** `firebase login`, `firebase use <PROJECT_ID_OR_ALIAS>`
5. **Run Emulators:** `firebase emulators:start`
6. **Access:** App: `http://localhost:5000`, Emulator UI: `http://localhost:4000`

10. Current Known Issues & Challenges

1. **Firestore Emulator Read Rule Discrepancy (BLOCKER FOR TESTING FEEDBACK VIEW):** The secure Firestore `read` rule for `/feedback` (`resource.data.breweryId == request.auth.uid`) consistently fails during list operations (`getDocs`) within the **emulator**, preventing aggregated feedback from being displayed on the dashboard. Confirmed code and data are correct when rule is temporarily relaxed. **Current Approach:** Maintain the secure rule; acknowledge feedback display testing *in the emulator* will fail; trust it will work in production.
2. **Cloud Function Error (** `functions/index.js` crashes on user creation (`TypeError: Cannot read properties of undefined (reading 'serverTimestamp')`). Needs fix for Admin SDK initialization/usage. **Low priority.**
3. **Client vs. Server Logic:** The original onboarding guide implies more Cloud Function usage (e.g., `createBatchMinimal`). The current client (`app.js`, `feedback-form.js`) interacts directly with Firestore via the client SDK for most operations (batch create, feedback submit). The functions in `index.js` (other than `createUserRecord`) may be unused/outdated

relative to the client implementation. This should be reviewed later for potential refactoring (e.g., moving logic to backend functions for security/validation).

4. **Missing** Visually confirm the `overall-rating` slider exists in `feedback/index.html`.
5. **Feedback Form Enhancement Pending:** Feedback form doesn't display `brewersNotes` from the batch document yet.
6. **Emulator Performance:** Occasional slowness observed during login and Firestore reads in the emulator environment. Restarting emulators often helps.

11. Path Forward: Next Steps

Based on the decision to implement a single QR code linking to a batch selection page:

Task: Implement Batch Selection Page

- **Goal:** Create a public page that lists active batches for a specific brewery, allowing patrons to select one and be redirected to the appropriate feedback form.
- **Steps:**
    1. **Create File:** Create `public/select-batch/index.html`. Add basic HTML structure (title, container div `#active-batches-list`, script tag).
    2. **Create File:** Create `public/assets/js/select-batch.js`.
    3. **Implement JS (**
        - Import Firebase config (`db`) and Firestore functions.
        - Read `breweryId` from URL query parameter (`?breweryId=...`). Handle missing ID.
        - Query `batches` collection: `where("breweryId", "==", breweryId)`, `where("isActive", "==", true)`, `orderBy("creationDate", "desc")`.
        - Handle empty results (no active batches).
        - Dynamically generate HTML in `#active-batches-list` for each active batch (display Name, Style, ABV). Make items clickable/tappable.
        - Add event listener to batch items: on click, get the `batchId` and redirect using `window.location.href =` `\`/feedback/index.html?batchId=${selectedBatchId}`;`.
    4.
    5. **Update Dashboard (** Add a section to the main dashboard area (outside the batch list) that clearly displays the brewery's unique "Select Batch URL" for them to copy (e.g., `Your public batch selection link: <link>`). This link will include their specific `breweryId`. (QR code generation for this link can be a later enhancement).

●