Northeastern University
CS5200 Database Management System
Spring 2025
Professor Derbinsky
Erdun E
Apr 17, 2025

Final Project Report

1 Abstract

Many coffee drinkers lack a structured way to track their tasting experiences and discover new coffees they might enjoy. The Bean Vibes App addresses this gap. It is a full-stack mobile application built to support specialty coffee enthusiasts in recording, analyzing, and exploring their tasting journeys. By offering a structured and intuitive logging platform, the system tackles the challenge of managing complex, multi-entity data, including roasters, products, beans, brew parameters, and personal preferences. Users can create detailed tasting logs with inputs such as brew method, grind size, bloom count, and rating, while also marking disliked or recently used items to streamline future entries.

Beyond logging, the app features a dynamic Explore page that showcases personalized recommendation cards based on user behavior and flavor preferences. These include curated suggestions like "Fruity Coffees," "Top Rated," or "Cold Brew Picks," powered by backend-native SQL queries and projection-based recommendation logic. The backend, built with Spring Boot, interacts with a normalized MariaDB database and exposes RESTful APIs consumed by the Kotlin-based Android frontend.

This report presents the system's motivation, architecture, database design, and personalized recommendation algorithms, along with screenshots, ER diagrams, and a retrospective of the development process.

2 Problem Description and Assumptions

Many coffee lovers enjoy exploring different beans, brewing methods, and roasters—but it's hard to remember what they've tried, what they liked, and how each cup turned out. The Bean Vibes App helps solve this problem by giving users a structured way to log their coffee tastings, track patterns in their preferences, and discover new products they're likely to enjoy. Instead of scribbled notes or scattered memories, everything is stored in one connected system.

At the heart of the system are entities like User, TastingLog, Roaster, Product, Bean, and RoastBatch. A user creates a tasting log for a specific brew session, which is linked to a unique RoastBatch that represents one roasting cycle of a product (same product, different roast date). Each Product belongs to a Roaster, and each product may be a blend of multiple Beans. Beans themselves have attributes like varietal, process method, and farm origin. These connections are captured using standard foreign key constraints and bridge tables like ProductBean (for blend composition) and TastingLogNote (to tag flavors experienced in a log).

Users can interact with their data by searching past logs, filtering by brew method, rating, or bean, and browsing personalized suggestions. The Explore page includes dynamic recommendation cards like "Fruity Coffees" or "Cold Brew Picks," powered by backend SQL queries that look at users' brewing history, recent behavior, and rated preferences. Additional tables like UserRoasterStatus or UserBeanStatus store whether a user disliked something or used it recently, and feed into the recommendation logic.

We assume each tasting log refers to a single brewing session and that users prefer filling structured fields over writing open-ended notes. We also assume that roasters, products, and beans are shared across users, while logs and preferences are user-specific. The database design follows third normal form (3NF), supports referential integrity, and uses indexes and bridge tables to optimize for both personalization and analytical use cases.

Altogether, this enables users not only to document their experiences but also to evolve their tasting journey over time with smart, data-driven support.

3 ER Diagrams

The complete Entity-Relationship Diagram (ERD) is included as a high-resolution image titled `02_CS5200_Erdun_E_ERD_V6.jpeg`, and models all core entities and relationships across the Bean Vibes system.

3.1 Global Overview

The ERD captures key real-world concepts, including:

- **Users** and their interaction with Tasting Logs, personalized status tracking (e.g. dislike/recent flags) across Roasters, Products, and Beans.
- **Roasters, Products, Beans** organized in a hierarchical structure, where each product belongs to a roaster, and each product is linked to one or more beans via `ProductBean`.
- **Roast Batches** the join point that connects a Product to a physical roasting event, acting as the bridge between Products and TastingLogs.
- **Tasting Logs** capturing user-input logs, connected to brew parameters, rating, notes, and the selected RoastBatch.
- **TastingNote, Varietals, Farms, Location Hierarchy** modularized into lookup tables to support advanced reporting and search.

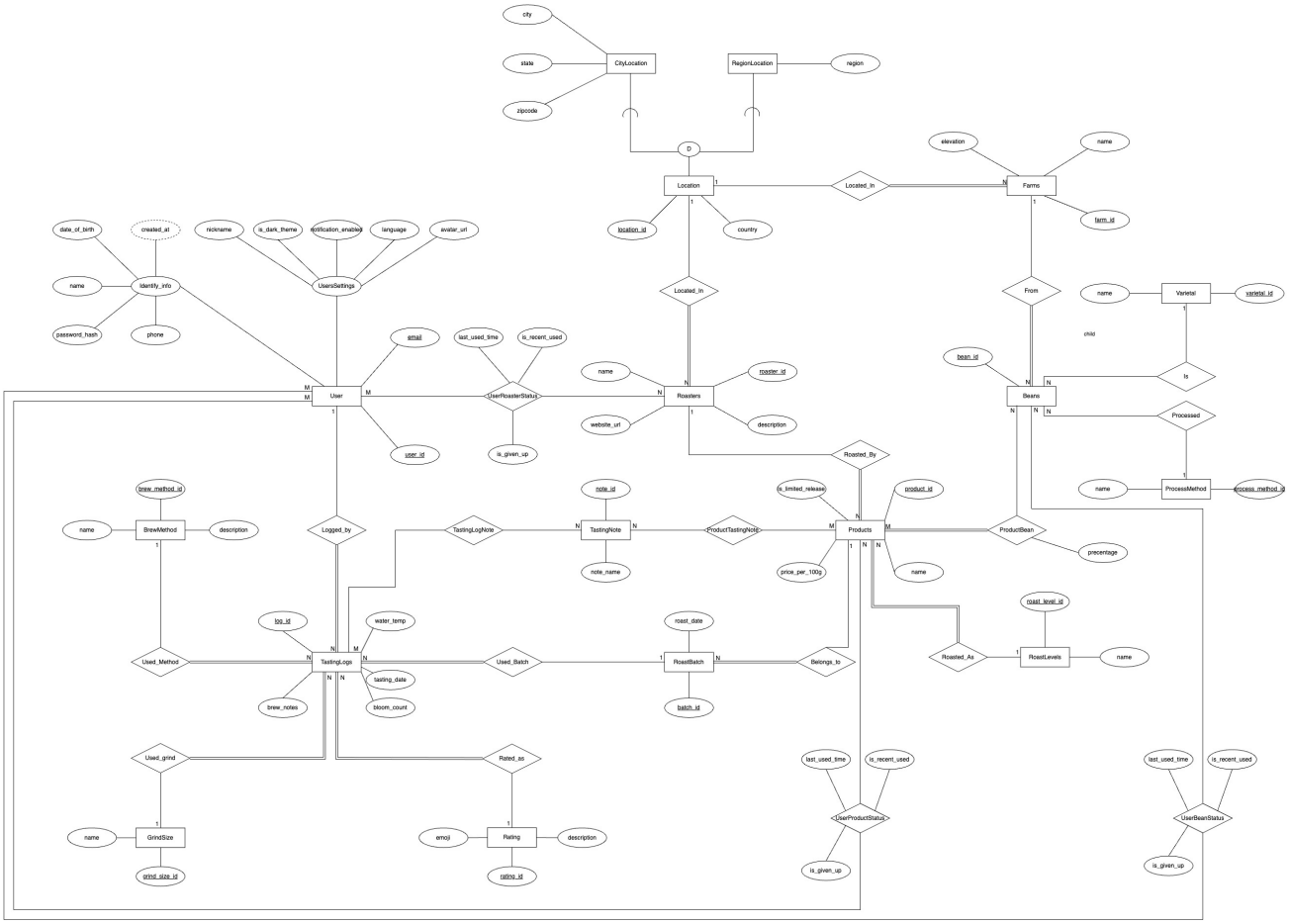
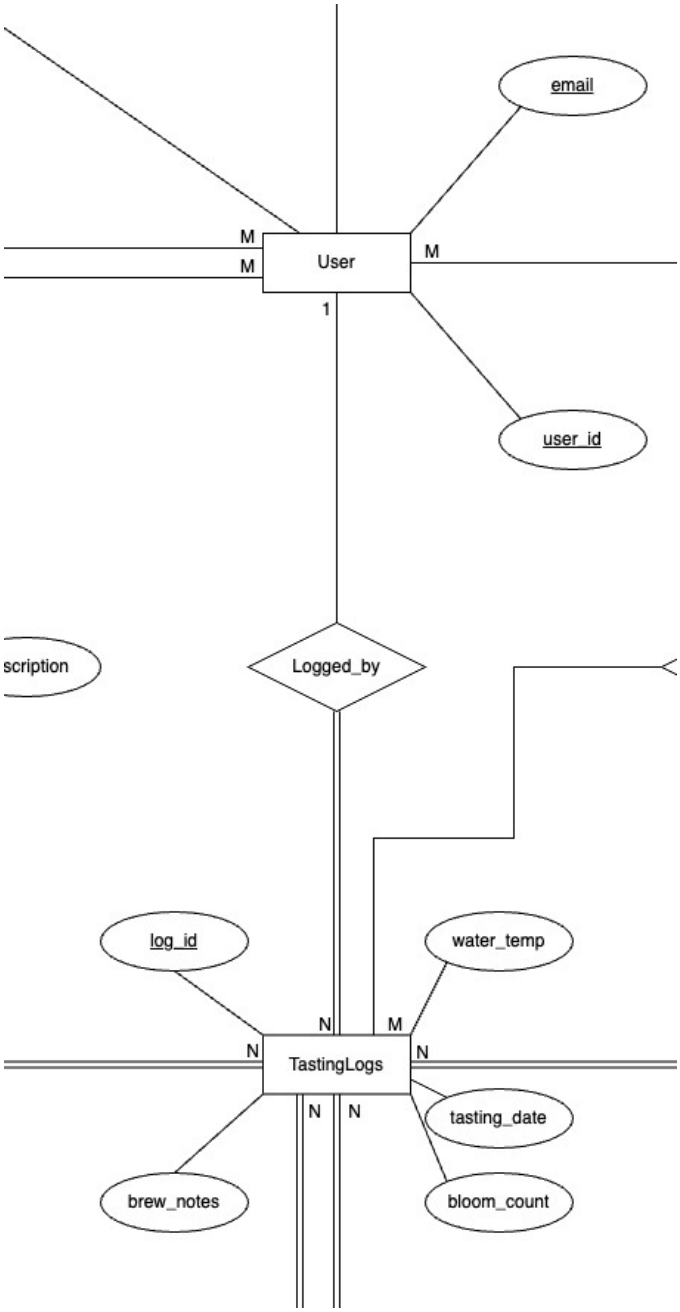


Figure 1: Global Overview

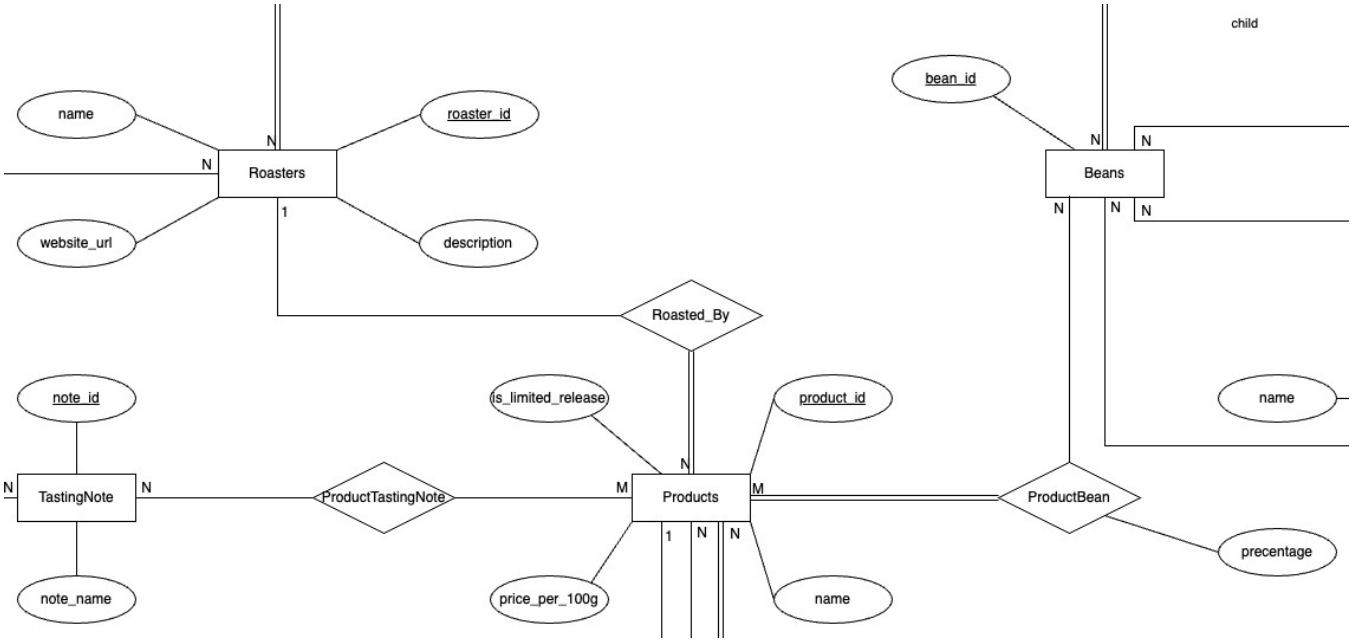
3.2 Modular Local Views

To enhance readability and reduce complexity, several logical modules within the ERD are designed as semi-independent views:

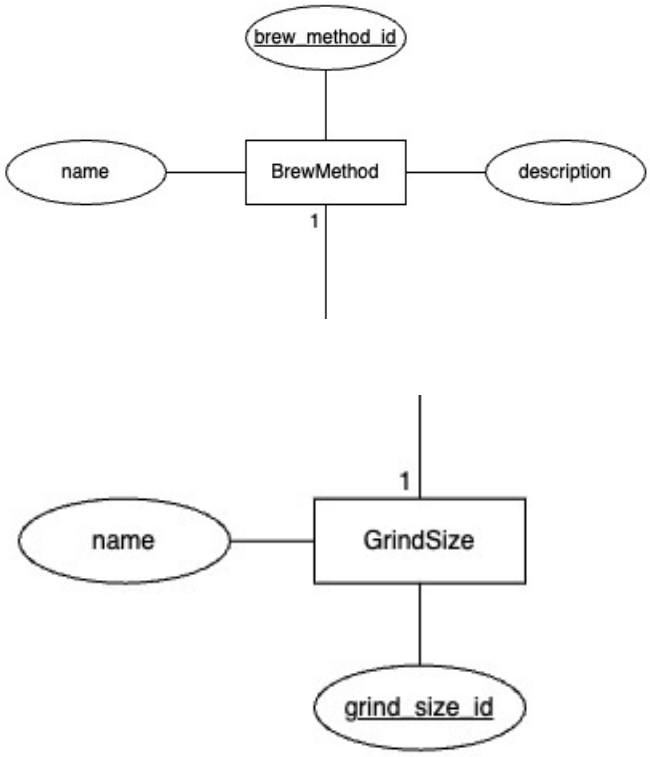
1. **User + Tasting Log Module:** Tracks user logs, ratings, notes, and brew parameters.



2. **Roaster/Product/Bean Chain:** Represents the sourcing pipeline and product composition.



3. **Lookup Tables:** Includes BrewMethod, GrindSize, Rating, Varietal, etc.



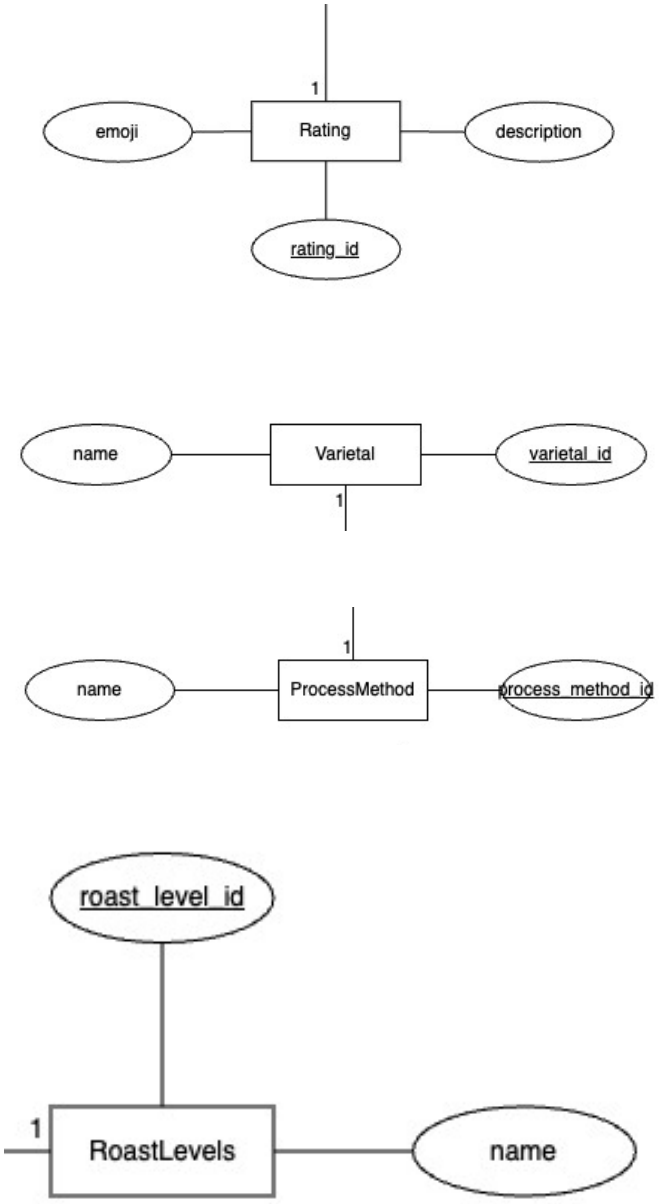


Figure 2: Enter Caption

4. **Location + Farm Structure:** Shows global origin tracking and farm-region-location mapping.

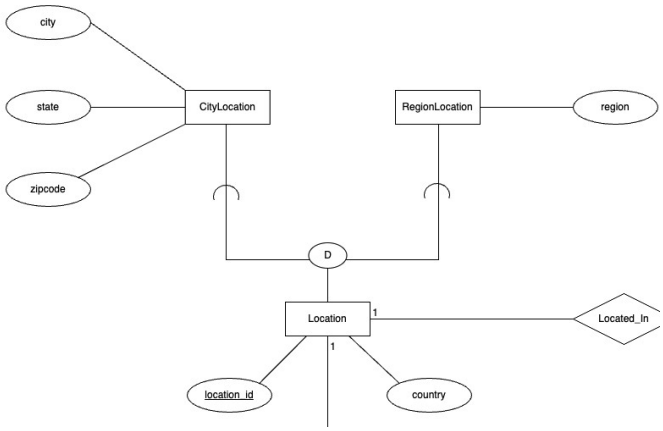


Figure 3: Enter Caption

3.3 Design Notes

The schema is fully normalized up to Third Normal Form (3NF), ensuring:

- Each attribute depends on the whole key, and nothing but the key.
- Multi-valued attributes are separated into join tables (e.g., `TastingLogNote`, `ProductBean`).
- All relationships with multiplicity greater than one are resolved via bridge tables.

All primary keys, foreign keys, and many-to-many structures are clearly indicated using crow's foot notation in the diagram.

3.4 Accessing the Diagram

The diagram is provided as a separate high-resolution JPEG file, and was originally designed using Draw.io. PDF version is also available upon request.

4 Normalized Relations (Logical Design)

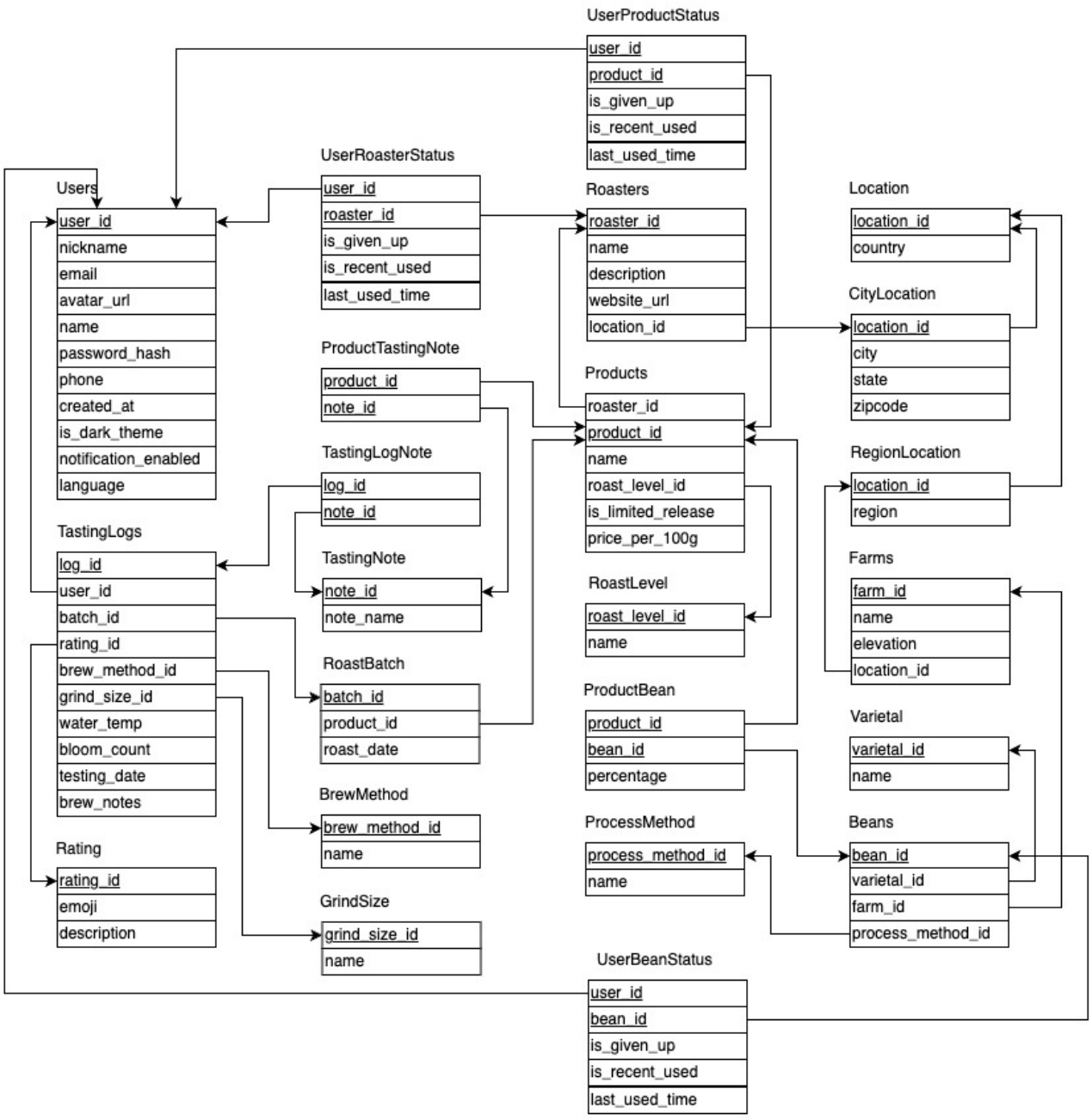
All entities and relations in the Bean Vibes system were carefully normalized into (3NF) to reduce redundancy, ensure data integrity, and simplify query operations.

Each table has clearly defined primary keys (underlined) and foreign key references, and bridge tables are used to resolve many-to-many relationships. Lookup tables are separated to enforce consistency in enumerated values (e.g., 'BrewMethod', 'RoastLevel', 'GrindSize', 'ProcessMethod'). The full list of relations is provided below:

- **Users**(user_id, email, password_hash, name, phone, date_of_birth, nickname, avatar_url, created_at, is_dark_theme, notification_enabled, language)
- **UserRoasterStatus**(user_id, roaster_id, is_given_up, is_recent_used, last_used_time)

- **UserProductStatus**(user_id, product_id, is_given_up, is_recent_used, last_used_time)
- **UserBeanStatus**(user_id, bean_id, is_given_up, is_recent_used, last_used_time)
- **TastingLogs**(log_id, user_id (FK), batch_id (FK), rating_id (FK), brew_method_id (FK), grind_size_id (FK), water_temp, bloom_count, brew_notes, testing_date)
- **TastingLogNote**(log_id, note_id)
- **Rating**(rating_id, emoji, description)
- **BrewMethod**(brew_method_id, name)
- **GrindSize**(grind_size_id, name)
- **TastingNote**(note_id, note_name)
- **Products**(product_id, roaster_id (FK), name, roast_level_id (FK), is_limited_release, price_per_100g)
- **ProductTastingNote**(product_id, note_id)
- **ProductBean**(product_id, bean_id, percentage)
- **Roasters**(roaster_id, name, description, website_url, location_id (FK))
- **RoastBatch**(batch_id, product_id (FK), roast_date)
- **RoastLevel**(roast_level_id, name)
- **Beans**(bean_id, varietal_id (FK), farm_id (FK), process_method_id (FK))
- **Varietal**(varietal_id, name)
- **Farms**(farm_id, name, elevation, location_id (FK))
- **ProcessMethod**(process_method_id, name)
- **Location**(location_id, country)
- **CityLocation**(location_id, city, state, zipcode)
- **RegionLocation**(location_id, region)

Each relation was carefully decomposed to eliminate transitive dependencies and repeated groups, and no denormalization was introduced in this version. The resulting schema supports efficient JOINs for analytics queries and serves as a foundation for both transactional and reporting functions.


UserProductStatus
user_id
product_id
is_given_up
is_recent_used
last_used_time
UserRoasterStatus
user_id
roaster_id
is_given_up
is_recent_used
last_used_time
Users
user_id
nickname
email
avatar_url
name
password_hash
phone
created_at
is_dark_theme
notification_enabled
language
Roasters
roaster_id
name
description
website_url
location_id
Location
location_id
country
CityLocation
location_id
city
state
zipcode
ProductTastingNote
product_id
note_id
Products
roaster_id
product_id
name
roast_level_id
is_limited_release
price_per_100g
TastingLogNote
log_id
note_id
RegionLocation
location_id
region
TastingLogs
log_id
user_id
batch_id
rating_id
brew_method_id
grind_size_id
water_temp
bloom_count
testing_date
brew_notes
TastingNote
note_id
note_name
RoastLevel
roast_level_id
name
Farms
farm_id
name
elevation
location_id
RoastBatch
batch_id
product_id
roast_date
ProductBean
product_id
bean_id
percentage
Varietal
varietal_id
name
BrewMethod
brew_method_id
name
ProcessMethod
process_method_id
name
Beans
bean_id
varietal_id
farm_id
process_method_id
Rating
rating_id
emoji
description
GrindSize
grind_size_id
name
UserBeanStatus
user_id
bean_id
is_given_up
is_recent_used
last_used_time

5 Physical Design

To ensure both performance and data integrity, several physical design strategies were applied. Foreign key columns such as `user_id`, `product_id`, `batch_id`, and `bean_id` are indexed using standard `KEY` declarations to improve the efficiency of joins and filtering, especially for frequent queries over tasting logs and product relationships.

Bridge tables like `ProductBean` and `TastingLogNote` use composite primary keys to enforce unique many-to-many mappings, preventing duplication and maintaining relational accuracy. All fields were assigned SQL types appropriate to their semantics, and lookup tables were used in place of enums or booleans to ensure extensibility and normalization.

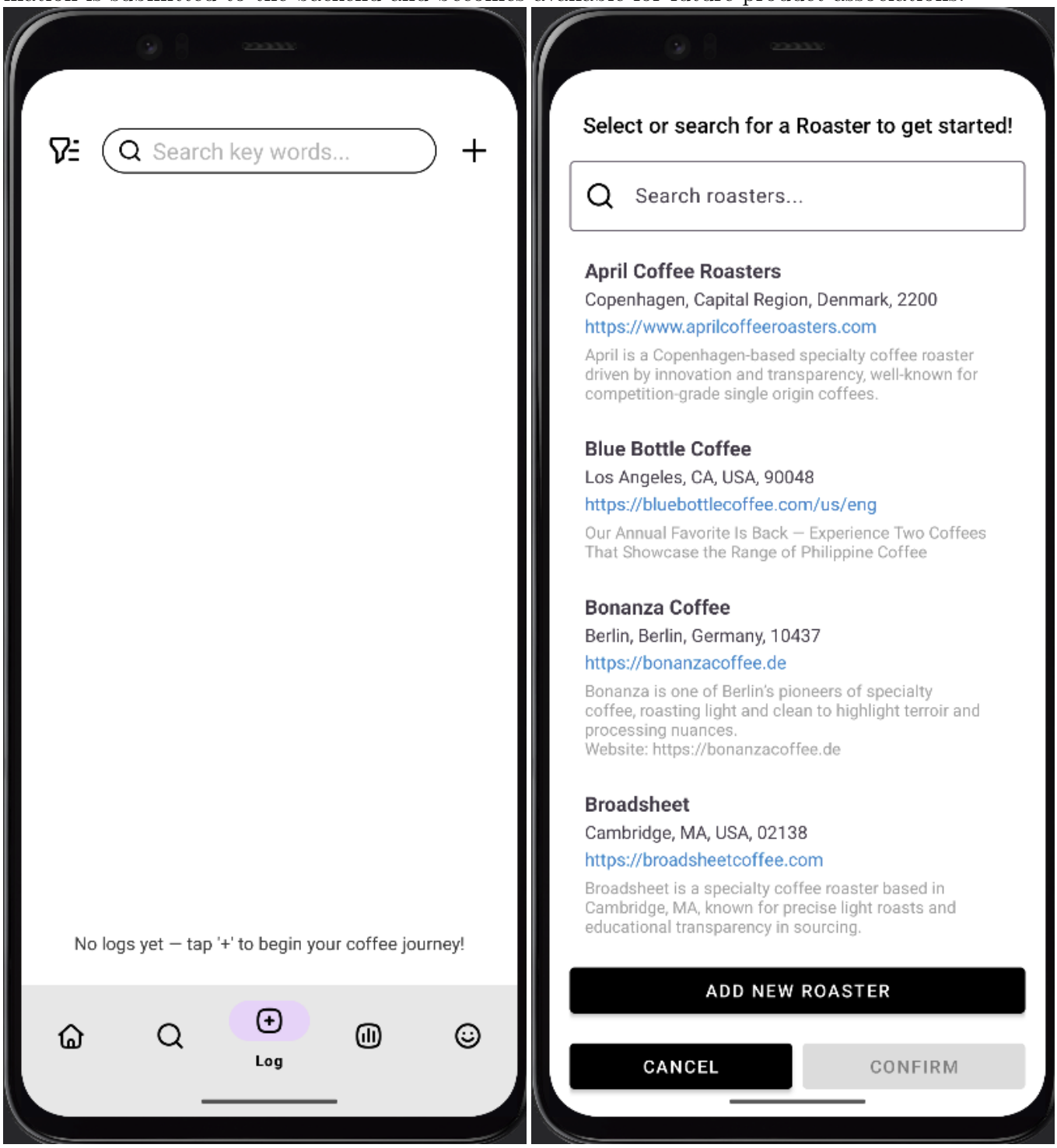
The schema follows 3NF without any denormalization. This helps ensure data consistency while maintaining acceptable performance, supported by indexing and thoughtful table design. The schema is designed to scale gracefully with increasing log entries and user activity, particularly by minimizing update anomalies and supporting analytical queries across normalized tables.

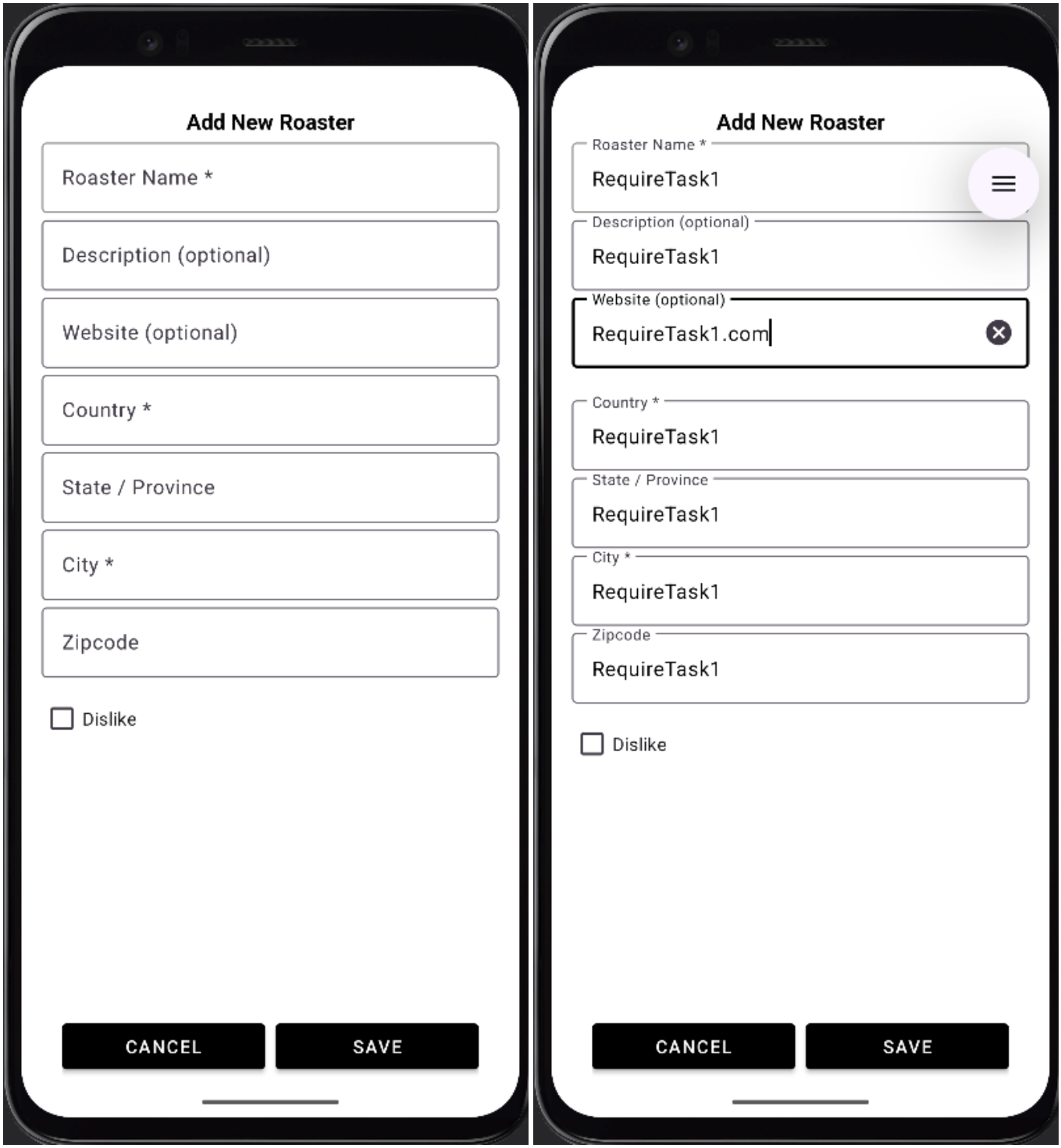
6 Screenshots of Execution

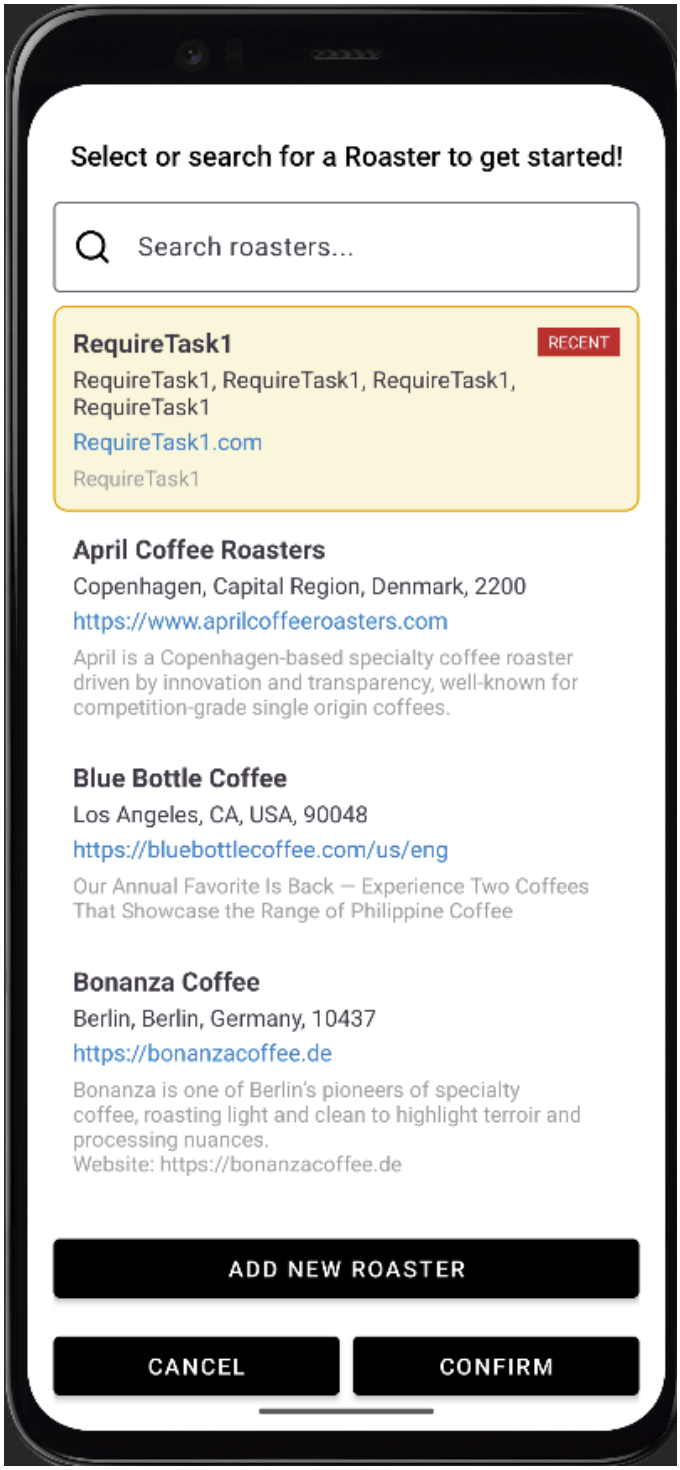
The following sections demonstrate the system fulfilling all required functional tasks and complex reports. Each example includes screenshots of the user interface and backend response where appropriate.
Each question will start on a new page.

6.1 a) Add a Newly Discovered Roaster

A user can add a new roaster by entering name, description, location, and website. The information is submitted to the backend and becomes available for future product associations.

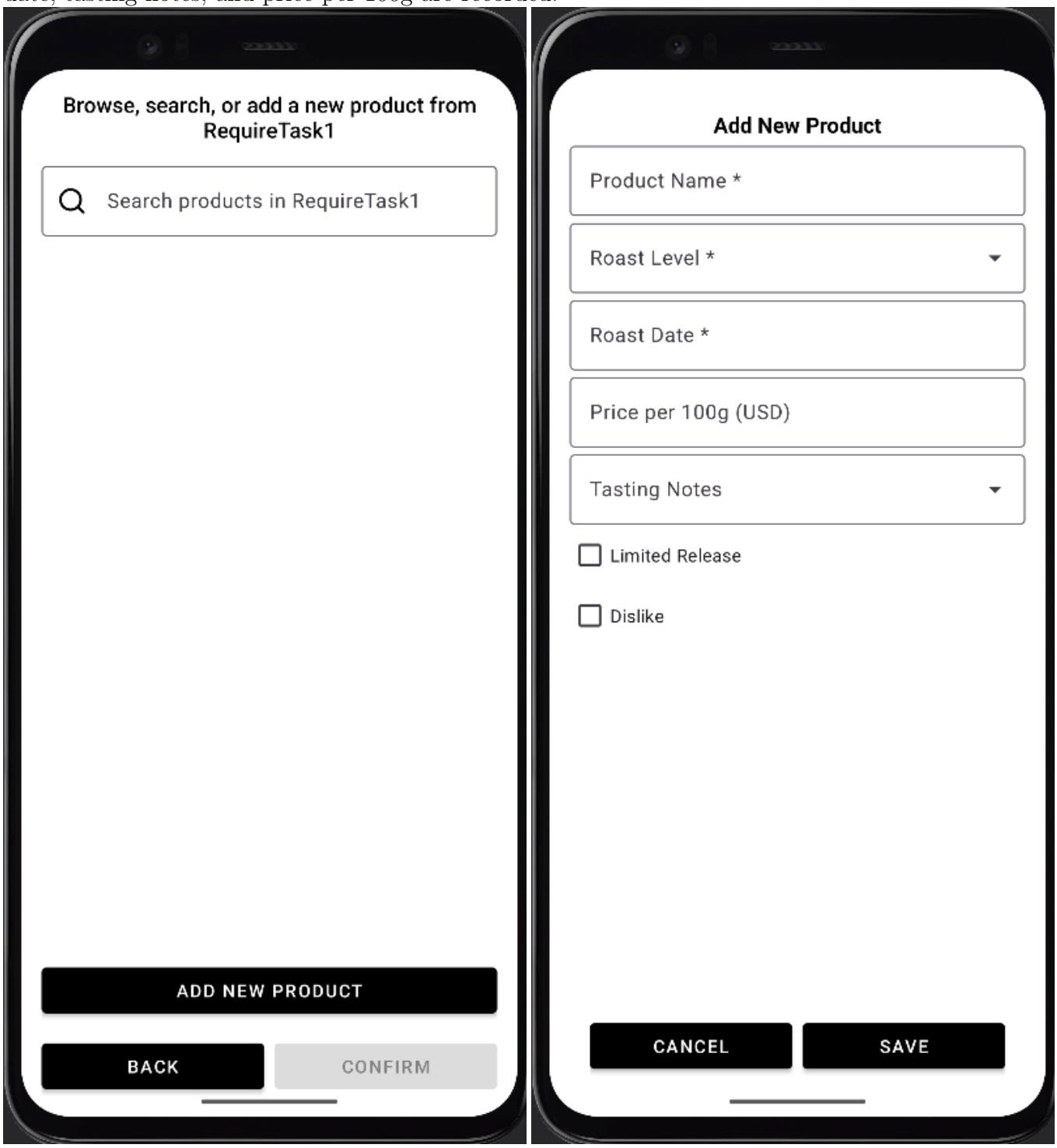


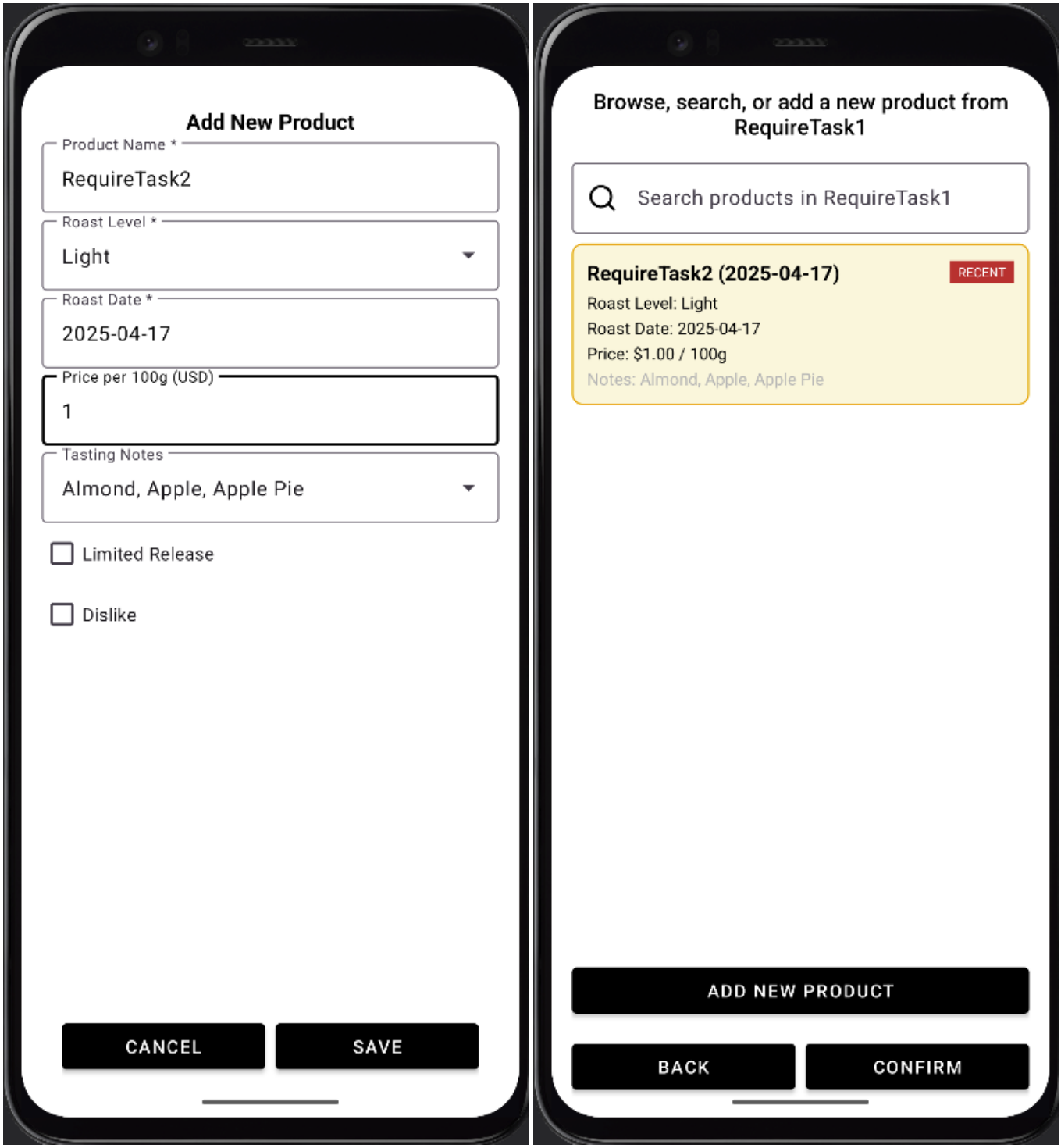

Add New Roaster
Roaster Name *
Description (optional)
Website (optional)
Country *
State / Province
City *
Zipcode
Dislike
CANCEL
SAVE
Add New Roaster
Roaster Name *
RequireTask1
Description (optional)
RequireTask1
Website (optional)
RequireTask1.com
Country *
RequireTask1
State / Province
RequireTask1
City *
RequireTask1
Zipcode
RequireTask1
Dislike
CANCEL
SAVE


Select or search for a Roaster to get started!
Search roasters...
RequireTask1
RECENT
RequireTask1, RequireTask1, RequireTask1, RequireTask1
RequireTask1.com
RequireTask1
April Coffee Roasters
Copenhagen, Capital Region, Denmark, 2200
https://www.aprilcoffeeroasters.com
April is a Copenhagen-based specialty coffee roaster driven by innovation and transparency, well-known for competition-grade single origin coffees.
Blue Bottle Coffee
Los Angeles, CA, USA, 90048
https://bluebottlecoffee.com/us/eng
Our Annual Favorite Is Back — Experience Two Coffees That Showcase the Range of Philippine Coffee
Bonanza Coffee
Berlin, Berlin, Germany, 10437
https://bonanzacoffee.de
Bonanza is one of Berlin's pioneers of specialty coffee, roasting light and clean to highlight terroir and processing nuances.
Website: https://bonanzacoffee.de
ADD NEW ROASTER
CANCEL
CONFIRM

6.2 b) Add a Newly Purchased Roast

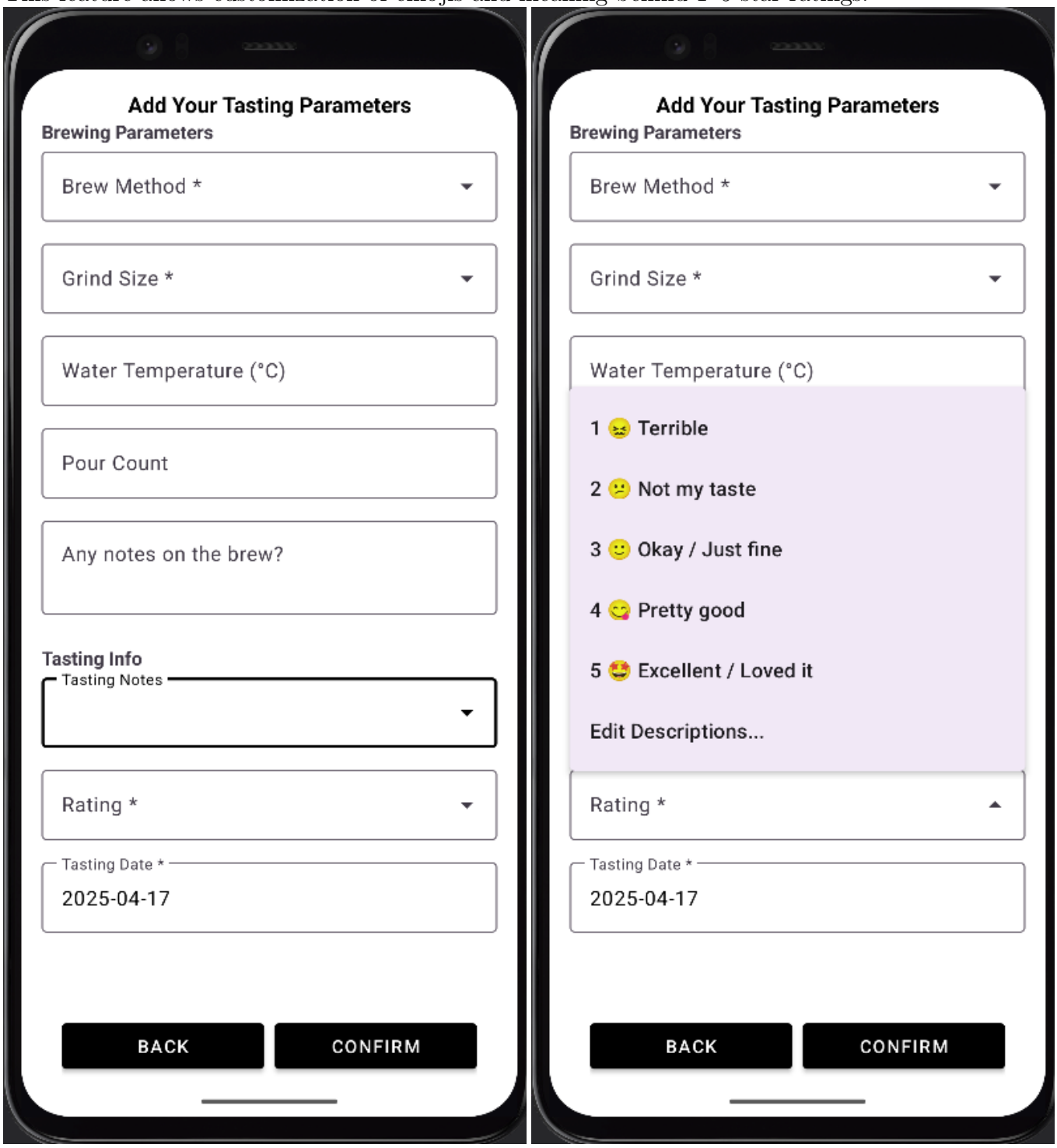
Newly purchased roasts are added under an existing or newly added roaster. Roast level, roast date, tasting notes, and price per 100g are recorded.

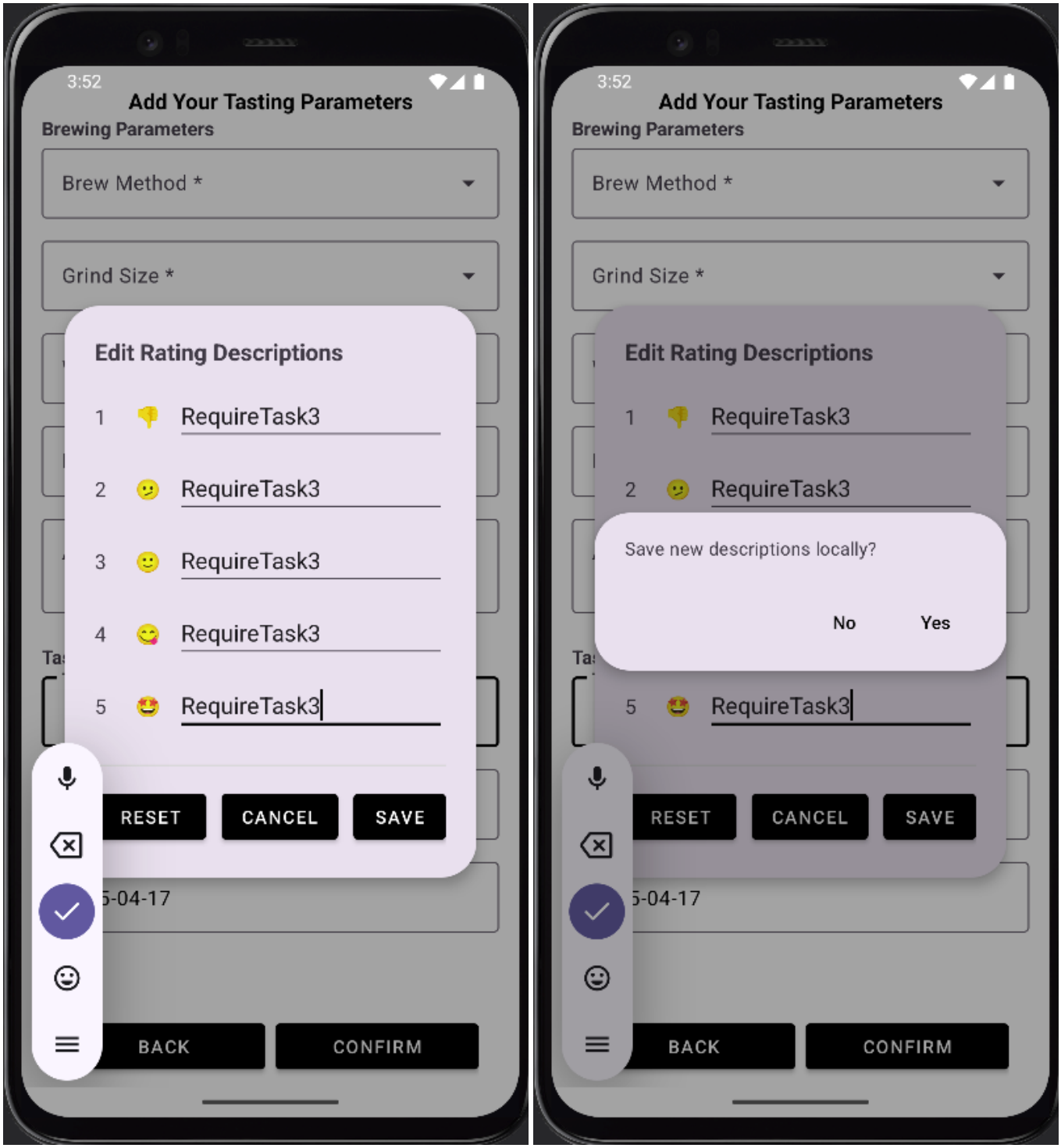


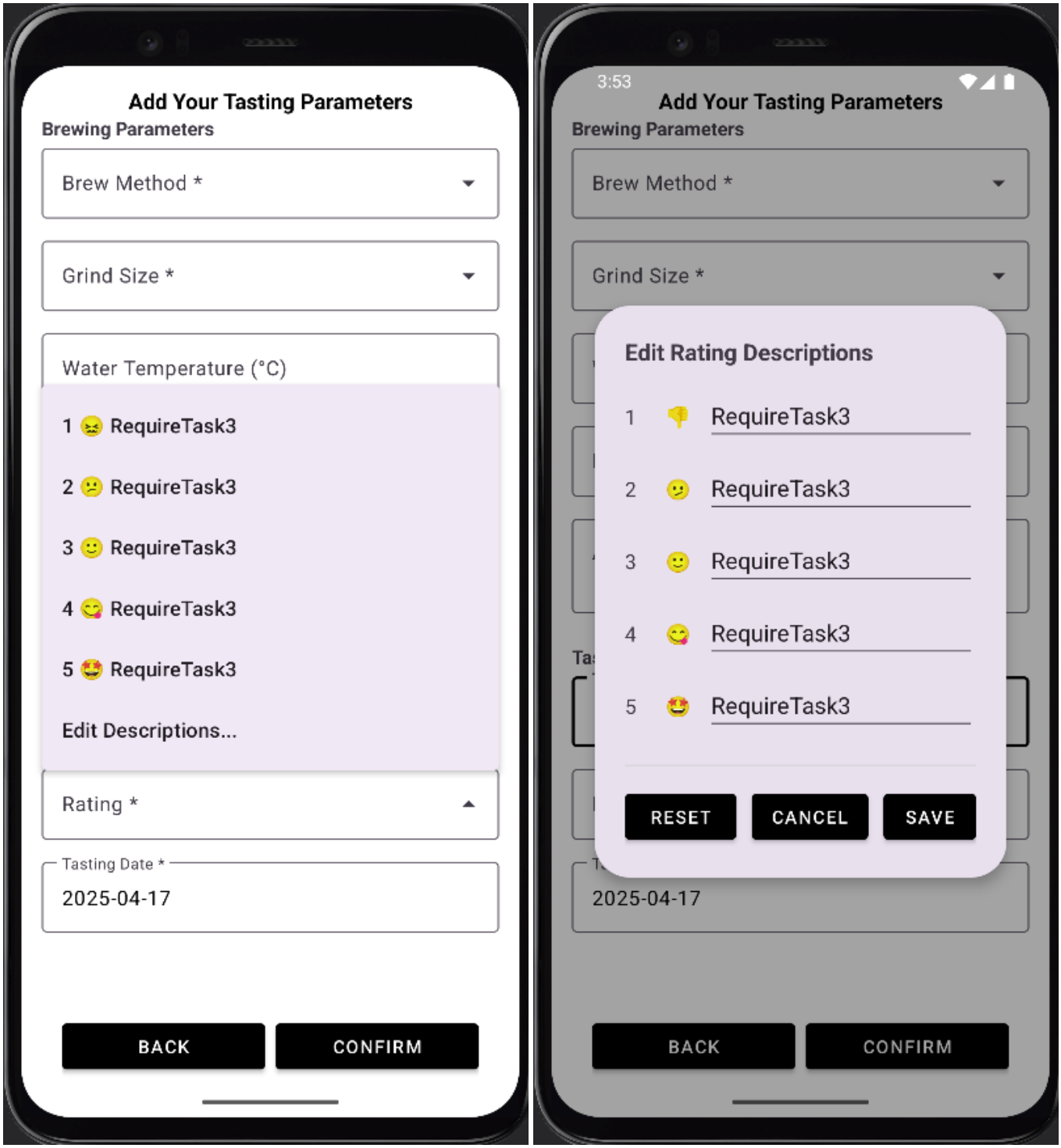

Add New Product
Product Name *
RequireTask2
Roast Level *
Light
Roast Date *
2025-04-17
Price per 100g (USD)
1
Tasting Notes
Almond, Apple, Apple Pie
Limited Release
Dislike
CANCEL
SAVE
Browse, search, or add a new product from RequireTask1
Search products in RequireTask1
RequireTask2 (2025-04-17)
RECENT
Roast Level: Light
Roast Date: 2025-04-17
Price: $1.00 / 100g
Notes: Almond, Apple, Apple Pie
ADD NEW PRODUCT
BACK
CONFIRM

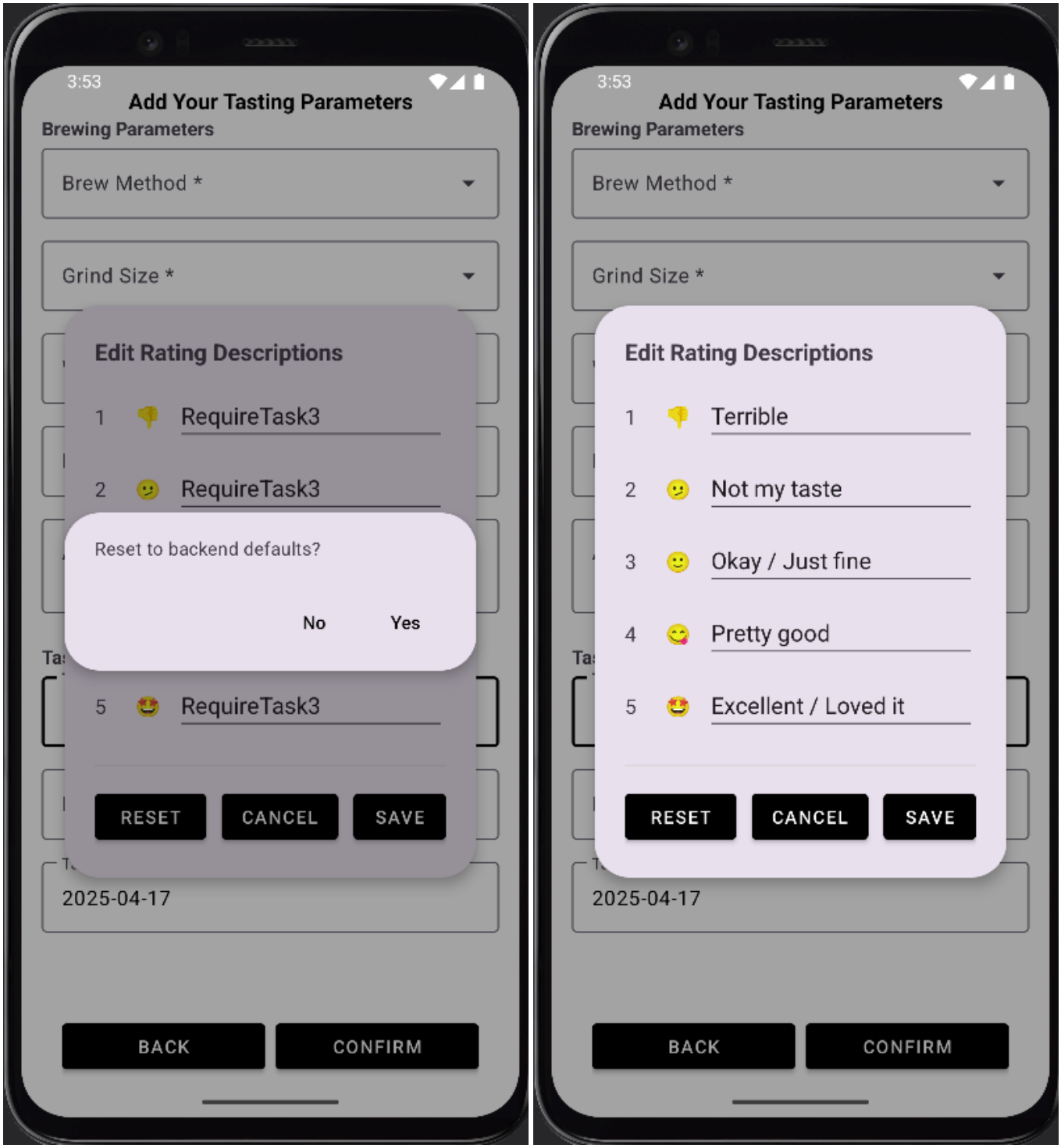
6.3 c) Update the Rating Description

Users can personalize their rating descriptions through the Edit Rating Descriptions dialog. This feature allows customization of emojis and meaning behind 1–5 star ratings.



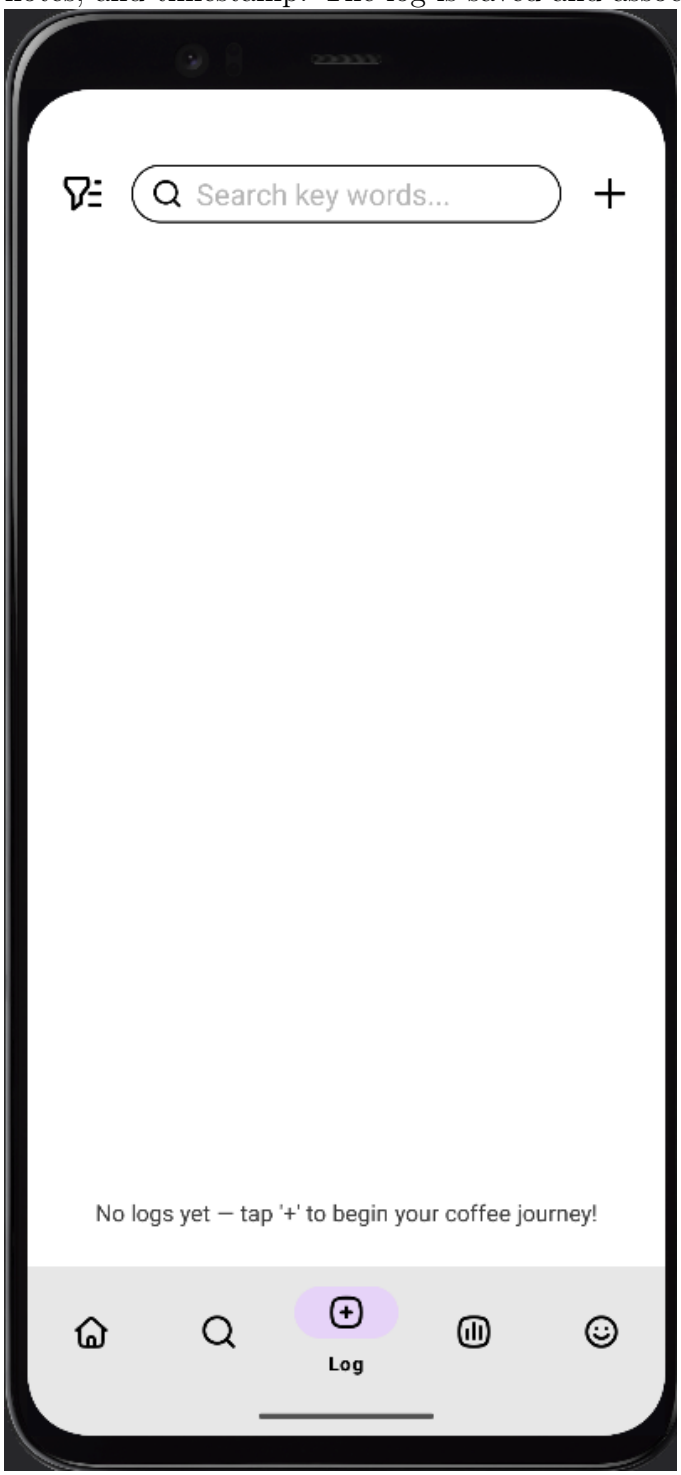

3:52
Add Your Tasting Parameters
Brewing Parameters
Brew Method *
Grind Size *
Edit Rating Descriptions
1 RequireTask3
2 RequireTask3
3 RequireTask3
4 RequireTask3
5 RequireTask3
RESET
CANCEL
SAVE
BACK
CONFIRM
Save new descriptions locally?
No
Yes

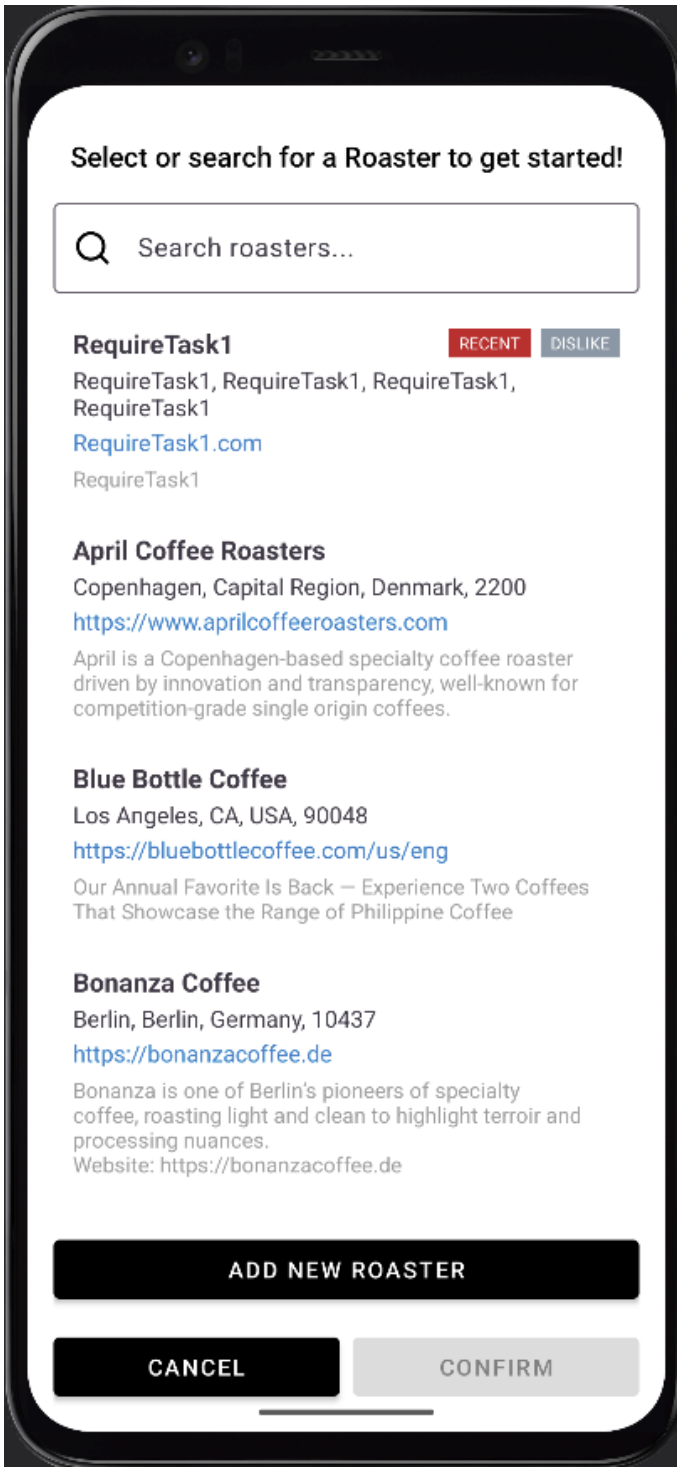

Add Your Tasting Parameters
Brewing Parameters
Brew Method *
Grind Size *
Water Temperature (°C)
1 😖 RequireTask3
2 😕 RequireTask3
3 🙂 RequireTask3
4 😋 RequireTask3
5 🤩 RequireTask3
Edit Descriptions...
Rating *
Tasting Date *
2025-04-17
BACK
CONFIRM
3:53
Add Your Tasting Parameters
Brewing Parameters
Brew Method *
Grind Size *
Edit Rating Descriptions
1 👎 RequireTask3
2 😕 RequireTask3
3 🙂 RequireTask3
4 😋 RequireTask3
5 🤩 RequireTask3
RESET
CANCEL
SAVE
2025-04-17
BACK
CONFIRM

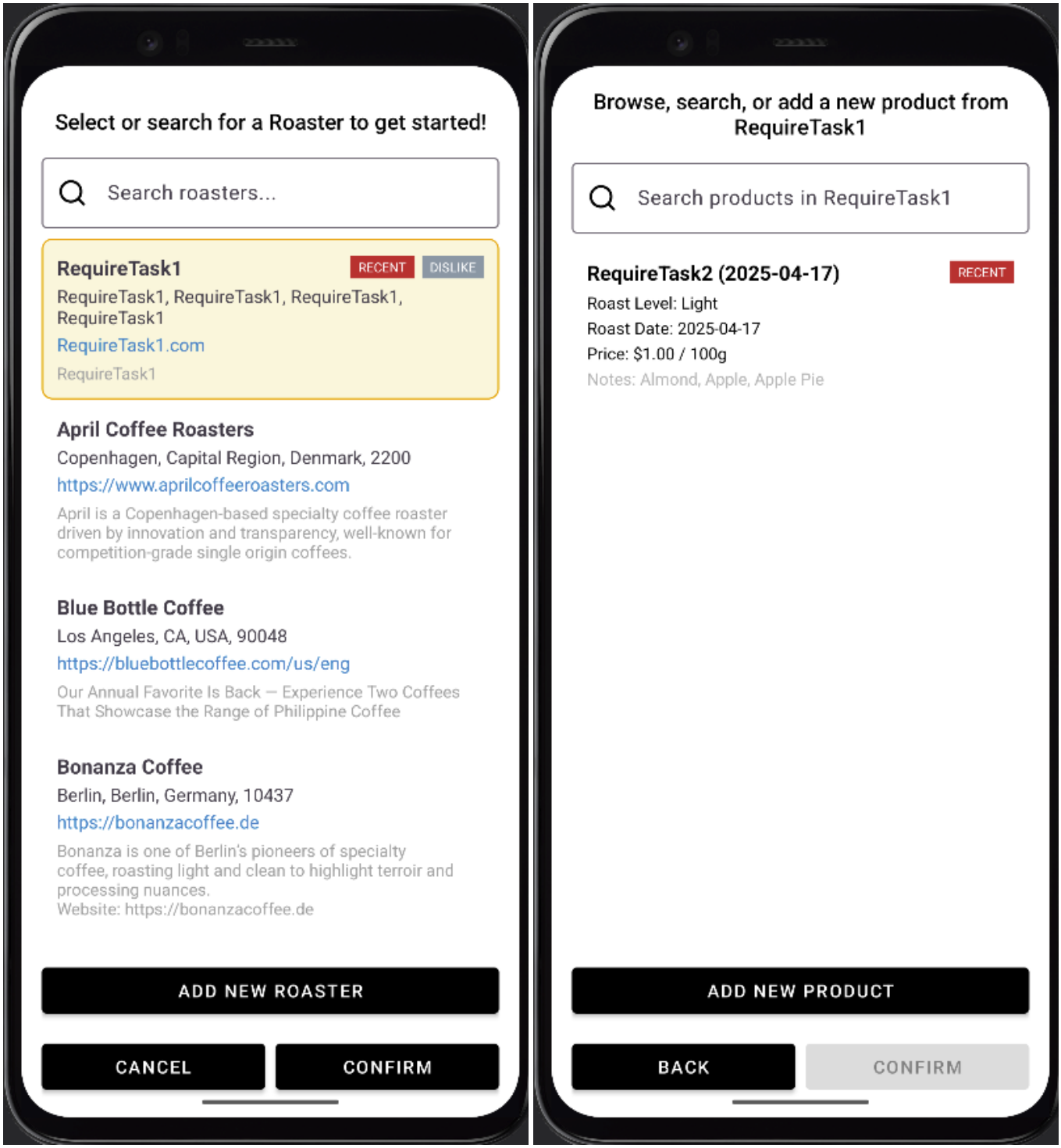

3:53
Add Your Tasting Parameters
Brewing Parameters
Brew Method *
Grind Size *
Edit Rating Descriptions
1 RequireTask3
2 RequireTask3
Reset to backend defaults?
No
Yes
5 RequireTask3
RESET
CANCEL
SAVE
2025-04-17
BACK
CONFIRM
3:53
Add Your Tasting Parameters
Brewing Parameters
Brew Method *
Grind Size *
Edit Rating Descriptions
1 Terrible
2 Not my taste
3 Okay / Just fine
4 Pretty good
5 Excellent / Loved it
RESET
CANCEL
SAVE
2025-04-17
BACK
CONFIRM

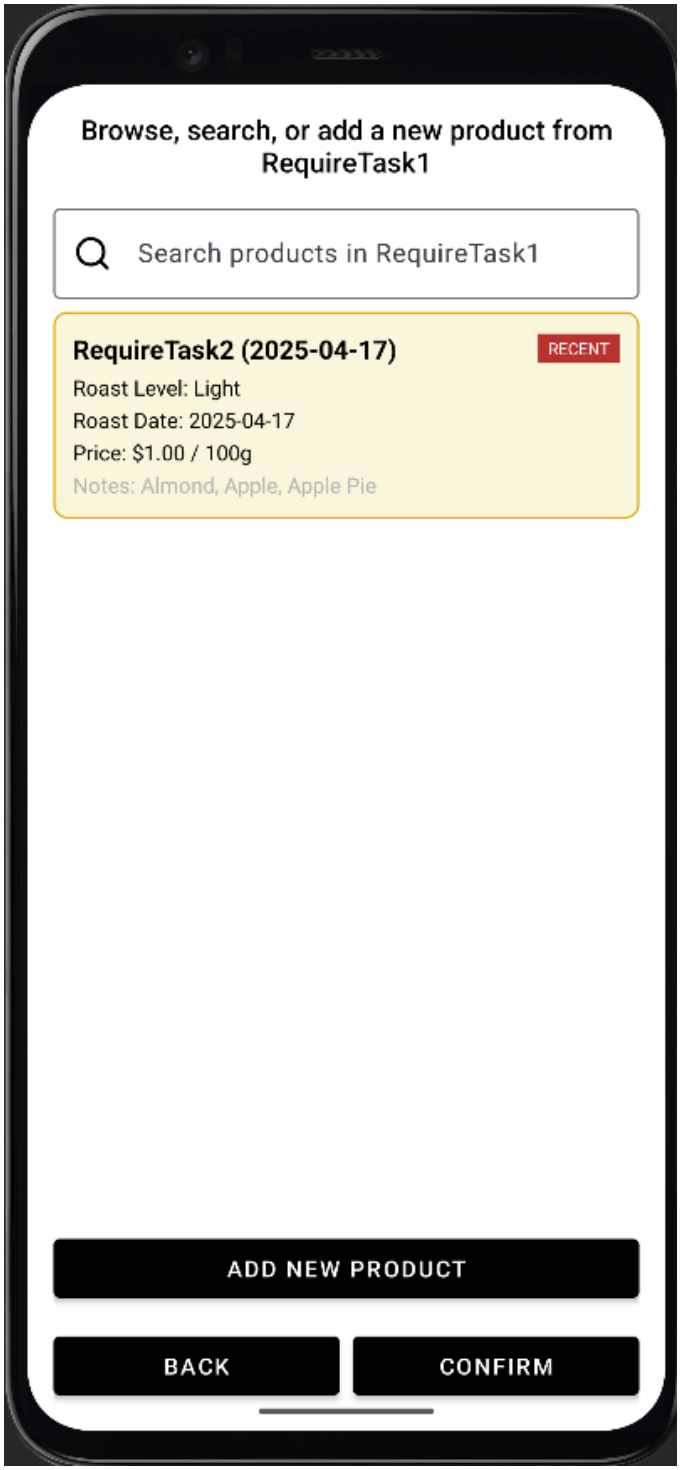
6.4 d) Log a Tasting

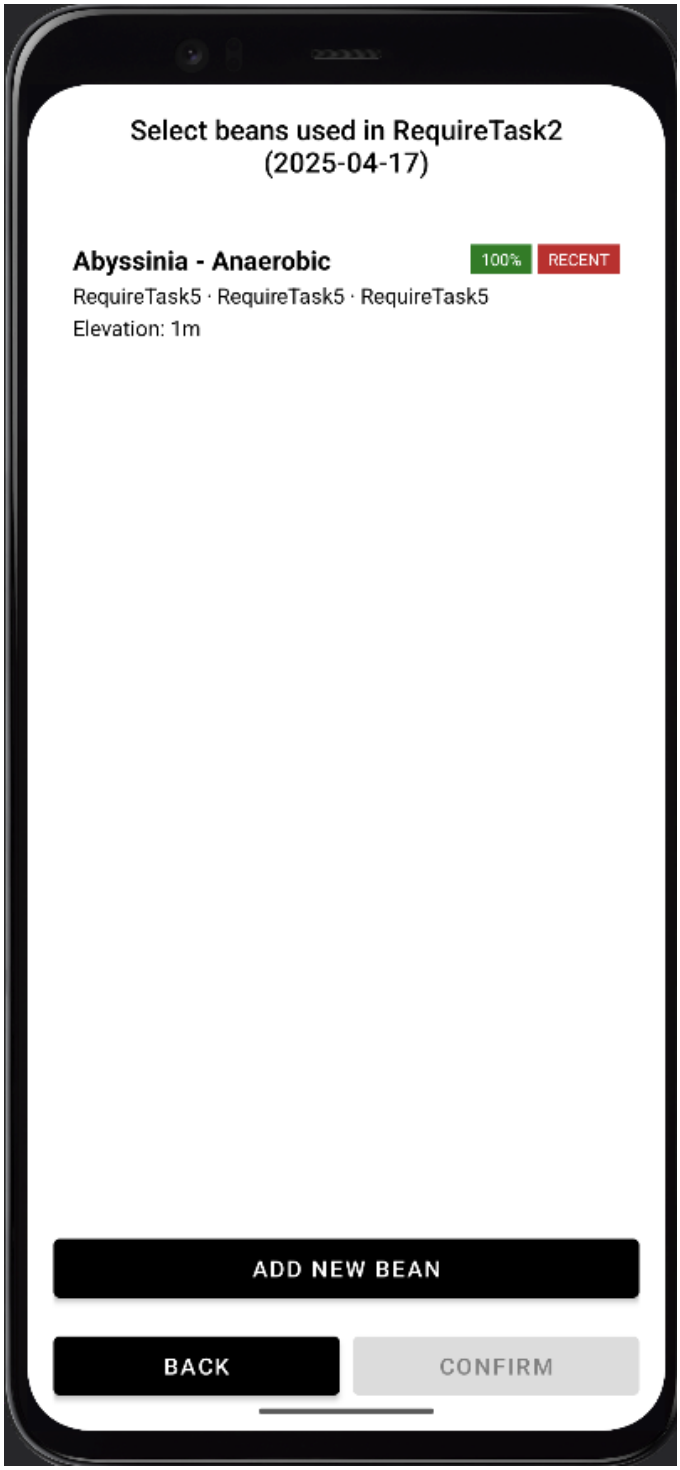
The system allows full tasting log creation, including bean selection, brewing parameters, rating, notes, and timestamp. The log is saved and associated with the selected roast batch.

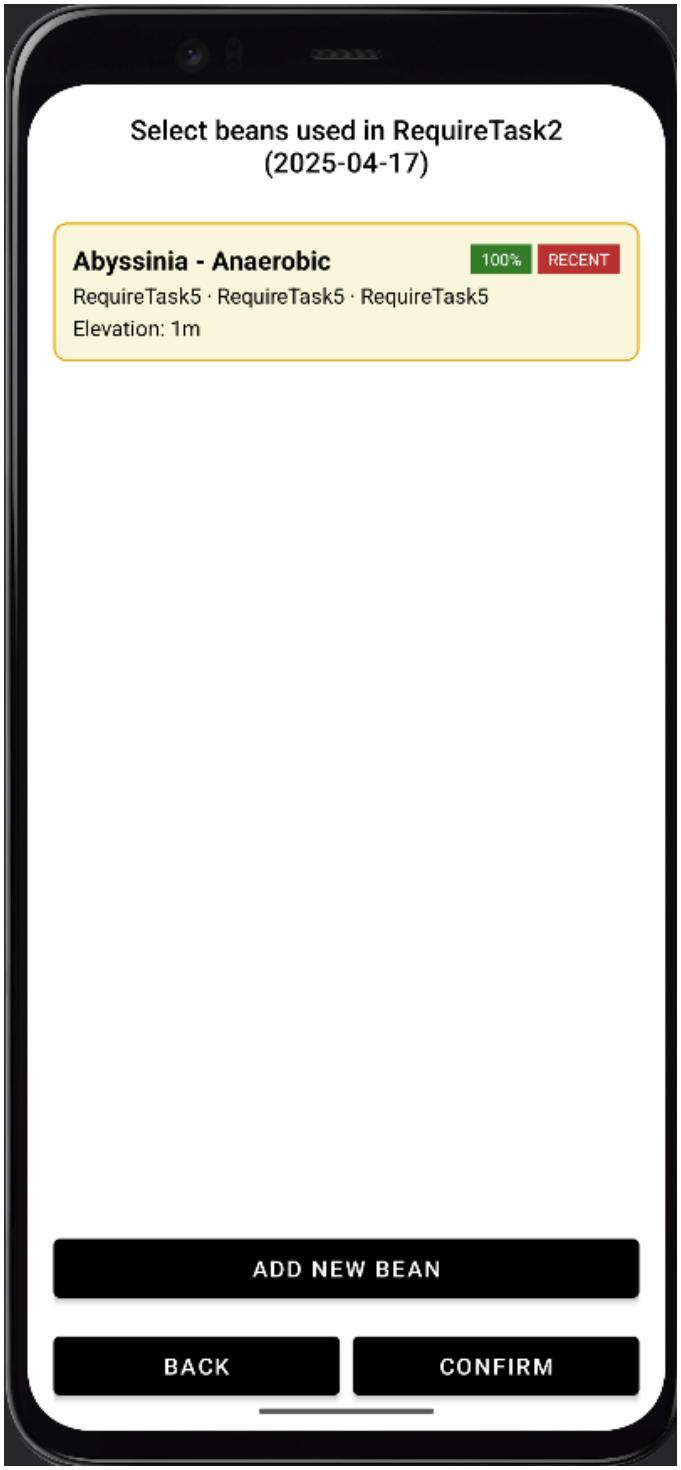


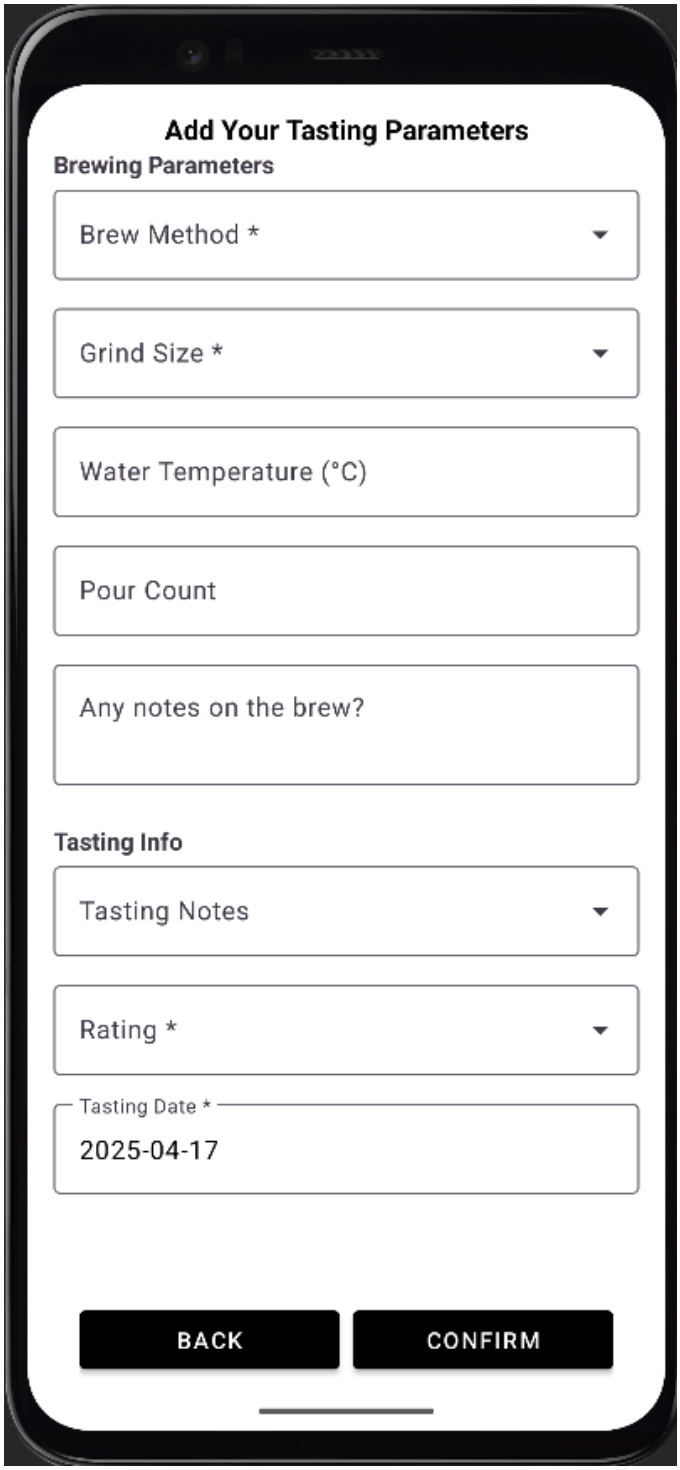


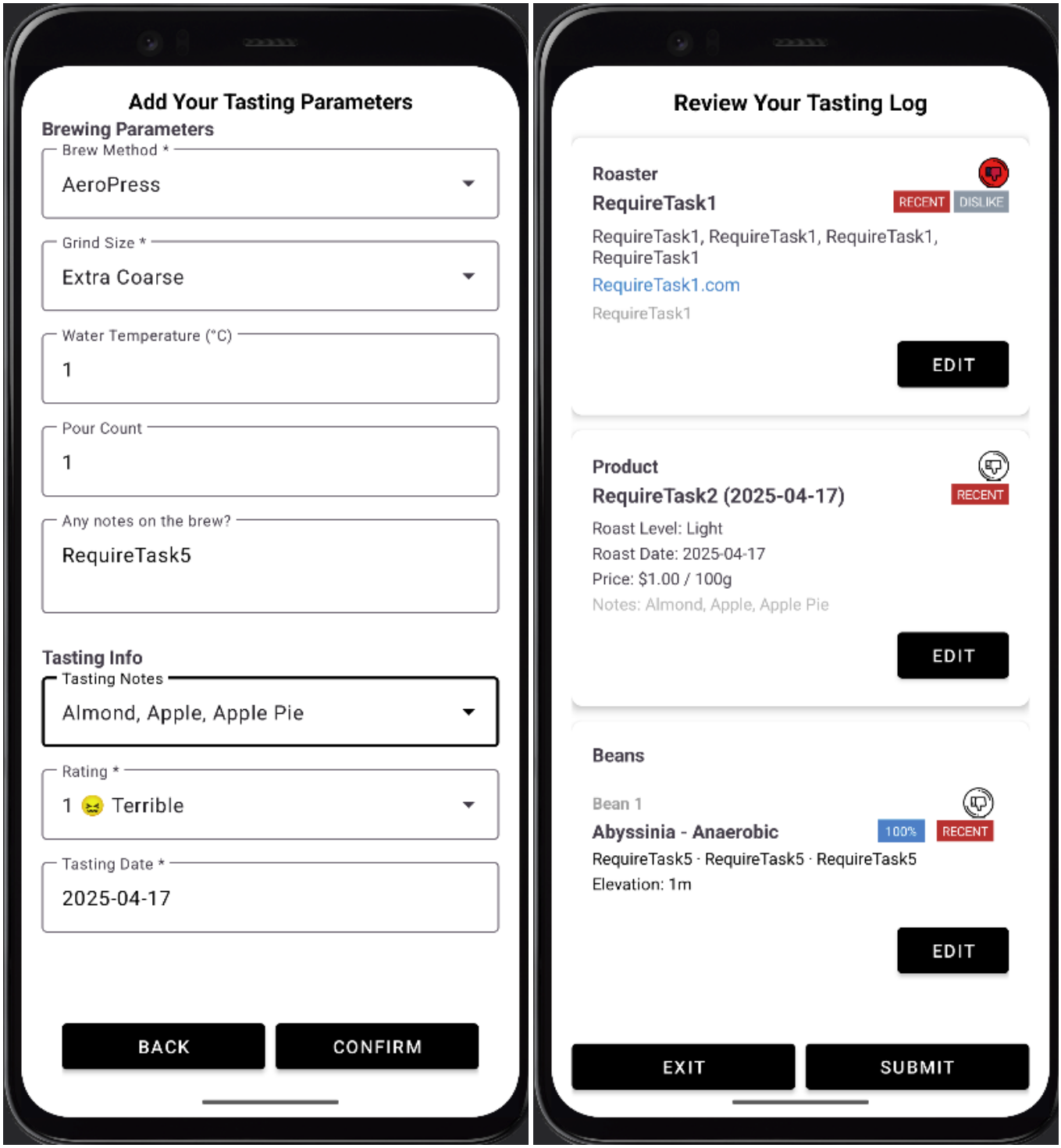

Select or search for a Roaster to get started!
Search roasters...
RequireTask1
RECENT
DISLIKE
RequireTask1, RequireTask1, RequireTask1, RequireTask1
RequireTask1.com
RequireTask1
April Coffee Roasters
Copenhagen, Capital Region, Denmark, 2200
https://www.aprilcoffeeroasters.com
April is a Copenhagen-based specialty coffee roaster driven by innovation and transparency, well-known for competition-grade single origin coffees.
Blue Bottle Coffee
Los Angeles, CA, USA, 90048
https://bluebottlecoffee.com/us/eng
Our Annual Favorite Is Back — Experience Two Coffees That Showcase the Range of Philippine Coffee
Bonanza Coffee
Berlin, Berlin, Germany, 10437
https://bonanzacoffee.de
Bonanza is one of Berlin's pioneers of specialty coffee, roasting light and clean to highlight terroir and processing nuances.
Website: https://bonanzacoffee.de
ADD NEW ROASTER
CANCEL
CONFIRM
Browse, search, or add a new product from RequireTask1
Search products in RequireTask1
RequireTask2 (2025-04-17)
RECENT
Roast Level: Light
Roast Date: 2025-04-17
Price: $1.00 / 100g
Notes: Almond, Apple, Apple Pie
ADD NEW PRODUCT
BACK
CONFIRM

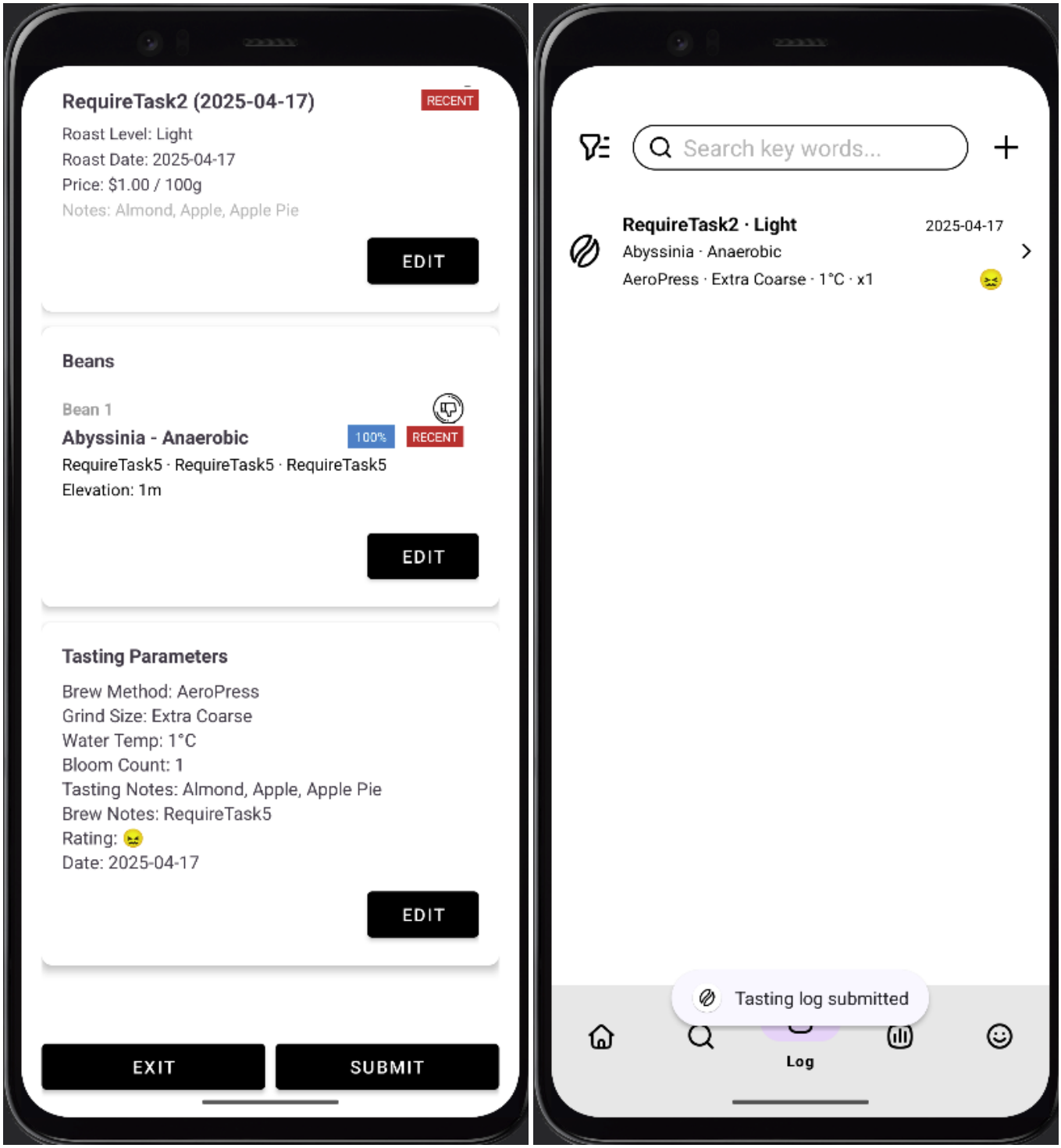

Browse, search, or add a new product from RequireTask1
Search products in RequireTask1
RequireTask2 (2025-04-17)
RECENT
Roast Level: Light
Roast Date: 2025-04-17
Price: $1.00 / 100g
Notes: Almond, Apple, Apple Pie
ADD NEW PRODUCT
BACK
CONFIRM


Select beans used in RequireTask2 (2025-04-17)
Abyssinia - Anaerobic
100%
RECENT
RequireTask5 · RequireTask5 · RequireTask5
Elevation: 1m
ADD NEW BEAN
BACK
CONFIRM


Select beans used in RequireTask2 (2025-04-17)
Abyssinia - Anaerobic
100%
RECENT
RequireTask5 · RequireTask5 · RequireTask5
Elevation: 1m
ADD NEW BEAN
BACK
CONFIRM

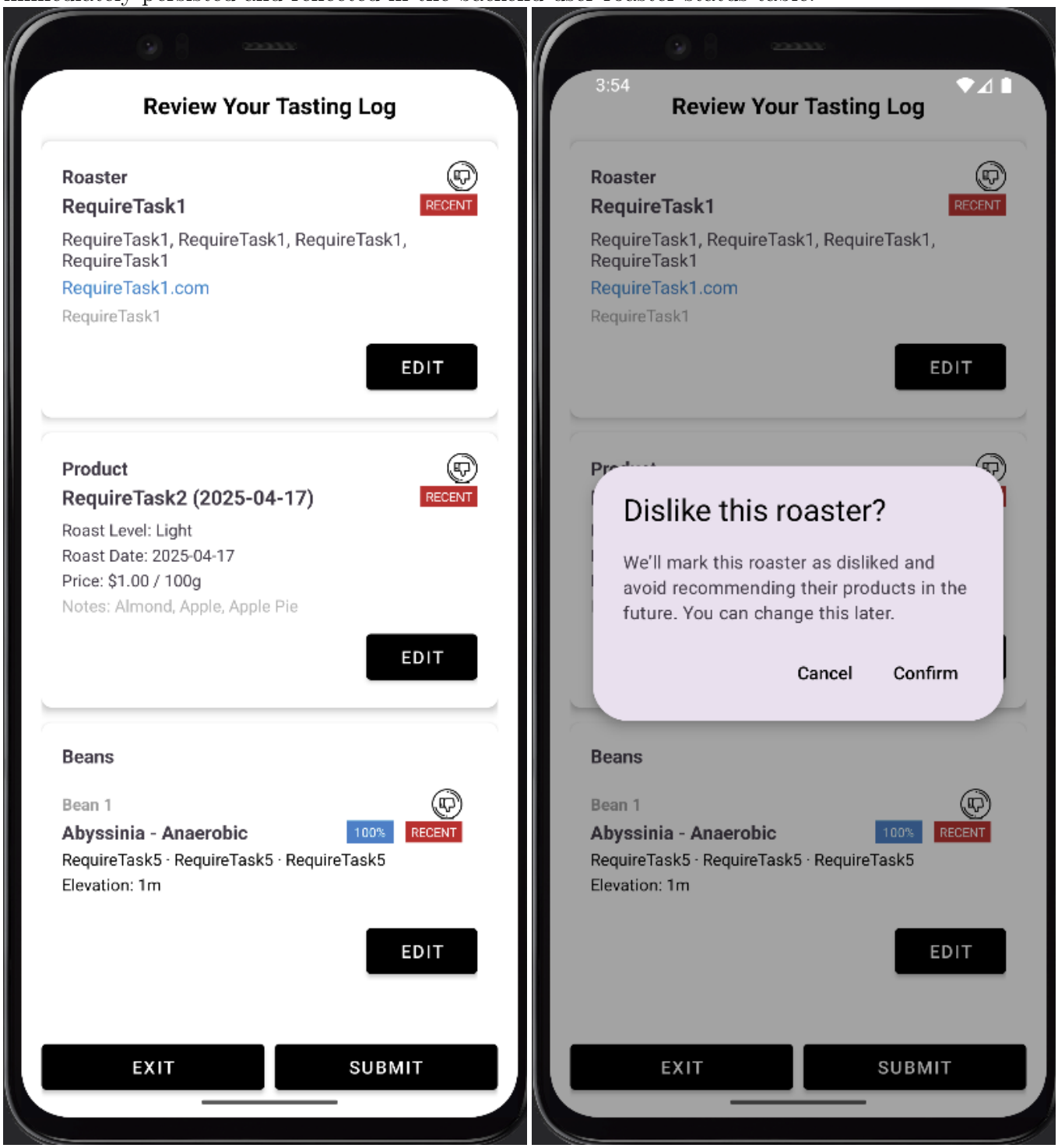

Add Your Tasting Parameters
Brewing Parameters
Brew Method *
Grind Size *
Water Temperature (°C)
Pour Count
Any notes on the brew?
Tasting Info
Tasting Notes
Rating *
Tasting Date *
2025-04-17
BACK
CONFIRM

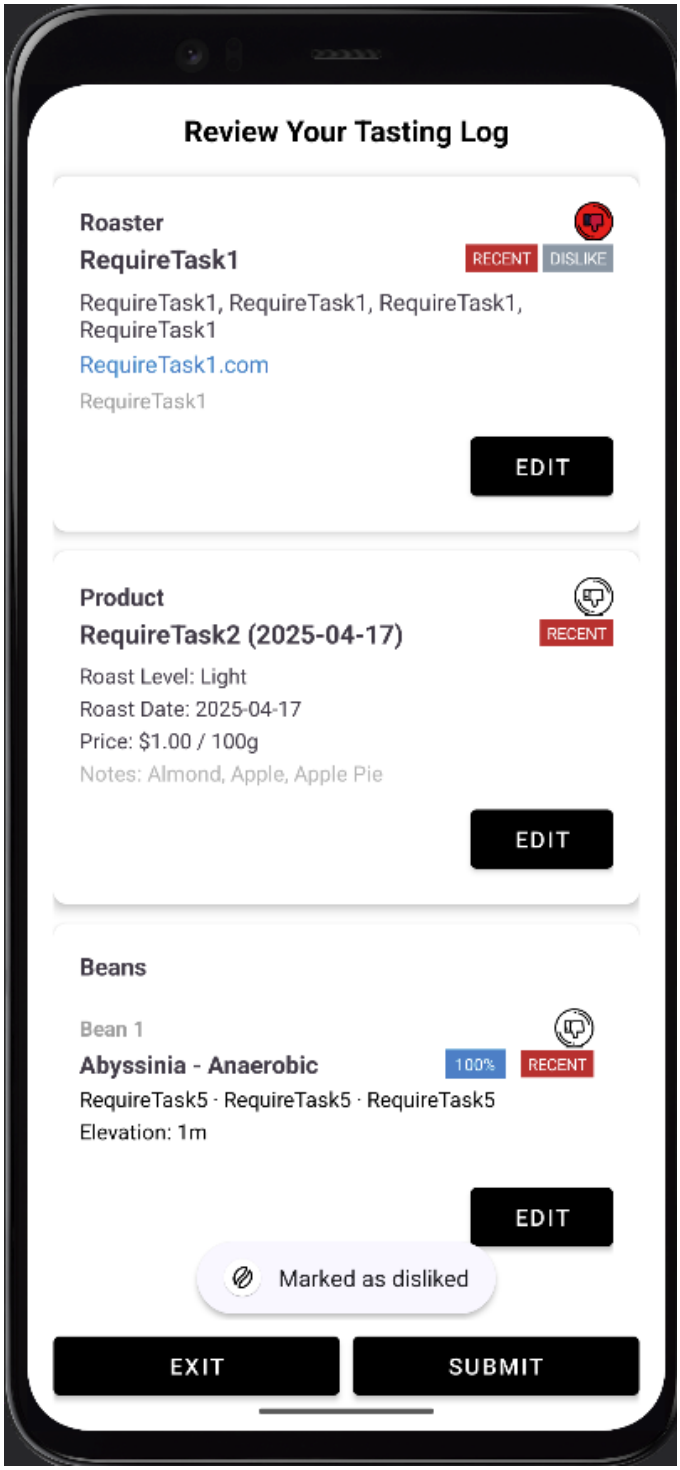

Add Your Tasting Parameters
Brewing Parameters
Brew Method *
AeroPress
Grind Size *
Extra Coarse
Water Temperature (°C)
1
Pour Count
1
Any notes on the brew?
RequireTask5
Tasting Info
Tasting Notes
Almond, Apple, Apple Pie
Rating *
1 Terrible
Tasting Date *
2025-04-17
BACK
CONFIRM
Review Your Tasting Log
Roaster
RequireTask1
RECENT
DISLIKE
RequireTask1, RequireTask1, RequireTask1, RequireTask1
RequireTask1.com
RequireTask1
EDIT
Product
RequireTask2 (2025-04-17)
RECENT
Roast Level: Light
Roast Date: 2025-04-17
Price: $1.00 / 100g
Notes: Almond, Apple, Apple Pie
EDIT
Beans
Bean 1
Abyssinia - Anaerobic
100%
RECENT
RequireTask5 · RequireTask5 · RequireTask5
Elevation: 1m
EDIT
EXIT
SUBMIT


RequireTask2 (2025-04-17)
RECENT
Roast Level: Light
Roast Date: 2025-04-17
Price: $1.00 / 100g
Notes: Almond, Apple, Apple Pie
EDIT
Beans
Bean 1
Abyssinia - Anaerobic
100%
RECENT
RequireTask5 · RequireTask5 · RequireTask5
Elevation: 1m
EDIT
Tasting Parameters
Brew Method: AeroPress
Grind Size: Extra Coarse
Water Temp: 1°C
Bloom Count: 1
Tasting Notes: Almond, Apple, Apple Pie
Brew Notes: RequireTask5
Rating:
Date: 2025-04-17
EDIT
EXIT
SUBMIT
Search key words...
RequireTask2 · Light
2025-04-17
Abyssinia · Anaerobic
AeroPress · Extra Coarse · 1°C · x1
Tasting log submitted
Log

6.5 e) Give Up on a Roaster

Users can "give up" on a roaster by toggling a dislike icon on the review page. This status is immediately persisted and reflected in the backend user-roaster status table.

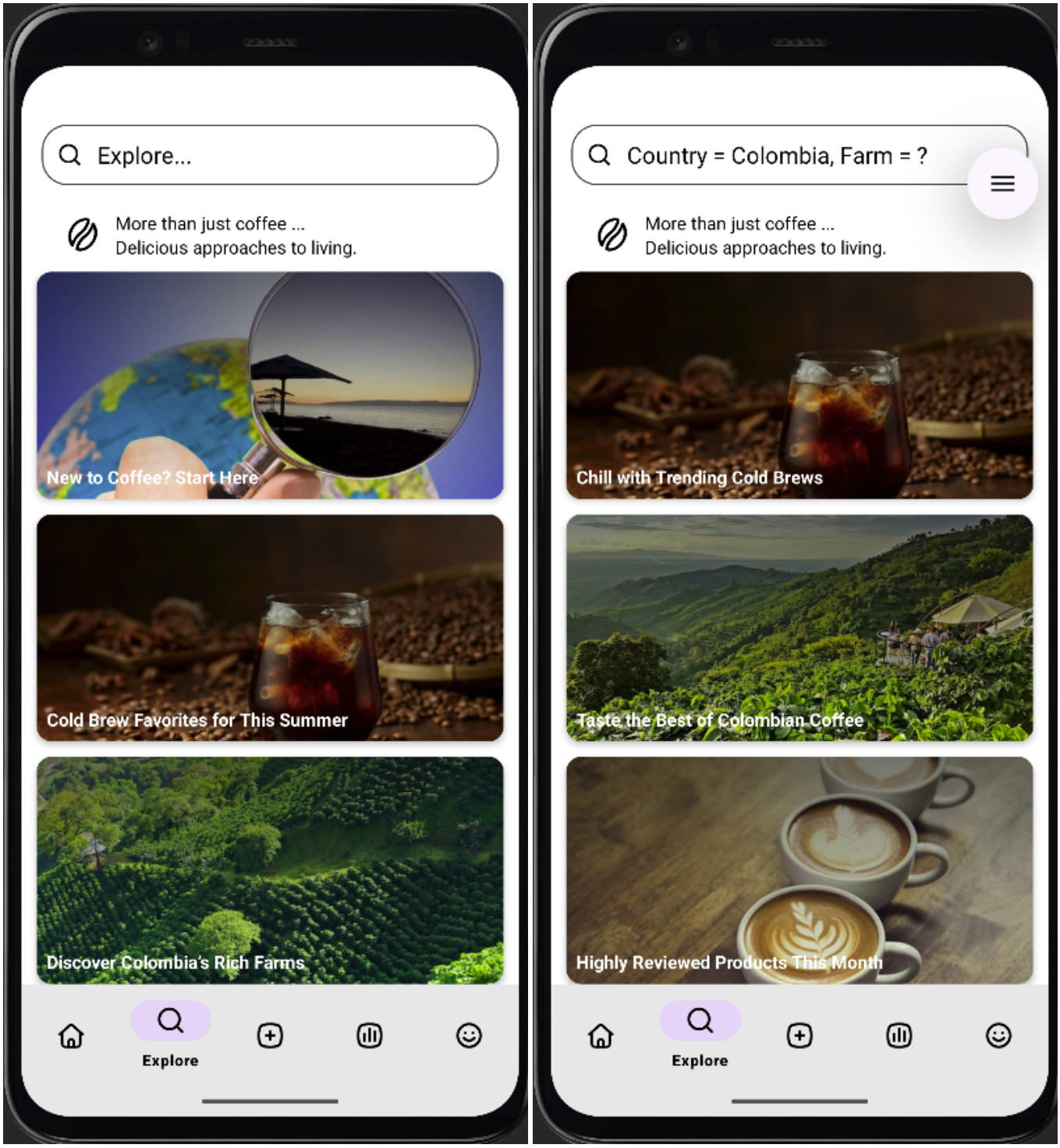


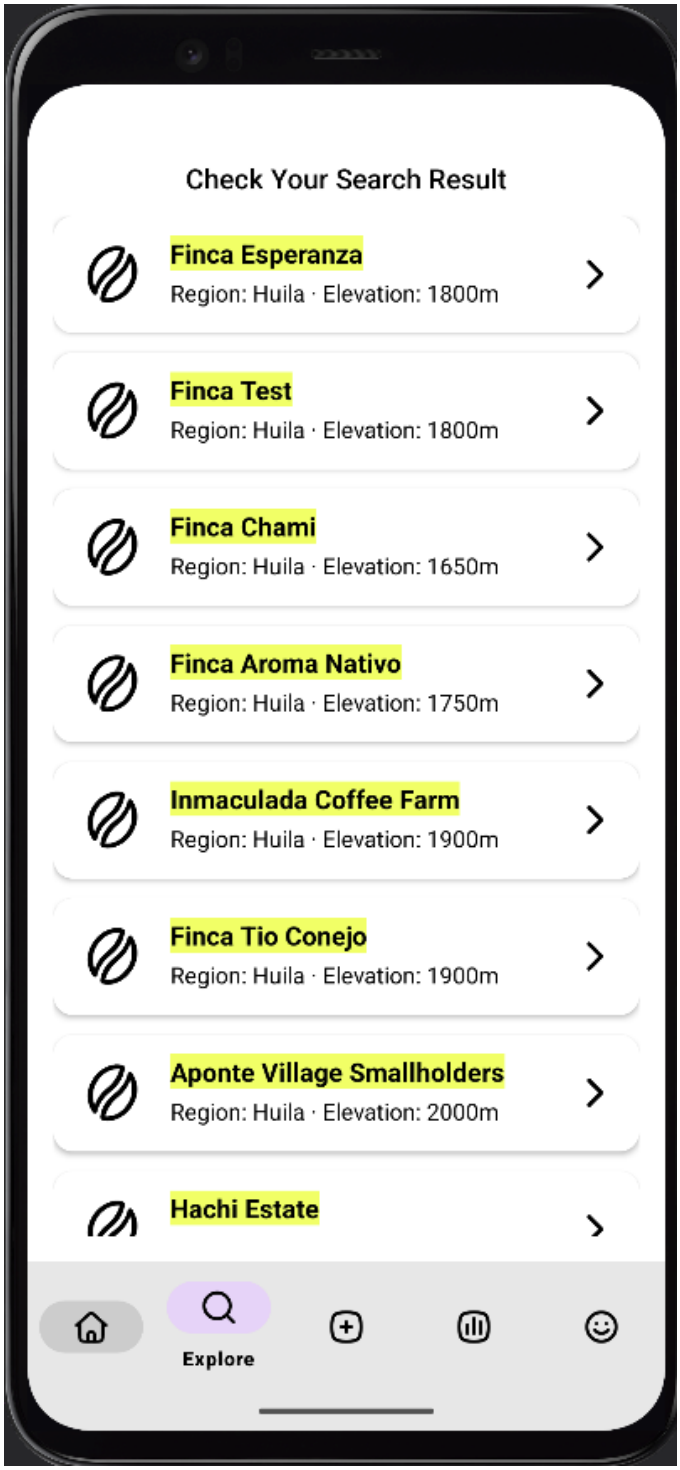

Review Your Tasting Log
Roaster
RequireTask1
RECENT
DISLIKE
RequireTask1, RequireTask1, RequireTask1, RequireTask1
RequireTask1.com
RequireTask1
EDIT
Product
RequireTask2 (2025-04-17)
RECENT
Roast Level: Light
Roast Date: 2025-04-17
Price: $1.00 / 100g
Notes: Almond, Apple, Apple Pie
EDIT
Beans
Bean 1
Abyssinia - Anaerobic
100%
RECENT
RequireTask5 · RequireTask5 · RequireTask5
Elevation: 1m
EDIT
Marked as disliked
EXIT
SUBMIT

6.6 f) Find all grower's in a particular country

Using the farm search feature, users can filter farms by country (e.g., Colombia). Region, elevation, and farm name are displayed.

- Country = $COUNTRY NAME, Farm = ?
- Country = colombia, Farm = ?

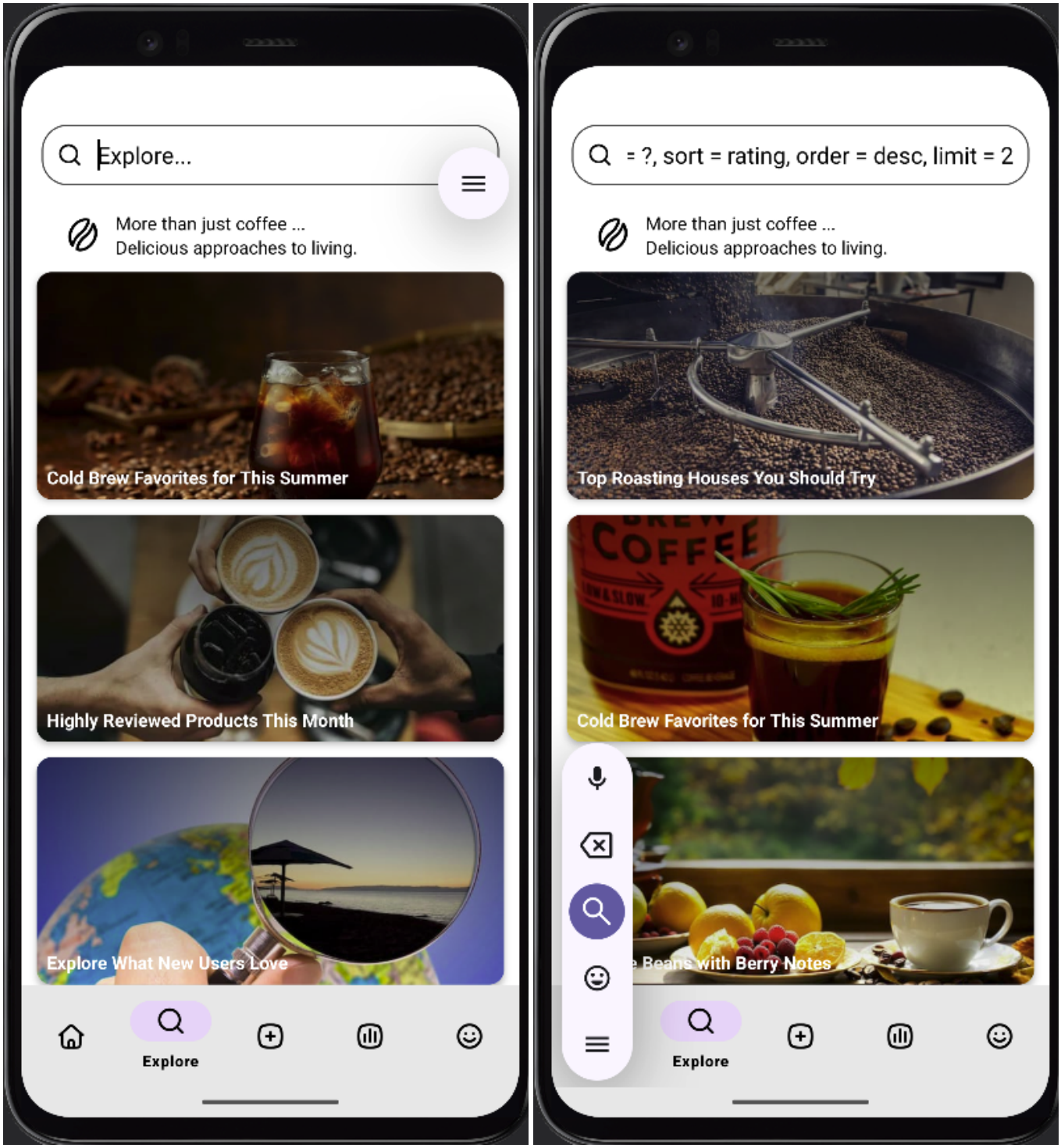


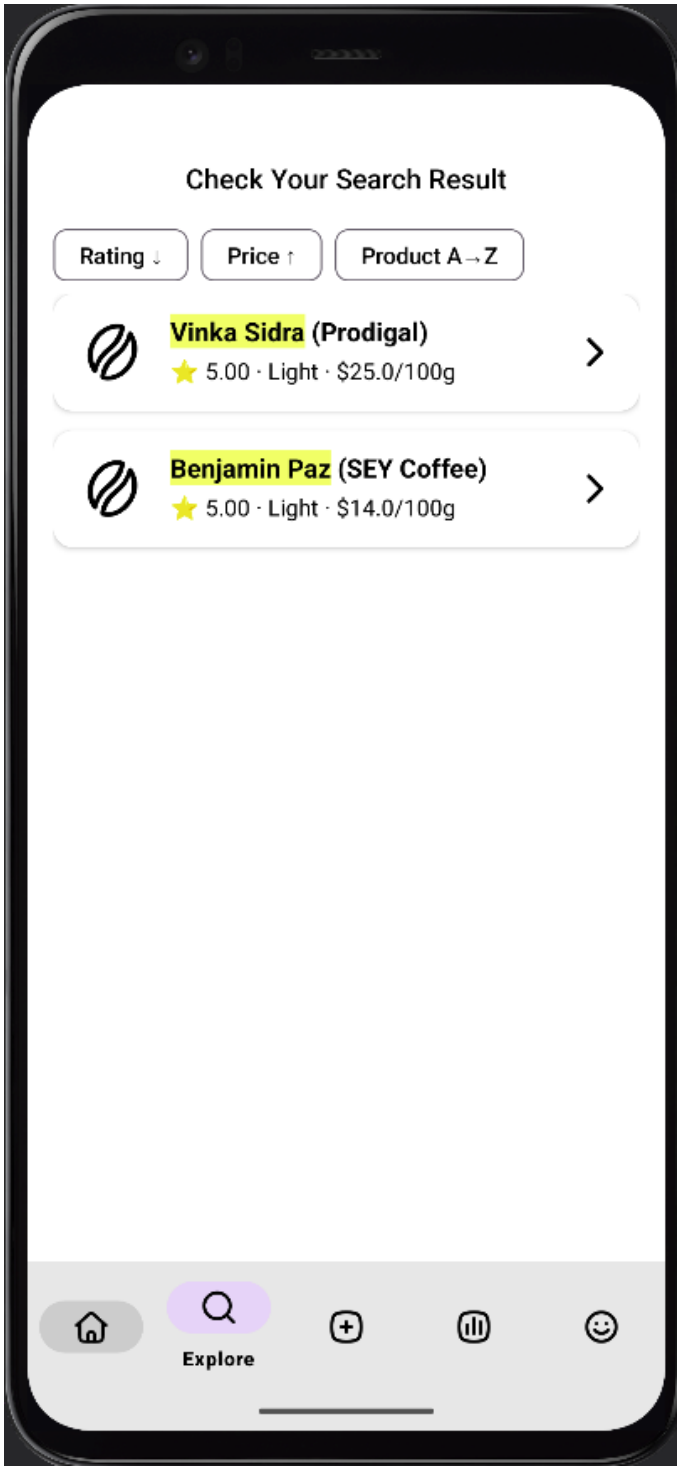

Check Your Search Result
Finca Esperanza
Region: Huila · Elevation: 1800m
Finca Test
Region: Huila · Elevation: 1800m
Finca Chami
Region: Huila · Elevation: 1650m
Finca Aroma Nativo
Region: Huila · Elevation: 1750m
Inmaculada Coffee Farm
Region: Huila · Elevation: 1900m
Finca Tio Conejo
Region: Huila · Elevation: 1900m
Aponte Village Smallholders
Region: Huila · Elevation: 2000m
Hachi Estate
Explore

6.7 g) Produce a list of all the top-rated products

A dedicated query returns the highest-rated coffee products across all logs. Results are sorted by average rating and total log count.

- product = ?, sort = rating, order = desc, limit = 2

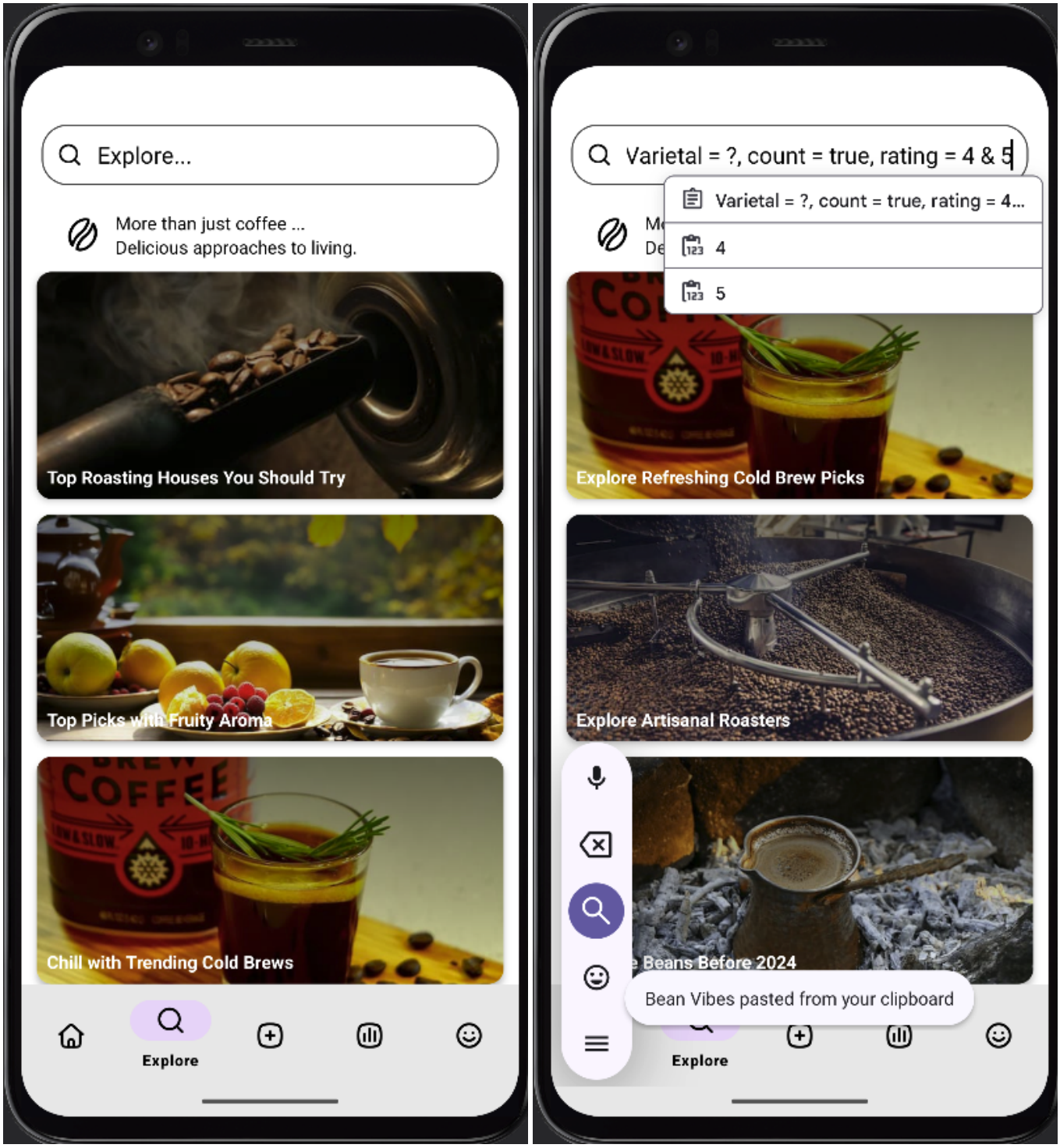


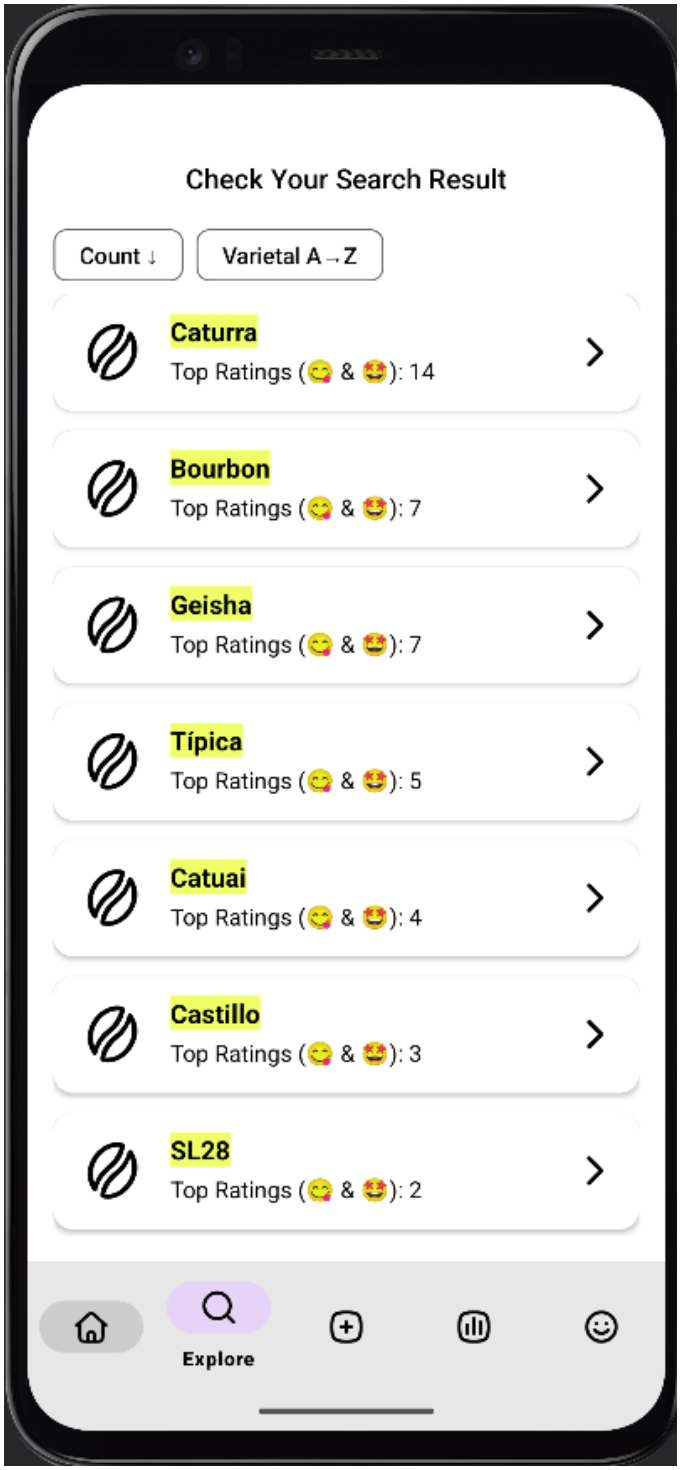

Check Your Search Result
Rating ↓
Price ↑
Product A→Z
Vinka Sidra (Prodigal)
5.00 · Light · $25.0/100g
Benjamin Paz (SEY Coffee)
5.00 · Light · $14.0/100g
Explore

6.8 h) Produce a list of the number of times each varietal has received one of the top-2 ratings

This query aggregates the number of times each varietal has received a rating of 4 or 5. Varietal name, count, and average rating are shown.

- Varietal = ?, count = true, rating = 4 & 5

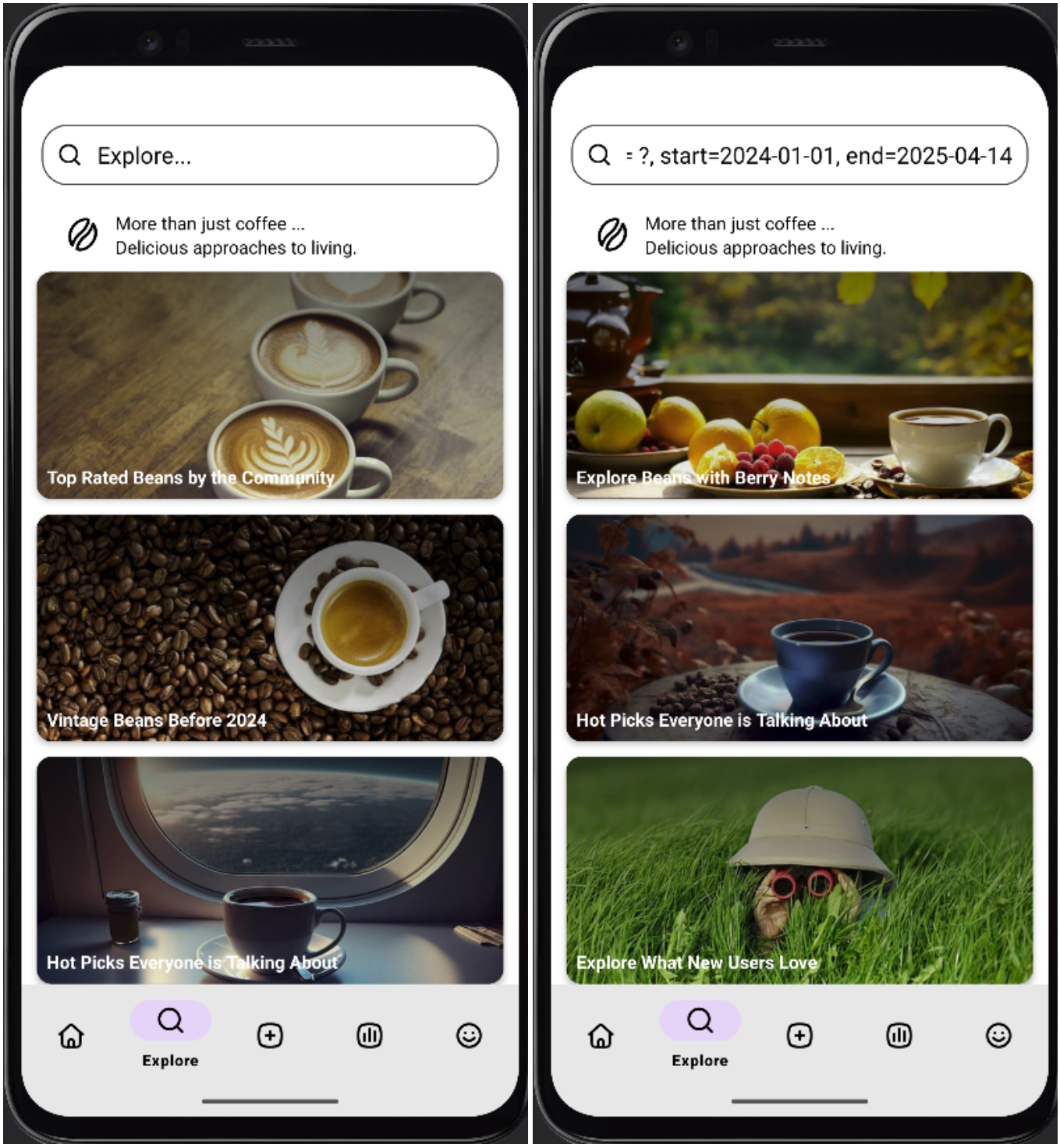


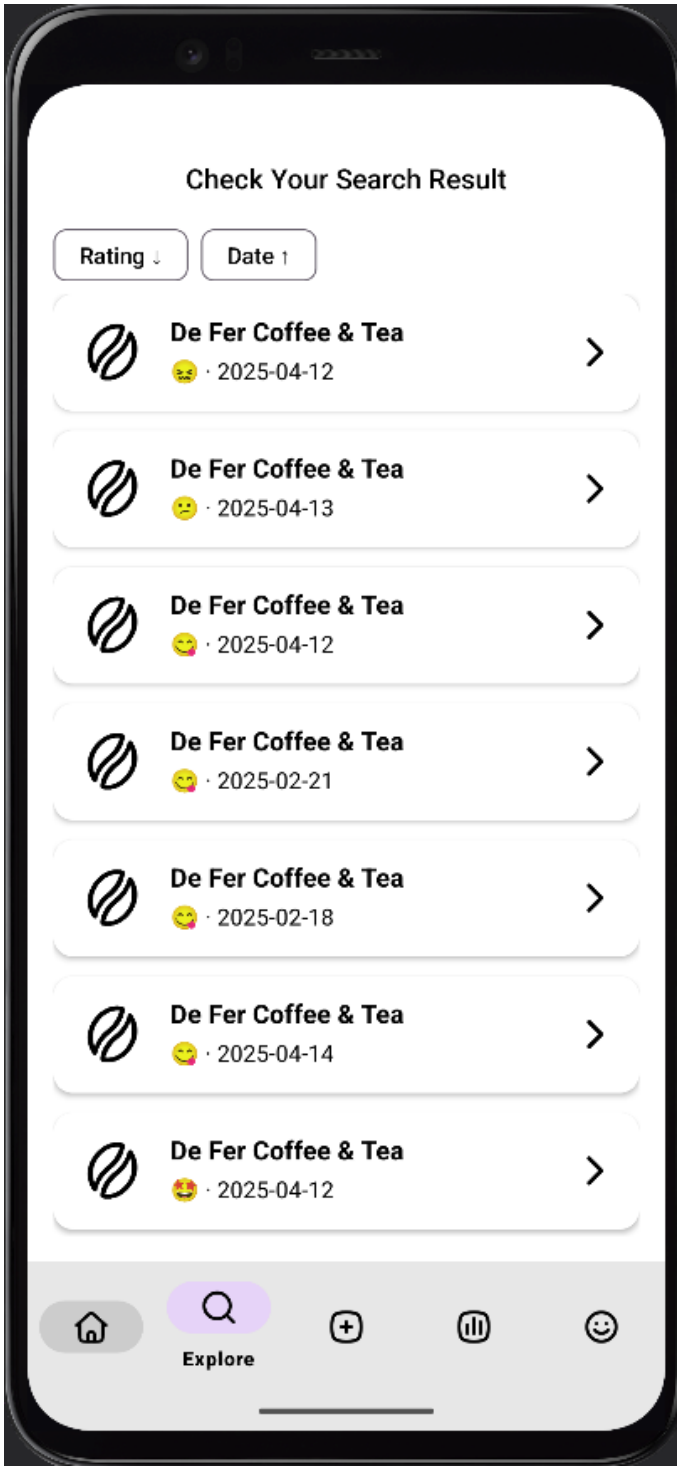

Check Your Search Result
Count ↓
Varietal A→Z
Caturra
Top Ratings (😋 & 🤩): 14
Bourbon
Top Ratings (😋 & 🤩): 7
Geisha
Top Ratings (😋 & 🤩): 7
Típica
Top Ratings (😋 & 🤩): 5
Catuai
Top Ratings (😋 & 🤩): 4
Castillo
Top Ratings (😋 & 🤩): 3
SL28
Top Ratings (😋 & 🤩): 2
Explore

6.9 i) Produce a log of tastings for a particular roaster in a time range

Users can view all logs associated with a specific roaster (e.g., "Verve") during a selected date range. Each log entry includes rating, date, and batch info.

- roaster= $ROASTER NAME, log = ?, start=$YYYY-MM-DD, end=$YYYY-MM-DD
- roaster=De Fer Coffee & Tea, log = ?, start=2024-01-01, end=2025-04-14

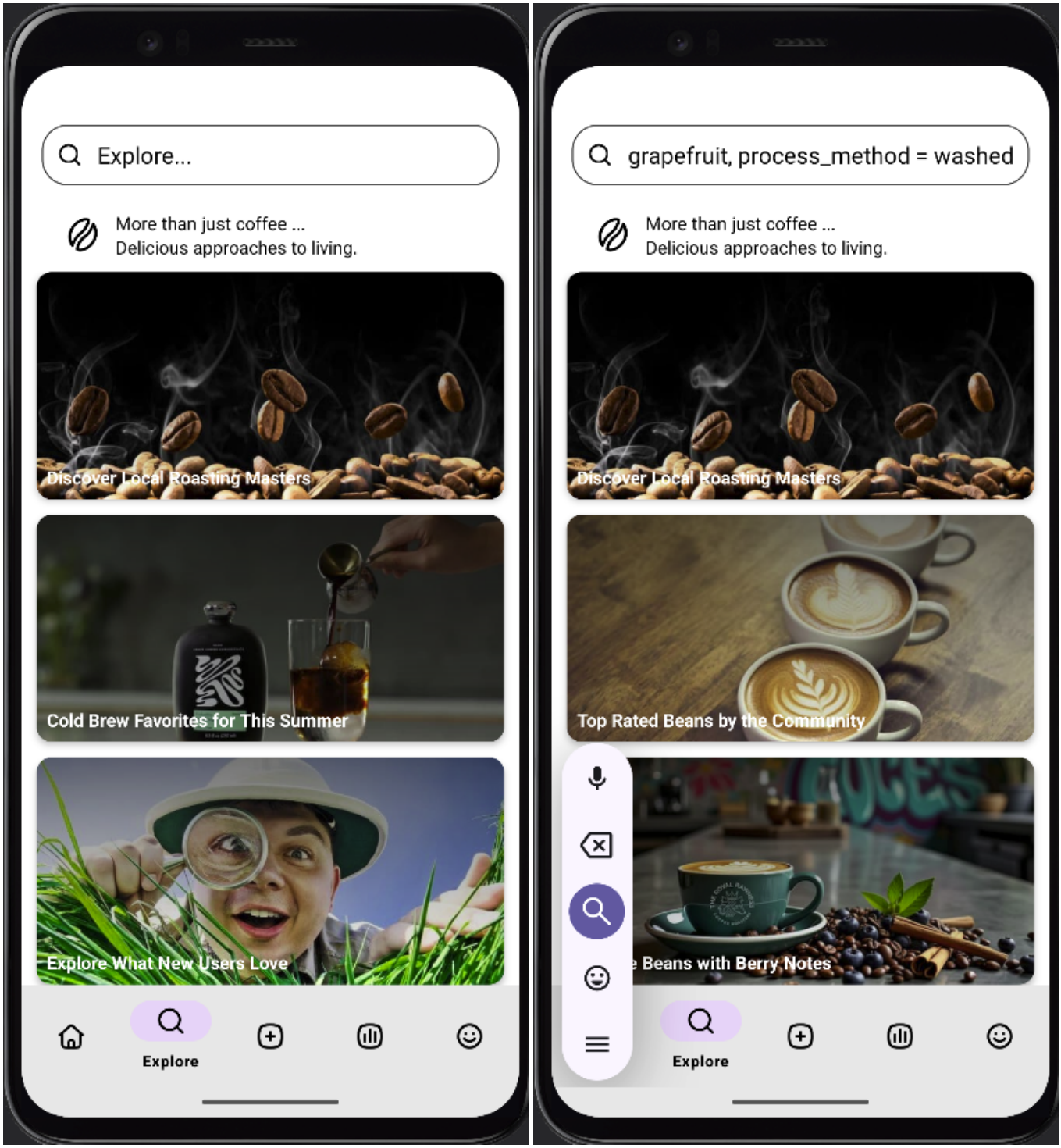


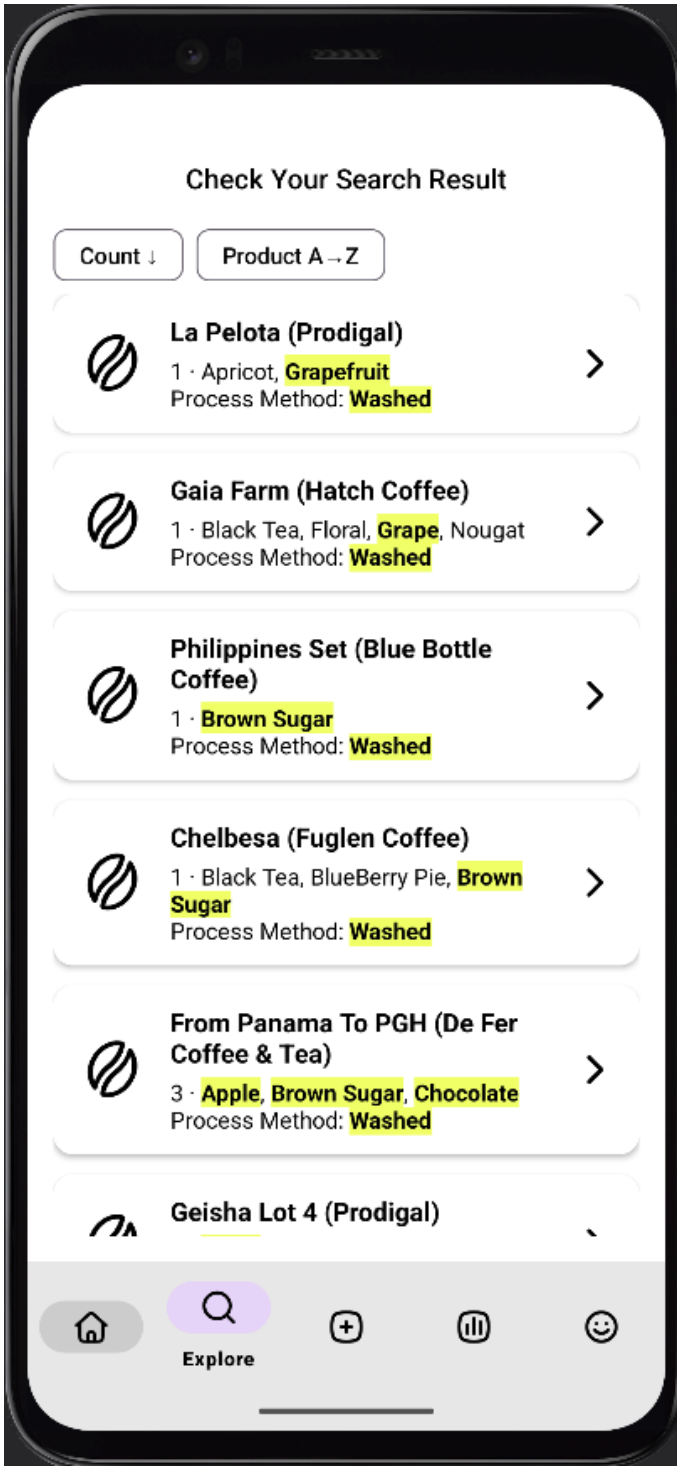

Check Your Search Result
Rating ↓
Date ↑
De Fer Coffee & Tea
😖 · 2025-04-12
De Fer Coffee & Tea
😕 · 2025-04-13
De Fer Coffee & Tea
😋 · 2025-04-12
De Fer Coffee & Tea
😋 · 2025-02-21
De Fer Coffee & Tea
😋 · 2025-02-18
De Fer Coffee & Tea
😋 · 2025-04-14
De Fer Coffee & Tea
🤩 · 2025-04-12
Explore

6.10 j) Produce a ranked list of products, given a set of notes and optionally a processing method

This feature returns products matching selected tasting notes and optionally filtered by process method. Matching notes are highlighted, and results are ranked.

- product = ?, tastingnote = $NOTE A & $NOTE B & xxx, process_method = $PROCESS METHOD
- product = ?, tastingnote = chocolate & apple & brown sugar & grape & grapefruit
- product = ?, tastingnote = chocolate & apple & brown sugar & grape & grapefruit, process_method = washed


Explore...
More than just coffee ...
Delicious approaches to living.
Discover Local Roasting Masters
Cold Brew Favorites for This Summer
Explore What New Users Love
Explore
grapefruit, process_method = washed
More than just coffee ...
Delicious approaches to living.
Discover Local Roasting Masters
Top Rated Beans by the Community
Beans with Berry Notes
Explore


Check Your Search Result
Count ↓
Product A→Z
La Pelota (Prodigal)
1 · Apricot, Grapefruit
Process Method: Washed
Gaia Farm (Hatch Coffee)
1 · Black Tea, Floral, Grape, Nougat
Process Method: Washed
Philippines Set (Blue Bottle Coffee)
1 · Brown Sugar
Process Method: Washed
Chelbesa (Fuglen Coffee)
1 · Black Tea, BlueBerry Pie, Brown Sugar
Process Method: Washed
From Panama To PGH (De Fer Coffee & Tea)
3 · Apple, Brown Sugar, Chocolate
Process Method: Washed
Geisha Lot 4 (Prodigal)
Explore

6.11 Report 1: Your Brew Style Preferences

This report analyzes the user's most frequently used combinations of brew method and grind size. It is parameterized by date range, minimum rating threshold, and result limit.

Query Logic: The SQL query joins `TastingLogs`, `BrewMethod`, `GrindSize`, and `Rating`. It filters logs by user ID, date range, and rating threshold, then groups by brew method and grind size to calculate average rating and log count. **SQL Query:**

```
SELECT
    bm.name AS brewMethod,
    gs.name AS grindSize,
    ROUND(AVG(r.rating_id), 2) AS avgRating,
    COUNT(*) AS totalCount
FROM TastingLogs t
JOIN BrewMethod bm ON t.brew_method_id = bm.brew_method_id
JOIN GrindSize gs ON t.grind_size_id = gs.grind_size_id
JOIN Rating r ON t.rating_id = r.rating_id
WHERE t.user_id = :userId
  AND t.test_date BETWEEN :startDate AND :endDate
  AND r.rating_id >= :minRating
GROUP BY bm.name, gs.name
ORDER BY totalCount DESC, avgRating DESC
LIMIT :limit
```

SQL Highlights:

- Joins: 4 tables
- Filtering: user ID, date range, min rating
- Aggregation: AVG and COUNT
- Grouping: by brew method and grind size
- Ordering: by log count and rating
- Parameterized: 4 dynamic user inputs

Complexity Score:

- Tables joined: +1 point
- Subquery count: 0
- Aggregates used: +1 point
- Grouping: +1 point
- Ordering fields >1: +1 point

- WHERE conditions: +1 point
- Domain justification: +1 point

Total Score: 6 points

Request JSON:

```
POST /api/complex-report/search
```

```
{
  "entity": "user_brew_grind",
  "query": {
    "user_id": "2",
    "startDate": "2024-01-01",
    "endDate": "2025-04-18",
    "minRating": "1",
    "limit": "10"
  }
}
```

Screenshot:

POST http://192.168.1.104:80

http://192.168.1.104:8080/api/complex-report/search

POST http://192.168.1.104:8080/api/complex-report/search

Params Authorization Headers (10) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary

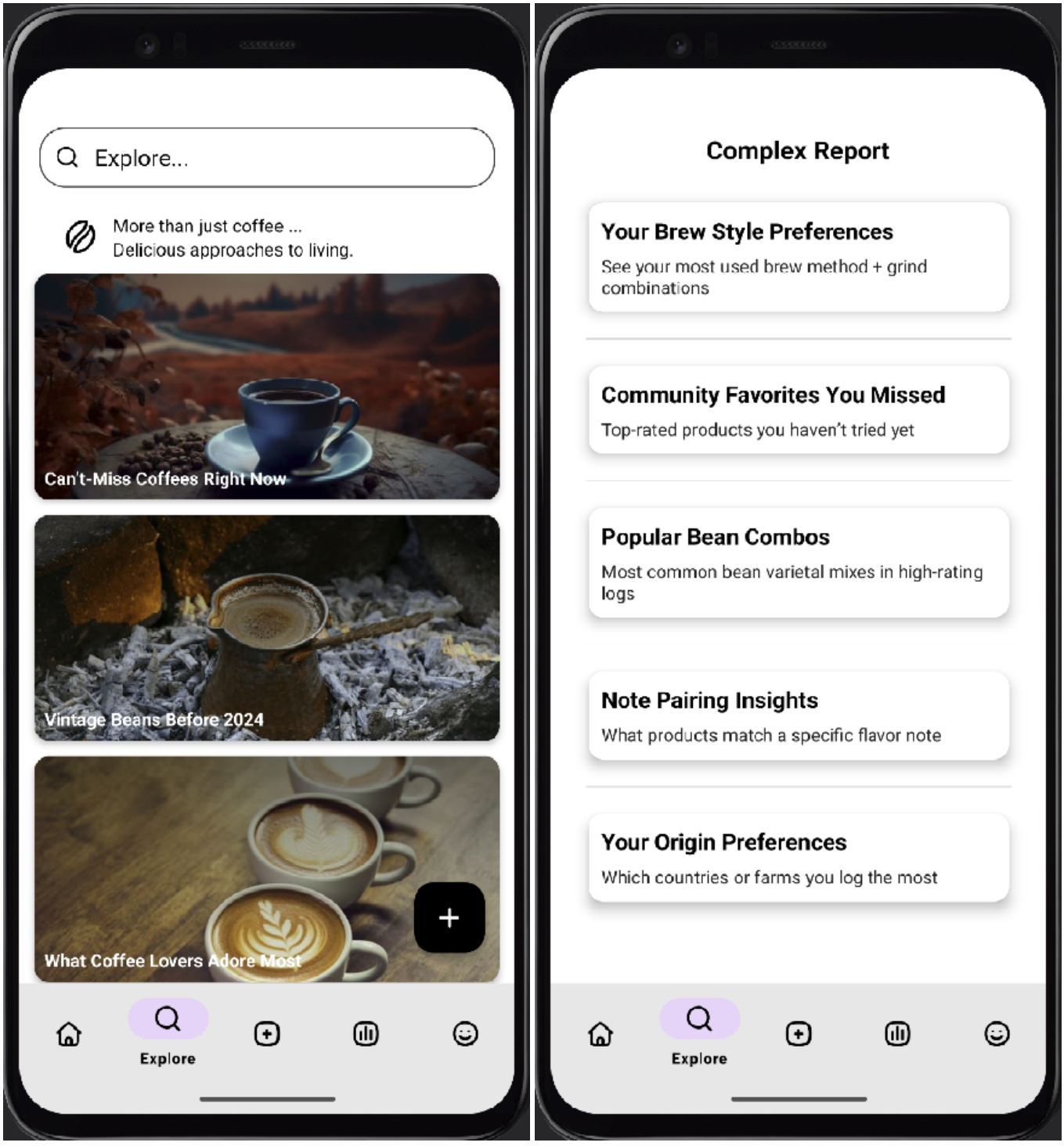
```
{
  "entity": "user_brew_grind",
  "query": {
    "user_id": "2",
    "startDate": "2024-01-01",
    "endDate": "2025-04-18",
    "minRating": "1",
    "limit": "10"
  }
}
```

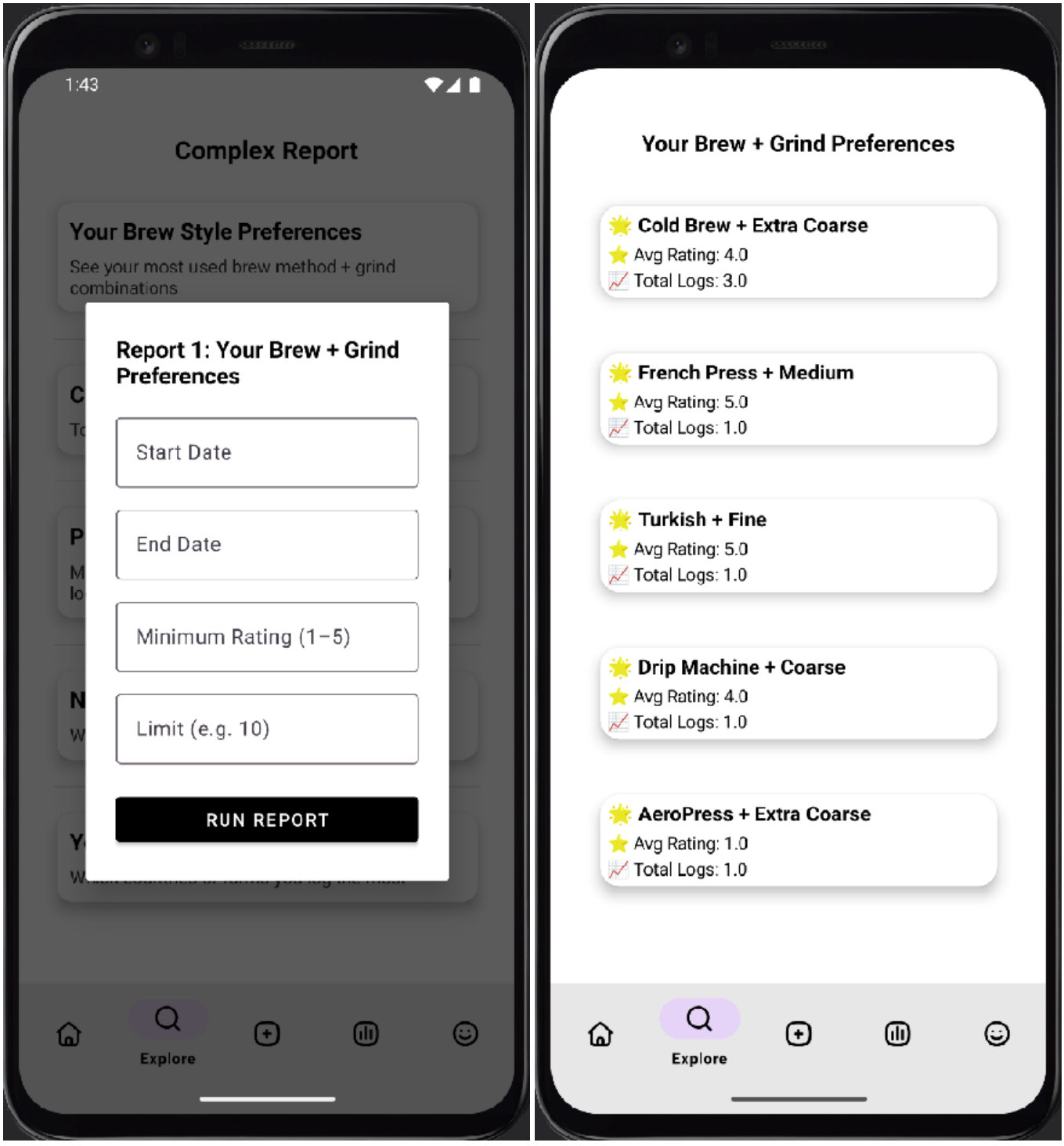
Body Cookies Headers (11) Test Results

{} JSON Preview Visualize

```
[
    {
        "logCount": 3,
        "avgRating": 4.0,
        "brewMethod": "Cold Brew",
        "grindSize": "Extra Coarse"
    },
    {
        "logCount": 1,
        "avgRating": 5.0,
        "brewMethod": "French Press",
        "grindSize": "Medium"
    },
    {
        "logCount": 1,
        "avgRating": 5.0,
        "brewMethod": "Turkish",
        "grindSize": "Fine"
    },
    {
```

```
// Complex Report 1 BrewMethod + GrindSize
@Query(value = """ 1 usage  Erdun E
    SELECT
        bm.name AS brewMethod,
        gs.name AS grindSize,
        ROUND(AVG(r.rating_id), 2) AS avgRating,
        COUNT(*) AS totalCount
    FROM TastingLogs t
    JOIN BrewMethod bm ON t.brew_method_id = bm.brew_method_id
    JOIN GrindSize gs ON t.grind_size_id = gs.grind_size_id
    JOIN Rating r ON t.rating_id = r.rating_id
    WHERE t.user_id = :userId
      AND t.test_date BETWEEN :startDate AND :endDate
      AND r.rating_id >= :minRating
    GROUP BY bm.name, gs.name
    ORDER BY totalCount DESC, avgRating DESC
    LIMIT :limit
""", nativeQuery = true)
List<UserBrewGrindReportProjection> findUserBrewGrindStats(
        @Param("userId") int userId,
        @Param("startDate") LocalDate startDate,
        @Param("endDate") LocalDate endDate,
        @Param("minRating") int minRating,
        @Param("limit") int limit
);   E, Today • Updated Complext Report
```


Explore...
More than just coffee ...
Delicious approaches to living.
Can't-Miss Coffees Right Now
Vintage Beans Before 2024
What Coffee Lovers Adore Most
Explore
Complex Report
Your Brew Style Preferences
See your most used brew method + grind combinations
Community Favorites You Missed
Top-rated products you haven't tried yet
Popular Bean Combos
Most common bean varietal mixes in high-rating logs
Note Pairing Insights
What products match a specific flavor note
Your Origin Preferences
Which countries or farms you log the most
Explore



This report helps users understand which brew and grind combinations they prefer most, based on their historical tasting logs. This insight can guide future purchases and brewing habits.

6.12 Report 2: Popular Products You Haven't Tried

This report displays a list of highly-rated products that the user has never logged before. It helps users discover well-reviewed coffees they haven't explored yet.

Query Logic: The SQL query retrieves products with an average rating greater than or equal to a user-defined threshold. It excludes any product the user has logged before by using a subquery. The results are ranked by rating and popularity.

SQL Query:

```
SELECT
    p.product_id AS productId,
    p.name AS productName,
    r.name AS roasterName,
    ROUND(AVG(rt.rating_id), 2) AS avgRating,
    COUNT(*) AS logCount
FROM TastingLogs t
JOIN RoastBatch rb ON t.batch_id = rb.batch_id
JOIN Products p ON rb.product_id = p.product_id
JOIN Roasters r ON p.roaster_id = r.roaster_id
JOIN Rating rt ON t.rating_id = rt.rating_id
WHERE rt.rating_id >= :minRating
  AND p.product_id NOT IN (
    SELECT DISTINCT p2.product_id
    FROM TastingLogs t2
    JOIN RoastBatch rb2 ON t2.batch_id = rb2.batch_id
    JOIN Products p2 ON rb2.product_id = p2.product_id
    WHERE t2.user_id = :userId
  )
GROUP BY p.product_id, p.name, r.name
ORDER BY avgRating DESC, logCount DESC
LIMIT :limit
```

SQL Highlights:

- Tables joined: 5
- Subquery used: yes (NOT IN clause to exclude user-logged products)
- Aggregation: AVG, COUNT
- Grouping: product and roaster
- Filtering: by rating and user ID
- Ordering: by average rating and log count
- Parameterized inputs: userId, minRating, limit

Complexity Score:

- Tables joined: +1 point
- Subquery (NOT IN): +1 point
- Aggregation functions: +1 point
- Grouping: +1 point
- WHERE condition (non-join): +1 point
- Ordering: +1 point
- Domain justification: +1 point

Total Score: 7 points (Complex)

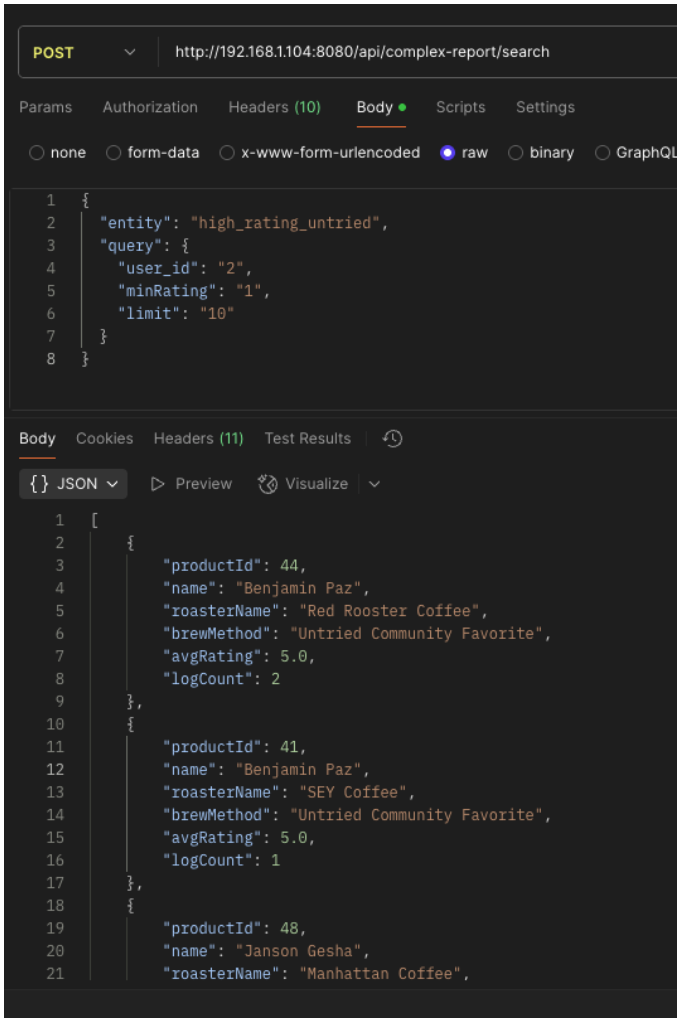
Request JSON:

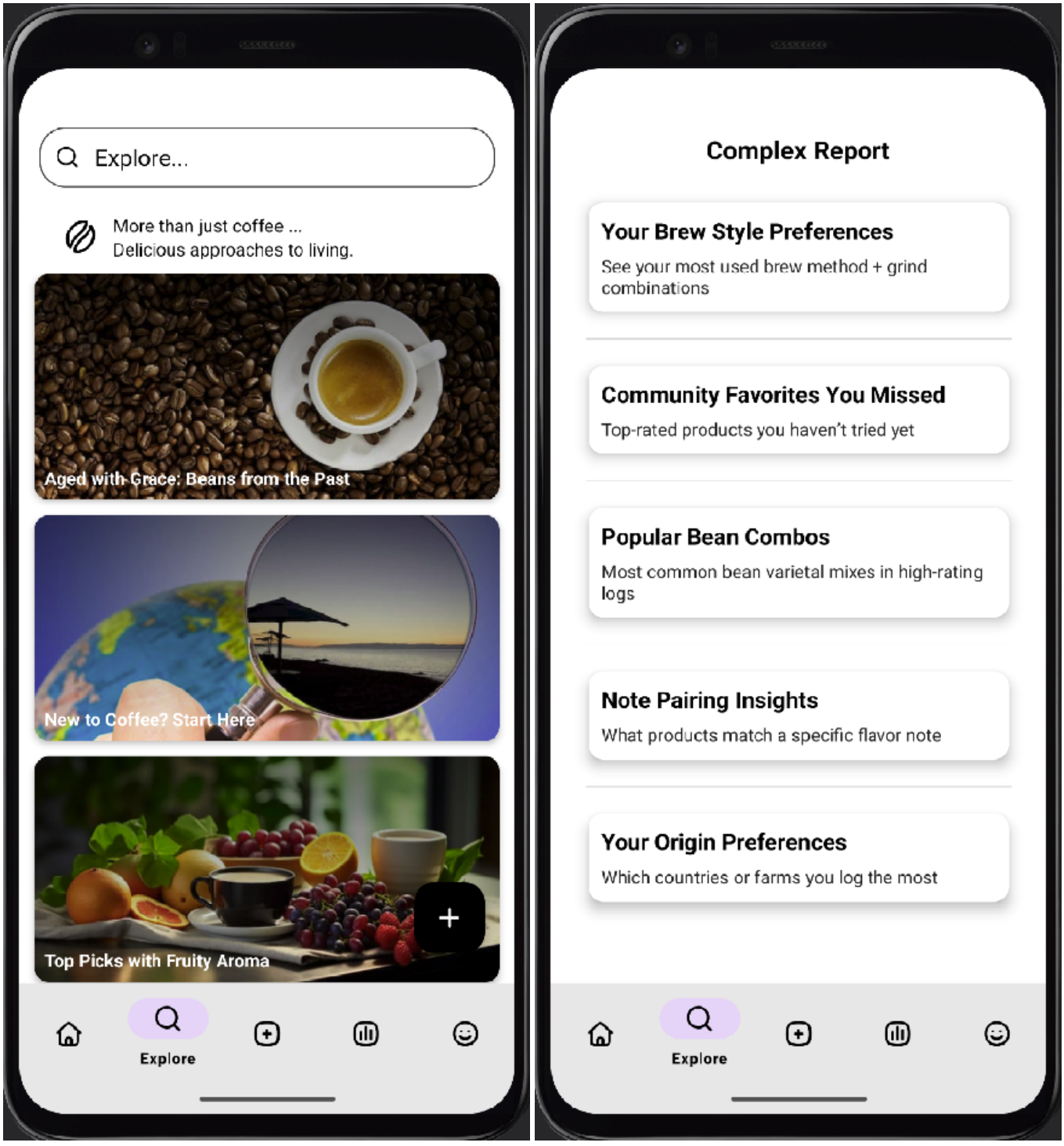
```
POST /api/complex-report/search
```

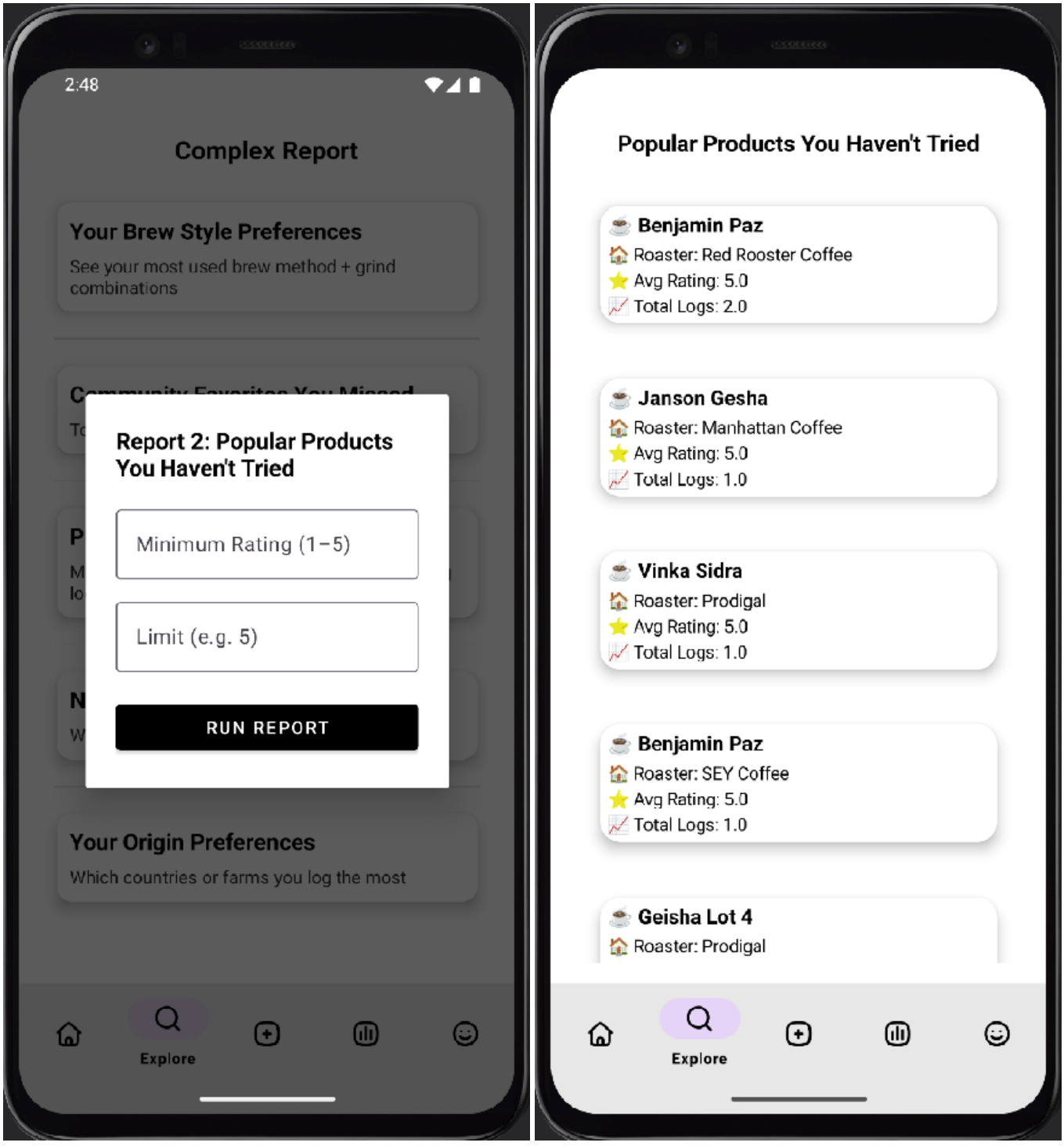
```
{
  "entity": "high_rating_untried",
  "query": {
    "user_id": "2",
    "minRating": "1",
    "limit": "10"
  }
}
```

Screenshot:

```
    @Query(value = """  1 usage   Erdun E
    SELECT
        p.product_id AS productId,
        p.name AS productName,
        r.name AS roasterName,
        ROUND(AVG(rt.rating_id), 2) AS avgRating,
        COUNT(*) AS logCount
    FROM TastingLogs t
    JOIN RoastBatch rb ON t.batch_id = rb.batch_id
    JOIN Products p ON rb.product_id = p.product_id
    JOIN Roasters r ON p.roaster_id = r.roaster_id
    JOIN Rating rt ON t.rating_id = rt.rating_id
    WHERE rt.rating_id >= :minRating
      AND p.product_id NOT IN (
        SELECT DISTINCT p2.product_id
        FROM TastingLogs t2
        JOIN RoastBatch rb2 ON t2.batch_id = rb2.batch_id
        JOIN Products p2 ON rb2.product_id = p2.product_id
        WHERE t2.user_id = :userId
      )
    GROUP BY p.product_id, p.name, r.name
    ORDER BY avgRating DESC, logCount DESC
    LIMIT :limit
""", nativeQuery = true)
    List<HighRatingUntriedProductProjection> findHighRatedUntriedProducts(
            @Param("userId") int userId,
            @Param("minRating") int minRating,
            @Param("limit") int limit
    );
```




Explore...
More than just coffee ...
Delicious approaches to living.
Aged with Grace: Beans from the Past
New to Coffee? Start Here
Top Picks with Fruity Aroma
Explore
Complex Report
Your Brew Style Preferences
See your most used brew method + grind combinations
Community Favorites You Missed
Top-rated products you haven't tried yet
Popular Bean Combos
Most common bean varietal mixes in high-rating logs
Note Pairing Insights
What products match a specific flavor note
Your Origin Preferences
Which countries or farms you log the most
Explore



This report surfaces community favorites that the user hasn't discovered yet. It's especially useful for new users seeking high-quality recommendations aligned with public feedback.

6.13 Report 3: Popular Bean Combinations

This report identifies the most frequently logged combinations of bean varietals that received high ratings. The query is parameterized by minimum number of beans in the combo, minimum rating, and result limit.

Query Logic: The SQL query joins `TastingLogs`, `RoastBatch`, `Products`, `ProductBean`, `Beans`, `Varietal`, and `Rating`. It groups each log by its unique set of bean varietals and filters out combinations that don't meet the minimum count or rating requirement. The outer query then computes log count and average rating for each unique varietal combination.

SQL Query:

```
SELECT
    bean_combo,
    ROUND(AVG(rating_id), 2) AS avgRating,
    COUNT(*) AS logCount
FROM (
    SELECT
        t.log_id,
        GROUP_CONCAT(DISTINCT v.name ORDER BY v.name SEPARATOR ', ') AS bean_combo,
        r.rating_id
    FROM TastingLogs t
    JOIN RoastBatch rb ON t.batch_id = rb.batch_id
    JOIN Products p ON rb.product_id = p.product_id
    JOIN ProductBean pb ON p.product_id = pb.product_id
    JOIN Beans b ON pb.bean_id = b.bean_id
    JOIN Varietal v ON b.varietal_id = v.varietal_id
    JOIN Rating r ON t.rating_id = r.rating_id
    GROUP BY t.log_id
    HAVING COUNT(DISTINCT v.varietal_id) >= :minBeans AND rating_id >= :minRating
) AS combo_logs
GROUP BY bean_combo
ORDER BY logCount DESC, avgRating DESC
LIMIT :limit
```

SQL Highlights:

- Joins: 7 tables
- Subqueries: 1 (derived table with `GROUP_CONCAT`)
- Aggregates: AVG, COUNT, GROUP_CONCAT
- Grouping: by bean combination
- Filtering: min rating, min beans
- Non-aggregation function: `GROUP_CONCAT`

- Parameterized: 3 user inputs

Complexity Score:

- Tables joined: +1 point
- Subquery used: +1 point
- Aggregates: +1 point
- Grouping: +1 point
- Ordering fields >1: +1 point
- Non-aggregation function in SELECT: +1 point
- WHERE/HAVING conditions: +1 point
- Strong domain use: +1 point

Total Score: 8 points

Request JSON:

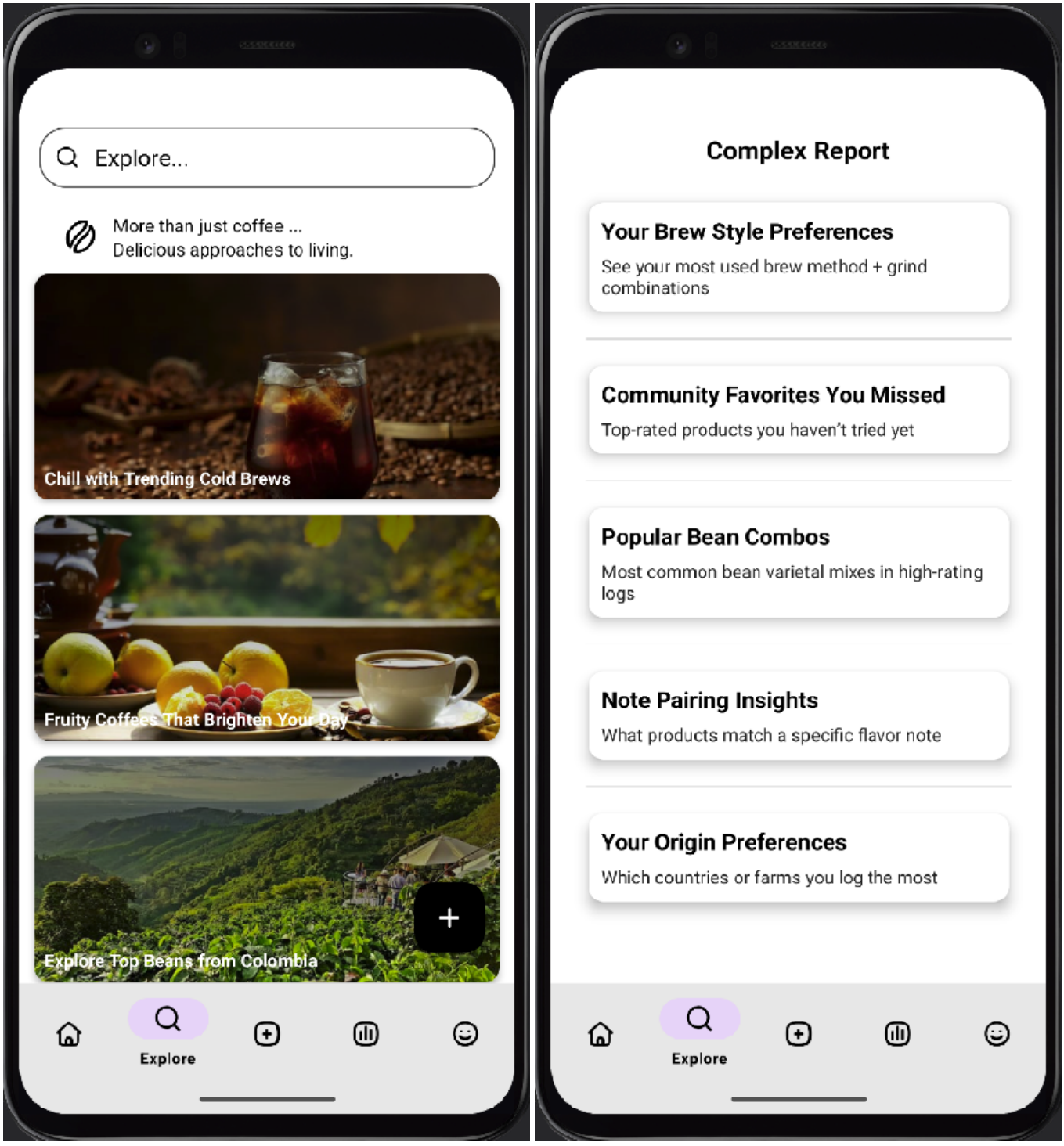
```
POST /api/complex-report/search

{
  "entity": "popular_bean_combos",
  "query": {
    "user_id": "2",
    "minBeans": "1",
    "minRating": "1",
    "limit": "10"
  }
}
```

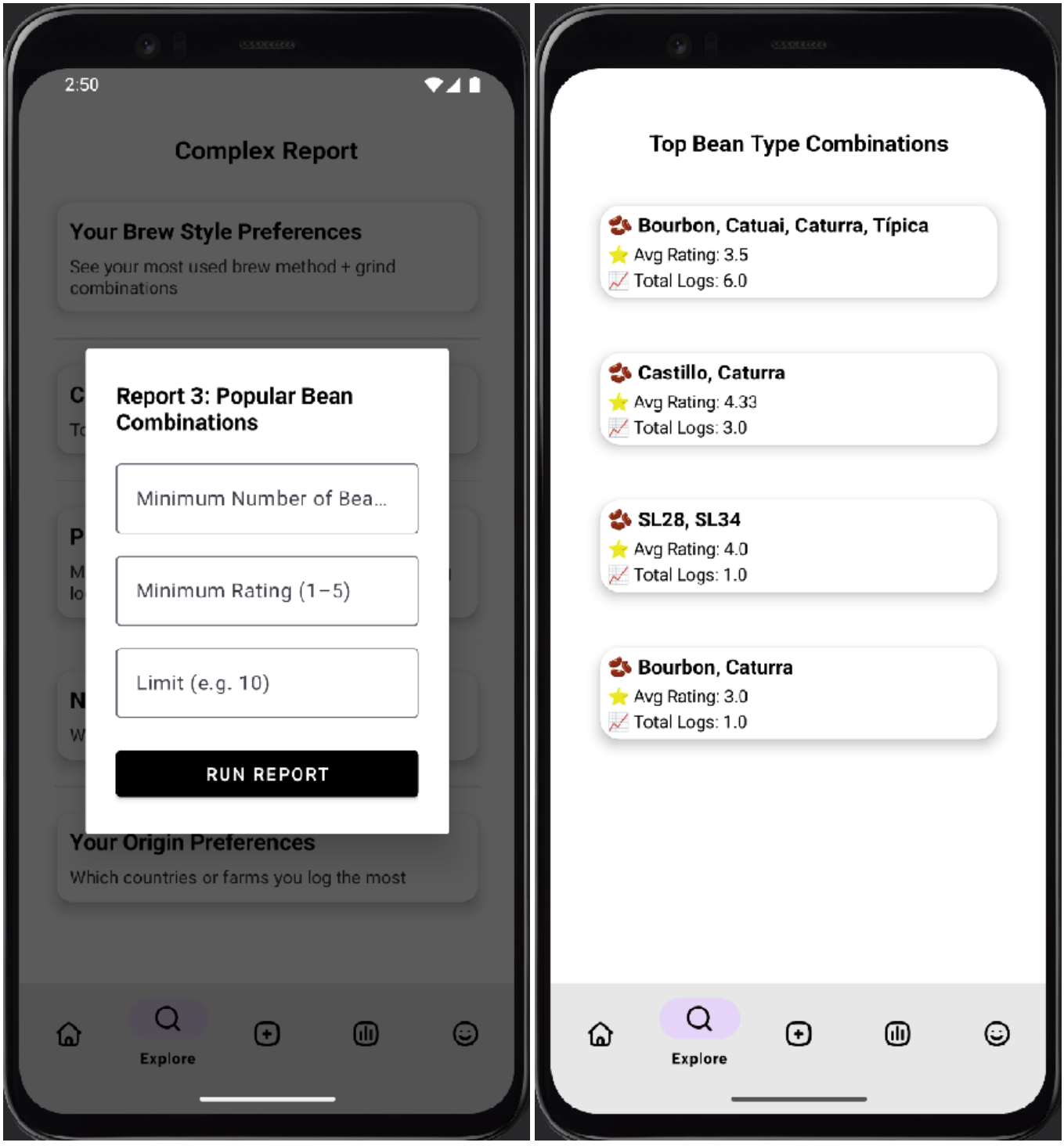
Screenshot:

```
@Query(value = """ 1 usage  Erdun E
    SELECT
        bean_combo,
        ROUND(AVG(rating_id), 2) AS avgRating,
        COUNT(*) AS logCount
    FROM (
        SELECT   E, Today • Updated Complext Report
            t.log_id,
            GROUP_CONCAT(DISTINCT v.name ORDER BY v.name SEPARATOR ', ') AS bean_combo,
            r.rating_id
        FROM TastingLogs t
        JOIN RoastBatch rb ON t.batch_id = rb.batch_id
        JOIN Products p ON rb.product_id = p.product_id
        JOIN ProductBean pb ON p.product_id = pb.product_id
        JOIN Beans b ON pb.bean_id = b.bean_id
        JOIN Varietal v ON b.varietal_id = v.varietal_id
        JOIN Rating r ON t.rating_id = r.rating_id
        GROUP BY t.log_id
        HAVING COUNT(DISTINCT v.varietal_id) >= :minBeans AND rating_id >= :minRating
    ) AS combo_logs
    GROUP BY bean_combo
    ORDER BY logCount DESC, avgRating DESC
    LIMIT :limit
""", nativeQuery = true)
    List<PopularBeanComboProjection> findPopularBeanCombos(
            @Param("minBeans") int minBeans,
            @Param("minRating") int minRating,
            @Param("limit") int limit
    );
```

```
POST  http://192.168.1.104:8080/api/complex-report/search

Params  Authorization  Headers (10)  Body •  Scripts  Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

{
  "entity": "popular_bean_combos",
  "query": {
    "user_id": "2",
    "minBeans": "1",
    "minRating": "1",
    "limit": "10"
  }
}
```

```
Body  Cookies  Headers (11)  Test Results

{} JSON  Preview  Visualize

[
    {
        "productId": -1,
        "name": "Geisha",
        "roasterName": "Bean Combo",
        "brewMethod": null,
        "avgRating": 3.9,
        "logCount": 10
    },
    {
        "productId": -1,
        "name": "Bourbon, Catuai, Caturra, Tipica",
        "roasterName": "Bean Combo",
        "brewMethod": null,
        "avgRating": 3.5,
        "logCount": 6
    },
    {
        "productId": -1,
        "name": "Bourbon",
        "roasterName": "Bean Combo",
```


Explore...
More than just coffee ...
Delicious approaches to living.
Chill with Trending Cold Brews
Fruity Coffees That Brighten Your Day
Explore Top Beans from Colombia
Explore
Complex Report
Your Brew Style Preferences
See your most used brew method + grind combinations
Community Favorites You Missed
Top-rated products you haven't tried yet
Popular Bean Combos
Most common bean varietal mixes in high-rating logs
Note Pairing Insights
What products match a specific flavor note
Your Origin Preferences
Which countries or farms you log the most
Explore



This report surfaces powerful insights into what varietal combinations contribute most to positively rated coffee experiences. It aids in discovering favored bean pairings among the community.

6.14 Report 4: Note Pairing Insights

This report helps users find products that best match a selected set of tasting notes, optionally filtered by a specific process method (e.g., "Washed"). It highlights overlapping notes and ranks products by matched note count.

Query Logic: The query scans through tasting notes associated with each product and uses conditional 'GROUP_CONCAT' to identify which of the selected notes are matched. Optional filtering by process method is supported.

SQL Query:

```
SELECT
    p.product_id AS productId,
    p.name AS name,
    r.name AS roasterName,
    pm.name AS processMethod,
    GROUP_CONCAT(DISTINCT tn.note_name) AS allTastingNotes,
    GROUP_CONCAT(DISTINCT
        CASE
            WHEN LOWER(tn.note_name) IN (:noteList) THEN tn.note_name
            ELSE NULL
        END
    ) AS matchedTastingNotes
FROM Products p
JOIN Roasters r ON p.roaster_id = r.roaster_id
JOIN ProductTastingNote ptn ON p.product_id = ptn.product_id
JOIN TastingNote tn ON ptn.note_id = tn.note_id
LEFT JOIN ProductBean pb ON p.product_id = pb.product_id
LEFT JOIN Beans b ON pb.bean_id = b.bean_id
LEFT JOIN ProcessMethod pm ON b.process_method_id = pm.process_method_id
WHERE (:processMethod IS NULL OR LOWER(pm.name) = LOWER(:processMethod))
GROUP BY p.product_id
HAVING COUNT(DISTINCT CASE WHEN LOWER(tn.note_name) IN (:noteList) THEN tn.note_name END) >
ORDER BY matchedTastingNotes DESC
LIMIT :limit
```

SQL Highlights:

- Tables joined: 7 (Products, Roasters, ProductTastingNote, TastingNote, ProductBean, Beans, ProcessMethod)
- LEFT JOINs: supported for optional process method
- Conditional GROUP_CONCAT and COUNT inside HAVING clause
- Parameterized inputs: note list + optional process method

Complexity Score:

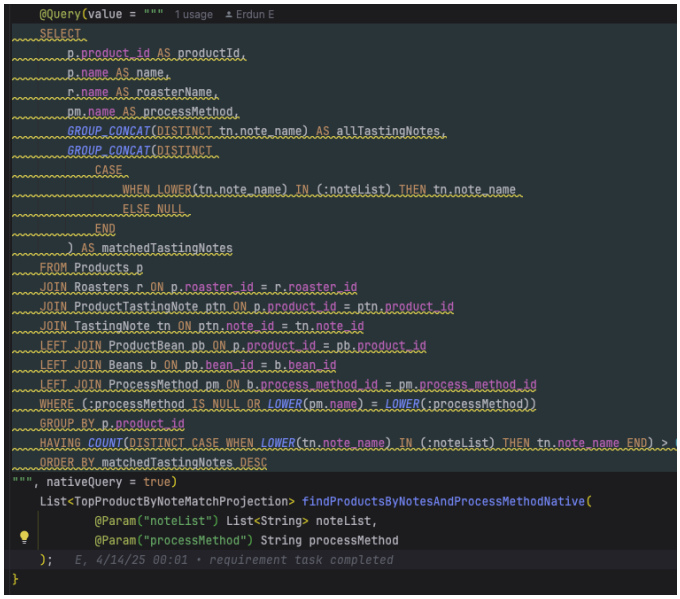
- Tables joined 3: +1
- Non-inner join (LEFT): +1
- Aggregate functions (GROUP_CONCAT, COUNT): +1
- Conditional expressions in SELECT + HAVING: +2
- WHERE + HAVING filters (non-join): +1
- Domain relevance: +1

Total Score: 7 (Complex)

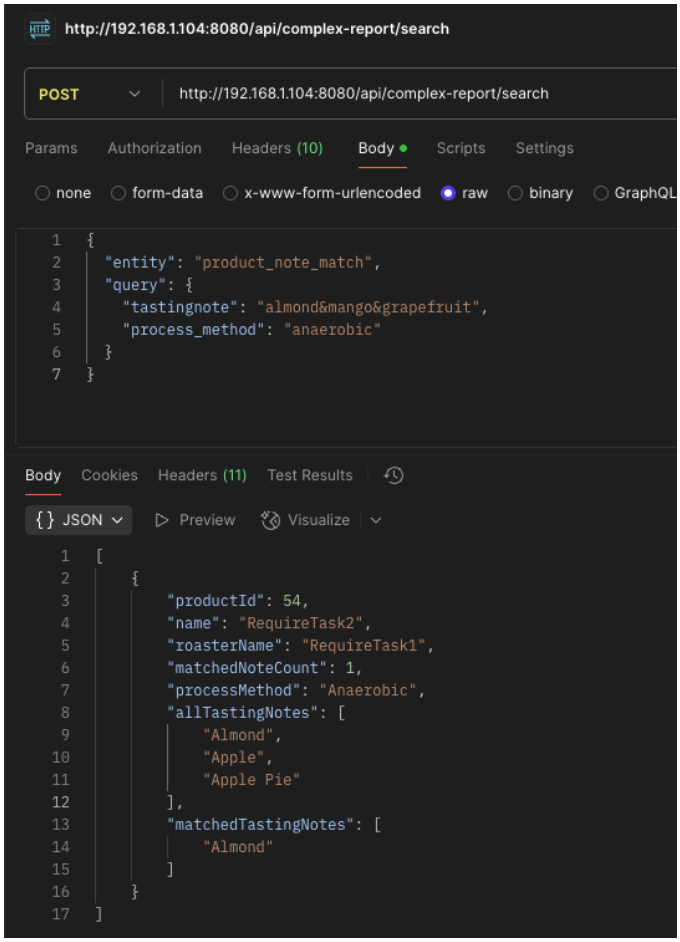
Example Request JSON:

```
POST /api/complex-report/search

{
  "entity": "product_note_match",
  "query": {
    "tastingnote": "almond&mango&grapefruit",
    "process_method": "anaerobic"
  }
}
```

Screenshot:



```
    @Query(value = """ 1 usage   ± Erdun E
    SELECT
        p.product_id AS productId,
        p.name AS name,
        r.name AS roasterName,
        pm.name AS processMethod,
        GROUP_CONCAT(DISTINCT tn.note_name) AS allTastingNotes,
        GROUP_CONCAT(DISTINCT
            CASE
                WHEN LOWER(tn.note_name) IN (:noteList) THEN tn.note_name
                ELSE NULL
            END
        ) AS matchedTastingNotes
    FROM Products p
    JOIN Roasters r ON p.roaster_id = r.roaster_id
    JOIN ProductTastingNote ptn ON p.product_id = ptn.product_id
    JOIN TastingNote tn ON ptn.note_id = tn.note_id
    LEFT JOIN ProductBean pb ON p.product_id = pb.product_id
    LEFT JOIN Beans b ON pb.bean_id = b.bean_id
    LEFT JOIN ProcessMethod pm ON b.process_method_id = pm.process_method_id
    WHERE (:processMethod IS NULL OR LOWER(pm.name) = LOWER(:processMethod))
    GROUP BY p.product_id
    HAVING COUNT(DISTINCT CASE WHEN LOWER(tn.note_name) IN (:noteList) THEN tn.note_name END) >
    ORDER BY matchedTastingNotes DESC
""", nativeQuery = true)
    List<TopProductByNoteMatchProjection> findProductsByNotesAndProcessMethodNative(
            @Param("noteList") List<String> noteList,
            @Param("processMethod") String processMethod
    );   E, 4/14/25 00:01 • requirement task completed
}
```



http://192.168.1.104:8080/api/complex-report/search

POST http://192.168.1.104:8080/api/complex-report/search

Params Authorization Headers (10) Body Scripts Settings

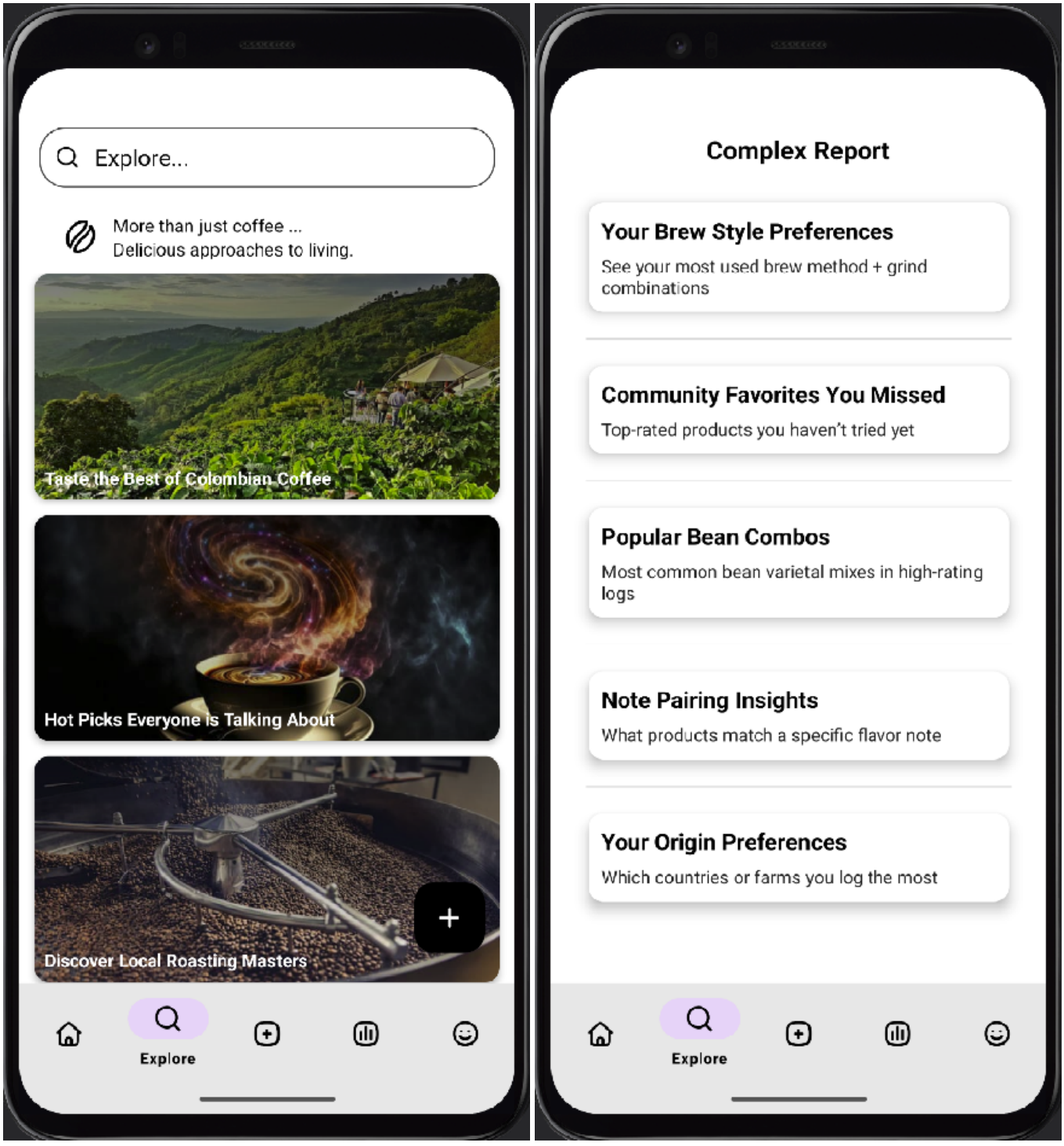
none form-data x-www-form-urlencoded raw binary GraphQL

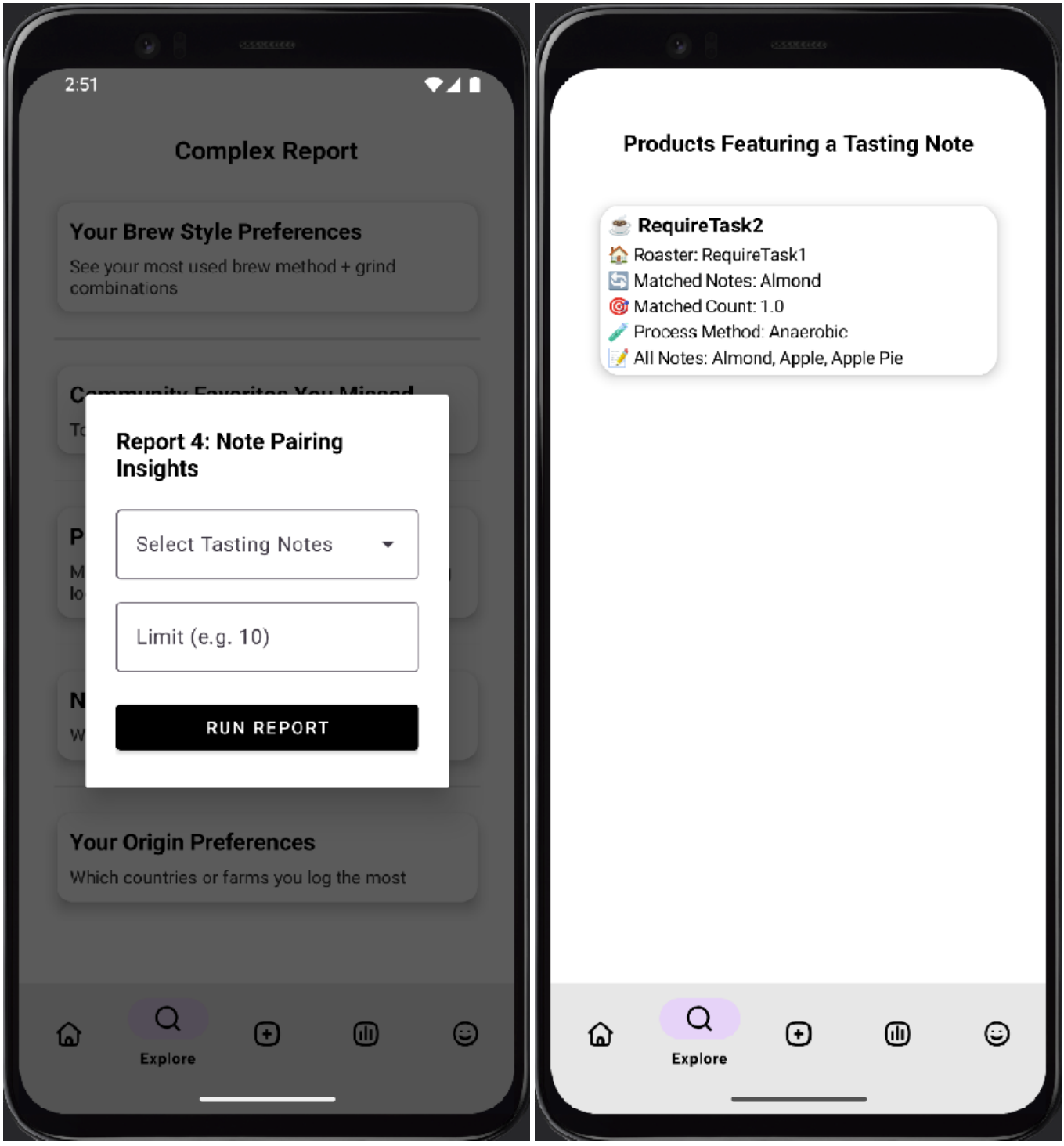
```
{
  "entity": "product_note_match",
  "query": {
    "tastingnote": "almond&mango&grapefruit",
    "process_method": "anaerobic"
  }
}
```

Body Cookies Headers (11) Test Results

{} JSON Preview Visualize

```
[
    {
        "productId": 54,
        "name": "RequireTask2",
        "roasterName": "RequireTask1",
        "matchedNoteCount": 1,
        "processMethod": "Anaerobic",
        "allTastingNotes": [
            "Almond",
            "Apple",
            "Apple Pie"
        ],
        "matchedTastingNotes": [
            "Almond"
        ]
    }
]
```


Explore...
More than just coffee ...
Delicious approaches to living.
Taste the Best of Colombian Coffee
Hot Picks Everyone is Talking About
Discover Local Roasting Masters
Explore
Complex Report
Your Brew Style Preferences
See your most used brew method + grind combinations
Community Favorites You Missed
Top-rated products you haven't tried yet
Popular Bean Combos
Most common bean varietal mixes in high-rating logs
Note Pairing Insights
What products match a specific flavor note
Your Origin Preferences
Which countries or farms you log the most
Explore



This report allows flavor-driven exploration by identifying coffees that align with user-selected tasting notes. It helps users discover new products that match their flavor preferences.

6.15 Report 5: Your Origin Preferences

This report summarizes the user's tasting trends based on selected dimensions such as brew method, grind size, or roast level. It returns each category's average rating and log count. The user can select the dimension to analyze and limit the number of results.

Query Logic: The backend supports three SQL branches depending on the selected trend dimension (‘brew_method‘, ‘grind_size‘, or ‘roast_level‘). Each query groups logs by the selected category, then computes average rating and total log count for that category.

SQL Query: (example for roast level)

```
SELECT
    rl.name AS category,
    ROUND(AVG(rt.rating_id), 2) AS avgRating,
    COUNT(*) AS logCount
FROM TastingLogs t
JOIN RoastBatch rb ON t.batch_id = rb.batch_id
JOIN Products p ON rb.product_id = p.product_id
JOIN RoastLevel rl ON p.roast_level_id = rl.roast_level_id
JOIN Rating rt ON t.rating_id = rt.rating_id
WHERE t.user_id = :userId
GROUP BY rl.name
ORDER BY logCount DESC, avgRating DESC
LIMIT :limit
```

SQL Highlights:

- Joins: 5 tables (TastingLogs, RoastBatch, Products, RoastLevel, Rating)
- Filtering: user ID
- Aggregation: AVG and COUNT
- Grouping: by selected origin category
- Ordering: by log count and rating
- Parameterized: 2 dynamic inputs (user ID and result limit)

Complexity Score:

- Tables joined (3): +1
- Aggregation (AVG, COUNT): +1
- Grouping used: +1
- Ordering >1 field: +1

- WHERE filter: +1
- Clear domain purpose: +1

Total Score: 6 points

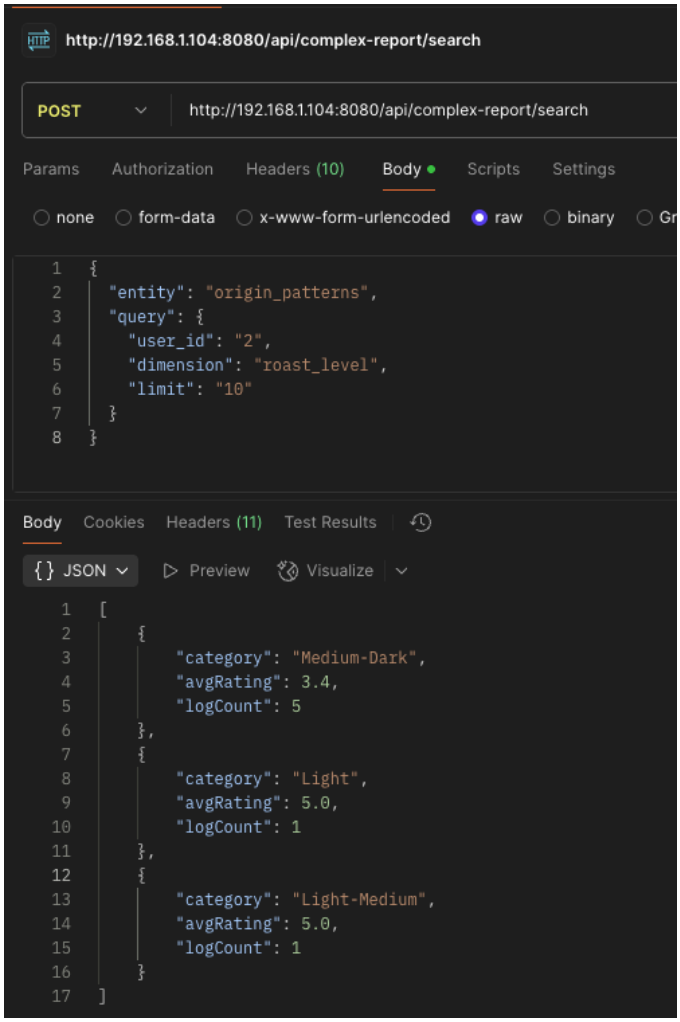
Request JSON:

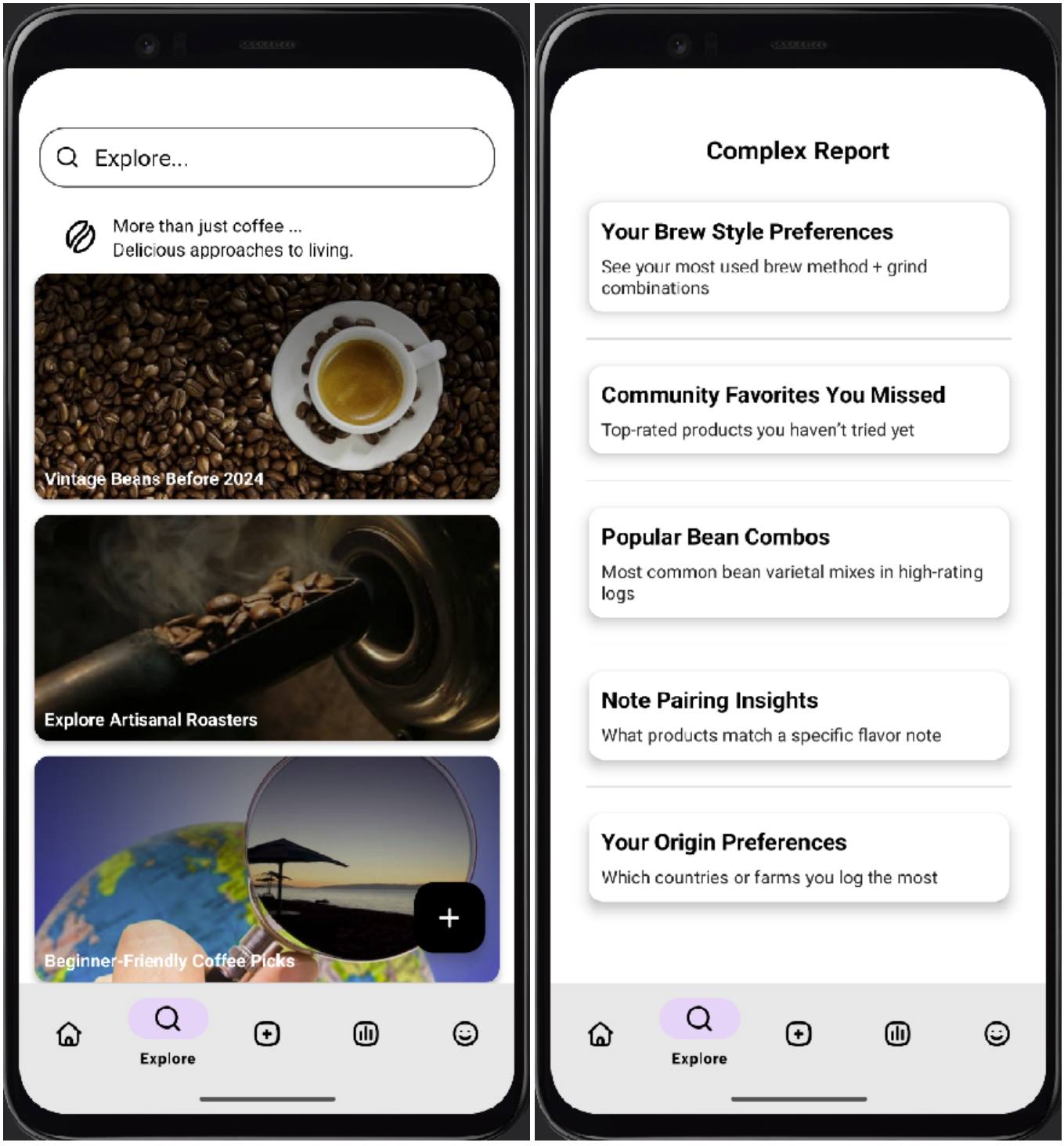
```
POST /api/complex-report/search
```

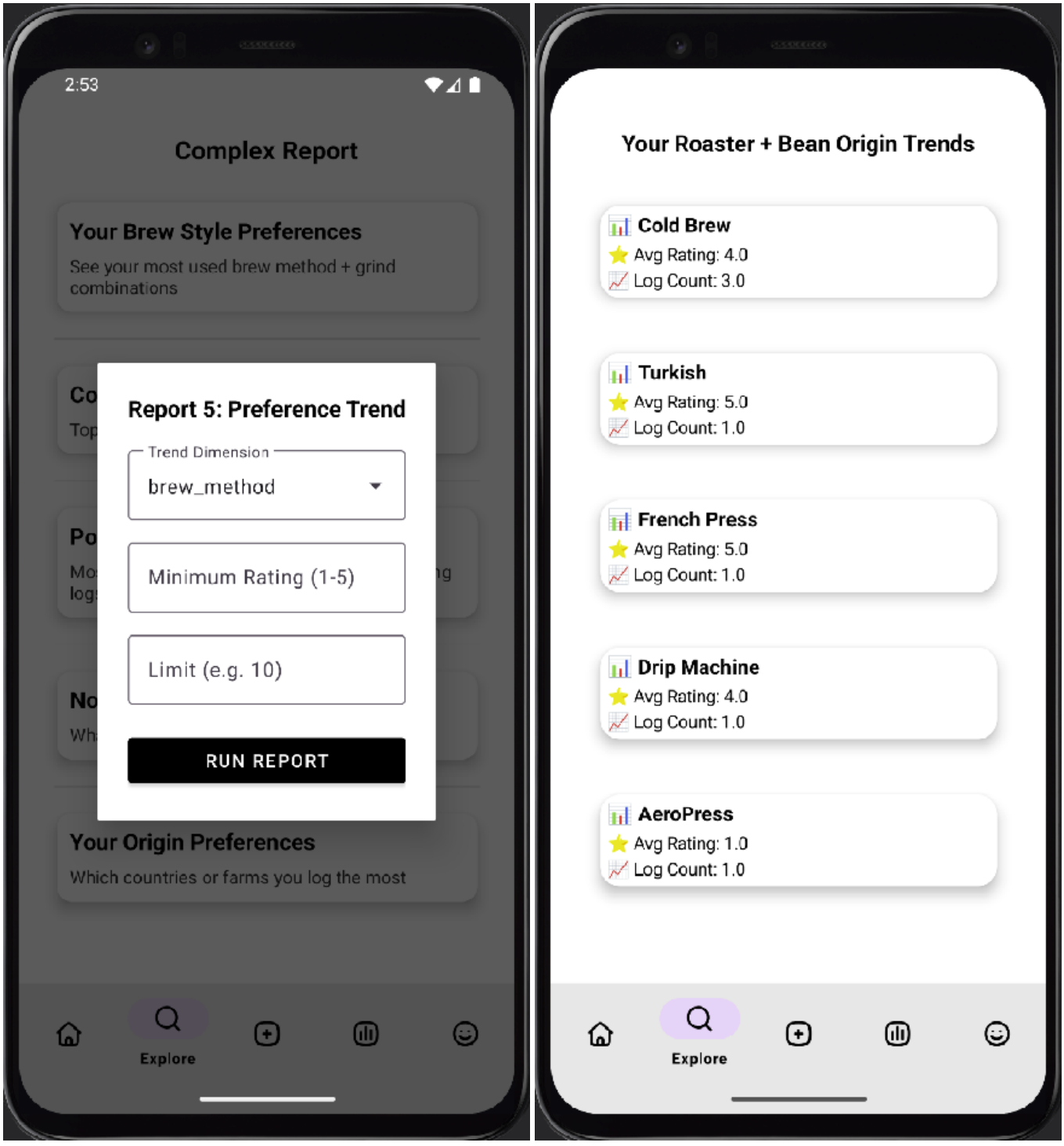
```
{
  "entity": "origin_patterns",
  "query": {
    "user_id": "2",
    "dimension": "roast_level",
    "limit": "10"
  }
}
```

Screenshot:

```
    @Query(value = """
SELECT
    bm.name AS category,
    ROUND(AVG(rt.rating_id), 2) AS avgRating,
    COUNT(*) AS logCount
FROM TastingLogs t
JOIN BrewMethod bm ON t.brew_method_id = bm.brew_method_id
JOIN Rating rt ON t.rating_id = rt.rating_id
WHERE t.user_id = :userId
GROUP BY bm.name
ORDER BY logCount DESC, avgRating DESC
LIMIT :limit
""", nativeQuery = true)
    List<UserPreferenceSummaryProjection> findUserBrewMethodSummary(
            @Param("userId") int userId,
            @Param("limit") int limit
    );

    @Query(value = """
SELECT    E, Today • Updated Complext Report
    gs.name AS category,
    ROUND(AVG(rt.rating_id), 2) AS avgRating,
    COUNT(*) AS logCount
FROM TastingLogs t
JOIN GrindSize gs ON t.grind_size_id = gs.grind_size_id
JOIN Rating rt ON t.rating_id = rt.rating_id
WHERE t.user_id = :userId
GROUP BY gs.name
ORDER BY logCount DESC, avgRating DESC
LIMIT :limit
""", nativeQuery = true)
    List<UserPreferenceSummaryProjection> findUserGrindSizeSummary(
            @Param("userId") int userId,
            @Param("limit") int limit
    );

    @Query(value = """
SELECT
    rl.name AS category,
    ROUND(AVG(rt.rating_id), 2) AS avgRating,
    COUNT(*) AS logCount
FROM TastingLogs t
JOIN RoastBatch rb ON t.batch_id = rb.batch_id
JOIN Products p ON rb.product_id = p.product_id
JOIN RoastLevel rl ON p.roast_level_id = rl.roast_level_id
JOIN Rating rt ON t.rating_id = rt.rating_id
WHERE t.user_id = :userId
GROUP BY rl.name
ORDER BY logCount DESC, avgRating DESC
LIMIT :limit
""", nativeQuery = true)
    List<UserPreferenceSummaryProjection> findUserRoastLevelSummary(
            @Param("userId") int userId,
            @Param("limit") int limit
    );
}
```




Explore...
More than just coffee ...
Delicious approaches to living.
Vintage Beans Before 2024
Explore Artisanal Roasters
Beginner-Friendly Coffee Picks
Explore
Complex Report
Your Brew Style Preferences
See your most used brew method + grind combinations
Community Favorites You Missed
Top-rated products you haven't tried yet
Popular Bean Combos
Most common bean varietal mixes in high-rating logs
Note Pairing Insights
What products match a specific flavor note
Your Origin Preferences
Which countries or farms you log the most
Explore



This report helps users understand their historical preferences across different bean origins or processing characteristics. It supports more informed future tasting decisions.

7 Project Retrospective

What I liked most about the project was the opportunity to design a full-stack application with a complex relational schema and real-world functionality. I especially enjoyed building the entity relationships from scratch and seeing them come to life through the personalized recommendation features in the frontend.

The most frustrating part was debugging front-end state synchronization between fragments and ViewModel, especially when managing complex input flows like the tasting log. There were also challenges with ensuring consistent data propagation when editing versus adding entities.

The easiest part of the project was implementing the core entity structure and setting up the backend API. Thanks to careful schema planning and normalization, the backend logic followed a predictable and modular structure.

The hardest part was coordinating the frontend-backend interaction in the Android app. Managing API calls, state updates, and UI feedback across asynchronous flows required careful debugging and incremental testing. Also, making the recommendation queries both performant and meaningful added significant complexity.

Overall, I learned how to design and implement a full-featured database-driven application from schema to UI. I also gained experience balancing normalization with usability, creating parameterized SQL queries for reports, and structuring a mobile app around complex data models.

8 Conclusion

This project produced a fully functional full-stack coffee tasting log application, including a normalized relational database, RESTful backend services, and an interactive Android frontend. The system supports adding and managing roasters, products, and beans; recording detailed tasting logs; tracking user preferences; and exploring personalized recommendations. All required queries and five complex report queries were successfully implemented, validated, and integrated into the user interface.

Some parts of the system could still be improved or extended. The statistics dashboard currently uses mock data and could be connected to real-time queries. Additional usability enhancements, such as better input validation, improved error handling, and offline support, could further refine the user experience. Finally, security measures like password hashing and authentication tokens could be added to support future user scaling.