Institute of Communication
and Computer Systems (ICCS)
of the National Technical
University of Athens (NTUA)

Choice

Shrink Your Food Mobile App



Nikos Tantaroudas – ICCS

8th July 2026


Shrink Your Food Waste APP
Smart choices.
Less waste.
Better planet.
GET IT ON Google Play
Daily Plan
February 2026
% Day Budget
Food
Carrots
0.2 kg CO_2
Potatoes
0.2 kg CO_2
Iceberg Lettuce
0.1 kg CO_2
Sustainability Assistant
Hello! I'm your Smart Food Sustainability Assistant!
I can help you:
• Analyze your eating patterns
• Get personalized suggestions
• Find low CO2 alternatives
• Track your sustainability score
What would you like to know?
Funded by the European Union
Ask me anything...
Track your CO_2 footprint
Get smarter alternatives
Discover engaging data stories
Follow your weekly progress

Problem Statement

Food systems drive 21–37% of global greenhouse gas emissions (IPCC) – and household choices are a major lever.

- Consumers face an intention–action gap: climate concern rarely translates into everyday food decisions.
- CO_2 data per product exists, but it is abstract – it must become tangible, personal and actionable.
- Food waste compounds the problem: emissions are generated for food that is never eaten.
- CHOICE Task 4.2: build a consumer-facing digital tool that links food choices to climate impact.

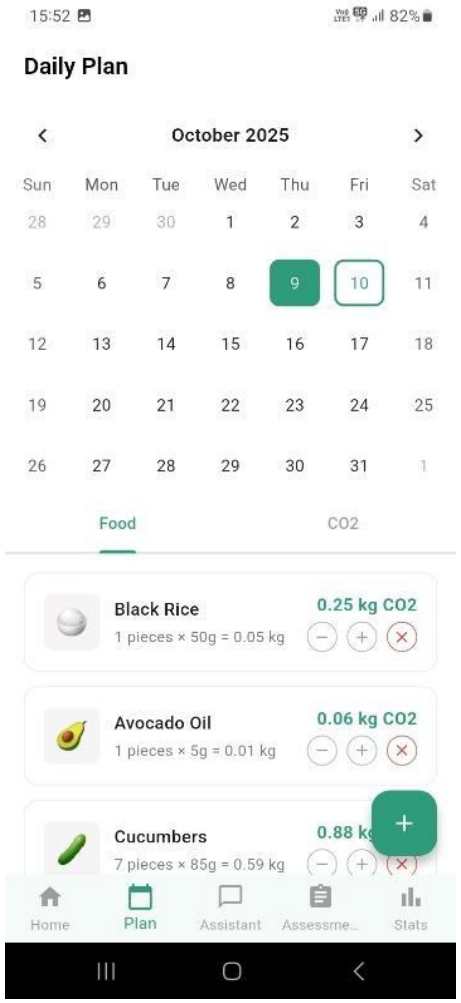


Proposed Solution

‘Shrink Your Food Waste’ – a mobile app that links personal food choices to climate impact.

- Flutter cross-platform app + Django REST backend, communicating over HTTPS with token authentication.
- Dual CO_2 data sources: inoqo product-level LCA + the Big Climate Database (530+ food items).
- Daily meal planning, pantry and weekly CO_2 tracking with charts.
- Integrated educational data stories (provided by LIBRA), evaluated with a built-in A/B testing framework.
- A rule-based sustainability chatbot for personalised guidance.





Implementation Pillars

- Backend & REST API
- Mobile App Experience
- Chatbot & A/B Testing

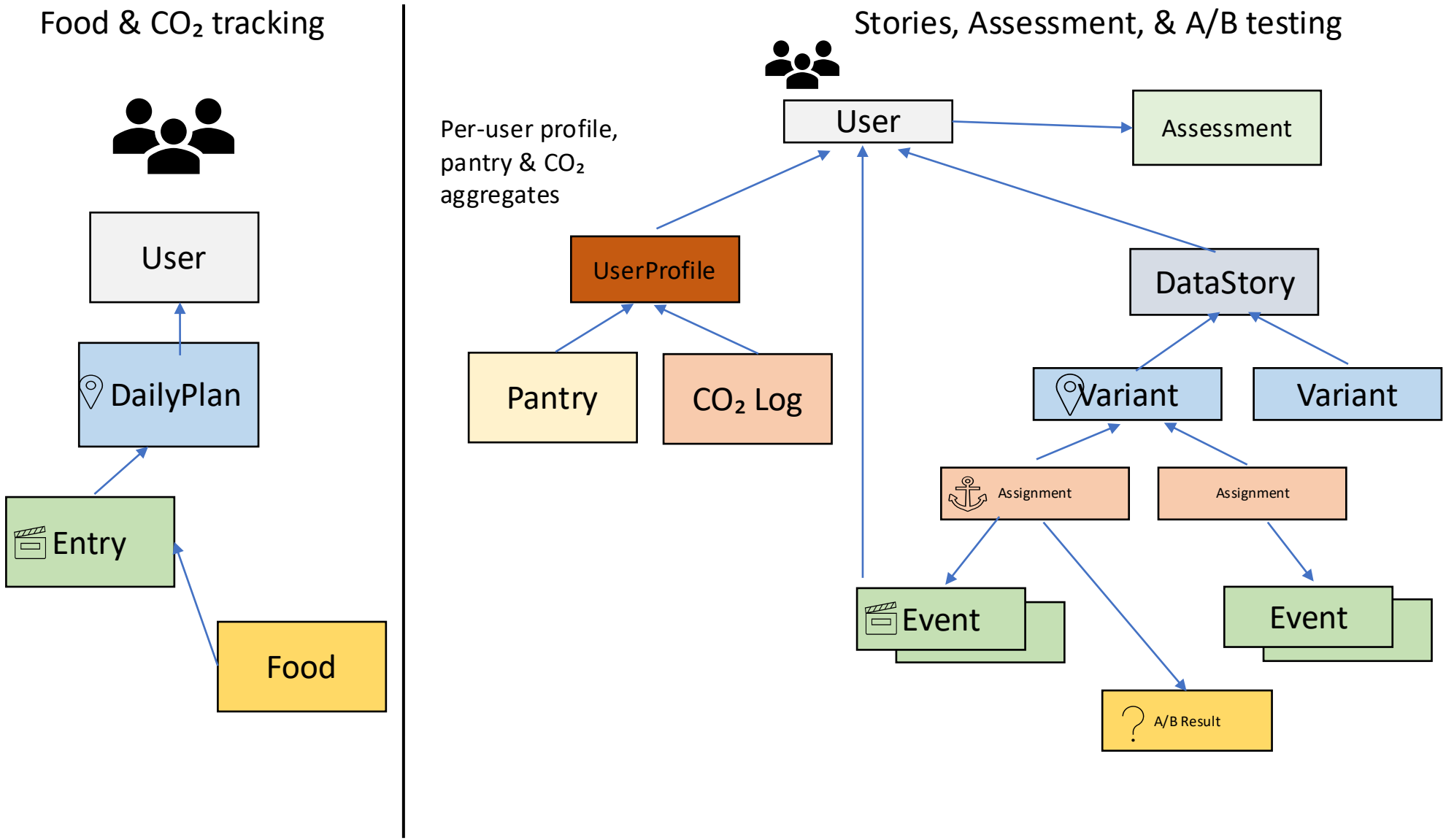
How do we model food, meal plans and CO_2 data, and serve them securely?

How does the mobile app turn climate data into everyday food decisions?

How do we personalise guidance and measure which messaging works?

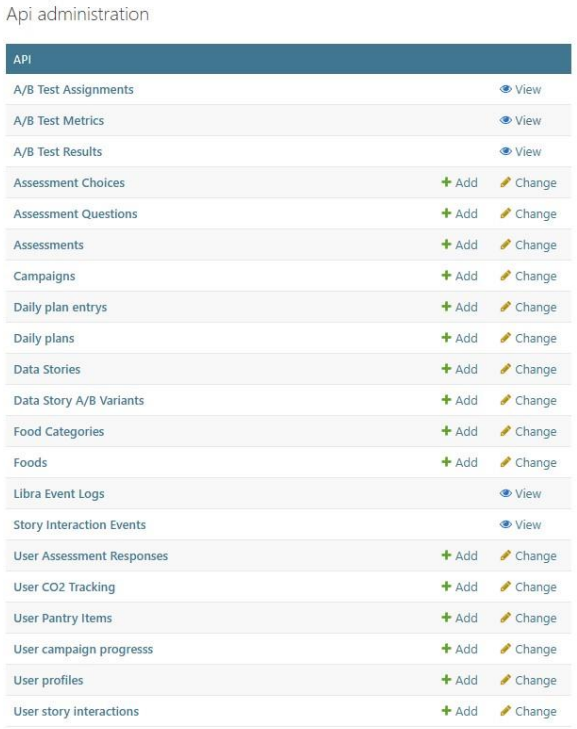


Backend Data Model

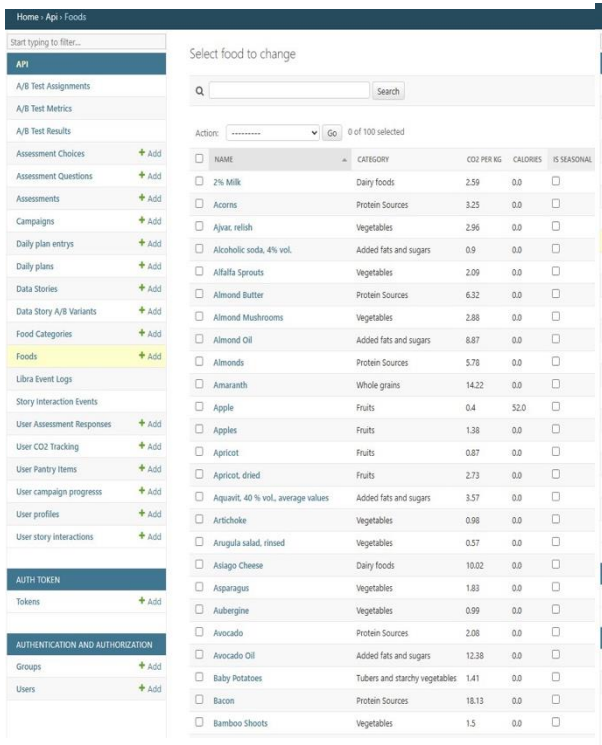


Backend Data Model

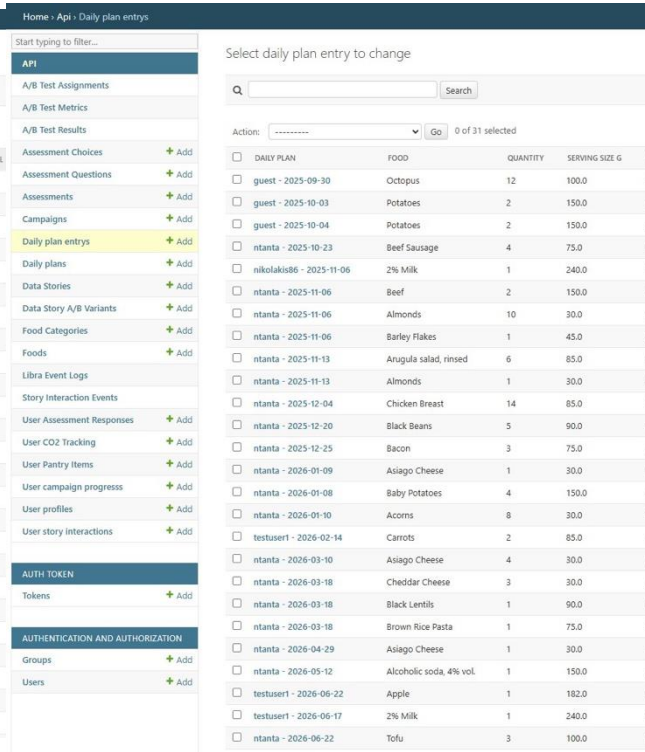
Main API



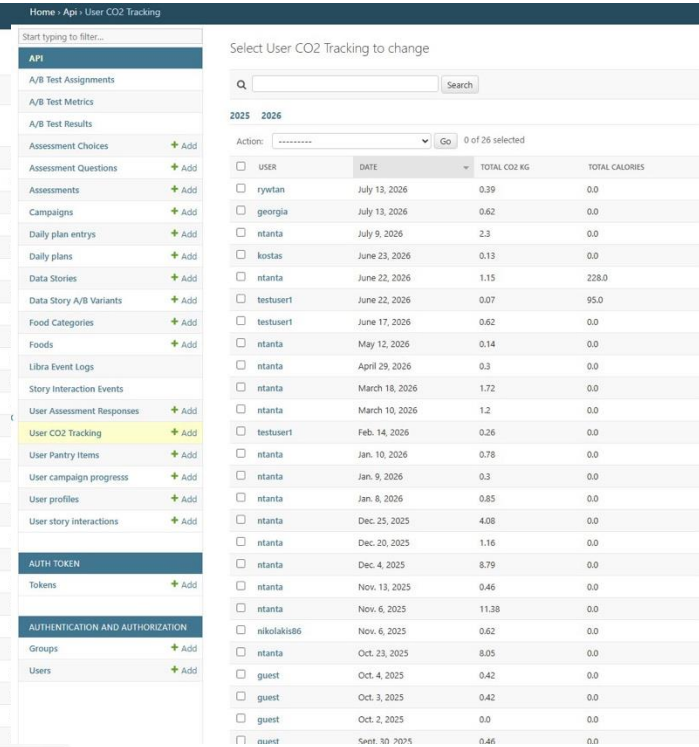
Food Items



Daily plan entry



User CO2 Tracking



Backend Software & Features

Backend

- Django 5.2 + Django REST Framework providing a RESTful API for every app feature
- SQLite database – 19 relational models (foods, plans, stories, A/B testing, etc.)
- DRF token authentication, per-endpoint permissions
- Backend served by on premises server

Mobile client

- Flutter app – all HTTP traffic through a single ApiService gateway
- Django admin as the authoring surface for foods, stories, campaigns & quizzes

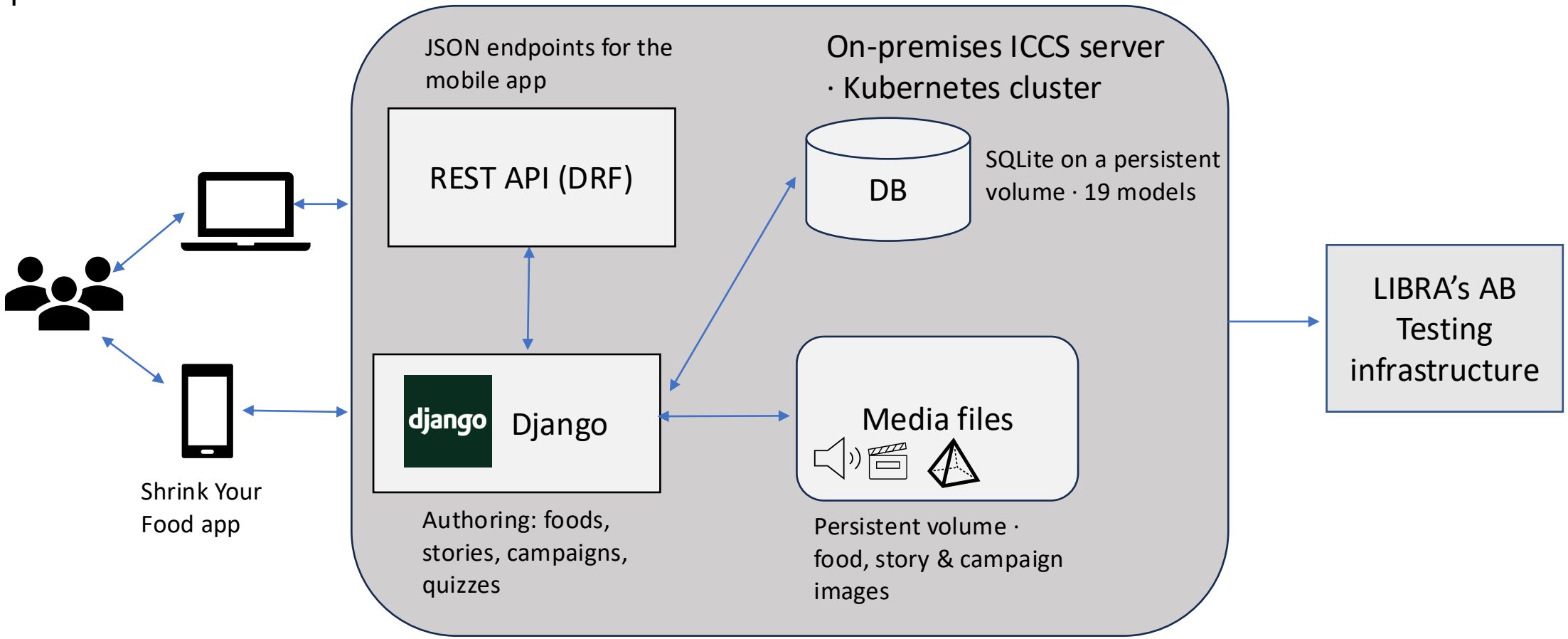
Main Features

- Full auth flows: register, login, password-reset e-mails, account deletion
- CRUD APIs for foods, categories, daily plans & pantry
- CO_2 tracking: per-food emission factors, daily & weekly aggregation
- Data stories including sustainability related messages
- Assessments: multiple-choice quizzes with scoring
- Rule-based chatbot endpoints (chat, history, status)
- A/B testing endpoints + Libra platform logging
- Admin statistics API powering in-app analytics
- Management commands: populate_data, merge_stories_to_variants, analyze_ab_tests



System Architecture

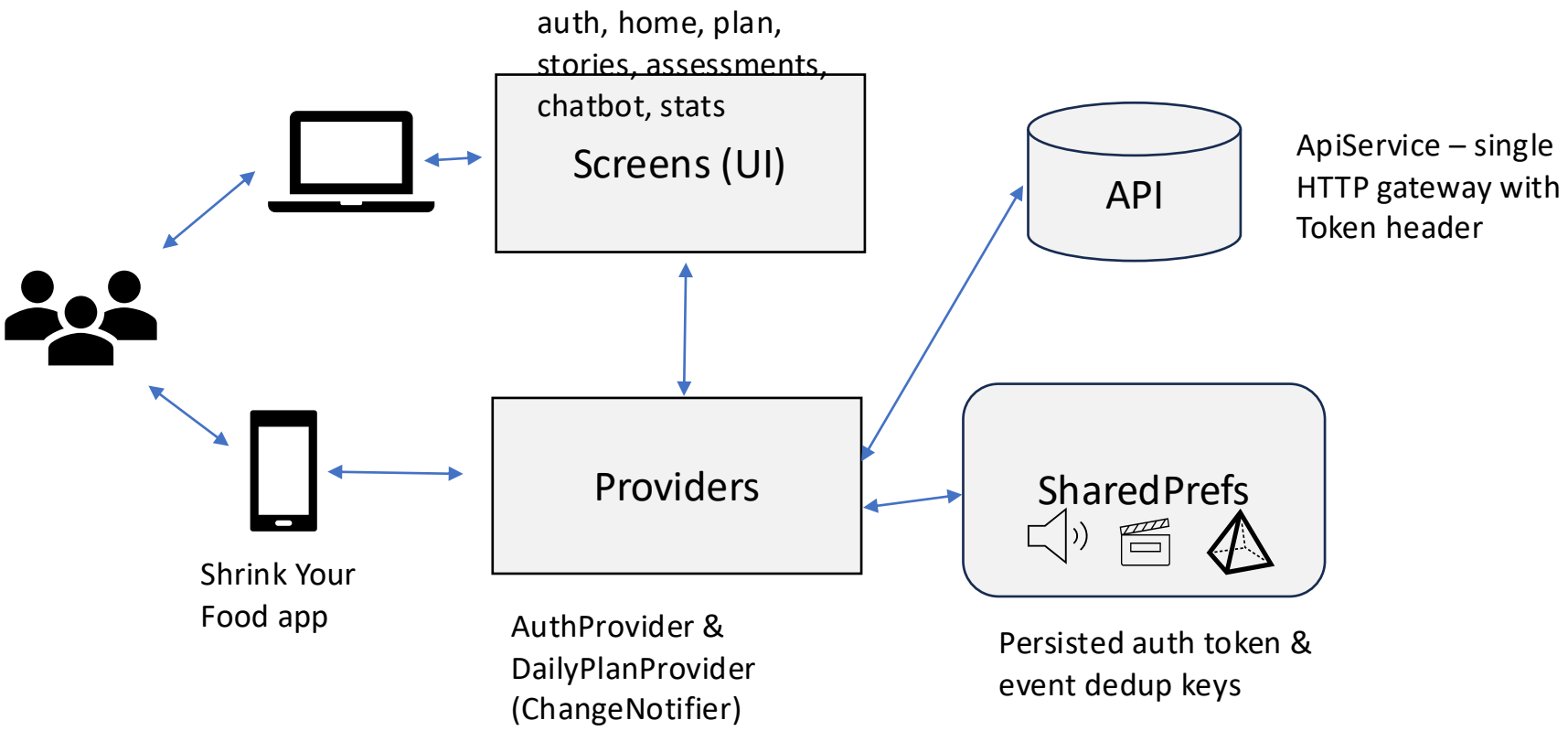
Client–server over HTTPS – a stateless, token-authenticated API



Containerised Django app orchestrated by Kubernetes — DB & media on persistent volumes

Mobile App Architecture (Flutter)

Layered structure: models → services → providers → screens

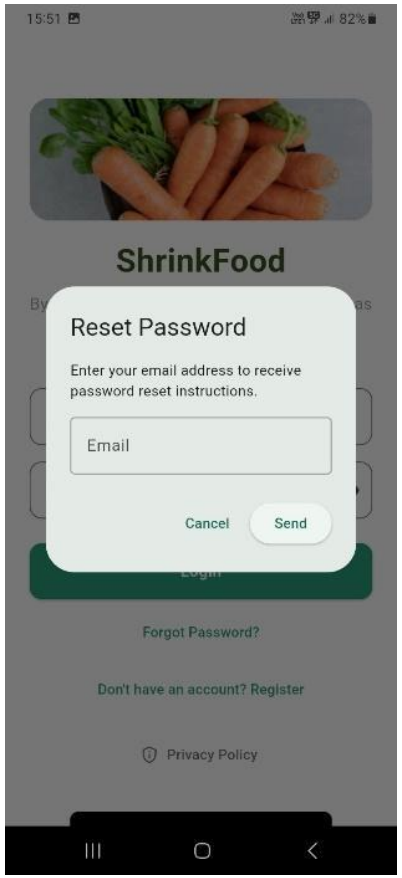


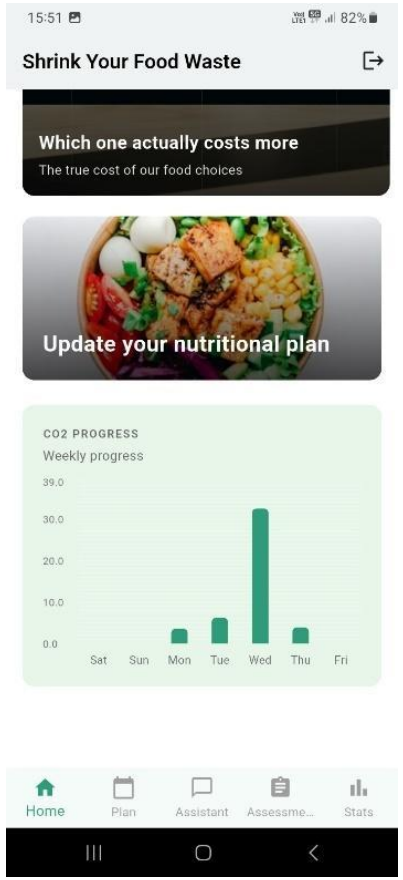
- Models: classes that mirror the API's JSON; define data shape
- Services: Gateway that makes http calls to backend and convert the responses into the Models classes above
- Providers: state management that hold the data in memory and notify the UI whenever it changes
- Screens: The UI widgets the user sees; they read the providers and rebuild automatically when the state updates

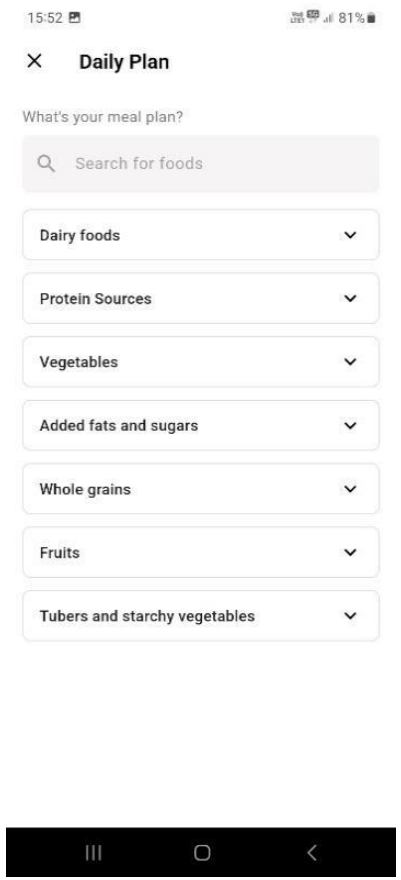
Implementation Pillars

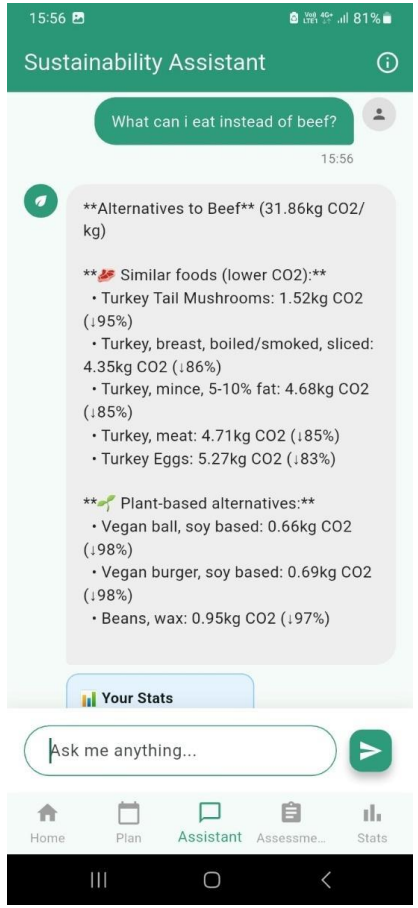
- Backend & REST API
- **Mobile App Experience**
- Chatbot & A/B Testing

A Flutter app that turns CO_2 data into everyday food decisions









App Walkthrough: Onboarding & Home

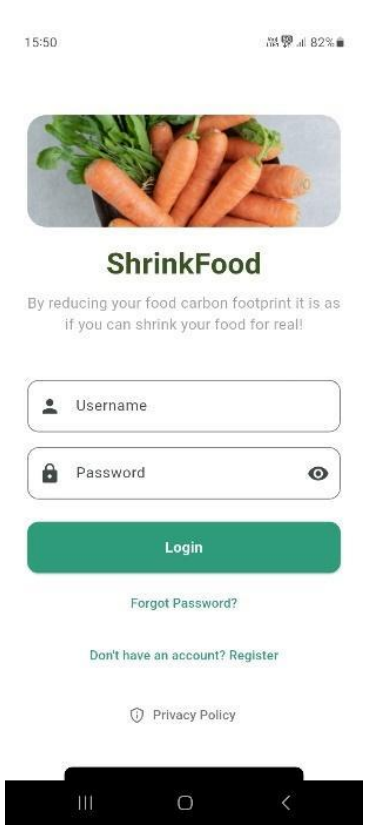
Authentication & Onboarding

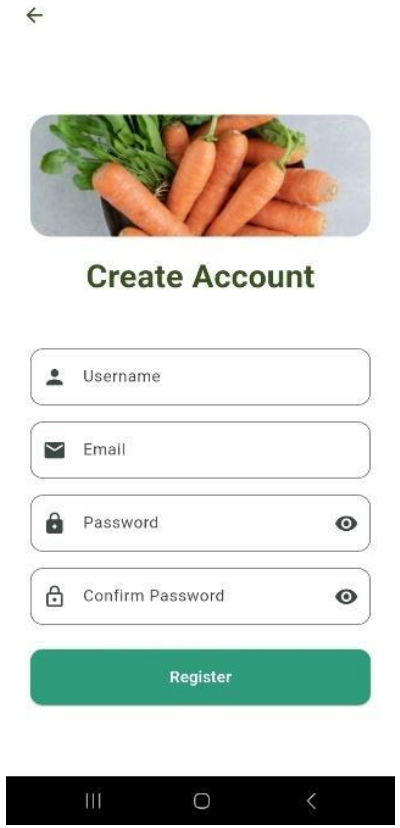
Register with username, e-mail & password – privacy policy up front; passwords PBKDF2-hashed, HTTPS only.

Login persists a DRF token; ‘Forgot password’ e-mails a reset link that deep-links back into the app.

Home – the sustainability dashboard

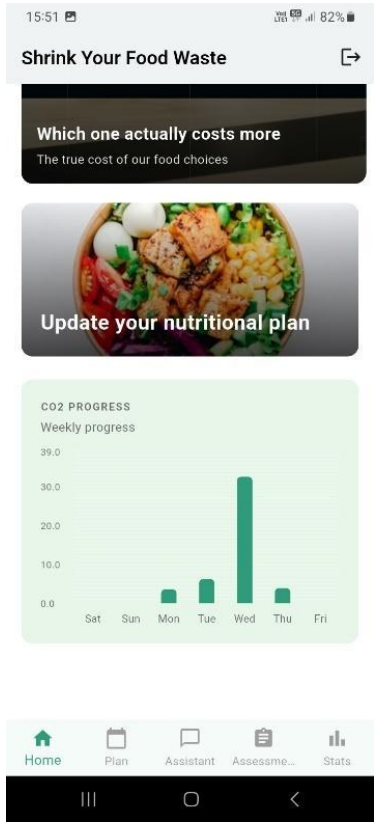
- Personal greeting and motivational tagline
- Top Stories carousel (LIBRA data stories)
- Nutritional-plan shortcut + weekly CO_2 progress chart
- Bottom navigation: Home, Plan, Assistant, Assessments, Stats





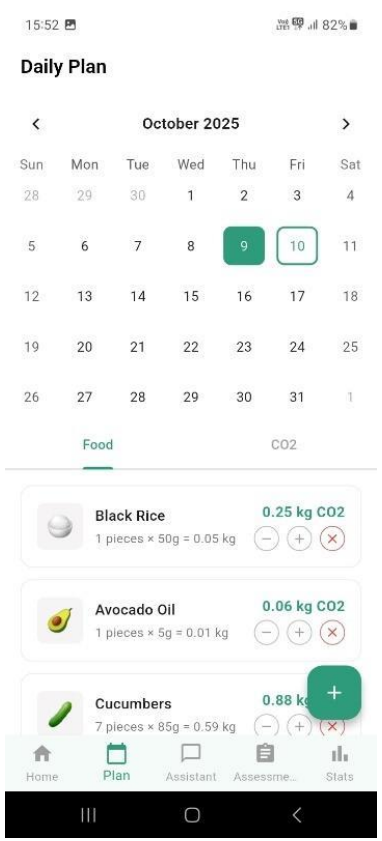


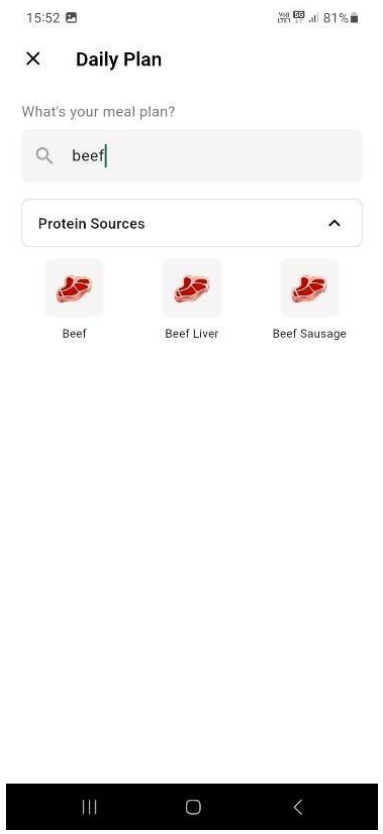


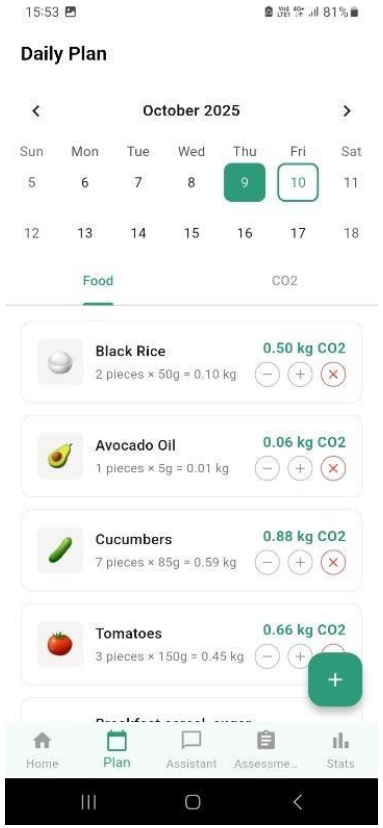


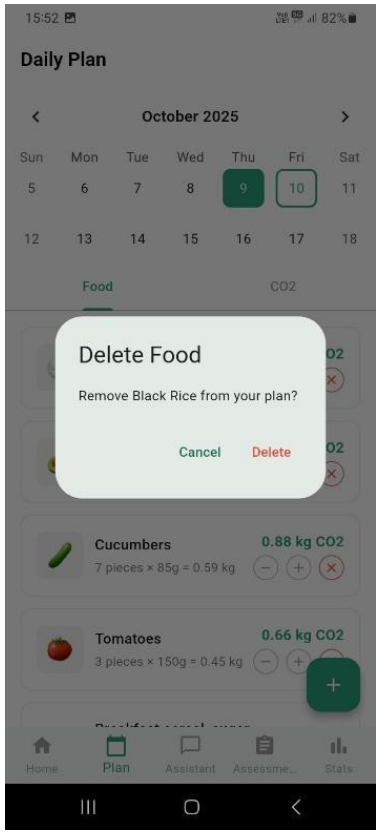
Daily Plan & CO_2 Tracking

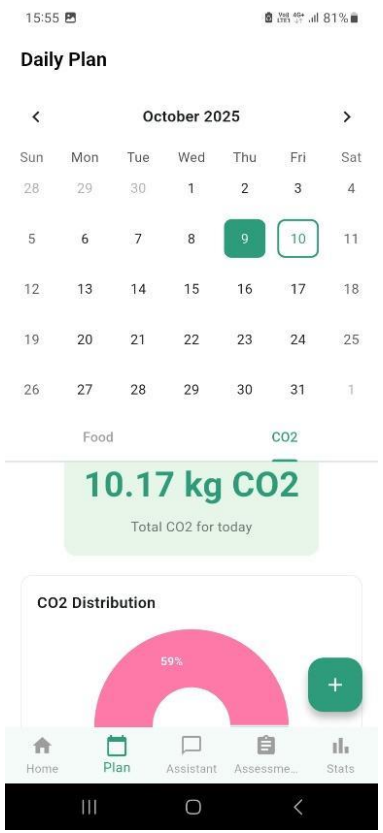
- Calendar-based daily plan: search 540 foods, set portions, log meals by type
- Per-item CO_2 from inoqo product LCA data + the Big Climate Database
- Daily totals, per-food CO_2 distribution and pie-chart visualisation

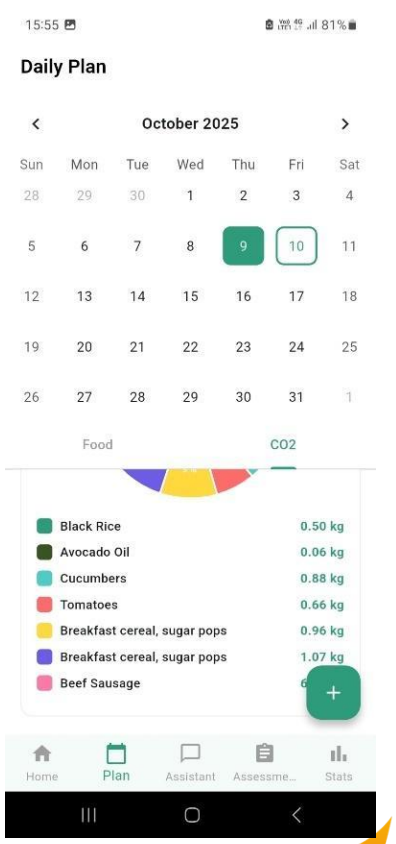












Interactive Data Stories





Short, visual stories explain the climate impact of food choices.

Carousel on Home with a dedicated detail view with rich imagery.

Topics: dairy vs plant milk, the true cost of food, food waste etc.

Engagement tracked per story: views, time spent, scroll depth, completion.

‘Interested’ / ‘Not interested’ feedback on every story.

Each story can run two A/B variants differing in narrative, visual style or nudge.

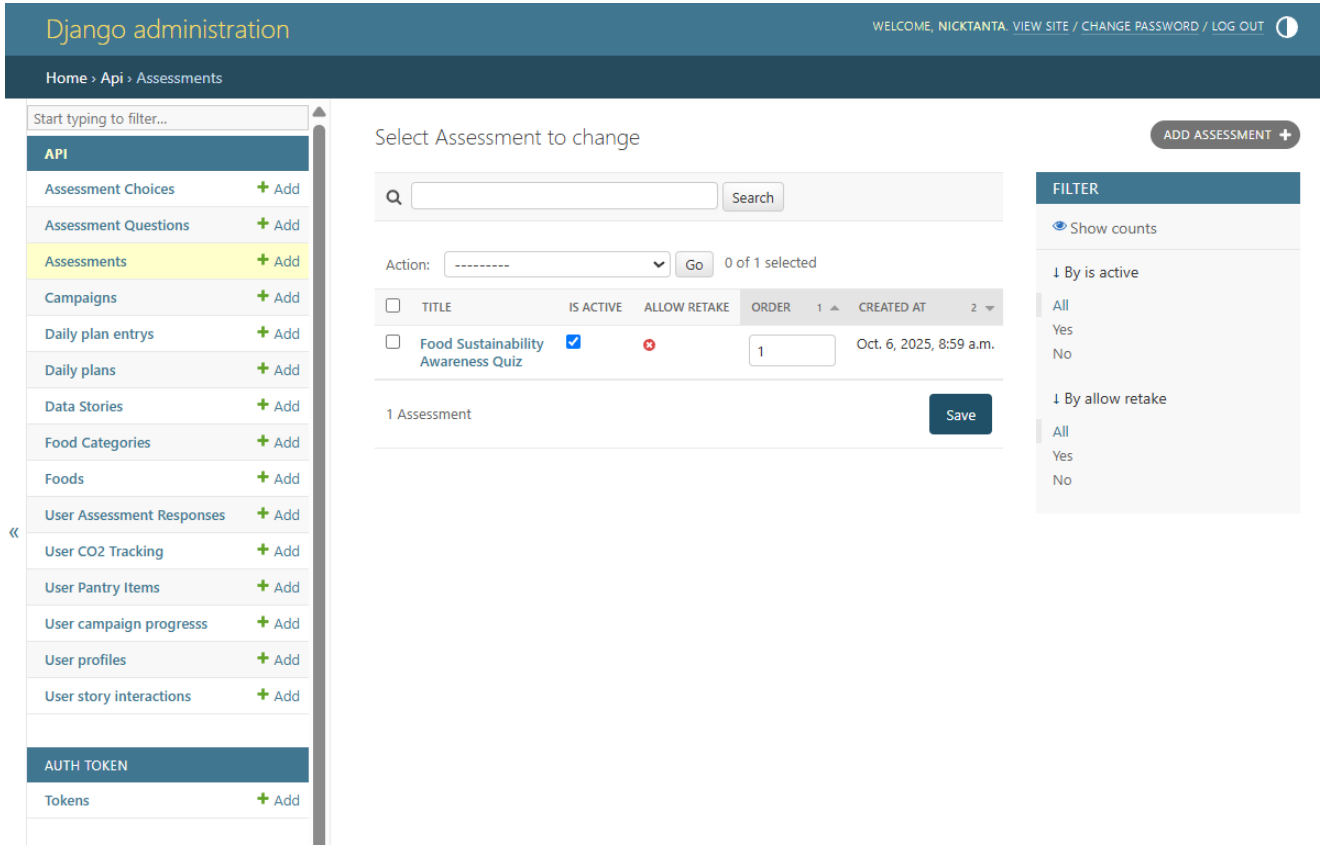
Variant assignment is per user and consistent across sessions.

All interactions feed the A/B framework – locally and on Libra.

Assessments & Knowledge Quizzes

How do we measure sustainability awareness?

Quizzes are authored in the Django admin and taken directly in the app.





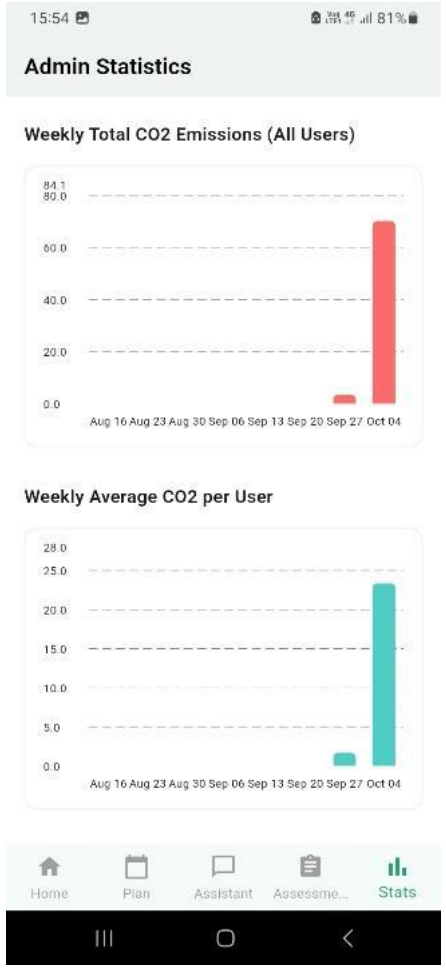
Quiz authored in Django admin → User takes the quiz in the app → Answers submitted via API → Scored & stored

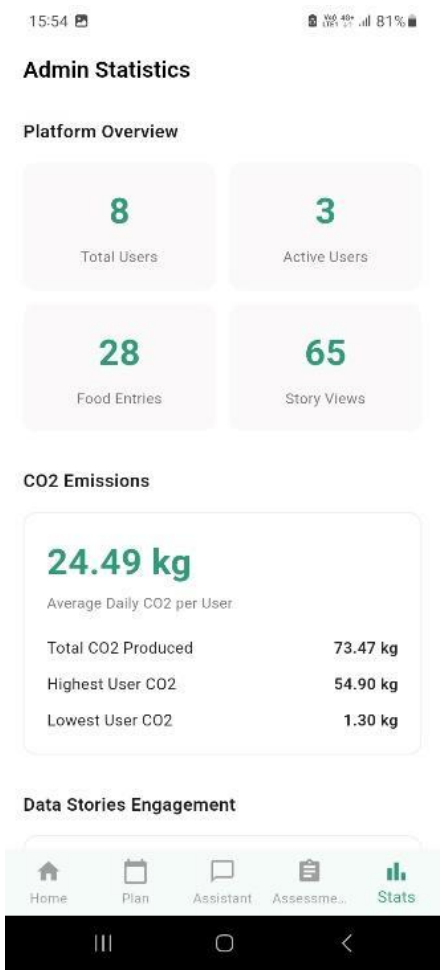
In-App Admin Analytics

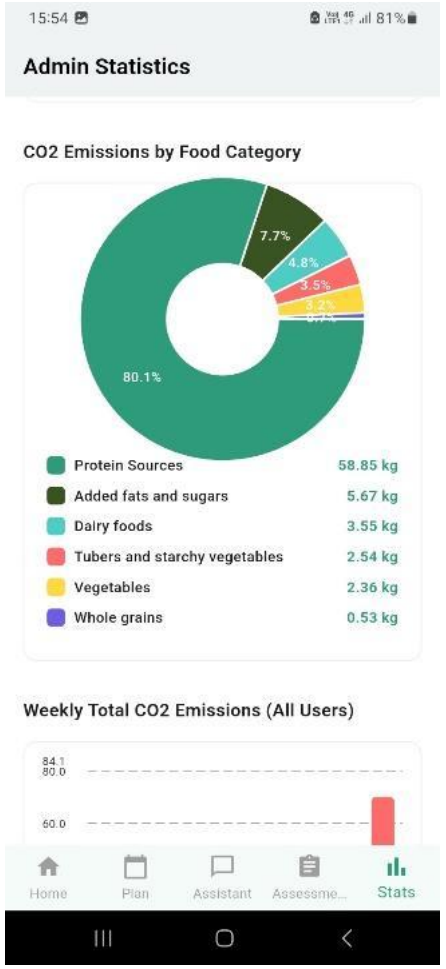
Admins get a Stats tab inside the mobile app – no Django admin needed.

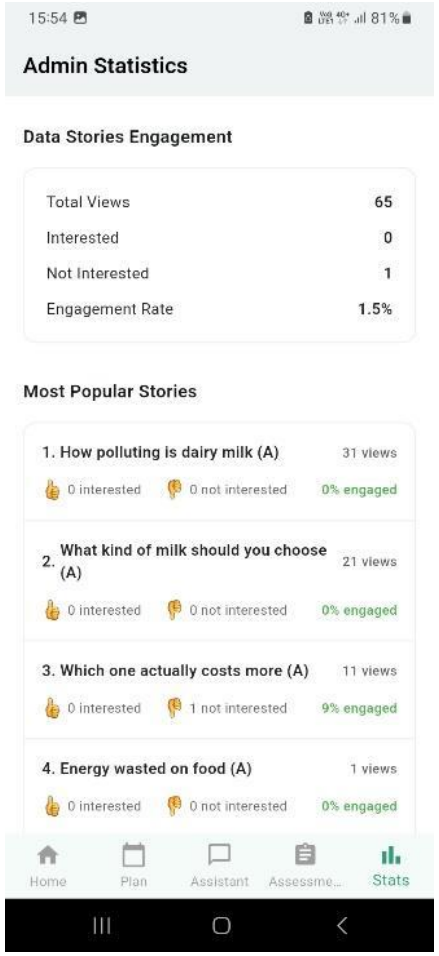
- Platform overview: total & active users, food entries, story views.
- CO_2 panel: average daily CO_2 per user, totals, highest & lowest values.
- Data-story engagement: views, engagement rates, most popular stories.
- Charts: CO_2 by food category, weekly total & average emissions.

Role-gated: the tab is rendered only for staff accounts (isStaff).









Implementation Pillars

- Backend & REST API
- Mobile App Experience
- **Chatbot & A/B Testing**

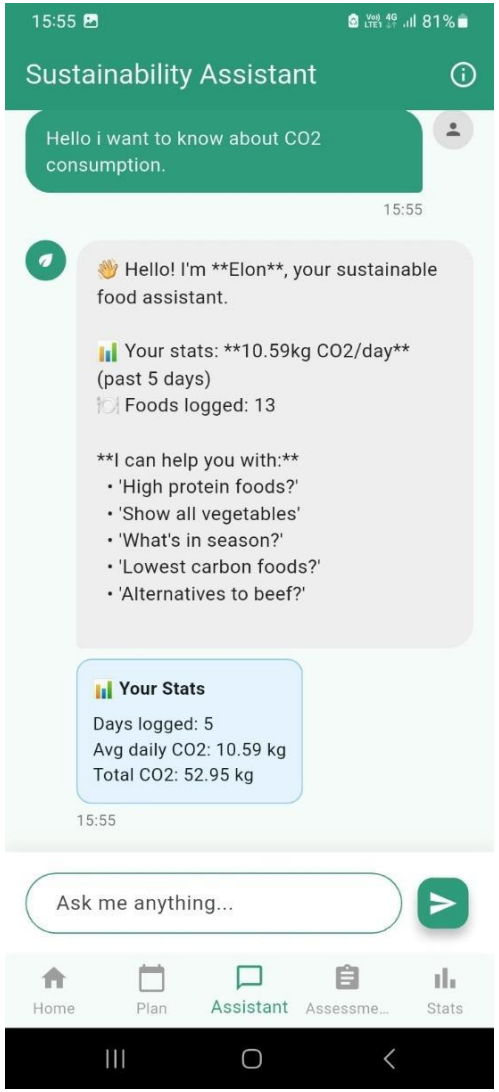
Built-in intelligence: a rule-based chatbot for personalised guidance, and A/B-tested data stories to learn which climate messaging works

Rule-Based Sustainability Chatbot

'Elon' is a pattern-based sustainability assistant that runs entirely on the Django backend with no external ML support, providing deterministic answers and thus instant responses.

Python rule engine · 1000+ question patterns · 540 foods cached in memory (singleton, loaded once)

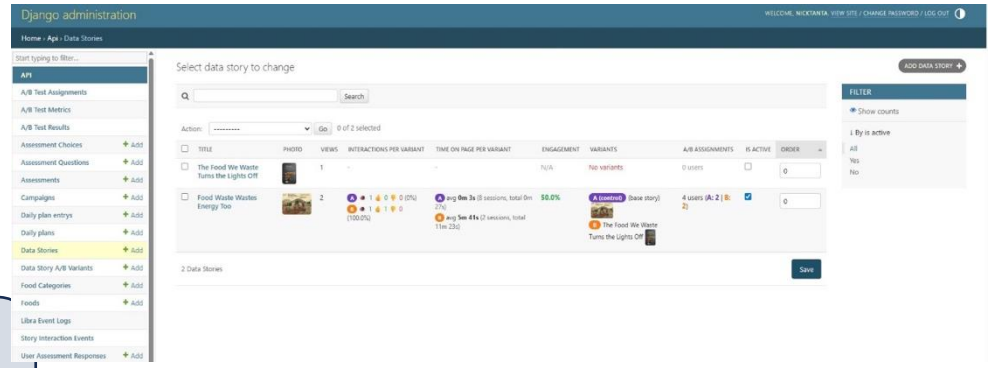
Decision Process (priority order):
1) Load the user's 7-day history → avg daily CO_2 & favourite foods.
2) Check 'X vs Y' comparisons → CO_2 difference in %.
3) Match intent patterns: nutrition, seasonal, recipes, CO_2.
4) Find foods via substring & word matching.
5) Suggest lower-CO_2 alternatives & plant-based swaps.
6) Personal stats, greetings, friendly fallback.
7) Reply instantly from in-memory data.



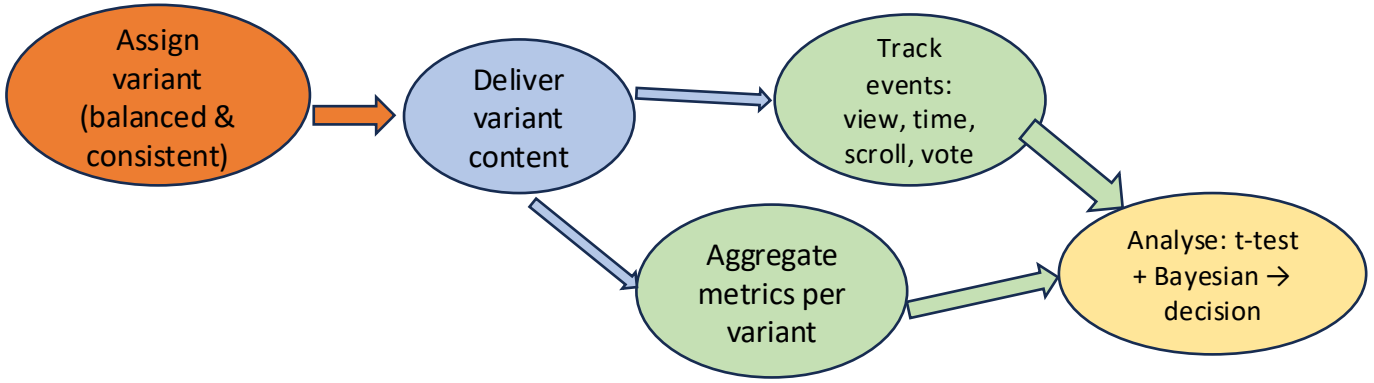
- On startup it loads all **540 foods into memory once** (a singleton), so no database query happens during a conversation → fast replies.
- It matches the user's message against **1000+ predefined question patterns** using simple text matching (substring + word matching), no vectors, no embeddings, no ML inference.
- It **personalises** by reading the user's **last 7 days** of logged meals to compute their average daily CO_2 and most-eaten foods, then weaves those into replies.
- A **priority decision tree** decides how to answer (comparisons first, then intent patterns, then food lookup, then alternatives, then personal stats, then greetings, then a friendly fallback) — so users always get a relevant response.
- It has a **personality** (introduces itself as "Elon", recognises the user's name, varies its off-topic replies) to keep it engaging.

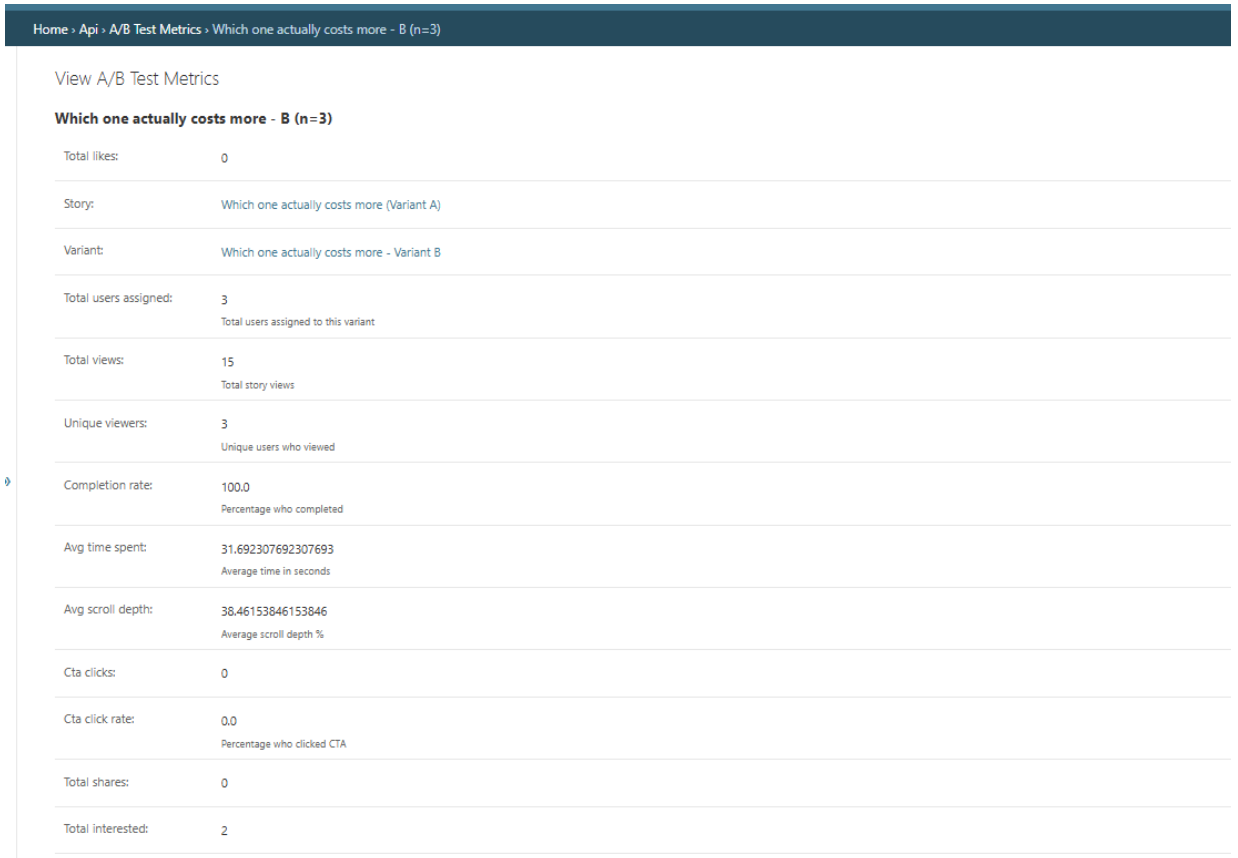
A/B Testing of Data Stories

Every data story can run as a randomised controlled experiment: variant A (control) vs variant B (treatment), differing in narrative approach, visual style or nudge type.



Input: one data story with two variants (A: control, B: treatment)





Django admin: A/B metrics & test results

- 1. measure rate per variant= “interested votes/views
- 2 compare through frequentist t-test-> p value to check if the different is statistically significant ($p<0.05$) ; Bayesian: Beta curves to check probability to be best + expected loss
- 3. Decide only if $p< 0.05$ and probability to be best >95% (minimum 30 users per variant)

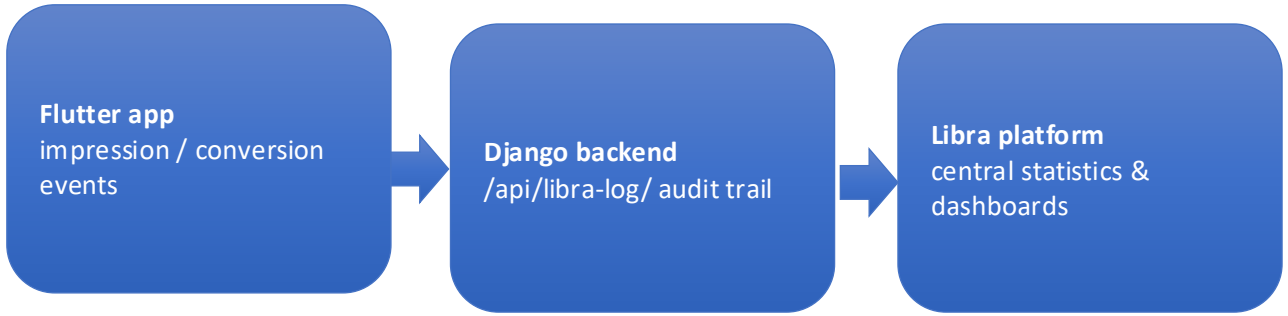
Libra A/B Platform Integration

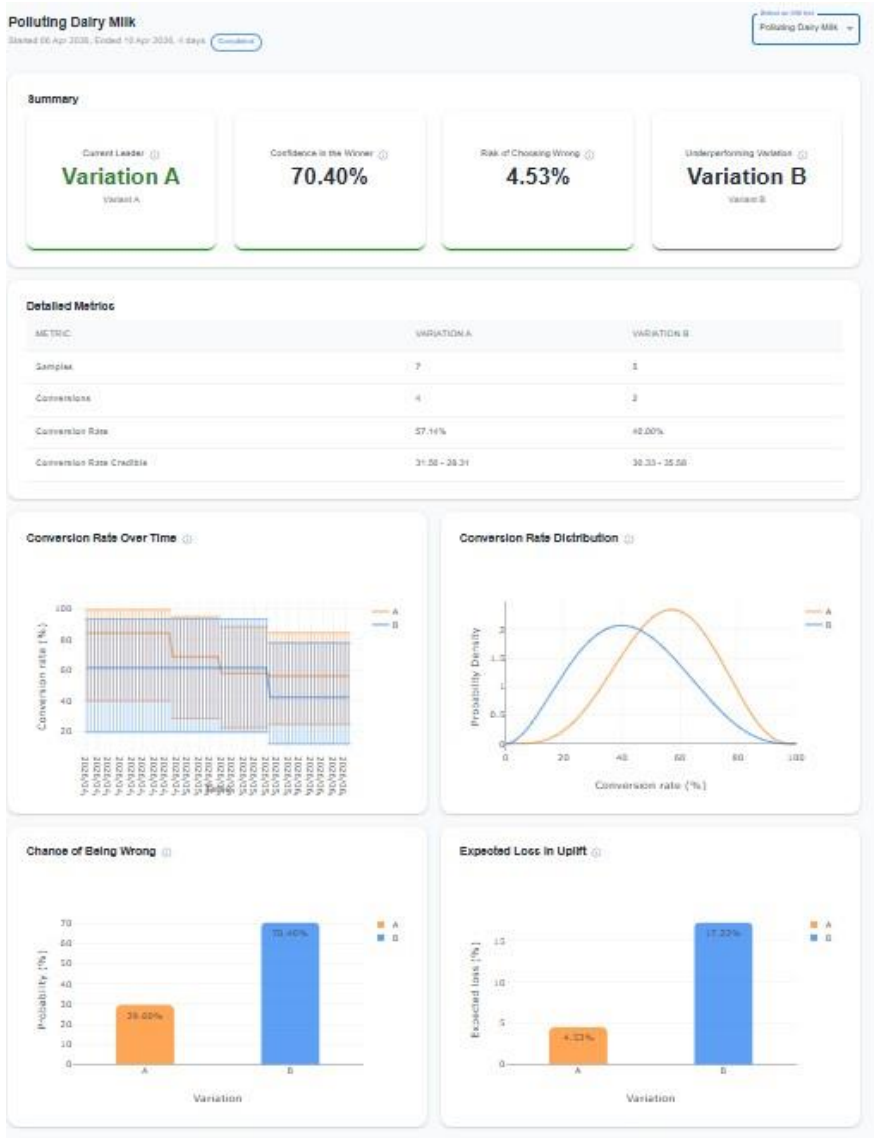
Motivation:
Need to evaluates all tools centrally – so every experiment is also tracked on LIBRA's A/B platform (abtesting.choice.libraservices.ai).

Dual tracking: full event detail stays in Django; impressions & conversions are pushed to Libra. Conversions fire only on 'Interested' votes.

Each story carries its own Libra test ID and variant A/B IDs; the app sends events fire-and-forget, deduplicated per user & story.

Admin workflow: merge_stories_to_variants → setup_libra_variants → analyze_ab_tests







Institute of Communication and Computer Systems (ICCS) of the National Technical University of Athens (NTUA)

Choice

Thank you