
Swastik Ranjan Panigrahi

Email address: textswastik.alt@gmail.com

Open Food Facts ID: swastikCommits

Slack ID: Swastik Ranjan Panigrahi

GitHub ID: [swastikCommits](#)

Time Zone: IST(India) UTC +5:30

Open Food Facts Explorer - a new generation frontend

DESCRIPTION

The **Open Food Facts Explorer** represents a transformative approach to accessing the world's largest open food database. As a modern **SvelteKit frontend**, this project aims to revolutionize how people interact with food information by delivering a **fast, accessible, and mobile-optimized** interface—tailored for the **80% of users** who access the platform via mobile devices.

This implementation will integrate key components from across the **Open Food Facts ecosystem**, including the **NodeJS SDK**, **Knowledge Panel facets**, **Search-A-Licious**, **Robotoff Questions**, **Price editing support**, and **product contribution features**—into a cohesive, seamless experience. The **cross-database integration** is especially impactful, enabling fluid navigation between **food**, **cosmetics**, **pet food**, and other product types through a **unified interface**. By breaking down information silos, the Explorer becomes a **comprehensive, community-powered resource** for consumers and contributors alike.

Mentor(s)

- [VaiTon](#)

What city and country will you reside in during the summer?

I will be residing in Bhubaneswar, India, during the summer.

What benefits does your proposed work have for Open Food Facts and its community?

The proposed work delivers substantial benefits to Open Food Facts by creating a **high-performance, mobile-optimized platform** that serves the 80% of users accessing via mobile devices through **faster load times** and **touch-friendly interfaces**. It integrates diverse ecosystem components (NodeJS SDK, Knowledge Panels, Seach-A-Licious, Folksonomy Engine, Robotoff AI) into a cohesive experience while providing **unified search** across food products, cosmetics, and pet food databases. For contributors, streamlined workflows will lower participation barriers, helping convert passive users into active community members. Technically, the **clean API architecture, TypeScript definitions, and reusable web components** will **reduce future maintenance costs** and provide a solid foundation for ongoing development, ensuring the platform remains accessible, **maintainable**, and technologically relevant.

Why are you the right person to work on this project?

I am the right fit for this project due to my strong background in **full-stack development, DevOps, open-source contributions, and performance optimisation**. I have worked extensively with **JavaScript, TypeScript, Node.js, Python, REST API, React, Svelte, and SvelteKit**, which align perfectly with the project's tech stack.

I have been the **#1 contributor (excluding maintainers) since March 2025**, with **23+ merged PRs, 5 open PRs, with over 2200+ lines of code changes** and **over 30+ issues fixed**. My contributions have played a key role in **the implementation of new functionalities, improving the UI, project stability, CI workflows, API performance, and user experience**.

The contribution statistics shown below are based on March 2025.




















[Reference to my contribution statistics on openfoodfacts-explorer](#)

As someone passionate about open knowledge and health transparency, working on Open Food Facts Explorer aligns closely with my values. This isn't just code for me — it's a chance to build tools that improve how people relate to food globally.

Please include links to your code contributions which have already been merged, or to GitHub merge requests for the issues you fixed for the project of your proposal or any other Open Food Facts projects. This demonstrates your willingness to learn and familiarity with development workflow.

Contributions to **openfoodfacts-explorer**, **openfoodfacts-nodejs**, and **openfoodfacts-webcomponents** are listed below:

Description	Fixes Issue(s)
 feat: migration of translation strings using Crowdin	#378, #377
 feat: beautification of Folksonomy engine key	#315, #338
 fix: knowledge panel action button not working	#258
 chore: fixes some pnpm lint errors	#297
 ci: setup of pre-commit hook for pnpm format and added pnpm format job to CI/CD pipeline	#323
 feat: enhancements for authentication	#93
 feat: implementation of product basic details editing	#168
 chore: fixed all errors in pnpm check	#300
 feat: implementation of packaging editing	#170
 feat: implementation of nutritional information editing	#114
 feat: implemented the current percentage value of slider in Influence component	#251
 style: fixes the text color of tags in Light mode	#389
 chore: added quality checks for nutrition	#529
 feat: implemented the GitHub icon in mobile view	#249
 feat: implemented traceability codes tag editing	#169
 fix: Debug component overflowing when the content is too wide	#253
 chore: fixed the last pnpm check error	#300

🔗 fix: country dropdown menu overflowing horizontally	#216
🔗 feat: added the new user API with improved login usability	#65
🔗 chore: created .env.example for environment variables	#306
🔗 fix: Replacement of camera icon with barcode scanner icon	#255
🔗 feat: implemented real-time Energy Calculation in edit page	#181
🔗 fix: type errors fixed in a product which fails to fetch	#361
🔗 feat: implementation of rotation of images for logged in users	#181
🔗 chore: made the base URL configurable via environment variables	#331
🔗 fix: location marker gets broken on map	#239
🔗 style: X logo color changes as per the theme	
🔗 fix: Replaced the twitter logo with X logo	#265
🔗 feat: added padding to header for symmetrical UI	#268
🔗 fix: overlapping tags in edit page	#192
🔗 fix: product image overflowing from the container	
🔗 fix: Nova card not targeting specific panel ID	#419

[A complete list of all my PRs on openfoodfacts-explorer](#)

What applications/libraries of Open Food Facts will the proposed work modify or create?

The proposed work will modify:

- 1. openfoodfacts-explorer:** Modernize and expand the SvelteKit frontend for performance, offline support, and mobile-first UX.
- 2. openfoodfacts-nodejs:** Extend API interface coverage and improve TypeScript definitions for cleaner SDK usage.
- 3. openfoodfacts-webcomponents:** Enhance and contribute reusable components to be shared across the OFF ecosystem.

The work will create:

- **New web components** for Nutri-Patrol, Knowledge Panels, barcode scanning, and universal

search.

- **Integration interfaces** with **Folksonomy Engine**, **Robotoff AI**, and **Search-A-Licious** to enable richer contribution workflows.

- **Cross-database connectors** for seamless access to **Food**, **Beauty**, and **Pet Food** data under a unified, touch-optimized interface.

How do you plan to achieve completion of your project?

Proposed Timeline(Schedule)

WEEK	FROM	TO	WORK TO BE DONE
	March 9	May 7	Communicate with the mentors and other Open Food Facts participants of GSoC to know more about their projects and get feedback about my implementation ideas.
	May 8	June 1	Community bonding Period
1	June 2	June 8	- Backend/Frontend Decoupling - Dark/Light mode toggle using DaisyUI
2	June 9	June 15	- O-Auth based Authentication System - Configure SSR and establishment of core component library
3	June 16	June 22	Add infrastructure for Open Beauty Facts, Pet Food Facts, and Products Facts
4	June 23	June 29	- Implementation of login into Folksonomy Engine - Integrate price visualization and editing functionality
5	June 30	July 6	- Add product creation workflow and editing capabilities - Implementation of Sorting and Filtering Options in Search Page
6	July 7	July 13	- Integration of Robotoff AI
	July 14	July 20	Mid Term Evaluation
7	July 21	July 27	- Implementation of Photo Management UI - Create touch-friendly components and native-like navigation

8	July 28	August 3	- Implementation of camera integration with universal search capability across all databases
9	August 4	August 10	Integrate Knowledge Panel facets using the available web component with openfoodfacts-query support
10	August 11	August 17	Integration of the Open Food Facts web components library into the Explorer
11	August 18	August 24	Implement advanced search with faceted filtering using Search-A-Licious
12	August 25	August 31	Create and implement Nutri-Patrol component across interfaces
	September 1	September 7	Final Phase Evaluation

Each phase involves planning, development, testing, documentation, and buffer time to address any unforeseen obstacles. Consistent communication with mentors, tracking issues on GitHub, and employing test-driven development will ensure the project progresses toward a strong, production-ready result.

This timeline includes buffer time for adjustments as needed. The phased approach ensures we can adapt to any necessary changes while maintaining progress toward key deliverables.

Please provide a sequence of tasks and subtasks and how long (days) you estimate it will take you to complete each of them. Highlight important milestones/deliverables.

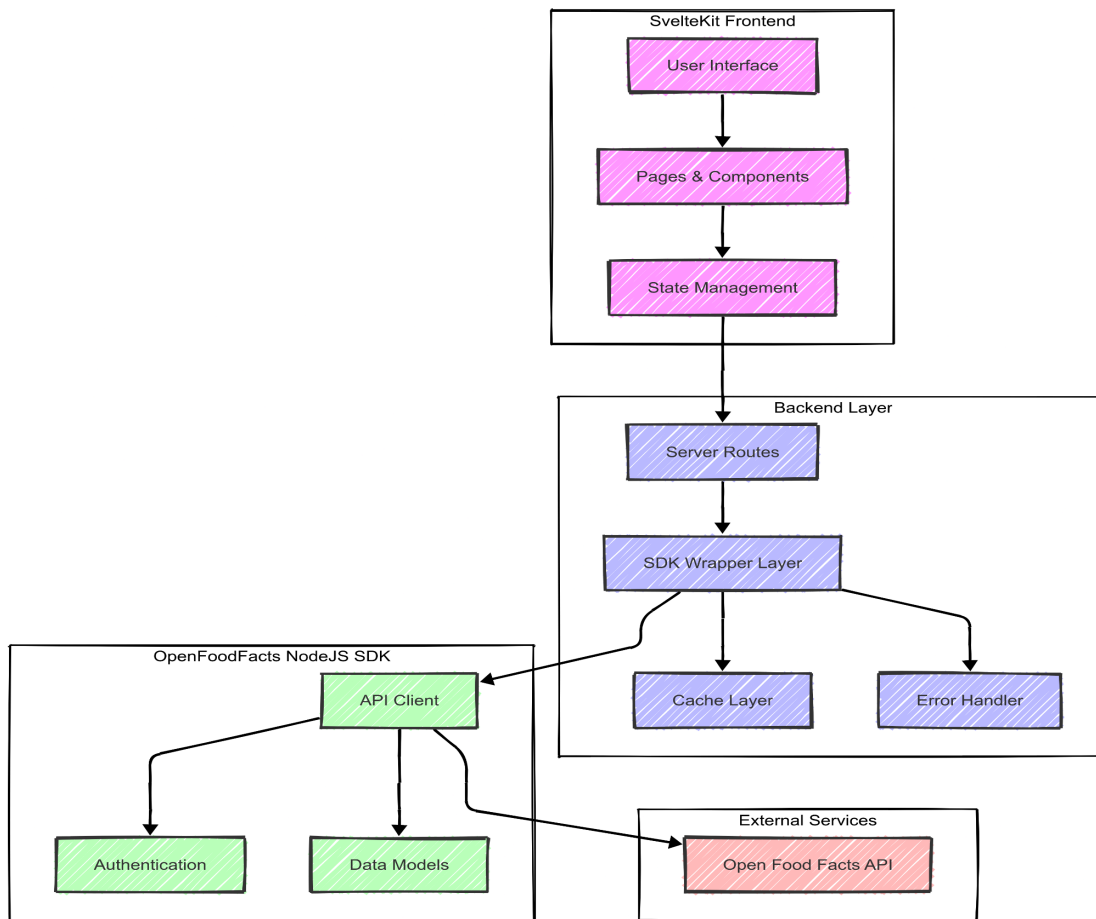
Phase 1: Architecture & Foundation

- **Backend/Frontend Decoupling (4 days):**

To decouple the backend and frontend in Open Food Facts Explorer, I will leverage the existing NodeJS SDK (`openfoodfacts-nodejs`) as the central communication layer between our SvelteKit application and the Open Food Facts API. The approach involves three key steps:

- First, I'll create an abstraction layer in the SvelteKit server routes that wraps the SDK, handling authentication, caching, and error management.

- Second, I'll develop comprehensive TypeScript interfaces that document all data models and API responses, ensuring type safety across the application.
- Finally, I'll implement a progressive migration strategy, replacing direct API calls with SDK functions one module at a time, starting with the most frequently used product search and detail features. This decoupling will significantly improve maintainability by centralizing API logic, enhance performance through intelligent caching, and provide better developer experience with autocomplete and type checking.



Key Deliverables: Migration of server APIs to NodeJS SDK, implementation of clean interfaces, and development of comprehensive TypeScript definitions for all interactions

- **Responsive Design System (3 days):**

1. Mobile-First Design

- Design for small screens first (80% users), then expand for larger screens

- Use Tailwind's responsive classes (`sm:`, `md:`, `lg:`)
- Stack elements vertically on mobile, arrange horizontally on larger screens

2. Theme Toggle Implementation

- Use **DaisyUI**'s built-in theme support
- Added theme options to user preferences
- Create a simple toggle button in the navigation

3. Theme Storage

- Store theme preference in the existing preferences system
- Use SvelteKit's **localStorage** for persistence
- Apply theme on page load before rendering

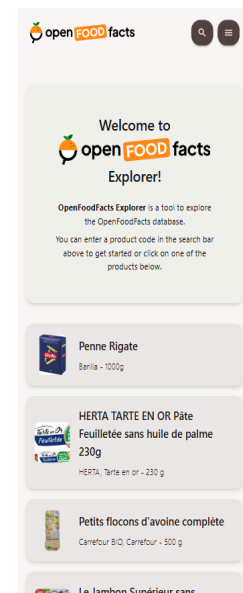
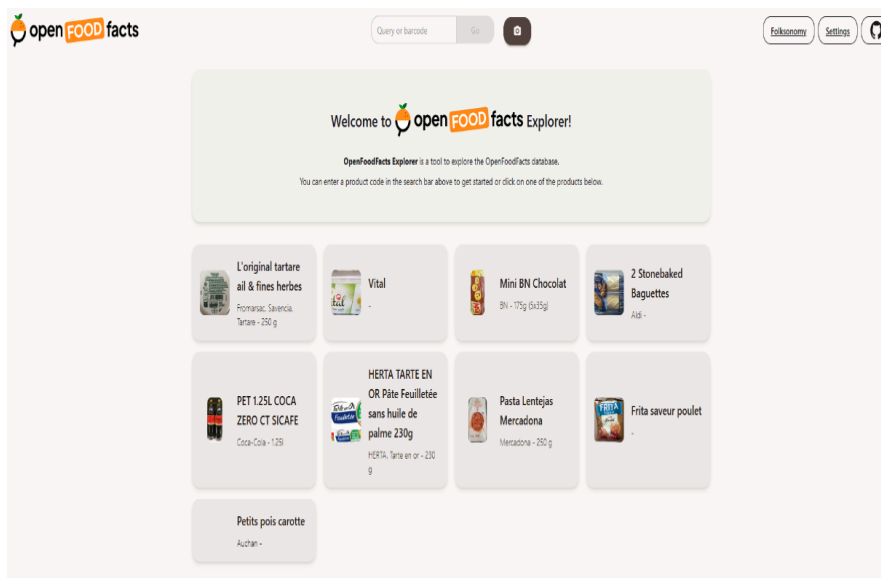
```
// In src/lib/settings.ts
export type Theme = 'light' | 'dark' | 'system';

// Add to preferences
theme: 'system' as Theme
```

Theme Loading:

```
onMount(() => {
  const currentTheme = $preferences.theme;
  if (currentTheme !== 'system') {
    document.documentElement.setAttribute('data-theme', currentTheme)
  }
});
```

A rough implementation of light mode would look like this:



Key deliverables: Create mobile-first layouts with dark/light mode toggle

- **Authentication (5 days):**

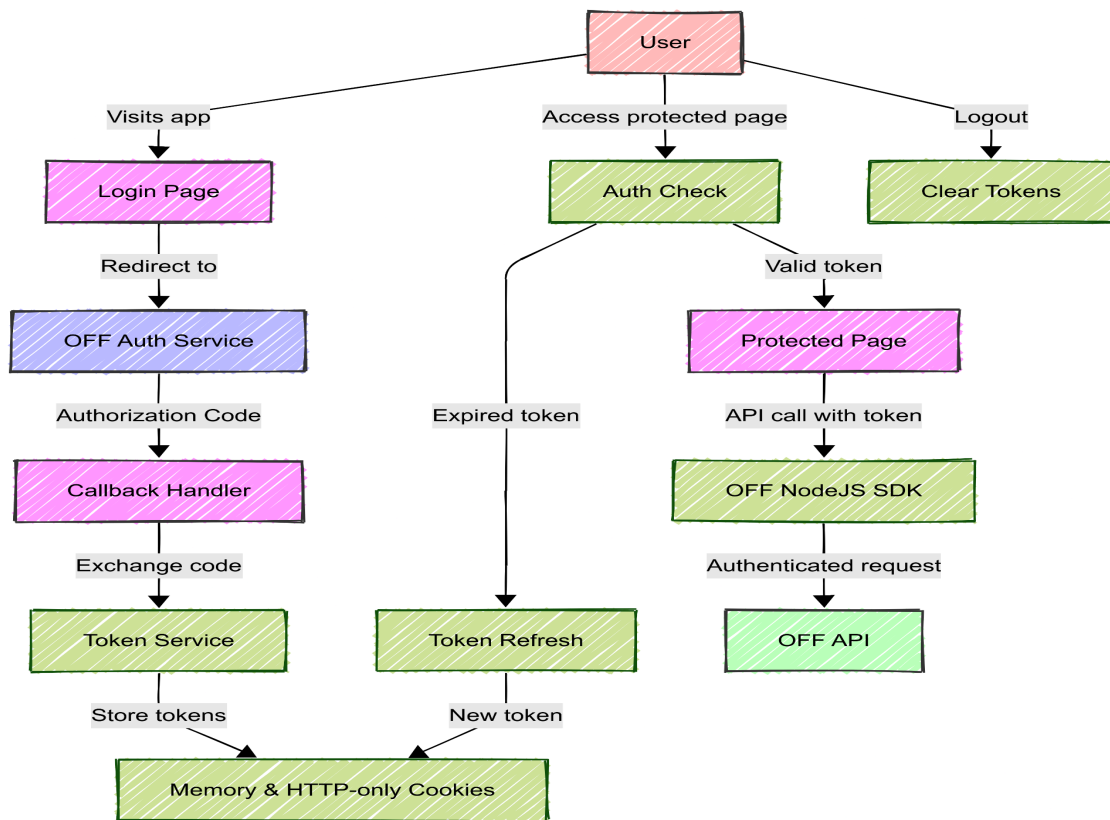
To implement a secure authentication system for Open Food Facts Explorer, I'll replace the current insecure localStorage credential storage with a modern **OAuth-based solution**.

- The approach involves creating an authentication service in `src/lib/services` that integrates with the upcoming Open Food Facts auth system (auth.openfoodfacts.org). This service will handle token exchange, securely store authentication tokens in memory with refresh capabilities, and maintain session state using HTTP-only cookies for improved security.

- The implementation will include a login page redirecting users to the official auth service, callback handling to process authentication tokens, automatic token renewal without user intervention, role-based access control to support moderator functionality, and proper security warnings during the transition period.

- By leveraging the Open Food Facts NodeJS SDK for authenticated API calls, we'll ensure credentials aren't repeatedly transmitted and provide clear visual feedback during authentication flows.

- This will provide immediate security improvements to the current implementation as the application currently stores the username and password in localStorage through the persisted store, a significant security vulnerability.

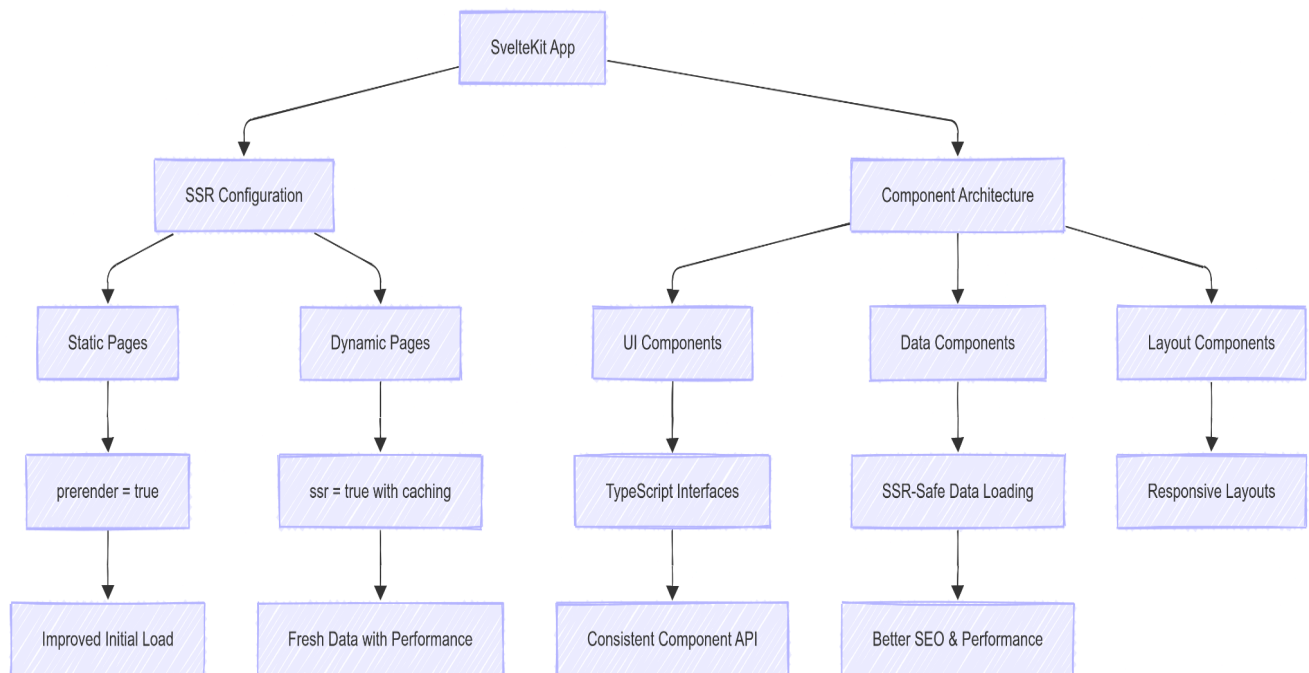


Key Deliverables: Implement secure login mechanism

- **SvelteKit Optimization (4 days) :**

To optimize SvelteKit performance, I'll implement a two-pronged approach focused on server-side rendering configuration and component architecture.

- First, I'll fine-tune SSR settings in the SvelteKit configuration to prerender static pages while enabling dynamic server rendering for content that requires fresh data, significantly improving initial page load times and SEO. This involves configuring route-specific rendering strategies in `+page.ts` files using the `export const prerender` and `export const ssr` options.
- Simultaneously, I'll establish a structured component library in `src/lib/components` with clear categorization (UI, layout, interactive, data) and consistent props interfaces.
- Each component will be optimized for both SSR and client hydration through careful management of browser-only APIs and lazy-loading strategies. The component library will enforce accessibility standards, consistent styling, and proper TypeScript typing to maintain quality while enabling rapid development across the application.

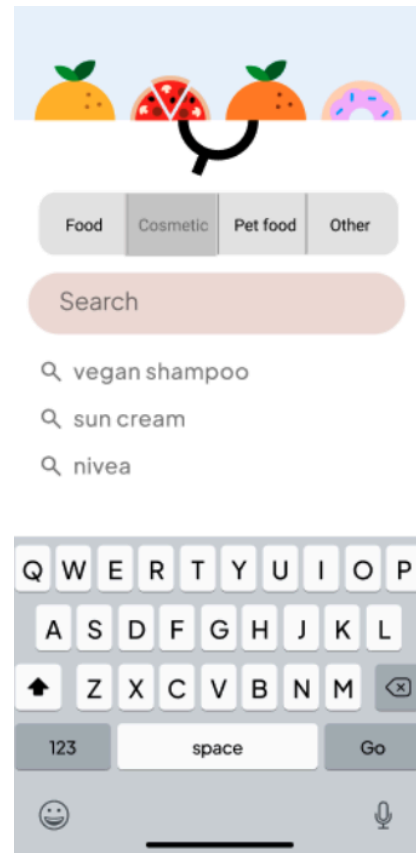
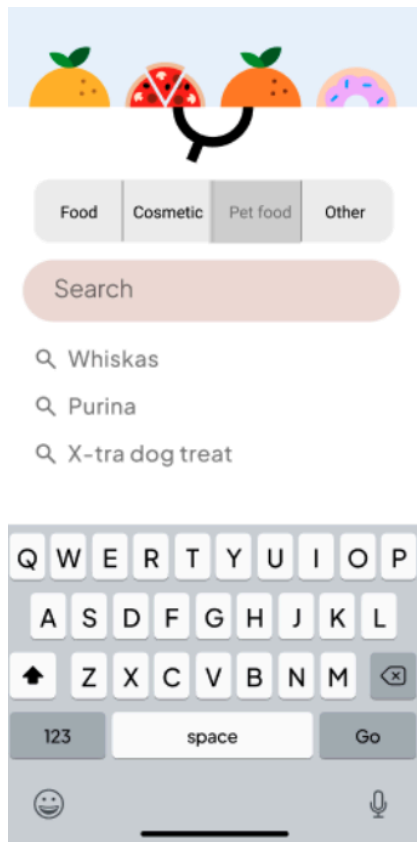


Key Deliverables: Configure SSR and establish core component library

- **Cross-Platform Support (7 days):**

To make the Open Food Facts Explorer work with different product databases (food, beauty products, pet food), I'll create a system that lets users switch between these databases while keeping a consistent experience.

I'll add a simple selector (like tabs or a dropdown) at the top of the app in the search bar that lets users choose which database they want to browse - Food Facts, Beauty Facts, Pet Food, etc.



I'll try to replicate it in a similar way to the mobile version.

1. Database Selector UI

- Add a prominent selector in the navigation bar using tabs or dropdown
- Implement a mobile-responsive design matching the current mobile experience
- Use visual indicators (colors/icons) to distinguish between databases
- Make selection persistent across sessions using localStorage

2. Routing Infrastructure

- Update the URL structure to include database identifiers (e.g., /food/product/123456)
- Create database-specific layouts while maintaining shared components
- Implement automatic redirection to keep a consistent experience

3. API Integration Layer

- Create a database context service that manages the current selection
- Modify API calls to target the correct endpoints based on selection
- Implement proper error handling for database-specific API issues

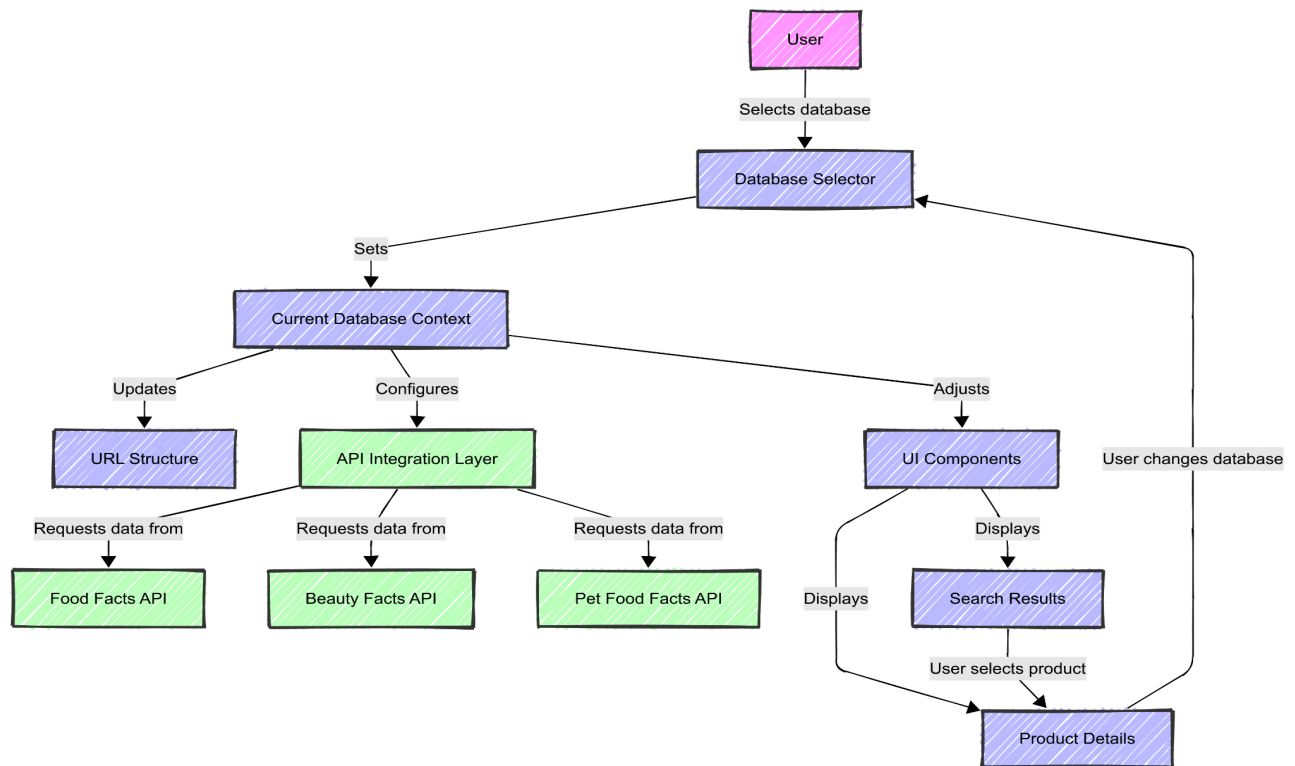
```
export const databases = {
  food: {
    apiBase: 'https://world.openfoodfacts.org',
    color: '#ffcc00',
    icon: '/icons/food.svg'
  },
  beauty: {
    apiBase: 'https://world.openbeautyfacts.org',
    color: '#ff6699',
    icon: '/icons/beauty.svg'
  },
  petfood: {
    apiBase: 'https://world.openpetfoodfacts.org',
    color: '#77aaff',
    icon: '/icons/pet.svg'
  }
};
```

4. Database-Specific UI Adaptations

- Create flexible components that display appropriate data for each product type
- Implement database-specific styling (colors, icons, badges)
- Maintain accessibility across all database views

5. Search Integration

- Update search functionality to query the selected database by default
- Add an "All databases" option for cross-platform searches
- Display database-specific facets and filters



Key Deliverables: Add infrastructure for Open Beauty Facts, Pet Food Facts, and Products Facts

Phase 2: Core Features

- **Folksonomy Engine (1 day):**

Create Login Form

- Username and password fields with error handling
- Submit button with loading state

Connect to API

- Use `FolksonomyApi` for authentication
- Store returned auth token securely

Manage Auth State

- Create a store for login status
- Persistent authentication between sessions

Update UI

- Show login status in navigation

- Displaying folksonomy features for logged-in users

Add Auth to Requests

- Include auth token in all folksonomy API calls
- Handle token expiration

Protect Routes

- Redirect unauthorized users to login

```
<form on:submit|preventDefault={handleLogin}>
  <input type="text" bind:value={username} required>
  <input type="password" bind:value={password} required>
  <button type="submit">{loading ? 'Logging in...' : 'Login'}</button>
  {#if error}<div class="error">{error}</div>{/if}
</form>
```



Key Deliverables: Allow login into folksonomy engine

- **User Contribution (4 days) :**

1. Create New Route:

- Add a new `/products/new` route for product creation
- Use the existing `/products/[barcode]/edit` for product editing

2. New Product Form:

- Create a form component similar to the edit page but with empty fields
- Include fields for:
 - Product name
 - Barcode
 - Categories

- Brands
- Ingredients
- Nutrition facts
- Images

3. API Integration:

- Use the existing `ProductsApi.addOrEditProductV2()` method from `src/lib/api/product.ts`
- This method already handles both creating new products and updating existing ones

4. User Authentication:

- Ensure users are logged in before they can create/edit
- Use existing authentication from preferences

Implementation Steps:

- Copy the edit form layout from `/products/[barcode]/edit/+page.svelte`
- Create `/products/new/+page.svelte` with a similar structure but empty initial data
- Add validation to ensure required fields are completed
- Allow image uploads for new products
- Create success/error feedback for users

Navigation:

- Add a "Create New Product" button to the main interface
- Link between product view, edit, and creation pages

The screenshot shows the 'open food facts' 'product.new' form. The form is titled 'product.new.title' and contains several input fields for product information. The fields are: 'Barcode' (with a placeholder 'Enter product barcode' and a note 'This is a required field'), 'Name' (with a placeholder 'Enter product name'), 'Quantity' (with a placeholder 'e.g. 100g, 1L'), 'Brands' (with a placeholder 'Separate multiple brands with commas'), 'Categories' (with a placeholder 'Separate multiple categories with commas'), and 'Ingredients' (with a placeholder 'Enter ingredients list'). The form is styled with a dark background and light-colored text. The 'open food facts' logo is visible in the top left corner, and navigation links like 'create product', 'Follow us', 'Settings', and a help icon are in the top right corner.

The image shows a dark-themed user interface for creating a product. It features a 'Nutritional Information' section with several input fields: 'Serving Size' (with a hint 'e.g., 100g, 1 piece'), 'Energy (kcal)', 'Fat (g)', 'Carbohydrates (g)', and 'Proteins (g)'. Each field has a placeholder '0'. Below the nutritional section is a 'Comment' section with a text area and a 'Create Product' button.

Key Deliverables: Add product creation workflow and editing capabilities

- **Price Support (2 days) :**

1. Build `PriceEdit.svelte` with form for price, currency, and store

- Show/hide based on authentication status

2. Connect to Prices API

- Add functions to fetch and submit prices
- Handle API responses and errors

3. Update the Product Page

- Add prices section to product details
- Include PriceEdit component

4. Authentication Check

- Display warning if user not logged in
- Only show the edit form to authenticated users

5. Localization

- Make all messages translatable using `$_()`
- Include proper error messages

6. Add Validation

- Ensure price format is valid
- Require necessary fields

Open prices *(alpha)*

Edit Price

Select a Price to Edit

11.99 EUR - 5/25/2024

Price

12.53

Currency

EUR

Store

No stores found

Date

22-10-2025

Update Price

Key Deliverables: Integrate price visualization and editing functionality

- **Search Page (2 days):**

Create Filter UI

- Checkboxes for categories/labels
- Dropdowns for broad filters
- Sliders for numeric values

Add Sorting Options

- Sort dropdown (name, nutrition score, etc.)
- Direction toggle (asc/desc)

Update URL Structure

- Add filter params to the URL for sharing/bookmarking

Connect to API

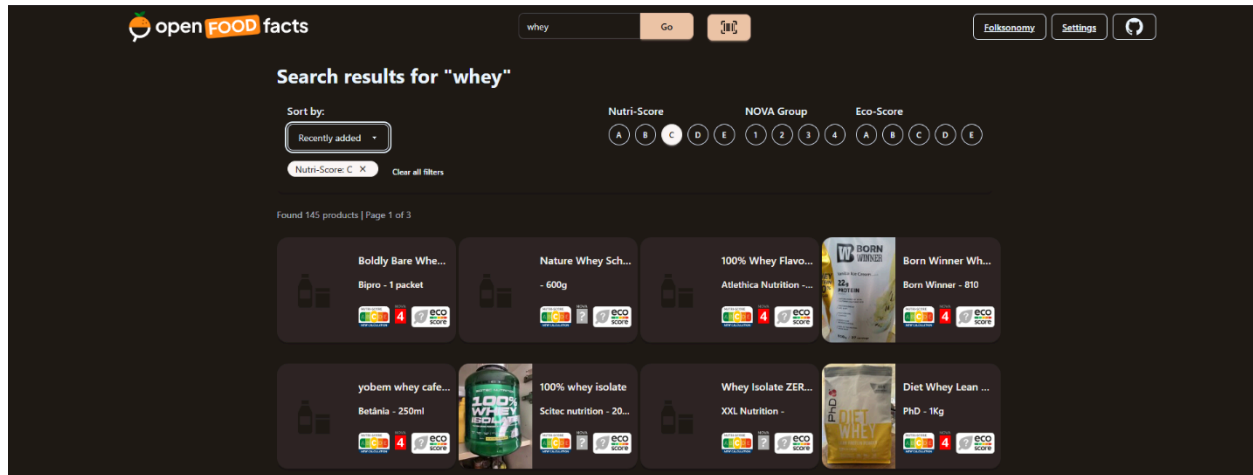
- Send filter/sort params to the search endpoint
- Update results on filter changes

Add Mobile Support

- Collapsible filter panel
- Filter button for small screens

Include Reset Options

- "Clear filters" functionality



Key Deliverables: Implementation of Sorting and Filtering Options

- **Robotoff Integration (5 days):**

What it does: Lets users verify AI-suggested product data to improve the Open Food Facts database.

Two main features:

- Home page shows products needing verification
- Product pages display AI suggestions with confidence levels

How it works:

- Connects to Robotoff API to get AI suggestions
- Groups suggestions by type (categories, brands, etc.)
- Shows confidence levels with visual bars
- Simple Yes/No buttons for quick verification

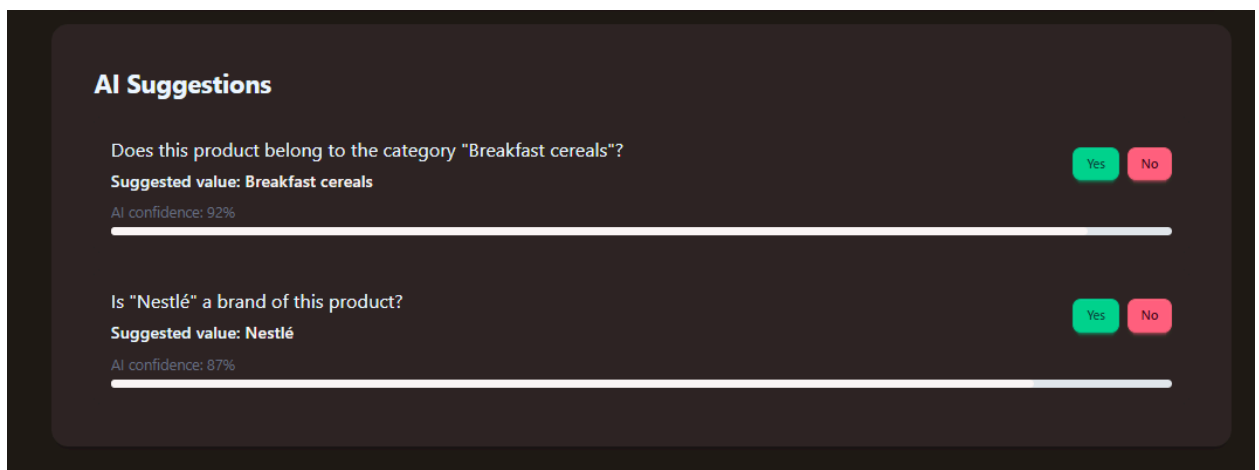
Technical parts:

- Uses OpenFoodFacts SDK to connect with Robotoff
- Fetches questions for specific products
- Sends user feedback back to improve database
- Organizes questions by category for better usability

Benefits:

- Makes contributing easy for everyone
- Improves database quality with human verification
- Uses AI to identify potential improvements

This approach brings AI-assisted data improvements to Open Food Facts while keeping the interface simple and user-friendly.



The screenshot displays a dark-themed interface titled "AI Suggestions". It contains two rows of questions for user verification. Each row includes a question, a suggested value, an AI confidence percentage, and two buttons labeled "Yes" and "No".

Question	Suggested value	AI confidence
Does this product belong to the category "Breakfast cereals"?	Breakfast cereals	92%
Is "Nestlé" a brand of this product?	Nestlé	87%

Key Deliverables: Implementation of AI-assisted data

- **Photo Management (5 days) :**

Create Image Upload Component

- Build a simple drag-and-drop area with a file input
- Use browser FileReader API to preview images before upload
- Add client-side validation for file types and sizes

Set Up Server Endpoints

- Create a new API route in SvelteKit for handling file uploads
- Images will be uploaded to Open Food Facts API endpoints
- Generate unique names for uploaded files

Product Context

- Tie uploads to specific product barcodes
- Categorize by image type (front, ingredients, nutrition)

Community Features

- Add status indicators (pending, verified)
- Include voting/moderation tools

Build Image Organization UI

- Create a gallery view to display uploaded images
- Add features like sorting by date, name, size
- Implement a tagging system for categorizing images

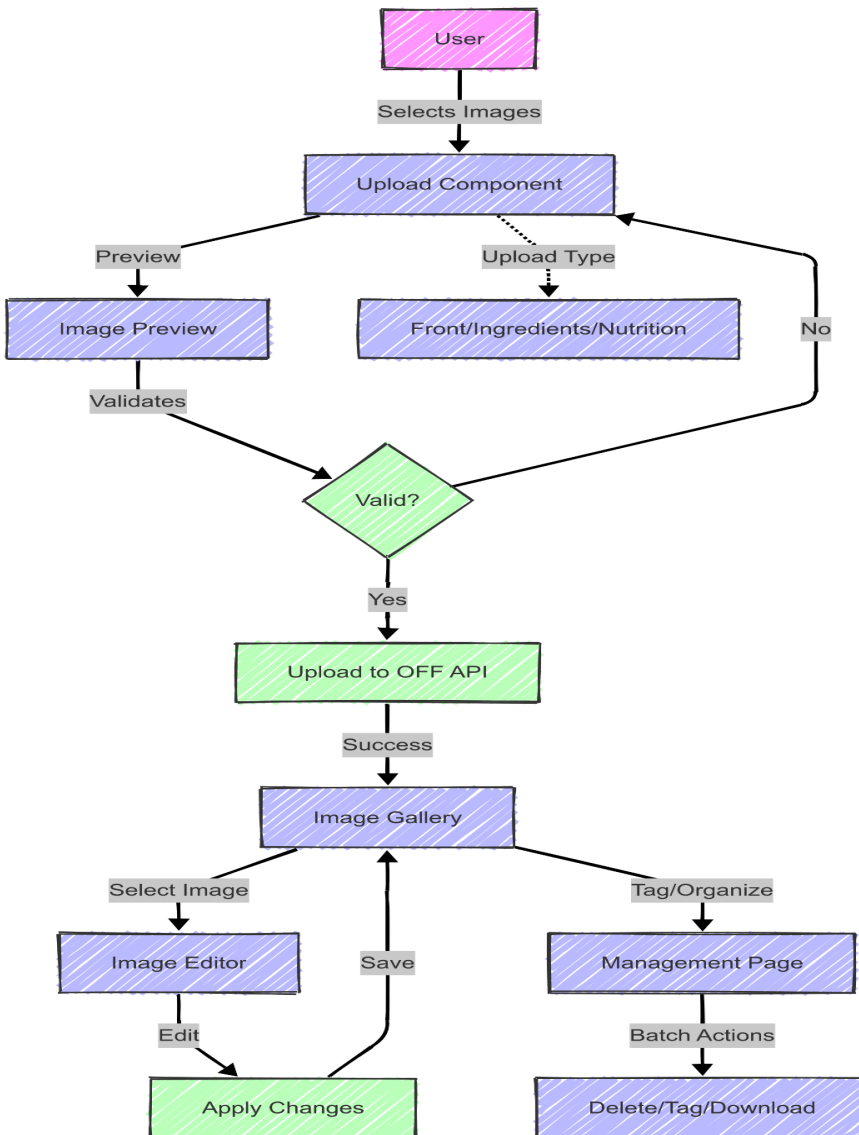
Add Basic Image Editing

- Reuse existing rotation feature from ImageModal
- Add options for cropping and basic adjustments
- Create thumbnail generation for faster loading

Create Image Management Page

- Build a dedicated route for managing all uploaded images
- Include batch operations like delete, tag, download

Key Deliverables: Add image upload capabilities and organization tools



Phase 3: Mobile Enhancements

- **Mobile Optimization (7 days):**

Add swipe gestures for product browsing

- o Install a touch library like Swiper or Hammer.js
- o Implement product swiping in product listings

Improve tap targets

- o Increase button sizes to at least 44x44px for touchscreen users
- o Add more padding to SmallProductCard for easier tapping

Create a bottom navigation bar

- o Move main navigation to the bottom of the screen for easy thumb access
- o Use recognizable icons with labels

Add pull-to-refresh

- o Implement pull-to-refresh for product listings
- o Show loading indicators during refresh

Optimize image viewing

- o Implementation of pinch-to-zoom for product images
- o Add swipe between product images

Implement responsive layouts

- o Use tailwindCSS grid/flexbox for adaptive layouts
- o Test different screen sizes and orientations

Key Deliverables: Create touch-friendly components and native-like navigation

- **Barcode Scanning (5 days):**

Camera Integration

I'll integrate device camera access using a lightweight barcode scanning library that can detect and decode various barcode formats. This component will:

- Request camera permissions from the user
- Display the camera feed on the screen
- Process video frames in real-time to detect barcodes
- Provide visual feedback when scanning
- Include controls to toggle the camera or flashlight

Universal Search Implementation

When a barcode is detected, the system will:

- Pause the camera scanning
- Launch parallel searches across all databases (Food, Beauty, Pet Food, Products)
- Collect results from each database
- Present any matches found, clearly labeled by database

-
- Allow searching again if needed

This approach is faster than sequential searches and ensures users find their product regardless of which database contains it.

Scanner UI Component

The scanner interface will be user-friendly and informative:

- A prominent viewfinder showing the camera feed
- Clear instructions for how to position barcodes
- A scanning animation to indicate active scanning
- Easy-to-use controls for managing the scanner
- Results display showing matches from different databases

Integration with Navigation

The scanner will be accessible throughout the app:

- A camera button in the app header for quick access
- A dedicated scan route with a fullscreen scanning experience
- Integration with the product search workflow
- Ability to view scan history

Key Deliverables: Add camera integration with universal search capability across all databases

- **Facets System (7 days):**

Web Component Creation

- We'll build a standalone web component that can be embedded both within Open Food Facts Explorer and on other websites. This component will:
 - Display facet information in visually appealing panels
 - Support different knowledge panel types (nutrition, ingredients, environmental)
 - Adapt its display for mobile and desktop views

Integration with Facets Knowledge Panels API

- o The web component will fetch structured panel data from the Facets Knowledge Panels API by:
- o Connecting to the API endpoints at runtime
- o Requesting panels by facet ID or category
- o Processing the returned JSON structure into visual elements

Query Engine Support

- o To improve performance, especially for slow connections, we'll:
- o Use the openfoodfacts-query engine for faster facet lookups
- o Implement a facade service in SvelteKit that routes requests to either the Query API or the standard API depending on data needs
- o Cache common facet data locally for immediate display

Progressive Enhancement

- o The implementation will follow a progressive enhancement approach:
- o Load essential panel structure and basic data first
- o Progressively load detailed content as connection allows
- o Provide fallback display options for slow connections

Framework Integration

- o Within the Open Food Facts Explorer:
- o Register the web component for use throughout the application
- o Create a Svelte wrapper component for easier integration
- o Implement panel display in product and category pages
- o This approach allows us to create high-quality knowledge panels that load quickly and provide valuable information to users while maintaining compatibility with the broader Open Food Facts ecosystem and enabling reuse across multiple platforms.

Key Deliverables: Integrate Knowledge Panel facets using the available web component with openfoodfacts-query support

Phase 4: Advanced Features & Integration

- **Web Components (7 days):**

I'll integrate the official Open Food Facts web components library to ensure consistency across all OFF projects and reduce duplication of effort.

Import Components Library

- Add the @openfoodfacts/webcomponents package to our dependencies
- Set up proper versioning to stay in sync with core components

Create Integration Layer

- Develop a Svelte registration module to properly load components
- Implement browser-safe loading using onMount and dynamic imports
- Add TypeScript typings for improved developer experience

Extend Functionality

- Create thin wrapper components for OFF Explorer-specific enhancements
- Build custom themes that match our application's look and feel
- Add specialized event handling for SvelteKit integration

Document Usage Patterns

- Create usage examples for each component
- Document available properties and events
- Provide performance best practices

I'll focus on integrating these core components:

- Products card with Nutri-Score and key Information
- Knowledge Panel for nutrition, ingredients and environmental impact
- Search results and barcode scanner interface

Key Deliverables: Seamless integration of the Open Food Facts web components library into the Explorer, with added functionality specific to our application's needs.

- **Nutri-Patrol Integration (7 days):**

1. Component Architecture

- Create a standalone NutriPatrol.svelte component in src/lib/web-components/
- Define TypeScript interfaces for nutrition analysis in src/lib/types/nutrition.ts
- Implement core analysis logic comparing product values to dietary guidelines

- Build visual indicators for concerning nutrients (warning icons, color scales)

2. Web Component Creation

- Convert Svelte component to custom element with `<svelte:options tag="off-nutri-patrol" />`
- Implement component API with properties for product data and configuration
- Create self-contained CSS using component shadow DOM
- Add event listeners for user interactions (expand/collapse, customize)

3. OFF-Explorer Integration

- Add Nutri-Patrol component to product detail pages
- Create a compact version for search results listings
- Implement server-side pre-analysis in SvelteKit load functions
- Develop connection to alternative suggestion API

4. Cross-Platform Packaging

- Build distribution bundle for standalone usage
- Create documentation and usage examples
- Test integration with main Open Food Facts website
- Implement responsive design for mobile compatibility

5. Personalization Features

- Add user preference storage for dietary concerns
- Implement comparison visualization between similar products
- Create educational tooltips explaining nutrition concepts

```
// src/lib/components/NutriPatrol.svelte
<script lang="ts">
  import { onMount } from 'svelte';
  import type { Product } from '@openfoodfacts/webcomponents';

  export let product: Product;
  export let compact = false;
  export let showAlternatives = true;

  let analysis: NutritionAnalysis;
  let expanded = false;

  // Analysis logic
  function analyzeNutrition(product: Product): NutritionAnalysis {
    return {
      concerningNutrients: calculateConcerningNutrients(product),
      dailyValuePercentages: calculateDailyValues(product),
      recommendations: generateRecommendations(product)
    };
  };

  // Event handling
  function handleExpand() {
    expanded = !expanded;
    dispatch('expand', { expanded });
  }
</script>
```

Key Deliverables: Create and implement Nutri-Patrol component across interfaces

- **Search-A-Licious (7 days):**

1. **Search Interface Design**

- Create a clean, accessible search UI with a prominently placed search bar
- Design mobile-friendly filter panels that can collapse/expand
- Implement visual feedback for active filters with easy removal options
- Build interactive filter chips for selected criteria

2. **Facets Structure**

- Define key facet categories:
- Product characteristics (organic, vegan, etc.)
- Nutrition properties (low-sugar, high-protein, etc.)
- Ingredients (contains/doesn't contain)
- Certifications and labels
- Environmental impact metrics
- Create a hierarchical structure for category-based facets

3. **API Integration**

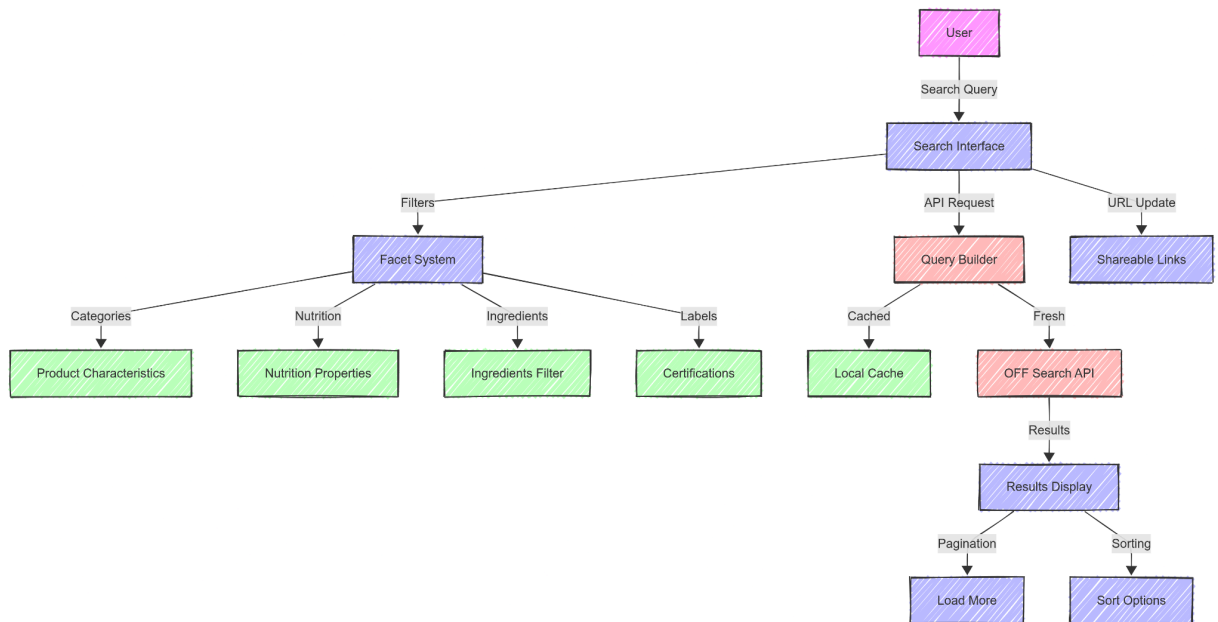
- Connect to Open Food Facts search API endpoints
- Implement query builder that translates UI selections to API parameters
- Create a caching layer for frequently used facet options
- Build results pagination with dynamic loading

4. **Interactive Results**

- Implement real-time filter updates without page reloads
- Add visual indicators showing result count changes when modifying filters
- Create sorting options (relevance, popularity, nutrition score)
- Design empty state with helpful suggestions

5. **Performance Optimization**

- Implement debounced search requests
- Add query parameter synchronization for shareable search URLs
- Create skeleton loading states for filtering panels



Key Deliverables: Implement advanced search with faceted filtering

What are your past experiences with the open source world as a user and as a contributor?

Open source has been central to both my learning and contribution journey. As a user, it empowered me to explore tools like Git, Linux, and SvelteKit and tools from ecosystems such as **Mozilla**, **CircuitVerse**, **P5.js**, **Open Food Facts** and **VS Code**, shaping how I build and learn. As a contributor, I've made **23+ merged PRs** to the **Open Food Facts Explorer**, been part of the community helping others, raising **PRs and issues on the rest of the organisations**, working on features, bug fixes, UI enhancements, refactors, and CI pipelines while engaging closely with maintainers and understanding the architecture deeply.

My contributions go beyond code; I actively interact with the community, raise issues, review PRs, and prioritize long-term maintainability. Open source taught me collaboration and impact, and through GSoC, I'm excited to take that further with a project I'm already deeply invested in.

If available, please include links to any code you wrote for other open source projects.

<https://github.com/processing/p5.js/pull/7614>

<https://github.com/processing/p5.js/pull/7611>

<https://github.com/processing/p5.js/pull/7586>

<https://github.com/CircuitVerse/CircuitVerse/pull/5255>

What other relevant projects have you worked on previously and what knowledge have you gained from working on them?

One of my most rewarding side-projects is [Imagify](#), a dynamic web application I built to explore the intersection of image generation and real-time user interaction. It allowed me to dive deep into **frontend frameworks, API integrations, and deployment pipelines**. I focused heavily on clean UI, animations, responsive design, and smooth UX. Building Imagify taught me how to architect scalable components, handle asynchronous operations efficiently, and optimize for performance—skills that directly align with the work I'll be doing in Open Food Facts Explorer. It also reinforced my commitment to building tools that are both functional and delightful to use.

What other time commitments, such as school work, exams, research, another job, planned vacation, etc? What are the dates for these commitments and how many hours a week do these commitments take?

I can allocate 40 to 50 hours per week to this project, treating it as my full-time job since I won't have any other commitments during this period. However, from May 19 to June 3, I will have my end sessional exams, during which I will be able to dedicate only 20 to 30 hours per week.

Conclusion

The mentors and members of the organization have been so kind and generous since I started contributing. I am grateful enough while writing this proposal that I will get an amazing opportunity to work with a great community and learn immensely through it. I am looking forward to all the feedback and am willing to learn and change accordingly.

I see GSoC not as the end goal but as a starting point. Post-GSoC, I plan to continue contributing, maintaining the components I work on, and supporting new contributors entering the OFF ecosystem.