# GaveGuiden: Master Project Specification & Technical Report

**Version:** 7.0 (Definitive Master Blueprint) **Date:** July 26, 2025 **Author:** Gemini, Lead Developer

## 1.0 Executive Summary & Vision

This document serves as the single source of truth for the GaveGuiden project. It is a comprehensive blueprint detailing the project's vision, business strategy, user experience, technical architecture, and operational workflows. It is designed to be fully self-contained, requiring no external context for complete understanding.

### 1.1 Project Vision

Our vision is to create the most intelligent, trusted, and user-friendly starting point for gift discovery in the Danish market. GaveGuiden will eliminate the friction and decision fatigue of traditional online shopping by providing a delightful, guided experience that results in perfect, personalized gift recommendations.

### 1.2 Business Model & Strategy

> **Core Concept: Affiliate Marketing** We build a highly useful tool that recommends products. When a user clicks our recommendation and buys the product from the seller's website, we earn a percentage of that sale.

The foundational business strategy is to achieve and maintain profitability by minimizing operational costs. We accomplish this through a **Serverless Architecture**.

> **What is a Serverless Architecture?** Traditionally, websites need a server that is always on, waiting for visitors. This incurs constant costs. A serverless approach means we don't manage a server. Our website is a collection of static files (HTML, CSS, JS) served globally from a Content Delivery Network (CDN), which is extremely fast and cheap. For dynamic actions (like tracking clicks), we use small, on-demand "serverless functions" that run for a few milliseconds, and we only pay for that exact

execution time. For a project like ours, this often falls within a generous free tier, making our operational costs effectively zero at the start.

### 1.3 Unique Selling Proposition (USP)

GaveGuiden's competitive advantage is a **superior, intelligent, and adaptive user experience**. While competitors offer simple filter pages, we provide a guided, conversational journey that feels both personal and magical. Our "smart" quiz dynamically adapts to user input, asking only the most relevant next question, a feature unmatched by existing tools.

## 2.0 The User Experience (UX) & Journey

The user's journey is designed to be a single, seamless flow from arrival to recommendation, feeling more like a polished mobile app than a traditional website.

### 2.1 User Journey Flowchart

The user's journey begins on the Landing Page. From there, they initiate the Quiz, which presents a series of dynamic questions. At any point after the first few questions, the user has an optional path to get an immediate recommendation. Otherwise, they complete the quiz naturally. Both paths lead to the Results Page, where they see the recommended product(s). The final step is clicking through to the partner's affiliate store to make a purchase.

### 2.2 Detailed Interaction Design

- **Transitions:** All view changes (e.g., landing to quiz) will use a gentle cross-fade and slight vertical slide animation, lasting approximately 300ms. This prevents jarring page reloads and maintains a sense of place.
- **Micro-interactions:** Buttons will subtly change size or shadow on hover. Selected quiz options will have a clear, colored border and checkmark. These small details make the interface feel responsive and alive.
- **The "Early Exit":** This button will appear with a soft fade-in animation, ensuring it doesn't distract from the question at hand but is available once the user has provided enough information for a basic recommendation.

## 3.0 Technical Architecture & System Design

### 3.1 System Architecture Diagram

The system has three core components: the User's Browser, the Hosting Provider, and our Serverless Functions.

1. The **User's Browser** loads the initial website files from the **Hosting Provider**.
2. All quiz logic happens within the browser for speed.
3. When a dynamic action is needed (like tracking a click), the browser sends an API call to a **Serverless Function**.
4. The Serverless Function then securely reads from or writes to our data files (like `analytics.json`), which are also stored with the Hosting Provider.

### 3.2 Technology Stack & Rationale

- **Languages:** HTML5, CSS3, JavaScript (ES6+).
    - **Why:** These are the universal languages of the web. By avoiding heavy frameworks, we ensure maximum performance, faster development for our specific needs, and zero external dependencies.
- **Styling:** Tailwind CSS.
    - **Why:** A utility-first CSS framework that allows us to build a custom, responsive design quickly without writing custom CSS files. It keeps our HTML clean and our design consistent.
- **Hosting & Backend:** Netlify or Vercel.
    - **Why:** These platforms are built specifically for our serverless architecture. They offer a global CDN for speed, an integrated environment for serverless functions, and a generous free tier that will cover our costs for the foreseeable future.

### 3.3 Detailed File Structure

```
/

├── index.html          // The single HTML file for the user-facing app.

├── admin.html          // The single HTML file for the admin dashboard.

|

├── assets/
```

```
│   ├── products.json      // The core database of all gift products.
│   ├── analytics.json     // Stores aggregated analytics data.
│   └── flags.json         // Stores user-submitted error reports.
|
└── functions/             // Directory for our serverless backend logic.
    ├── update-analytics.js // Handles events like a product click.
    └── submit-flag.js     // Handles user submissions of product errors.
```

## 4.0 The Dynamic Recommendation Engine (Core Logic)

This is the "brain" of GaveGuiden, composed of two distinct but interconnected parts.

### 4.1 The Scoring Algorithm

After every answer, this algorithm re-evaluates all `active` products.

**Example Walkthrough:**

A user selects "Mand" (Gender) and "Kaffe" (Interest).

- **Product A (Coffee Machine):** `tags: { gender: ["alle"], interests: ["Kaffe", "Hjemmet"] }`
  - Gets +3 points for `gender: "alle"`.
  - Gets +10 points for `interests: "Kaffe"`.
  - **Total Score: 13**
- **Product B (Perfume):** `tags: { gender: ["Mand"], interests: ["Mode", "Wellness"] }`
  - Gets +3 points for `gender: "Mand"`.
  - Gets 0 points for interests.
  - **Total Score: 3**

The Coffee Machine is now the clear front-runner.

### 4.2 The Dynamic Question Engine

This engine decides what to ask next to provide the most value.

**Example Walkthrough:**

The top 3 products are now all coffee machines from the same brand, with the `product_group_id: "COFFEE-MACHINE-X"`.

1. **Engine State:** The candidate pool is dominated by one product group.
2. **Analyze Group:** The engine checks the first product in the group and sees `differentiator_tags: ["color", "material"]`.
3. **Check User History:** The engine checks if it has already asked about "color" or "material". It has not.
4. **Select Question:** It picks the first item from the `differentiator_tags` array: `"color"`.
5. **Generate Question:** It creates a question: "Hvilken **farve** ville passe bedst?"
6. **Generate Answers:** It looks at all products in the `COFFEE-MACHINE-X` group and finds the unique values for the `color` tag: `["Sort", "Stål", "Hvid"]`. These become the answer options.

## 5.0 Visual Identity & Design System

- **Brand Name:** GaveGuiden
- **Core Values:** Simplicity, Clarity, Trust, Intelligence.
- **Color Palette:**
  - **Primary (Action/Focus):** Slate Blue
  - **Text & Headers:** Charcoal
  - **Background:** Off-White
  - **Borders & Accents:** Light Gray
- **Typography:**
  - **Font Family:** 'Poppins' (from Google Fonts).
  - **Headings:** Bold weight.
  - **Body Text:** Regular weight.
- **Layout & Spacing:**
  - A consistent 8-point grid system will be used.
  - All interactive elements will have a soft corner radius.

# 6.0 Admin Panel & Operations Dashboard

The `admin.html` page is a comprehensive internal tool for managing the entire application.

## 6.1 Access & Security

Initial access will be via a simple JavaScript password prompt, layered with `.htpasswd` basic authentication on the server for production.

## 6.2 Analytics Dashboard

This is the landing page for the admin panel, providing key performance indicators:

- **Top-Level Metrics:** Quiz Completions, Click-Through Rate (CTR), Early Exit Rate.
- **Top 5 Performing Products:** A list of the most-clicked items.
- **Manual Sales Tracking:** A dedicated section for logging confirmed affiliate sales.

## 6.3 Flagged Product Review

This section provides tools to act on user-submitted feedback:

- **Configuration:** An input field to set an "Alert Threshold" (e.g., `3` flags).
- **Alerts Dashboard:** A list of products that have crossed the flag threshold.
- **Flag Aggregation Logic:** The system counts identical, unresolved flags for each product ID to prevent noise from single errors.
- **Detailed Flag View:** A detailed list of all flags for an item, with options to "Resolve Flag" or "Deactivate Product".

## 6.4 Product Management

This section contains the core content management tools, including a product table with analytics data and forms for adding/editing products individually or in bulk.

## 6.5 Saving Mechanism & Architecture

"Write" actions (like saving a change) are handled by sending a secure request to a serverless function, which then safely updates the appropriate JSON file on the server.

## 7.0 Automated Lead Generation & Workflow

### 7.1 LLM Prompt for Automated Discovery & Variation Expansion

This is the primary prompt for finding new products.

**Role:** You are an expert e-commerce curator and data analyst specializing in unique and high-quality gifts available in Denmark.

**Objective:** Find 10 unique gift *ideas* that fit a specific niche. For each product idea, you must perform detailed analysis...

**CRITICAL INSTRUCTION: Before you begin, you MUST visit the following URL and use its contents as an exclusion list...**

**Exclusion List URL:** https://[our-future-domain.com]/assets/products.json

**Variation Handling Clause:**

If a product you select has multiple variations (e.g., 3 colors), you must create a separate JSON object for *each individual variant*. These variations do not count towards your goal of finding 10 unique product ideas.

**Tagging & Variation Analysis Instructions:**

1. **Analyze Variations:** ...list these in the `differentiator_tags` array.

2. **Populate Tags:** ...populate the `tags` object.

**Output Format (Strict):**

The output MUST be a valid JSON array...

### 7.2 Content Curation Workflow

The workflow for turning automated leads into live products is as follows:

1. The process begins with an **Automated LLM Run**, which generates a list of product leads in JSON format.
2. This list undergoes **Human Review & Curation**.
3. During the review, products are either **Discarded** (if low quality) or **Approved**.
4. Approved products move to a **Verify & Enhance Data** step, where the curator confirms the accuracy of all information.
5. Finally, the verified data is published to the live site via the **Admin Panel**.

## 8.0 Go-to-Market & Launch Strategy (Phase 1)

This phase focuses on achieving maximum viability and user acquisition at the lowest possible initial cost.

### 8.1 Minimum Viable Product (MVP) Feature Set

The initial launch will include all features detailed in sections 1-7, plus:

- **The "Share Results" Viral Loop:** A button on the results page to generate a unique, shareable URL. This is our primary engine for free, organic marketing.
- **Occasion-Specific SEO Landing Pages:** Static HTML pages targeting high-intent search terms (e.g., `gaveguiden.dk/julegaveideer`) to funnel users directly into the quiz.

### 8.2 Cost & Resource Optimization

- **Hosting & Domain:** We will use the free tiers of **Netlify** or **Vercel**. The only initial cost is a domain name. **Projected initial cost: <$20.**
- **Initial User Acquisition:** We will capitalize on sign-up offers from **Google Ads** and **Microsoft Advertising** ("spend $25, get $100") to provide a virtually free initial ad budget.

## 9.0 Post-Launch Roadmap & Growth (Phase 2)

Once the model is validated, we will reinvest a small monthly budget to enhance features and automate processes.

### 9.1 Projected Budget & Allocation

- **Target Monthly Budget:** $50 - $100 USD.
- **Allocation:** Analytics (~$15), Serverless Functions (~$10), Advanced LLM Calls (~$25+), Ad Experiments (Remainder).

### 9.2 Roadmap: Upcoming Features & Integrations

- **Update 1: Advanced Analytics Integration**
    - **Feature:** Integrate a privacy-focused analytics service (e.g., Fathom).
    - **Goal:** Gain deep insights into quiz drop-off points for data-driven optimization.
- **Update 2: Wishlists & Gift Registries**
    - **Feature:** Allow users to save recommendations to a public "wishlist" with a shareable URL.
    - **Goal:** Increase user retention and create a new viral sharing mechanism.
- **Update 3: Automated Product Health Monitoring**
    - **Feature:** A scheduled serverless function (`cron job`) that daily checks for 404 errors and price changes on product URLs.
    - **Goal:** Automate database quality control and reduce manual maintenance.
- **Update 4: Smarter LLM Curation Pipeline**
    - **Feature:** A two-step LLM chain that uses a powerful model to quality-check the output of a cheaper discovery model.
    - **Goal:** Reduce manual curation time and improve the quality of product leads.

## 10.0 Discarded Concepts & Strategic Rationale

To maintain focus and manage costs, the following concepts have been explicitly rejected for the foreseeable future:

- **LLM-Powered Live Quiz:** Prohibitively high and unpredictable API costs, slower response times, and lack of result consistency.
- **User Accounts & Profiles:** Adds significant user friction and introduces major data privacy (GDPR) and security complexities.

- **Complex Database (SQL, Firestore, etc.):** Gross over-engineering for our needs. A static `products.json` file is the fastest and cheapest possible data source at our scale.

## Appendix A: Data Model Schemas

### `analytics.json`

```
{
  "products": {
    "101": { "suggestions": 1502, "clicks": 234, "sales": 12 },
    "102": { "suggestions": 1490, "clicks": 188, "sales": 8 }
  },
  "site_totals": {
    "quizzes_started": 2105,
    "quizzes_completed": 1402,
    "early_exits": 703
  }
}
```

### `flags.json`

```
[
  {
    "flag_id": "f001",
    "product_id": 201,
    "issue_type": "Wrong Price",
```

    "timestamp": "2025-07-26T21:00:00Z",

    "status": "unresolved"

  }

]

## Appendix B: Serverless Function APIs

`POST /functions/update-analytics`

- **Purpose:** To log an analytics event.

**Request Body:**
{

  "eventType": "suggestion" | "click",

  "productId": 101

}

- 
- **Action:** Reads `analytics.json`, increments the appropriate counter, and saves the file.

`POST /functions/submit-flag`

- **Purpose:** To submit a user-generated error report.

**Request Body:**
{

  "productId": 201,

  "issueType": "Wrong Price" | "Missing Image" | "Broken Link"

}

-

- **Action:** Reads `flags.json`, appends a new flag object, and saves the file.