Product Requirements Document: AI-Powered Product Wishlist

1. Introduction

1.1 Purpose

This document outlines the requirements for an AI-powered product wishlist application. The primary goal is to provide a user-friendly tool that helps users manage their desired products by automatically extracting and summarizing key information from a product URL. The application will be a single-page web app built with a modern framework (e.g., React, Angular, or a single-file HTML/JS/CSS app) and will use Firestore for data persistence.

1.2 Scope

- **In-Scope:**
  - User authentication (anonymous or custom token).
  - Adding product URLs to a list.
  - AI-powered extraction and summarization of product details.
  - Real-time updates to the wishlist.
  - Displaying a list of products with their details.
  - Deleting products from the list.
  - Public and private wishlist sharing.
- **Out-of-Scope:**
  - User account management (password reset, email verification, etc.).
  - E-commerce integration (e.g., purchasing directly from the app).
  - Price tracking alerts.
  - Browser extensions.

1.3 Target Audience

This application is for individuals who want an easy and efficient way to manage their online shopping wishlists without manual data entry.

2. User Stories

2.1 User

- **As a user**, I want to add a product to my wishlist by simply pasting its URL, so I don't have to manually type in all the details.
- **As a user**, I want the app to automatically summarize the product's name, description, and key features, so I can see the most important information at a glance.
- **As a user**, I want to see a list of my saved products with their summarized details, so I

can easily browse my wishlist.
- **As a user**, I want to be able to delete a product from my list, so I can keep it organized.
- **As a user**, I want to see my userId displayed on the page so I can easily share my list with others.

2.2 Shared List User

- **As a shared list user**, I want to view another user's public wishlist by entering their userId, so I can see what they're interested in.

3. Functional Requirements

3.1 User Interface (UI)

- The application will be a single-page web application.
- The UI must be clean, responsive, and easy to use on both desktop and mobile devices.
- It will have a clear input field for pasting a product URL.
- There will be a Add to Wishlist button.
- A loading indicator will be shown while the AI is processing the URL.
- The wishlist will be displayed as a list of cards or rows, each representing a product.
- Each product card will display the following information:
  - Product Name (from AI)
  - Product Description (from AI)
  - Image (from AI)
  - Original URL
- Each product card will have a Delete button.
- The userId of the current authenticated user will be prominently displayed on the page.
- There will be a separate input field and button to view a public wishlist by userId.

3.2 AI Integration

- The application will use a Large Language Model (LLM) via the Gemini API to process the URL content.
- **API Model:** gemini-2.5-flash-preview-05-20
- **Input:** A POST request to the generateContent endpoint with the following payload:
  - **System Instruction:** Act as a helpful product expert. When given a URL, extract the product name, a concise summary of the product's features and description (less than 100 words), and a URL for the main product image. Respond with a valid JSON object only.
  - **User Query:** A string containing the raw HTML content of the provided product URL. The HTML will be fetched using a server-side or proxy method to avoid CORS issues.
- **Output:** The LLM will return a JSON object with the following structure:

```
{
  "productName": "string",
  "description": "string",
  "imageUrl": "string"
}
```

- **Error Handling:** If the AI fails to extract information or returns an invalid JSON, the application should display a user-friendly error message.

3.3 Data Persistence

- The application will use Firestore for data storage.
- Anonymous authentication will be used initially. If __initial_auth_token is provided, sign in with it; otherwise, sign in anonymously.
- Each user's wishlist will be stored in a collection specific to their userId.
- **Collection Path (Private):** /artifacts/{__app_id}/users/{userId}/wishlist
- **Collection Path (Public):** /artifacts/{__app_id}/public/data/wishlists
- Documents will be added with an auto-generated ID.
- Each document will contain the following fields:
  - productName (string)
  - description (string)
  - imageUrl (string)
  - originalUrl (string)
  - isPublic (boolean, default false)
- **Real-time Updates:** The application will use Firestore's onSnapshot listener to automatically update the UI whenever a product is added, modified, or deleted.

4. Non-Functional Requirements

- **Performance:** The application should be responsive. The AI processing time should be indicated with a loading state, and the UI should not freeze.
- **Usability:** The interface should be intuitive for users with no technical background.
- **Security:** Data will be stored in Firestore, with security rules set to allow authenticated users to read and write their own data. The public wishlists will have rules allowing public read access.
- **Scalability:** The architecture using Firebase Authentication and Firestore is inherently scalable.
- **Accessibility:** The application should be accessible to users with disabilities, following standard web accessibility guidelines.

5. Technical Details

- **Frontend:** Single-file HTML/CSS/JS, React, or Angular app. (Developer's choice based on tool capabilities, but a single file is mandatory for this environment).
- **Backend/API:** All API calls will be made from the frontend to the Gemini API endpoint. Due to CORS, this will likely require a simple proxy or a platform that handles this, which is a common feature of AI development tools.
- **Database:** Firestore.

- **Authentication:** Firebase Authentication (Anonymous or Custom Token).
- **Global Variables:** The application must use the following global variables for configuration:
  - __app_id: string
  - __firebase_config: object
  - __initial_auth_token: string

6. Data Model

- **Firestore Collection:** wishlist
- **Document Structure:**
  - productName: string
  - description: string
  - imageUrl: string
  - originalUrl: string
  - timestamp: server timestamp (for ordering)

7. Acceptance Criteria

- [ ] A user can paste a valid URL and click "Add".
- [ ] A loading indicator appears while the AI processes the request.
- [ ] A new product card appears in the list with the extracted name, description, and image.
- [ ] The data is successfully saved to the user's private Firestore collection.
- [ ] A user can click a "Delete" button to remove a product from their list, and it is removed from Firestore.
- [ ] The user's userId is displayed on the screen.
- [ ] A user can enter another user's userId and view their public wishlist.
- [ ] The app responds gracefully to invalid URLs or network errors.

8. Go-to-Market

- The product will be launched as a proof-of-concept for internal use and a demonstration of AI integration.
- The focus is on functionality and proving the concept before considering a public release or further feature development.