Skincare Recommender Design Document

Project Name: Skincare Website

Table of Contents

1. Introduction

Purpose

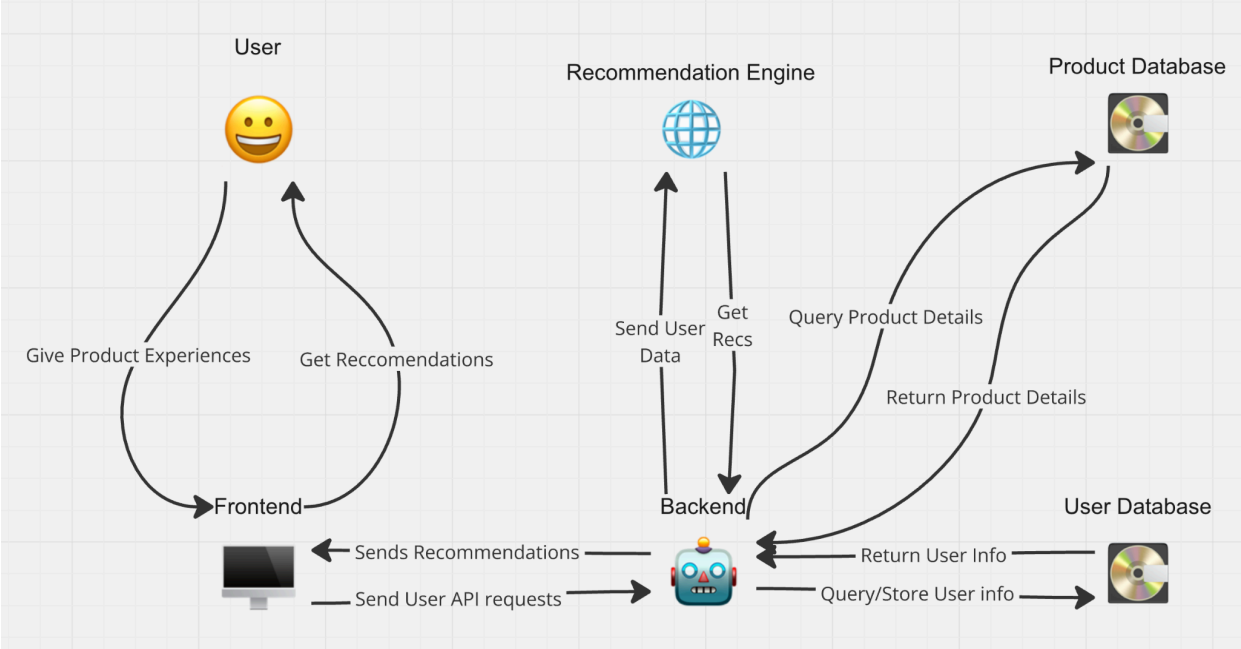
The purpose of this document is to outline the design specifications for the Skincare Website, which aims to provide personalized skincare product recommendations based on user profiles. It will serve as a guide for developers, designers, and stakeholders throughout the project lifecycle.

Scope

The Skincare Website will include features such as user account creation, personalized recommendations, ingredient filtering, product comparisons, and community engagement through reviews and comments. The project will not include e-commerce functionalities like direct product purchases..

2. Architecture Overview

System Architecture Diagram



Component Descriptions

Frontend:

React-based UI handles user interactions (registration, filtering, recommendations) and communicates with the backend for data retrieval and recommendations.

Backend:

Node.js/Express server processes frontend requests, interacts with the database, and calls the ML model API for recommendations. It formats requests, handles filtering, and manages database interactions.

User Database:

MongoDB stores user profiles, including skin type, concerns, preferences, and authentication details. The backend performs CRUD operations for user data.

Product Database:

MongoDB stores product data (name, category, ingredients, reviews). The backend handles filtering and querying for product information.

Recommendation Engine:

ML model (Flask/FastAPI) provides personalized product recommendations based on user data. The backend calls the API and returns the recommendations to the frontend.

File Structure:

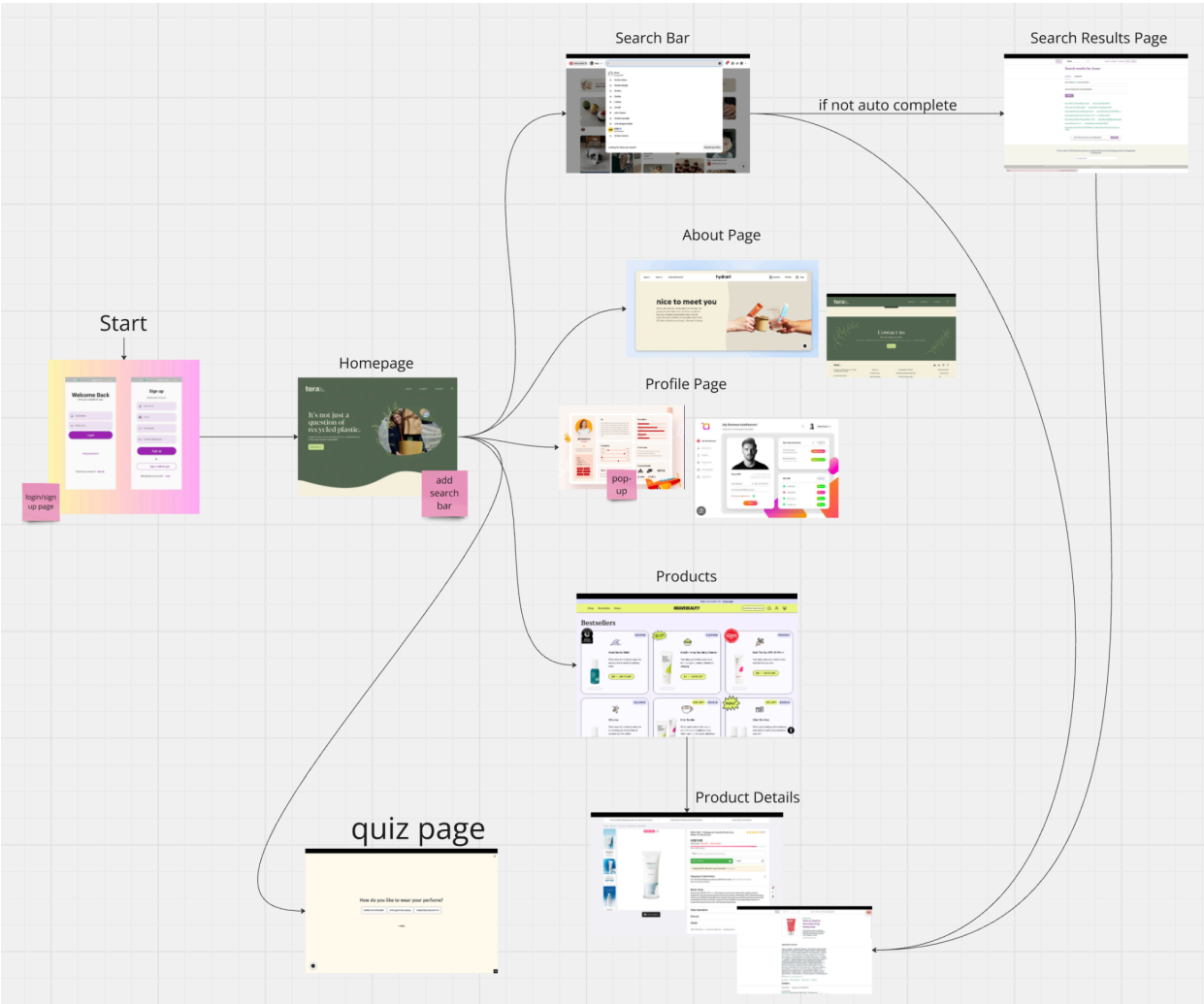
```
project-root/
├── frontend/               # React Frontend
│   ├── src/
│   │   ├── components/       # Reusable React components
│   │   ├── pages/            # Page components, each representing a route
│   │   ├── services/         # API calls to backend services
│   │   ├── hooks/            # Custom React hooks for reusable logic
│   │   ├── utils/            # Utility functions (e.g., formatters, validators)
│   │   ├── context/          # Context API for global state management
│   │   ├── App.js            # Main application component
│   │   └── index.js          # Entry point for React application
│   └── package.json          # React dependencies and project metadata
├── backend/                # Express/Node.js Backend
│   ├── controllers/          # Logic for handling incoming requests
│   ├── models/               # Database models/schemas (e.g., MongoDB models)
│   ├── routes/               # API route definitions
│   ├── services/             # Logic for communication with external services (e.g., databases,
ML service)
│   ├── middleware/           # Custom middleware for request processing
│   ├── config/               # Configuration settings (e.g., environment variables, database
configs)
│   ├── utils/                # Utility functions for backend (e.g., error handling, logging)
│   └── server.js             # Entry point for the backend server
│   └── package.json          # Node.js dependencies and project metadata
├── recommender/              # Machine Learning Model Service
│   ├── app/                  # Main folder for ML microservice
│   │   ├── model.py          # Python script to load and run the ML model
│   │   ├── service.py        # API endpoints for the ML service (e.g., using Flask or FastAPI)
│   │   ├── utils.py          # Data pre/post-processing utilities specific to ML
│   ├── models/               # Folder for storing trained models
│   │   └── model.pkl         # Pickled file of the trained ML model
│   ├── Dockerfile            # Dockerfile to containerize the ML service
│   ├── requirements.txt      # Python dependencies (e.g., FastAPI, numpy, scikit-learn)
│   └── app.py                # Main entry point to run the ML service
├── docker-compose.yml        # (Optional) To orchestrate Docker containers for backend and
ML service
└── README.md                 # Project documentation
```

3. User Interface Design

Wireframes/Mockups

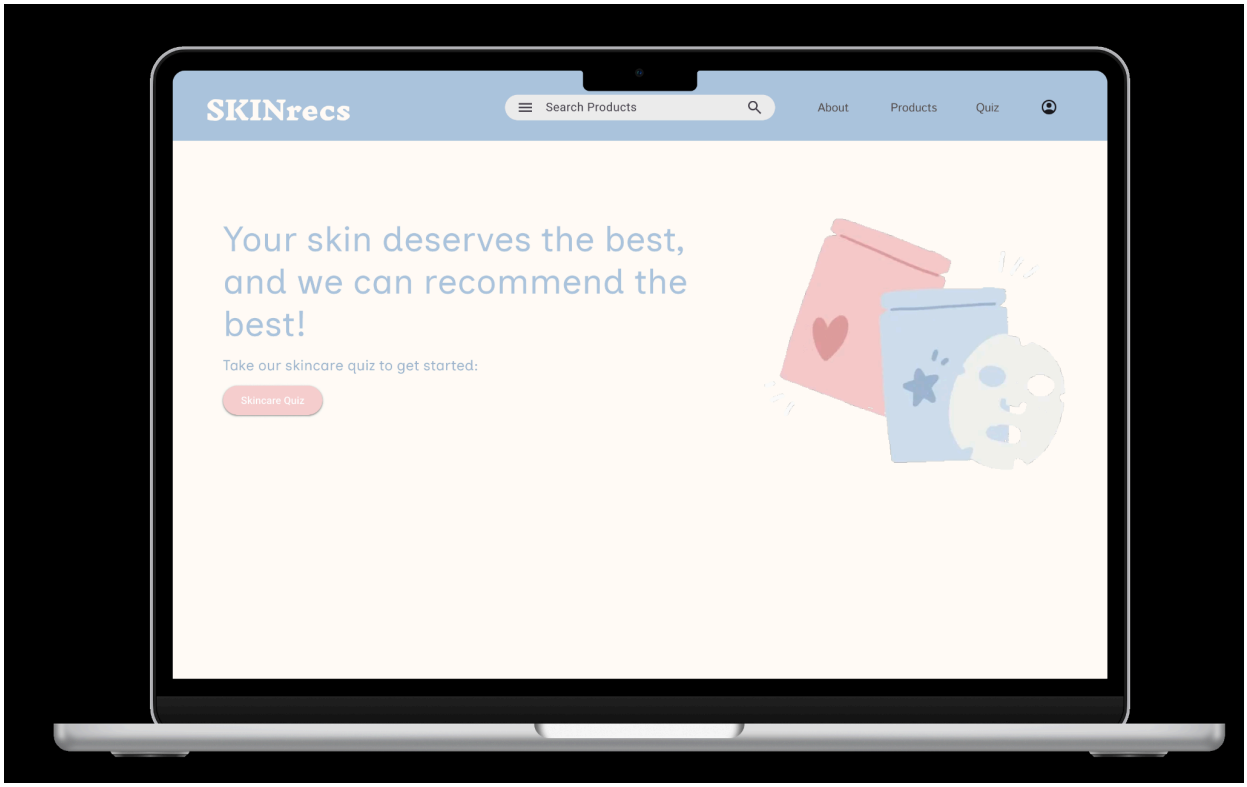
- **Home Page**: Design tbd.
- **Product Recommendation Page**: Lists recommended products with filter options.
- **Product Detail Page**: Shows detailed information, reviews, and comments.

User Flow Diagrams



Figma Designs

Home Page:



4. Database Design

Database Schema

Users Table:

- `userId` (ObjectId): Unique identifier for each user, automatically generated by MongoDB.
- `username` (String): Unique username for each user.
- `email` (String): User's email address, unique across all users.
- `password` (String): User's hashed password for security.
- `skinType` (Number): Represents the user's skin type using bitwise operations (e.g., oily, dry, combination).
- `concerns` (Number): Represents user's skin concerns as a bitwise number (e.g., acne, aging).

Products Table:

- `productId` (ObjectId): Unique identifier for each product.
- `name` (String): Name of the product.

- `price` (Number): Price of the product.
- `ingredients` (Array of Strings): List of ingredients used in the product.
- `category` (String): Category of the product for filtering (e.g., moisturizer, cleanser).

Reviews Table:

- `reviewId` (ObjectId): Unique identifier for each review.
- `userId` (ObjectId): Reference to the `Users` table.
- `productId` (ObjectId): Reference to the `Products` table.
- `rating` (Number): Product rating from 1 to 5.
- `comment` (String): Optional user comment about the product.

5. API Design

API Endpoints

- **POST /api/users/register**: Register a new user.
- **POST /api/users/login**: Authenticate a user.
- **GET /api/products/recommendations**: Get personalized product recommendations.
- **GET /api/products**: Fetch product details.
- **POST /api/products/review**: Submit a review for a product.

Request/Response Formats

- **POST /api/users/register**:

Request:

```
{
  "email": "user@example.com",
  "password": "password123",
}
```

  -

Response:

```
{
  "message": "User registered successfully.",
  "userId": "12345"
}
```