**Fake Store Project Documentation**

**Project Overview**

The Fake Store project is an e-commerce platform designed to showcase CRUD (Create, Read, Update, Delete) operations in a full-stack application. It integrates a PostgreSQL database, a Flask-based backend, and a ReactJS frontend to manage user authentication, product management, cart functionalities, and wishlist management.

**Technologies Used**

* **Frontend:** ReactJS
  + React components and pages structure
  + React Router for navigation
* **Backend:** Flask (Python 3)
  + RESTful API development
  + Data persistence and retrieval with SQLAlchemy
* **Database:** PostgreSQL
  + Tables: users, categories, products, carts, cart\_items, wishlist, wishlist\_items
* **Authentication:** Firebase (Initial Implementation)
  + Managed user authentication
  + Integrated with the users table in PostgreSQL
* **Deployment:**
  + Local server setup for demonstration

**System Architecture**

**Frontend**

* Components:
  + **Navbar:** Navigation and search query input.
  + **ProductList:** Displays available products fetched from the database.
  + **Cart:** Manages user-added cart items.
  + **Wishlist:** Manages user-added wishlist items.
* Pages:
  + **Home:** Landing page after login.
  + **Login/Signup:** Handles user authentication.
  + **Checkout:** Finalizes the purchase process.
  + **ProductDetails:** Displays details of individual products.

**Backend**

* Routes:
  + routes/products.py: Manages product-related operations.
  + routes/cart.py: Handles cart functionalities.
  + routes/wishlist.py: Handles wishlist functionalities.
  + routes/users.py: Manages user data and authentication integration.
* Models:
  + models/user.py: Defines the users table schema.
  + models/product.py: Defines the products table schema.
  + models/cart.py: Defines carts and cart\_items table schemas.
  + models/wishlist.py: Defines wishlist and wishlist\_items table schemas.

**Database Design**

**Database Schema**

* **Users Table:**
  + Fields: user\_id, firebase\_uid, email, name, created\_at
* **Products Table:**
  + Fields: product\_id, category\_id, name, price, description, stock
* **Cart and Cart Items Tables:**
  + carts: cart\_id, user\_id
  + cart\_items: item\_id, cart\_id, product\_id, quantity
* **Wishlist and Wishlist Items Tables:**
  + wishlist: wishlist\_id, user\_id
  + wishlist\_items: item\_id, wishlist\_id, product\_id
* **Categories Table:**
  + Fields: category\_id, name

**CRUD Operations**

**Create:**

* Adding a new product to the database via the admin console.
* Adding items to the cart or wishlist.

**Read:**

* Viewing products, cart items, and wishlist items from the database.

**Update:**

* Updating product details.
* Modifying the quantity of items in the cart.

**Delete:**

* Removing items from the cart or wishlist.
* Deleting products from the database.

**Key Features**

* **User Authentication:**
  + Login and signup functionalities backed by Firebase and PostgreSQL.
* **Product Management:**
  + Admin console for adding and updating products.
* **Cart and Wishlist:**
  + Add, view, update, and delete items from the cart and wishlist.
* **Checkout Process:**
  + Finalize and persist cart data.

**API Endpoints**

**User Endpoints**

* POST /api/users/login: Authenticate and fetch user data.
* GET /api/users/populate: Populate test user data.

**Product Endpoints**

* GET /api/products: Fetch all products.
* POST /api/products/add: Add a new product (Admin only).

**Cart Endpoints**

* GET /api/cart/get: Retrieve cart items for a user.
* POST /api/cart/add: Add items to the cart.

**Wishlist Endpoints**

* GET /api/wishlist/get: Retrieve wishlist items for a user.
* POST /api/wishlist/add: Add items to the wishlist.

**Project Workflow**

1. **Login/Signup:**
   * Users log in via Firebase.
   * User data is synced with the users table in PostgreSQL.
2. **Product Browsing:**
   * Products are fetched from the database and displayed.
3. **Adding Items:**
   * Users add items to the cart or wishlist.
4. **Checkout:**
   * Users proceed to checkout from the cart.
5. **Admin Console:**
   * Admin adds or updates products in the database.

**System Requirements**

1. **Persistence:**
   * Data such as user details, cart items, and wishlist items are saved into the database.
2. **Data Retrieval:**
   * Information such as available products, cart contents, and wishlist items is retrieved from the database and displayed.

**Conclusion**

This project demonstrates a comprehensive e-commerce application with complete CRUD operations and an integrated database. It showcases the power of PostgreSQL, Flask, and ReactJS in building robust and scalable web applications.