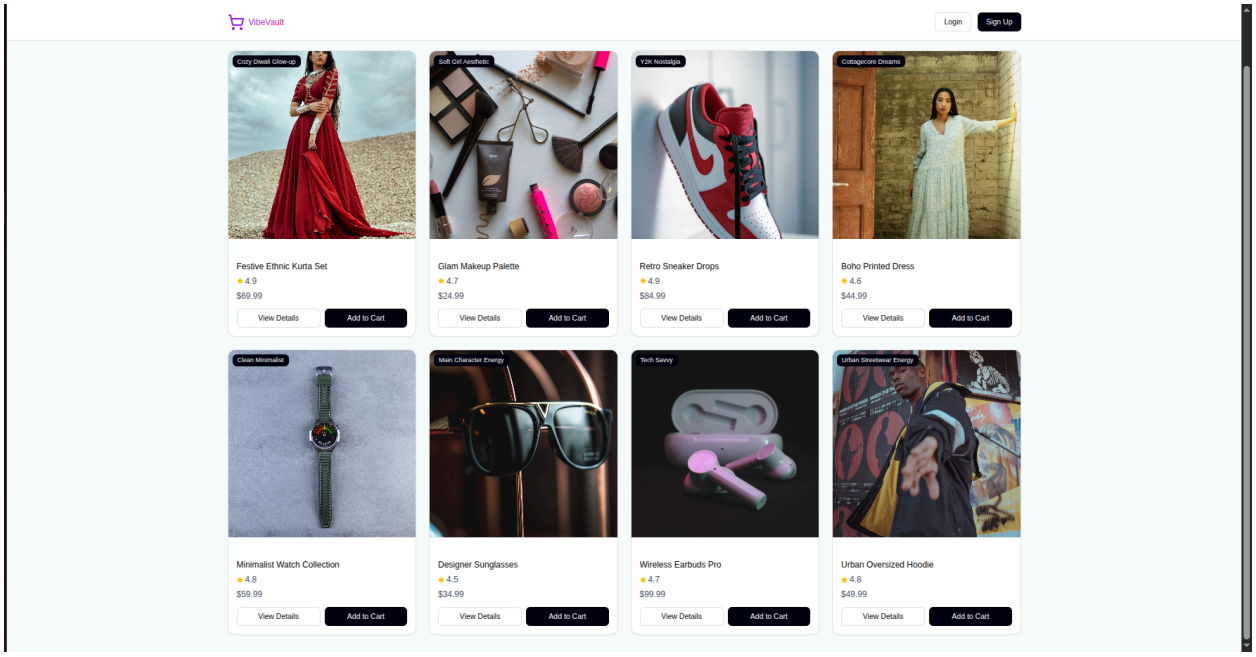
Project Walkthrough: VibeVault Full-Stack E-Commerce Application

Document Purpose

This document provides a functional walkthrough of the VibeVault e-commerce application. It details the user flow and connects the frontend user interface (UI) to the backend database structure that powers it, demonstrating a complete, end-to-end implementation of all core features.

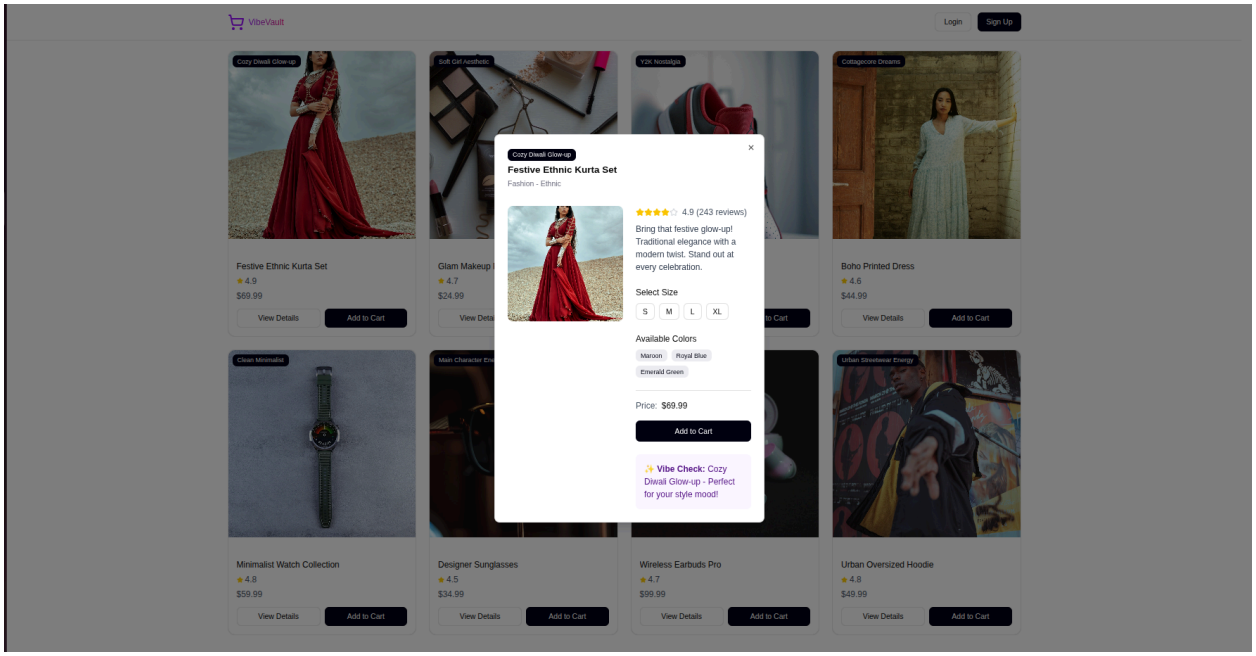
Part 1: Frontend User Experience & UI

Figure 1: Product Catalog (Logged-Out State)



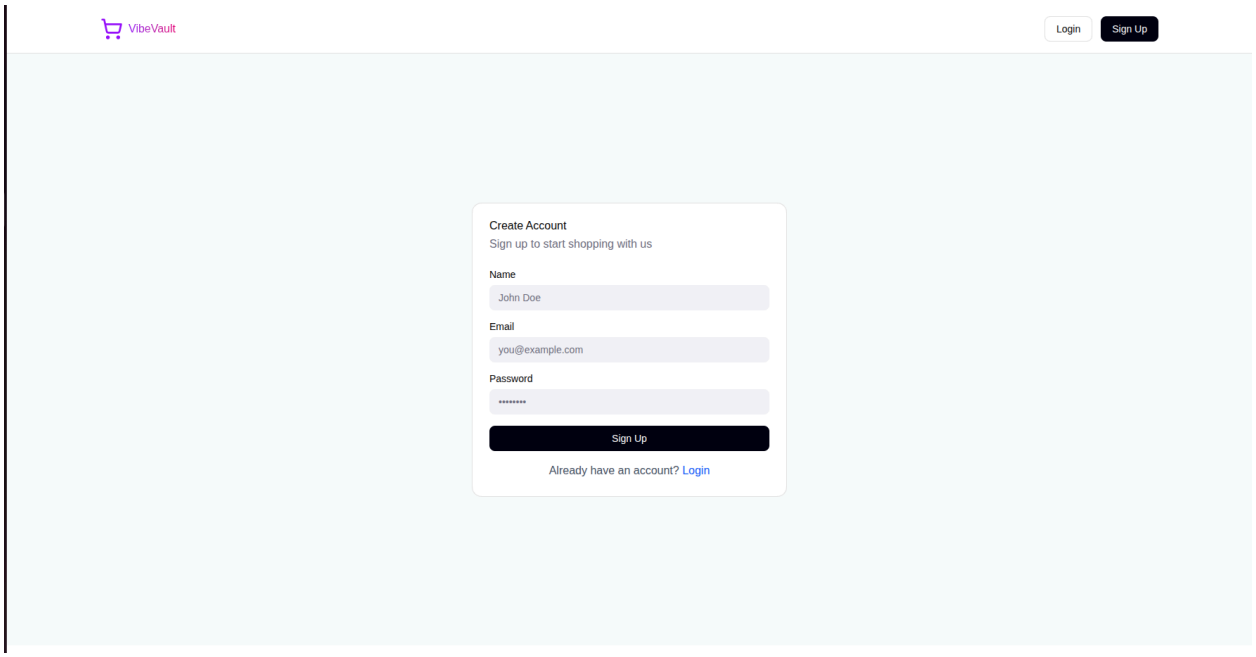
- **Description:** This is the main product catalog page and the primary landing view for new or logged-out users.
- **Functionality:**
  - **Product Display:** The grid dynamically fetches all product data (images, names, prices) from the backend via a GET /api/products API call.
  - **Authentication State:** The header correctly displays a "Login" and "Sign Up" button, indicating the application recognizes the user is not authenticated. The "Cart" icon is hidden, as cart functionality is reserved for logged-in users.
  - **User Actions:** From here, a user can browse all available items, click "View Details" to see more, or navigate to the authentication pages.

Figure 2: Product Detail Modal



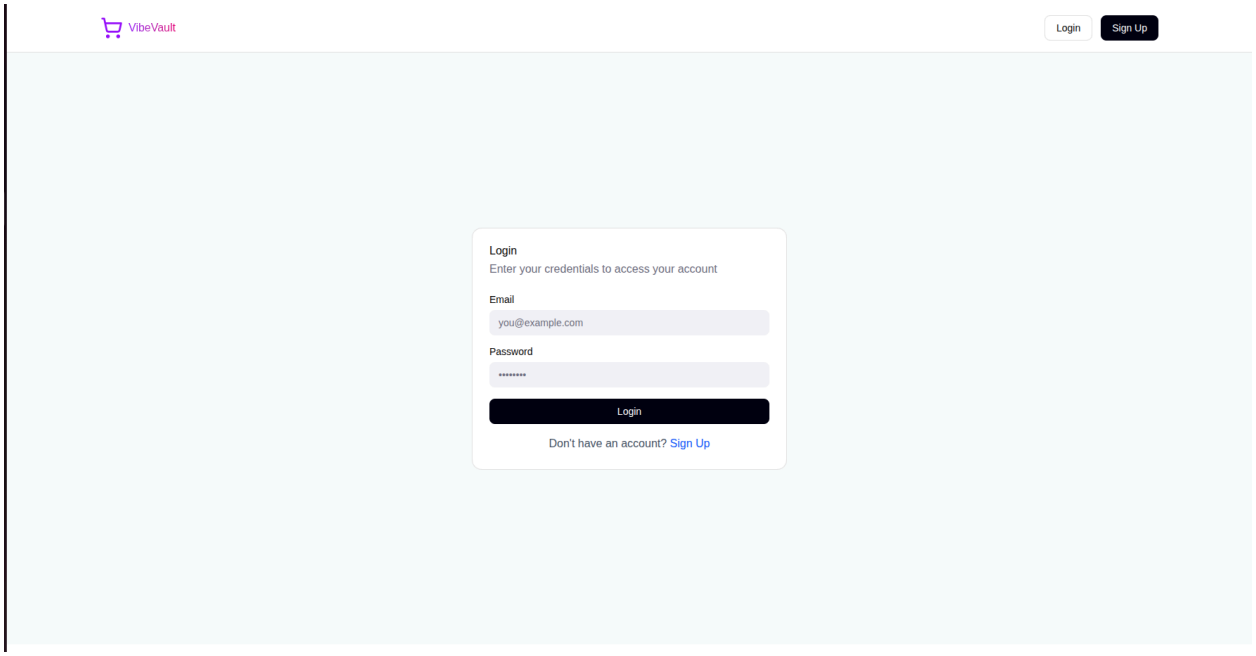
- **Description:** This modal appears when a user clicks the "View Details" button on any product card.
- **Functionality:**
  - **Detailed View:** Provides an expanded view with more images, a full description, and product-specific options like size and color.
  - **User Interaction:** This component demonstrates the ability to handle complex UI elements for product variations.
  - **Core Action:** The "Add to Cart" button is the primary call to action, which will trigger the cart logic (and an authentication check if the user is logged out).

Figure 3: User Registration Page



- **Description:** This is the secure account creation form.
- **Functionality:**
  - **New User Onboarding:** Captures a new user's Name, Email, and Password.
  - **API Integration:** On submission, this form sends a POST /api/auth/register request to the backend.
  - **Security:** The backend logic (as shown in Figure 5) hashes the password using bcryptjs before saving the new user to the database, ensuring no plain-text passwords are ever stored.

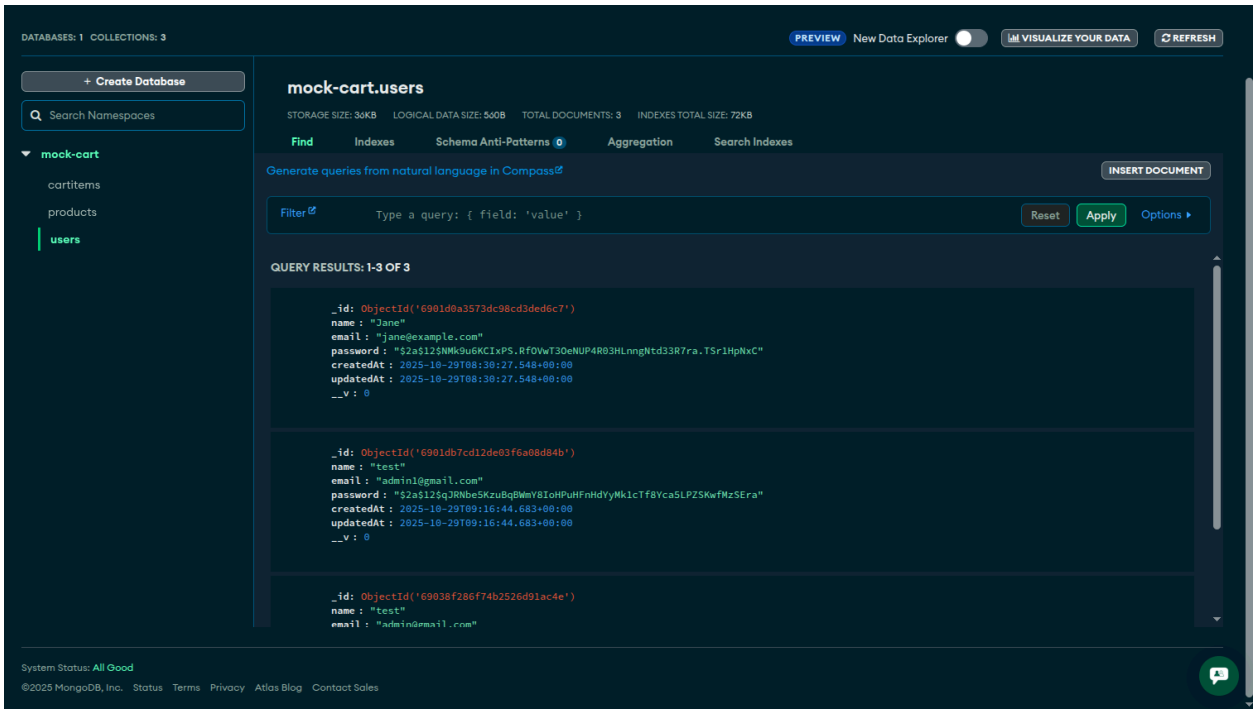
Figure 4: User Login Page



- **Description:** This is the secure login portal for existing users.
- **Functionality:**
  - **Authentication:** This form sends a POST /api/auth/login request with the user's Email and Password.
  - **Session Management:** The backend validates the credentials against the hashed password in the database. If successful, it generates and returns a JSON Web Token (JWT).
  - **Client-Side:** The frontend receives this JWT and stores it, using it to authenticate all future API requests and to conditionally render the "Logged-In" UI state (e.g., showing the cart icon).

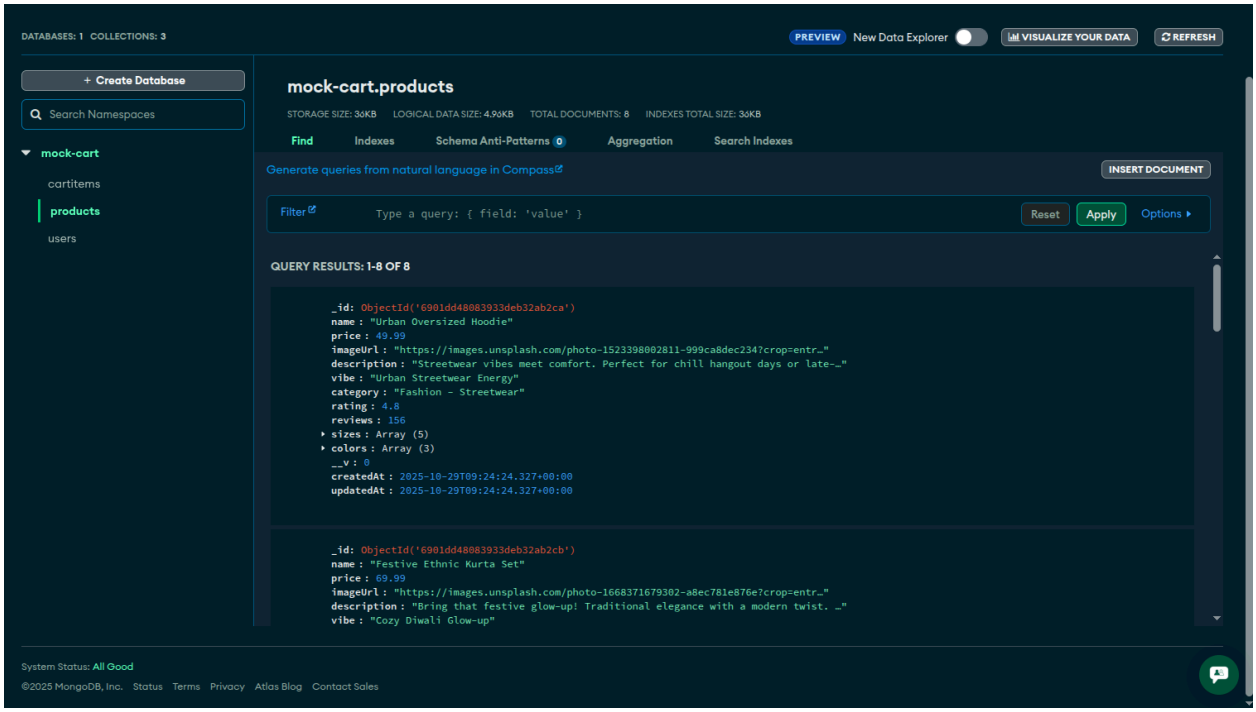
Part 2: Backend Database & Data Integrity

Figure 5: Backend - users Collection



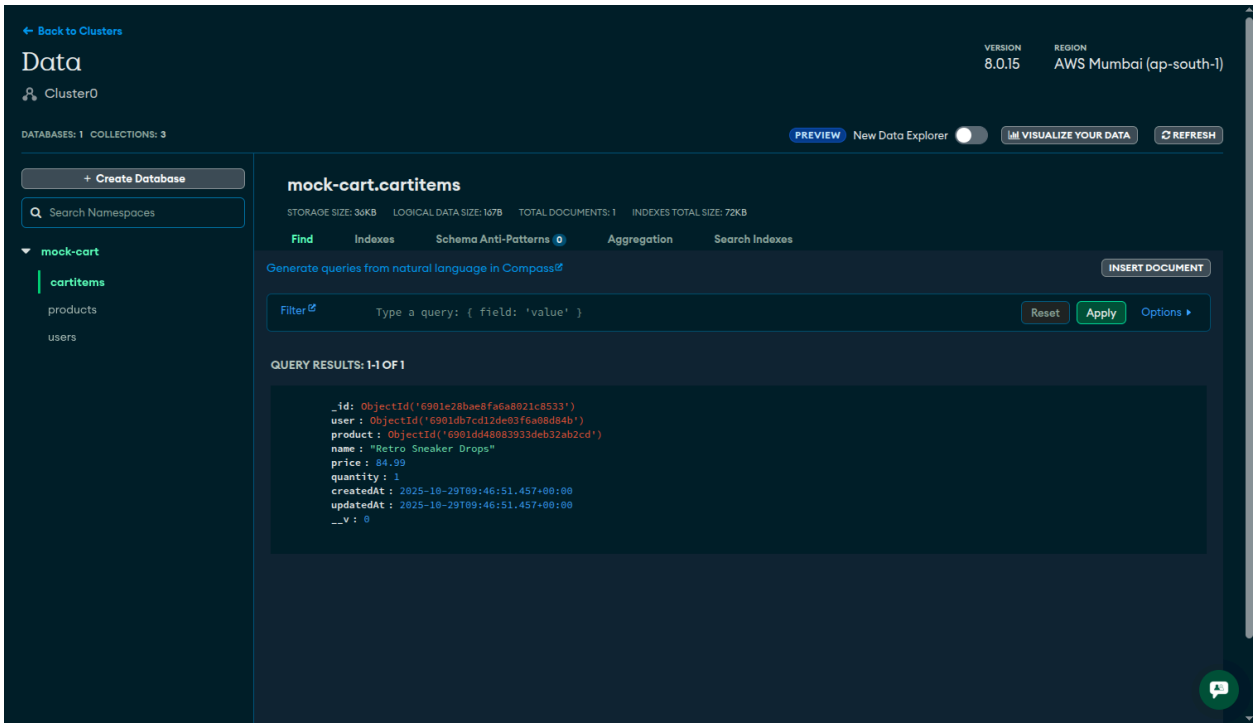
- **Description:** This screenshot from MongoDB Compass shows the users collection in the mock-cart database.
- **Functionality Proof:**
  - **Successful Registration:** This data is the direct result of a successful submission from the "Sign Up" page (Figure 3).
  - **Password Encryption:** The password field for each user contains a long, complex hash (e.g., "$2a$12...''). This confirms that bcryptjs is correctly implemented and that user credentials are secure.

Figure 6: Backend - products Collection



- **Description:** This screenshot shows the products collection, which acts as the master catalog for the entire store.
- **Functionality Proof:**
  - **Single Source of Truth:** This collection is the single source of truth for all product information.
  - **API-Driven Frontend:** The data seen here (e.g., "Urban Oversized Hoodie," "Festive Ethnic Kurta Set") is exactly what is fetched by the GET /api/products endpoint and rendered on the frontend in Figure 1.

Figure 7: Backend - cartitems Collection (Persistent Cart)



- **Description:** This screenshot shows a document within the cartitems collection and is the most critical piece of the application's core logic.
- **Functionality Proof:**
  - **Persistent Storage:** This collection stores the shopping carts for all users.
  - **User-Item Relationship:** This document demonstrates that an item (product: ...b3b2a) has been successfully added to the cart for a specific, authenticated user (user: ...a6d8d4b).
  - **Data Integrity:** This relational link via ObjectId is what enables the persistent cart. When this user logs out and logs back in, the application will query this collection for their user ID and immediately re-populate their cart with all items they had saved.