**Data Flow Diagram (DFD) for Dynamic Page App**

**Dynamic Page App** is a React-based application with routing, product listings, and dynamic pages for displaying watch details. Below is the **DFD representation**:

**Level 0 (Context Diagram)**

**At the highest level, the system consists of:**

1. **External Entity: User → Navigates between pages and interacts with the app.**
2. **Process: Dynamic Page System → Handles navigation, product display, and dynamic content.**
3. **Data Store: React State & API Backend → Stores product details, user selections, and API responses.**

+--------------------+

| External Entity: |

| User |

+--------------------+

|

v

+--------------------+

| Dynamic Page App | (Process 1.0)

+--------------------+

|

v

+--------------------+

| React State & API| (Stores Products, User Actions)

+--------------------+

**Level 1 DFD (Decomposed Processes)**

**Breaking down Process 1.0 (Dynamic Page App) into detailed steps:**

+--------------------+ +------------------------+

| External Entity: | | Process 1.1 - User |

| User | -----> | Clicks Navigation |

+--------------------+ +------------------------+

| |

v v

+--------------------+ +------------------------+

| Process 1.2 - | | Process 1.3 - Fetch |

| Route Matching | <-----> | (Fetch Product Data) |

| (React Router) | | (From API / State) |

+--------------------+ +------------------------+

| |

v v

+--------------------+ +------------------------+

| Process 1.4 - | | Process 1.5 - Render |

| Display Page | -----> | (Show Dynamic Page) |

| (Load Component) | | (Product Details) |

+--------------------+ +------------------------+

**Explanation of Level 1 DFD Processes**

1. **Process 1.1 - User Clicks Navigation**
   * The user selects a link (Home, About, Contact, Products) from the navbar.
2. **Process 1.2 - Route Matching (React Router)**
   * The system checks which route (/, /about, /products, /watch/:id) is selected.
   * It finds and matches the corresponding React Component.
3. **Process 1.3 - Fetch Product Data**
   * If the user visits /watch/:id, the app fetches product details from the API.
   * The fetched data is stored in React state (useState).
4. **Process 1.4 - Display Page**
   * The respective page component is dynamically rendered based on the selected route.
5. **Process 1.5 - Render Dynamic Page**
   * If a product is selected, the WatchDetails.js page is loaded with product details.
   * If an invalid route is accessed, the NotFound.js component is displayed.

**Data Flow Summary**

**✔** User navigates pages → React Router processes route → Fetches product data → Renders correct component  
**✔** Uses React State (useState & useEffect) for managing data. **✔** Backend API integration for fetching product details dynamically**.**