# Dynamic Page with React Router

### ****Level 0 - Context Diagram****

At the highest level, the system is represented as a single process interacting with external entities.

* **External Entity:** User
* **Process:** Dynamic Page Management System
* **Data Store:** Fetched API Data, React Router State

#### **Diagram Representation:**

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

| External Entity |

| User |

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

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

| Dynamic Page |

| Management System |

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

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

| Data Store: API |

| Fetched Data |

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

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

| Data Store: Router |

| Current State |

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

### ****Level 1 - Detailed Breakdown****

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

| External Entity | | External Entity |

| User | | (API Server) |

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

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

| Route | | Fetch |

| Selection (Click) | | API Data |

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

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

| Load | | Data Store: API |

| Component Based on | | Response Data |

| Route (useParams) | +----------------------+

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

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

| Render |

| Dynamic Page Content|

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

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

| Handle |

| Unknown Routes (404) |

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

1. **(Route Selection):** User clicks a navigation link, selecting a route (e.g., /post/1).
2. **(Load Component Based on Route):** React Router matches the route and loads the appropriate component using useParams.
3. **(Fetch API Data):** The application fetches dynamic content from an API based on the id parameter in the URL.
4. **(Render Dynamic Page Content):** The component displays the retrieved data dynamically.
5. **(Handle Unknown Routes):** If the user navigates to an undefined route, a 404 Not Found page is displayed.

The user selects a route → React Router updates the current route state.

The application loads the corresponding component → useParams extracts route parameters.

The system fetches data from an API → Data is stored and displayed.

If the route is invalid, the 404 component is shown.