Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the implementation details for **AsyncFetch - API Data Fetcher**, a React-based frontend application. The app fetches and displays data from a public API, demonstrating asynchronous JavaScript and API call handling, styled with Tailwind CSS.

---

1. System Components

Component	Description	Key Responsibilities
App	Root component	Initialize app, layout, routing
FetchData	Main data-fetching UI component	Trigger API calls, manage state
DataDisplay	Displays fetched data	Render data in UI
ErrorMessage	Shows error messages	Display errors to user
Loader	Shows loading indicator	Indicate loading state

---

2. Class/Interface Overview

Component/Class	Key Props/State	Key Methods/Handlers	Relationships
FetchData	data, loading, error	fetchData(), handleRetry()	Uses DataDisplay, ErrorMessage, Loader
DataDisplay	data	-	Child of FetchData
ErrorMessage	error, onRetry	-	Child of FetchData
Loader	-	-	Child of FetchData

Key Methods (FetchData):

```
function fetchData() {
  setLoading(true);
  fetch(API_URL)
    .then(res => res.json())
    .then(setData)
    .catch(setError)
    .finally(() => setLoading(false));
}
```

---

3. Data Structure Overview

Data Model	Structure / Fields	Example
API Response	Array of objects (id, title, body)	`[ { id: 1, title: "foo", body: "bar" } ]`
State	data, loading, error	`{ data: [], loading: false, error: null }`

---

4. Algorithms / Logic

Data Fetching Flow (pseudocode):

```
On component mount or retry:
  Set loading = true
  Fetch data from API
    If success:
      Set data, loading = false
    If error:
      Set error, loading = false
```

UI Rendering Logic:

```
If loading: show Loader
Else if error: show ErrorMessage
Else: show DataDisplay
```

---

5. Error Handling

Scenario	Handling Approach
Network/API failure	Set error state, show ErrorMessage
Invalid API response	Validate, set error if malformed
User retry	Retry fetchData() on user action

End of Document