# High Level Design Document

## Introduction

This High Level Design (HLD) document outlines the architecture and core design of **CloudOps - Technology Resource Manager UI**. The project delivers a modern, responsive frontend for managing cloud technology resources, featuring dashboards, resource cards, and interactive forms. The UI is built with React, advanced CSS, and Tailwind, emphasizing a futuristic, open-source theme.

---

## 1. System Architecture Overview

**Architecture Description:**
CloudOps UI is a single-page application (SPA) structured around modular React components. It interacts with backend APIs (not in scope) to fetch and manage cloud resources. The UI is styled using Tailwind CSS for responsiveness and modern aesthetics.

| Module/Component | Role/Responsibility |
|---|---|
| App Shell | Root component, routing, global state/context |
| Dashboard | Displays resource summaries, metrics, and quick actions |
| Resource Cards | Visualizes individual cloud resources |
| Resource Forms | Interactive forms for resource creation/editing |
| API Service Layer | Handles API requests and data transformation |
| UI Theme/Styling | Implements futuristic look via Tailwind and custom CSS |

---

## 2. Component Interactions

| Source Component | Target Component | Interaction Description |
|---|---|---|
| App Shell | Dashboard | Renders dashboard on main route |
| Dashboard | Resource Cards | Maps resource data to cards for display |
| Resource Cards | Resource Forms | Triggers form modal/dialog for edit/create actions |
| Resource Forms | API Service Layer | Submits form data to backend APIs |
| API Service Layer | All UI Components | Provides data via hooks/context for rendering |

**Sequence Flow Example:**
User navigates to dashboard → Dashboard fetches resource data via API Service → Resource Cards display data → User clicks “Edit” → Resource Form opens → On submit, API Service updates resource.

---

## 3. Data Flow Overview

| Data Source | Flow Direction | Data Consumer | Purpose |
|---|---|---|---|
| Backend API | → API Service Layer | UI Components | Fetch resource data |
| Resource Forms | → API Service Layer | Backend API | Create/update resource |
| API Service Layer | → Context/State | UI Components | Provide up-to-date resource info |

---

## 4. Technology Stack

| Layer/Area | Technology/Framework |
|---|---|
| UI Framework | React (Functional Components) |
| Styling | Tailwind CSS, Custom CSS |
| Language | JavaScript (ES6+), HTML5 |
| State Mgmt | React Context/State, Hooks |
| API Handling | Fetch/Axios (RESTful APIs) |
| Tooling | npm, Webpack/Vite |

---

## 5. Scalability & Reliability

- **Component Modularity:** UI is decomposed into reusable, independent components for maintainability and scalability.
- **Responsiveness:** Tailwind ensures adaptive layouts for all device sizes.
- **Error Handling:** API Service Layer centralizes error management for robust user feedback.
- **Open Source Readiness:** Codebase structured for community contributions and extensibility.
- **Security:** Follows best practices for frontend security (input validation, minimal data exposure).

---

**End of Document**