High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **StyleSwitch - CSS Theme Toggler**, a web application enabling users to switch between multiple CSS themes (light, dark, custom) using React, JavaScript, and Tailwind CSS. The project demonstrates dynamic theme switching via CSS selectors and variables.

---

1. System Architecture Overview

Architecture Description:
StyleSwitch is a single-page React application. The UI is rendered client-side, with theme state managed in React and styles applied using Tailwind CSS and CSS variables.

Module/Component	Role/Responsibility
ThemeProvider	Manages current theme state and provides context
ThemeToggle UI	User interface for selecting and toggling themes
App Component	Main container, integrates ThemeProvider and UI
Tailwind/CSS Layer	Defines theme styles using Tailwind and CSS variables

---

2. Component Interactions

Source Component	Target Component	Interaction Description
ThemeToggle UI	ThemeProvider	Sends user-selected theme change requests
ThemeProvider	App Component	Supplies current theme context to all child components
App Component	Tailwind/CSS	Applies theme classes/variables to DOM

Sequence Flow:

1. User selects a theme via ThemeToggle UI.
2. ThemeProvider updates theme state and context.
3. App Component applies new theme styles using Tailwind/CSS variables.

---

3. Data Flow Overview

Data	Source	Destination	Purpose
Theme Selection	User (UI)	ThemeProvider	Update current theme state
Theme State	ThemeProvider	App/Child Components	Render UI with correct theme

CSS Variables/Classes	ThemeProvider/App	Tailwind/CSS Layer	Apply theme-specific styles

---

4. Technology Stack

Layer/Functionality	Technology/Framework
Frontend Framework	React
Language	JavaScript (ES6+)
Styling	Tailwind CSS, CSS Variables
State Management	React Context API
Build Tooling	Vite or Create React App

---

5. Scalability & Reliability

- **Scalability:**
  The app is client-side and stateless; it can be deployed as static assets and scales horizontally via CDN.
- **Reliability:**
  Minimal dependencies and no backend reduce failure points. Theme state is managed in-memory; for persistence, localStorage can be added.
- **Security:**
  No sensitive data is handled. Standard React and Tailwind practices mitigate common vulnerabilities.

End of Document