High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **Formify - Dynamic Form Builder**. Formify is a web application enabling users to create, preview, and manage dynamic forms, leveraging React, JavaScript, and Tailwind CSS. The design focuses on modularity, usability, and maintainability.

---

1. System Architecture Overview

Architecture Description:
Formify is a single-page application (SPA) built with React. The system is organized into modular components for form creation, preview, and management, with a centralized state for form data.

Module/Component	Role/Responsibility
UI Layer	Renders the interface using React and Tailwind CSS
Form Builder Module	Allows users to add, edit, and remove form fields
Form Preview Module	Displays a live preview of the current form
Form Management Module	Handles saving, loading, and deleting forms
State Management	Manages form schema and user interactions

---

2. Component Interactions

Source Component	Target Component	Interaction Description
Form Builder	State Management	Updates form schema on user actions
State Management	Form Preview	Provides current form schema for rendering
Form Management	State Management	Loads/saves form schemas to/from storage
UI Layer	All Modules	Routes user actions to appropriate modules

Sequence Flow:

1. User interacts with Form Builder to modify form fields.
2. State Management updates the form schema.
3. Form Preview reflects changes in real-time.
4. Form Management enables saving/loading forms.

---

3. Data Flow Overview

Data Source	Data Destination	Data Type/Description

User Input	Form Builder	Field definitions, labels, types, options
Form Builder	State Management	Updated form schema (JSON)
State Management	Form Preview	Current form schema for rendering
Form Management	Local Storage/API	Persisted form schemas

---

4. Technology Stack

Layer/Area	Technology/Framework
Frontend Framework	React
Language	JavaScript (ES6+)
Styling	Tailwind CSS
State Management	React Context/State
Storage	Browser Local Storage
Build Tooling	Vite or Create React App

---

5. Scalability & Reliability

- **Scalability:**
  The modular React component structure supports easy extension (e.g., new field types). Local storage is suitable for single-user or demo use; for multi-user or persistent storage, integration with a backend API is recommended.

- **Reliability:**
  Stateless UI components and centralized state management ensure predictable behavior. Input validation and error handling should be implemented for robust form creation.

- **Security:**
  As a client-side app, sensitive data should not be stored. For production, consider authentication and secure storage if backend integration is added.

End of Document