Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the implementation details for **Formify - Dynamic Form Builder**, a web application for creating, previewing, and managing dynamic forms. The project leverages React, JavaScript, and Tailwind CSS to demonstrate modern form handling and UI practices.

---

1. System Components

Component	Description	Key Responsibilities
FormBuilder	Main UI for form creation	Add/edit/remove form fields
FormPreview	Renders live preview of the form	Display current form configuration
FormManager	Handles form state and persistence	Save/load forms, manage form list
FieldEditor	UI for editing individual field properties	Configure label, type, validation
FieldRenderer	Renders individual form fields	Render input based on field type

---

2. Class/Interface Overview

Name	Type	Description	Key Methods/Props
`FormBuilder`	Component	Main form builder UI	`addField()`, `removeField()`, `onFieldEdit()`
`FormPreview`	Component	Live form preview	`fields` (prop), `onSubmit()`
`FormManager`	Module	Form state & persistence	`saveForm()`, `loadForms()`, `deleteForm()`
`FieldEditor`	Component	Edit field properties	`field` (prop), `onChange()`
`FieldRenderer`	Component	Render field by type	`field` (prop)

Relationships:

- `FormBuilder` uses `FieldEditor` and `FormPreview`.
- `FormManager` provides data to `FormBuilder` and `FormPreview`.

---

3. Data Structure Overview

Model	Fields
Form	`id: string`, `name: string`, `fields: Field[]`

Field	`id: string`, `label: string`, `type: string`, `required: boolean`, `options?: string[]`, `validation?: object`

---

4. Algorithms / Logic

Form Submission Flow (Pseudocode):

```
onFormSubmit(formData):
  for each field in form.fields:
    if field.required and !formData[field.id]:
      showError(field.label + " is required")
      return
    if field.validation and !validate(formData[field.id], field.validation):
      showError(field.label + " is invalid")
      return
  saveFormData(formData)
  showSuccess("Form submitted")
```

Field Addition:

```
addField(type):
  newField = { id: uuid(), label: "", type, required: false }
  form.fields.push(newField)
```

---

5. Error Handling

Scenario	Handling Approach
Invalid field input	Show inline error message
Missing required field	Prevent submission, highlight field
Form save/load failure	Show notification, retry option
Unsupported field type	Fallback to text input, log warning
Duplicate field IDs	Regenerate ID, prevent collision

End of Document