Team 33

LocalNotes
Initial Architecture Design

Version <1.0>

Team Members

Name	**GitHub Account**
Abdulaziz Ali	Abdulaziz-Ali123
Subat Qutlan	Shaun-Was-Taken
Atharva Patil	a157p624
Malek Kchaou	MK720-dev
Matthew Nash	m518n748
Wesley McDougal	wesmcdougal

Table of Contents

Initial Architecture Design

1. Introduction

1.1 Purpose

This document provides the initial architectural overview of the **LocalNotes** system. It describes the structure, design rationale, and implementation strategies for the development of user-friendly, and visually appealing note-taking app.

1.2 Scope

The **LocalNotes** application covers:

- **Frontend Architecture:** Developed with **Next.js** and **React**, providing a responsive and fluid user interface.
- **Electron Integration:** Enables cross-platform desktop deployment (Windows, macOS, Linux).
- **Core Features:** Markdown editing with live preview, LaTeX rendering for mathematical notation, freehand drawing support, and local file storage for privacy.
- **UI Design:** Built with either custom components or libraries such as **shadcn/ui** and **Magic UI**, focusing on modern, minimalistic aesthetics.
- **Development Guidelines:** Establishes standards for code structure, modularity, and maintainability.

This provides the foundation for a consistent and scalable foundation for the **LocalNotes** system.

1.3 Definitions, Acronyms, and Abbreviations

This section provides a glossary of terms and acronyms used in the document:

Term	Definition
Markdown	A lightweight markup language for creating formatted text using a plain-text editor.
LaTeX	A typesetting system used for rendering mathematical equations and scientific notations.
Electron	A framework that enables building cross-platform desktop applications using web technologies like HTML, CSS, and JavaScript.
Next.js	A React-based framework providing server-side rendering, static site generation, and optimized build processes.
TypeScript	A strongly typed superset of JavaScript that enhances code reliability and scalability.

UI (User Interface)	The visual elements and interactive components through which users interact with the application.
Local Storage	The method of storing user data directly on the device to ensure privacy and offline access.

1.4 References

Electron Documentation
Source: https://www.electronjs.org

GitHub
Source: https://github.com/Abdulaziz-Ali123/LocalNotes

ReactJs Documentation
Source: https://react.dev/reference/react

TailwindCss Documentation
Source: hhttps://tailwindcss.com/docs/installation/using-vite

1.5 Overview

This Software Architecture Document is organized as follows:

- **Section 2: Architectural Description** – Outlines the components and composition representing the system.
- **Section 3: Use Case Diagrams** – Outlines the models and diagrams representing the system

The document provides a top-down analysis of the architecture to ensure comprehensive understanding and effective use.

2. Architecture Description

2.1 Overview

LocalNotes is a cross-platform desktop application for note-taking, built using Electron, Next.js, React, and TypeScript. It enables users to write and manage notes with Markdown, LaTeX, and drawing capabilities, which will all be stored locally on the user's device to maximize privacy and offline accessibility.

The system is structured into three major architectural layers:

1. Main Process (Backend Core) – Electron-based controller managing app lifecycle, file I/O, and inter-process communication (IPC).
2. Renderer Process (Frontend UI) – React/Next.js interface responsible for user interactions, rendering, and editing logic.
3. Helpers and Utilities – Shared TypeScript modules supporting both processes, handling tasks like Markdown parsing, LaTeX rendering, and file management.

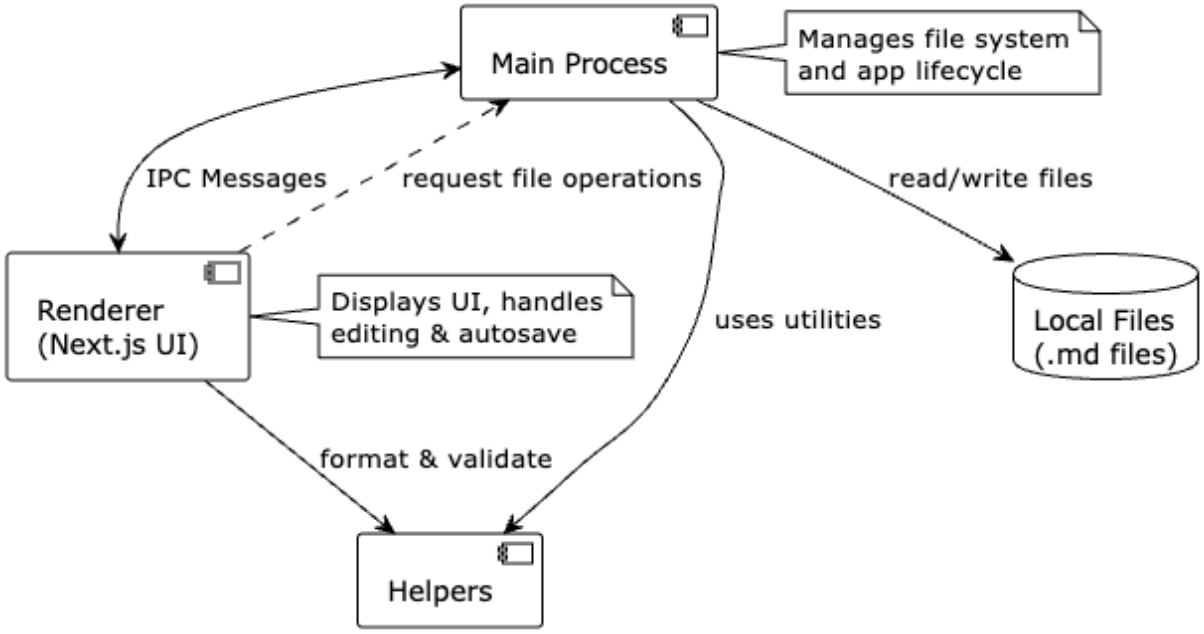
This modular approach ensures separation of concerns, enhances maintainability, and supports scalability for future extensions such as syncing, encryption, or collaboration.

2.2 Component Breakdown

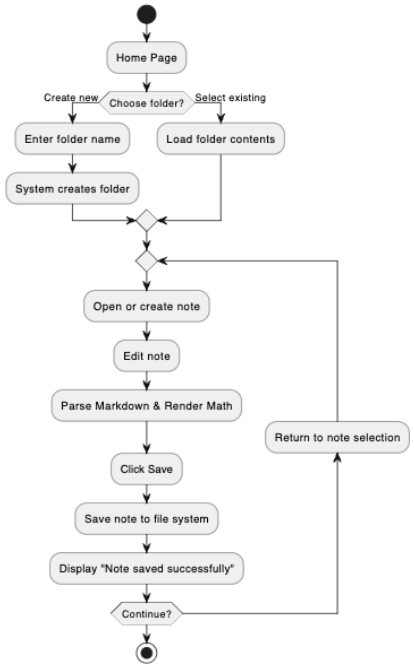
Component	Description
main/background.ts	Electron's main entry point. Creates application windows, handles app lifecycle events (launch, minimize, quit), and manages IPC between the UI and system-level operations.
main/preload.ts	Defines secure bridges for the renderer to call system-level functions (e.g., reading/writing files) using Electron's contextBridge API.
main/helpers/	Contains utility functions for file system access, directory creation, settings management, and local caching.
renderer/	Hosts the React/Next.js frontend responsible for UI rendering and user interactions.
renderer/components/	Contains reusable React components (note editor, toolbar, file explorer, drawing canvas).
renderer/pages/	Defines Next.js routes (e.g., /, /editor, /settings).
resources/	Stores icons, build assets, and packaging files used by Electron Builder for cross-platform deployment.

3. Use Case Diagrams

3.1 Component Diagram



3.2 User Flow Diagram



3.3 Save File Flow Diagram