PROJECT ABSTRACT

JSON Formatter and Validator Desktop App

An offline desktop JSON formatter and validator with pretty-print, minify, precise line-column error reporting and a collapsible tree explorer — delivered built-to-order with full source, report, PPT and viva kit.

01. Title

JSON Formatter and Validator Desktop App.

02. Introduction / Background

JSON is the working language of modern software: APIs return it, configurations are written in it, and developers read it all day. A single missing comma can waste twenty minutes because server responses arrive as one unreadable line and most error messages say little more than “unexpected token”. This project builds a focused desktop utility covering the whole loop: instant pretty-printing with syntax highlighting, one-click minification, validation that pinpoints the exact line and column of the first syntax error, and a collapsible tree view for deeply nested payloads — all fully offline, since pasted JSON often contains production keys and customer data.

03. Problem Statement

Web-based formatters require pasting potentially sensitive API responses into a third-party site, and their error messages rarely locate the problem precisely. Developers need a private, offline tool that formats, validates with exact error locations, and lets them explore large payloads structurally.

04. Objectives

- Pretty-print JSON with syntax highlighting and minify to a single line.
- Validate strictly and report the exact line and column of the first syntax error.
- Render large payloads as a collapsible, type-colored tree.
- Show live document statistics: keys, max depth, array count, size.
- Run 100% offline as an Electron desktop app on Windows, macOS and Linux.

05. Proposed Methodology

The app uses the native JSON.parse/JSON.stringify engine for spec-compliant parsing. On each keystroke the input is parsed; the character count and VALID/INVALID badge update live. Formatting re-serializes with indentation and applies regex-based syntax highlighting. Validation failures convert the parser's character offset into line/column coordinates. The tree view recursively renders the parsed object as collapsible DOM nodes with type-colored values; a single walk computes the statistics panel.

06. System Architecture / Working

Two-pane layout: input editor (textarea with live parse) and output pane (formatted JSON, error panel, or tree view). A toolbar exposes Format, Minify, Validate, Tree and Clear. The status bar shows character count and the validity badge; a side panel shows document statistics and feature documentation. The Electron shell adds file open/save and native menus.

07. Hardware / Software Requirements

- Desktop/laptop running Windows, macOS or Linux.
- Node.js 18+ and Electron for building; packaged app has no runtime dependencies.
- No network connection required.
- Any machine with 2 GB RAM handles multi-MB documents in text mode.

08. Expected Outcomes

- A working offline formatter/validator demonstrably faster than web tools for the edit-fix loop.
- Precise error locations verified against crafted broken payloads in the report.
- Complete source, user manual, report, PPT and viva Q&A.;

09. Applications

- API developers debugging responses during integration.
- DevOps engineers validating configuration files before deployment.
- Students learning the JSON data model and parser behavior.
- Anyone handling JSON containing secrets that must not leave the machine.

10. Future Scope

- JSON Schema-based semantic validation.
- Virtualized tree rendering for hundred-thousand-node documents.
- JSON ↔ YAML/XML conversion.