**《软件工程》**

**项目需求说明书**

项目名称 支持Markdown实时预览的文本编辑器设计与实现

学 院 大数据与智能信息工程学院

专 业 软件工程

学 生 孙智健

校 号 2401127

指导教师 曹秀平

目 录

1. 项目背景及研究的目的和意义 1

1.1项目背景 1

1.2研究目的和意义 1

2. 国内外在该方向的研究现状及分析 1

2.1国内研究现状及分析 1

2.2国外研究现状及分析 1

3. 研究内容 2

3.1 核心功能 2

3.2 增强功能 2

3.3 非功能性需求 3

4. 拟采取的研究方法和技术路线 3

4.1 研究方法 3

4.2 技术路线与系统架构 3

5. 进度安排、预期达到的目标 4

5.1 进度安排 4

5.2 预期达到的目标 4

6. 项目已具备和所需的条件 5

7. 研究过程中可能遇到的困难和问题，解决的措施 5

8. 参考文献 6

1. 项目背景及研究的目的和意义

1.1项目背景

随着信息化时代的深入，Markdown作为一种轻量级标记语言，因其简洁的语法和强大的内容表达能力，在文档撰写、笔记记录、博客写作和技术文档编写等领域获得了广泛应用。然而，传统的Markdown编辑方式通常需要用户在纯文本编辑器中进行写作，然后在另一个预览窗口或工具中查看渲染后的效果。这种“编写-预览”的割裂体验严重影响了创作流畅度和效率。因此，开发一款具备实时预览功能的Markdown编辑器，将编辑与预览过程无缝融合，具有显著的实际需求和市场价值。

1.2研究目的和意义

本项目的核心目的是设计并实现一个用户友好、功能完善的支持Markdown实时预览的文本编辑器。旨在通过技术创新，解决用户在使用Markdown时面临的核心痛点，提升写作体验和效率。

1. 国内外在该方向的研究现状及分析

2.1国内研究现状及分析

国内在该领域已有不少优秀产品。例如：

Typora：作为一款广受好评的国产软件，它以其“无缝实时预览”的理念重新定义了Markdown编辑体验，用户可在同一界面下直接看到渲染后的样式。

语雀、有道云笔记：这些在线文档平台内置的Markdown编辑器功能强大，支持实时预览，并与云存储、协作等功能深度集成。

此外还有开源组件如mavonEditor等，为国内开发者提供了在Web项目中快速集成Markdown编辑能力的解决方案。

国内产品更注重用户体验的流畅性和功能的集成度。Typora的成功证明了市场对优秀单机版编辑器的需求。然而，许多在线产品受制于网络环境，且功能繁杂，而独立软件则可能存在定制化程度不高的问题。

2.2国外研究现状及分析

国内市场中，诸如“有道云笔记”、“语雀”、“Typora”等产品都集成了优秀的Markdown编辑器。特别是Typora，以其“无缝实时预览”的设计理念获得了大量用户的青睐。这些产品通常功能丰富，集成度高，但部分高级功能需要付费，且作为大型应用的一部分，可能不够轻量级和灵活。国内的开源社区也有如 mavonEditor 等优秀的Vue组件，主要服务于Web开发者，作为插件集成到其他系统中。

国外在该领域起步更早，生态成熟。例如：

Visual Studio Code：其内置的Markdown支持非常强大，通过插件可实现极其丰富的预览和编辑功能。

StackEdit：一款功能全面的在线Markdown编辑器。

开源库：如 Marked.js, Showdown.js 等解析库，为开发者构建自定义编辑器提供了强大的底层支持。 国外的产品和开源库在性能、可扩展性和生态建设上具有明显优势，是本项目重要的技术参考和学习对象。

分析结论：尽管市场存在成熟产品，但开发一款轻量级、高性能、界面简洁、专注于核心实时预览体验的独立编辑器，仍然具有其独特的价值和生存空间，尤其适合作为技术验证和定制化开发的起点。

1. 研究内容

3.1 核心功能

Markdown文本编辑：

提供基本的文本输入、编辑、复制、粘贴、撤销、重做功能。

支持标准的Markdown语法（如标题、列表、代码块、表格、链接、图片等）。

实时预览：

编辑器界面分为左右（或上下）两个面板：编辑区和预览区。

用户在编辑区输入内容时，预览区需即时（延迟低于500ms）渲染出格式化后的效果。

支持同步滚动，即编辑区和预览区的滚动位置保持基本同步。

语法高亮：

在编辑区内，对Markdown语法关键词（如 #, -, “```” 等）进行高亮显示，提升代码可读性。

对代码块中的编程语言进行语法高亮。

3.2 增强功能

工具栏支持：

提供图形化工具栏，用户可通过点击按钮快速插入常用Markdown标记（如加粗、斜体、插入链接等）。

文件操作：

支持新建、打开（.md, .txt 文件）、保存Markdown文件。

主题切换：

提供浅色与深色两种界面主题，供用户根据环境和个人喜好切换。

导出功能：

支持将编辑好的Markdown内容导出为HTML文件或PDF文件。

3.3 非功能性需求

性能需求：

响应时间：实时预览的响应延迟应低于500毫秒。

资源占用：应用启动速度快，在常规配置的台式机和笔记本电脑上运行流畅。

可用性需求：

界面设计简洁、直观，符合主流操作习惯，用户无需培训即可上手使用。

提供清晰的操作反馈。

兼容性需求：

目标平台为Windows 10/11和macOS（近期版本）操作系统。

（Web版）兼容Chrome, Firefox, Edge等主流浏览器的近期版本。

1. 拟采取的研究方法和技术路线

4.1 研究方法

文献研究法：深入研究Markdown语法规范，学习现有优秀编辑器的交互设计。

原型设计法：首先使用设计工具（如Figma）绘制软件界面的高保真原型，明确UI/UX设计。

敏捷开发法：采用迭代式开发，优先实现核心功能（编辑、预览），再逐步添加增强功能（工具栏、文件操作等），每个周期都进行测试和验证。

测试法：进行单元测试、集成测试和用户验收测试，确保软件质量和功能稳定性。

4.2 技术路线与系统架构

本项目拟采用Electron + React + Node.js的技术栈。

Electron: 用于构建跨平台的桌面应用程序。它允许使用Web技术（HTML, CSS, JavaScript）来开发桌面应用，完美契合本项目需求。

React: 作为前端UI框架，用于构建用户界面。其组件化特性有利于代码的复用和维护，特别适合构建编辑器、预览面板这样的复杂交互界面。

Node.js: 作为后端运行时，用于处理文件系统的读写操作（新建、打开、保存文件）。

核心实现方案：

界面布局：使用React组件构建一个分栏布局，左侧为编辑区（textarea 或使用 CodeMirror / Monaco Editor 等专业编辑器组件以获得更好的编码体验），右侧为预览区（一个 div 容器）。

Markdown解析与渲染：

使用高效的JavaScript Markdown解析库，如 Marked.js 或 Unified 生态系统。

在编辑区的 onChange 事件中，获取输入的Markdown文本，调用解析库将其转换为HTML字符串。

将生成的HTML字符串插入到预览区的 div 中，完成实时渲染。

语法高亮：在预览区，使用 highlight.js 库对代码块进行语法高亮处理。

文件操作：通过Electron主进程模块（如 dialog, fs）调用操作系统原生文件对话框，实现文件的打开与保存。

导出功能：利用解析库将Markdown转换为HTML，再结合其他库（如 puppeteer）将HTML转换为PDF。

1. 进度安排、预期达到的目标

5.1 进度安排

| 阶段 | 时间周期 | 主要任务 | 交付物 |
| --- | --- | --- | --- |
| 第一阶段 | 第1-2周 | 需求分析与技术选型，环境搭建 | 需求说明书，开发环境 |
| 第二阶段 | 第3-5周 | 实现核心功能：编辑区、预览区、实时同步 | 具备实时预览功能的原型 |
| 第三阶段 | 第6-8周 | 实现增强功能：工具栏、文件操作、主题切换 | 功能完整的Alpha版本 |
| 第四阶段 | 第9-10周 | 实现导出功能，进行系统测试与优化 | 可用的Beta版本 |
| 第五阶段 | 第11-12周 | 修复Bug，完善文档，打包发布 | 最终Release版本 |

5.2 预期达到的目标

交付一个功能完整的桌面应用程序：具备上述所有核心与增强功能，可在Windows和macOS上稳定运行。

提供优秀的用户体验：实时预览响应迅速，界面美观易用。

产出完整的技术文档：包括系统设计文档、用户使用手册和源代码注释。

完成项目总结报告：详细记录开发过程、遇到的问题及解决方案。

1. 项目已具备和所需的条件

已具备条件：

技术基础：具备JavaScript、HTML、CSS以及React框架的开发经验。

开发环境：个人电脑，可安装必要的开发工具（VS Code, Git, Node.js等）。

参考资料：有丰富的在线技术文档、开源项目和社区论坛作为支持。

所需条件：

深入的Electron框架学习与实践时间。

对Markdown解析库和语法高亮库的选型与集成需要时间调试。

跨平台打包与分发需要一定的配置和学习。

1. 研究过程中可能遇到的困难和问题，解决的措施

在处理长篇文档时，频繁的Markdown解析与DOM（文档对象模型）更新可能会引发界面卡顿，严重影响用户体验。为应对此挑战，本研究采取了多项优化措施。首先，选用性能卓越的解析库（如Marked.js）以提升基础解析效率。其次，对键盘输入事件实施防抖处理，仅在用户输入暂停后才触发渲染，有效避免了不必要的频繁更新。最后，针对超长文档，引入虚拟滚动技术，仅渲染可视区域内的内容，从而显著降低了内存占用和渲染负担，保证了界面的流畅性。由于编辑器中的原始文本与预览面板中渲染后的HTML内容高度通常不一致，实现二者之间精确、流畅的滚动同步是一项关键技术难点。本研究摒弃了简单的像素级映射方案，转而设计并实现了一种基于滚动比例的同步算法。该算法通过实时计算两个面板各自的滚动位置与总高度的百分比，来同步对方的滚动进度，确保了内容对应关系的准确性。此外，我们还借鉴了多个成熟开源项目的实现思路，对算法进行了优化与验证。基于Electron框架开发的应用需在Windows、macOS等不同操作系统上运行，而各平台在文件路径规范、系统快捷键、原生窗口样式等方面存在固有差异，这些差异可能导致应用行为不一致或功能异常。为解决此问题，我们充分利用了Electron提供的跨平台API，以统一的方式处理底层系统调用。同时，针对平台特有的功能（如菜单栏快捷键），通过编写条件代码进行适配。在开发流程中，我们建立了严格的跨平台测试机制，确保在主流操作系统上进行充分的兼容性测试，及时发现并修复平台相关问题。Markdown允许嵌入HTML，直接渲染用户输入的内容存在跨站脚本（XSS）注入的安全隐患，攻击者可能利用此漏洞执行恶意脚本。本应用将安全性置于重要位置。我们采取了两层防护策略：其一，对Markdown解析库生成的HTML输出进行严格的转义处理，确保所有用户输入均被视为纯文本显示。其二，在解析器配置中，默认禁用内联HTML标签以及对javascript:等危险URL协议的支持，从源头上杜绝了潜在的安全威胁。

1. 参考文献
2. 王继成,高珍.软件需求分析的研究[J].计算机工程与设计,2002,(08):18-21.DOI:10.16208/j.issn1000-7024.2002.08.005.
3. 宋洪英.M软件开发项目需求分析风险管理研究[D].山东大学,2020.DOI:10.27272/d.cnki.gshdu.2020.001468.
4. 孟亚辉.浅谈软件项目开发过程中的需求分析[J].科技信息,2009,(11):435-436.DOI:CNKI:SUN:KJXX.0.2009-11-338.
5. 张宏升.软件架构的非功能性需求指标和区域化支持[J].电脑知识与技术,2011,7(09):2085-2086+2089.DOI:CNKI:SUN:DNZS.0.2011-09-052.
6. 杨波,吴际,徐珞,等.一种软件测试需求建模及测试用例生成方法[J].计算机学报,2014,37(03):522-538.DOI:CNKI:SUN:JSJX.0.2014-03-004.
7. 杨长春.软件需求分析实战[M].北京:清华大学出版社,2020.
8. 杨芙清,何新贵.软件工程进展[M].北京:清华大学出版社,1996.
9. 冯玉琳.软件工程[M].合肥:中国科学技术出版社,1992.
10. 朱三元,钱乐秋,宿为民.软件工程技术概论[M].北京:科学出版社,2002.
11. 郑人杰,殷人昆,陶永乐.使用软件工程[M].北京:清华大学出版社,1997.
12. 吕云翔.软件工程实用教程[M].北京:清华大学出版社,2015.
13. 冯冲,江贺,冯静芳.软件体系结构理论与实践[M].北京:人民邮电出版社,2004.
14. 于卫,杨万海,蔡希尧.软件体系结构的描述方法研究[J].计算机研究与发展,2000(10).
15. 孙力群.基于模式系统软件体系结构的质量分析[D].合肥:合肥工业大学,2005.
16. 杨文龙,姚淑珍,吴芸.软件工程[M].北京:电子工业出版社,1999.
17. Filman R E.面向方面的软件开发[M].莫倩,译.北京:机械工业出版社,2006.
18. 张欣.软件项目开发过程中的需求分析[J].信息与电脑,2016(18).
19. Pressman R S.Software Engineering:A Practitioner's Approach[M].New York:McGraw-Hill,Inc.,1992.
20. Jeffrey L.Whitten,Lonnie D.Bentley.系统分析与设计方法(第七版)[M].北京:机械工业出版社,2012.