# Markdown编辑器FlowyNote详细设计说明书

## 1. 引言

### 1.1 目的

本详细设计说明书旨在为Markdown编辑器FlowyNote提供详细的设计细节，包括系统架构、模块设计、类设计、接口设计、数据库设计等。通过本说明书，开发团队可以清晰地了解每个模块的具体实现细节，确保项目的顺利开发和实现。

### 1.2 目标用户

* **开发团队**：负责项目的开发和实现。
* **测试团队**：负责项目的测试和验证。
* **产品经理**：负责项目的整体规划和需求管理。

### 1.3 项目范围

本项目将开发一个跨平台的Markdown编辑器FlowyNote，使用Qt框架实现，支持Windows、Linux和Mac OS操作系统。编辑器将提供基本的Markdown语法支持，并具备实时预览、自定义界面、插件扩展等功能。

## 2. 系统架构

### 2.1 总体架构

FlowyNote采用分层架构设计，主要分为以下几个层次：

* **用户界面层**：使用Qt框架实现，提供编辑、预览、分栏三种模式，支持自定义缩放、拖动、标题栏、快捷菜单和背景样式。
* **业务逻辑层**：负责处理用户输入、Markdown语法解析、文件管理、插件管理等业务逻辑。
* **数据访问层**：使用SQLite3作为数据库管理系统，存储用户信息、笔记内容、标签、分类等数据。
* **日志层**：使用Log4cpp记录系统日志，方便调试和问题排查。

### 2.2 技术选型

* **开发框架**：Qt
* **Markdown解析引擎**：Google V8 Engine + Markd.js
* **数据库**：SQLite3
* **日志系统**：Log4cpp
* **许可证**：GPL (GNU GENERAL PUBLIC LICENSE)

## 3. 模块设计

### 3.1 用户界面模块

#### 3.1.1 编辑区

* **类名**：MarkdownEditor
* **功能**：提供Markdown文本编辑功能，支持语法高亮。
* **接口**：
  + void setText(const QString& text)：设置编辑器内容。
  + QString getText() const：获取编辑器内容。
  + void setSyntaxHighlighting(bool enable)：启用或禁用语法高亮。

#### 3.1.2 预览区

* **类名**：MarkdownPreview
* **功能**：实时显示Markdown文本的渲染效果。
* **接口**：
  + void setHtml(const QString& html)：设置预览区内容。
  + QString getHtml() const：获取预览区内容。
  + void setSyncScroll(bool enable)：启用或禁用同步滚动。

#### 3.1.3 分栏模式

* **类名**：SplitView
* **功能**：支持编辑区和预览区分栏显示。
* **接口**：
  + void setSplitMode(SplitMode mode)：设置分栏模式（水平、垂直、无分栏）。
  + SplitMode getSplitMode() const：获取当前分栏模式。
  + void setSplitRatio(double ratio)：设置分栏比例。

#### 3.1.4 自定义界面

* **类名**：CustomUI
* **功能**：支持自定义缩放、拖动、标题栏、快捷菜单和背景样式。
* **接口**：
  + void setZoomLevel(double level)：设置缩放级别。
  + double getZoomLevel() const：获取当前缩放级别。
  + void setDragEnabled(bool enable)：启用或禁用拖动功能。
  + void setTitleBar(const QString& title)：设置标题栏内容。
  + QString getTitleBar() const：获取标题栏内容。
  + void setBackgroundColor(const QColor& color)：设置背景颜色。

### 3.2 业务逻辑模块

#### 3.2.1 Markdown解析

* **类名**：MarkdownParser
* **功能**：使用Google V8 Engine + Markd.js解析Markdown语法。
* **接口**：
  + QString parse(const QString& markdown)：解析Markdown文本并返回HTML。
  + void setParserOptions(const QMap<QString, QVariant>& options)：设置解析器选项。

#### 3.2.2 文件管理

* **类名**：FileManager
* **功能**：支持新建、打开、保存、另存为Markdown文件。
* **接口**：
  + bool newFile(const QString& filePath)：新建文件。
  + bool openFile(const QString& filePath)：打开文件。
  + bool saveFile(const QString& filePath)：保存文件。
  + bool saveAsFile(const QString& filePath)：另存为文件。

#### 3.2.3 插件管理

* **类名**：PluginManager
* **功能**：支持第三方插件的安装和使用，扩展编辑器功能。
* **接口**：
  + bool installPlugin(const QString& pluginPath)：安装插件。
  + bool uninstallPlugin(const QString& pluginName)：卸载插件。
  + QList<QString> getInstalledPlugins() const：获取已安装插件列表。

### 3.3 数据访问模块

#### 3.3.1 用户信息管理

* **类名**：UserManager
* **功能**：存储和管理用户信息。
* **接口**：
  + bool addUser(const User& user)：添加用户。
  + bool updateUser(const User& user)：更新用户信息。
  + bool deleteUser(const QString& userId)：删除用户。
  + User getUser(const QString& userId) const：获取用户信息。

#### 3.3.2 笔记管理

* **类名**：NoteManager
* **功能**：存储和管理笔记内容、标签、分类等数据。
* **接口**：
  + bool addNote(const Note& note)：添加笔记。
  + bool updateNote(const Note& note)：更新笔记内容。
  + bool deleteNote(const QString& noteId)：删除笔记。
  + Note getNote(const QString& noteId) const：获取笔记内容。

### 3.4 日志模块

#### 3.4.1 日志记录

* **类名**：Logger
* **功能**：使用Log4cpp记录系统日志，方便调试和问题排查。
* **接口**：
  + void log(LogLevel level, const QString& message)：记录日志。
  + void setLogLevel(LogLevel level)：设置日志级别。

## 4. 数据库设计

### 4.1 表设计

#### 4.1.1 用户表 (users)

| 字段名 |
| --- |
| user\_id |
| username |
| password |
| email |
| created\_at |

#### 4.1.2 笔记表 (notes)

| 字段名 |
| --- |
| note\_id |
| user\_id |
| title |
| content |
| created\_at |
| updated\_at |

#### 4.1.3 标签表 (tags)

| 字段名 |
| --- |
| tag\_id |
| tag\_name |

#### 4.1.4 笔记标签关联表 (note\_tags)

| 字段名 |
| --- |
| note\_id |
| tag\_id |
| PRIMARY KEY (note\_id, tag\_id) |

#### 4.1.5 分类表 (categories)

| 字段名 |
| --- |
| category\_id |
| category\_name |

#### 4.1.6 笔记分类关联表 (note\_categories)

| 字段名 |
| --- |
| note\_id |
| category\_id |
| PRIMARY KEY (note\_id, category\_id) |

### 4.2 关系模型

* **用户表 (users)** 与 **笔记表 (notes)** 之间是一对多的关系，一个用户可以拥有多篇笔记。
* **笔记表 (notes)** 与 **标签表 (tags)** 之间是多对多的关系，通过 **笔记标签关联表 (note\_tags)** 进行关联。
* **笔记表 (notes)** 与 **分类表 (categories)** 之间是多对多的关系，通过 **笔记分类关联表 (note\_categories)** 进行关联。

## 5. 技术实现

### 5.1 用户界面

* **Qt框架**：使用Qt框架实现跨平台的图形用户界面。
* **自定义控件**：实现自定义缩放、拖动、标题栏、快捷菜单和背景样式。

### 5.2 Markdown解析

* **Google V8 Engine**：使用Google V8 Engine作为JavaScript引擎，动态链接库形式调用。
* **Markd.js**：使用Markd.js解析Markdown语法，效率高。

### 5.3 数据库

* **SQLite3**：使用SQLite3作为数据库管理系统，存储用户信息、笔记内容、标签、分类等数据。

### 5.4 日志系统

* **Log4cpp**：使用Log4cpp记录系统日志，方便调试和问题排查。

## 6. 附录

### 6.1 术语表

* **Markdown**：一种轻量级标记语言，用于格式化文本。
* **实时预览**：在编辑器中实时显示Markdown文本的渲染效果。
* **插件**：扩展编辑器功能的独立模块。

### 6.2 参考资料

* [Qt框架](https://www.qt.io/)
* [Google V8 Engine](https://v8.dev/)