|  |  |
| --- | --- |
| 一、项目概述  答案之书 | 1.一款单机运行的开源软件，为用户提供随机建议、灵感或答案。用户可以通过输入问题或关键词，从内置的答复库中获取随机答复。软件允许用户自定义答复库和界面样式，提供本地存储功能以保存历史记录。软件的设计和功能将更专注于本地操作和用户体验。  2.一款为用户提供随机建议、灵感或答案的开源软件。用户可以通过输入问题或关键词，获得由算法生成的随机回复或引用。软件可以集成多种数据源，如名人名言、文学作品、用户自定义的答复等。软件的设计和功能将更专注于个性化设计和实时互动。 |
| 二、功能需求 | 问题输入：用户可以通过界面输入问题或关键词。  答复生成：根据用户输入，从内置的答复库中随机选择一个答复。  自定义答复库：允许用户添加、编辑或删除答复库中的条目。  历史记录：记录用户的问题和对应的答复，保存在本地数据库中。  自定义设置：允许用户自定义界面样式、字体等。 |
| 三、技术实现 | 前端技术：使用适合桌面应用开发的框架，如Electron（结合Web技术如HTML、CSS、JavaScript）或本地GUI框架（如Qt、wxWidgets等），以构建跨平台的桌面应用。  数据存储：使用SQLite等轻量级数据库或JSON文件等本地存储方案，保存用户数据、答复库和历史记录。  用户界面：设计直观易用的用户界面，包括输入框、答复展示、历史记录查看、设置等部分。 |
| 四、项目结构 | 前端：  入口文件（如main.js或main.cpp，取决于所使用的框架）  组件库（包括输入框、答复展示、历史记录、设置等组件）  样式文件（CSS、Sass或Less等）  静态资源（如图片、图标等）  答复库文件（如SQLite数据库文件或JSON文件）  数据存储：  用户数据表（存储用户设置等）  答复库表（存储答复数据）  历史记录表（存储用户的问题和答复） |
| 五、开发计划 | 需求分析：明确软件的功能需求和技术实现方式。  原型设计：设计软件的界面原型和交互流程。  前端开发：根据原型设计实现前端界面，并集成答复生成、历史记录查看、设置等功能。  数据存储：设计并实现本地数据存储方案，用于保存用户数据、答复库和历史记录。  测试：对软件进行功能测试、性能测试和用户体验测试。  打包发布：将软件打包成可执行文件，并发布到开源平台（如GitHub）或软件分发网站。  后期维护：定期更新软件、修复漏洞和添加新功能（如果需要）。 |
| 六、开源协作 | 代码托管：将项目代码托管在GitHub等开源平台上，方便参与协作。  贡献指南：制定贡献指南，明确代码风格、提交流程等要求。  问题跟踪：使用GitHub的Issue功能跟踪问题和需求。  文档编写：编写项目文档，包括用户手册、开发者指南等。  社区支持：通过GitHub的讨论区或邮件列表等方式，提供社区支持和用户反馈渠道。 |

|  |  |
| --- | --- |
| 1. 初始化Git仓库 | 在开始编写代码之前，需要初始化一个Git仓库。  git init |
| 2. 添加文件和提交更改 | 开始编写代码、设计界面、添加资源文件等时，Git追踪这些更改。一旦完成了某个功能或修复了一个问题，可以将这些更改添加到Git仓库中，并提交一个说明性的提交信息。  # 添加所有更改到暂存区  git add .  # 提交更改，并附带提交信息  git commit -m "Add initial user interface and question input functionality" |
| 3. 分支管理 | 为每个功能创建一个新的分支，并在该分支上开发该功能，而不影响主分支的代码。  # 创建一个新分支用于开发新功能  git checkout -b feature-login  # 在feature-login分支上进行开发...  # 完成开发后，将更改合并回主分支  git checkout main  git merge feature-login  # 删除已合并的分支（可选）  git branch -d feature-login |
| 4. 协作开发 | 开发者可以在自己的分支上开发功能，然后将更改推送到远程仓库。其他开发者可以拉取这些更改，并在自己的本地仓库中合并它们。  # 将本地更改推送到远程仓库  git push origin feature-login  # 从远程仓库拉取最新更改  git pull origin main |
| 5. 代码审查 | 创建一个Pull Request来请求审查代码更改，允许讨论、审查和批准更改。 |
| 6. 发布版本 | 使用Git的标签功能来标记这个版本。标签通常用于标记重要的里程碑，如软件的发布版本。  # 创建一个新的标签来标记发布版本  git tag -a v1.0.0 -m "Release version 1.0.0"  # 将标签推送到远程仓库  git push origin v1.0.0 |
| 7. 持续集成/持续部署 (CI/CD) | 可以使用Git与CI/CD工具（如Jenkins、GitLab CI等）结合使用，以实现自动化构建、测试和部署。这可以确保在每次代码更改时都进行必要的验证，并自动将更改部署到生产环境（如果需要）。 |