# 漫画阅读平台

成员1：王品寰

成员2：肖家庆

成员3：任宁凯

**一、项目背景**

本项目主题聚焦于漫画阅读器的开发，灵感与参考主要来源于bilibili漫画这一知名平台。在项目启动之初，我们三人团队面临着一定的挑战，因为我们在漫画阅读器开发这一领域并没有很好的技术基础。

然而，正是基于这样的背景，我们更加明确了学习与创新的重要性。为了弥补技术上的不足，我们决定深入研究和参考两个在漫画阅读器领域具有显著影响力的平台：Bilibili漫画和腾讯动漫。通过仔细分析这两个平台的用户界面、功能设计、交互体验以及技术实现，我们逐渐积累了漫画阅读器开发的相关知识，并形成了自己的开发思路。

在项目开发过程中，我们充分发挥了各自的优势，分工合作，共同攻克了一个又一个技术难题。从需求分析、设计规划到编码实现、测试优化，每一个环节都凝聚了我们的智慧和汗水。最终，我们成功完成了这个项目，开发出了一个功能完善、用户体验良好的漫画阅读器。

通过这样的项目经历，我们不仅提升了自身的技术能力，还学会了如何在没有很好基础的情况下，通过学习和参考优秀案例来快速推进项目。未来，我们将继续探索和实践，为开发出更多优秀的软件产品而努力。

1. **市场分析**

当前市场上已存在多款漫画阅读应用，但它们在功能、用户体验和漫画资源方面仍存在诸多不足。例如，部分应用漫画资源有限，无法满足用户的多样化需求；部分应用在用户界面和交互设计方面存在缺陷，导致用户阅读体验不佳；还有一些应用在性能和稳定性方面存在问题，影响了用户的正常使用。因此，开发一款具有竞争力的漫画阅读器具有重要意义。

# 三、用户需求

通过市场调研和用户访谈，我们了解到用户对漫画阅读器的需求主要包括以下几个方面：

1.丰富的漫画资源：用户希望漫画阅读器能够提供大量的漫画作品，涵盖各种类型和题材。

2.良好的阅读体验：用户希望漫画阅读器能够提供良好的阅读界面和交互体验，如翻页效果、缩放功能、夜间模式等。

3.个性化推荐：用户希望漫画阅读器能够根据他们的阅读历史和偏好，为他们推荐相关的漫画作品。

4.离线下载功能：用户希望能够在有网络的情况下下载漫画，以便在无网络环境下进行阅读。

5.社区互动：用户希望漫画阅读器能够提供一个社区平台，让他们能够与其他漫画爱好者交流心得、分享作品。

1. 团队和角色

**1）任宁凯**（前端开发工程师）

负责漫画阅读器的前端开发工作，包括用户界面（UI）的创建、交互逻辑的实现以及与后端服务的对接，具体任务如下：

1.HTML/CSS/JavaScript开发：使用HTML构建页面结构，CSS进行样式设计，JavaScript实现页面交互。

2.UI组件开发：开发漫画列表、漫画详情、阅读页面等UI组件。

3.响应式设计：确保漫画阅读器在不同设备和屏幕尺寸上都能提供良好的用户体验。

4.前端性能优化：优化页面加载速度、减少资源消耗，提高用户体验。

5.与后端对接：通过API接口与后端服务进行数据交互，实现用户登录、漫画搜索、下载等功能。

**2）王品寰**（后端开发工程师）

负责漫画阅读器的后端开发工作，包括服务器搭建、数据库设计、API接口开发以及数据处理。

1.服务器搭建与配置：选择合适的服务器环境，进行搭建和配置。

2.数据库设计与优化：设计数据库结构，存储漫画信息、用户信息等数据，并进行性能优化。

3.API接口开发：开发用户认证、漫画搜索、漫画详情获取等API接口。

4.数据处理与缓存：处理用户请求，从数据库中获取数据并返回给前端；使用缓存技术提高数据访问速度。

5.安全性与稳定性：确保后端服务的安全性，防止数据泄露和攻击；监控服务器状态，确保服务的稳定性。

**3）肖家庆**（项目管理与UI/UX设计）

负责项目的整体管理、进度跟踪以及UI/UX设计工作，确保项目按时完成并符合用户需求。

具体任务：

1.项目规划与进度管理：制定项目计划，分配任务，跟踪进度，确保项目按时完成。

2.需求分析与文档编写：与用户沟通，收集需求，编写需求文档；与团队成员沟通，确保需求准确传达。

3.UI/UX设计：设计漫画阅读器的用户界面和交互流程，确保用户能够轻松上手并享受阅读过程。

4.原型制作与评审：使用原型工具制作UI原型，与团队成员进行评审和修改。

5.质量保障与测试：参与测试工作，确保漫画阅读器的功能和性能符合预期；收集用户反馈，进行持续改进。

# **五、技术栈**

后端：spring、springmvc、mybatis

前端：vue

开发工具：IDEA

六、项目开发过程

这一部分，我们将较为详细的记录我们项目的开发过程，为了记录逻辑清晰，我们采用了时间顺序，并记录了开发中的细节。

10.18 - 项目启动和任务分配

今天我们正式启动项目，因为主题已定，是漫画阅读器，因此我们省去了较多讨论时间。但是我们先前的个人项目的同学项目基础比较差，似乎只能运行来显示界面，而且是安卓端的。因此我们几乎是0基础重新开发。为了确保项目的呈现，我们三人直接进入了用户需求分析和初步设计阶段。我们三人列出五点总体需求，我们明确了项目的核心功能，决定分成以下几个模块：

1.首页推荐和分类：显示推荐的漫画和分类展示，方便用户选择。

2.收藏功能：用户可以收藏喜欢的漫画。

3.用户界面：包括用户信息展示页面。

4.阅读界面：漫画阅读界面包括章节目录功能，用户可以跳转章节。

5.用户管理系统，增删查改的基础功能以及评论留言等功能。

我们为每个模块指派了负责的组员，制定了开发的初步计划。三个人在项目中负责不同的组件，我们还梳理了页面的基本路由结构和目录结构，决定使用 Vue Router 来管理页面跳转，并且准备调用API接口。

### 10.19 - 完成项目初始化和基础页面结构

今天我们完成了项目的初始化、发开工具的环境配置，创建了 Vue 项目并配置了项目目录，包括 components、router、store 等文件夹。开始阶段十分顺利，我们就几乎没有遇到太多苦难就顺利地建立了整体框架并创建了构建页面地基本组件，代码研发监管遇到不少bug，但通过查阅资料，也没花多少功夫就成功解决了。

在实现过程中遇到了 Vue 版本兼容性的问题，我们参考了 Vue 官方文档，并查找了相关论坛，最终通过安装合适版本的依赖解决了问题。

### 10.20 - 首页推荐和分类功能开发

今天的重点是实现首页推荐和分类功能，展示热门和分类漫画。首页分类和推荐我们已经有所进展，分类可以通过手动添加，但推荐功能今日可能完成不了了，需要完善后形成完整的分类功能后才能有更深一步的进展。同时后端部分已经较有成效，利用管理员身份进行小说的发布和内容的增加删除看起来问题不大。另外我们创建了 Home.vue 组件，试图通过调用 API 获取推荐漫画列表。

遇到的困难是 API 数据渲染不正确，导致页面为空白。我们通过 Vue Devtools 调试，发现是数据的异步加载问题，试图解决。

### 10.21 - 实现收藏功能

今天的任务是实现漫画的收藏功能，用户可以将喜欢的漫画加入收藏列表。我们在 Library.vue 中添加了收藏的列表显示，并使用 Vuex 来管理收藏数据。

昨日剩下两个难题，用户推荐功能和调用API。分类功能今日已经基本完善，因此开始攻克昨日剩下的难题。

今日的任务也遇到了困难0： Vuex 状态更新时页面未刷新。查阅资料后，我们直接通过在组件中使用 mapState直接绑定 Vuex 的收藏状态，使页面实时更新。

### 10.22 - 阅读界面开发及目录功能

今天开始开发漫画阅读界面 Read.vue，实现阅读页面并添加章节目录功能。用户可以在阅读时随时切换章节。但是事务繁多，加上之前堆积的难题尚未攻破，我们反而将精力投入到后端。

今日遇到的新的问题是章节数据的加载导致页面卡顿。我们参考了 Vue 文档中的懒加载方案，将章节数据分页加载，提升了页面的加载性能。

但今日项目新的部分进展缓慢。

### 10.23 - 完善用户界面

今天我们完成了用户界面 User.vue 的开发，用户界面主要显示用户的基本信息和收藏列表。我们在界面中实现了简单的样式，并确保收藏数据可以实时展示。收藏功能基本完成。同时我们也在今日完成了评论留言功能。

遇到的问题是页面样式显示不正确。我们通过调整 CSS 样式并引入 PostCSS 处理，使界面在不同分辨率下显示一致。

### 10.24 - 修复已知 Bug 并优化代码

今天主要是测试和修复前几天开发中的 Bug，并对代码进行了优化。例如修复了用户在快速切换章节时目录显示异常的问题。通过设置加载动画，我们改善了加载体验。

经验教训：在响应时间较长的地方添加加载动画可以改善用户体验。

### 10.25 - 添加全局状态管理

今天的主要任务是完善 Vuex 状态管理，包括首页推荐、收藏列表和阅读进度等全局状态。我们将各页面的状态集中到 Vuex 中管理，方便后续的扩展和维护。

遇到的问题是 Vuex 的状态更新有时会延迟导致页面不同步。通过查阅文档和论坛，我们学习了 Vuex 的异步操作，通过 Action 和 Mutation 更好地控制了状态更新。

学习与资料：阅读了 Vuex 的异步操作文档，了解了 Action 和 Mutation 的使用方法。

### 10.26 - 项目测试与修复

今天是项目的全面测试。我们在不同设备和浏览器中测试了项目功能，发现了一些兼容性问题。通过调整 Babel 和 PostCSS 配置，解决了部分浏览器不兼容的情况。

遇到的问题是部分低版本浏览器样式失效。通过修改配置文件并使用 Babel 的兼容性插件解决。

前几日留下的难题未能完全攻破。我们原本的阅读界面只做出了框架，没能来得及添加具体数据（API调用不成功），同时用户阅读推荐功能也未成型。但后端功能较为完整和稳定，因此我们在最后时间里对页面进行了美化设计，试图呈现最好的结果。

### 10.27 - 项目总结和展示准备

今天我们对项目进行了最后的优化和总结。由于时间有限，我们无法实现一些高级功能，例如用户注册/登录、评论与评分系统、推荐系统等。尽管如此，项目的核心功能已经完成，并可以正常运行。

我们制作了项目展示的 PPT，并在文档中记录了项目的开发历程和学习成果，准备展示。

# 七、代码审查与规范

1. 项目结构规范

项目根目录：包含 .mvn、.settings、src、target文件夹，以及常用的 Maven 项目文件（如 .classpath、.gitignore、.project、LICENSE、pom.xml）。这些文件均符合 Maven 项目的标准布局。

目录说明：

.setting：Eclipse IDE 的配置文件，包含 Java、Maven、和项目相关的偏好设置（如org.eclipse.core.resources.prefs、org.eclipse.jdt.core.prefs等）。 src：源代码文件夹，符合 Maven 标准结构。

main/java：存放 Java 源代码。

main/resource：存放资源文件（如配置文件、静态资源等）。

main/webapp：适用于 Web 项目的静态页面或 JSP 文件。

test：用于单元测试的文件夹。

target：编译输出文件夹，包含编译后的类文件、生成的源码、测试文件和构建状态。此文件夹通常在 .gitignore中排除。

2. 命名规范

类和接口命名：遵循 PascalCase（如 ComicReaderService），接口通常以 `I` 为前缀（如IComicService）。

包命名：使用全小写字母且遵循项目命名规则（如com.projectname.module）。

变量与方法命名：变量采用 camelCase（如 `comicTitle`），方法名应简洁且描述功能（如 fetchComics()）。

3. 代码规范

代码格式：

缩进：使用 4 个空格，不使用 Tab。

行长度：建议每行不超过 100 个字符，便于阅读。

空格与换行：适当使用空行分隔逻辑模块，提高代码的可读性。

注释：

Javadoc 注释：为所有公共类和方法提供 Javadoc 注释，描述其功能和参数。

单行注释：复杂逻辑或算法部分应有单行注释，以便其他开发人员理解代码意图。

异常处理：使用具体异常而非通用 Exception，便于准确定位错误来源。

4. 依赖与构建管理

Maven 配置文件 (pom.xml)：

仅包含必要的依赖项，移除冗余依赖。

使用 dependencyManagement 管理依赖版本，避免版本冲突。

明确指定 JDK 版本，以确保代码兼容性（建议在properties节中配置 maven.compiler.source 和 maven.compiler.target）。

配置插件（如 maven-compiler-plugin、maven-surefire-plugin），确保编译和测试的顺利进行。

5. 测试规范

单元测试：src/test 文件夹中为主要功能提供单元测试，确保代码正确性。

测试框架：推荐使用 JUnit 进行测试，且测试方法命名应清晰表达测试内容（如 testFetchComicById）。

测试覆盖率：尽量保证核心模块达到至少 80% 的测试覆盖率。

# 八、功能

**（部分功能展示，因在PPT中已有详细说明，故此仅作大致介绍）**

**漫画展示**

# 058c1b2193ed67e1cfa035c69154310

**公告展示**



**后台管理**



# 九、总结

这次漫画阅读器项目使我们深入了解了 Vue.js 及其生态系统的使用，并学会了使用 Vue Router 和 Vuex 等工具来管理页面和状态。虽然未能实现所有功能，但我们通过实践提升了前端开发能力，并积累了项目管理和问题解决的经验。

1. 技术提升：通过项目开发，我们掌握了更多前端开发技术、后端开发技术和数据库管理技术。

2. 团队协作：我们学会了如何在团队中有效沟通、分工和协作，提高了团队的整体效率。

3. 用户体验：我们深入了解了用户对漫画阅读器的需求和期望，为后续产品开发提供了宝贵经验。

想法很美好，现实很骨感，我们本来是想完成一个完整的漫画阅读系统，但受限于时间、技术基础有限以及我们是从零开发，并没有之前同学留下的个人项目的基础，所以前端部分做的确实不尽如人意。我们也将反思我们在这次开发过程的经验教训，我们的确在任务分配、团队协商沟通中出现了很多问题。并且我们的技术水平也较为有限，应该花更多时间学习相关技术，争取在团队项目中有所突破。