# 项目开发总结报告

# 1 引言

**软件项目的任何文档都包含引言部分，需介绍文档的编写目的、背景、预期读者、以及参考资料，还可包含文档历史演化版本。在课程实践中，引言部分可不必填写。**

# 2 实际开发结果

## 2.1 产品

我们的最终产品名为《E 路相伴》，是一个面向北京大学师生的网页应用，提供路况的实时情况与生活指南的服务。
我们的产品希望可以满足以下需求：

- 实时提供学校道路拥堵情况
- 提供校内各场所信息，如办公/开放时间，商店/咖啡店，自助打印机位置
- 以地点为节点，集成地点的常用信息，成为一个可视化生活指南
- 学生间的交流平台
- 校内平台，重点提供功能性信息的地图软件

对于前端的构建，产品共有 10 个组件文件，分别对应着不同的功能，它们是：

- Avatar.vue：主页中的头像组件组件与登出功能，
- HelloWorld.vue：选择路况还是生活指南（在最终版本已不用）
- Home.vue：最初页面，显示登录与注册功能
- Login.vue：登录页面
- Map.vue：地图页面
- MapHeader.vue：搜索地点功能
- Register.vue：注册页面
- Roadsidebar.vue：反馈路况功能
- VenueSidebar.vue：显示地点信息功能
- initMap.js：初始化地图功能，将北大校园圈出来

对于后端的构建，产品主要有 3 个数据库，分别存储着产品所需的不同信息，它们是：

- AppUser：存储用户信息，确保用户输入为学校用户，密码传输时使用加密
- Location：存储学校地点内的信息，存储地点标签方便搜索，提供评论功能
- Road：存储学校路况信息，提供路况反馈功能，根据反馈计算当前路段拥堵程度

## 2.2 主要功能和性能

逐项列出本软件产品所实际具有的主要功能和性能，对照项目开发计划、功能需求说明书的有关内容，说明原定的开发目标是达到了、未完全达到、或超过了，并分析原因。

| 立项目标 | 实际情况 | 偏差有无 | 原因分析 |
| --- | --- | --- | --- |
| 地图浏览与展示 | 用户可以自由缩放、平移、旋转地图，查看不同地区的地图细节。 | 达到 | 软件采用了高德 API 接口，确保了地图的流畅展示和清晰度。 |
| 地图搜索 | 用户可以通过白标签搜索兴趣点，并在地图上进行定位 | 达到 | 集成了高效的搜索引擎和精准的地理编码服务 |
| 地图数据更新 | 地图数据能够定期更新，保证信息的时效性和准确性。 | 达到 | 实施了自动化的数据更新机制，定期从数据源获取最新信息 |
| 社交评论功能 | 用户可以在地图内显示的地点进行评论 | 达到 | 允许用户对地点功能进行评论和交流 |
| 用户界面友好性 | 软件具有直观易用的用户界面。 | 达到 | 进行了多次用户测试和界面迭代，优化了用户体验。 |
| 安全性 | 软件保护用户隐私和数据安全。 | 达到 | 实施了数据加密和安全的认证机制。 |
| 性能优化 | 软件运行流畅，无明显的卡顿或延迟。 | 达到 | 对软件性能进行了优化。 |

以上是对本地图软件产品的功能与性能的逐项评估。总体上，开发目标得到了实现，部分功能甚至超过了预期。

## 2.3 进度

列出原定计划进度与实际进度的对比，明确说明，实际进度是提前了、还是延迟了，分析主要原因。

| 里程碑 | 预定日期 | 实际日期 | 偏差有无 | 原因分析 |
| --- | --- | --- | --- | --- |
| 需求分析与设计 | 2023-09-29 | 2023-09-29 | 准时 | 需求收集过程中较为顺利，没有过多的需求变化 |
| 系统架构搭建 | 2023-10-28 | 2023-11-04 | 延迟 | 技术选型过程中遇到了一些未曾预料的技术难题，导致架 |

| | | | | |
|---|---|---|---|---|
| | | | | 构搭建延期。 |
| 功能开发与实现 | 2023-12-17 | 2023-12-20 | 延迟 | 部分功能实现复杂，开发过程中遇到了较多的技术挑战，导致开发进度延期。 |
| 系统集成与测试 | 2023-12-17 | 2023-12-20 | 延迟 | 集成过程中发现了一些接口兼容性问题，需要进行调整，导致测试阶段延期。 |

综合来看，实际进度普遍延迟于预定日期。主要原因包括技术难题、功能实现的复杂性、系统集成问题。为了改进未来的项目进度管理，团队需要更加细致地评估项目风险，提前预留充足的时间缓冲，并加强项目各个阶段的沟通与协作。

# 3 开发工作评价

## 3.1 对生产效率的评价

给出实际生产效率，包括：

a. 程序的平均生产效率，即每人月（或人日）生产的行数；

b. 文件的平均生产效率，即每人月（或人日）生产的页数或字数；

c. 测试的平均生产效率，即每人日执行的测试用例数目

并列出原订计划数作为对比。

| 项目 | 实际生产效率 | 计划生产效率 | 比较结果 | 原因分析 |
|---|---|---|---|---|
| 程序开发 | 每人月 1000 行 | 每人月 800 行 | 提前 | 开发团队采用了更高效的编程技术和工具，提高了生产效率。 |
| 文件编写 | 每人月 1 页或 1 图 | 每人月 3 也或 2 图 | 提前 | 文档编写人员对项目需求理解更深入，减少了返工，提高了效率。 |
| 测试执行 | 每人日 10 个测试用例 | 每人日 5 个测试用例 | 提前 | 测试团队采用了自动化测试工具，提高了测 |

| | | | | |
|---|---|---|---|---|
| | | | | 试效率。 |

## 3.2 对产品质量的评价

测试中没有监测到程序编制的错误。

## 3.3 对技术方法的评价

在软件开发生命周期中，选择合适的技术、方法、工具和手段对于项目的成功至关重要。以下是对本次项目利用的技术栈和方法的一些评价：

1. Vue.js：
    - 优点：Vue.js 是一个渐进式 JavaScript 框架，易于上手，文档齐全，社区支持力度强。它的响应式数据绑定和组合式 API 设计使得开发复杂的单页应用变得简单。此外，Vue 的轻量级特性和组件化开发方式，有利于提高开发效率和项目的可维护性。
    - 局限：相比其他成熟的框架如 React 和 Angular，Vue 在某些方面可能不那么成熟，特别是在生态系统和第三方库的支持上。对于大型企业级应用，可能需要更多的考虑和评估。
2. JavaScript：
    - 优点：作为 Web 开发的核心语言之一，JavaScript 的普及度和灵活性非常高。它支持面向对象、命令式以及函数式编程范式，允许开发者根据项目需求选择合适的编码风格。随着 ES6 及之后版本的更新，JavaScript 的语法更加现代化，大大提升了开发体验。
    - 局限：JavaScript 的异步编程模型（回调、Promise、async/await）有时候会让异步代码的编写和理解变得复杂。另外，JavaScript 的运行环境（如浏览器和 Node.js）在某些 API 和特性的支持上存在差异，需要开发者额外注意兼容性问题。
3. Element.js：
    - 优点：Element.js 是基于 Vue 2.0 的桌面端组件库，提供了丰富的组件，方便开发者构建复杂的界面。它遵循了 Vue 的设计哲学，易于集成和使用，且样式美观，适合快速开发企业级的后台产品。
    - 局限：Element.js 依赖于 Vue 2.x，如果项目需要升级到 Vue 3，可能需要寻找替代方案或等待 Element.js 的更新。
4. Django Database：
    - 优点：Django 是一个高级 Python Web 框架，它鼓励快速开发和干净、实用的设计。Django 提供了一个强大的、直观的数据库操作接口，让开发者可以用 Python 类来表示数据库中的表，并通过 Python 代码来执行数据库操作，大大简化了数据库编程。

- 局限：Django 虽然强大，但在处理复杂 SQL 查询时可能不如直接编写 SQL 语句高效。此外，对于非关系型数据库的支持，Django 可能不如其他框架灵活。

5. 面向对象开发方法：
    - 优点：面向对象开发方法（OOD）通过抽象和封装，提高了代码的重用性和可维护性。它鼓励开发者编写模块化和可扩展的代码，有助于大型项目的开发和维护。
    - 局限：在某些情况下，面向对象的方法可能会增加开发的复杂性，尤其是在处理简单或者小型项目时，过度设计可能会降低开发效率。
6. 敏捷开发方法：
    - 优点：敏捷开发强调快速响应变化、小批量快速迭代、团队协作和持续改进。它有助于更快地交付可用的软件，更好地满足客户需求，提高开发过程透明度和团队协作效率。
    - 局限：敏捷开发方法需要团队成员之间有很高的协作能力和沟通效率，对于团队规模较大或项目较为复杂时，敏捷方法的实施难度和成本可能会增加。

总的来说，选择技术栈和方法应当基于项目需求、团队技能和项目规模来决定。实际开发过程中，也需要不断评估和调整，以确保项目能够高效、顺利地进行。

## 3.4 出错原因的分析

开发中出现的错误的原因分析如下：

1. 编码问题
    a. 编码标准不统一，导致代码风格混乱，难以阅读和维护。
    b. 缺乏代码复用和模块化，导致代码重复和冗余。
    c. 忽视编码最佳实践，如未能遵循单一职责原则、开闭原则等。
2. 工具和环境问题
    a. 使用的开发工具（高德 API）文档与版本有些混乱，导致项目前期进展不顺
3. 知识和技能不足
    a. 由于全员小组皆是第一次进行开发工作，因此对特定技术或工具缺乏足够的了解和经验，导致在项目进展时走了一些弯路。

# 4 经验与教训

在软件工程中，每个项目都是一次学习和成长的机会。以下是从本次开发项目中得到的一些主要经验与教训，以及对未来项目开发工作的建议：

1. 需求管理的重要性：
    a. 经验：明确、稳定且可测试的需求是项目成功的关键。
    b. 教训：需求变更应谨慎处理，且需有严格的变更控制流程。

2. 编码规范和最佳实践的遵循：
    a. 经验：统一的编码风格和遵循最佳实践可以提高代码质量和团队协作。
    b. 教训：应持续培训和监督以确保团队成员遵循规范。
3. 测试的全面性：
    a. 经验：全面的测试可以及早发现和修复缺陷。
    b. 教训：测试计划应与开发计划同步，且测试用例应覆盖所有功能和场景。
4. 文档的必要性：
    a. 经验：良好的文档可以提高项目的可维护性和可追溯性。
    b. 教训：文档应随代码一起更新，以保持其准确性和相关性。
5. 团队协作和沟通的重要性：
    a. 经验：高效的团队协作和良好的沟通可以减少误解和冲突。
    b. 教训：应建立有效的沟通机制和团队协作工具。
6. 项目管理的关键角色：
    a. 经验：有效的项目管理可以确保项目按时按质完成。
    b. 教训：项目经理应具备良好的组织和沟通能力，并能够适应变化。
7. 模块化与复用：
    a. 经验：通过面向对象设计，可以创建高度模块化的系统，易于维护和扩展
    b. 教训：在设计类和对象时，应充分考虑其复用性，避免过度耦合
8. 快速迭代与增量开发：
    a. 经验：敏捷开发鼓励快速迭代，有助于及时反馈和调整。
    b. 教训：在迭代过程中，应确保每个迭代都有明确的目标和可交付的成果。