**迭代评估报告**

　　　　　　　　　　　　　　　　　　评估日期：

|  |  |  |  |
| --- | --- | --- | --- |
| 组号 | 第2组 | 项目名称 | 兴趣爱好社区(Hobbitat) |
| 迭代名称 | 界面原型迭代计划 | 实际起止日期 | 10.1-10.27 |
| 任务达成情况：（完成的任务、实现的功能、进度、质量等）  1. 设计迭代计划并撰写迭代计划表。  2. 需求调研，使用软件需求规约文档来细化需求和确定软件方向  3. 完成用例建模，构建Use-case模型  4. 编写Vision 文档完成需求规约与概述。  5. 初版界面原型设计（采用草图和在mockplus网站上构建元素的方式）  6. 花费更多的时间完成HTML代码，完成主要页面的界面设计。 | | | |
| 评审/测试的结果：（执行了哪些评审和测试？评审和测试的结果如何？）  1. 对于软件需求规约文档，Vision文档，Use-case模型等，我们小组采用先统一讨论，然后由一人具体写作，之后根据课堂内容，进一步的讨论，对各文档进行评审，然后再细化文档。  2. 对于界面原型的设计，我们小组先分别查找资料，之后展示各自的草图和构思,然后分别负责一部分界面的设计。在具体设计过程中，我们首先使用了mockplus, codepen等网站上的工具，然后编写初版HTML代码，之后进行评审和美化。最终形成结果。 | | | |
| 问题、变更和返工：（遇到的问题、发生的变更、是否需要返工等）  **问题和解决：**  1. 对要设计的软件的定义不清晰。在设计之初，我们的方向并不明确，导致在撰写相关文档时，难以达到一致。之后我们对软件的功能和定位进行了更明确的讨论，从而确定了开发的具体方向。  2. 对于理论知识掌握的不够。对软件工程理论知识掌握不够导致我们对于软件和文档的理解有分歧，并且造成了多次返工。但通过在课堂上的学习，帮助我们加深了对理论知识的理解。  3. 开发能力不高。小组内成员对于开发技术的掌握普遍不高，因此需要付出较大的学习成本来进行学习。此阶段我们小组学习了较多的HTML设计的知识。  **变更：**  1. 对于Vision 文档和用例建模等，自己组内进行了多次的变更和修改。主要是最初对软件的定位不明确，并且理论和开发知识的不充足导致的。  2. 对于软件的界面原型，从最初的草图到mockplus 绘制界面元素，之后编写HTML并完善页面布局。  3. 在设计完初版HTML页面后，我们小组进行评审和变更，将界面进一步细化。 | | | |
| 经验和教训：  经验：  1. 小组之间的分工合作非常重要。通过有效的分工，可以有效地提高开发的进度和效率。  2. 要善于合作寻找网络资源。秉持着“不重复造轮子”的原则，在原型设计方面，我们小组的部分页面也参考了网上的必要资源，从而帮助我们在开发过程中一边做，一边学习技术，加快了效率。  3. 使用git与github来做版本管理。使用git的分支功能等方便多人开发。  教训：  1. 在撰写相关文档和开始开发之前，要对软件的各个方面有明确清晰的认识，在迭代周期之初，我们小组由于认识不明确，而导致多个文档的返工，降低了效率。  2. 在最初，没有充分利用git 等版本控制工具，使得文档的传递效率和开发效率收到影响。 | | | |