**开发团队半年度绩效评估表**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | | Roger Zhang | | 岗位 | .NET开发工程师 | | | 评估时间 | | 2018-07-04 |
| 一、计划关键任务记录 | | | | | | | | | | |
| 序号 | 职责分类 | | 关键任务描述 | | | 其他参与人员 | 计划完成时间 | | 计划滞后比例 | 完成结果（是否有质量问题） |
| 1 | Partner & SSO | | Partner & SSO项目需求文档、设计文档编写、SSO开发环境搭建、SSO配置页面开发、Partner custom CSS & rebranding、Comm100提交Azure AD gallery申请、接口限流 | | | Michael, Damon, Kim, Joe, Grubby, Lizz, Fiona, Gorgeous, Sera, Stones | 2018-07-15 | | 尚未截止 | 尚未上线 |
|  |  | |  | | |  |  | |  |  |
|  |  | |  | | |  |  | |  |  |
|  |  | |  | | |  |  | |  |  |
| 二、半年度工作小结（包括优秀表现、待提高方面、个人感受体会。）  **优秀表现**：1. 积极完成项目经理安排的任务；2. 虚心接受他人的意见，并及时改正；Code Review或项目开发过程中也会及时提出自己的意见。3. 积极参与同事间的技术讨论，协助进行一些技术问题的验证；4. 遇到问题能熟练使用英文搜索处理方案，能熟练阅读英文技术文档，学习能力较强。  **待提高方面**：1. 作为新员工，需要进一步熟悉公司的各项目的业务逻辑及实现。同时需要进一步熟悉公司的各项规范。2. 学海无涯，需要提高的方面很多。  **个人感受体会**：  1. 公司的氛围比较融洽，同事之间互帮互助，新员工能找到对应老员工了解项目；遇到具体技术问题，团队成员也会互相讨论，协助找到解决方案。  2. 公司有许多能力比较强，工作负责的同事；当项目比较紧急或邻近发布的时候，总能看到认真负责的同事加班的身影。另外，在与其他员工交流时，能感受到他们对各自各做非常熟悉，非常专业。  3. 公司对员工的工作环境非常重视。各项设备配备齐全，不错的电脑配置加上双显示器能提高开发的工作效率。每天下午的水果或零食也体现了公司对员工的关怀和重视。  4. 公司每周都组织羽毛球等活动，体现了公司希望丰富员工的业余生活，让员工能在休闲的同时锻炼身体。  5. 公司非常重视对员工的培训，鼓励员工自我学习，提升各项技能。 | | | | | | | | | | |
| 三、需要公司提供的帮助或希望参加的培训  1. 对于研发人员来说，希望有更多针对现有产品或模块的比较深入的培训或针对开发工作流程的培训。  2. 希望公司可以尽快解决长沙现有网络和编码环境缓慢的问题。 | | | | | | | | | | |
| 四、对工作、团队、主管等意见或建议  1. 项目估时需要考虑工作效率及其他可能占用上班时间的可能，充分考虑沟通或部门协作的成本。如果设计不够详细或未评审也可能会导致估时出现很大偏差。  2. 需要在项目的前期尽早确定需求及设计，并充分评审。目前遇到过的困难是需求需要多个部门的人员评审，但是容易出现人员聚齐比较困难、意见又有出入的情况。  3. 测试参与需求讨论。因为某些原因，Partner & SSO需求阶段测试未介入项目，但是实际上还是要花时间对测试进行需求和设计的培训，效果也没有一开始就有测试参与讨论的效果好。  4. 设计文档尽量详细。关于设计文档有一个矛盾的情况，就是设计文档可以写的更详细，但是如果过于详细的话，其实可能花的时间不会比编码实现花的时间少。但是设计文档如果比较详细的话，有利于测试了解产品内部逻辑，方便测试编写测试用例。同时，详细的设计文档也可以让新员工或者对该模块感兴趣但是没有代码权限的人自行了解此模块。  5. 听说长沙机房里的服务器没有使用UPS（不是很确定是不是这样），如果是这样的话，建议购买UPS（备用电源的作用），如果出现了断电的情况可以防止服务器突然断电。我来公司的这一个季度出现了最少3次断电的情况，其中有两次是正在上班的时候，有一次因为断电，导致整个办公室最少半天不能正常工作。  6. 目前的Code Review方式不太规范，通过往源代码中添加注释的方式给他人提意见，不能确保他人进行更改。我在进行Partner & SSO项目Web API项目的Code Review时，我提议所有的注释前面添加TODO，因为这样的注释是可以从VS中查找到的，但是这样还是比较奇怪。  以下是我之前使用GIT进行开发的流程：  Rebase到开发分支（如develop）→ 自己的分支更改 → 创建Pull Request请求合并到develop分支 → TFS中配置策略进行Build，包括代码编译、执行单元测试、检查代码指标（如测试覆盖率是否下降等）→ 开发人员确保Build成功 → 团队成员对代码进行Code Review，可以配置策略至少多少人审核通过后才允许将代码合并到开发分支。只要有任何人Reject，该代码就不能合并成功。开发人员需要根据注释修改后，对Comment执行Resolve操作，邀请审核人员重新审核，审核人员可以根据代码修改情况选择Approve，如果觉得还是不满足条件，也可以添加新的Comment或者将原来的注释Reactive。  我觉得这个流程最大的优点是充分利用了TFS的策略，可以有效利用Unit Test和Code Review的手段，保障我们开发分支上的代码质量，同时也能始终保证我们的日构建都是成功的。  7. 由于公司部分产品是较久之前就开发好的，部分代码使用的技术比较保守，这是可以理解的，因为目前的版本使用比较稳定的情况下，公司不会考虑花费较大成本对原有代码进行大面积修改。但是现有代码存在的一些情况我们也要做到心里有底：  a. 绝大部分后台代码未进行单元测试。  单元测试的优点的话，我们先不谈什么测试驱动开发之类的，比较显而易见的优点有：防止一些逻辑错误、边界错误等；防止开发人员修改某模块时影响其他模块而未发觉；开发人员在发布测试前可以针对各种不同输入情形进行测试，提前发现问题，提高质量；好的单元测试能帮助他人理解作者的编码逻辑或意图。  b. 未使用面向接口或依赖注入的编码方式，这也会导致单元测试比较困难；  c. 未使用目前比较流行的异步编程（async, await）；异步编程的优点：例如我们ADO.NET访问数据库的方法都有末尾带async和不带async的方法。使用异步方法的话，在进行数据库操作这种可能耗时比较长的的操作时，可以先将计算资源释放用来处理其他请求。这对于数据库访问、IO操作等这些耗时较长又不需要任何计算资源的方法来说，可以让我们的服务器可以同时处理更多的请求。  d. 当我们在选择Web API的框架时，因为考虑到目前的环境都是使用的.NET Framework 4.5.1，因而放弃了目前比较新的.NET Core。这是可以理解的，因为公司其他的.NET代码还没有 | | | | | | | | | | |
| 基于.NET Core的，如果这些Web API项目选用了.NET Core的话，可能需要在发布的环境下安装.NET Core。但是当时有些开发提到了.NET Core，这里说下.NET Core的优点：跨平台，可以发布到Linux下，也可以在多种环境下进行开发；微软现在主推的是.NET Core，以后也基本都只会对.NET Core继续投入资源， 最终.NET开发人员还是要转移到.NET Core去；.NET Core到了2.1的版本，在国外已经有较久的使用时间，已经比较稳定；由于.NET Core是微软最新重新开发的，微软进行了很多的优化，使整个项目的结构非常的清爽；如果要使用docker（基于Linux）之类很火的容器化工具，需要支持跨平台。  8. 目前提供给国外团队查看的Use Case文档，很难找到一个平衡点。如果写的过于详细，有人会认为过于啰嗦；如果写的过于简单，可能又会抱怨很多场景都没有描述到。  首先Use Case原来的意义应该主要是用来描述需求指导开发人员进行开发的，里面的内容包含很多的属性，场景不仅包含成功的情况，还要包含各个步骤的可能的扩展。但是我们现在的Use Case应该是在原来Use Case基础上的阉割版，理由是这些东西是给国外的市场人员查看的，他们不需要了解这么多。但是首先一个软件行业的销售人员没有能力看懂一个详细的Use Case吗？其次他们没有必要看详细的Use Case吗？如果客户在向销售人员了解功能的时候问的比较细致，他们如何作答？  9. 发布测试流程比较乱；开发提交测试时没有任何说明文档，也没有部署文档；开发一天内提交的发布过多；  10. 随着用户越来越多，可能的请求越来越多，环境越来越复杂，未来是否有考虑通过使用micro services， docker等方式来应对？  11. 目前代码中的缓存方式多种多样，未来会不会使用如Redis之类的统一下？  12. 如果冒烟失败的话，需要打回并暂停测试。  13. 如果是时间跨度大一点的项目，建议采用迭代的模式进行开发和测试，而不是等项目做的差不多了，才叫测试介入进行测试。  14. 由于我们存在免费、hosted、ent等平台，在进行发布的时候是否考虑分批次进行发布，比如先在免费平台上面试用一个星期左右再推广到其他平台上？  15. 需求或设计评审完后，如果之后的开发过程中出现了较大的变动，是否需要一个书面一点的流程？ | | | | | | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 五、绩效评价（参照指标描述，在分数栏中打分，并按其分值相加计算出评价结果。） | | | | | |
| 评价项目 | 评价指标 | 指标描述 | | 自我评价 | 主管评价 |
| 工  作  完  成  质  量  40分 | 代码格式  （10分） | 代码非常清晰，简洁易懂。（8－10） | | 8 | 9 |
| 代码比较清晰，可读性尚可。（4－7） | |
| 代码条理不清，可读性较差。（1－3） | |
| 代码  逻辑性  （10分） | 代码具有逻辑性，流程最简化，依赖性小。（8－10） | | 8 | 9 |
| 代码逻辑性尚可，但有一定依赖性。（4－7） | |
| 代码逻辑不强，可能存在维护性风险。（1－3） | |
| 代码  可用性  （10分） | 代码实现合理，效率较高，可保证软件运行。（8－10） | | 8 | 9 |
| 代码实现基本合理，软件运行速度一般，。（4－7） | |
| 代码实现不太合理，效率较低，影响软件运行。（1－3） | |
| 代码  复用性  （10分） | 代码可多次运用，复用性优秀（8－10） | | 8 | 9 |
| 代码只能少量运用，复用性一般。（4－7） | |
| 代码完全不能复用。（1－3） | |
| 工  作  难  度  20分 | 项目  重要性  （10分） | 大型项目，项目优先级高。（8－10） | | 8 | 8 |
| 一般项目，项目重要程度一般。（4－7） | |
| 小型项目，如Bug修复之类。（1－3） | |
| 专业  复杂度  （5分） | 项目非常复杂，有很大的挑战。（5） | | 4 | 4 |
| 项目难度一般，有一定的挑战性。（3－4） | |
| 小型项目，无难度。（1－2） | |
| 工作  创新度  （5分） | 具有创新思维，不断思考优化开发工作。（5） | | 4 | 5 |
| 有一定的创新，可以提出一些创新性建议（3－4） | |
| 按部就班，较少思考创新。（1－2） | |
| 工作量  10分 | | 工作量很饱满，在完成开发任务的同时，还承担了项目之外的工作。（8－10） | | 8 | 8 |
| 工作量饱满，基本完成项目开发任务。（4－7） | |
| 工作量不饱满，尚有空闲。（1－3） | |
| 工  作  态  度  30分 | 主动高效  （10分） | 独立提出切实可行的改进方案，并推进实施，取得良好的成效。（9－10） | | 7 | 8 |
| 工作中主动发现问题，提出有价值的改进建议。（7－8） | |
| 调动各方面资源以达成目标。（4－6） | |
| 被动执行安排的工作，遇到困难被动等待，对工作中问题视而不见。（1－3） | |
| 团队协作  （10分） | 协助对方获取成功，在达成团队整体目标的同时实现个人目标。（9－10） | | 7 | 9 |
| 认真听取对方意见，清楚表达自己意见，并且提出有价值的建议。（7－8） | |
| 与团队成员良好沟通，保证项目按期进行。（4－6） | |
| 不与团队成员沟通，完全按照个人设想工作。（1－3） | |
| 学习分享  （10分） | 主动学习，工作技能明显提高，分享并帮助团队成员进步。（9－10） | | 8 | 8 |
| 有意识地学习知识技能和业界先进经验，并在工作中实践，乐于分享。（7－8） | |
| 能够参加公司培训，在某些方面提高自己，有一定的分享。（4－6） | |
| 缺乏自我培养和提高意识，随大流。（1－3） | |
| 自评结果： 78 | | | 主管评分： 86 | | |
| 主管反馈：包括继续发挥的优点和下一步努力方向。  （若评价指标自评与主管评分差距较大，建议在此列出具体的案例来说明问题，以便于后续沟通。）   1. 技术能力突出，攻关能力很强，很多技术问题都会自己的独到见解 2. 工作非常主动，对其它同事指导也很大，愿意主动的接受挑战 3. 全局观比较弱，很少会从整体来看待一个项目，任务优先级经常还要通过PM的不断强调, 如果后期要成为技术经理一类的角色，这个方面是一定要加强的   主管签名： | | | | | |