TeamIcal 产品方案设计

1、项目实施可行性

1.1、行业市场分析

现代的人们正处于一个快速发展的时代,每个人都有着自己的事情工作，面临各种各样的选择。在这种快节奏的工作生活中，人们很容易忘记自己的时间日程安排，团队项目成员之间的沟通协作也可能因成员的时间安排遗忘和任务分配不明确等而效率低下。因此拥有一款可以面向各类需求人群，比如个人普通用户、团队项目负责人和团队项目成员的时间日程管理 APP 可以解决时间安排遗忘，团队成员间沟通协作效率低下等问题，这样用户可以合理地安排日程时间不会遗忘，团队成员可以高效率地沟通协作完成任务项目。

1.2、同类产品分析

APP 类型	服务对象	个人定制	时间规划	时间提醒	任务分配	日志记录
Worktile	较全面	无	良	优	良	优
Ticktick	单一	无	优	优	差	优
日程吧	单一	无	良	良	差	良
时光序	单一	无	优	优	差	优

通过对现有较多人使用的日程管理APP的调查了解,相类似的日程管理APP虽然某些方面的功能如时间提醒，时间规划等功能有着较好的用户体验，但它们的功能不是特别全面，服务的对象人群较为单一，而我们设计日程管理 APP 功能较为全面，服务人群较广包含个人普通用户和团队型用户，且使用简单，对于某些专业人群还提供了个性化定制需求。

1.3、自身条件分析

我们希望我们设计的这款日程管理 APP 可以为不同类型的用户提供便捷的日程时间管理安排，满足用户的各类需求。

服务对象：服务对象全面，既包含普通个人用户，也可以服务团队负责人及成员
个人定制：系统提供了一些 API 供专业人员二次开发，从而实现个性化的定制需求
时间规划：根据用户提供的空闲提供适合个人用户或团队的时间规划安排
时间提醒：对用户的工作任务安排进行提醒，避免用户遗忘带来不便
任务分配：对于团队负责人用户可以对团队合作项目进行任务分配，团队成员

获取任务

信息进行分工合作

日志记录：用户可以看到任务工作完成进度

2、产品定位及目标（用户群分析等）

此款产品致力于提供最方便的日常管理操作，为不同角色的用户打造多功能全方位的产品，把多类角色的日程管理整合成一个日程表。此款产品不仅基本包含市面上同款产品的功能，更有一些差异化价值点。

2.1、产品需求定位：

便于用户对自己的日程进行安排以及及时获得提醒。

满足团队对于团队成员的时间管理规划需求。

2.2、差异化价值点定位：

2.2.1 用户群体方面：

与市面上的同类型时间管理产品相比，此款产品不仅面向个体，同时面向团队。便于团队对其成员进行活动安排的设置和调整。

与市面上的同类型时间管理产品相比，此款产品提供二次开发的接口，便于用户开发个性化功能。

2.2.2 使用功能方面：

与市面上的同类型时间管理产品相比，此款产品提供文本自动识别功能，方便用户进行日程设置。

与市面上的同类型时间管理产品相比，此款产品自动进行国内外时间转换，方便国外的活动提醒。

与市面上的同类型时间管理产品相比，在团队使用方面，此款产品可以收集某一次活动的团队成员的空闲时间表并自动统计时间。

与市面上的同类型时间管理产品相比，在团队使用方面，此款产品可以收集成员

的完成情况。

2.3、主要用户：

需要进行日程安排和管理的用户。
对于团队成员日程安排有需求团队用户。
对于有个性化开发需求的用户。

3、推广方案

3.1、企业的产品销售策略

主要以差异化价值点为主要优势，建立以团体和有编程能力的人为核心人群，并以此为基点，扩散用户范围。即先通过各种宣传手段稳定核心用户，并通过核心用户的宣传以及其人脉关系带动周边用户使用此产品。
产品先在小范围内做试运行，后逐步扩大。

3.2、产品的推广计划

先在集中的几个月做好核心人群的宣传，宣传方式包括群发产品海报等。可以考虑以学校为基点进行推广，由于学校内有部分学院的学生修过编程类的课程，具备一定的开发能力，其次学校内的同学大部分会参加项目如大创、金种子等，以及团队比赛如 ACM、数学建模等。因此，学校内的学生符合产品的核心人群要求。

之后以这些用户为基点，逐步扩大用户所在范围。

4、运营规划书

4.1、运营目标

前期：团队用户占主体的 70%，其余为有编程能力的用户以及个人用户
中期：团队用户占主体 60%，有编程能力的用户 10%，个人用户 30%
后期：个人用户占 70%，团队用户占 30%，其中包括有编程能力的用户

4.2、维护计划

维护范围：“智能日程管理软件”app
维护周期：一个月
维护流程：
1. 对用户反馈进行回复
2. 采纳合理的用户建议
3. 修护漏洞
4. 调整需求变更的功能

5、产品内容总策划

5.1、应用流程规划

游客通过 app 注册账号，成为 app 用户，用户拥有个人日程管理的一系列功能，比如添加日程，查看日程，删除日程，到期提醒，快速添加日程等功能。app 用户也可以创建团队，邀请其他用户进入团队，从而实现团队日程管理等功能。

5.2、设计与测试规范

设计规范：
（1）我们采用 XUI 的组件库，在整体风格上保持一致。
（2）菜单深度尽量控制在三层以内。
（3）相似或相近的功能按钮尽量放一起。

（4）工具栏太多的时候可以使用工具箱。
（5）一条工具栏的长度最长不能超出屏幕宽度。
（6）菜单前的图标尽量能直观的代表要完成的操作。
（7）提示、警告、或错误说明应该清楚、明了、恰当。
（8）布局要合理,不宜过于密集，也不能过于空旷，合理的利用空间。
（9）当选项特别多时，可以采用列表框，下拉式列表框。
（10）对重要信息的删除操作最好支持可逆性处理，如取消系列操作。

测试规范：

(1) 利用 Postman 等工具对接口进行测试。
(2) 使用 Appium 等自工具进行自动化测试
(3) 利用 Monkey、LoadRunner 进行压力测试。

5.3、开发日程表

时间	需要完成的需求
第四周（9.28-10.11）	1. 确定开发团队的组织结构，人员组成与人员分工。 2. 确定项目，编写用户故事、可行性分析和需求分析
第六-七周（10.12-10.25）	设计阶段以及前期准备
第八周-第九周（10.26-11.4）	完成个人日程管理中前四个功能模块，分别为：添加日程、查看日程、删除日程、编辑日程
第九周-第十周（11.5-11.15）	完成个人日程管理中快捷添加的模块，分别为：文本抽取、对接邮件、接口 API
第十一周-第十二周（11.16-11.25）	完成团队日程管理中的前三个功能模块，分别为：创建团队、团队管理、发布邀请链接
第十二周-第十三周（11.26-12.2）	完成团队日程管理中的剩余的团队负责人的模块，分别为：创建团队日程、空余时间调查
第十三周-第十四周（12.3-12.13）	完成团队成员的模块，分别为：加入团队、查看团队
第十五周（12.14-12.17）	列出第一版的总结与需要调整的地方
第十六周（12.21-12.27）	完成第二版
第十七周（12.28-1.3）	项目结束总结

6、技术方案

6.1、客户端开发：

客户端开发拟采用安卓原生开发，选用 java 为开发语言，运用 google 官方推荐的 IDE Android Studio 进行开发。尽管 google 推荐使用最新的 kotlin 作为下一代的的安卓应用开发语言，但我们经过考虑后认为目前情况下 kotlin 的技术支持还不够丰富，故选择了具备更多参考资料的 java 作为项目开发语言。

6.2、界面框架选择

6.2.1 选择 XUI 框架的原因

经过比较多个现有的安卓开源用户界面组件库，最终我们选用了 XUI（https://github.com/xuexiangjys/XUI）作为我们的界面框架，具体的理由有如下几点：

1、与直接使用原生的安卓组件开发相比，运用现有的组件库能够较为方便快捷的开发出美观的界面，同时兼顾开发质量和开发效率。

2、XUI 组件库提供了许多常用的组件，如轮播条组件、常用布局组件、按钮组件等，基本覆盖了我们项目的实际需要。

3、XUI 组件库支持高度定制化，仅通过修改少量的顶层样式就可以控制整体的 UI 风格，能够方便的根据项目的实际开发需要修改组件的样式。

6.2.2、选择其他组件/库的说明

此外，考虑到本项目是一个日程管理软件，需要使用日历组件显示日程条目信息，而 XUI 中并没有提供该组件。经过调研比较开源项目中的几款日历组件，最后决定选用 CalenderView 组件(https://github.com/huanghaibin-dev/CalendarView) 作为项目的日历组件，具体的理由如下：

1、该组件是基于 Canvas 绘制实现的，与基于 ViewGroup 实现的其他同类型组件相比，具备渲染效率高的优点。

2、该组件支持自定义样式，通过调整样式规则能够使得其风格与 XUI 组件库的风格整体相近，从而保证一致的 UI 风格。

3、在 XUI 官方的 demo 示例程序中，有提供载入该组件的例程供参考，一定程度上可以避免不同组件库间的兼容性问题。

与此同时，客户端开发除了考虑界面效果，还需要实现的一些功能，为此还需要引入其他的第三方支持库，例如：

1、okhttp：okhttp 库封装了 http 请求的常见操作，如 get，post 等，与直接调用 android 原生的 api 实现相应请求相比，调用封装接口能够提高编程效率和代码的可读性。

2、fastjson：fastjson 是由阿里巴巴开源的一款 json 序列化和反序列化支持库，它提供了类实例与 json 序列化文本间的相互转换功能，在调用服务器接口及解析服务端数据时都需要使用。

其他的一些第三方支持库随项目开发进度的推进根据实际的需要再进行补充。

6.3、客户端安全：

由于 java 语言自身的特性，编译所产生的字节码如果不经过任何特殊处理可以非常轻松的反编译出源码，为了避免这种情况发生，我们考虑将编译产生的 apk 进行加壳保护。市面上所提供的免费加壳服务有梆梆加固、360 加固等，届时将视实际情况进行选择。

6.4、服务端开发：

服务端开发主体上选用基于 python 的轻量级 web 框架 flask 实现，遵循 restful API 的风格规范，为客户端提供接口。
数据库选用关系式数据库 mysql 作为 backbone 替换 flask 内置支持的 sqlite。之所以进行这样的选择，是考虑到 sqlite 作为轻量级数据库，拓展性较差，往往仅支持单机存储，不利于后期的横向拓展。
为了更好的操作数据库，我们选用 SQLAlchemy 库作为操作数据库的支持库。与直接执行 SQL 语句相比，使用这一类 ORM 框架的好处包括：
1、不需要过多的关注 SQL 语句的编写实现，还可以利用 ORM 框架自带的语句优化功能提高查询语句的执行效率。
2、避免手动拼接 SQL 语句过滤不严造成 SQL 安全漏洞，直接运用 ORM 框架提高了 web 服务的安全性。

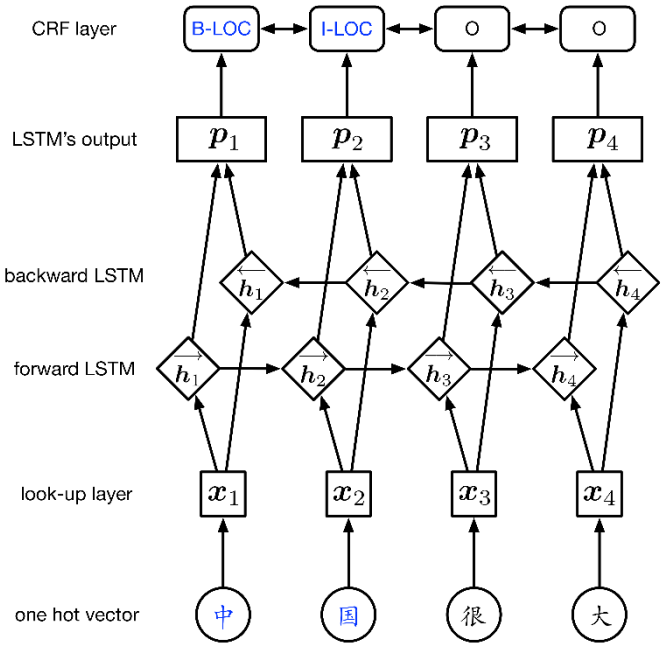
6.5、系统架构：

考虑到日程管理软件需要具备实时性的的功能，即事件推送需要准确靠谱，涉及消息推送的部分我们选用 RabbiMQ 信息队列来实现。
RabiitMQ 是一套开源的消息队列服务软件，同时提供了各编程语言实现的调用接口。在本项目中，我们会将服务端作为消息队列的生产者，同时另外实现一个推送信息发送程序作为消费者，生产者根据用户提供的日程计划将日程信息添加进消息队列，推送消息发送程序定时读取消息队列中的待处理消息，并对符合条件的日程项发送推送消息，及时提醒用户。
与此同时，为了减轻后端数据库的压力，对于高频访问的日程数据，我们选用 redis 作为我们的缓存数据库。Redis 为非关系式数据库，能够储存 key-value 型的键值对数据，对于一个查询请求，后端程序会先查看 Redis 中是否有缓存的查询结果，如果有并且是在缓存有效期内的，那么就直接使用，否则的话才去 mysql 数据库中执行查询，可以减轻数据库的访问压力。

6.6 智能识别日程解决方案

为了进一步优化用户的使用体验，我们对新建日程的流程做了全新的优化，用户可以通过粘贴一段文本（如微信群的通知），软件能够自动的识别这段文本中的日程要素，并抽取时间、

事件等关键因素自动填充。为了实现这个功能，我们拟采用如下的方案：
运用 NLP 中的 NER 命名实体识别技术，自动识别一段文本中的日程要素。经过调研现有的开源项目，我们选取了如下开源项目模型作为我们的基础模型：
Zh-NER-TF(https://github.com/Determined22/zh-NER-TF)
该模型的结构如下图所示：



关于训练数据集，我们拟采用现有的开放数据集：MSRA corpus
后续如果项目需要，我们还将考虑收集数据集手动标注扩充样本空间。

在模型训练完后，我们有两种模型落地方案：
1、 由客户端发送请求服务端返回识别结果。
2、 由客户端本地进行识别。

如果是由服务端识别，那么意味着这一项核心功能在离线状态下将不可用，用户的体验可能会有一定的影响。
如果是在客户端识别，那么模型文件有可能会被其他窃取。
具体的做法后续再做决定。

如果是在客户端识别，那么还需要考虑移动设备的计算性能问题。为此，我们将采用腾讯开源的移动端深度学习推理框架 ncnn(https://github.com/Tencent/ncnn)加速优化

6.7、项目部署：

为了更好的践行 devops 的理念，我们选用 docker 容器部署技术作为我们后端服务最终上线的部署方案，运用 docker 容器，可以方便快捷的打包开发好的服务应用并且部署到其他的 linux 服务器上，避免了繁琐的各类配置。

最后，完整的技术解决方案示意图如下：