# 项目简介

**创意来源**

随着移动智能设备的普及，人们彼此之间交流的手段也越来越丰富了。短信、邮件、微信、公司或团队协作使用的软件……人们使用的软件越来越多，得到消息的渠道越来越多，随之而来的是一个问题：如何管理自己的待办事项？如何把分散在各处的信息整合到一起？为了应对这个问题，市面上出现了层出不穷的TODO软件，像是wunderlist，todoist，以及各个系统自带的待办事项，目的就是方便用户进行备忘。

然而使用这些软件其实本身就是一件很不方便的事情，因为它们都要求操作者自己一步一步靠着点击“新建”、“选择时间”、“选择地点”、“输入内容”才能完成一条备忘录，这些操作及其繁琐机械。稍微有些进步的可能会通过手机上的智能语音助手来辅助输入，然而从操作习惯上来讲，在公众场合进行语音输入并不符合中国人的含蓄的个性，而且在图书馆会议室等一些地方语音输入也会受到限制。

我们的创意点在于，结合传统手动操作的隐私性便捷性，以及语音识别处理中的能自动判别语义的自动化特性，为用户带来一款真正好用易用并乐于使用的备忘录工具。

具体来说，我们团队设想出了这样一种方法：用户利用屏幕截屏截取与备忘录相关的部分，我们的程序就能通过OCR(Optical Character Recognition)来识别出上面的文本信息，再利用NLP(Natural Language Process)来解析出文本中的语义，从而省去用户的手动输入，自动生成一条备忘录。当然，我们的目的是简化用户的输入流程，所以也支持用户直接进行文本的输入，或者在后续版本中可以加入对手写笔的支持，给用户带来更好的体验。

在创新性和原创性方面，我们调查分析了市场上存在的一些备忘录、TODO类软件、笔记软件，发现绝大部分都是通过用户分项逐个输入或者语音输入。通过识别文字输入的软件也有，比如三星的Spen应用就支持用户手写后识别成文本再进行“动作”（如拨打电话，发送邮件等）。但是从总的来说，它并不能应对复杂文本进行复杂的语义分析，同时，它的输入还是通过用户输入，只是从键盘换成了手写。可以说，我们的应用在创新和原创性上面还是做得比较好的，尤其是在输入方式上，圈划非常符合人的直观操作，且简单方便，把复杂的事情交给计算机，还给用户一个易用的备忘功能。

**目的意义**

本项目主要要完成的内容有以下几点：

1. 完成从图片到文字的转换，通过各种手段（如机器学习等）提高OCR的精确度
2. 从文字中提取出相关的语义信息，如时间、地点、事件、人物、电子邮件、手机号等
3. 探索一种新颖的用户输入信息的方式（通过截屏圈点的简单操作完成复杂的输入）
4. 完成应用的客户端跨平台开发，包括windows PC端和phone端，android端
5. 完成同一个用户账户在多个客户端之间的同步问题

**开发环境**

操作系统：windows 10

数据库系统：mysql

IDE: Visual Studio 2015 Community, Android Studio

版本管理工具及平台：Github

测试工具：VS2015 unit测试工具、junit工具

平台：Azure(替代方案：阿里云、自搭建VPS)

**技术路线**

我们打算利用微软提供的project Oxford的API作为算法运行的一个后端，将Azure云服务作为账户间同步的后端。前端即客户端部分，一是我们打算在Android Studio上利用谷歌提供的SDK开发安卓端；二是我们打算在VS上利用UWP平台一次性开发Windows的PC端和手机端。前端和后端的交互考虑到前端跨平台，计划使用HTTP请求来完成。

前端实现

安卓端将采用Java语言实现，目的是编写基于Android系统的智能手机上的备忘录App。我们选用官方推荐的Android Studio作为IDE进行开发，SDK计划使用6.0版本，目标兼容版本号4.4以上的安卓系统。

Windows的PC端和手机端采用C#语言实现，并且采用微软的UWP框架，完成一个横跨PC和手机端的备忘录应用，并且根据PC和手机的操作的不同采用不同的交互逻辑。我们计划采用Visual Studio 2015的社区版作为IDE开发。

后端实现

该应用的后台计划采用Java语言实现，并通过REST架构进行和客户端的交互。由于Java的全平台兼容性，我们的服务端的机器可以是Azure也可以是阿里云等服务，甚至更基础的VPS，这使得我们的迁移成本大大减少。采取REST架构是因为HTTP式的前后端交互可以使得多平台的客户端成为可能，而且将来可以支持更多的前端，如网页和ios系统等。后台所负责的工作为解析用户上传的图片，通过OCR和NLP处理之后，返回客户端我们自定义的备忘录的数据结构。另外的工作是统一管理用户的账户，并且将同一账户的不同的客户端进行同步。

算法设计

本应用所需要的核心算法是OCR算法和NLP解析算法。考虑到团队成员的技术有限，我们主要是希望通过微软的project Oxford完成这两部分的工作。鉴于我们的图片来源主要是屏幕截取，所以OCR算法的识别准确率一般比较高，但是我们的试验表明OCR模块并不会关注上下文的信息，所以当一行文字被截取一半的时候会返回乱码。而NLP的模块也并不能自动对乱码进行筛选处理。

所以我们打算在发送图片之前，通过简单的图像处理技术对图片选择的范围进行一个预处理，例如将用户裁剪了一半的文字根据包含在整个图片的程度选择扩大范围或者缩小范围，更精确地裁剪屏幕截图的一部分。

在OCR到NLP之间也计划完成一些比较简单的文本的预处理，例如去除OCR返回中常识别错误的部分，像project Oxford中经常将时间的“16：04”识别成“16℃ 4”，这时我们可以加上一些上下文的信息判断改正，从而提高整个识别的精确度。

在多个客户端的同步方面，我们打算采用一个消息队列的机制来完成多个客户端的同步，当某个客户端连接网络时，自动检查判断当前的客户端的队列里面比服务端多出哪些消息，缺少哪些消息，将两者之间做一个合并的操作。

**预期成果**

完成SnapMemo的开发，在学校范围内征集试用者，获取用户的反馈。后面还可以将SnapMemo投往其他的比赛，包括技术性的比赛和非技术性的，比如创业竞赛。待SnapMemod的完成度比较高的时候正式在各个安卓市场，以及微软官方商店上推出市场版本。