

WITHYOU

IOS APP 介绍文档

第一版：2020年4月

第二版：2020年7月

总共页数：18

撰写：史瑞琪 邓萌达

withYou，取与子同袍之意

愿每一位来到这里的人，都不再是一座孤岛

愿每一个在黑暗中踽踽独行的时刻

你的身边，都不再是空茫一片

目 录

一. 摘要 (3-4)

- 前言 (3)
- 内容 (3)
- 目标 (3)
- 方法 (3)
- 实现情况(加粗为第二赛段加入部分) (4)
 - 已实现功能 (4)
 - 未实现功能 (4)

二. 过去与未来 (5)

- 立项背景 (5)
- 应用前景 (5-6)

三. APP成果展示 (6-15)

- 引导动画 (6)
- 注册 (7)
- 登录 (8)
- 主页 (9)
- 消息系统 (10)
- 个人信息 (11)
- 添加双人任务 (12-13)
- 双人任务表 (13)
- 添加个人任务及个人任务界面 (14)
- 好友 (14)
- 整体框架 (15-16)

四. 结语 (17)

二. 摘要

前言

本工程是我和我的partner从空工程一点一点构建起来的，面对一个刚发布半年的新构架SwiftUI：资料缺少，IOS相关开发经验缺乏，是我们面对的困难。但是作为两个拥有apple全家桶的资深果粉，这个过程又是极度令人享受的。苹果的美学贯彻在编程语言中，工程中，实体中，也是我们想要贯彻到我们自己作品中的灵魂。这次比赛将成为我们两个的一个契机，一个步入IOS开发的契机，**希望我们能在这条路上走得足够远。**

在第二赛段，主要完成了前端和后端的分离，更换了数据库，按照后端借口重构了前端的功能。新加入消息系统，可以由服务端发送信息给客户，提醒任务匹配进度；也可以由客户发信息到服务端，加强反馈；同时后端加入了匹配机制，进行用户的匹配。

内容

本app是基于SwiftUI的适配于苹果iOS系统的移动app。

核心思想是“陪伴”。应用为不同用户进行匹配，令他们同时完成一个schedule，在完成的过程中，双方每天的计划表和完成情况是对对方可见的，并且可以在完成一项任务时进行留言，互相鼓励。

核心功能是双人共享任务表。平台提供四种类型的任务表进行选择（学习，健身，生活，读书），同时为选择共同任务表的用户进行匹配。

辅助功能还有好友功能，社交广场，单人时间表，定制时间表等。

目标

随着互联网的普及，网络社交增多，但是有效社交却在减少。现在的社交平台提供的大多是供个人发表看法，获取信息的服务。但是，一个人能够获得真正的，有效的陪伴，却所剩无几。所以，年轻人逐渐变得内敛，活在自己的世界，变得越来越“孤独”。

聊天软件，朋友圈，微信，抖音，这些软件并不能消除孤独感，反而，只会更加增加“孤独”。真正能够消除孤独的，是充实的生活和有效的陪伴。

使两个人能够做同样一件困难但是有意义的事情，并在完成这件事享受到充实的同时，享受他人的陪伴，分享完成的成就感，减少单人工作时动力不足的问题，消除孤独感，是本app的核心目标。

方法

本app基于SwiftUI进行开发，个别功能借用UIkit在SwiftUI中的接口进行实现。

后端利用go实现。

数据库选用进行实现。

实现情况（加粗为第二赛段完成部分）

已实现：

- 引导动画
 - 时间线以及动画效果对文字与图形的控制
 - **修改UI**
- 视图之间的切换
 - sheet, NavigationLink, view, alert的多种视图模式的使用与切换
 - **加入登录加载视图，并且为了实时刷新加入刷新视图。**
- 登录注册
 - 与后端接入，判断登录是否正确匹配、注册时邮箱是否冲突。出现冲突进行alert提醒
 - 注册时姓名邮箱是否符合正则表达式要求
 - **对UI界面进行修改，对接新UI特点的圆角属性**
 - **登陆时获得本人信息的同时，获取好友信息，并且在匹配期间尝试获得匹配情况**
 - **链接后端并且进行数据交互时，加入登陆加载界面，界面显示头像**
- 个人信息
 - 根据注册信息初始化信息。
 - 可修改添加更多信息并接入后端（头像，姓名，签名，密码，生日，星座，血型）
 - 对填入信息判断是否符合正则表达式要求
 - **对于信息修改进行完善，因为需要进行对后端的数据请求，因此进行了对界面实时刷新的优化**
- 任务表
 - 可切换私人任务与双人任务
 - 私人任务可自己添加定制自己的时间表
 - **双人任务可进入查看细节，进行评论，查看partner的任务评论，评论接入后端**
 - 对完成任务进行勾选，表示删除，更新数据。
- 添加新的任务表
 - 选择任务表类别（健身、学习、生活、读书）
 - 不同任务可以查看详情（每日完成时间，内容介绍）
 - 匹配等待动画，匹配成功提醒，成功不可再次匹配提示
 - **后端添加匹配逻辑，对所有的账户进行匹配检索，并且发送提示信息到个人的收件箱**
- 好友
 - 可查看好友信息
 - **将匹配成功的partner加入好友**
- 消息系统
 - **可以由服务端发送信息给客户，提醒任务匹配进度**

- 可以由客户端发往服务端，反馈使用问题

待实现：

- 不同机型的适配
 - 由于前期缺少经验，导致在快要完成时才考虑到这个问题，因位置限制多用具体距离，导致现在只适配iphone 11 pro max。在之后的完善中会进行比例化约束
- 任务完善
 - 任务的内容亟待打磨

三. 过去与未来

立项背景

随着互联网的普及，网络社交增多，但是有效社交却在减少。现在的社交平台提供的大多是供个人发表看法，获取信息的服务。但是，一个人能够获得真正的，有效的陪伴，却所剩无几。所以，年轻人逐渐变得内敛，活在自己的世界，变得越来越“孤独”。聊天软件，朋友圈，微信，抖音，这种类型的软件并不能消除孤独感，反而，只会更加增加“孤独”。真正能够**消除孤独**的，是充实的生活和有效的陪伴。有效陪伴，即可以相互鼓励共同完成一件有意义的事情。

缺少动力是步出学生时代，进入大学或者社会时，很多人所面临的问题。这种缺少动力的现象，在一定程度上是缺少显形竞争者的结果，因此，很多人都需要一种来自身边的鼓励，不管是陪伴的鼓励，还是竞争的激励，同时做一件事可以有效**提高效率**。

“陪伴类”app在市场上并不算少，并且在近些年来一直有着很大的发展空间。比如，健身陪伴类app keep，学习专注陪伴类 Forest，然而这些app大多是无交互的，是一个虚拟的形象帮助用户完成目标。交互更多一点的，大多属于游戏，比如独树一帜的生命线和爆火的游戏 旅行青蛙。后者正是因为他给人一种陪伴感，且并不会占用用户很多时间，收到了很多年轻人（包括我和身边的人）的喜爱。

应用前景

对于陪伴，我有过很多思考，游戏类的居多。但是在疫情期间我偶然在b站直播区看到有人直播学习，直播间动辄上万的热度，很多学生党、考研党在这里打卡，相互鼓励，对我有了很大的触动。游戏可以给人提供陪伴，弥补空虚的心灵，但是如果是一种**鼓励人们利用时间而不是消磨时间的应用形式，能够为现代人在提供陪伴的同时，给予他们更多的力量**，这样的app，定然是同时具有社会价值和商业价值。

受众方面，对于学生，竞争和陪伴有利于督促学习以及提供动力；对于那些拖着疲惫的身躯孤身一人穿梭于城市之中，因为种种原因，还没有条件寻找灵魂伴侣的人，他们更需要一个代价更小的社交方式，这正是我们可以提供的。

商业化方面，以不同的任务为单位的收费单元可以带来收益，因为受众的特性（白领，大学生，知识性工作者），这方面的前景还是比较宽敞的。

基于以上思考，我和我的伙伴制作了这款app。

withYou，取与子同袍之意，愿每一位来到这里的人，都不再是一座孤岛，愿每一个在黑暗中踽踽独行的时刻，你的身边，都不再是空茫一片。

四. APP成果展示

1. 引导动画

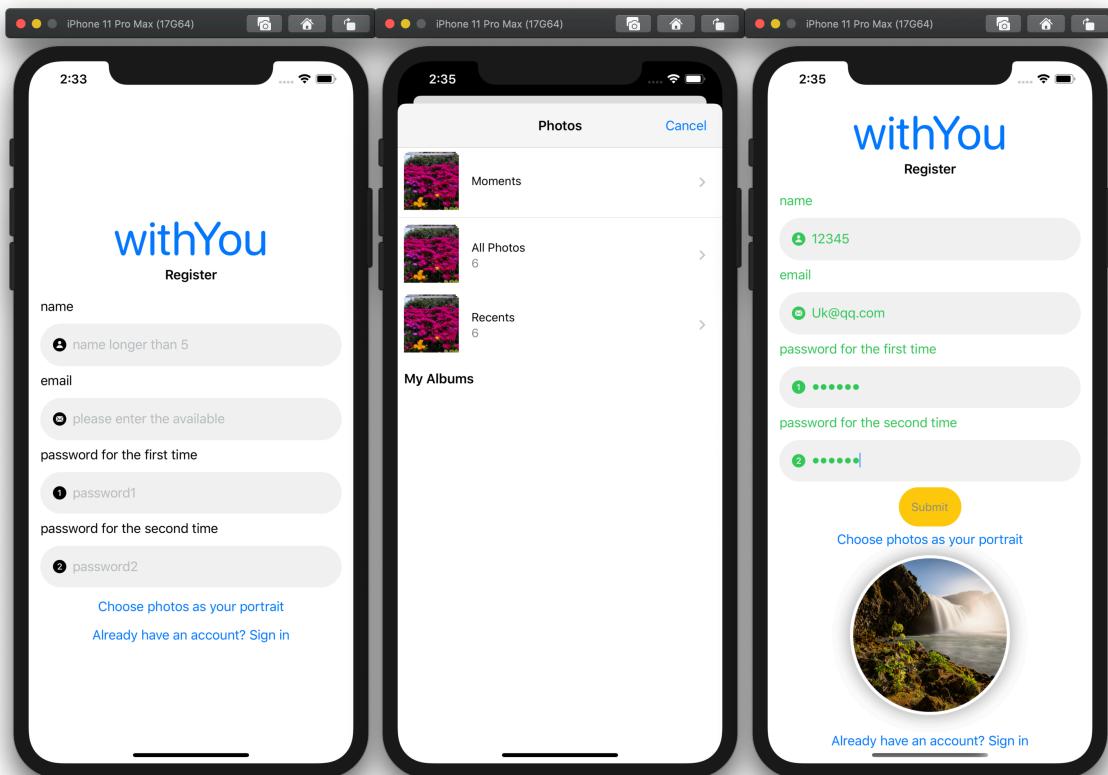
动画效果可以在首次进入app时播放，动画内容是：黑色的“i”“Y”代表着两个独立的人，两秒后，“w”“th”“ou”出现，讲“i” “ou”拉到一起，同时颜色由黑色变成蓝色，同时浮现“Life is better”，凸显本app“withYou”，也侧面说明本app的主旨：**生活会因为陪伴变得更美好。**

两秒后出现let's Start的按钮，可进入下一步注册或者登录。



2. 注册

注册页面需要正确填入姓名格式（大于5个字母），有效email（符合email格式并且在数据库中未被注册），两次相同是密码，以上三项满足后弹出submit的按钮。头像不是必须的（可从自己手机选择照片），也可在进入app后修改。如果已有账户，也可以直接登录。当邮箱重复时可弹窗提醒。



发送头像：

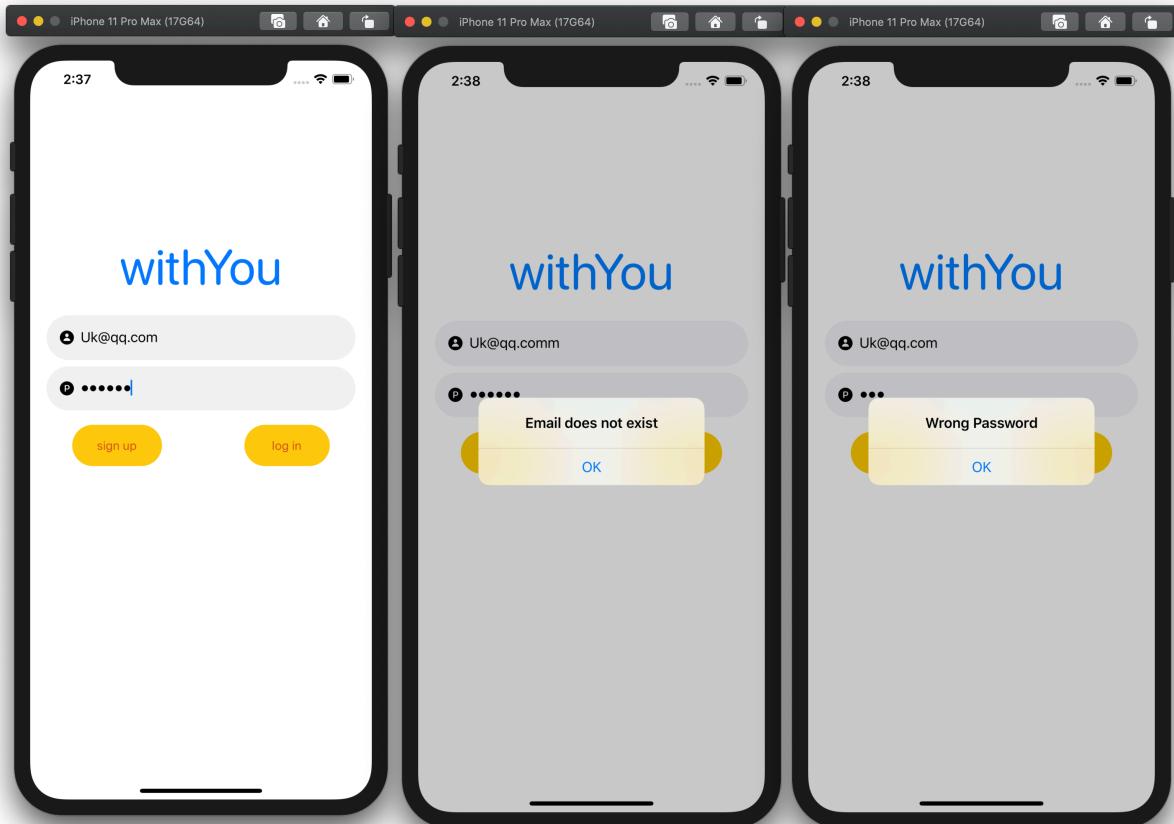
头像有时会出现过大的情况，这个时候进行发送可能造成时间过长，则应进行压缩后进行发送。同时，因为要通过NSURL进行图片定位，还需要对头像的地址进行转化，然后利用已经写好的接口进行发送。发送用到了Alamofire库。

```
import Foundation
import Alamofire
import SwiftyJSON
import SSZipArchive

func PostRegister(completion: @escaping (_ code: Int, _ msg: String) -> (), name: String, email:String, password: String) {
    var code = 100
    var msg = ""
    print(getSize(url: PostImagePath!))
    print(type(of: PostImagePath!))
    AF.upload(multipartFormData: { (MultipartFormData) in
        print(type(of: PostImagePath!))
        MultipartFormData.append(PostImagePath!, withName: "file")
        MultipartFormData.append(name.data(using: String.Encoding.utf8, allowLossyConversion: false)!, withName: "name")
        MultipartFormData.append(email.data(using: String.Encoding.utf8, allowLossyConversion: false)!, withName: "email")
        MultipartFormData.append(password.data(using: String.Encoding.utf8, allowLossyConversion: false)!, withName: "pass")
    }, to: "https://withyou-cetaciis.dev/api/register")
    .responseJSON { (response) in
        let json = JSON(response.data!)
        code = json["code"].intValue
        msg = json["msg"].string!
        print(code)
        completion(code, msg)
    }
}
```

3. 登录

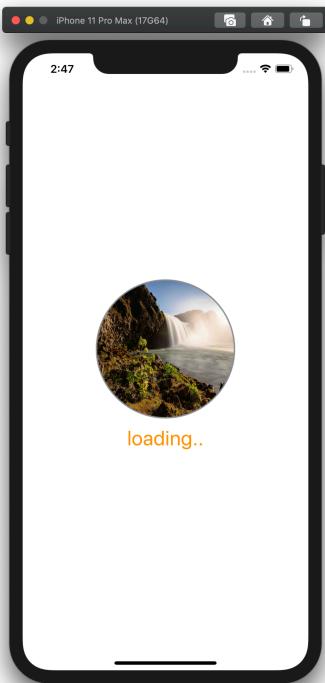
登录时可检测邮箱密码的匹配情况，如果出错则弹窗提醒。



登陆时，向后端发送请求，进行匹配，如果密码账号匹配成功，返回账户信息存在本地。同时，获取本人匹配信息。获得用户信息接口如下：

```
import Foundation
import Alamofire
import SwiftyJSON

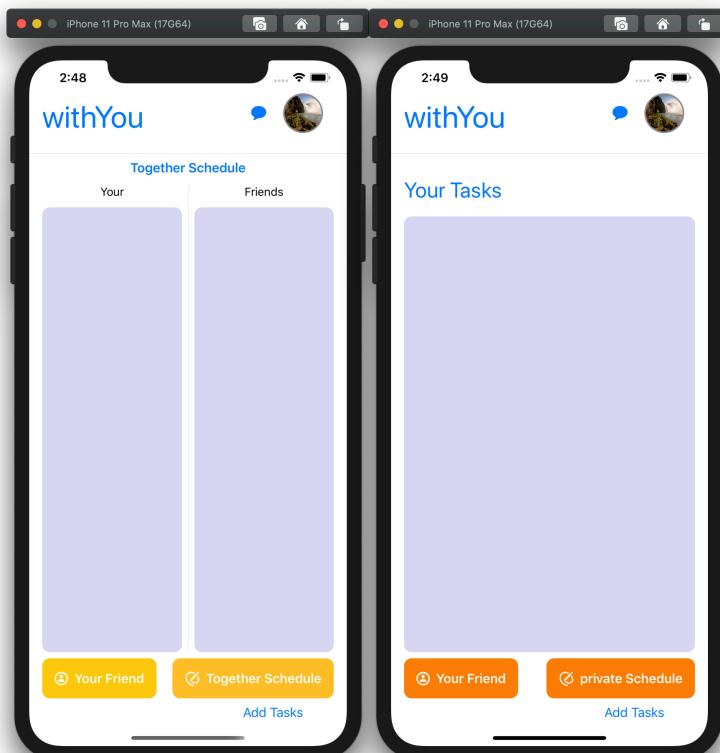
func PostGetUserInfo(completion: @escaping (_ RtData: UserInfo) -> (), email:String, pass: String) {
    var UserData: UserInfo = UserInfo()
    let para = LoginInfo(email: email, pass: pass)
    AF.request("https://withyou.cetacis.dev/api/UserInfo",
              method: .post,
              parameters: para
    ).responseJSON { (response) in
        // print(json)
        do {
            UserData = try JSONDecoder().decode(UserInfo.self, from: response.data!)
            completion(UserData)
        } catch {
            print(error)
        }
    }
}
```



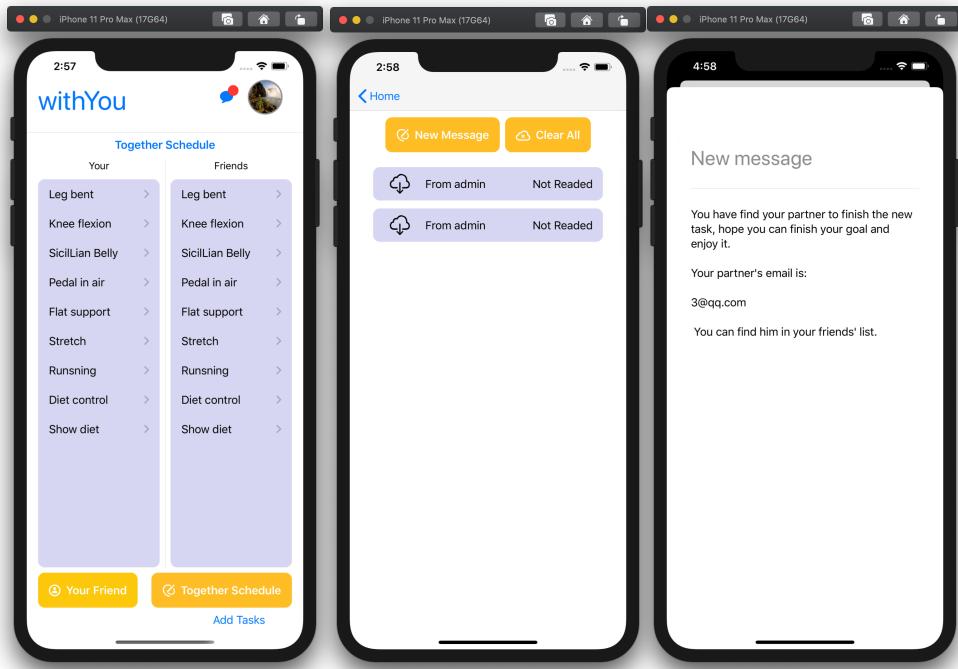
完成信息的获取的同时，加入加载动画。

4. 主页

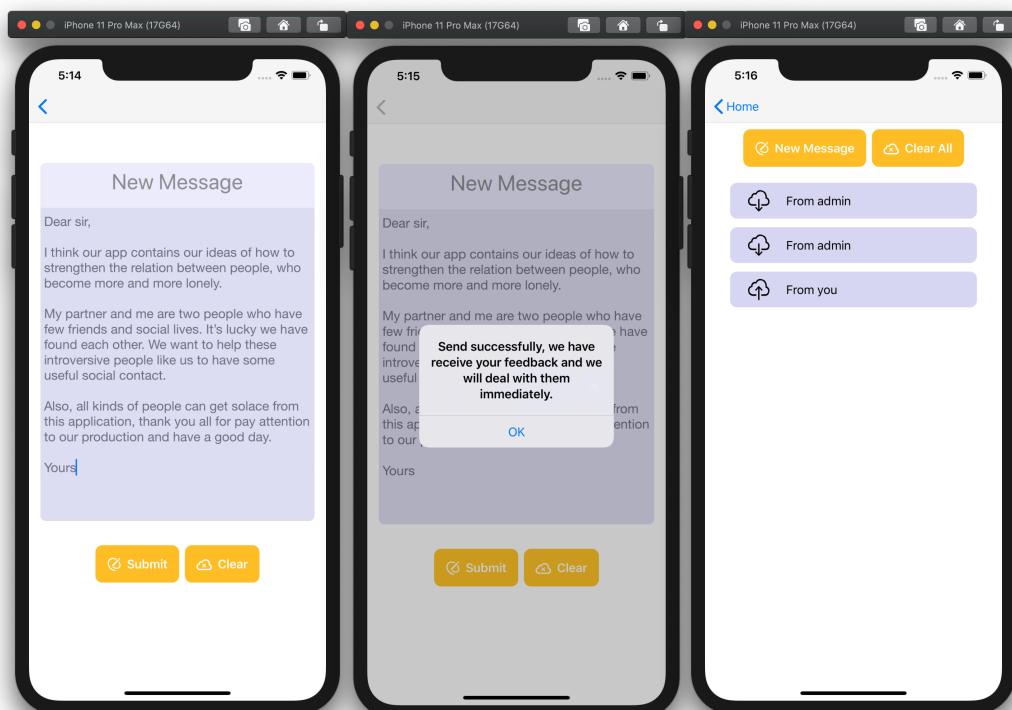
主页主要包括六部分，消息部分（弹出消息列表，消息发送），个人信息（点击头像弹出，可查看详细信息，也可修改信息），任务部分（信息详细），好友列表按钮，切换个人/双人任务按钮和添加任务表。



5. 消息系统



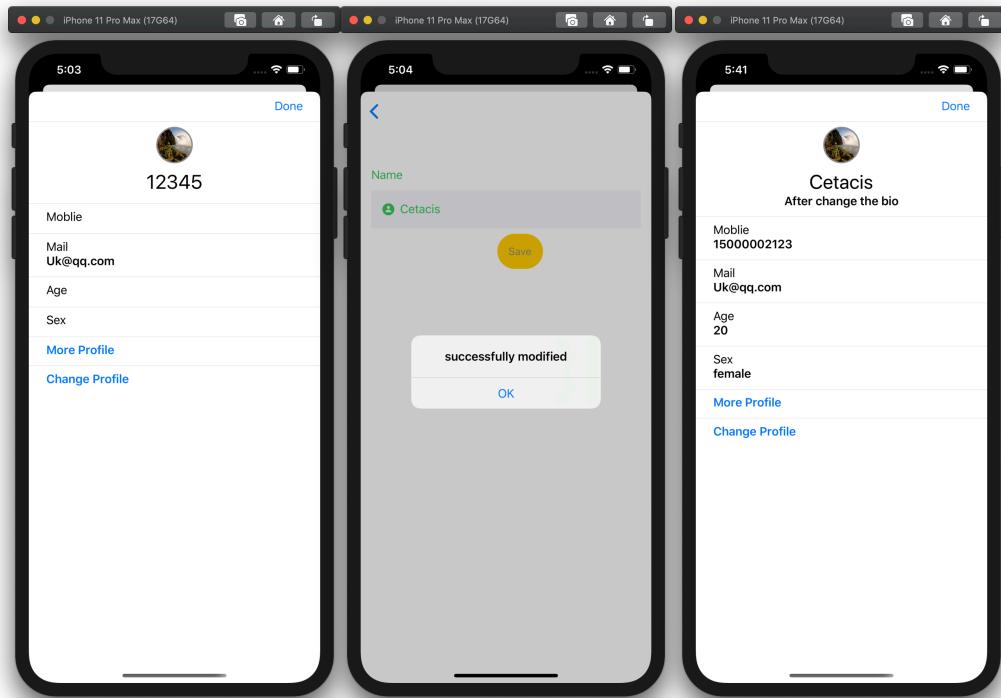
收件：如果收件箱里有未读的信息，就会有在主页提示，读完信息后，则会去掉not readed标记。可以通过clear all，一键删除全部消息。



写件：用户可以向服务端发送反馈信息，在信息箱中也可以看见信息。

6. 个人信息

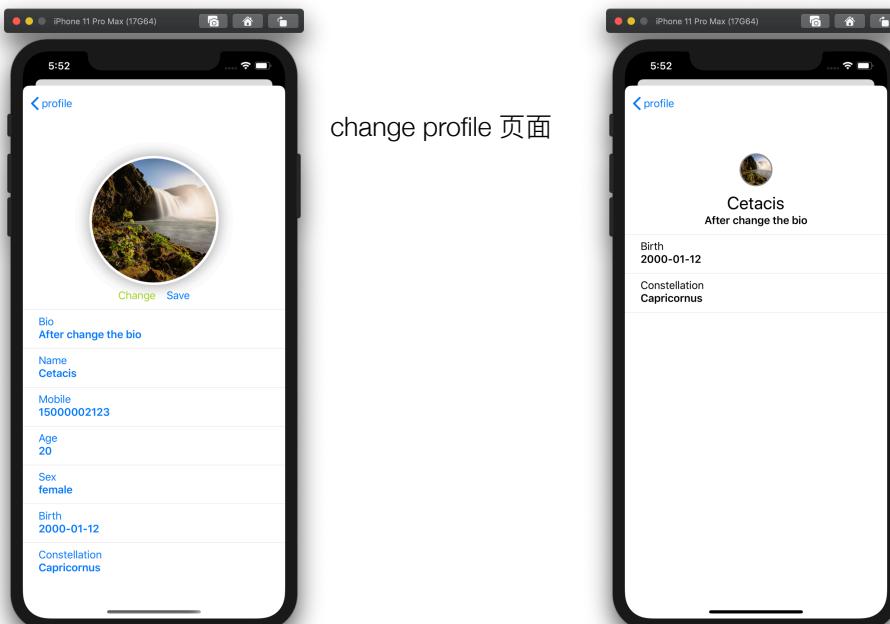
点击头像可弹出个人信息，在这个界面可以查看更多信息也可以修改信息，修改的结果会同步到数据库。在修改的过程中依旧要注意所修改的数据是否符合数据格式，如果不合适，依旧会弹出alert提醒。



打开个人信息

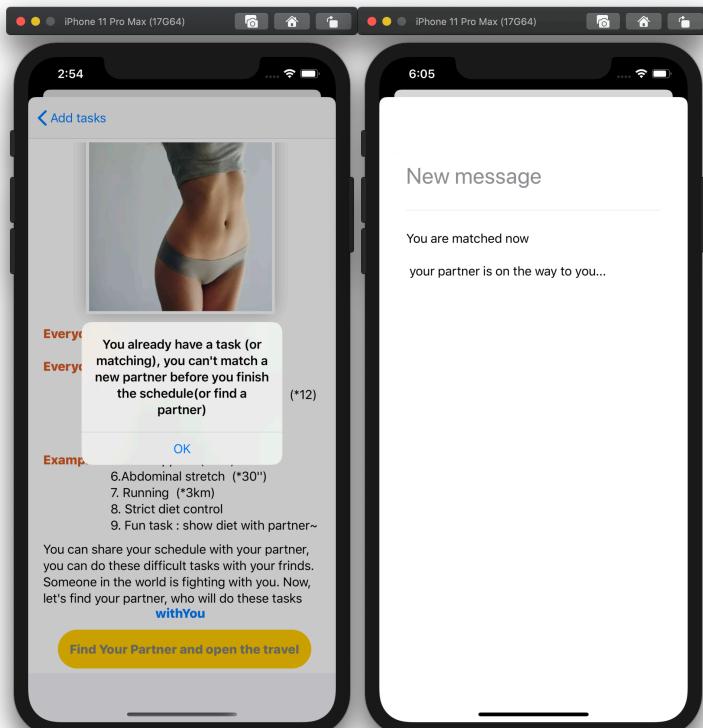
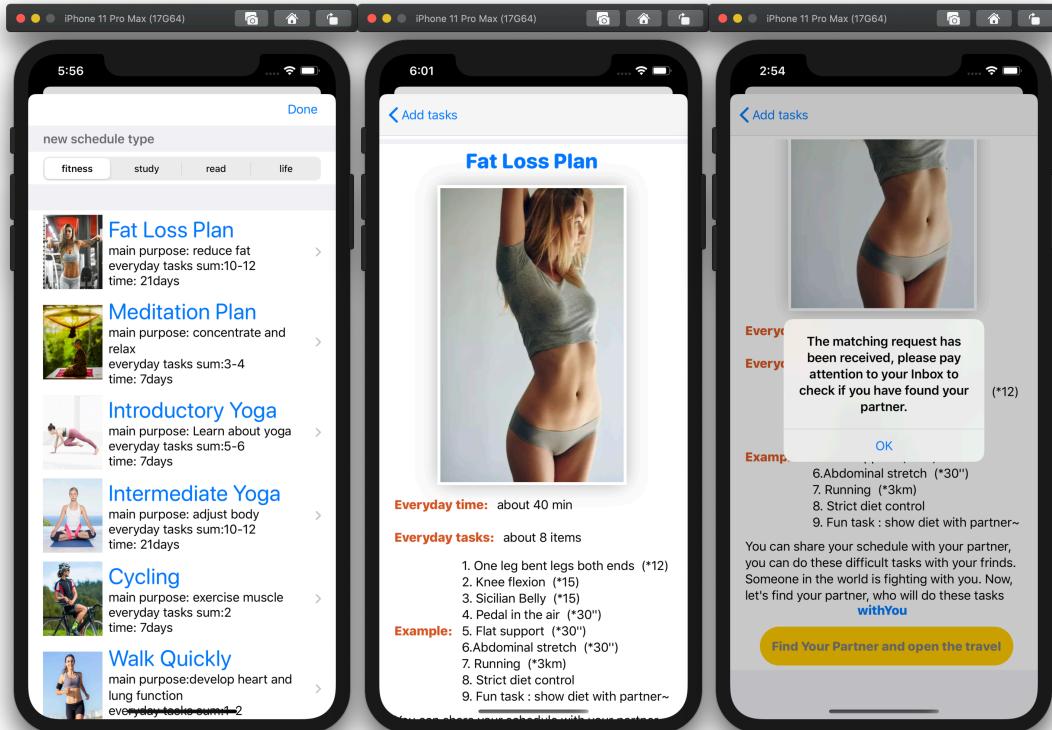
修改

修改后的个人信息页面

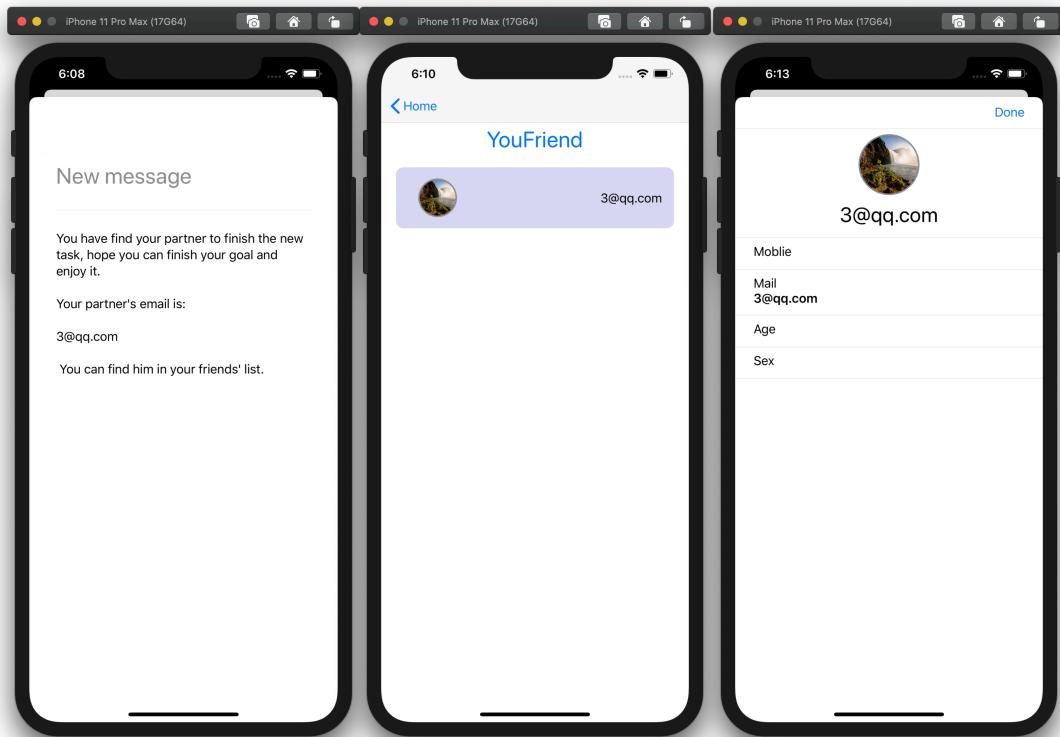


7. 添加任务表 (together)

在主页点击add task即可进入添加任务的sheet，这里演示fitness模块，选择任务，查看细节，进行匹配。



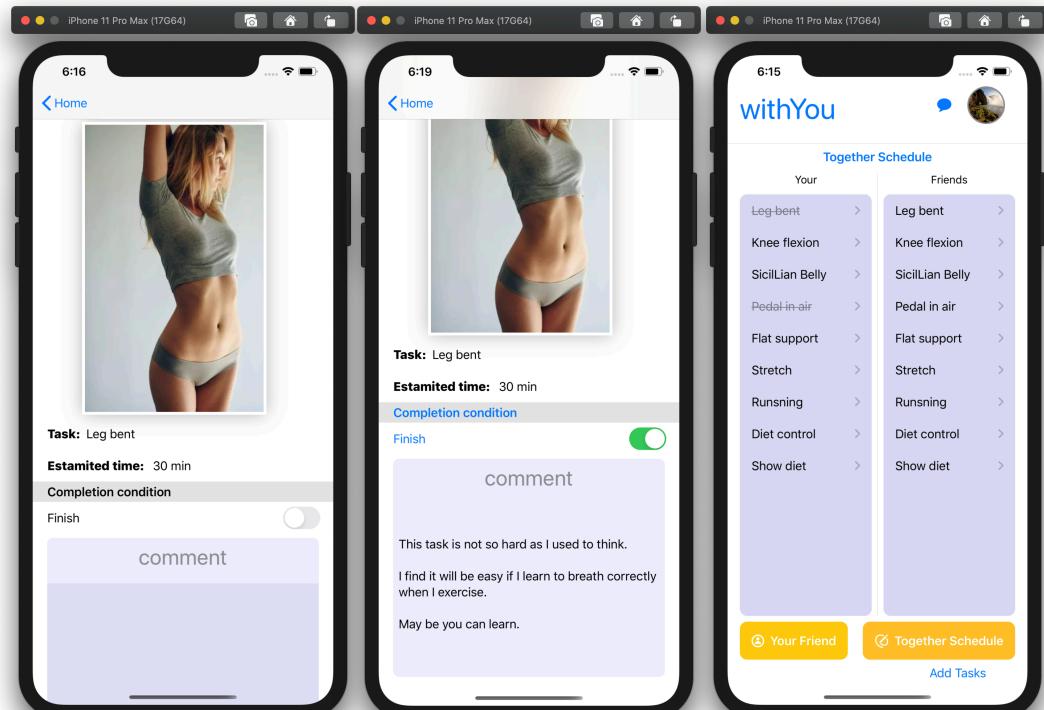
如果成功进入匹配队列，会收到相应的message，提醒匹配正在进行。如果在此时再次匹配，会提醒正在队列或者已经匹配成功。



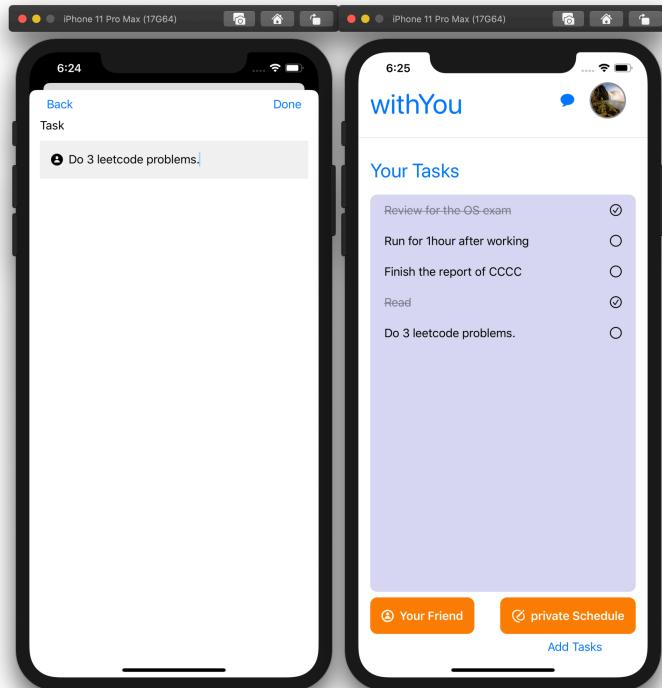
当成功匹配，系统发送信息到个人信息，并且自动加入好友系统，在这里可以看到好友的信息

8. 双人任务表 (together)

6中最后一张示意图即为双人任务表的样子。点入自己的任务，可以查看任务细节，并且在做完任务后发表自己的评论和看法，同时，进入对方的任务，可以查看他的评价和完成情况。（实际上，在主页就可以看到一个任务是否已经被完成被完成的任务会被划掉）

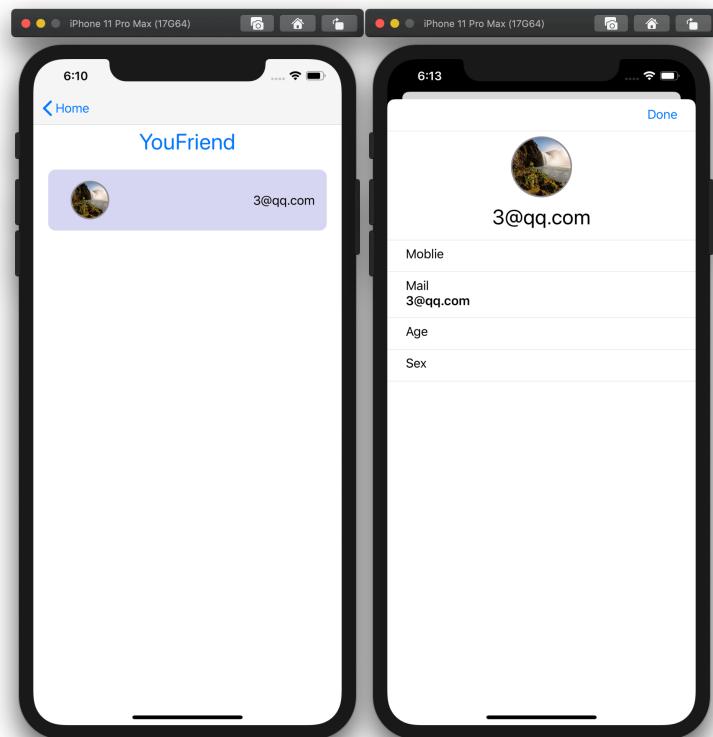


9. 添加任务表-个人任务表 (single)



个人任务可以自主添加，并且通过勾选代表完成。

10. 好友



匹配的好友会在Your Friend中查看简介（下图显示的是之前那次匹配到的好友）

11. 整体框架

用户的登录与注册

- IOS 发送请求， 提交用户信息
- web go 后端处理， 返回

匹配

- IOS 发出请求， 提交匹配信息
- web go 后端匹配完成后发给ios
- 单独开一个 小铃铛（通知） 获得系统公告和匹配状态

任务

- IOS 修改/添加任务时通过API发送给后端修改
- 每次onload （可以做一个下拉刷新）的时候GET服务器任务信息

匹配

- 初步想法是做一个队列
- 如果队列为空则塞入队列
- 如果队列中有等待匹配的信息
- 则依次检查队列中另一个用户与这个用户是否（为黑名单， 为指明不愿意匹配）， 如果没有满足条件则继续加入队列
- 如果没有，则匹配成功
- 然后服务器中加入这两个人的together task， 并且更新通知

数据库

- 采用mongodb
- 数据库结构如下：（和withyou 1.0 类似）

```
// private task
type PrivateTask struct {
    Name string `json:"name"`
    Number int `json:"number"`
    IsFinished bool `json:"is_finished"`
}

// together task
type TogetherTask struct {
    Name string `json:"name"`
    Number int `json:"number"`
    Comment string `json:"comment"`
    FriendEmail string `json:"friend_email"`
    IsFinished bool `json:"is_finished"`
}

//user info
type User struct {
    Username string `json:"username"`
    Email string `json:"email"`
    Mobile string `json:"mobile"`
    Password string `json:"password"`
    Age string `json:"age"`
    Bio string `json:"bio"`
    Constellation string `json:"constellation"`
    Blood string `json:"blood"`
    Birthday string `json:"birthday"`
    Sex string `json:"sex"`
    ImgPath string `json:"img_path"`
    TogetherTasks []TogetherTask `json:"together_tasks"`
    PrivateTasks []PrivateTask `json:"private_tasks"`
}
```

UI

- 增加通知栏 和 意见反馈栏

四. 结语

第一赛段：

从开始构思到写完文档，我和我的partner共同度过了十分有意义的二十天。我在其中收获良多。这是我第一次和别人这么深入的合作，真正体会到了github在开发合作中的应用，而不是只是将其当做云盘和素材库。

因为swiftui从提出到现在不过半年多，资料也显得更加少而新，在建立工程的过程中，我学会了如何搜索有效的资料，如何整合不同来源的资料，如何鉴别资料的有效性，如何利用现有的轮子造出自己的轮子，从英文到日文，从官方文档到各种blog... 这些都是我从前自课本中，自完善成熟的技术领域中无法学到的，也是我自己在平时的实践中缺少的。

看着文档的发布时间不过是几个前，看到所用库的star和folk只有几百个人，当我自己也可以在不同的技术贴下发表自己的看法，**我从swiftui中，感觉到了一种勃勃的生命力，一种对于未来的新鲜憧憬，让我也想为它做些什么，让我也想去做一些属于自己的东西。**

不过我们的工程还有缺陷之处，距离它能够真正承载我们的希望，在社会中发挥价值，还需要再走很长的路。

希望我们，还有它，都能够走得足够远。

第二赛段：

现在看第一赛段说的话，感觉真是中二啊，缺少文档的后果在第二赛段真真切切地感觉出来了。特别是和后端go庞大的社区比起来，利用swift实现前端接口确实让人费了一番功夫... 比如我用了异步发送post请求的库，但是我想同步发送，本来想要使用信号量达到目的，但是主进程会阻塞。我查这个问题的时候发现很多人都有这个问题，就是没有答案。最后还是在stackoverflow一个其他问题的角落里找到了解决方法。比如下面这个修改头像的api。

```
import Foundation
import Alamofire
import SwiftyJSON

func PostChangeProt(completion: @escaping (_ code: Int, _ msg: String) -> (), email:String, password: String) {
    var code = 100
    var msg = ""
    print(getSize(url: PostImagePath!))
    print(type(of: PostImagePath!))
    AF.upload(multipartFormData: { (MultipartFormData) in
        print(type(of: PostImagePath!))
        MultipartFormData.append(PostImagePath!, withName: "file")
        MultipartFormData.append(email.data(using: String.Encoding.utf8, allowLossyConversion: false)!, withName: "email")
        MultipartFormData.append(password.data(using: String.Encoding.utf8, allowLossyConversion: false)!, withName: "pass")
    }, to: "https://withyou.cetacis.dev/api/upload")
    .responseJSON { (response) in
        let json = JSON(response.data!)
        code = json["code"].intValue
        msg = json["msg"].string!
        completion(code, msg)
    }
}
```

在进一步完成app的过程中，还有很多很现实的问题出现，他们很琐碎，但是都是真实存在的，比如头像太大怎么压缩后发送，发送的链接类型是要URL还是NSURL，怎么在修改属性后实时刷新页面，并且在一些不会刷新界面的情况下刷新界面（比如sheet的后退，外界变量变化带来的刷新等），两个账号的同步，读者写者问题等……这些都是小问题，但是就是能一点一点将人的耐心磨碎。在这种情况下，解决问题的方法和思路显得尤为关键。但是我觉得最重要的思路，其实是不畏惧困难。

当一个问题出现，它意味着或者数个小时，或者一晚上的寻找答案的过程，这个过程本身就会带来很大的恐惧感，让人难以开始。在经历了很多次解决问题的过程后，我才慢慢发现，开始着手解决问题，才是解决问题中最难的部分。一旦开始，很多困难的事情，就不再那么困难了。

这个感悟真的是意外之喜。

虽说我现在看我几个月前写的话觉得很中二，但是我觉得还是不无道理，总有人要来解决问题，我不想只是成为一个只会google和查文档的工程师，我想成长为一个能够拥有缜密思维能写出来属于自己东西的开发者。

中二也没什么不好，是吧。