武汉大学计算机学院

实验报告

课程名称：移动编程技术

专业年级：

姓　　名：

学　　号：

协 作 者：无

实验学期：2021-2022 学年 2 学期

课堂时数：36 学时

课外时数：32 小时

填写时间：2021 年 6 月 16 日

目　录

一、 实验概述

(一) 实验项目名称

Skylark Delivery 外卖点餐系统

(二) 实验目的

设计一个餐厅点菜系统 APP，实现的主要功能页有两个 tab，一个是菜单，一个是订单。

菜单页功能包括：

（1） 选择餐厅功能：进入系统后，可以选择在哪家餐厅点菜；

（2） 餐厅点菜功能：选择某家餐厅后，可以浏览餐厅有哪些菜，每个菜品栏目有图标，名字；

（3） 菜品详情：点击某菜品后，可以查看该菜品详细信息，包括该菜品大图，单价，名字和详情介绍；在详情页可以将该菜品加入订单；加入订单后，在订单页 tab 上显示菜品数量。

订单页功能包括：

（1） 所有选择的菜品的详细列表；

（2） 可以对该列表中的菜品进行删除；

（3） 确认后显示一个欢迎页面，表示该用户已经选择好；确认返回主菜单。取消后返回上一级。

所有数据包括餐厅，菜品，订单信息全部存储在 Sqlite3 数据库

(三) 实验环境

虚拟机操作系统：macOS 10.13 High Sierra （64bit）

使用软件：Xcode

(四) 参考资料

老师的课堂讲解

老师的录屏材料

《精通 ios 开发 第 8 版》周庆成

https://www.jianshu.com/u/40dca9fcba01

https://developer.apple.com/tutorials/swiftui/building-lists-and-navigation

CSDN、博客园、stackoverflow 等网站内容

二、 实验内容

(一) 实验方案设计

1. 程序的主要模块

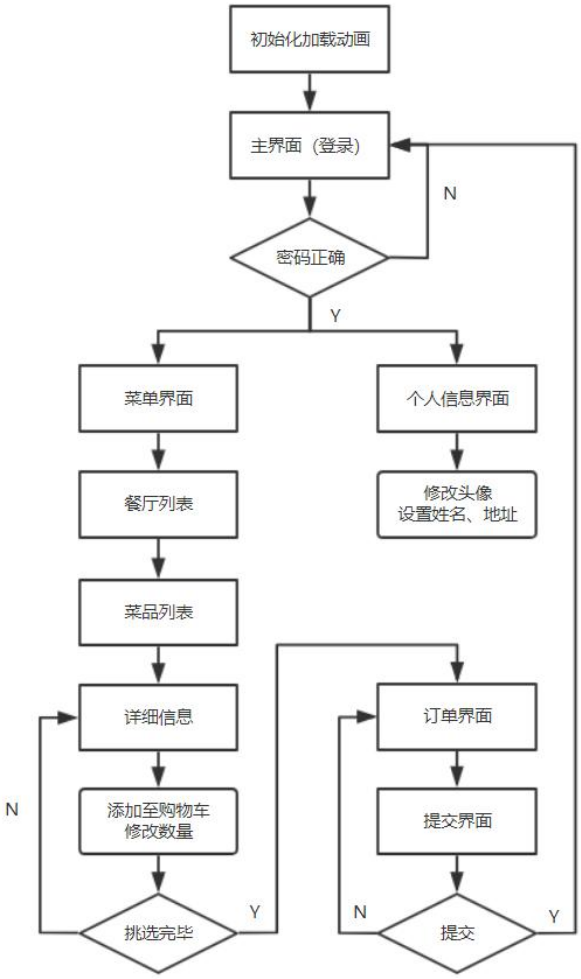

图 1

根据实验目的，画出基本程序使用流程图，如图 1 所示，根据流程图在 Xcode 中进行页面基本跳转逻辑布局设计。

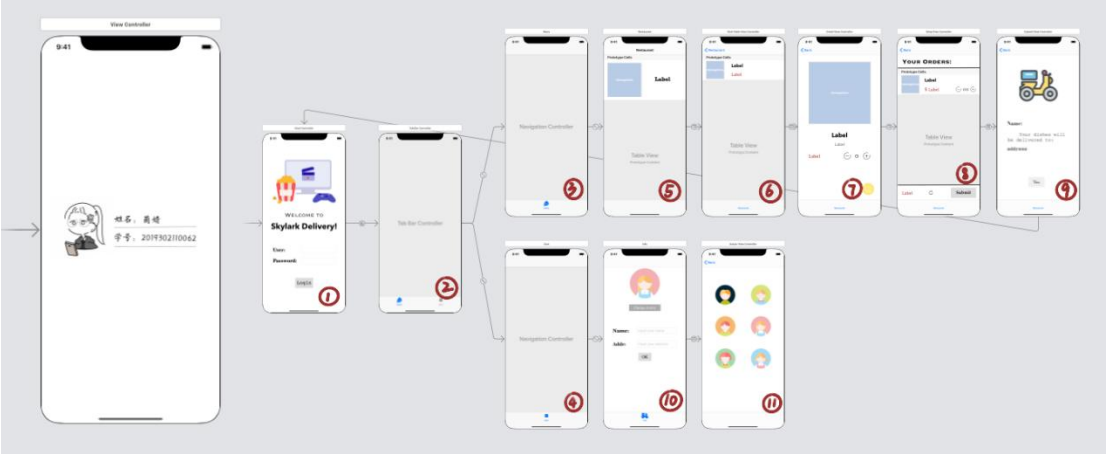

图 2

该程序的主要模块和页面分布如图 2 所示，左侧是初始化的加载动画。界面 1 是程序的主界面，同时是登录界面，只有正确输入设置好的账号和密码时才可以顺利登录，点击登录按钮跳转至界面 2，该界面是一个 table view，整体来看分为两个 tab，分别为 menu（菜单）模块和 info（个人信息）模块（订单界面通过按钮跳转界面进行实现，在程序完成之后才看到要求其中一个为订单），tab 后续跳转均由 navigation view 实现。

在个人信息界面 10，可以进行头像的修改和姓名、地址的设置；在菜单界面 5，显示餐厅信息，在界面 6 显示各个餐厅的餐品列表，在界面 7 显示每个餐品的详细信息，在该界面可以添加菜品至购物车（订单），并支持修改数量，点击购物车图标进入购物车界面 8，在该界面可以删除订单或修改数量，选择完毕后点击提交进入提交界面，点击返回，相当于取消回到订单界面，点击确认，可以返回至主界面。

2. 定义的类

根据界面布局设计，为每一个 view 创建 controller 文件对其进行控制，具体类如图 3 所示。

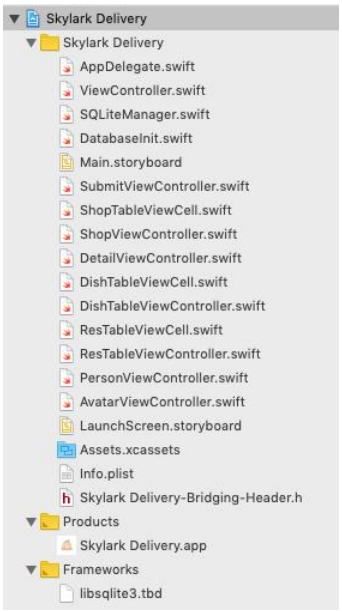

图 3

如图 3 所示，工程文件夹中包含初始文件：AppDelegate.swift、ViewController.swift、Main.storyboard、LaunchScreen.storyboard、Assets.xcassets 和 Info.plist。其中在 Main.storyboard 中进行主要页面的布局设置，在 LaunchScreen.storyboard 文件中进行初始化加载动画的设计，在 Assets.xcassets 中保存使用到的图片资源，在 ViewController.swift 中编写主界面（登录界面）的控制代码。

在数据库方面，创建 SQLiteManager.swift 文件封装针对 sqlite3 数据库的操作，包括打开、关闭数据库，对数据库进行创建、插入、更新、删除和查询操作，以便后续使用 SQL 语句对数据库进行操作。创建 DatabaseInit.swift 文件，创建餐厅表、菜品表、订单表，并定义后续会用到的增加、减少订单功能。

在个人信息 tab 方面，创建 PersonViewController.swift 文件用来显示用户头像和修改保存用户姓名和地址，创建 AvatarViewController.swift 文件供用户修改头像。

在餐品选择 tab 方面，创建 ResTableViewController.swift 文件来控制餐厅列表界面，创建 ResTableViewCell.swift 文件对餐厅的自定义 cell 进行设置。创建 DishTableViewController.swift 文件来控制各个餐厅餐品的列表界面，创建 DishTableViewCell.swift 文件对餐品的自定义 cell 进行设置。创建

DetailViewController.swift 文件来控制餐品详情页的界面。创建 ShopTableViewController.swift 文件来控制订单列表界面，创建 ShopTableViewCell.swift 文件对订单的自定义 cell 进行设置。创建 SubmitViewController.swift 文件来控制订单的提交界面。

3. 定义的功能函数及关键代码

（1）ViewController.swift

```swift
//
//  ViewController.swift
//  Skylark Delivery
//
//  Created by 雀栎 on 2021/5/28.
//  Copyright  2021 雀栎. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        DatabaseInit.initDB()
        // Do any additional setup after loading the view.

    }

    @IBOutlet weak var loginBtn: UIButton!
    @IBOutlet weak var txtUser: UITextField!
    @IBOutlet weak var txtPass: UITextField!

    @IBAction func loginPress(_ sender: Any) {
        let userword = txtUser.text!
        let password = txtPass.text!

        if userword == "skylark" && password == "20001003"{
            //账号密码正确时跳转
            self.performSegue(withIdentifier: "loginToMain", sender: self)
        }else{   //否则提示密码错误，清空密码
            let p = UIAlertController(title: "Login Failed",
            message: "Wrong username or password.", preferredStyle: .alert)
            p.addAction(UIAlertAction(title: "Yes", style: .default,
            handler: {(act:UIAlertAction)in self.txtPass.text=""}))
            present(p, animated: false,completion: nil)
        }
    }
    @IBAction func endUser(_ sender: UITextField) {
        txtUser.resignFirstResponder()
    }
    @IBAction func endPass(_ sender: UITextField) {
        txtPass.resignFirstResponder()
    }
}
```

在 ViewController.swift 中编写主界面（登录界面）的控制代码。在 viewDidLoad 函数中显示界面，并调用类 DatabaseInit 中的 initDB 函数初始化数据库中的表。在 loginPress 函数中对姓名和密码正误进行判断，当密码正确

时，跳转至 tableView，当用户名或密码错误时弹出提示框，并清空密码框内容。在 endUser 和 endPass 函数中设置键盘，使其点击 return 收起键盘。

（2）SQLiteManager.swift

```swift
//  SQLiteManager.swift
//  Skylark Delivery
//
//  Created by 雀栎 on 2021/6/9.
//  Copyright  2021 雀栎. All rights reserved.
//

import Foundation

class SQLiteManager:NSObject //封装数据库操作
{
    private var dbPath:String!
    private var database:OpaquePointer? = nil

    static var sharedInstance:SQLiteManager
    {
        return SQLiteManager()
    }

    override init()
    {
        super.init()

        let dirpath = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask).first!
        dbPath = dirpath.appendingPathComponent("app.sqlite").path
    }

    //open database 打开数据库
    func openDB() -> Bool{
        //let cPath = dbPath.cString(using:String.Encoding.utf8)
        let result = sqlite3_open(dbPath, &database) //文件名不存在，自动创建
        if result != SQLITE_OK{
            print("fail to open database")
            return false
        }
        return true
    }

    //close database 关闭数据库
    func closeDB(){
        sqlite3_close(database)
    }

    //execute the statement: creat, insert, update, delete 增删改数据库
    func execNoneQuerySQL(sql : String) -> Bool{
        var errMsg:UnsafeMutablePointer<Int8>? = nil
        let cSql = sql.cString(using: String.Encoding.utf8)!

        /*
        参数
        1. db：OpaquePointer！ 已打开数据库句柄
        2. sql：执行的 sql
        3. callback：回调函数
        4. 自定义指针，会传递到回调函数内
        5. errmsg：错误信息
        */

        if sqlite3_exec(database, cSql, nil, nil, &errMsg) == SQLITE_OK{
            return true
        }
        let msg = String.init(cString: errMsg!)
```

```
        print(msg)
        return false
    }

    //execute the statement: select  查数据库
    func execQuerySQL(sql : String) -> [[String: AnyObject]]? {
        let cSql = sql.cString(using: String.Encoding.utf8)!
        var statement:OpaquePointer? = nil

        /*
        参数
        1. db：已打开数据库句柄
        2. zSql：要执行的 SQL 语句
        3. nByte：以字节为单位的 sql 语句长度，-1 自动计算
        4. ppStmt：语句句柄，据此获取查询结果
           需要调用 sqlite3_finalize 释放
        5. _ pzTail：未使用的指针地址，通常传入 nil
        */
        if sqlite3_prepare_v2(database, cSql, -1, &statement, nil) != SQLITE_OK{
            sqlite3_finalize(statement)
            print("执行\(sql)错误\n")
            let errmsg = sqlite3_errmsg(database)
            if errmsg != nil{
                print(errmsg!)
            }
            return nil
        }

        var rows = [[String: AnyObject]]()

        while sqlite3_step(statement) == SQLITE_ROW{
            rows.append(record(stmt: statement!))
        }
        sqlite3_finalize(statement)
        return rows
    }

    private func record(stmt: OpaquePointer) -> [String: AnyObject]{
        var row = [String: AnyObject]()
        //遍历所有列，获取每一列的信息
        for col in 0 ..< sqlite3_column_count(stmt){
            let cName = sqlite3_column_name(stmt, col) //获取列名
            let name = String(cString: cName!, encoding: String.Encoding.utf8)

            var value: AnyObject?
            switch(sqlite3_column_type(stmt, col)){
            case SQLITE_FLOAT:
                value = sqlite3_column_double(stmt, col) as AnyObject
            case SQLITE_INTEGER:
                value = Int(sqlite3_column_int(stmt, col)) as AnyObject
            case SQLITE_TEXT:
                let cText = sqlite3_column_text(stmt, col)
                value = String.init(cString: cText!) as AnyObject
            case SQLITE_NULL:
                value = NSNull()
            default:
                print("The data type is not supported!")
            }
            row[name!] = value ?? NSNull()
        }
        return row
    }
}
```

在 SQLiteManager.swift 文件封装针对 sqlite3 数据库的操作。在 openDB 函数中定义打开数据库操作，在 closeDB 函数中定义关闭数据库操作，在

execNoneQuerySQL 函数中定义对数据库的创建、插入、修改、删除操作，在 execQuerySQL 函数中定义对数据库的查询操作。

（3）DatabaseInit.swift

```swift
//
// DatabaseInit.swift
// Skylark Delivery
//
// Created by 雀栎 on 2021/6/9.
// Copyright © 2021 雀栎. All rights reserved.
//
import Foundation
class DatabaseInit
{
 static func initDB()
 {
 let sqlite = SQLiteManager.sharedInstance

 if !sqlite.openDB(){return}

 let createRestaurant = "CREATE TABLE IF NOT EXISTS restaurant('img' TEXT, 'name' TEXT);" //创建餐厅表
 if !sqlite.execNoneQuerySQL(sql: createRestaurant) { sqlite.closeDB() ; return}

 let deleteRestaurant = "DELETE FROM restaurant"
 if !sqlite.execNoneQuerySQL(sql: deleteRestaurant) { sqlite.closeDB() ; return }

 //let resetRestaurant = "DELETE FROM sqlite_sequence WHERE name = 'restaurant';"
 //if !sqlite.execNoneQuerySQL(sql: resetRestaurant) { sqlite.closeDB() ; return }

 let restaurant01 = "INSERT INTO restaurant(img,name) VALUES('McDonalds','McDonald''s');"
 let restaurant02 = "INSERT INTO restaurant(img,name) VALUES('Dunkin','Dunkin'' Donuts');"
 let restaurant03 = "INSERT INTO restaurant(img,name) VALUES('kfc','KFC');"
 let restaurant04 = "INSERT INTO restaurant(img,name) VALUES('Starbucks','Starbucks');"

 if !sqlite.execNoneQuerySQL(sql: restaurant01) { sqlite.closeDB() ; return }
 if !sqlite.execNoneQuerySQL(sql: restaurant02) { sqlite.closeDB() ; return }
 if !sqlite.execNoneQuerySQL(sql: restaurant03) { sqlite.closeDB() ; return }
 if !sqlite.execNoneQuerySQL(sql: restaurant04) { sqlite.closeDB() ; return }

 let dropDish = "DROP TABLE IF EXISTS dish"
 if !sqlite.execNoneQuerySQL(sql: dropDish) { sqlite.closeDB() ; return}

 let createDish = "CREATE TABLE IF NOT EXISTS dish('img' TEXT, 'name' TEXT, 'rest' TEXT, 'price' TEXT)" //创建各个餐厅的菜品
 if !sqlite.execNoneQuerySQL(sql: createDish) { sqlite.closeDB() ; return}

 let deleteDish = "DELETE FROM dish"
 if !sqlite.execNoneQuerySQL(sql: deleteDish) { sqlite.closeDB() ; return }

 //let resetDish = "DELETE FROM sqlite_sequence WHERE name = 'dish';"
 //if !sqlite.execNoneQuerySQL(sql: resetDish) { sqlite.closeDB() ; return }

 let dish01 = "INSERT INTO dish(img, name, rest, price) VALUES('kfc01', 'Original Recipe', 'kfc', '12');"
 let dish02 = "INSERT INTO dish(img, name, rest, price) VALUES('kfc02', 'Zinger Burger', 'kfc', '17');"
 let dish03 = "INSERT INTO dish(img, name, rest, price) VALUES('kfc03', 'Popcorn Chicken', 'kfc', '10');"
 let dish04 = "INSERT INTO dish(img, name, rest, price) VALUES('kfc04', 'Mexican Twister', 'kfc', '16');"
 let dish05 = "INSERT INTO dish(img, name, rest, price) VALUES('kfc05', 'Hot Wings', 'kfc', '13');"

 let dish06 = "INSERT INTO dish(img, name, rest, price) VALUES('mc01', 'Cheese Burger', 'mc', '8');"
 let dish07 = "INSERT INTO dish(img, name, rest, price) VALUES('mc02', 'French Fries', 'mc', '12');"
 let dish08 = "INSERT INTO dish(img, name, rest, price) VALUES('mc03', 'Hot Chocolate', 'mc', '14');"
 let dish09 = "INSERT INTO dish(img, name, rest, price) VALUES('mc04', 'Dip Cone', 'mc', '6');"
 let dish10 = "INSERT INTO dish(img, name, rest, price) VALUES('mc05', 'Shakes', 'mc', '11');"

 let dish11 = "INSERT INTO dish(img, name, rest, price) VALUES('star01', 'Caramel Macchiato', 'star', '34');"
```

```
61.   let dish12 = "INSERT INTO dish(img, name, rest, price) VALUES('star02', 'Coffee Frappuccino', 'star', '25');"
62.   let dish13 = "INSERT INTO dish(img, name, rest, price) VALUES('star03', 'Cheese Cake', 'star', '29');"
63.   let dish14 = "INSERT INTO dish(img, name, rest, price) VALUES('star04', 'Tiramisu', 'star', '16');"
64.   let dish15 = "INSERT INTO dish(img, name, rest, price) VALUES('star05', 'Vegetable Salad', 'star', '23');"
65.
66.   let dish16 = "INSERT INTO dish(img, name, rest, price) VALUES('duk01', 'Icing Doughnuts', 'duk', '48');"
67.   let dish17 = "INSERT INTO dish(img, name, rest, price) VALUES('duk02', 'Ice Caramel Latte', 'duk', '12');"
68.   let dish18 = "INSERT INTO dish(img, name, rest, price) VALUES('duk03', 'Coffee', 'duk', '20');"
69.   let dish19 = "INSERT INTO dish(img, name, rest, price) VALUES('duk04', 'Croissants', 'duk', '21');"
70.   let dish20 = "INSERT INTO dish(img, name, rest, price) VALUES('duk05', 'Cinnamon Roll', 'duk', '15');"
71.
72.   if !sqlite.execNoneQuerySQL(sql: dish01) { sqlite.closeDB() ; return }
73.   if !sqlite.execNoneQuerySQL(sql: dish02) { sqlite.closeDB() ; return }
74.   if !sqlite.execNoneQuerySQL(sql: dish03) { sqlite.closeDB() ; return }
75.   if !sqlite.execNoneQuerySQL(sql: dish04) { sqlite.closeDB() ; return }
76.   if !sqlite.execNoneQuerySQL(sql: dish05) { sqlite.closeDB() ; return }
77.
78.   if !sqlite.execNoneQuerySQL(sql: dish06) { sqlite.closeDB() ; return }
79.   if !sqlite.execNoneQuerySQL(sql: dish07) { sqlite.closeDB() ; return }
80.   if !sqlite.execNoneQuerySQL(sql: dish08) { sqlite.closeDB() ; return }
81.   if !sqlite.execNoneQuerySQL(sql: dish09) { sqlite.closeDB() ; return }
82.   if !sqlite.execNoneQuerySQL(sql: dish10) { sqlite.closeDB() ; return }
83.
84.   if !sqlite.execNoneQuerySQL(sql: dish11) { sqlite.closeDB() ; return }
85.   if !sqlite.execNoneQuerySQL(sql: dish12) { sqlite.closeDB() ; return }
86.   if !sqlite.execNoneQuerySQL(sql: dish13) { sqlite.closeDB() ; return }
87.   if !sqlite.execNoneQuerySQL(sql: dish14) { sqlite.closeDB() ; return }
88.   if !sqlite.execNoneQuerySQL(sql: dish15) { sqlite.closeDB() ; return }
89.
90.   if !sqlite.execNoneQuerySQL(sql: dish16) { sqlite.closeDB() ; return }
91.   if !sqlite.execNoneQuerySQL(sql: dish17) { sqlite.closeDB() ; return }
92.   if !sqlite.execNoneQuerySQL(sql: dish18) { sqlite.closeDB() ; return }
93.   if !sqlite.execNoneQuerySQL(sql: dish19) { sqlite.closeDB() ; return }
94.   if !sqlite.execNoneQuerySQL(sql: dish20) { sqlite.closeDB() ; return }
95.
96.   let dropOrders = "DROP TABLE IF EXISTS orders"
97.   if !sqlite.execNoneQuerySQL(sql: dropOrders) { sqlite.closeDB() ; return}
98.
99.   let createOrders = "CREATE TABLE IF NOT EXISTS orders('img' TEXT, 'name' TEXT, 'num'
100. INTEGER, 'price' INTEGER)" //创建各个餐厅的菜品
101.  if !sqlite.execNoneQuerySQL(sql: createOrders) { sqlite.closeDB() ; return}
102.
103.  let deleteOrders = "DELETE FROM orders"
104.  if !sqlite.execNoneQuerySQL(sql: deleteOrders) { sqlite.closeDB() ; return }
105.
106.  let order01 = "INSERT INTO orders(img, name, num, price) VALUES('kfc01', 'Original Recipe', 0, 12);"
107.  let order02 = "INSERT INTO orders(img, name, num, price) VALUES('kfc02', 'Zinger Burger', 0, 17);"
108.  let order03 = "INSERT INTO orders(img, name, num, price) VALUES('kfc03', 'Popcorn Chicken', 0, 10);"
109.  let order04 = "INSERT INTO orders(img, name, num, price) VALUES('kfc04', 'Mexican Twister', 0, 16);"
110.  let order05 = "INSERT INTO orders(img, name, num, price) VALUES('kfc05', 'Hot Wings', 0, 13);"
111.
112.  let order06 = "INSERT INTO orders(img, name, num, price) VALUES('mc01', 'Cheese Burger', 0, 8);"
113.  let order07 = "INSERT INTO orders(img, name, num, price) VALUES('mc02', 'French Fries', 0, 12);"
114.  let order08 = "INSERT INTO orders(img, name, num, price) VALUES('mc03', 'Hot Chocolate', 0, 14);"
115.  let order09 = "INSERT INTO orders(img, name, num, price) VALUES('mc04', 'Dip Cone', 0, 6);"
116.  let order10 = "INSERT INTO orders(img, name, num, price) VALUES('mc05', 'Shakes', 0, 11);"
117.
118.  let order11 = "INSERT INTO orders(img, name, num, price) VALUES('star01', 'Caramel Macchiato', 0, 34);"
119.  let order12 = "INSERT INTO orders(img, name, num, price) VALUES('star02', 'Coffee Frappuccino', 0, 25);"
120.  let order13 = "INSERT INTO orders(img, name, num, price) VALUES('star03', 'Cheese Cake', 0, 29);"
121.  let order14 = "INSERT INTO orders(img, name, num, price) VALUES('star04', 'Tiramisu', 0, 16);"
122.  let order15 = "INSERT INTO orders(img, name, num, price) VALUES('star05', 'Vegetable Salad', 0, 23);"
123.
124.  let order16 = "INSERT INTO orders(img, name, num, price) VALUES('duk01', 'Icing Doughnuts', 0, 48);"
125.  let order17 = "INSERT INTO orders(img, name, num, price) VALUES('duk02', 'Ice Caramel Latte', 0, 12);"
126.  let order18 = "INSERT INTO orders(img, name, num, price) VALUES('duk03', 'Coffee', 0, 20);"
127.  let order19 = "INSERT INTO orders(img, name, num, price) VALUES('duk04', 'Croissants', 0, 21);"
128.  let order20 = "INSERT INTO orders(img, name, num, price) VALUES('duk05', 'Cinnamon Roll', 0, 15);"
129.
```

```
130.  if !sqlite.execNoneQuerySQL(sql: order01) { sqlite.closeDB() ; return }
131.  if !sqlite.execNoneQuerySQL(sql: order02) { sqlite.closeDB() ; return }
132.  if !sqlite.execNoneQuerySQL(sql: order03) { sqlite.closeDB() ; return }
133.  if !sqlite.execNoneQuerySQL(sql: order04) { sqlite.closeDB() ; return }
134.  if !sqlite.execNoneQuerySQL(sql: order05) { sqlite.closeDB() ; return }
135.
136.  if !sqlite.execNoneQuerySQL(sql: order06) { sqlite.closeDB() ; return }
137.  if !sqlite.execNoneQuerySQL(sql: order07) { sqlite.closeDB() ; return }
138.  if !sqlite.execNoneQuerySQL(sql: order08) { sqlite.closeDB() ; return }
139.  if !sqlite.execNoneQuerySQL(sql: order09) { sqlite.closeDB() ; return }
140.  if !sqlite.execNoneQuerySQL(sql: order10) { sqlite.closeDB() ; return }
141.
142.  if !sqlite.execNoneQuerySQL(sql: order11) { sqlite.closeDB() ; return }
143.  if !sqlite.execNoneQuerySQL(sql: order12) { sqlite.closeDB() ; return }
144.  if !sqlite.execNoneQuerySQL(sql: order13) { sqlite.closeDB() ; return }
145.  if !sqlite.execNoneQuerySQL(sql: order14) { sqlite.closeDB() ; return }
146.  if !sqlite.execNoneQuerySQL(sql: order15) { sqlite.closeDB() ; return }
147.
148.  if !sqlite.execNoneQuerySQL(sql: order16) { sqlite.closeDB() ; return }
149.  if !sqlite.execNoneQuerySQL(sql: order17) { sqlite.closeDB() ; return }
150.  if !sqlite.execNoneQuerySQL(sql: order18) { sqlite.closeDB() ; return }
151.  if !sqlite.execNoneQuerySQL(sql: order19) { sqlite.closeDB() ; return }
152.  if !sqlite.execNoneQuerySQL(sql: order20) { sqlite.closeDB() ; return }
153.
154.  sqlite.closeDB();
155.
156.  }
157.
158.  static func addNum(str:String){ //增加份数
159.
160.  let sqlite = SQLiteManager.sharedInstance
161.  if !sqlite.openDB(){return}
162.
163.  let addItem = "update orders set num = num + 1 where img = '" + str + "'"
164.  if !sqlite.execNoneQuerySQL(sql: addItem) { sqlite.closeDB() ; return }
165.
166.  sqlite.closeDB()
167.  }
168.
169.  static func subNum(str:String){ //减少份数
170.  let sqlite = SQLiteManager.sharedInstance
171.  if !sqlite.openDB(){return}
172.
173.  let subItem = "update orders set num = num - 1 where img = '" + str + "'"
174.  if !sqlite.execNoneQuerySQL(sql: subItem) { sqlite.closeDB() ; return }
175.
176.  sqlite.closeDB()
177.  }
178.
179.  static func GetRes() //测试用
180.  {
181.  let sqlite = SQLiteManager.sharedInstance
182.  if !sqlite.openDB(){return}
183.  let queryResult = sqlite.execQuerySQL(sql: "SELECT * FROM orders")
184.
185.  print(queryResult!)
186.  for row in queryResult!{
187.  print(row["name"])
188.  }
189.  sqlite.closeDB();
190.  }
191.
192.
193. }
```

DatabaseInit.swift 文件用于定义一些与数据库有关的操作。initDB 函数用来初始化定义数据库中的餐厅表、餐品表和订单表，addNum 函数用来添加订

单列表中某种餐品的数量，subNum 函数用来减少订单列表中某种餐品的数量，GetRes 函数用于编写代码过程中的测试，查看 SQL 语句是否有误，并排除问题。

（4）PersonViewController.swift

```
//
//  PersonViewController.swift
//  Skylark Delivery
//
//  Created by 雀栎 on 2021/6/9.
//  Copyright  2021 雀栎. All rights reserved.
//

import UIKit

class PersonViewController: UIViewController {

    @IBOutlet weak var avater: UIImageView!
    @IBOutlet weak var nameTxt: UITextField!
    @IBOutlet weak var addrTxt: UITextField!
    @IBOutlet weak var okBtn: UIButton!

    @IBAction func namein(_ sender: UITextField) {
        nameTxt.resignFirstResponder()
    }
    @IBAction func addrin(_ sender: UITextField) {
        addrTxt.resignFirstResponder()
    }

    static var i = 0

    override func viewDidLoad() {
        super.viewDidLoad()

        switch PersonViewController.i{
        case 0:avater.image = UIImage(named: "people01_30")
        case 1:avater.image = UIImage(named: "people02_30")
        case 2:avater.image = UIImage(named: "people03_30")
        case 3:avater.image = UIImage(named: "people04_30")
        case 4:avater.image = UIImage(named: "people05_30")
        case 5:avater.image = UIImage(named: "people06_30")
        default:avater.image = UIImage(named: "people01_30")
        }

        nameTxt.text = PersonViewController.name
        addrTxt.text = PersonViewController.addr
        // Do any additional setup after loading the view.
    }

    static var name = "Skylark"
    static var addr = "Wuhan University"
    @IBAction func okClick(_ sender: Any) {
        PersonViewController.name = nameTxt.text ?? "Skylark"
        PersonViewController.addr = addrTxt.text ?? "Wuhan University"
    }
    override func viewWillAppear(_ animated: Bool) {
        super.viewDidLoad()

        switch PersonViewController.i{
        case 0:avater.image = UIImage(named: "people01_30")
        case 1:avater.image = UIImage(named: "people02_30")
        case 2:avater.image = UIImage(named: "people03_30")
        case 3:avater.image = UIImage(named: "people04_30")
        case 4:avater.image = UIImage(named: "people05_30")
        case 5:avater.image = UIImage(named: "people06_30")
        default:avater.image = UIImage(named: "people01_30")
```

```
        }

        nameTxt.text = PersonViewController.name
        addrTxt.text = PersonViewController.addr
    }

}
```

创建 PersonViewController.swift 文件用来显示用户头像和修改保存用户姓名和地址。ViewDidLoad 函数用来初始化界面，变量 i 用于对头像进行标号，使界面显示对应图片，okClick 函数用于保存姓名和地址文本框中的内容至静态变量 name 和 addr，viewWillAppear 函数用于在修改完头像之后的页面刷新，会根据修改的头像实时刷新页面显示内容。

（5）AvatarViewController.swift

```
//
//  AvatarViewController.swift
//  Skylark Delivery
//
//  Created by 雀栎 on 2021/6/11.
//  Copyright  2021 雀栎. All rights reserved.
//

import UIKit

class AvatarViewController: UIViewController {

    @IBOutlet weak var icon1: UIButton!
    @IBOutlet weak var icon2: UIButton!
    @IBOutlet weak var icon3: UIButton!
    @IBOutlet weak var icon4: UIButton!
    @IBOutlet weak var icon5: UIButton!
    @IBOutlet weak var icon6: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }
    @IBAction func icon1click(_ sender: Any) {
        PersonViewController.i = 0
    }
    @IBAction func icon2click(_ sender: Any) {
        PersonViewController.i = 1
    }
    @IBAction func icon3click(_ sender: Any) {
        PersonViewController.i = 2
    }
    @IBAction func icon4click(_ sender: Any) {
        PersonViewController.i = 3
    }
    @IBAction func icon5click(_ sender: Any) {
        PersonViewController.i = 4
    }
    @IBAction func icon6click(_ sender: Any) {
        PersonViewController.i = 5
    }
}
```

AvatarViewController.swift 文件用于修改个人信息中的头像，在该界面，

根据用户对不同按钮的点击修改静态变量 i 的数值，从而在 person 界面对界面进行刷新，显示已更换的头像。

（6）ResTableViewController.swift 和 ResTableViewCell.swift

```
1.  //
2.  //  ResTableViewController.swift
3.  //  Skylark Delivery
4.  //
5.  //  Created by 雀栎 on 2021/6/10.
6.  //  Copyright  2021 雀栎. All rights reserved.
7.  //
8.
9.  import UIKit
10.
11. class ResTableViewController: UITableViewController {
12.
13.     override func viewDidLoad() {
14.         super.viewDidLoad()
15.     }
16.
17.     // MARK: - Table view data source
18.
19.     override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
20.
21.         let sqlite = SQLiteManager.sharedInstance
22.         sqlite.openDB()
23.         let queryResult = sqlite.execQuerySQL(sql: "SELECT * FROM restaurant")
24.         sqlite.closeDB()
25.         return queryResult?.count ?? 4
26.     }
27.
28.
29.     override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
30.         let cell = tableView.dequeueReusableCell(withIdentifier: "resCell", for: indexPath) as! ResTableViewC
    ell
31.
32.         let sqlite = SQLiteManager.sharedInstance
33.         sqlite.openDB()
34.         let queryResult = sqlite.execQuerySQL(sql: "SELECT * FROM restaurant")
35.         sqlite.closeDB()
36.
37.         cell.resImg.image = UIImage(named: queryResult?[indexPath.row]["img"] as! String)
38.         cell.nameLabel.text = queryResult?[indexPath.row]["name"] as! String
39.
40.         return cell
41.     }
42.
43.     static var num = 0
44.
45.     override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
46.         if segue.identifier == "ResToDish"{
47.             let selRes = self.tableView.indexPathForSelectedRow
48.             ResTableViewController.num = selRes?.row ?? 0
49.             print(ResTableViewController.num)
50.         }
```

```
    }
}
```

```
//
//  ResTableViewCell.swift
//  Skylark Delivery
//
//  Created by 雀栎 on 2021/6/10.
//  Copyright  2021 雀栎. All rights reserved.
//

import UIKit

class ResTableViewCell: UITableViewCell {

    @IBOutlet weak var resImg: UIImageView!
    @IBOutlet weak var nameLabel: UILabel!
    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
    }

    override func setSelected(_ selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }
}
```

在 ResTableViewCell.swift 文件中进行餐厅列表自定义 cell 的设置，包括餐厅图片和餐厅名称。在 ResTableViewController.swift 文件中设置 tableview 的行数（餐厅表中的行数），和每一行中的内容，通过 SQL 语句的查找实现。

（7）DishTableViewController.swft 和 DishTableViewCell.swift

```
//
//  DishTableViewController.swift
//  Skylark Delivery
//
//  Created by 雀栎 on 2021/6/10.
//  Copyright  2021 雀栎. All rights reserved.
//

import UIKit

class DishTableViewController: UITableViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    let selKfc = "SELECT * FROM dish WHERE rest = 'kfc'"
    let selMc = "SELECT * FROM dish WHERE rest = 'mc'"
    let selStar = "SELECT * FROM dish WHERE rest = 'star'"
    let selDuk = "SELECT * FROM dish WHERE rest = 'duk'"
    var selRes:String = ""

    // MARK: - Table view data source

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        // #warning Incomplete implementation, return the number of rows
        let sqlite = SQLiteManager.sharedInstance
        sqlite.openDB()
```

```
        let queryResult = sqlite.execQuerySQL(sql: selKfc)
        sqlite.closeDB()
        return queryResult?.count ?? 5
    }


    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{

        let cell = tableView.dequeueReusableCell(withIdentifier: "dishCell", for: indexPath) as! DishTableVi
ewCell

        switch ResTableViewController.num{
        case 0:selRes = selMc
        case 1:selRes = selDuk
        case 2:selRes = selKfc
        case 3:selRes = selStar
        default:selRes = selKfc
        }

        let sqlite = SQLiteManager.sharedInstance
        sqlite.openDB()
        let dishResult = sqlite.execQuerySQL(sql: selRes)
        sqlite.closeDB()

        cell.dishImg.image = UIImage(named: dishResult?[indexPath.row]["img"]as! String)
        cell.dishName.text = dishResult?[indexPath.row]["name"]as? String
        cell.dishPrice.text = "Price: $" + (dishResult?[indexPath.row]["price"]as? String ?? "")

        // Configure the cell...

        return cell
    }

    static var num = 0
    static var res = ""
    static var imgname = ""

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        if segue.identifier == "DishToDetail"{
            let selDish = self.tableView.indexPathForSelectedRow
            DishTableViewController.num = selDish?.row ?? 0

            switch ResTableViewController.num{
            case 0:DishTableViewController.res = "mc0"
            case 1:DishTableViewController.res = "duk0"
            case 2:DishTableViewController.res = "kfc0"
            case 3:DishTableViewController.res = "star0"
            default:DishTableViewController.res = "mc0"
            }

            DishTableViewController.imgname = DishTableViewController.res +
            String(DishTableViewController.num + 1)

            print(DishTableViewController.imgname)
        }
    }
}
```

```
//
//  DishTableViewCell.swift
//  Skylark Delivery
//
//  Created by 雀栎 on 2021/6/10.
//  Copyright  2021 雀栎. All rights reserved.
//

import UIKit
```

```
10.
11.  class DishTableViewCell: UITableViewCell {
12.
13.      @IBOutlet weak var dishImg: UIImageView!
14.      @IBOutlet weak var dishName: UILabel!
15.      @IBOutlet weak var dishPrice: UILabel!
16.
17.      override func awakeFromNib() {
18.          super.awakeFromNib()
19.          // Initialization code
20.      }
21.
22.      override func setSelected(_ selected: Bool, animated: Bool) {
23.          super.setSelected(selected, animated: animated)
24.
25.          // Configure the view for the selected state
26.      }
27.  }
```

在 DishTableViewCell.swift 文件中进行餐品列表自定义 cell 的设置，包括餐品图片、餐品名称和餐品价格。在 DishTableViewController.swift 文件中设置 tableview 的行数（餐厅表中的行数），和每一行中的内容，通过 SQL 语句的查找实现。判定上一页餐厅列表点击的是哪一个餐厅进行的跳转，在餐品界面仅显示该餐厅的餐品。

（8）DetailViewController.swift

```
1.  //
2.  //  DetailViewController.swift
3.  //  Skylark Delivery
4.  //
5.  //  Created by 雀栎 on 2021/6/10.
6.  //  Copyright  2021 雀栎. All rights reserved.
7.  //
8.
9.  import UIKit
10.
11. class DetailViewController: UIViewController {
12.
13.     @IBOutlet weak var DetImg: UIImageView!
14.     @IBOutlet weak var DetName: UILabel!
15.     @IBOutlet weak var DetPrice: UILabel!
16.     @IBOutlet weak var DetDe: UILabel!
17.     @IBOutlet weak var addBtn: UIButton!
18.     @IBOutlet weak var subBtn: UIButton!
19.     @IBOutlet weak var orderNum: UILabel!
20.
21.
22.
23.     override func viewDidLoad() {
24.         super.viewDidLoad()
25.
26.         // Do any additional setup after loading the view.
27.
28.         let selDish = "SELECT * FROM orders WHERE img = '" + DishTableViewController.imgname + "'"
29.         let sqlite = SQLiteManager.sharedInstance
30.         if !sqlite.openDB(){return}
31.         let dishResult = sqlite.execQuerySQL(sql: selDish)
32.
33.         print(dishResult)
34.
35.         let detimg = DishTableViewController.imgname
```

```
        let detname = dishResult?[0]["name"]
        let detprice = dishResult?[0]["price"]
        let detnum = dishResult?[0]["num"]

        DetImg.image = UIImage(named: detimg)
        DetName.text = detname as! String
        DetPrice.text = "$ " + (detprice as! NSNumber).stringValue
        DetDe.text = "This is a" + (detname as! String)
        orderNum.text = (detnum as! NSNumber).stringValue

        sqlite.closeDB()

    }

    var num = 0

    @IBAction func addNum(_ sender: UIButton) {

        num = num + 1
        orderNum.text = String(num)

        let sqlite = SQLiteManager.sharedInstance
        if !sqlite.openDB(){return}
        DatabaseInit.addNum(str: DishTableViewController.imgname)
        sqlite.closeDB()

        let selDish = "SELECT * FROM orders WHERE img = '" + DishTableViewController.imgname + "'"
        if !sqlite.openDB(){return}
        let dishResult = sqlite.execQuerySQL(sql: selDish)
        let detnum = dishResult?[0]["num"]
        print(detnum)

    }

    @IBAction func subNum(_ sender: UIButton) {
        if num>0{

            num = num - 1
            orderNum.text = String(num)


            let sqlite = SQLiteManager.sharedInstance
            if !sqlite.openDB(){return}
            DatabaseInit.subNum(str: DishTableViewController.imgname)
            sqlite.closeDB()

            let selDish = "SELECT * FROM orders WHERE img = '" + DishTableViewController.imgname + "'"
            if !sqlite.openDB(){return}
            let dishResult = sqlite.execQuerySQL(sql: selDish)
            let detnum = dishResult?[0]["num"]
            print(detnum)
        }
    }

}
```

DetailViewController.swift 文件来控制餐品详情页的界面，根据餐品界面点击的菜品通过订单表显示其详细信息，初始数量为 0，通过 addNum 和 subNum 函数对订单表中该菜品的数量进行增加或减少，同时改变界面显示内容。

（9）ShopTableViewController.swft 和 ShopTableViewCell.swift

```
//
//  ShopViewController.swift
//  Skylark Delivery
```

```swift
//
//  Created by 雀栎 on 2021/6/11.
//  Copyright  2021 雀栎. All rights reserved.
//

import UIKit

class ShopViewController: UIViewController, UITableViewDataSource {
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        let selShop = "SELECT * FROM orders WHERE num <> 0"
        let sqlite = SQLiteManager.sharedInstance
        sqlite.openDB()
        let queryResult = sqlite.execQuerySQL(sql: selShop)
        sqlite.closeDB()
        return queryResult?.count ?? 5
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "shopCell", for: indexPath) as! ShopTableViewCell

        let selShop = "SELECT * FROM orders WHERE num <> 0"
        let sqlite = SQLiteManager.sharedInstance
        sqlite.openDB()
        let queryResult = sqlite.execQuerySQL(sql: selShop)
        sqlite.closeDB()

        cell.shopImg.image = UIImage(named: queryResult?[indexPath.row]["img"]as! String)
        cell.shopName.text = queryResult?[indexPath.row]["name"]as! String
        cell.shopPrice.text = (queryResult?[indexPath.row]["price"]as! NSNumber).stringValue
        cell.shopNum.text = (queryResult?[indexPath.row]["num"]as! NSNumber).stringValue

        return cell
    }

    func tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath) -> Bool {
        return true
    }

    func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {
        if editingStyle == .delete {
            tableView.deleteRows(at: [indexPath], with: .fade)
        }
    }

    static var tota = 0

    override func viewDidLoad() {
        super.viewDidLoad()
        ShopViewController.tota = ShopViewController.calculate()
        total.text = "Total: $ " + String(ShopViewController.tota)
        // Do any additional setup after loading the view.
    }

    @IBOutlet weak var total: UILabel!
    @IBAction func flushed(_ sender: Any) {
        total.text = "Total: $ " + String(ShopViewController.tota)
    }

    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a little preparation before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Get the new view controller using segue.destination.
        // Pass the selected object to the new view controller.
    }
    */
```

```
    static func calculate ()->Int{
        let selShop = "SELECT * FROM orders WHERE num <> 0"
        let sqlite = SQLiteManager.sharedInstance
        sqlite.openDB()
        let queryResult = sqlite.execQuerySQL(sql: selShop)
        sqlite.closeDB()

        var totalprice = 0
        for row in queryResult!{
            let numstr = (row["num"] as! NSNumber).stringValue
            let pricestr = (row["price"] as! NSNumber).stringValue
            let numint = Int(numstr)
            let priceint = Int(pricestr)
            totalprice = totalprice + numint! * priceint!
        }
        return totalprice
    }
}
```

```
//
//  ShopTableViewCell.swift
//  Skylark Delivery
//
//  Created by 雀栎 on 2021/6/11.
//  Copyright  2021 雀栎. All rights reserved.
//

import UIKit

class ShopTableViewCell: UITableViewCell {

    @IBOutlet weak var shopImg: UIImageView!
    @IBOutlet weak var shopName: UILabel!
    @IBOutlet weak var shopPrice: UILabel!
    @IBOutlet weak var shopSub: UIButton!
    @IBOutlet weak var shopAdd: UIButton!
    @IBOutlet weak var shopNum: UILabel!
    @IBOutlet weak var shopTop: UILabel!

    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
    }

    override func setSelected(_ selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }


    @IBAction func subNum(_ sender: Any) {
        let sqlite = SQLiteManager.sharedInstance
        if !sqlite.openDB(){return}
        sqlite.closeDB()

        let selDish = "SELECT * FROM orders WHERE img = '" + DishTableViewController.imgname + "'"
        if !sqlite.openDB(){return}
        let dishResult = sqlite.execQuerySQL(sql: selDish)
        let detnum = dishResult?[0]["num"]
        print(detnum)

        if ((detnum as! NSNumber) as! Int)>0{
            let sqlite = SQLiteManager.sharedInstance
            if !sqlite.openDB(){return}
            DatabaseInit.subNum(str: DishTableViewController.imgname)
            sqlite.closeDB()

```

```
            let selDish = "SELECT * FROM orders WHERE img = '" + DishTableViewController.imgname + "'"
            if !sqlite.openDB(){return}
            let dishResult = sqlite.execQuerySQL(sql: selDish)
            let detnum = dishResult?[0]["num"]
            print(detnum)

            shopNum.text = (detnum as! NSNumber).stringValue

            ShopViewController.tota = ShopViewController.calculate()
        }
    }

    @IBAction func addNum(_ sender: Any) {
        let sqlite = SQLiteManager.sharedInstance
        if !sqlite.openDB(){return}
        DatabaseInit.addNum(str: DishTableViewController.imgname)
        sqlite.closeDB()

        let selDish = "SELECT * FROM orders WHERE img = '" + DishTableViewController.imgname + "'"
        if !sqlite.openDB(){return}
        let dishResult = sqlite.execQuerySQL(sql: selDish)
        let detnum = dishResult?[0]["num"]
        print(detnum)

        shopNum.text = (detnum as! NSNumber).stringValue

        ShopViewController.tota = ShopViewController.calculate()
    }
}
```

在 ShopTableViewCell.swift 文件中进行订单列表自定义 cell 的设置，包括餐品图片、餐品名称、餐品价格、餐品数量和增加减少按钮，创建 addNum 和 subNum 函数供用户进行订单表中餐品数量修改，并对页面显示的数量进行刷新。在 ShopTableViewController.swift 文件中设置 tableview 的行数（订单表中数量不为 0 的餐品），和每一行中的内容，通过 SQL 语句的查找。在 tableview 下方设置 label 显示总价，通过 calculate 函数实现，计算查询出的表每一行数量与价格乘积的和，点击右侧刷新按钮对 label 显示的总价进行刷新。点击右下角提交按钮可以跳转至提交界面。

（10）SubmitViewController.swift

```
//
//  SubmitViewController.swift
//  SkyLark Delivery
//
//  Created by 雀栎 on 2021/6/11.
//  Copyright  2021 雀栎. All rights reserved.
//

import UIKit

class SubmitViewController: UIViewController {

    @IBOutlet weak var name: UILabel!
    @IBOutlet weak var addr: UILabel!
    override func viewDidLoad() {
        super.viewDidLoad()

```

```
        name.text = PersonViewController.name
        addr.text = PersonViewController.addr
        // Do any additional setup after loading the view.
    }

    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a little preparation before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Get the new view controller using segue.destination.
        // Pass the selected object to the new view controller.
    }
    */
}
```

SubmitViewController.swift 文件来控制订单的提交界面。界面的姓名和地址内容由个人信息界面的 testfield 输入的静态变量控制。由于是 navigation 结构，点击返回可以回到订单界面，点击确认可以跳转至主界面。

（二） 结论

在导入 sqlite3 数据库并添加头文件将其导入之后，程序可以正常运行，运行结果如下。

1.初始化加载动画

该程序的初始化加载动画，包括我的姓名和学号。

图 4

2.主界面

主界面如图 5 所示，中间是欢迎语：欢迎来到雀栎外卖系统！下方是用户命

名和密码的输入框，当用户名或密码输入错误时会弹出提示框显示登录失败，错误的用户名或密码。点击确定，重新输入密码时，密码框中的内容会自动清空。当输入正确的用户名和密码时，点击登录按钮，可以成功跳转至点餐界面。

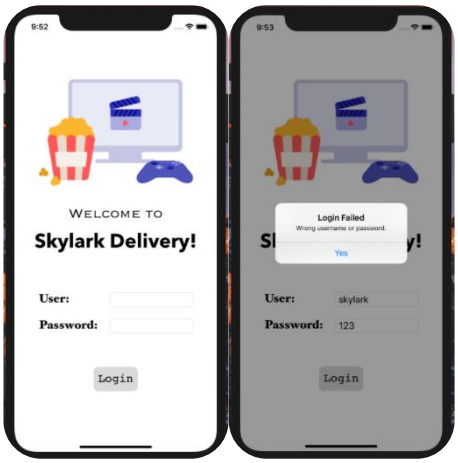

图 5

3. 点餐界面

点餐界面如图 6 所示，餐厅界面总共有四家餐厅，点击进入餐品界面，显示每个餐厅的详细餐品列表，可以显示其图片、名称和价格，点击菜品进入详情页面，包括餐品大图、餐品名、详细介绍和价格，以及添加至购物车功能，点击加或减可以对餐品数量进行调整，右下角为购物车图标，点击可以进入订单界面。

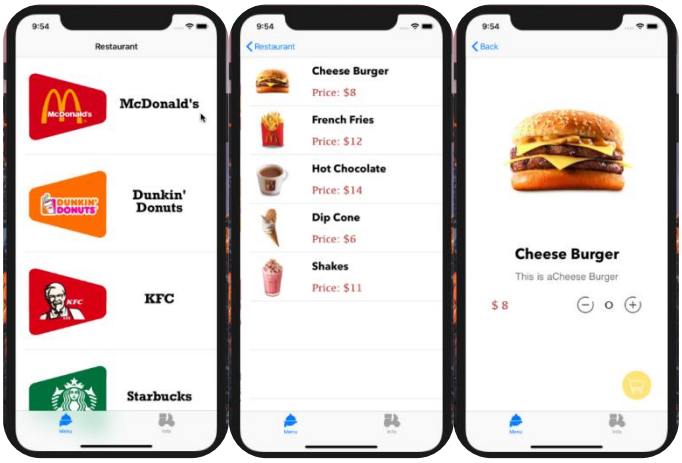

图 6

4. 订单界面

订单界面如图 7 所示，显示已加入购物车的餐品，可以显示其图片、名称、价格和数量，

左滑可以进行删除，点击增加、减少按钮可以对数量进行修改，修改后点击下方刷新按钮可以更新总价，右下角提交按钮可以提交订单。在提交订单界面可以显示用户设置的姓名和地址，点击返回回到订单界面，点击确认回到主界面。

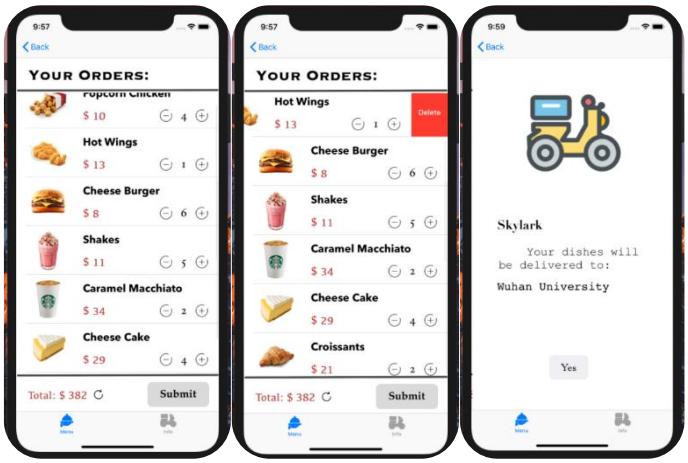

图 7

5. 个人信息界面

个人信息界面如图 8 所示，显示个人信息，包括头像、姓名和地址，头像可以进入修改头像界面进行选择，姓名和地址可以输入文字进行修改。

图 8

三、 小结

本次实验基本完成了主题要求的任务内容，解决了大部分的问题，但仍有一些不足之处。

在写代码的过程中，我遇到了很多问题，也有了很多收获。

首先，在页面跳转方面，点击按钮跳转界面可以直接在 storyboard 中通过拖拽进行实现，所以我对编写代码进行跳转的方式有些陌生，在进行登录时需要对账号密码进行判断，正确则跳转，否则提示密码错误，最后通过上网搜索资料与示例，学习了页面跳转的方法，成功解决的问题。

其次在更换头像方面，由于我使用的是 navigation view，所以在选择好头像之后进行返回时，页面并不会自动刷新，最后通过学习，使用 viewWillAppear 函数对页面进行了刷新，成功实现了更新头像的功能。

然后，在虚拟机键盘的使用方面，起初无法通过点击 textfield 弹出键盘，在修改了系统设置之后解决了该问题，但弹出的键盘无法通过点击 return 或空白区域收起键盘，导致无法点击被键盘遮盖住的按钮，最后通过查阅资料，通过在与 textfield 绑定的函数中调用 resignFirstResponder()方法，实现了点击 return 收起键盘。

之后，在点餐页面进行跳转时，没有找到合适的方法，使得餐品界面显示所点击的指定餐厅的餐品，最后通过 prepare 方法，传给餐品界面所选择的餐厅行号，在餐品界面对该静态变量进行 switch 选择，从而显示餐厅名属性为该餐厅的列表。在跳转细节界面的时候，由于该界面的餐品属于不同餐厅且数量较多，通过行号进行选择是不合适的，最后通过将所点击的 cell 的图片名传至细节界面，通过查询图片名等于该静态变量的字段进行显示。

最后，在连接号数据库，并将数据库内容成功显示在界面上之后，在一次的调试过程中，突然发生了无法显示的情况，通过排除问题，最后发现不是 SQL 语句或其他逻辑代码的问题，而是数据库的连接出现了问题，在重新连接时，创建 Objiect-C 文件时，未出现自动创建头文件的提示，最后通过在文件设置中头文件一栏中进行了删除，并连接新的头文件解决了问题。

在解决了这些问题之后，仍有一些遗留问题无法解决。

首先，在增加减少餐品的数量时，按钮的点击效果有一些滞后，首次点击没有反应，后面的点击均会滞后一次点击，该问题我在网络上未找到相关解决办法，

也在多次修改之后没有效果，这是第一个遗留问题。其次，在订单页面，在删除订单时，会中断报错，无法进行删除，我猜测原因可能是该界面不是一个UItableview而是在view中新增了一个tableview，所以可能代码会有区别，但我未找到正确的解决方法，虽然无法进行删除，但可以使用减少按钮将数量减为零。最后，在个人信息界面，虽然和主页面的代码相同都调用了resignFirstResponder()方法，烦键盘弹出后无法通过点击return收起，至今仍未找到问题所在。

通过本次大作业，我在学习了很多知识的同时，也收获了很多感悟。

一方面，要时刻紧跟课程步伐，夯实基础。由于该课程需要使用虚拟机，所以电脑经常会出现卡顿，反应速度也比较缓慢，所以无法在课堂上跟着老师的代码进行操作，课后也因为种种原因除了第一个“武汉加油”的例子，没有及时把之后每节课的例子重写运行，这导致在最后完成大作业时，很多知识都要重新学起，包括很多非常基础的内容都要查询很长时间，这对编程进度产生了很大影响。

另一方面，这次大作业锻炼了我查阅资料获取有效信息的能力。由于该课程是ios编程，在网络上的信息较少，而且很多都是国外的博客，均为英文网页，这需要我仔细阅读并从中获取有用的信息。而且在图书馆找到的一些工具书，很多都是通过具体的例子来进行讲解，跟具有个性化的程序内容适配性并不是很强，无法进行直接修改，所以还需要进行很多调整，才能正确运行，这需要将知识融会贯通，真正理解其中内涵。

总而言之，本次实验中我体会到了一个程序从无到有的历程，最终的成品虽然有一些瑕疵，但整体来看整个程序模块清晰，流程明了，界面也比较美观，基本实现了要求的任务内容，我个人而言还是非常满意的。与此同时，本次编程中带给我的一些宝贵经验也可以运用在之后的学习与实践过程中，对我拥有长远的有利影响。

四、 指导教师评语及成绩

附件：

实验报告说明

1．实验项目名称：要用最简练的语言反映实验的内容。要求与实验指导书中相一致。

2．实验目的：目的要明确，要抓住重点，符合实验任务书中的要求。

3．实验环境：实验用的软硬件环境（配置）。

4．实验方案设计（思路、步骤和方法等）：这是实验报告极其重要的内容。包括概要设计、详细设计和核心算法说明及分析，系统开发工具等。应同时提交程序或设计电子版。

对于**设计型和综合型实验**，在上述内容基础上还应该画出流程图、设计思路和设计方法，再配以相应的文字说明。

对于**创新型实验**，还应注明其创新点、特色。

5．结论（结果）：即根据实验过程中所见到的现象和测得的数据，做出结论（可以将部分测试结果进行截屏）。

6．小结：对本次实验的心得体会，所遇到的问题及解决方法，其他思考和建议。

7．指导教师评语及成绩：指导教师依据学生的实际报告内容，用简练语言给出本次实验报告的评价和价值。