2018 年中国高校计算机大赛微信小程序应用开发赛作品报告

作品名称： RunningLeg

电子邮箱： 18791767622@163.com

提交日期： 2018/6/12

目 录

第一章 作品概述

1.1 背景及研究意义

随着现在生活消费观念的改变，人们越来越多实现了足不出户的购买物品，因此也出现了外卖跑腿等业务，我们开发的微信小程序——RunningLeg 也正是在人们便捷购物上做出了一部分贡献。其次，RunningLeg 更是一款基于高校环境的跑腿服务平台，不但利于师生的购物便捷，也可以代收快递，捎带东西，服务方面多样且人性化。服务于广大师生，并且营造校园互帮互助氛围。

1.2 相关工作

据调查，从 2017 年 8 月入以来，在校大学生有超过 3654 万人，再加上高校教师，用户群体相当多，也意味着需求数量和种类非常多，因此 RunningLeg 也正是填补这一空缺，为众多学生教师群体提供更加多样的跑腿服务。

1.3 作品简介

我们的产品针对高校师生设计，在高校环境中，提供所有可能需要的帮助。只要是在同一所学校中的跑腿需求，RunningLeg 都可以提供中介服务，有人在平台上发单求助，并根据自己的需求指定金额，有人在平台上接单给予帮助，凡是领取外卖，帮买东西，代收快递，传送资料等等都可以在平台上发布。

1.4 创新性描述

据我们对城市跑腿于外卖服务的调查，在多数高校考虑到校园安全问题，禁止外卖人员进入校园，这就为用户造成了一定的障碍，而 RunningLeg 就提供了可以连接校园内外的跑腿服务。而城市跑腿的功能与外卖服务大同小异，同样不能保证服务的最后一公里，且服务仅局限于生活消费品与食物等，对于需要在校园中流转的物品不能提供服务。基于以上需求，促使我们团队开发 RunningLeg，旨在为全国高校提供

更加人性完善的服务。

1.5 小结

在本章，对我们的作品做了一个整体的介绍，包括作品的背景及研究意义、相关工作以及特色描述。从跑腿需求入手，进行了市场调研，分析了本作品的主要研究内容和优势，为后期作品的设计与实现奠定基础。

第二章 作品设计

本章主要对本系统的结构、算法和功能，设计进行设计和阐述，通过规范的设计方法，明确系统的框架和流程，为后一章节的实现奠定基础。

2.1 系统总体设计

本系统采用微信小程序独有的架构方式,如图 2-1。

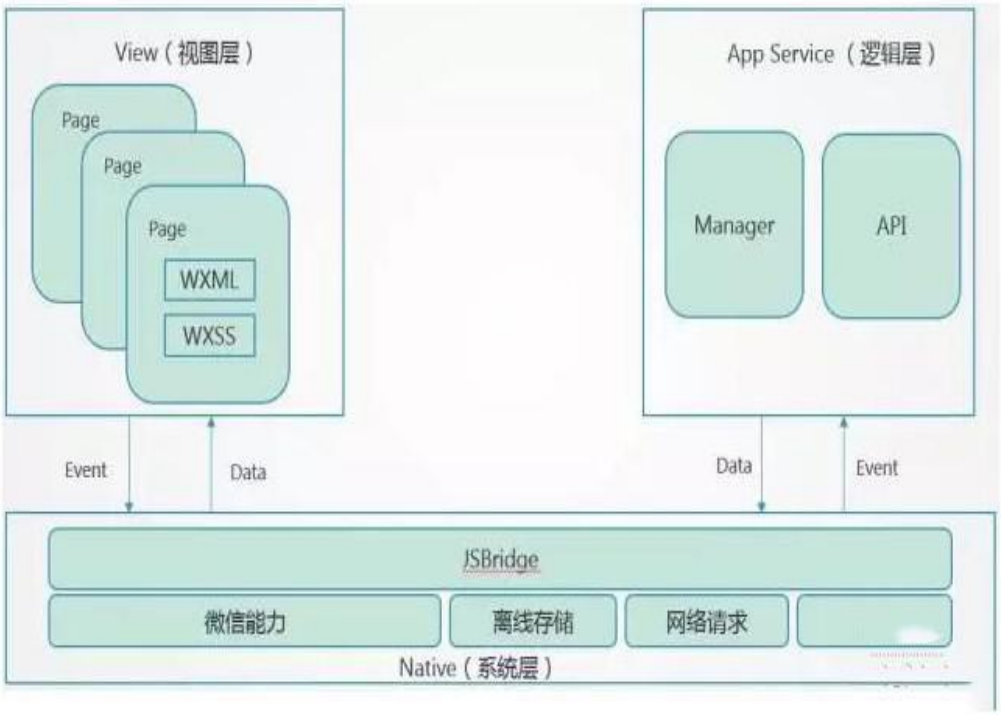


图 2-1

在微信小程序端，主要分为视图层（View），逻辑层（Logic），系统层（System）。视图层（View）由 WXML 和 WXSS 来共同实现，前者其实就是一种微信定义的模板语言，而后者类似 CSS。在逻辑层（Logic）中，其中利用 JS 语言负责业务逻辑的实现。微信小程序借助的是系统层（System）的 JSBridge 实现了对底层 API 接口的调用，脱离 IOS，Android 的限制。

在真正的服务器后端，建立管理 Mysql 数据库，处理大部分和微信 api 接口无关的逻辑，通过 json 格式数据包传送数据，并提供 API 接口，如图 2-2 所示。

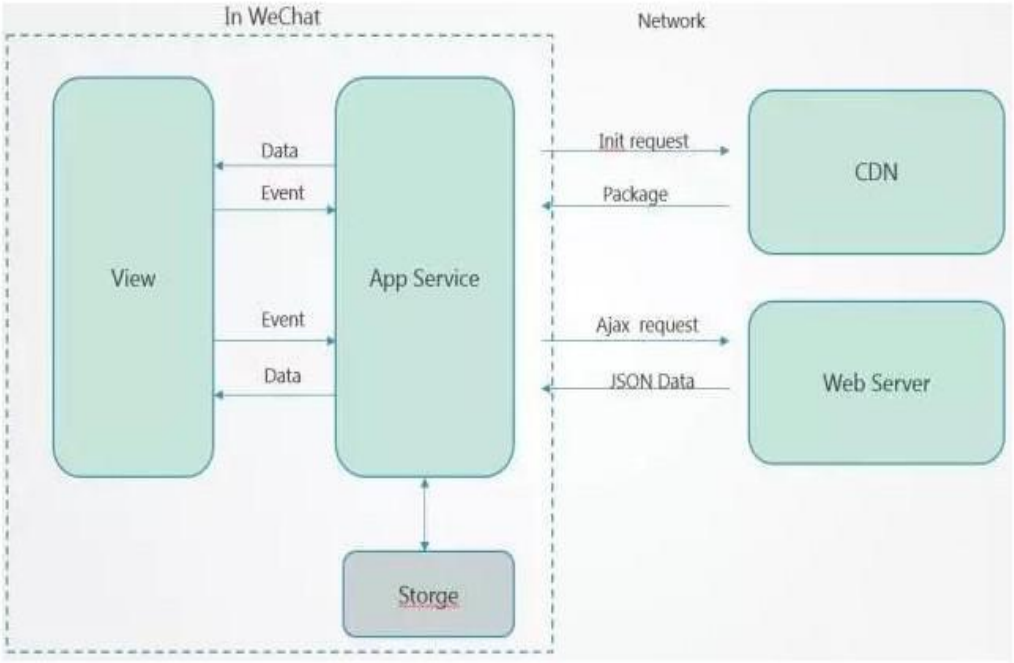


图 2-2

2.2 核心算法设计

核心的算法是为用户推荐合适的订单，为了提高用户体验感，我们选择最新的取货距离最近的订单提供给用户。其二的是在订单数量超过一定限制时，我们根据用户发布订单的时间优先顺序展示订单，用户在发布订单自行设置期待时间，若超过用户的期待时间，订单自动取消。

2.3 系统界面及功能设计

RunningLeg 共分七个页面，首先用户登录授权之后进入，个人信息完善页面，如图 2-3 所示，在个人信息页展示微信授权获取到的用户微信昵称与头像展示在页面上部，并且询问用户获得用户地理位置展示在个人信息页。在页面下部有四个选项：账号信息图 2-4、订单状态图 2-5、钱包（支付功能暂未实现）、历史记录（发单历史图 2-6 与接单历史图 2-7）

图 2-3

图 2-4

图 2-5

图 2-6

图 2-7

点击上方区域或点击账号信息可进入账号详情页面图 2-8 进行修改与管理。用户信息有用户名，手机号码，收货地址三项，用户名根据自己需求任意填写不可为空，手机号码需要填写合理数字否则部能通过修改，收货地址点击之后进入选取位置页面图 2-9，选取合适位置

图 2-8

图 2-9

在发单页面中用户看到如图 2-10 所示的信息，用于填写订单详细信息，送达地址与手机号码自动补全也可修改，时间采用 picker 选择图 2-11，位置同个人信息位置选择，备注信息选填，在所有除备注以外的所有信息都填写完整且无误（确保截至时间在现在时间之后图 2-12 展示错误信息）之后可以点击发单，发单完成后，弹框提示图 2-13 跳转首页查看订单展示信息或留在此页再次发单（未完成订单不可超过三份图 2-14 展示提示）

图 2-10

图 2-11

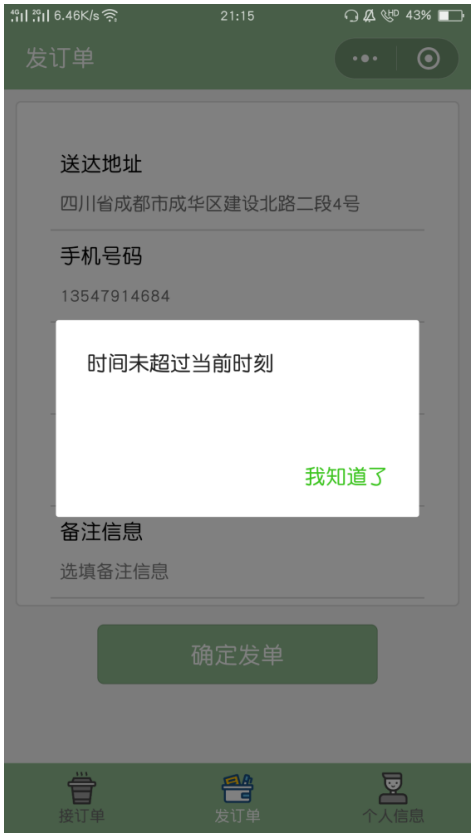


图 2-12

图 2-13

图 2-14

在接收订单页面，会根据实时调度展示适合区域内部分订单，如图 2-15 所示。右下角图标可直接跳转订单状态查看订单。点击每个订单会跳转至订单详情图 2-16，查看信息（此处信息不可修改）点击接单按钮即可接收订单，并给予接单提示，再跳转至首页订单展示页面。

图 2-15

图 2-16

2.4 运营设计

我们的微信小程序的定位是即时的校园跑腿平台。做的目的是为高校提供高效准确的跑腿服务，解决众多高校学生老师的跑腿需求。明显的优势是随着快节奏的生活，校园跑腿需求量的提高。暂时的运营目标实在本校实现小程序的推广，并逐渐扩散的众多其他高校。并以用户体验为核心理念，将微信小程序的即拿即用特性和跑腿及时性结合，提供一个好用的小程序。

2.5 小结

本章我们从系统总体设计、层次架构设计、核心算法设计、功能设计、界面设计五个方面对作品进行了分析设计。在总体架构设计中，表述了本作品的整体架构和层次架构。接着对本作品中提出的核心算法进行介绍。然后分别对本作品功能模块做了功能分析和流程设计。

第三章 作品实现

3.1 系统环境搭建

在我们的作品中，服务器采用阿里云 ECS，系统为 Ubuntu 操作系统（Ubuntu16.04），搭建在 Apache2 运行平台上，利用 mod_wsgi 插件，采用 Mysql 数据库，并申请了证书，进行 https 连接，主域名 https：//theeighthday.cn。

3.2 服务端实现

服务器开发语言为 Python，采用了当前流行的 Flask 框架，Flask 更加轻巧，非常适合小程序的敏捷开发。采用 ORM 建模,类及数据表的建立如图 3-1、图 3-2，账单表如图 3-3，用户表如图 3-4，API 接口及作用如表 3-1 所示。

```
class SendBill(db.Model):
    """ 发账单 """
    __tablename__ = 'sendbill'
    __table_args__ = {"extend_existing":True}
    id = db.Column(db.Integer, primary_key=True)
    send_user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    bill_id = db.Column(db.Integer, db.ForeignKey('bill.id'))


class ReceiveBill(db.Model):
    """ 接账单 """
    __tablename__ = 'receivebill'
    __table_args__ = {"extend_existing":True}
    id = db.Column(db.Integer, primary_key=True)
    receive_user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    bill_id = db.Column(db.Integer, db.ForeignKey('bill.id'))
```

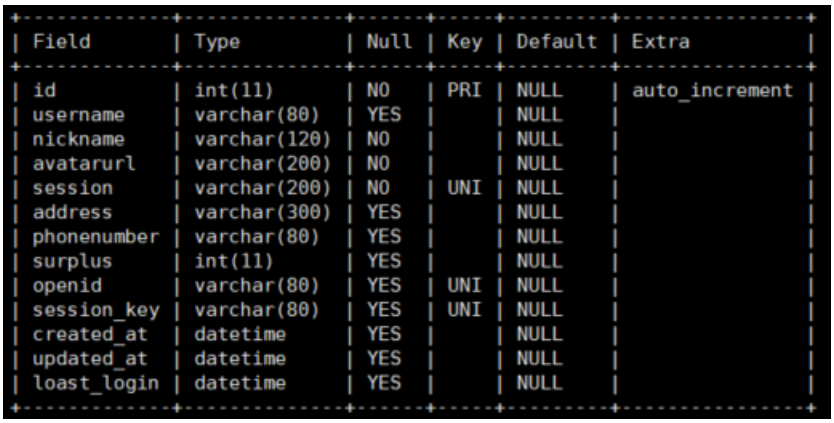
图 3-1

```python
class User(db.Model):
    """ 用户 """
    __tablename__ = 'user'
    __table_args__ = {"extend_existing":True}
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80))
    nickname = db.Column(db.String(120), nullable=False)
    avatarurl = db.Column(db.String(200), nullable=False)
    session = db.Column(db.String(200), unique=True, nullable=False)
    address = db.Column(db.String(300))
    phonenumber = db.Column(db.String(80))
    surplus = db.Column(db.Integer,default=0)
    openid = db.Column(db.String(80),unique=True)
    session_key =db.Column(db.String(80),unique=True)

    created_at = db.Column(db.DateTime)
    updated_at = db.Column(db.DateTime)
    loast_login = db.Column(db.DateTime)

    send_at = db.relationship('SendBill', backref='user')
    receive_from = db.relationship('ReceiveBill', backref='user')

    def __repr__(self):
        return '<User %r>' % self.username


class Bill(db.Model):
    """账单"""
    __tablename__ = 'bill'
    __table_args__ = {"extend_existing":True}
    id = db.Column(db.Integer, primary_key=True)
    send_username = db.Column(db.String(80), unique=True)
    send_phonenumber = db.Column(db.String(80))
    send_address = db.Column(db.String(300))
    goal_address = db.Column(db.String(300))
    remark = db.Column(db.String(500))
    hope_time = db.Column(db.DateTime)
    receive_username=db.Column(db.String(80))
    receive_phonenumber = db.Column(db.String(80))

    created_at = db.Column(db.DateTime)
    status = db.Column(db.Integer)
    finished_at = db.Column(db.DateTime)

    sentbill = db.relationship('SendBill', backref='bill')
    receivedbill = db.relationship('ReceiveBill', backref='bill')
```

图 3-2

```
mysql> show columns from bill;
+--------------------+-------------+------+-----+---------+----------------+
| Field              | Type        | Null | Key | Default | Extra          |
+--------------------+-------------+------+-----+---------+----------------+
| id                 | int(11)     | NO   | PRI | NULL    | auto_increment |
| send_username      | varchar(80) | YES  |     | NULL    |                |
| send_phonenumber   | varchar(80) | YES  |     | NULL    |                |
| send_address       | varchar(300)| YES  |     | NULL    |                |
| goal_address       | varchar(300)| YES  |     | NULL    |                |
| remark             | varchar(500)| YES  |     | NULL    |                |
| hope_time          | datetime    | YES  |     | NULL    |                |
| receive_username   | varchar(80) | YES  |     | NULL    |                |
| receive_phonenumber| varchar(80) | YES  |     | NULL    |                |
| created_at         | datetime    | YES  |     | NULL    |                |
| status             | int(11)     | YES  |     | NULL    |                |
| finished_at        | datetime    | YES  |     | NULL    |                |
+--------------------+-------------+------+-----+---------+----------------+
```

图 3-3



Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(80)	YES		NULL	
nickname	varchar(120)	NO		NULL	
avatarurl	varchar(200)	NO		NULL	
session	varchar(200)	NO	UNI	NULL	
address	varchar(300)	YES		NULL	
phonenumber	varchar(80)	YES		NULL	
surplus	int(11)	YES		NULL	
openid	varchar(80)	YES	UNI	NULL	
session_key	varchar(80)	YES	UNI	NULL	
created_at	datetime	YES		NULL	
updated_at	datetime	YES		NULL	
loast_login	datetime	YES		NULL	

图 3-4

/confirmbill	结束订单触发
/receivebill	接受订单触发
/createbill	创建订单触发
/getbill	获取最新 8 个未接订单触发
/updateuser	更新用户信息触发
/getsentbill	获取本人创建的被完成的订单触发
/getreceivedbill	获取本人接的已完成的订单触发
/getsendingbill	获取本人正在发的订单触发
/getreceivingbill	获取本人正在接订单触发
/registeruser	用户注册时触发
/userinfo	获取用户信息时触发

表 3-1　API 接口及作用

3.3 客户端实现

客户端共分三个模块：个人信息模块、发送订单模块、接收订单模块。个人信息模块中记录用户的用户名，手机号码，常用收货地址，后期会加上学校，用于筛选订

单。用户名做了判空处理如代码 3-1，如果未填写不通过修改，手机号码利用正则表达式处理，用户名手机号与地址有任意一个非法填入都不能修改信息代码 3-2

```
//wxml
<icon wx:if="{{!ishidden.username}}" type="warn" size="20"></icon>
//js
handleInput: function (e) {
    const type = e.currentTarget.dataset.type;
    if (type === 'username' || type === 'address') {
      if (e.detail.value === '') {
        let temp = this.data.ishidden;
        temp[type] = false;
        this.setData({
          ishidden: temp
        })
      }else {
        let temp = this.data.ishidden;
        temp[type] = true;
        this.setData({
          ishidden: temp
        })
      }
    } else if (type === 'phonenumber') {
      if (/^1[35789]\d{9}$/.test(e.detail.value)) {
        let temp = this.data.ishidden;
        temp[type] = true;
        this.setData({
          ishidden: temp
        });
      } else {
        let temp = this.data.ishidden;
        temp[type] = false;
        this.setData({
          ishidden: temp
        });
      }
    }
    this.setData({
      [type]: e.detail.value
    });
```

代码 3-1

```
chooseaddress:function(){
    wx.chooseLocation({
      success: (res) => {
        this.setData({
          address:res.address
        });
        if(res.address) {
            let temp = Object.assign({}, this.data.ishidden);
            temp.address = true;
            this.setData({
              ishidden: temp
            });
        } else {
          let temp = Object.assign({}, this.data.ishidden);
          temp.address = false;
          this.setData({
            ishidden: temp
          });
        }
      },
    })
  }
```

代码 3-2

发送订单模块，在首次进入页面更新用户数据，提前为用户填入地址与手机号码，代码 3-3，时间与地点选择如代码 3-4，发送订单绑定调用接口时间如代码 3-5。

```
onShow : function(){
    var app=getApp();
    if (app.globalData.userInfo.address==null){
      app.globalData.userInfo.address='';
    }
    if (app.globalData.userInfo.phonenumber == null) {
      app.globalData.userInfo.phonenumber = '';
    }
    this.setData({
      userInfo: app.globalData.userInfo,
      goal_address: app.globalData.userInfo.address,
      phonenumber: app.globalData.userInfo.phonenumber,
    })
  }
```

代码 3-3

```
bindTimeChange: function (e) {
    var myDate=new Date();
    var timearr = e.detail.value.split(":")
    var time = myDate.getFullYear() + ',' + (myDate.getMonth()+1) + ',' + myDate.getDate()
+','+timearr[0]+','+timearr[1]+','+'0'
    this.setData({
      hope_time: time,
      showtime: timearr[0] + '时' + timearr[1] + '分'
    })
  },
  chooseaddress: function () {
    var that = this;
    wx.chooseLocation({
      success: function (res) {
        console.log(res);
        that.setData({
          send_address: res.address
        });
      },
    })
  }
```

代码 3-4

```
sendbill: function () {
    var that = this;
    app.request({
      url: "https://theeighthday.cn/createbill",
      data: {
        "goal_address": that.data.goal_address,
        "hope_time": that.data.hope_time,
        "send_address": that.data.send_address,
        "remark": that.data.remark,
        "phonenumber": that.data.phonenumber,
      },
      success: function (res) {
        console.log(res)
        if (res.data.success ==1) {
          wx.showModal({
            content: '发单成功，现在去看？',
            cancelText: '再发一单',
            confirmText: "好的",
```

```
          success:(res) => {
            if (res.confirm) {
              wx.switchTab({
                url: './../ReceiveBill/ReceiveBill',
              });
            }
          }
        });

      }
      else{
        if (res.data.msg){
          wx.showModal({
            content: res.data.msg,
            confirmText: "我知道了",
            showCancel: false,
          });
        }else{
          wx.showModal({
            content: '信息填写有误，请确认后点击完成',
            confirmText: "我知道了",
            showCancel: false,
          });
        }
      }
    }
  })
}
```

代码 3-5

在接收订单模块，每次进入页面（如代码 3-6）与下拉刷新（如代码 3-7）都会重新获取数据，对于用户接单的操作进行轮询处理，如代码 3-8 展示

```
onShow: function() {
    this.getbill(true);
    this.animation = wx.createAnimation({
      duration: 150,
      timingFunction: 'ease-in-out',
      delay: 100,
      transformOrigin: 'top 0'
    })
    this.animation1 = wx.createAnimation({
      duration: 150,
      timingFunction: 'ease-in-out', //
"linear","ease","ease-in","ease-in-out","ease-out","step-start","step-end"
      delay: 100,
      transformOrigin: 'top 0 center 0 bottom 0'
    })
    this.animation.rotate(-3.5).step()
    this.setData({
      animation: this.animation.export()
    })
    var n=-1;
    setInterval(function () {
      // animation.translateY(-60).step()
      n = n*(-1);
      // k=(k+1)%2;
      // console.log(n);
      this.animation.rotate(7 * (n)).step()
      this.setData({
        animation: this.animation.export()
      })
      this.animation1.rotate(180 * (n)).step()
      this.setData({
        animation1: this.animation1.export()
      })
    }.bind(this), 310)
  }
```

代码 3-6

```
onPullDownRefresh :function(){
    this.getbill(true);
  },
  getbill: function () {
    var that = this;
    app.request({
      url: "https://theeighthday.cn/getbill",
      success: function (res) {
          if(res.data.success==1){
            console.log(res.data);
            let foodlist = res.data.data;
            foodlist.forEach((food) => {
              const dateObj = new Date(`${food.hope_time}+0800`);
              const date = dateObj.toLocaleDateString();
              const hour = dateObj.toLocaleTimeString();
              food.hope_time_format = `${date} ${hour}`;
            });
            that.setData({
              foodlist,
            });
            that.generateRandom();
            wx.stopPullDownRefresh();
            console.log(res.data);
          }
          else{
            console.info("获取订单失败");
          }

      },
      fail:function(){
        console.log("fail 获取订单失败")

      }
    })

  }
```

代码 3-7

```
onLaunch: function () {
    var that = this;
    var call= function () {
      that.request({
        url: "https://theeighthday.cn/getsendingbill",
        success: function (res) {
          if(res.data.success==1){
            that.globalData.sendingbill_id = res.data.data.map(item => item.id);
            var B = that.globalData.sendingbill_id;
            if (B.length < that.globalData.tem){
              console.log("有个订单被接啦");
              wx.showToast({
                title: '有个订单被接啦,点击小风车查看吧',
                duration: 2000
              })

            }
            that.globalData.tem = B.length;
            if (B.length!=0) {
              for (var i = 0; i < B.length; i++) {
                var id = B[i];
                that.request({
                  url: "https://theeighthday.cn/getsendingbilltime",
                  data: {
                    bill_id: B[i]
                  },
                  success: function (res) {
                    console.log("结束超时订单");
                  }
                })
              }
            }
          }
          else{
            console.log("获取订单失败");
          }
        }
      })
    };
    setInterval(call,2000);
  }
```

代码 3-8

第四章 作品测试

4.1 测试方案

本章对我们的微信小程序作品的实现进行详细测试，以验证本系统的准确性、实用性、高效性等特点。测试主要分为三部分：算法测试、功能测试以及性能测试。

对于算法方面，本章主要讲解测试距离上最短算法和时间上最优算法。

对于功能方面，本章主要为针对了本系统的三个核心功能模块—“接受订单模块”、“发送订单模块”、 “个人信息管理模块”的测试以及用户登录和个人信息修改模块的测试；

对于性能方面，本章分别从实时性、准确性、可移植性的角度进行了测试，在硬件配置方面选用了三类主流品牌的个人电脑并装配不同的操作系统；在软件配置方面，本章选用了六类流行的浏览器作为客户端测试工具，以验证本系统具有很好的性能；

本作品的测试均采用黑盒测试方法，将预先设置的正确结果和系统运行的结果进行对比，判定系统是否正确运行、是否达到预期目标。

4.2 测试环境

在测试环境方面，为充分考虑测试环境的复杂性、多样性和容错性。本章在硬件环境上选择了三款不同配置的个人电脑分别是惠普、mac、thinkpad以及三款不同配置的移动设备分别是华为、三星、iphone并各自配备了不用的操作系统；在客户端（浏览器）方面，本章测试了360浏览器、Firefox、IE 8、Chrome、360等浏览器。

4.3 算法测试

距离最短算法测试，根据用户设置的接单地址，优先展示离接单用户默认地址最近且未被接受的订单。具体测试见表 4-3-1

用例编号	NO.1		模块名称	测试距离最短算法
测试方式	黑盒测试		测试日期	2018.6.12
测试说明	帮助用户获取最近的订单			
预置条件	获取到订单			
判断标准	观察接单的地址			
测试输入	无			
测试输出	获取到的是离自己较近的未接订单			
测试评价	测试通过			

表 4-3-1

时间最优算法测试，根据用户设置的期望时间，优先展示离期望时间最近且未被接受的订单。具体测试见表 4-3-2

用例编号	NO.		模块名称	测试时间最优算法
测试方式	黑盒测试		测试日期	2018.6.12
测试说明	帮助用户获取离期望时间最近且未被接受的订单			
预置条件	获取到订单			
判断标准	观察接单的期望时间			
测试输入	无			
测试输出	获取离期望时间最近且未被接受的订单			
测试评价	测试通过			

表 4-3-2

4.4 功能测试

功能模块测试包括用户登录及个人信息获取测试、接受订单测试、发送订单测试、订单状态展示测试、个人信息修改测试、历史订单展示测试，这 6 大功能进行测试。

4.4.1 用户登录及个人信息获取测试

微信小程序通过微信本身获取用户的基本信息，如昵称、头像。此时微信小程序会提示个人信息不完整，需要用户手动完善。设计测试用例如表 4-1 所示

用例编号	NO.1		模块名称	用户登录及个人信息获取
测试方式	黑盒测试		测试日期	2018.6.12
测试说明	用户使用微信小程序进行授权			
预置条件	保持网络连接，同意授权			
判断标准	在个人页面中出现个人头像和昵称			
测试输入	无			
测试输出	在个人页面中出现个人头像和昵称			
测试评价	测试通过，成功获取用户基本信息			

表 4-1

成功获取个人信息页面如图 4-1

图 4-1

4.4.2 个人信息修改测试

用户授权后，此时微信小程序会提示个人信息不完整，需要用户手动完善用户名、手机号码、根据定位获取默认位置。设计测试用例如表 4-2 所示

用例编号	NO.2		模块名称	个人信息完善
测试方式	黑盒测试		测试日期	2018.6.12
测试说明	检测个人信息完善是否成功			
预置条件	手机号码格式正确			
判断标准	在个人页面不再出现不完善信息			
测试输入	用户名、手机号码、根据定位获取默认位置			
测试输出	点击完成按钮，出现成功弹框			
测试评价	测试通过，成功完善保存个人信息			

表 4-2

个人信息完善页面之前如图 4-2-1，完善后如图 4-2-2

图 4-2-1

图 4-2-2

4.4.3 发送订单测试

用户个人信息完善之后，用户就可以进行进行发送订单。设计测试用例如表 4-3 所示

用例编号	NO.3		模块名称	用户发送订单
测试方式	黑盒测试		测试日期	2018.6.12
测试说明	检测发送订单是否成功			
预置条件	接单时间在此刻之后，取货地址存在			
判断标准	出现成功弹框			
测试输入	接单时间、取货地址、选填备注信息			
测试输出	点击完成按钮，出现成功弹框			
测试评价	测试通过，成功发送订单			

表 4-3

发送订单页面如图 4-3-1，成功效果页面如图 4-3-2

图 4-3-1

图 4-3-2

4.4.4 接受订单测试

用户个人信息完善之后，用户就可以进行进行接受订单操作。设计测试用例如表

4-4 所示

用例编号	NO.4		模块名称	用户接受订单
测试方式	黑盒测试		测试日期	2018.6.12
测试说明	检测接受订单是否成功			
预置条件	点取某订单，并进行接单			
判断标准	出现成功弹框			
测试输入	无			
测试输出	点击完成按钮，出现成功弹框			
测试评价	测试通过，成功接受订单			

表 4-4

接受订单页面如图 4-4-1，成功效果页面如图 4-4-2

图 4-4-1

图 4-4-2

4.4.5 订单状态展示测试

用户接受或者发送订单而未结束时之后，用户可以查看订单状态的详细信息，同

时可以结束交易。设计测试用例如表 4-5 所示

用例编号	NO.5		模块名称	用户查看订单状态信息
测试方式	黑盒测试		测试日期	2018.6.12
测试说明	查看订单状态检测是否接受订单、发送订单是否成功			
预置条件	已接受或者发送订单			
判断标准	查看页面即可			
测试输入	无			
测试输出	无			
测试评价	测试通过			

表 4-5

订单状态展示页面如图 4-5

图 4-5

4.4.6 历史订单展示测试

用户接受或者发送订单结束时之后，用户可以查看订单历史状态的详细信息，同

时可以结束交易。设计测试用例如表 4-6 所示

用例编号	NO.6		模块名称	查看订单历史状态信息
测试方式	黑盒测试		测试日期	2018.6.12
测试说明	查看结束的业务操作			
预置条件	已接受或者发送订单操作结束或者正常完成交易			
判断标准	查看页面即可			
测试输入	无			
测试输出	无			
测试评价	测试通过			

表 4-6

接单历史页面如图 4-6-1，发单历史页面如图 4-6-2

图 4-6-1

图 4-6-2

4.5 性能测试

性能方面，我们从实时性，稳定性和可移植性角度对本系统进行了全面的性能测试。下面将分别进行介绍。

4.5.1 实时性

当此微信小程序同时有多人进行发单接单操作时，我们就需要进行多线程上的操作，从而保证多人访问时尽量减小系统带来的压力，保证每个用户操作的实用性。

从设置 200 人同时访问系统的压力分析，响应速度很快，完全在用户的感觉快速响应时间内，时间为：2.76 秒，首页的访问时间相比较长，首页部分图片和动画较多，如果用户量访问量继续加大，必定会影响系统性能。

4.5.2 稳定性

对此微信小程序的可视化数据信息，刷新获取不同的订单时。可见表 4-5-2，在网络负载加重的情况下，本系统性能基本稳定。

表 4-5-

并发线程数	每次时间间隔（秒）	平均响应时间（秒）	成功次数	失败次数	成功率	处理能力（次/分）	web 服务器 CPU 占用率（%）	数据库服务器 CPU 占用率（%）
1	0	4.365	30	0	100%	9.00	24.45%	50.45%
1	0	4.252	36	0	100%	9.20	28.45%	52.33%
1	0	4.785	50	0	100%	8.89	30.45%	51.11%
3	0	15.465	50	0	100%	9.89	44.46%	81.46%
3	0	16.002	50	0	100%	10.00	42.46%	88.36%
4	0	20.466	50	0	100%	10.00	42.98%	90.45%

4	0	21.498	50	0	100%	10.00	43.32%	90.95%
6	0	27.321	50	0	100%	10.00	45.11%	93.19%
6	0	27.981	50	0	100%	10.00	45.31%	96.05%
6	0	27.936	50	0	100%	10.00	45.99%	97.65%

4.5.3 可移植性

在不同的操作系统及不同的浏览器下的可移植性测试结果显示，本系统的客户端具有良好的移植性。测试结果如表 4-5-3-1，表 4-5-3-2 所示。

表 4-5-3-1

操作系统	Windows	Unix	Macintosh	Linux	IOS	Android
	√	√	√	√	√	√

表 4-5-3-2

浏览器	Safari	Firefox	Opera	Google Chrome	IE 8	360 浏览器	UC 浏览器
	√	√	√	√	√	√	√

4.6 小结

在本章，我们对作品进行了算法、功能和性能测试与分析，采用了黑盒测试技术和规范的方法对本系统进行了测试，确保了本系统的各项功能正常运行。

第五章 总结与展望

我们的作品结合了微信小程序本身的语法规则与界面统一的规则上进行 UI 设置等，本着简洁、轻便的原则上开发我们的微信小程序。我们的作品几乎完美的贴合与大学生校园的生活，并且它最初的意义也在于帮助、扶助部分大学生进行校内简单兼职工作。

在技术上我们采取 python 进行后台开发，结合前端技术和微信小程序接口和规则进行开发。在安全性上，前后台都进行了相关的校验，防止用户输入不合法或者敏感信息。后台采用距离优先算法，使得向用户展示的订单是离用户默认地址最近的，这一实现很大程度上节省接单者的时间，同时减少了发单者的等待时间。

但本作品仍有发展的空间，比如：

1. 在订单金额上面允许用户自己设置金额，而不是默认金额。
2. 钱包功能涉及的东西还没有开发完善，没有使用微信小程序接口设置钱包功能。
3. 整个页面 UI 设计上还需再完善一下。

之后我们还会将对本作品进行持续地开发和完善，最终完成一个简洁、实用、有意义存在的跑腿微信小程序平台。