Ambition To Do

Author: Wenxin Lai

Co-worker: Zhifei Chai

Finished Time: 2021/09/05

Archived in 2021/12/6

写在前面

这个项目是本科四年级第一学期的软件开发实习和舍友菲菲一起做的大作业，想法主要来源于市面上常见的番茄时钟，额外融合了“成就清单”软件的创意——完成设定任务所花费的时间可转化为积分，用于在欲望商店消费，以此可以更直观的激励用户，提高用户的工作效率。

成就清单是我在大一大二时就接触到的一款软件，它的逻辑是：自己设定任务和对应的积分奖励以及欲望项和对应的积分消耗，用奖励分去满足自己的欲望，实现一定程度的自我约束。在这之前我也有过模糊的这种类型的想法，所以发现这样的软件的时候感到非常惊喜，也有想着把它和专注提醒类的功能相结合，于是做了个这样的项目。

选择以微信小程序作为载体，是因为想接触下不同环境下的开发流程，微信小程序现在也非常的方便和流行。

该微信小程序基本页面参考了b站视频微信小程序实战——番茄时钟，但是代码都是新建项目独立完成，本人在编写相应的功能时都有预先独立思考和查阅其他博客，有些功能的实现方式完全不同。

Ⅰ、开发的准备

微信小程序提供了从0起步的微信小程序开放文档，在做小程序之前，我首先了解了一下小程序账号申请和框架介绍，然后就直接上手了，后续用到相关功能时再具体查阅。该文档讲的非常详细，推荐阅读。

我们的番茄时钟的功能实现主要分成四个大部分：**专注任务设置、倒计时、欲望满足和统计。**每个部分各写成一个页面，而每个页面的实现需要完成**页面结构**（使用wxml语言，类似html）、**页面样式**（使用wxss语言，类似css）和**页面逻辑**（使用JavaScript语言）的编写。以下将对这些部分分别进行介绍。

II、专注任务

如图所示，专注页是小程序的主页（相关文件在pages/index/index），包含了选择专注时间、选择任务分类的和开始专注任务并跳转到计时页的部分。此外，在页面中下部还附带“当前剩余专注积分”的提示。

在该页面下方还有导航栏，包含三个部分，一个是“代办”（即主页/专注页），一个是“欲望”，另一个是“统计”，可以随时跳转。

专注页结构分成五块。

i. 专注时间滑块

我们将滑条的范围为1-60（理解为我们的专注时间范围在1-60分钟），默认值为25（番茄学习法的标准时间）；

将滑条所代表的值传到对应JavaScript（后面简称为js）文件中的timing_time变量中保存，以便后面进行引用（在wxml文件中，可以利用”{{}}”引用同名js文件中的变量进行显示）和统计；

将拖动滑块这个事件与js文件中的TimingTimeRefresh函数进行绑定。该函数的功能是：当滑块被拖动时，timing_time的值也随之更新。也就是页面中所显示的专注时间能随着我们的调整进行变化，正常显示。

截图（不同的专注时间）：

相关代码：

```
<!--index.wxml-->
<view class="timing-slider">
  <slider min='1' max="60" value="{{timing_time}}" backgroundColor="#E7624F" show-value="true" bindchange="TimingTimeRefresh"></slider>
</view>
```

```
// index.js
Page({
  data: {
    timing_time: 25, // 专注时间
},

```

```
  TimingTimeRefresh(e){ //调整专注时间
    console.log(e)

    this.setData({
      timing_time:e.detail.value
    })
  },
})

```

```
/**index.wxss**/
.timing-slider{
  width: 600rpx;
  margin: auto;
}

```

ii. 提示标语

这一块就是页面提示，提醒用户选择专注任务，以及提示用户完成当前专注任务可以获得的专注积分。

相关代码：

```
<!--index.wxml-->
<view class="motto">
  <!-- 选择专注任务 -->
  <view style="margin: 20rpx auto; font-size: large;">{{motto_1}}</view>
  <!-- 完成任务以获得xx专注积分 -->
  <view style="margin: 20rpx auto; font-size: xx-small; color:  gray;">
{{motto_2}}{{timing_time}}专注积分</view>
</view>
```

```
// index.js
Page({
  data: {
    motto_1: '选择专注任务',
    motto_2: '完成任务以获得',
},
})

```

```
/**index.wxss**/
.motto{
  font-size: 30rpx ;
  text-align: center;
  margin: 30rpx 10rpx auto;
}

```

iii. 任务分类

我们将专注任务初步分成了六类：工作、学习、思考、写作、运动和阅读，并且将专注任务以列表的形式（cateArr）存在js文件中（带有两个属性，分类名字和text图标路径icon）。

在wxml中，我们利用微信小程序提供的遍历变量的接口wx.for，将分类中每一项以流式的方式（不会溢出画面）在页面上展示。

并且，我们为点击分类的事件绑定了分类切换的函数clickCate，将当前选择的分类突出显示为红色。

截图：

(选工作)

(选运动)

相关代码：

```
<!--index.wxml-->
<view class="task_cate">

  <view wx:for="{{cateArr}}" class="cate_item" wx:key="cate" bindtap="clickCate"
    data-index="{{index}}">
      <view class="cate_icon"> <image src="../../images/{{item.icon}}.png"> </image> </view>
      <view class='cate_text {{index == cateActive ? "cate_text_active" : ""}}'>{{item.text}}</view>
  </view>
</view>

```

```
// index.js
Page({
  data: {
    cateArr:[
      {
        icon: 'work',
        text: '工作'
      },
      {
        icon: 'study',
        text: '学习'
      },
      {
        icon: 'think',
        text: '思考'
      },
      {
        icon: 'write',
        text: '写作'
      },
      {
        icon: 'sport',
        text: '运动'
      },
      {
        icon: 'read',
        text: '阅读'
```

```
      }
    ],
    cateActive: '0',
  },

  clickCate:function(e){
   // console.log(e.currentTarget.dataset.index);
    this.setData({
      cateActive:e.currentTarget.dataset.index
    })
  },
})

```

```
/**index.wxss**/
.task_cate{
  width: 660rpx;
  margin: 5% auto 3%;
  display: flex;
  flex-wrap: wrap;
}
.task_cate .cate_item{
  width: 220rpx;
  height: 130rpx;
  text-align: center;
  margin-bottom: 50rpx;
}
.task_cate .cate_item .cate_icon{
  height: 70rpx;
}
.task_cate .cate_item .cate_icon image{
  width: 50rpx;
  height: 50rpx;
}
.task_cate .cate_item .cate_text{
  height: 60rpx;
  line-height: 60rpx;
  font-size: 30rpx;
}
.task_cate .cate_item .cate_text_active{
  color: #E7624F;
}

```

iv. 当前剩余积分

用户的记录是统计在本地缓存中的，包括积分的积累和欲望的消费。剩余积分的计算主要是通过读取本地缓存中关于积攒（如下logs）和消费（如下consumed）的记录，然后将获得的积分累加，减去消费掉的积分得到。

关于本地缓存的读取，下面细说。

（参考下方相关代码中的index.js）

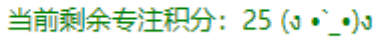
微信小程序中提供wx.getStorage和wx.getStorageSync命令，可以让我们从地缓存中异步获取指定key的内容。其中带Sync后缀表示的是以同步的方式从缓存中存或取，这种方式以牺牲速度的代价来保证数据的安全，只有当数据成功的存取之后才会进行后续的操作。

于是，我们用 wx.getStorageSync('logs')和wx.getStorageSync('consumed')取出类型为logs和consumed的缓存记录到var logs和var consumed里。其中logs是记录专注事件的，consumed是记录欲望花销的。（这个类型是根据我们怎么存的来的，是人为规定的）

这里需要提一下，我们在读取缓存的时候，在后面加了个 || []，我经过实践之后的理解是：当读取的数据为空时，wx.getStorageSync返回的是'undefined'，会导致后面引用该数据时因为类型不匹配报错，为了避免这个情况，利用"|| []"，使读取数据为空时，变量得到的是一个空列表的赋值。

此外，我们在存记录时，是用unshift往单条记录里存入键值对。于是我们就可以直接利用logs.length获取已记录的专注事件的个数，循环遍历其中的每一个，用logs[i].credit提取其中第i条记录的键credit所对应的值，将所有获得的积分累加起来。同理，得到已花费的总积分，两者相减便得到了当前剩余的积分。然后在wxml页面引用js中计算到的剩余积分的值就可以了。

截图：



相关代码：

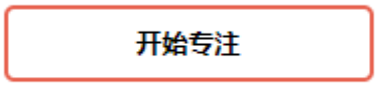
```
<!--index.wxml-->
<view style="margin: 0 auto; font-size: small; color: green;">当前剩余专注积分：{{remain_credit_index}} (ง •`_•)ง</view>

```

```
// index.js
Page({
  data: {
remain_credit_index: null,

},

  onShow() {

    //加载当前剩余专注积分

    var logs = wx.getStorageSync('logs') || [];
    var consumed = wx.getStorageSync('consumed') || [];
    var remain_credit = 0;
    var all_credit = 0;
    var consumed_credit = 0;

    if(logs.length > 0){
        for(var i = 0;i < logs.length;i++){
          all_credit += logs[i].credit;
        }
    };

    if(consumed.length > 0){
      for(var i = 0;i < consumed.length;i++){
        consumed_credit += consumed[i].credit;
      }
    };

    remain_credit = all_credit - consumed_credit;

    this.setData({
        remain_credit_index: remain_credit,
    })
  },

})

```

v. 开始专注按钮

这一块主要是为按钮绑定一个跳转到计时页的功能函数go_to_timing_page，当我们单击这个按钮时，就能通过wx.navigateTo定向到计时页（pages/timing/timing），并且向计时页传递专注时间和任务分类，以便在计时页的计时结束时，可以直接往本地缓存写入统计记录。

截图：



相关代码：

```
<!--index.wxml-->
<view>
  <button class="button" bindtap="go_to_timing_page"> 开始专注 </button>
</view>
```

```
// index.js
Page({
  go_to_timing_page: function(){
    var that = this
    // console.log(that.data.cateActive)
    wx.navigateTo({
      url: '../../pages/timing/timing?timing_time=' + that.data.timing_time + '&cateActive=' + that.data.cateActive,
    })
  },

})

```

Ⅲ、计时页

该页是单独的页面，只能从专注页进入。计时页分为两个部分，一个是**倒计时时钟**（包含倒计时时间和进度圈的显示），一个是下面的**按钮栏**（包括：暂停、继续、中止、结束四种按钮，在不同的状态下显示）。

i. 倒计时时钟

这个部分包含倒计时时间和进度圈的显示。

首先，我们从专注页进入到倒计时页的时候会传入需要专注的时间，这个就作为我们的初始倒计时分钟数。后面这个时间会根据倒计时功能函数进行更新。当倒计时顺利结束后，这块文字会变成“你已获得xx专注积分”，如下图所示。

然后先讲讲圆形进度条的绘制，这是在wxml中用canvas标签（图形容器）与js中编写的绘制函数（绘制脚本）绑定来实现的。圆形进度条分成两个部分，一个是黑色的背景环，一个是不断变化的白色进度环，我们需要用两个函数完成这两个环的绘制（下面贴的相关代码中，timing.js里的，drawBkdg函数对应背景环的绘制，drawDnmc函数对应进度环的绘制）。

画圆首先要确定圆心位置、半径、起始点和终止点，由于我们在wxss中预先定义过了该块视图所占的总长宽（400rpx*400rpx。rpx是微信特有的一种尺寸计算单位，是动态规定的，所有的机型的宽度设为750rpx，作为标准），所以这些都比较好设定。再然后就是引入一个变量rate_of_progress计算当前倒计时占总时间的比了，就能确定白色进度环的终止点了。

背景环在计时页加载出来的时候就进行绘制，而进度环后续绘制。

最后是编写倒计时的功能函数countDown。逻辑是，设定一个计时器变量Timer，调用setInterval来实现按照指定的周期（以毫秒计）来调用内部函数。在内部函数中，进行倒计时减一以及分秒分开显示的换算，并且同时调用进度环的绘制函数更新进度圈的更新。

相关代码：

```
<!--timing.wxml-->
<view class="countdownContainer" >
<canvas class="counting_circle" canvas-id="bkgd_circle"></canvas>
<canvas class="counting_circle" canvas-id="dnmc_circle"></canvas>
<view class="counting_number" wx:if="{{!end_status}}">
{{countdown_min}}:{{countdown_sec}}</view>
<view class="ending_tips" wx:if="{{end_status}}">你已获得
{{counting_seconds/60}}专注积分</view>
</view>
```

```
// timing.js
Page({
```

```
  data: {
counting_seconds: null, //该次任务总时长

    circle_size: 400,
    countdown_seconds: null, //当前倒计时（秒为单位）
    countdown_min: null, //当前倒计时（剩余分钟）
    counting_sec: null, //当前倒计时（剩余秒钟）

rate_of_progress: 0, //记录当前计时进度百分比

Timer: '',
},


  onLoad: function (options) {
    this.setData({
      counting_seconds: options.timing_time*60,
      countdown_seconds: options.timing_time*60,
      countdown_min: options.timing_time*60/60,
      countdown_sec: options.timing_time*60%60,

      cateActive: options.cateActive,
      rate_of_progress: 0,
    }),

    // console.log(this.data.cateActive)

    this.drawBkdg();
  },

  drawBkdg: function () {

    //获取全局变量“比例”
    // var app = getApp()
    const ratee = app.globalData.rate;
    var line_width = 15/ratee;

    //画圆
    var ctx = wx.createCanvasContext('bkgd_circle');
    ctx.setLineWidth(line_width);
    ctx.setStrokeStyle("#000000");
    ctx.setLineCap('round');
    ctx.beginPath();
```

```
    // ctx.arc(400/ratee/2,400/ratee/2,400/ratee/2-line_width,1.5*Math.PI,2*Math.PI);
    ctx.arc(400/ratee/2,400/ratee/2,400/ratee/2-line_width,2*Math.PI);
    ctx.closePath();
    // ctx.fill();//填充
    ctx.stroke();//边缘
    ctx.draw();
  },

  drawDnmc: function () {

    //获取全局变量“比例”
    // var app = getApp()
    const ratee = app.globalData.rate;
    var line_width = 15/ratee;

    //画圆
    var ctx = wx.createCanvasContext('dnmc_circle');
    ctx.setLineWidth(line_width);
    ctx.setStrokeStyle("#FFFFFF");
    ctx.setLineCap('round');
    ctx.beginPath();
    // ctx.arc(400/ratee/2,400/ratee/2,400/ratee/2-line_width,1.5*Math.PI,1.5*Math.PI+this.data.rate_of_progress*2*Math.PI);
    ctx.arc(400/ratee/2,400/ratee/2,400/ratee/2-line_width,1.5*Math.PI,(1.5+this.data.rate_of_progress*2)*Math.PI);

    // ctx.fill();//填充
    ctx.stroke();//边缘
    ctx.draw();
  },

  onShow: function() {
    this.countDown();
  },

  countDown: function () {
    var that = this;
    var countDownNum = that.data.countdown_seconds;//获取倒计时初始值

    that.setData({
```

```
      Timer: setInterval(function () {//这里把setInterval赋值给变量名为timer的变量

        if(!that.data.pause_status){
          //每隔一秒countDownNum就减一，实现同步
          countDownNum--;
          //然后把countDownNum存进data，好让用户知道时间在倒计着
          that.setData({
            countdown_seconds: countDownNum,
            countdown_min: Math.floor(countDownNum/60),
            countdown_sec: countDownNum%60,
            rate_of_progress: 1-countDownNum/that.data.counting_seconds,
          })
          //在倒计时还未到0时，这中间可以做其他的事情，按项目需求来

          that.drawDnmc();

          if (countDownNum <= 0 || that.data.rate_of_progress>=1) {
            //关闭定时器
            that.drawDnmc();
            clearInterval(that.data.Timer);
            that.setData({
              end_status: true,
            })
          }
        }

      }, 10)  //*单位为毫秒，多少毫秒更新一次的意思，记得改回1000(测试的时候用倍速，正常的话是1000ms更新一次，使倒计时减1s)
    })
  },

})
```

```
/**timing.wxss**/

page{
  background-color: #E7624F;
}

.countdownContainer {
  width: 400rpx;
  height: 400rpx;
```

```
  /* background-color: aliceblue; */
  margin-top: 15% auto;
  position: relative;
}

.countdownContainer .counting_circle {
  width: 400rpx;
  height: 400rpx;
  position: absolute;
  left: 0;
  top: 0 ;
}

.counting_number {
  position: relative;
  top: 43%;
  font: bolder;
  text-align: center;
  font-size: large;
}
```

ii. 按钮栏

这一部分包括：**暂停、继续、中止、结束**四种功能按钮，这四种按钮在不同的状态下显示。

暂停按钮：当倒计时正常进行/结束时显示；按下之后切换到暂停状态，暂停按钮隐藏；对应js文件中的suspend_timing函数；

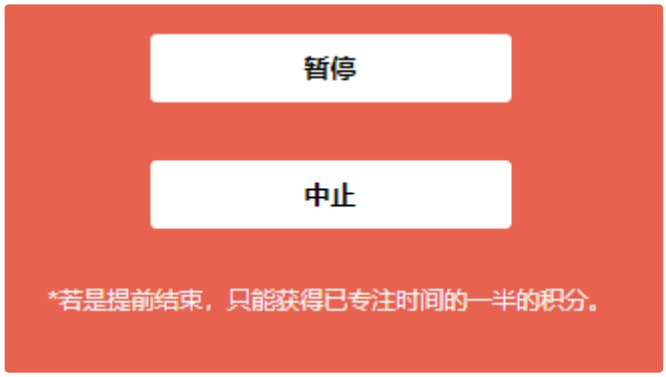
继续按钮：暂停状态显示；按下之后切换到倒计时状态，继续按钮隐藏；对应js文件中的resume_timing函数；

中止按钮：当倒计时没结束时显示；按下之后记录当前专注事件（当前时间、分类、已专注时间、获得的专注积分（中止倒计时得到的积分只有当前已专注时间的一半））；对应js文件中的abort_timing函数；

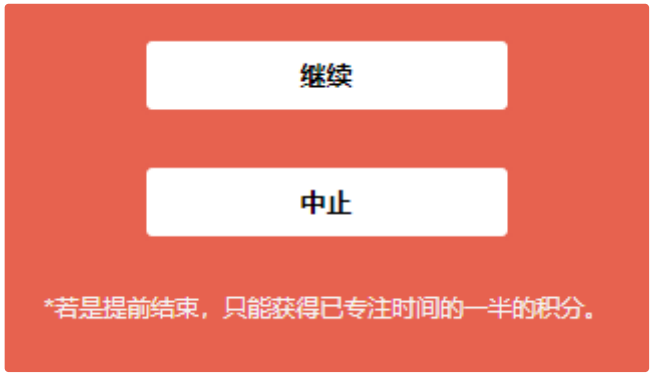
结束按钮：当倒计时结束的时候显示；按下之后记录当前专注事件（当前时间、分类、已专注时间、获得的专注积分），并且跳转回首页（专注页）；对应js文件中的finish_timing函数。

截图：

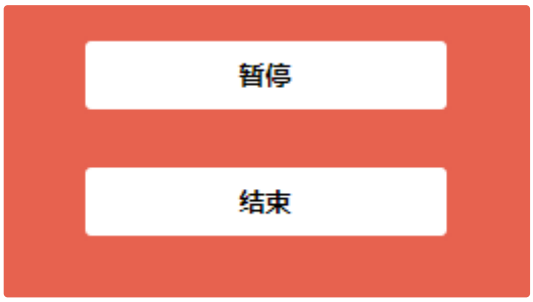
正常倒计时状态，未结束



停止状态，未结束



已结束



相关代码：

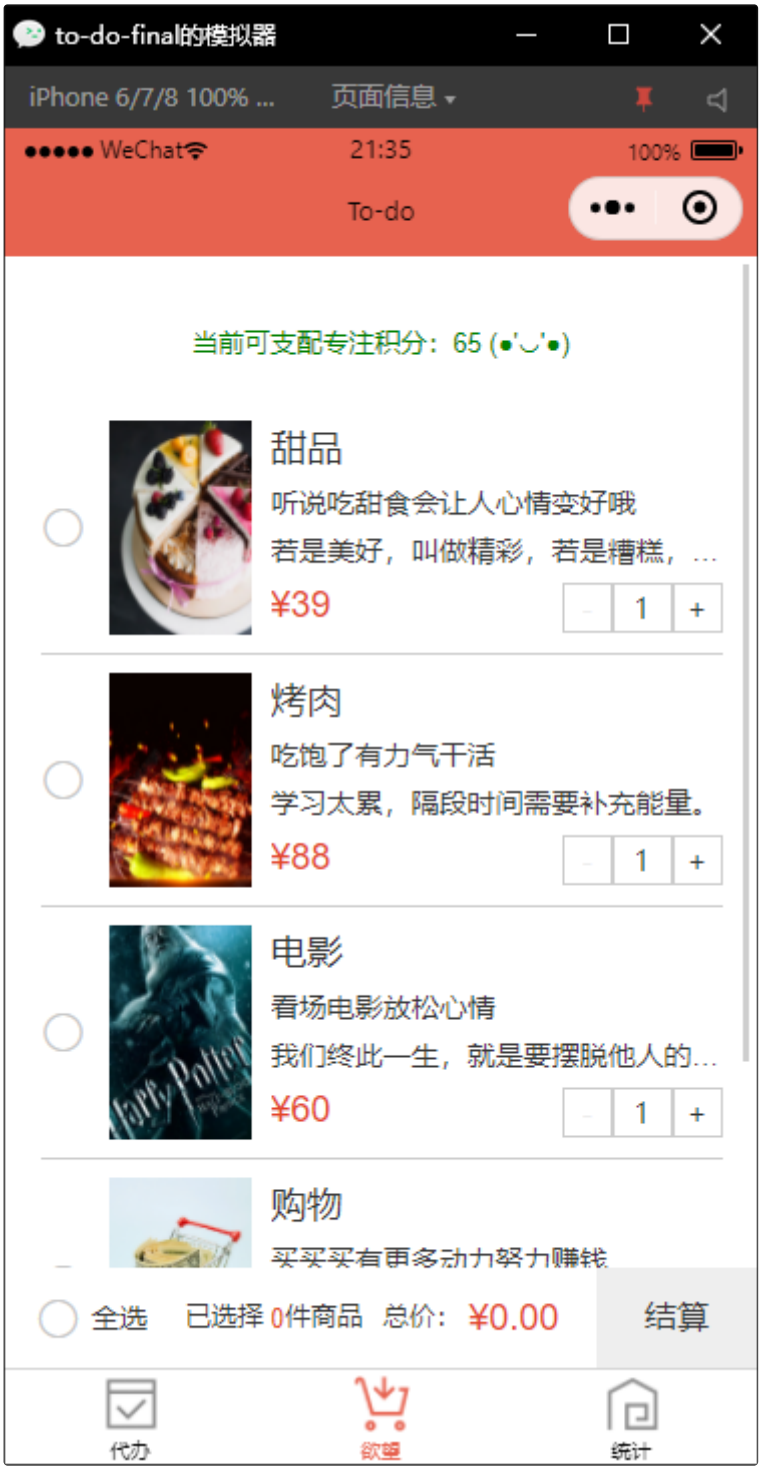
```
<!--timing.wxml-->
<view class="timingButton" style="margin: 10rpx auto;">
<button class="button" bindtap="suspend_timing" wx:if="{{!pause_status}}"> 暂停 </button>
<button class="button" bindtap="resume_timing" wx:if="{{pause_status}}"> 继续 </button>
<button class="button" bindtap="abort_timing" wx:if="{{!end_status}}"> 中止 </button>
<view style="margin: 50rpx auto; font-size: xx-small; color: whitesmoke;" wx:if="{{!end_status}}">*若是提前结束，只能获得已专注时间的一半的积分。</view>
<button class="button" bindtap="finish_timing" wx:if="{{end_status}}"> 结束 </button>
</view>
```

```
// timing.js
Page({
  data: {
},

onHide(){
    var that = this
    clearInterval(that.data.Timer);
  },

  suspend_timing: function() {
    this.setData({
      pause_status: 1,
    })
  },

  resume_timing: function() {
    this.setData({
      pause_status: 0,
    })
  },

  abort_timing: function() {
    var that = this;
    clearInterval(that.data.Timer);

    app.globalData.gained_credit += Math.floor((that.data.counting_seconds-that.data.countdown_seconds)/60/2); //*若是提前结束，只能获得已专注时间的一半的积分。
    app.globalData.remain_credit += Math.floor((that.data.counting_seconds-that.data.countdown_seconds)/60/2);

    // console.log(app.globalData.gained_credit)

    var logs = wx.getStorageSync('logs') || [];
    logs.unshift({
      date: util.formatTime(new Date),
      cate: that.data.cateActive,
      time: Math.floor((that.data.counting_seconds-that.data.countdown_seconds)/60),
```

```
      credit: Math.floor((that.data.counting_seconds-that.data.countdown_seconds)/60/2),
    })
    wx.setStorageSync('logs', logs);

    wx.switchTab({
      url: '/pages/index/index',
    })
  },

  finish_timing: function() {
    var that = this;
    clearInterval(that.data.Timer);

    app.globalData.gained_credit += that.data.counting_seconds/60;
    app.globalData.remain_credit += that.data.counting_seconds/60;

    console.log(app.globalData.gained_credit)

    var logs = wx.getStorageSync('logs') || [];
    logs.unshift({
      date: util.formatTime(new Date),
      cate: that.data.cateActive,
      time: that.data.counting_seconds/60,
      credit: that.data.counting_seconds/60,
    });
    console.log(logs.cate)
    wx.setStorageSync('logs', logs);

    wx.switchTab({
      url: '/pages/index/index',
    })

})
```

```
/** timing.wxss**/

.ending_tips {
  position: relative;
  top: 43%;
  font: bolder;
  text-align: center;
  font-size: small;
}

.timingButton {
  margin:0 auto;
}
```

IV、欲望页



欲望页也设置为底下导航栏的一项，可以随时跳转。该页主要包含三个部分，**剩余积分提示、欲望商品、结算栏**，页面的设计比较像淘宝等购物软件的购物车，意在激发用户的欲望，而更努力地去专注积攒积分换取欲望商品。

i. 剩余积分提示

该部分与专注页的剩余积分提示的部分一样，不再赘述。

截图：

当前可支配专注积分：65 (●'◡'●)

相关代码：

```
<!--ambition.wxml-->

<view style="margin: 70rpx auto 40rpx; text-align: center; font-size: small; color: green;">当前可支配专注积分：{{remain_credit_ambition}}（●'◡'●）</view>

```

```
// ambition.js
Page({
  data: {
"remain_credit_ambition": 0,
},

  onShow: function () {

      //加载当前剩余专注积分

      var logs = wx.getStorageSync('logs') || [];
      var consumed = wx.getStorageSync('consumed') || [];
      var remain_credit = 0;
      var all_credit = 0;
      var consumed_credit = 0;

      if(logs.length > 0){
        for(var i = 0;i < logs.length;i++){
          all_credit += logs[i].credit;
        }
      };

      if(consumed.length > 0){
       for(var i = 0;i < consumed.length;i++){
         consumed_credit += consumed[i].credit;
       }
      };

      remain_credit = all_credit - consumed_credit;

      this.setData({
```

```
            remain_credit_ambition: remain_credit,
        })
    },

})
```

ii. 欲望商品和结算栏

这里将常见的欲望写在一个列表'goodList'里，其中每一项带有商品名、商品标语、商品描述、商品图片、商品数目、商品单价、商品是否选择等参数。

定义好商品之后，就能进行页面的显示排版了。我们采用了类似淘宝购物车的方式，将每个商品列在欲望车里。

然后，就需要编写其中的一些功能函数，比如：

checkboxChange单击商品前的圆圈以勾选/取消勾选商品（改变该商品的是否被选择的参数），并且更新结算栏的状态，如果当前已有选择的商品，结算键亮起；

subtracttap&addtap单击加减号改变物品购买个数，最少数量为1；

selectalltap全选商品；

calculateTotal统计当前已选择商品的个数和总价；

settlement结算函数（若当前商品总额超出所剩余的积分，则不允许购买；如果够，那么写入欲望满足记录（包含当前时间、欲望商品名字、单个所需积分、数量））；

相关代码：

```
<!--ambition.wxml-->

<view class='container'>

  <!-- 商品栏 -->
  <view class='section section-good'>
    <checkbox-group bindchange="checkboxChange">
      <view class='good' wx:for='{{goodList}}' wx:for-item="good" wx:key="good.name">

        <label class="checkbox">
          <checkbox value="{{good.isbn}}" checked="{{good.checked}}" hidden='hidden' />
          <icon type="circle" size="23" wx:if="{{!good.checked}}"></icon>
          <icon type="success" size="23" wx:if="{{good.checked}}"></icon>
```

```
14        </label>
15
16        <image class='cover' src='{{good.cover}}'></image>
17
18        <view class='content'>
19          <view class='text name'>{{good.name}}</view>
20          <view class='text author'>{{good.author}}</view>
21          <view class='text desc'>{{good.desc}}</view>
22          <view class='text price'>¥{{good.price}}</view>
23        </view>
24
25        <view class='stepper'>
26          <view class='subtract {{good.count == 1 ? "disabled": ""}}' data-
index='{{index}}' catchtap='subtracttap'>-</view>
27          <input class='count' type='number' value='{{good.count}}'>
</input>
28          <view class='add' data-index="{{index}}" catchtap='addtap'>+
</view>
29        </view>
30
31      </view>
32    </checkbox-group>
33  </view>
34
35  <!-- 结算栏 -->
36  <view class='section-bottom'>
37
38    <checkbox-group bindchange="selectalltap">
39      <label class='checkbox-allcheck'>
40        <checkbox value="{{!checkAll}}" checked="{{checkAll}}"
hidden='hidden' />
41        <icon type="circle" size="23" wx:if="{{!checkAll}}"></icon>
42        <icon type="success" size="23" wx:if="{{checkAll}}"></icon>
43        <text class='check-all-text'>全选</text>
44      </label>
45    </checkbox-group>
46
47    <view class="total">
48      <view class='totalCount'>已选择
49        <text>{{totalCount}}</text>件商品
50      </view>
51      <view class='totalPrice'>总价:
52        <text>¥{{totalPrice}}</text>
```

```
    </view>
  </view>

  <view class='btn {{totalCount > 0 ? "btn-primary" : "btn-default"}}' bindtap="settlement">结算</view>

 </view>

</view>

```

```
// ambition.js
Page({
  data: {
      'goodList': [
      {
        'name': '甜品',
        'author': '听说吃甜食会让人心情变好哦',
        'cover': '/images/cover_1.png',
        'desc': '若是美好，叫做精彩，若是糟糕，叫做经历。',
        'press': '',
        'price': 39,
        'count': 1,
        'checked': false
      },
      {
        'name': '烤肉',
        'author': '吃饱了有力气干活',
        'cover': '/images/cover_2.png',
        'desc': '学习太累，隔段时间需要补充能量。',
        'press': '',
        'price': 88,
        'count': 1,
        'checked': false
      },
      {
        'name': '电影',
        'author': '看场电影放松心情',
        'cover': '/images/cover_3.png',
        'desc': '我们终此一生，就是要摆脱他人的期待，找到真正的自己。',
        'press': '',
        'price': 60,
        'count': 1,
```

```
        'checked': false
      },
      {
        'name': '购物',
        'author': '买买买有更多动力努力赚钱',
        'cover': '/images/cover_4.png',
        'desc': '或许，命运就是一条孤独的河流，我们都会遇见灵魂的摆渡人。',
        'press': '',
        'price': 36,
        'count': 1,
        'checked': false
      },
      {
        'name': '游戏',
        'author': '游戏一小时放松一下',
        'cover': '/images/cover_5.png',
        'desc': '人一生中最可怕是无所事事，最可恨是无所追求，最可悲是无所作为。',
        'press': '',
        'price': 10,
        'count': 1,
        'checked': false
      }
    ],
    'checkAll': false,
    'totalCount': 0,
    'totalPrice': 0,

    "remain_credit_ambition": 0,

    //存放购买的各项物品的信息，用于历史统计
    "buy_list_name": [],
    "buy_list_credit": [],

},

/**
   * 生命周期函数--监听页面初次渲染完成
   */
  onReady: function () {
    this.calculateTotal();

  },
```

```
75
76
77  /**
78     * 计算商品总数
79     */
80    calculateTotal: function () {
81      var goodList = this.data.goodList;
82      var totalCount = 0;
83      var totalPrice = 0;
84      for (var i = 0; i < goodList.length; i++) {
85        var good = goodList[i];
86        if (good.checked) {
87          totalCount += good.count;
88          totalPrice += good.count * good.price;
89        }
90      }
91      totalPrice = totalPrice.toFixed(2);
92      this.setData({
93        'totalCount': totalCount,
94        'totalPrice': totalPrice
95      })
96    },
97
98    /**
99     * 用户点击商品减1
100    */
101   subtracttap: function (e) {
102     var index = e.target.dataset.index;
103     var goodList = this.data.goodList;
104     var count = goodList[index].count;
105     if (count <= 1) {
106       return;
107     } else {
108       goodList[index].count--;
109       this.setData({
110         'goodList': goodList
111       });
112       this.calculateTotal();
113     }
114   },
115
116   /**
117    * 用户点击商品加1
```

```
118      */
119     addtap: function (e) {
120       var index = e.target.dataset.index;
121       var goodList = this.data.goodList;
122       var count = goodList[index].count;
123       goodList[index].count++;
124       this.setData({
125         'goodList': goodList
126       });
127       this.calculateTotal();
128     },
129     /**
130      * 用户选择购物车商品
131      */
132     checkboxChange: function (e) {
133       console.log('checkbox发生change事件，携带value值为：',
    e.detail.value);
134       var checkboxItems = this.data.goodList;
135       var values = e.detail.value;
136       for (var i = 0; i < checkboxItems.length; ++i) {
137         checkboxItems[i].checked = false;
138         for (var j = 0; j < values.length; ++j) {
139           if (checkboxItems[i].isbn == values[j]) {
140             checkboxItems[i].checked = true;
141             break;
142           }
143         }
144       }
145 
146       var checkAll = false;
147       if (checkboxItems.length == values.length) {
148         checkAll = true;
149       }
150 
151       this.setData({
152         'goodList': checkboxItems,
153         'checkAll': checkAll
154       });
155       this.calculateTotal();
156     },
157 
158     /**
159      * 用户点击全选
```

```
160     */
161    selectalltap: function (e) {
162      console.log('用户点击全选，携带value值为：', e.detail.value);
163      var value = e.detail.value;
164      var checkAll = false;
165      if (value && value[0]) {
166        checkAll = true;
167      }
168
169      var goodList = this.data.goodList;
170      for (var i = 0; i < goodList.length; i++) {
171        var good = goodList[i];
172        good['checked'] = checkAll;
173      }
174
175      this.setData({
176        'checkAll': checkAll,
177        'goodList': goodList
178      });
179      this.calculateTotal();
180    },
181
182    settlement: function(e){
183      var that = this;
184
185      //够积分
186      if(that.data.totalPrice <= that.data.remain_credit_ambition){
187        console.log("够钱")
188        var goodList = that.data.goodList;
189
190        for(var i = 0; i < goodList.length; i++){
191          var good = goodList[i];
192
193          if(good.checked){
194            var consumed = wx.getStorageSync('consumed') || [];
195            consumed.unshift({
196              date:util.formatTime(new Date),
197              cate:good.name,
198              number:good.count,
199              credit:good.count*good.price,
200            });
201            console.log(consumed.name)
202            wx.setStorageSync('consumed',consumed);
```

```
          }
        };
        this.onShow();
      }else{
        console.log("不够钱");
        this.onShow();
      }

    },
})
```

```
/* ambition.wxss */

/* 这个部分太多了，跟主要的功能没啥关联，就是排版，就不贴了 */
```

V、统计页

(上图有个bug，显示的是历史专注事件，忘记刷新了T T)

统计页分成两个部分，一个部分是**主要的数据统计**，另一个部分是**专注/欲望的今日/历史记录**。

i. 主要的数据统计

这一部分的数据统计主要通过本地缓存来计算，跟前面专注页的剩余专注积分的计算有异曲同工之处，不详细写了。（功能实现在logs.js文件里的 onShow函数里）

今日番茄次数是通过匹配记录里面条目的日期和当前日期是否一致来计算的，还用到了util.js工具中的时间标准化。

截图：


今日番茄次数
2
累计番茄次数
4
今日专注时长
65分钟
累计专注时长
77分钟
已经花费积分
10分
累计专注积分
75分

相关代码：


```wxml
<!--logs.wxml-->
<view class="sum">
  <view class="sum_item" wx:for="{{sum}}" wx:key="sum">
  <!-- wx:for用于循环数组，所以下面的是遍历js文件里sum数组内的item -->
    <view class="sum_item_title">{{item.title}}</view>
    <view class="sum_item_val">{{item.val}}</view>
  </view>
</view>

```

```
//logs.js
```

```
const util = require('../../utils/util.js')

Page({
  data: {
    logs: [],
    dayList:[],
    dayList_ambitionL:[],
    list:[],
    list_ambition:[],

    list_show:[],

    sum:[
      {
        title:'今日番茄次数',
        val:'0'
      },
      {
        title:'累计番茄次数',
        val:'0'
      },
      {
        title:'今日专注时长',
        val:'0分钟'
      },
      {
        title:'累计专注时长',
        val:'0分钟'
      },
      {
        title:'已经花费积分',
        val:'0分'
      },
      {
        title:'累计专注积分',
        val:'0分'
      },
    ],
  },
  onShow: function () {
    // this.setData({
    //   logs: (wx.getStorageSync('logs') || []).map(log => {
    //     return util.formatTime(new Date(log))
```

```
    //   })
    // })

    var logs = wx.getStorageSync('logs') || [];
    var consumed = wx.getStorageSync('consumed') || [];

    var day = 0; // 今日番茄次数
    var total = logs.length; //累计番茄次数
    var dayTime = 0; // 今日专注时长
    var totalTime = 0; // 累计专注时长
    var consumed_credit = 0; // 已经花费积分
    var all_credit = 0; // 累计专注积分

    var dayList = [];
    var dayList_ambition = [];

    //统计专注事件的部分
    if(logs.length > 0){
      for(var i = 0;i < logs.length;i++){

        // 今日的
        if(logs[i].date.substr(0,10) == util.formatTime(new
Date).substr(0,10)){
          day = day + 1;
          dayTime = dayTime + parseInt(logs[i].time);
          dayList.push(logs[i]);
        };

        // 累计的
        all_credit = all_credit + parseInt(logs[i].credit);
        totalTime = totalTime + parseInt(logs[i].time);
      }
    };

    //统计欲望花费的部分
    if(consumed.length > 0){
      for(var i = 0;i < consumed.length;i++){

        // 今日的
        if(consumed[i].date.substr(0,10) == util.formatTime(new
Date).substr(0,10)){
          dayList_ambition.push(consumed[i]);
        };
```

```
        //累计的
        consumed_credit += consumed[i].credit;
      };
    };

    //传数值
    this.setData({
      'sum[0].val':day,
      'sum[1].val':total,
      'sum[2].val':dayTime+'分钟',
      'sum[3].val':totalTime+'分钟',
      'sum[4].val':consumed_credit+'分',
      'sum[5].val':all_credit+'分',

      dayList:dayList,
      dayList_ambition:dayList_ambition,
      list:logs,
      list_ambition:consumed,

      list_show: logs, //默认显示“今日” “专注”

    })
  },
})
```

```
// util.js

const formatTime = date => {
  const year = date.getFullYear()
  const month = date.getMonth() + 1
  const day = date.getDate()
  const hour = date.getHours()
  const minute = date.getMinutes()
  const second = date.getSeconds()

  return `${[year, month, day].map(formatNumber).join('/')} ${[hour, minute, second].map(formatNumber).join(':')}`
}

const formatNumber = n => {
  n = n.toString()
  return n[1] ? n : `0${n}`
```

```
}

module.exports = {
  formatTime
}

```

ii. 专注/欲望的今日/历史记录

本部分主要是通过wx.getStorageSync获取本地缓存的“专注”事件和“欲望”花费的记录，在专注页的**显示当前剩余专注积分**功能实现里面已经详细说明了，这里也不再赘述。

主要是添加了当前分类栏的切换功能。分类统计分成了两类，一类是专注/欲望，另一类是今日/历史，然后我们将点击的事件分别绑定了changeType 和changeType2的功能函数，在点击该项时做出相应的切换，并且显示相应的条目。此外，如果没有相关的条目，则显示“暂无数据”。

截图（截图日期为2021年9月6日）：

今日专注

历史专注

今日欲望

历史欲望

相关代码：

```
<!--logs.wxml-->

```

```
<view class="detail">
  <view class="detail_title">
    <view class="detail_title_text">分类统计</view>
    <view class="detail_title_type">
      <text class="{{activeIndex2 == 2 ? 'active':''}}" data-index="2" bindtap="changeType2">专注</text>
      <text  style="color:darkgray"> | </text>
      <text class="{{activeIndex2 == 3 ? 'active':''}}" data-index="3" bindtap="changeType2">欲望</text>

      <text  style="size:10rpx; color:blue;"> | </text>

      <text class="{{activeIndex == 0 ? 'active':''}}" data-index="0" bindtap="changeType">今日</text>
      <text  style="color:darkgray"> | </text>
      <text class="{{activeIndex == 1 ? 'active':''}}" data-index="1" bindtap="changeType">历史</text>
    </view>
  </view>


  <view class="detail_list" wx:if="{{list_show.length > 0}}">

    <view class="list_item" wx:for="{{list_show}}" wx:if="{{activeIndex2 == 2}}">
      <view class="list_item_date">{{item.date}}</view>
      <view class="list_item_cate">{{cateArr[item.cate].text}}</view>
      <view class="list_item_time">{{item.time}}分钟 {{item.credit}}积分</view>
    </view>

    <view class="list_item" wx:for="{{list_show}}" wx:if="{{activeIndex2 == 3}}">
      <view class="list_item_date">{{item.date}}</view>
      <view class="list_item_cate">{{item.cate}}</view>
      <view class="list_item_time">{{item.credit}}积分*{{item.number}}</view>
    </view>
  </view>

  <view class="detail_list" wx:if="{{list_show.length == 0}}">
    暂无数据
  </view>
```

```
38
39 </view>
40
41
```

```
1  // ambition.js
2  Page({
3    data: {
4      activeIndex:0,//默认显示“今日”
5      activeIndex2:2,//默认显示“专注”
6
7  },
8
9
10   changeType:function(e){
11     var index = e.currentTarget.dataset.index;
12     console.log("currentbuttonindex="+index);
13
14     this.setData({
15       activeIndex:index
16     })
17
18     if(index == 0){
19       if(this.data.activeIndex2==2){
20         this.setData({
21           list_show:this.data.dayList
22         });
23       }else{
24         this.setData({
25           list_show:this.data.dayList_ambition
26         });
27       }
28
29     }else if(index == 1){
30       if(this.data.activeIndex2==2){
31         this.setData({
32           list_show:this.data.list
33         });
34       }else{
35         this.setData({
36           list_show:this.data.list_ambition
37         });
38       }
```

```
    }

  },

  changeType2:function(e){
    var index = e.currentTarget.dataset.index;
    console.log("currentbuttonindex2="+index);

    this.setData({
      activeIndex2:index
    })

    if(index == 2){
      if(this.data.activeIndex==0){
        this.setData({
          list_show:this.data.dayList
        });
      }else{
        this.setData({
          list_show:this.data.list
        });
      }

    }else if(index == 3){
      if(this.data.activeIndex==1){
        this.setData({
          list_show:this.data.dayList_ambition
        });
      }else{
        this.setData({
          list_show:this.data.list_ambition
        });
      }
    }

  },

})

```