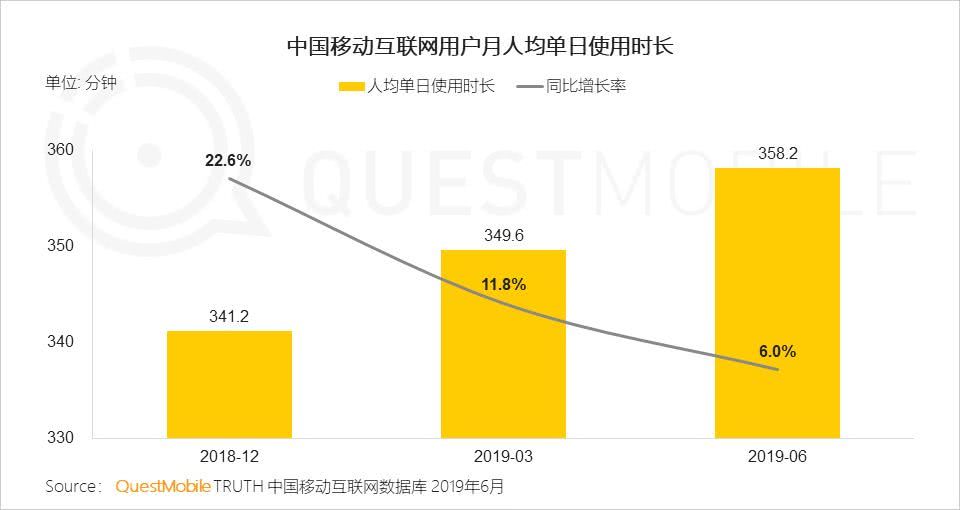
**“放下你的手机”产品方案设计**

1. **项目实施可行性报告**
2. 行业市场分析

通过调查分析，我们发现随着智能手机的大范围普及、网络的快速发展，以及各式各样、功能齐全的软件的日渐丰富，智能手机在现代人的日常生活中发挥的作用越来越大，人们对手机的依赖性也越来越高，这也导致很多人沉迷手机无法自拔，白白浪费了很多的时间和精力。《中国移动互联网2019半年大报告》的数据显示，2019年夏季中国移动互联网月活动用户人均单日使用时长为358.2分钟，同比增长6.0%。也就是中国人均一天有1/4的时间消耗在手机上，中国用户的手机依赖性可见一斑。



同时，通过各界媒体的调查，有很多的网民认为自己浪费在手机上的时间过多，对手机的依赖性强，希望能减少对手机的依赖，但自己又很难控制自己使用手机的时间，非常矛盾。

因此我们小组推出一款名为“放下你的手机”APP，它分析用户使用手机时间的历史记录数据，自动在新的一天为用户设置这一天使用手机的合理时间，逐步帮助帮助用户逐步减弱对手机依赖性。

1. 机遇

目前我国智能手机用户规模高达11.38亿，高速发展给我们生活带来了许多便利，但同时也带来了问题——许多人非常依赖手机，沉迷手机游戏无法自拔；同时他们也对自己的行为有一定的愧疚感，意识到自己的自控力低下，我们APP潜在的市场规模非常大，若有一款能够为他们减少对手机依赖的软件，必然能够占有一定的市场份额。

1. 威胁

为用户提供这类服务的APP并非少数，想要在这里面脱颖而出，则不能复制这些软件的路线，而需要另辟蹊径。因为市场调查手段有限，资源也有限，我们没有办法覆盖到大的用户群，因此也没有办法了解到更多市场用户的想法，难以挖掘潜在用户。

1. 优势

我们优势在于可以根据用户手机情况为其提供一个减少对手机依赖的方案，既可以自定义方案，也可以对我们提供的方案进行修改，定制一个更适合自己的方案。即我们可以给用户充分的自由空间，同时也给用户提供个性化的建议，从中找到一个微妙的平衡。

1. 弱势

缺乏资源优势，初期难有足够的用户数据可以验证算法是否有效，因此还需要一定的数据量与时间对算法进行完善修改。同时也缺乏资金优势，不具备与第一梯队企业抗衡的实力。

1. 竞争对手或同类产品分析

通过市场调查，我们发现存在一款与我们的产品相近的软件，名为“不做手机控”。这款软件主要功能有“番茄工作”、“监督玩机”、“睡眠计划”、“使用统计”、“白噪声”。

“番茄计划”是由用户开启一个任务，设置一段时间，任务开始之后，软件开启强制屏保，用户在该段时间内不能解锁使用手机。“监督玩机”是监督用户在设定的时间段内手机的使用情况，以及在监督用户的某款APP的使用时间。“睡眠计划”是用户设置睡眠时间，软件在睡眠时间内强制屏保，不可使用。“使用统计”是用户手机使用的日表、周表等。“白噪音”是播放白噪音音乐。

这款软件为手机依赖性强的用户提供了一个减轻手机依赖的平台，用户可自行开启“番茄工作”任务，设置不使用手机的时间段，强制自己在接下来的时间内不使用手机。同时也可以为用户统计手机是使用时间，让用户对自己的手机使用情况有一个全面的了解。

但是这款软件也有不足之处，在这款软件的使用过程中，“番茄计划”的限制时间的设置、任务的开启都是由用户自行决定，用户在整个过程中既是“运动员”，也是“裁判员”，自己约束自己使用手机的时间。这款软件在使用之初，对于那些希望减少自身对手机依赖的用户来说，确实可以起到一时的帮助作用，但是想要坚持约束自己使用手机，还需要强大的自制能力。而对手机有较大依赖性的用户，其本身就无法通过自身的自制力来约束自己使用手机的时间，否则他也将不需要一款软件来协助他减少手机的依赖，那又怎么能寄希望于他们可以依靠自身自制力和这款软件来约束自己、从而逐渐降低对手机的依赖程度呢？对于这款软件来说，这是软件在设计上的自相矛盾。

对于那些自制力不强，有又希望通过其他一款软件来协助降低对手机的依赖的用户来说，他势必需要一款“活”的软件，以外力的身份介入，自行为他安排手机的使用时间，这样才能更好的帮助用户减少对手机的依赖。

1. 自身条件分析
2. 团队中有两名软件技术方向成员、一名人工智能方向成员，能相互结合自身所学知识。
3. 团队成员有相对丰富的软件开发经验，熟悉软件开发流程，有能力完成该软件开发。
4. 团队成员都在学习安卓应用开发课程，并且对学习安卓开发都有相当高的热情。
5. 团队成员有大创项目合作经验，能做好项目开发进度安排与成员顺畅沟通。
6. 团队成员曾一起克服过开发中遇到的困难问题，具备解决问题的能力。
7. **产品定位及目标**
8. 产品定位

放下你的手机，通过掌握自己的手机使用情况，有针对性地减少对手机的依赖。

1. 产品核心目标

放下你的手机app目的是为了让用户逐步减少对手机的依赖程度。用户关心的是自己减少对手机依赖的效果，而我们app关心的是如何达到这个效果。因此，我们的核心目标就是——提高用户自控力，降低对手机的依赖程度。接下来的部分，我们将讨论如何达到这么一个目标。

1. 自控力是什么

自控力，主要体现在“控”字上，它区别于“自制力”两者却又密不可分。自控，可以理解为自我控制，对外界诱惑及自身行为习惯的一种控制。缺乏自控力，就不会形成良好的习惯行为，严重点会因为抵挡不住外界诱惑，走上犯罪道路。自我的控制，通过调节自我的心态，增强自我的意志力来提高。

1. 面对手机为何会出现自控力不足

拖延以及分心是削弱自控力的罪魁祸首，而在现代社会中，手机已然成为分心的首要原因。当有了拿起手机的冲动，人的自控能力此时已经下降到了一个较低水平。但问题是，为什么面对手机，人的自控力就瞬间不起作用了？

这是一个习惯问题，我们习惯了使用手机，手机管理着我们的生活大小事——小到买早餐付款，大到出行导航规划，我们难以离开手机，因为它是如此的便利，提高了我们的生活质量。但是这无形中提高了我们对手机的依赖程度，这些生活大小事并不会集中分布在我们的一天中，而是会均匀的穿插在我们的日程，导致我们没办法放下手机，因为你随时要用。可是，在大部分时间里，我们并不是用手机处理这些事情，而是用来刷朋友圈、刷微博、甚至拿着手机左右滑动不知道干什么，纯粹在浪费时间。一放下手机，不久后就又有了拿起手机的冲动了，使自己分心。

1. 如何抵消这种冲动

在产生分心拖延的冲动时，停下来，不要真的去做，哪怕只有几分钟，冲动就会自然消散，坚持下去，可以形成习惯，自控力就会不断增强。但是有一部分的人，没有办法停下来这几分钟来思考。

这种时候，就需要借助外力来帮助这一部分人。这种“外力”，可以是提醒，也可以是强制，只要能停下来，就都是一个办法。

我们要做的，就是提供这么一种“外力”。我们需要提醒用户，监督用户，但不能强迫用户。我们需要在强制与提醒之间，找到一个微妙的平衡——既能让用户抵消掉冲动，也不至于让用户反感。

1. 目标用户定位
2. 中小学生

年龄：6-18岁

收入水平：无

消费潜力：小

特点：自控力低，需要家长或教师监督。其中有一部分有自己的智能手机，对游戏的抵抗力较低，难依靠自己摆脱对手机的依赖。特别对于中高考学生，因为手机影响学习的情况较为严重。

1. 大学生

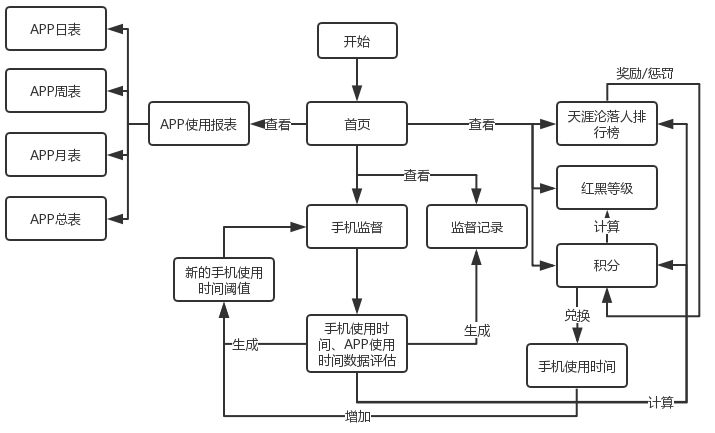
年龄：18-24岁

收入水平：低

消费潜力：中

特点：拥有一定的自控意识，但自控力仍处于薄弱阶段。在上课期间，玩手机成为一部分的大学生的常态，但他们有意识到自己对手机的依赖，也望摆脱对手机的依赖。特别在考试周期间或对于考研党来讲，需要借助一定的外力来约束自己。

1. **产品内容总策划**
2. 应用流程规划



1. 设计与测试规范
2. 设计规范
3. 命名规则
4. 包的命名规则

包是一组相关类和接口的集合，它将类和接口有机地组织成层次结构，使类和接口具有清晰的名称空间。包的命名规则如下：

1. 包名应该有意义，能反映包中的内容。
2. 包名应该是独有的、不可重复的。
3. 包名可以采用倒序的公司域名，本系统的所有包都以“com”开头。
4. 类和接口的命名规则

类和接口是 Java 的核心成员，必须要有一个中心目的，其命名规则如下。

1. 类与接口的名字应该表达其中心目的。
2. 类与接口的名字一般是大写字母开头。
3. 类与接口的名字可以由若干单词组成，单词的第一个字母采用大写字母，其余字母采用小写字母。
4. 一般不用动词来命名
5. 方法的命名规则

方法用来描述对象所具有的功能或操作，反映对象的行为，其命名规则如下。

1. 方法一般使用动词。
2. 方法名的第一个字母应该是小写。
3. 在多个单词混合的情况下，第一个单词后的所有单词的第一个字母大写，其余字母小写。
4. 变量的命名规则

变量包括成员变量、静态变量、类对象以及局部变量等。变量的命名规则如下：

1. 变量名应该易于记忆、理解，紧凑而有意义。
2. 变量名的第一个字母应该小写，避免使用 |“\_”或“ $”作为第一个字母。
3. 在多个单词混合的情况下， 第一个单词后的所有但系的第一个字母大写， 其余小写。
4. 常量的命名规则

常量一般采用大写字母单词命名，单词之间用下划线连接。

1. 源程序文件地注释

源程序文件注释一般在文件开头说明文件地功能，目的以及开发的相关信息，主要包括文件名、 功能、 目的、版本、 开发者、 开发时间、 最后修改时间、 修改人信息。

1. 块注释

块注释通常用于说明类、方法或程序段的意义及用途。块注释一般位于类、方法、或程序段的前面，应该与所描述的代码处于同一个缩进层次上。

1. 行注释

行注释可为单独行注释、 行尾注释与行头注释 3 种方式。 单独行注释是在单独的一行中书写注释信息， 一般用来描述较短的注释信息； 行尾注释位于代码后面， 与所要注释的代码行处于同一行； 行头注释位于代码的前面， 通常用来在程序调试中使少量代码失效。

1. 源文件编排格式

一个程序如果写的密密麻麻的，分不清层次，就会让人看不懂。好的程序应该充分的利用缩进、 空白、 续行等编排手段展现程序的可读性。

1. 缩进规则

缩进是指照程序语句的隶属关系向右做阶梯式移动。缩进单位一般为4个空格键，相当于半个tab键。有时缩进单位也采用一个tab键，即8个空格键，这种缩进称为深度缩进。缩进要注意逻辑层次， 同一层次的语句必须在相同的缩进位置上。

1. 空格的应用原则

空格的合理应用可以增强程序源代码的清晰性和可读性，其应用规则如下。

1. 关键字与括号之间要加空格。
2. 参数列表中的逗号后面要插入空格。
3. 表达式的双目运算符与操作数之间应该添加空格。
4. For语句中的表达式之间应该用空格分开。
5. 空行的应用规则

在同一源文件的类或接口的定义之间可以插入两行空白行。

1. 在两个方法应该插入一行空白行。
2. 在方法内部的局部变量与第一条语句之间应该插入一行空白行。
3. 块注释与单行注释之前应该插入一行空白行。
4. 在方法内部的两个代码片段之间应该插入一行空白行。
5. 折行规则

一般来说java代码行的长度应该小于80个字符，如果一行代码超过80个字符，就应该折行，当完整的一行代码难以编排在同一个代码行中时，可以根据下面的规则对java源代码进行折行编排。

1. 续行应该采用深度缩进，即缩进8个空格。
2. 在逗号后折行，避免在逗号的前面折行。
3. 在运算符前面折行。
4. 将紧密关联的元素放在一起。
5. 可将复杂的语句续写到多行。
6. 变量与对象的使用规则

声明、使用变量或对象应遵循以下的规则。

1. 每行尽量只声明一个变量。如果变量名称较短且是同一个数据类型，也可在一行中声明多个变量。
2. 变量声明应尽量接近其首次使用的位置，而且应该尽量在变量声明的地方对其初始化。
3. 尽量避免在一个单行语句中多次赋值。应避免使用以下语句： i = j = 0 ;
4. 尽量在需要的时候创建对象，避免提前创建对象。
5. 避免在循环体内声明并创建对象。
6. 对一些不在使用的复杂对象（如数组、队列、树、图等），应尽早释放对它的引用，这样就可以加速垃圾回收工作，释放的方法是将对象引用赋空值。
7. 表达式规则
8. 括号规则

使用括号清晰的表达算数表达式和逻辑表达式的运算顺序，避免由于运算符的优先等级的问题而导致错误。

1. 单一意图规则

表达式应尽量表达单一意图，一个表达式最好只做一件事，避免使用一些复杂的表达式。

1. 语句编写规则

编写 java 语句是应遵循以下规则。

1. 每行仅写一条语句。如下列程序将两条写在同一行中，降低了可读性，不利于程序的调试。
2. 控制结构中的语句应使用一对大括号“ { ”、“ }”括起来形成复合语句。大括号应采用以下示例中的位置形式。
3. 在使用条件语句时， 应把最常见的情放在最前面处理， 保证代码在正常的情况下的执行路径非常清晰。
4. 循环应尽可能地缩短，通常把循环嵌套限制在 3 层以内， 如果循环过长， 应把循环中的部分内容移动另一个方法中。
5. 尽量避免在循环体中使用 tru-catch 块。
6. 测试规范

测试过程按4个步骤进行，即单元测试、组装测试、确认测试和系统测试。单元测试集中对用源代码实现的每一个程序单元进行测试，检查各个程序模块是否正确地实现了规定的功能。然后，进行集成测试，根据设计规定的软件体系结构，把已测试过的模块组装起来，在组装过程中，检查程序结构组装的正确性。确认测试则是要检查已实现的软件是否满足了需求规格说明中确定了的各种需求，以及软件配置是否完全、正确。最后是系统测试，把已经经过确认的软件纳入实际运行环境中，与其它系统成份组合在一起进行测试。严格地说，系统测试已超出了软件工程的范围。

1. 单元测试

单元测试的对象是软件设计的最小单位——模块。单元测试的依据是详细设计描述，单元测试应对模块内所有重要的控制路径设计测试用例，以便发现模块内部的错误。单元测试多采用白盒测试技术，系统内多个模块可以并行地进行测试。

1. 单元测试任务包括：
2. 1 模块接口测试；
3. 2 模块局部数据结构测试；
4. 3 模块边界条件测试；
5. 4 模块中所有独立执行通路测试；
6. 5 模块的各条错误处理通路测试。
7. 测试接口正确与否应该考虑下列因素：
8. 输入的实际参数与形式参数的个数是否相同；
9. 输入的实际参数与形式参数的属性是否匹配；
10. 输入的实际参数与形式参数的量纲是否一致；
11. 调用其它模块时所给实际参数的个数是否与被调模块的形参个数相同；
12. 调用其它模块时所给实际参数的属性是否与被调模块的形参属性匹配；
13. 调用其它模块时所给实际参数的量纲是否与被调模块的形参量纲一致；
14. 调用预定义函数时所用参数的个数、属性和次序是否正确；
15. 是否存在与当前入口点无关的参数引用；
16. 是否修改了只读型参数；
17. 对全程变量的定义各模块是否一致；
18. 是否把某些约束作为参数传递。
19. 如果模块内包括外部输入输出，还应该考虑下列因素：
20. 文件属性是否正确；
21. OPEN/CLOSE语句是否正确；
22. 格式说明与输入输出语句是否匹配；
23. 缓冲区大小与记录长度是否匹配；
24. 檔使用前是否已经打开；
25. 是否处理了档尾；
26. 是否处理了输入/输出错误；
27. 输出信息中是否有文字性错误；
28. 检查局部数据结构是为了保证临时存储在模块内的数据在程序执行过程中完整、正确。局部数据结构往往是错误的根源，应仔细设计测试用例，力求发现下面几类错误：
29. 不合适或不相容的类型说明；
30. 变数无初值；
31. 变数初始化或省缺值有错；
32. 不正确的变量名（拼错或不正确地截断）；
33. 出现上溢、下溢和地址异常。
34. 集成测试

在单元测试的基础上，需要将所有模块按照设计要求组装成为系统。这时需要考虑：

1. 在把各个模块连接起来的时侯，穿越模块接口的数据是否会丢失；
2. 一个模块的功能是否会对另一个模块的功能产生不利的影响；
3. 各个子功能组合起来，能否达到预期要求的父功能；
4. 全局数据结构是否有问题；
5. 单个模块的误差累积起来，是否会放大，从而达到不能接受的程度。
6. 单个模块的错误是否会导致数据库错误。
7. 确认测试

确认测试的任务是验证软件的有效性，即验证软件的功能和性能及其它特性是否与用户的要求一致。在软件需求规格说明书描述了全部用户可见的软件属性，其中有一节叫做有效性准则，它包含的信息就是软件确认测试的基础。

1. 系统测试

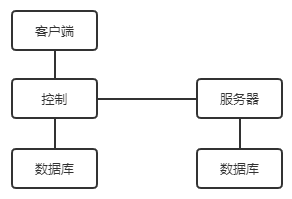
所谓系统测试，是将通过确认测试的软件，作为整个基于计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员等其它系统元素结合在一起，在实际运行（使用）环境下，对计算机系统进行一系列的组装测试和确认测试。

1. 开发日程表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 任务名称 | | | | | 工期 | 开始时间 | 结束时间 | 前置任务 | 备注 |
| 1 | APP开发 | | | | | 75天 | 2019.10.13 | 2019.12.12 |  |  |
| 2 |  | 准备阶段 | | | | 4天 | 2019.10.13 | 2019.10.16 |  |  |
| 3 |  | | 开发环境搭建 | | 1天 | 2019.10.13 | 2019.10.13 |  |  |
| 4 | 建立项目相关规范 | | 3天 | 2019.10.14 | 2019.10.16 |  |  |
| 5 |  | 界面规范 | 3天 | 2019.10.14 | 2019.10.16 |  |  |
| 6 | 编码规范 | 3天 | 2019.10.14 | 2019.10.16 |  |  |
| 7 | 数据库规范 | 3天 | 2019.10.14 | 2019.10.16 |  |  |
| 8 | 分析设计阶段 | | | | 21天 | 2019.10.17 | 2019.11.04 |  |  |
| 9 |  | | 需求分析 | | 7天 | 2019.10.17 | 2019.10.23 |  |  |
| 10 | 概要设计 | | 2天 | 2019.10.24 | 2019.10.25 |  |  |
| 11 |  | 建立ER模型 | 2天 | 2019.10.24 | 2019.10.25 |  |  |
| 12 | 功能模块划分 | 2天 | 2019.10.24 | 2019.10.25 |  |  |
| 13 | 详细设计 | | 12天 | 2019.10.26 | 2019.11.04 |  |  |
| 14 |  | 数据库建表 | 2天 | 2019.10.26 | 2019.10.27 | 11 |  |
| 15 | 系统ER模型的熟悉 | 2天 | 2019.10.26 | 2019.10.27 | 11 |  |
| 16 | 数据库的详细设计 | 4天 | 2019.10.28 | 2019.10.31 | 11 |  |
| 17 | 客户端、服务端详细设计 | 4天 | 2019.11.01 | 2019.11.04 | 7 |  |
| 18 | 编码阶段 | | | | 30天 | 2019.11.05 | 2019.12.04 |  |  |
| 19 |  | 客户端编码 | | | 30天 | 2019.11.05 | 2019.12.04 | 17 |  |
| 20 | 服务端编码 | | | 30天 | 2019.11.05 | 2019.12.04 | 17 |  |
| 21 | 单元测试 | | | 30天 | 2019.11.05 | 2019.12.04 | 19 |  |
| 22 | 测试阶段 | | | | 10天 | 2019.12.05 | 2019.12.12 |  |  |
| 23 |  | 测试计划与方案制定 | | | 2天 | 2019.10.17 | 2019.10.18 |  |  |
| 24 | 集成测试 | | | 3天 | 2019.12.05 | 2019.12.07 | 23 |  |
| 25 | BUG修正与系统调整 | | | 5天 | 2019.12.08 | 2019.12.12 | 24 |  |

1. **技术解决方案**
2. 开发框架

开发的系统中所应用的技术基于Java和Android，UI设计简单，后端技术成熟稳定。采用C/S架构，可保证系统的高速运行；采用MVC的开发模式，系统的设计逻辑比较清楚。在具体设计上，本系统把通过客户端与用户交互，同时收集手机APP的使用数据，并把数据存储在本地，在监督的小周期结束之后，再在本地进行数据分析，得到评估结果，评估结果一方面保存在本地，供用户以后查看，另一方面上传服务器，用户天涯沦落人排行榜排榜。服务器排行榜数据有效期为一天，天涯沦落人排行榜有效期也为一天，每天更新数据，这样可大大减少服务器负担。



1. 客户端

UI界面采用Android设计，本款软件功能数量较少，大约5个页面界面即可满足全部的功能需要，即使为了美化界面增加设计，也并不会增加太大的难度，Android的界面设计非常成熟，可参考的样式代码非常多。

1. 控制

本系统的控制模块采用Java语言进行设计，Java成熟可靠，功能齐全，在系统控制模块一直倍受行业欢迎。控制一方面通过客户端与用户交互，主要用于获取手机APP使用数据，响应用户的操作，这些都有对应的方法和API，对系统设计要求不高。另一方面客户端还肩负数据分析的功能，团队成员都修习过数学建模课程，在数据分析上有较多的经验，对数据分析驾轻就熟，而系统采用的数据分析方法中经常使用到的矩阵计算，Java无论是Ejml还是Jama都可以提供高效便捷的矩阵计算。

1. 数据库

本系统无论是客户端还是服务器，数据库都采用mySQL数据，mySQL数据库开源免费、轻量化、软件成熟，比较适合本系统的使用。

1. 服务器

本系统的服务器将采用腾讯云，腾讯安全可靠，操作简单，大大降低了软件开发难度。

1. **推广方案**

本软件与时代热点结合，着眼于当代人的“手机依赖症”，有相应的舆论议题支撑，能得到很好的宣传效果和市场回应。我们注意到大学生对手机的依赖尤其严重，同时大学生有进行自律或自我管理的需求和欲望，我们的软件能切入他们的痛点。我们的定位是立足于华南师范大学，面向全国大学生群体，进而推广到其他人群。

|  |  |
| --- | --- |
| 阶段 | 营销方案 |
| 第一阶段 | 1. 在华南师范大学利用学生之间的社群进行推广。利用校内资源让软件获得第一批用户，收集用户反馈。 2. 在学校相关微信公众号发动“放下手机，立地学习”的相关倡议，作为推荐软件向学生推广。 3. 通过建立微信公众号宣传相关“戒断手机”的相关知识和自律相关的推文，辅助进行宣传。 |
| 第二阶段 | 1. 通过第一阶段在校内建立的用户群体，由个人介绍向校外大学生拓展用户数量。 2. 由在APP中进行“邀请得积分”等活动促进用户邀请朋友加入使用。 3. 通过举办“戒断手机，全家幸福”、“九月我不刷”等打卡相关活动，增加用户粘性和扩大影响力。 |
| 第三阶段 | 1. 通过第二阶段的积累，可以预估有其他有需求人群比如办公室上班族等加入。 2. 扩大在中小学生家长当中的推广力度，举办“当心！别让你的孩子碰这个！已有先例！”等相关活动，让中小学生家长主动为孩子安装。 3. 在青年群体中宣传使用APP戒断手机的成功案例，在知乎等平台进行大牛分享，吸引小资青年使用。 |

1. **运营规划**

本APP运营规划阶段与推广计划密切相关，为了做好风险规避与防范，将运营规划分阶段进行概述。

|  |  |
| --- | --- |
| 阶段 | 运营规划 |
| 第一阶段（开发） | 1. 进行市场调研，商讨研究项目可行性和创新性。 2. 分配开发任务，进行Android技术学习，完善知识结构。 3. 根据需求分析共同分配APP主体开发任务，并完成软件测试。 4. 建立版本管理。 5. 根据内测反馈对APP进行调整和修改。 |
| 第二阶段（上线） | 1. 安排服务器管理员负责后续服务器运维操作，安排软件工程师跟进APP状态，发生安全事件及时告警修复。 2. 开放用户反馈渠道，根据反馈记录对后续升级项目进行研讨。 3. 安排进行例行维护和更新。 4. 跟进社会和舆论风险，及时做出公关响应。 |
| 第三阶段（后续发展） | 1. 参考社会和自身发展需要添加新功能，比如群组讨论、交友、专注度监控等的功能。 2. 考虑开发微信小程序等附属工具。 3. 扩大团队规模，作为初创者成立公司。 |