**开发过程记录文档如下：**

**发现问题：**项目组想要为“坐不住”软件开发一个可以统计和管理软件使用时间的功能，可以记录使用时间、回溯桌面和分类管理。功能的目标用户是那些想要提高工作效率和保持健康生活的人，项目组的问题是如何设计和实现这样一个软件功能。

**多个解决方案：**根据需求，项目组提出多个可能的解决方案，比如：

解决方案一：使用一个简单而清晰的列表，显示用户每天使用的不同的应用和窗口的名称和时间，按照时间的降序排列，让用户能够一目了然地看到自己的软件使用情况，如下图所示：

应用/窗口 时间

微信 2 小时 15 分钟

Visual Studio Code 1 小时 45 分钟

Microsoft Edge 1 小时 30 分钟

Steam 45 分钟

Excel 30 分钟

…

解决方案二：使用一个动态而有趣的饼图，显示用户每天使用的不同的应用和窗口的名称和时间的百分比，按照时间的降序排列，让用户能够直观地看到自己的软件使用比例。

解决方案三：使用一个详细而丰富的时间线日历，显示用户每天使用的不同的应用和窗口的名称和时间的分布，按照时间的顺序排列，让用户能够精确地看到自己的软件使用历史。

解决方案四：使用一个简洁而实用的仪表盘，显示用户每天使用的不同的应用和窗口的名称和时间的总和，以及用户设定的目标和提醒，让用户能够方便地看到自己的软件使用总览。

**技术选择：**

使用.NET框架，利用其丰富的类库和工具，开发一个Windows平台的桌面应用程序，使用WinForms或WPF等技术构建用户界面，使用System.Diagnostics.Process等类获取软件使用时间和窗口信息，使用System.Drawing等类实现桌面截图和回溯功能，使用SQLite等数据库存储数据，使用ScottPlot等工具生成图表，使用TagLib等库实现分类管理功能。

使用HTML、CSS和JavaScript等技术构建用户界面，使用.NETSDK获取软件使用时间和窗口信息，使用electron-screenshot-desktop等模块实现桌面截图和回溯功能，使用IndexedDB等数据库存储数据，可能使用ECharts等工具生成图表，使用electron-tags等模块实现分类管理功能。

使用NanUI，利用其将.NET和HTML结合的特性，开发一个Windows平台的桌面应用程序，使用HTML、CSS和JavaScript等技术构建用户界面，使用.NET框架1的相关类库和工具获取软件使用时间和窗口信息，实现桌面截图和回溯功能，存储数据，生成图表，实现分类管理功能。

**筛选解决方案：**

时间线日历和列表都能够清晰地显示每天使用的不同的应用和窗口的名称和时间，让用户能够准确地了解用户的软件使用情况。

时间线日历和列表都能够方便地让查看用户的软件使用的历史和趋势，让用户能够分析用户的软件使用的变化和规律。

时间线日历和列表都能够简单地让软件使用进行分类和标签，让用户能够管理用户的软件使用的类别和优先级。

解决方案二：使用饼图来显示用户的软件使用比例，虽然可以让用户直观地看到自己的软件使用占比，但是也有以下的缺点：

饼图的空间利用率低，不能显示太多的应用和窗口，否则会导致饼图的切片过小，难以辨认和比较。

饼图的数据展示方式单一，只能显示用户的软件使用的百分比，不能显示用户的软件使用的具体时间，也不能显示用户的软件使用的历史和趋势。

饼图的交互方式有限，用户只能点击饼图的切片来查看更多的细节，不能对饼图进行缩放、拖动、筛选等操作，也不能对饼图进行分类和标签等管理。

解决方案四：使用仪表盘来显示用户的软件使用总览，虽然可以让用户方便地看到自己的软件使用的总和，以及用户设定的目标和提醒，但是也有以下的缺点：

仪表盘的空间利用率高，但是也会导致仪表盘的信息密度过高，难以突出重点和关注点，也会增加用户的认知负担和视觉疲劳。

仪表盘的数据展示方式复杂，需要用户在不同的图表和指标之间进行切换和对比，不能让用户一目了然地看到自己的软件使用情况，也不能让用户精确地看到自己的软件使用历史和趋势。

仪表盘的交互方式多样，但是也会导致仪表盘的操作难度和复杂度增加，用户需要在不同的图表和指标之间进行选择和调整，不能对仪表盘进行简单和快捷的操作，也不能对仪表盘进行分类和标签等管理。

**确定解决方案：**最终考虑到以上因素，项目组选择：解决方案一（使用一个简单而清晰的列表）和解决方案三（使用一个详细而丰富的时间线日历，）

根据筛选结果，确定最终要实施的解决方案，以及项目组要达成的目标和指标。项目组制定了软件功能的需求：

细化需求：可以通过以下方式细化需求：

将解决方案的需求分解为以下的子需求：

时间线日历：显示用户每天使用的不同的应用和窗口的名称和时间的分布，按照时间的顺序排列，让用户能够精确地看到自己的软件使用历史。

列表：显示用户每天使用的不同的应用和窗口的名称和时间，按照时间的降序排列，让用户能够一目了然地看到自己的软件使用情况。

分类和标签：让用户能够对自己使用的应用和窗口进行分类和标签，管理自己的软件使用的类别和优先级。

设置：让用户能够设置自己的软件使用的目标和提醒，调整自己的软件使用的习惯和健康。

定义每个子需求的目标、范围、优先级、验收标准等，例如：

时间线日历的目标：让用户能够清晰地看到自己每天使用的不同的应用和窗口的名称和时间的分布，回溯自己的软件使用的历史和趋势。

时间线日历的范围：包括用户每天使用的所有的应用和窗口，以及用户可以选择的不同的时间段，如今天、昨天、本周、本月等。

时间线日历的优先级：高，因为这是用户了解自己的软件使用情况的主要方式。

时间线日历的验收标准：用户能够在时间线日历上看到自己每天使用的不同的应用和窗口的名称和时间的分布，能够切换不同的时间段，能够点击每个应用或窗口查看更多的细节，能够在时间线日历上看到自己的软件使用的目标和提醒。

设计规范：可以通过以下方式设计规范：

将解决方案的设计细节，如颜色、字体、图标、动效等，统一为一套规范和标准，例如：

颜色：使用淡蓝色作为主色调，表示清新和健康，使用深蓝色作为辅助色调，表示稳定和专业，使用红色作为提醒色调，表示警示和重要。

字体：使用微软雅黑作为主要的字体，表示简洁和易读，使用楷体作为次要的字体，表示优雅和美观。

图标：使用坐不住的LOGO图标，表示不同的功能和类别，如日历、列表、设置、游戏、工作、娱乐等。

动效：使用平滑而有趣的动效，表示不同的交互和反馈，如切换、点击、拖动、弹出等。

技术架构：可以通过以下方式构建架构：

将解决方案的技术组件，如数据库、服务器、框架、接口等，组织为一个合理和高效的架构和模式，例如：

数据库：使用 SQLite 作为本地数据库，存储用户的软件使用的数据，如应用和窗口的名称和时间，分类和标签，目标和提醒等。

服务器：使用自购的服务器，提供用户的软件使用的数据的备份和同步，以及用户的软件使用的数据的分析和报告等服务。

框架：使用 .NET作为跨平台桌面应用的框架，创建软件使用时间统计的功能，如时间线日历、列表、分类和标签、设置等。

接口：使用本地接口，获取用户的前台应用和窗口的名称和时间。

**开发文档如下：**

**# 坐不住软件时间统计功能需求文档**

## 1. 引言

### 1.1 目的

本文档的目的是描述.NET桌面健康管理工具的软件的功能需求，为软件的设计、开发、测试和维护提供依据。

### 1.2 定义、缩略语和缩写

- .NET框架：一个由微软开发的跨平台的应用程序开发框架，提供了丰富的类库和工具，支持多种编程语言。

- WinForms：一个基于.NET框架的图形用户界面技术，使用Windows窗体控件构建用户界面。

- WPF：一个基于.NET框架的图形用户界面技术，使用XAML语言和.NET对象构建用户界面。

- SQLite：一个轻量级的嵌入式关系型数据库，支持SQL语言和ACID事务。

- ScottPlot：一个基于.NET框架的图表生成工具，支持多种类型的图表和交互功能。

- TagLib：一个基于.NET框架的标签管理库，支持多种格式的文件的标签读写。

- 系统：指.NET桌面健康管理工具的软件系统。

- 用户：指使用系统的人员。

- 应用：指用户在计算机上运行的软件程序。

- 窗口：指应用在桌面上显示的图形界面。

- 模块：指系统的功能组成部分。

- 标签：指用户为应用或窗口添加的分类标识。

## 2. 总体描述

### 2.1 产品描述

系统是一个基于.NET框架的桌面应用程序，可以统计和管理用户的软件使用时间，具有记录使用时间、回溯桌面和分类管理等功能，旨在帮助用户提高工作效率和生活质量。

### 2.2 产品功能

系统的主要功能如下：

- 记录使用时间：系统可以记录用户每个应用和窗口的使用时间，精确到秒，并以图表的形式展示用户的软件使用情况，包括总时间、平均时间、最长时间、最短时间等指标。

- 回溯桌面：系统可以定期对用户的桌面进行截图，保存用户的桌面图像，并提供一个回溯功能，让用户可以查看过往的桌面图像，回顾自己的工作和生活状态。

- 分类管理：系统可以让用户为应用或窗口添加标签，按照标签进行分类管理，统计不同类别的软件使用时间，帮助用户分析自己的软件使用习惯和偏好。

### 2.3 用户特征

系统的目标用户是那些想要提高工作效率和生活质量的人，他们需要一个可以统计和管理自己的软件使用时间的工具，帮助他们了解自己的软件使用情况，优化自己的软件使用习惯，提高自己的工作和生活水平。

系统的用户应具备以下特征：

有一定的计算机操作基础，能够熟练地使用Windows系统和常用的桌面应用程序，如浏览器、文档编辑器、邮件客户端等。

- 有一定的数据分析和图表理解能力，能够通过系统提供的图表和指标，了解自己的软件使用情况，发现自己的软件使用问题和改进点。

- 有一定的自我管理和自我调节能力，能够根据系统提供的分类管理功能，为自己的软件使用设置合理的目标和计划，调整自己的软件使用习惯，提高自己的工作效率和生活质量。

### 2.4 约束和假设

系统的开发和运行受到以下的约束和假设：

- 系统的开发基于.NET框架，使用C#语言编写代码，使用Visual Studio作为开发工具，使用Git和GitHub作为版本控制和协作工具。

- 系统的运行需要Windows 10或以上的操作系统，需要.NET Framework 4.8或以上的运行时环境，需要至少4GB的内存和500MB的磁盘空间。

- 系统的用户需要拥有合法的Windows系统和桌面应用程序的许可证。

- 系统的用户需要保证自己的计算机安全和网络稳定，需要定期备份自己的数据和设置，不得对系统进行修改或破坏，不得将系统的数据和功能泄露或转让给他人。

## 3. 具体需求

### 3.1 功能需求

系统的功能需求如下：

#### 3.1.1 记录使用时间

- 功能描述：系统可以记录用户每个应用和窗口的使用时间，精确到秒，并以图表的形式展示用户的软件使用情况，包括总时间、平均时间、最长时间、最短时间等指标。

- 输入：用户的软件使用行为，如启动、关闭、切换、最小化、最大化等应用或窗口的操作。

- 输出：用户的软件使用时间的数据和图表，如饼图、柱状图、折线图等。

- 异常处理：如果系统无法获取用户的软件使用时间或窗口信息，或者用户的软件使用时间或窗口信息不完整或不准确，系统应该提示用户，并尽可能地修复或补全数据。

- 优先级：高

#### 3.1.2 回溯桌面

- 功能描述：系统可以定期对用户的桌面进行截图，保存用户的桌面图像，并提供一个回溯功能，让用户可以查看过往的桌面图像，回顾自己的工作和生活状态。

- 输入：用户的桌面截图的设置，如截图的频率、保存的位置、保留的时间等。

- 输出：用户的桌面图像的文件和回溯界面，如缩略图、幻灯片、日历等。

- 异常处理：如果系统无法对用户的桌面进行截图，或者用户的桌面图像的文件丢失或损坏，系统应该提示用户，并尽可能地恢复或重新生成文件。

- 优先级：中

#### 3.1.3 分类管理

- 功能描述：系统可以让用户为应用或窗口添加标签，按照标签进行分类管理，统计不同类别的软件使用时间，帮助用户分析自己的软件使用习惯和偏好。

- 输入：用户的标签的设置，如添加、删除、修改、搜索等标签的操作。

- 输出：用户的标签的数据和界面，如列表、图标、颜色等。

- 异常处理：如果系统无法为应用或窗口添加或删除标签，或者用户的标签的数据不完整或不准确，系统应该提示用户，并尽可能地修复或补全数据。

- 优先级：低

### 3.2 性能需求

系统的性能需求如下：

- 响应时间：系统应该在用户的操作后，尽快地给出响应，不超过3秒。

- 准确率：系统应该尽可能地保证用户的软件使用时间和窗口信息的准确性，不超过5%的误差。

- 可靠性：系统应该尽可能地保证用户的桌面图像和标签的完整性和安全性，不超过1%的丢失或损坏率。

- 可用性：系统应该尽可能地保证用户的软件使用情况的可视化和可操作性，不超过1%的故障或卡顿率。

### 3.3 界面需求

系统的界面需求如下：

- 系统的主界面应该包括以下的元素：

- 一个菜单栏，提供系统的基本功能，如设置、帮助、退出等。

- 一个工具栏，提供系统的主要功能，如记录使用时间、回溯桌面、分类管理等。

- 一个状态栏，显示系统的当前状态，如时间、内存、CPU等。

- 一个主区域，显示系统的核心内容，如图表、图像、列表等。

- 系统的主界面应该遵循以下的原则：

- 简洁：系统的界面应该尽量简洁，避免不必要的元素和信息，突出重点和功能。

- 一致：系统的界面应该保持一致，避免不同的风格和布局，提高用户的认知和操作的一致性。

- 直观：系统的界面应该直观，避免模糊和歧义的元素和信息，提高用户的理解和操作的直观性。

- 友好：系统的界面应该友好，避免冷漠和刺激的元素和信息，提高用户的满意和操作的友好性。

### 3.4 数据需求

系统的数据需求如下：

- 系统需要存储以下的数据：

- 用户的软件使用时间和窗口信息的数据，如应用名、窗口名、使用时长、使用次数等。

- 用户的桌面图像的文件，如截图的时间、位置、大小等。

- 用户的标签的数据，如标签名、标签颜色、标签应用等。

- 系统需要使用以下的数据格式：

- 用户的软件使用时间和窗口信息的数据，使用SQLite数据库存储，使用SQL语言查询和操作。

- 用户的桌面图像的文件，使用PNG格式存储，使用System.Drawing类库读写和处理。

- 用户的标签的数据，使用XML格式存储，使用System.Xml类库读写和处理。

## 4. 验收标准

系统的验收标准如下：

- 功能测试：系统应该能够实现所有的功能需求，没有明显的错误和缺陷，满足用户的操作和使用。

- 性能测试：系统应该能够达到所有的性能需求，没有明显的延迟和卡顿，满足用户的响应和体验。

- 界面测试：系统应该能够符合所有的界面需求，没有明显的错位和乱码，满足用户的视觉和感觉。

- 数据测试：系统应该能够保证所有的数据需求，没有明显的丢失和损坏，满足用户的存储和分析。

## 5. 附录

### 5.1 术语表

- .NET框架：一个由微软开发的跨平台的应用程序开发框架，提供了丰富的类库和工具，支持多种编程语言。

- WinForms：一个基于.NET框架的图形用户界面技术，使用Windows窗体控件构建用户界面。

- WPF：一个基于.NET框架的图形用户界面技术，使用XAML语言和.NET对象构建用户界面。

- SQLite：一个轻量级的嵌入式关系型数据库，支持SQL语言和ACID事务。

- ScottPlot：一个基于.NET框架的图表生成工具，支持多种类型的图表和交互功能。

- TagLib：一个基于.NET框架的标签管理库，支持多种格式的文件的标签读写。

- 系统：指.NET桌面健康管理工具的软件系统。

- 用户：指使用系统的人员。

- 应用：指用户在计算机上运行的软件程序。

- 窗口：指应用在桌面上显示的图形界面。

- 模块：指系统的功能组成部分。

- 标签：指用户为应用或窗口添加的分类标识。