ChronoSpect Design Document

Alex Rupp-Coppi, 4/22/15

Contents

Abstract

ChronoSpect is a time-usage-tracking app that operates by either passively monitoring how the user is spending their time or by actively reminding them of activities they should be doing.

Users define a time interval (e.g. 15 minutes) at the end of which to be queried by an alert from the app whether or not they are performing the goal activity they defined at the beginning of tracking.

When passively monitoring time usage, the app collects three pieces of information: the current time, the user's current activity, and on a scale of 1 to 5, their current engagement with that activity.

When actively monitoring, a fourth piece of information, the user's intended activity, is recorded.

This information is stored to a server-side database linked to a user's unique ID. The app will provide some data analysis (like scatter plots of time vs. productivity) to the user, but if they wish to statistically analyze their recorded data themselves, they have the option of downloading it in raw CSV format directly from the database. (In this way, the app is simple and intuitive for casual users but still potentially useful for hardcore statisticians.)

The time data recorded by the app can be combined with other self-reported datasets like amount of sleep, calorie intake, and GPS data to help notice more correlations between conditions of activity and variations in productivity.

The app will ship with a default set of activities and data categories, but will let users define their own custom activities to track very easily.

The success of the app depends on the seamlessness of its front-end UX experience and the robustness of its back end.

Because the app will interrupt the user fairly frequently (by default 4 times an hour) to collect time usage data, the duration of its interruptions needs to be minimized so as not to prove annoying or tedious to the user. Queries should take no more than 5 seconds, and will implement autocomplete and user history to accelerate the recording process.

On the backend, the app depends on having a well-structured, human-readable database supported by a robust server and user account system. Data analysis will be handled server-side; the app client itself will only show analysis results (like graphs).

The app will implement gamification to make it more engaging to users: productivity goals, competitions with other users, achievement badges, and level stats will keep users motivated to keep using the app over the long term. Tying into social media, users can share their progress statistics directly to platforms like Twitter.

Ideally this app will target iOS, Windows Phone, and web browsers. It will be developed using a combination of PhoneGap and MeteorJS.

Minimum Viable Product (MVP)

The app at its simplest contains two modes:

- Passive monitoring
- Active reminding

Passive mode asks the user at a fixed time interval (e.g. 15 minutes) what activity they are performing and how engaged they are with it (on a scale of 1 to 5). (Should users also be asked about their mood?)

Active mode asks the user at a fixed time interval if they are performing a pre-determined activity. If yes, it then queries user engagement. If no, it asks the user what other activity they are performing instead and reminds the user of the activity they should be doing.

At the end of a recording session the app will ask more general questions, like “Did you sleep last night?” or “Did you use a smartphone during this session?”

Data from these sessions will be stored in two separate tables, one for passive monitoring and the other for active reminding.

Passive Data

Time (t)	Activity	Engagement (1-5)	Mood? (1-5)
0:00	Work	5	4
0:15	Work	4	3

0:30	Work	2	1
0:45	Lunch	5	4

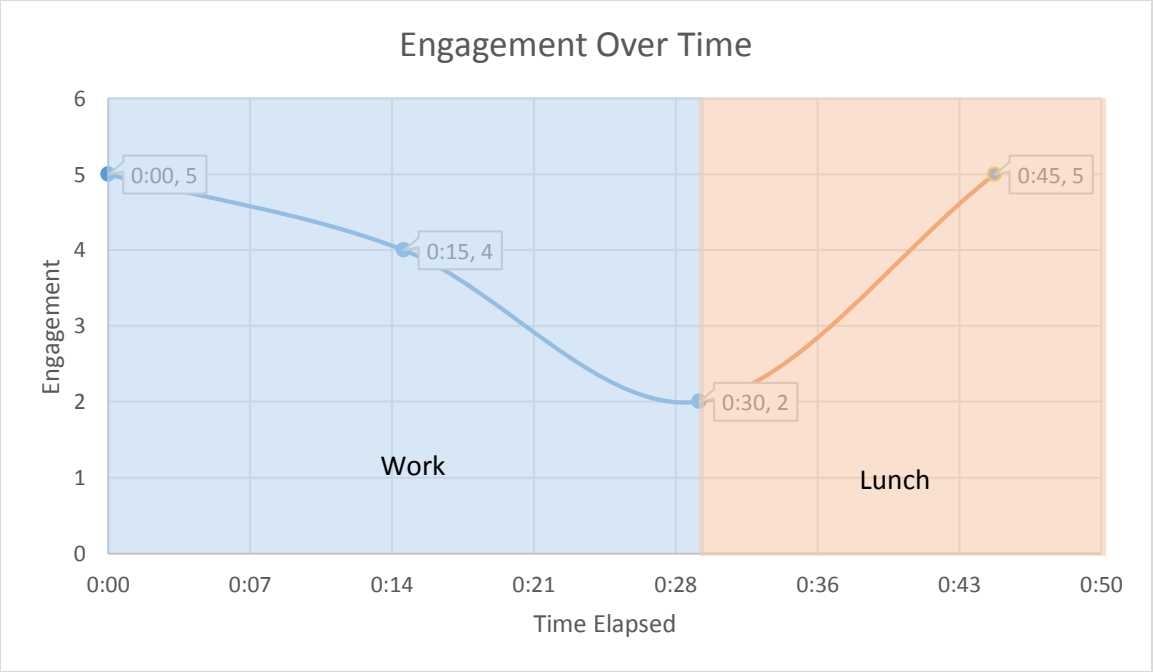
Active Data

Time (t)	Intended Activity	Actual Activity	Engagement (0-5) with Int. Activity	Mood? (1-5)
0:00	work	work	3	3
0:15	work	work	1	3
0:30	Work	YouTube	0	4
0:45	Work	YouTube	0	4

(When actual activity and intended activity differ, engagement with intended activity is 0.)

This data would be processed server-side and then be served to the user as a graph.

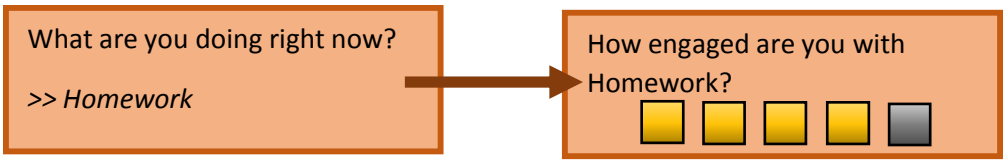
Graph for example passive data:



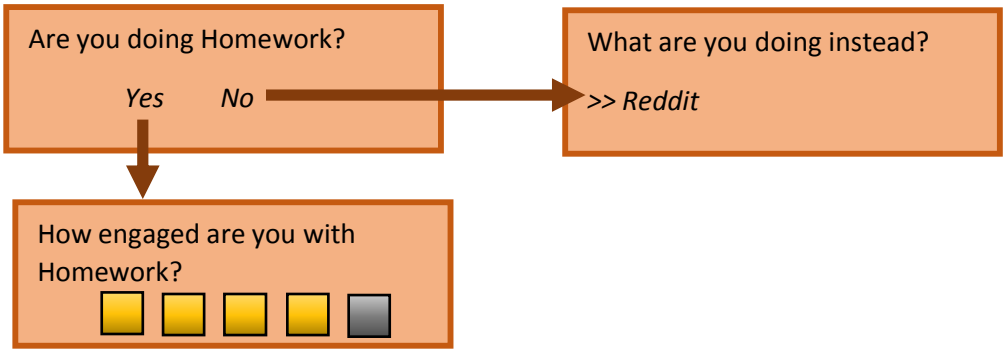
Users can access their account data and record new data sets from any platform as long as they sign in with their account.

App Flow

Passive:



Active:



Secondary Features

- Goals
  - Set goals for short term, long term
    - Session
    - Day
    - Week
    - Month
  - Productivity goals: hit certain percentage of time doing intended task for a given interval
  - Engagement goals: stay as engaged for a certain amount of time before losing focus
- Competitions
  - Try to get to goals before your friends do.
    - Who's being more productive?
    - Who's focusing better?
    - Who's sleeping more?
  - Badges, levels, XP, rewards
- Secondary data sets
  - Amount of sleep
  - Caloric intake

        - How much did you eat?
        - When did you eat it?
    - Walking distance
    - Time spent in a car
    - GPS data
- **Studies on aggregate user behavior** using anonymous user data
    - Look at lots of user's habits collectively and draw conclusions about them (publish all information CC-0, only use data with consent – "Do you want to participate anonymously in a productivity study?")
- Extended data analysis
    - Weekly productivity report – sent by an automated email?
    - Long-term trends
    - Most/least productive activities/times of day/conditions/correlation of productivity with sleep
- Alerts/push notifications, live tiles
- Gamify: have a pet - the more productive you are, the happier it is (taking an idea from Yale health app)
- Social media integration: Twitter, Facebook, Google+
- **User Analytics –** figure out how people are actually using the app, then tune it to feedback

Timetable

By **Wednesday, 4/29/15**, have a functional minimum viable product (just the two core modes, basic front-end UI, database) completed and running on a phone. (Do a lot of work over Long Weekend.)

Directed Study Self-Evaluation, 4/22/15

I am significantly behind in this project. By this point, according to my initial schedule, I was supposed to be on my second or third prototype. Because of the craziness of Junior Spring, I unfortunately just haven't had time to seriously develop anything. I've been reading through various tutorials for PhoneGap, and have actually followed a tutorial for Meteor. I've been reading documentation for both platforms in order to figure out how to use them, and have been looking through articles on best development practices and how to avoid getting rejected from the Apple app store. I've also been spending time setting up my development environment, configuring developer licenses, doing app design mockups on paper, and writing this design document. I'm going to spend a significant amount of time actually coding over Long Weekend in order to hopefully have a presentable, minimally-viable prototype up and running by next week.

You asked me in class to think of a grade I would give myself: based on my lack of progress and stunning record of missed deadlines so far, I think I merit somewhere around a C+. (I intend to turn that around.)