Princeton University
Department of Computer Science
Independent Work Spring 2018

Keep Asking: A System for Administering Recurring Surveys

Sebastian Hallum Clarke
Advisor: Prof. Jérémie Lumbroso

Abstract

Constructive and timely feedback is an important tool for any person or organisation seeking to improve. Recurring surveys are a useful mechanism for gathering feedback and identifying trends over time. Existing online survey platforms do not focus on recurring surveys; any support for temporality is treated as a niche ancillary feature. I built Keep Asking, a web application for designing, managing, and analysing recurring surveys. The system is flexible and empowers educational, enterprise, and individual casual users to gather feedback on a recurring basis. Keep Asking successfully enabled recurring collection of data by Professor Lumbroso from a cohort of his students.

Introduction

Receiving specific and timely external feedback is critical for any person seeking to improve themselves and his or her actions. While it is possible for a person to assess their own performance with moderate success, it is only through listening to feedback from other people that the greatest gains can be achieved. In a meta-analysis of 12 studies of the impact of external feedback in education, Hattie (2007) found that, on average, all types of external feedback had a positive impact on educational outcomes.

Surveys excel at facilitating the collection of feedback from many people about a single person or topic. Direct verbal communication can be a highly effective way to deliver feedback in a one-on-one situation, such as when a teacher is providing feedback to a student or a worker is providing feedback to a colleague. Direct verbal feedback does not scale well in cases where many people are providing feedback to a single person, such as a class of students providing feedback to a teacher on his or her teaching style. In this many-to-one situation, a person receiving unstructured oral or written feedback is likely to forget some of the feedback and have a biased perspective on the feedback he or she does remember. This highlights the advantages of surveys in collecting feedback in many-to-one relationships.

While conducting a single survey on a topic can gather an informative *snapshot* in time of the opinions of the surveyed population, gathering feedback on a recurring basis over time is essential for creating an effective positive feedback loop. Whereas the first round of feedback establishes the *baseline* and identifies areas for improvement, it is only through repeatedly asking about the same topics that a person can evaluate the impact of their efforts to improve.

Recurring surveys can be useful across a wide variety of applications. In education, instructors can survey their students on a regular basis across time to gather insight into the effectiveness of their pedagogical decisions. This is particularly relevant for instructors with fewer years of teaching experience. In a corporate setting, managers can survey employees periodically to understand employee sentiment and identify areas for improvement for individual managers and the workplace as a whole. In social groups, such as recreational sports teams or Princeton's eating clubs, group leaders can regularly survey their members about topics such as catering.

As discussed in the related work section of this paper, the systems currently in existence do not adequately allow individuals to administer recurring surveys in an easy way. For this project, I sought to design and develop an online system that makes it easy to create, distribute, and manage recurring surveys.

This project revolves around two core human-computer interaction (HCI) design problems relating to each of the classes of users interacting with the system: surveyors and respondents.

Firstly, the recurring survey system must delicately manage the relationship with survey respondents so that a sufficient number of high-quality responses can be collected without unduly annoying the survey respondents. This requires carefully considering the respondents' emotions across all of the points of contact with the recurring survey system.

Secondly, the recurring survey system must empower the surveyor to achieve their surveying goals without demanding significant effort. To succeed, this project requires designing a method for the surveyor create a recurring survey in constant time with respect to the number of times the respondents are surveyed. After responses have been collected, the system must display and facilitate the analysis of large numbers of survey responses across the duration of the recurring survey.

Properly integrating and addressing these HCI problems will be critical to the success or failure of any system focused on the administration of recurring surveys. Existing platforms with support for recurring surveys do not adequately address these difficult design challenges, and therefore are insufficient solutions to the problem.

Motivation and Problem Background

The impetus for this project was a discussion with Professor Lumbroso about how to gather actionable feedback from the many groups of people he interacts with, such as students in introductory computer science classes, his departmental academic advisees, project advisees, and grading meetings for which he arranges the catering. He used to manually orchestrate a system using Google Forms, but he felt that the time required to administer, aggregate, and analyse the results far outpaced the time that he had available for this work.

Based off this initial description, we investigated the field in several ways to get a better picture of whether these challenges were echoed elsewhere, how they are being addressed by existing tools, and whether it seems like a useful problem to improve that situation.

Qualitative Interviews of Princeton Faculty

The education sector is an area where recurring feedback has the potential to make a significant impact. As we are following the concept of a user-driven development, now a standard approach in modern HCI design, I separately interviewed three Princeton University professors to learn how each of them currently manages soliciting, receiving, and processing feedback from students in their courses.

The three instructors I interviewed were:

- Elizabeth Bogan[1], Senior Lecturer in economics, who has taught at Princeton since 1990. Professor Bogan teaches large (200-300 student) introductory courses, along with smaller (50-75 student) upper-level courses. She primarily lectures, but also runs several seminar-style precepts each semester.
- Laura Kalin[2], Assistant Professor of linguistics, who has taught at Princeton since 2016. She has twice been the primary instructor of the introductory linguistics course (175-200 students) and has taught several intermediate-level departmental courses (20-55 students).
- Dan Leyzberg[3], Lecturer in computer science, who has taught at Princeton since 2014. Professor Leyzberg was the lead instructor of the introductory computer science course (300-350 students) for the spring 2018 semester, having previously been a co-lead

[1] Elizabeth Bogan: https://www.princeton.edu/~ebogan/
[2] Laura Kalin: https://www.laurakalin.com
[3] Dan Leyzberg: http://www.danleyzberg.com

preceptor of the course. He also runs a computing mentorship community-based learning initiative course for ten students each semester.

All three professors emphasised the unsuitability of the university's student course evaluation system as a mechanism for gathering effective feedback on their pedagogical decisions. The quantitative information collected is too coarse to be useful; it is very difficult to identify specific improvements to make based off a score out of five representing the "quality of the lectures", for instance. On the other hand, the volume of qualitative comments left by students was too "overwhelming" to be effectively parsed.

Each of the professors had created their own system for collecting feedback from students half-way through the semester. These mid-semester feedback surveys are officially encouraged by the University and, from my anecdotal experience as a student, are a relatively common practice.[4] Professors Kalin and Leyzberg use surveys created using Google Forms, while Professor Bogan uses a paper survey and asks her secretary to digitise the results.

All of the professors expressed enthusiasm about the prospect of collecting feedback from their students more often than under the university's system. They cited a desire to improve their teaching methods and give their students "sovereignty" about their education. One professor, however, explained that they would be cautious about asking for feedback too frequently because the gains from education are "not meant to be immediate".[5]

The two professors using digital surveys were gathering yielding useful results, while the professor using paper survey reported learning relatively little from her survey. Based on my comparison of the three surveys, I think the unremarkable outcomes of the paper survey are the result of poor question selection rather than any inherent problem with the survey format itself. This hypothesis is supported by an empirical study which found negligible differences in results between paper and online medical surveys, all other factors being held equal (Basnov et al. 2009). Internet-based surveys, however, often have significantly lower response rates than surveys administered on paper (Kongsved et al. 2007), although this difference in response rate can be greatly reduced through the use of periodic emails reminding the target group to respond to the online survey (Skonnord et al. 2016). The professors using online surveys both were

[4] "Mid-Semester Course Evaluation Forms." *McGraw Center for Teaching & Learning*, Princeton University, mcgraw.princeton.edu/node/1536.

[5] Because there is likely to be a positive relationship between a respondent's perception of the likelihood that their feedback will have an impact and their response rate, it will be important to assist surveyors to appropriately manage the expectations of survey respondents about the impact of their feedback. Professor Kalin could, for example, publicly commit to her students to *read* all the feedback they submit and publish acknowledgements of suggestions when she feels it is appropriate. This explanation about how the feedback will be used could be included in the survey invitation and reminder emails.

pleased with their response rates, which they credited to the fact that students at Princeton are invested in their studies and have every incentive to suggest ways for their instructors to improve.

The professor who was receiving unremarkable responses from her paper survey did report receiving useful information through in-person conversations with her students. While casual, in-person conversations are effective for fostering an emotional connection and gathering anecdotal feedback, they are a poor substitute for a formal feedback system. There is likely to be multiple layers of selection bias impacting which students engage in these conversations, students have a propensity to give socially desirable responses (Halvorsrud and Kalfoss 2014), and the format does not scale well as the cohort size increases.

This primary research shows that there is a strong appetite in the education sector for better feedback mechanisms that enable educators to gather more useful feedback about the impact of their pedagogical choices. This strengthens the case for a recurring survey management system.

Existing Software Solutions

The existing systems for managing recurring surveys provide incomplete solutions. They either require too much manual intervention from the survey administrator or are prohibitively expensive for non-commercial users. While I initially explored building a recurring survey system on top of an existing survey platform due to the limitations of the existing systems' APIs, this was not possible. I elaborate on this challenge later in this paper.

Google Forms[6] is a popular and easy-to-use online form creation system offered by Google. The system is free and integrates with the Google Drive suite of productivity software. The system has no support for recurring surveys. To collect survey responses over time the survey administrator must either collect all responses in a single survey and ask respondents to manually indicate which "round" of the survey they are responding to, or make duplicates of the same master survey and manually track trends. Google Forms also lacks tools for defining a group of respondents (such as a class or team) to target with multiple surveys about different topics and has no capacity for sending reminder emails. Accessing a survey does not require authentication; anyone who knows a survey's address can respond to it. Google Forms has an attractive user interface and is free, so it very popular.

SurveyMonkey[7] is an online survey system that is primarily targeted at commercial customers. Although it has a free-tier service, the cheapest plan that includes recurring survey features costs

[6] Google Forms: https://www.google.com/forms/about/
[7] SurveyMonkey: https://www.surveymonkey.com

$384 per year. SurveyMonkey has minimal support for organising survey respondents into groups and sending user-triggered reminder emails. There is no capacity for sending reminder emails automatically after a certain period of time.

Survey Sparrow[8] is similar to SurveyMonkey, but the respondent-facing interface is primarily conversational and resembles a chatbot. The cheapest plan that includes recurring survey features costs $588 per year. While Survey Sparrow has limited support for organising survey respondents into groups, it does have the facility to configure automatic sending of reminder emails.

This analysis shows that there is a strong demand for web-based survey systems. The free survey system that is most popular among individual users, Google Forms, has no support for recurring surveys, cohort management, or reminder emails; essentially the entire survey administration process must be manually managed by the survey creator. While some survey systems, such as SurveyMonkey and Survey Sparrow, have modest support for recurring surveys, I have found no survey system with a primary focus on the management of recurring surveys. This research suggests that creating a system dedicated to the administration of recurring surveys would be a clear contribution to an area that is presently underdeveloped.

Opportunities for Improvement

From our interviews with Princeton faculty and external research about the existing solutions, it is clear that there exist ample opportunities to design a system that provides an improved approach to the problem of managing recurring surveys. The primary areas of improvement that I will seek to implement in this project are:

- Focusing primarily on administering recurring surveys, with the entire system treating recurring surveys as a first-class problem to be solved.
- Minimising the marginal effort necessary to survey a group of people multiple times; it should be as easy to survey people one time as it is to survey them every day.
- Simplifying the process of managing survey recipients, by avoiding the need for surveyors to manually send emails, copy-and-paste survey URLs, or worry about how to embed a form in an email.
- Automating the process of following-up with survey targets who have not yet submitted their survey response.

[8] Survey Sparrow: https://surveysparrow.com

- Improving the presentation of survey response data across time, so that it is easy to identify trends and draw actionable insights.

In light of the previously discussed challenges that online surveys experience with attaining sufficiently high response rates, the design of the system will need to take into consideration the factors that induce survey targets to respond. The World Bank, which administers hundreds of surveys each year, reports that the two most significant environmental factors that influence survey participation are social responsibility and social cohesion (Iarossi 2006). A survey will receive more responses when potential respondents feel that it is their duty to respond and that completing the survey is worth their time. Behavioural science tells us that people can be significantly influenced by learning about the actions of their peers (Sunstein 2014).[9] This project can benefit from these insights by helping surveyors to highlight the positive impact of each respondent's participation in the survey and invoking peer pressure to increase response rates.

Approach

I designed and built Keep Asking[10], a web app for creating, managing, and analysing recurring surveys. The system intentionally approaches recurring surveys in an abstract way that does not presuppose any particular use case; Keep Asking could be used in education, business, by social groups, or any other situation where a small number of people are seeking to gather information from a large number of people on a recurring basis.

Structural Architecture

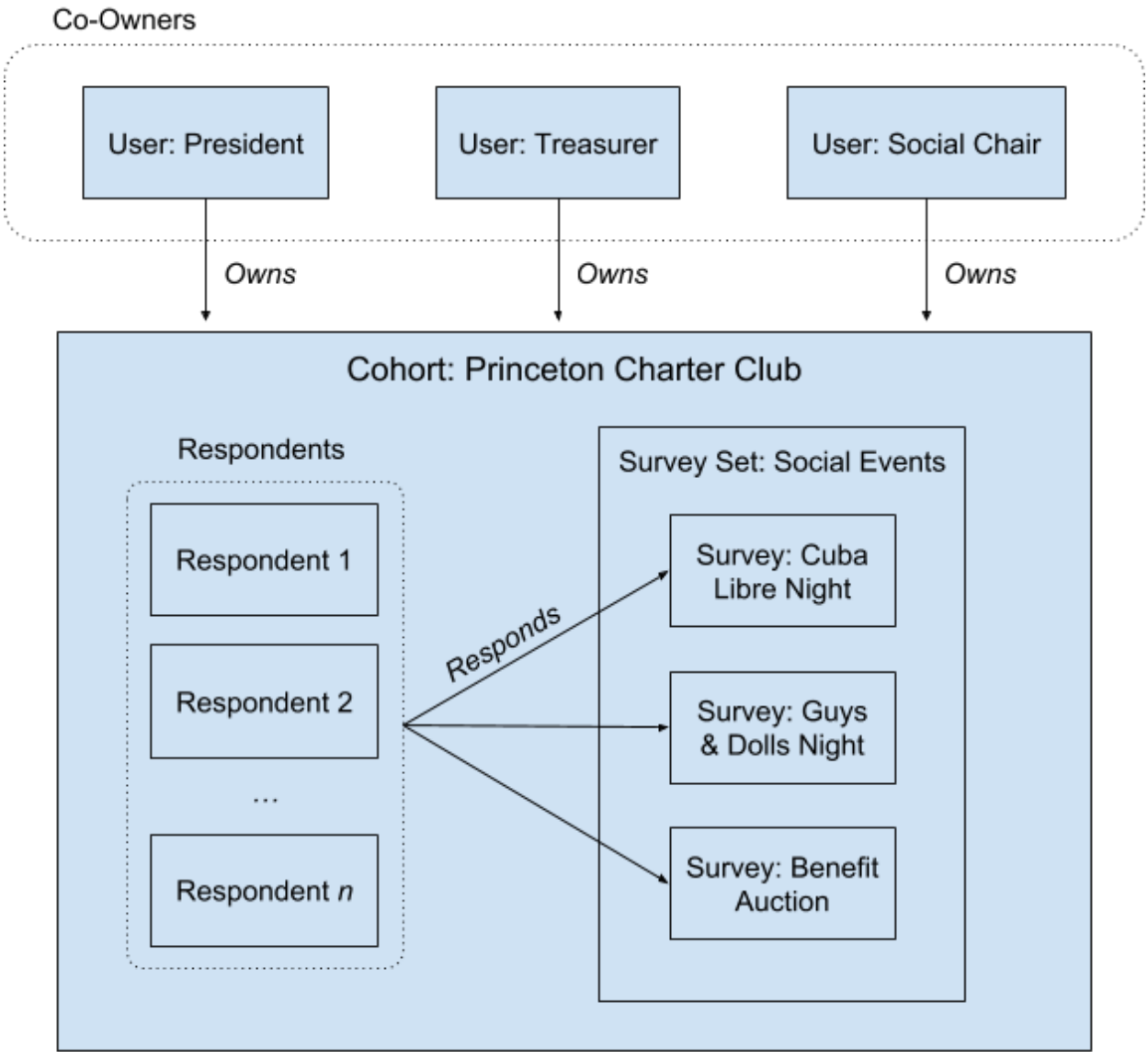
I thought carefully about the data model underpinning the system, knowing that this backend choice would have important consequences for the simplicity of the frontend. Objects within the Keep Asking system have a hierarchical relational structure. The top-level classes of objects are *cohorts* (a named group of respondents) and *users* (surveyors). Each cohort is owned by one or more co-owners who share administrative control over the cohort. The cohort contains *respondents*, who are uniquely identified by their email addresses. The cohort also contains one or more *survey sets*, which is a collection of *surveys*. All of the surveys in a survey set have the same questions, but each survey is sent on a different day and can be about a different topic.

Figure 1 illustrates the relationships between these classes in a fictitious example where Keep Asking is used by the Princeton Charter Club, a Princeton University social organisation where

[9] In an experiment run by the UK tax authority, HMRC, compliance with tax deadlines rose by 6.8 percentage points when historically late taxpayers were told that they were one of few delinquents in their hometowns. (Byrnes 2012)
[10] Keep Asking: https://www.keepasking.io

student members eat their meals. In this example, Charter's officers (the president, treasurer, and social chair) are the cohort owners and the club members are the respondents. The officers create different survey sets for each aspect of the club about which they wish to solicit feedback. In the diagram, the officers have created a survey set focusing on social events, with specific surveys (each containing the same questions) about each specific social event. The officers could easily create additional survey sets about other topics, such as club meals. This example serves to clarify how the different objects within the Keep Asking system relate to each other.

Figure 1. Keep Asking Structural Architecture



The decision to make a cohort the top-level object within Keep Asking's hierarchical structure because this reduces duplication of data within the system. It would be possible to construct the system with survey sets as the top-level object, however, this would require the surveyor to

provide the list of respondents every time a new survey set is created. Given that I foresee surveyors will relatively frequently wish to ask existing groups of respondents about new topics, I optimised the system for this use case. Treating cohorts as the top-level object does not make it significantly more complicated to ask a cohort about only one topic, so the optimisation is a reasonable trade-off.

Cohort and Survey Administration Process

The Keep Asking system is designed to mediate interactions between its two types of users: *surveyors* and *respondents*. The surveyor is responsible (possible jointly with other surveyors) for creating cohorts and surveys. The respondents are the individual members of a cohort who each respond to surveys designed by the surveyors. In the following subsection, I describe how these two types of users interact with Keep Asking and each-other using the system.

Account Management

The surveyor visits the Keep Asking website and creates an account by authenticating with a Princeton University or Google account. By using these third-party authentication mechanisms Keep Asking reduces both the amount of friction in the sign-up process and the burden of maintaining secure account management infrastructure (both Princeton and Google support two-factor authentication, for instance).

Cohort Creation

The surveyor creates a new cohort by clicking the "Create New Cohort" button and typing or pasting a list of email addresses representing the members of the cohort. The surveyor can add demographic questions to the cohort. These are questions which will be asked of survey respondents only the first time they submit a response to the survey. These questions are best suited for gathering demographic details about the respondents. For example, a professor might ask her respondents to identify their area of study or class year. These responses can then be used to segment survey results and assist with identifying differences between the responses of different constituencies within the broader survey population.

Through the cohort management page, the owner of a cohort can add other people as co-owners. The added co-owner will receive an email inviting them to create a Keep Asking account (if required) and join the cohort. All cohort co-owners have equal privileges, so can create new surveys, modify existing surveys, and view survey results.

Survey Set Creation

Within the cohort, the surveyor then creates a survey set by clicking the "Create New Survey" button. The surveyor is invited to name their survey set and use a calendar widget to choose the

dates on which the surveys should be sent. The surveyor can assign descriptive titles to each of these dates, which are helpful contextual reminders for survey respondents when completing the surveys and for the surveyor when reviewing each survey's results. The surveyor is invited to choose the number of days that each survey will accept responses after its scheduled survey send date.

Finally, the system prompts the user to create the questions for the survey set. Keep Asking currently supports four types of questions: free text (one- and multiple-line), choice (single- and multiple-selection), Likert-type scale[11], and rank[12]. I chose to focus on these question types because they give surveyors a large scope in the sort of questions that can be asked. Three of the question types are quantitative questions, which means that it remains easy for surveyors to interpret the results to these questions as the number of respondents grows. While surveyors may be familiar with more expressive systems, such as Google Forms, restricting the complexity of surveys encourages surveyors to focus on short, targeted surveys that are more suited for eliciting frequent responses.

Sending of Survey Invitations

According to the schedule defined by the surveyor, Keep Asking will automatically send emails to the cohort members inviting them to respond to the survey. These emails contain a secure link that uniquely identifies and authenticates the respondent without any user effort.

Based on the background research on maximising survey response rates, the text of this message tries to elicit feelings of social responsibility and social cohesion. See Appendix A for an example of a message sent by Keep Asking.

Half-way through the response acceptance period for each survey, an email is automatically sent to all cohort members who have not yet responded to the survey. Surveyors can also at any time click a button in their dashboard to manually trigger a wave of reminder emails. Sending a reminder email increases the likelihood of the cohort member responding to the survey because the email re-surfaces in their email inbox. Making the email reminders send automatically reduces the cognitive burden on survey administrators.

Responding to Surveys

Cohort members who click on the link in the survey invitation email are brought to a webpage where they respond to the survey. If the respondent has not previously responded to the cohort's

[11] A Likert-type scale question asks the respondent to choose a response between two extremes, such as "How much do you like fruit?" with the extremities labeled "Very little" and "Very much.

[12] A rank question asks the respondent to manually sort a list of items according to criteria defined in the question, such as "Sort these fruit in descending order of tastiness:" with the list "Apples", "Oranges", and "Pears".

demographic questions, they will be presented with these questions in addition to the survey questions. The form is straightforward and focuses on the questions being asked.

The webpage will display an error message if the survey is no longer accepting responses or the link used to access the survey no longer authenticates the respondent as a current member of the survey cohort. This ensures that survey responses are gathered only from the correct people during the appropriate survey response collection period.

Survey Response Analysis

At any time, the surveyor can view the survey results through the Keep Asking dashboard. The survey set results page shows the proportion of cohort members who have responded to each survey within the survey set.

The aggregated responses to each question are broken down for each survey (i.e. for each date) within the survey set. This allows the surveyor to easily view changes in the results over time. The results are displayed using a display representation that best suits each question type. The results from scale and choice questions are depicted using column charts. For rank questions, a table shows the average rank position chosen by the survey respondents, while text questions simply result in a list of all of the text responses.

The survey responses can be filtered to show only the responses submitted by respondents who answered a demographic question with a specific surveyor-selected answer. For example, if Keep Asking was being used in an academic setting, a professor could choose to view only the responses from students in certain class years or academic majors. This allows surveyors to understand the different views held by different constituencies within their cohort.

Event Tracking

Throughout the lifecycle of a Keep Asking survey, every interaction with survey respondents is immediately recorded in the event history log. A user can view the events that have occurred within their cohorts on the Events page of their dashboard. The events that are currently logged are emails being sent to cohort members, cohort members opening the survey response page, and cohort members submitting their responses. This event log gives survey administrators confidence that their survey request emails are being sent, respondents are opening their surveys, and their responses are being collected. The event log also provides useful insight into the effectiveness of reminder emails, as a surveyor can immediately track the actions that occur as a result of sending the survey reminder emails.

The event history log serves as an insurance policy for surveyors. It is likely that many surveyors care deeply about their relationships with their cohort members (whether these are students,

customers, or colleagues) and therefore may be nervous about delegating to an unfamiliar system the power to send scheduled emails on their behalf. The events page is designed specifically to assuage these concerns and make surveyors feel more in control about how the Keep Asking system is interacting with survey respondents in their name.

Implementation

Technologies

I implemented Keep Asking as a web application using the MEVN web development stack. Keep Asking uses a Node.js server-side app with the Express web framework. The application uses a MongoDB database and the Mongoose object document modelling system. This collection of open-source JavaScript-based systems integrate excellently with each other, scale well, have a wealth of plugins and are supported by strong development communities. For instance, authentication to the Keep Asking system is managed using the popular Passport Node.js module, with strategies for authenticating using Princeton University's Central Authentication Service, a Google account, or with a Keep Asking API key.

The website front-end uses standard, vanilla JavaScript, supported by the jQuery and Bootstrap front-end libraries. I refrained from using a more comprehensive front-end user interface framework, such as React or Angular, because of the steep learning curve to use these systems. I wanted to allow sufficient time to build Keep Asking's core features and felt that I could build these sufficiently well using traditional front-end web development techniques.

Choice of Survey Platform

Instead of building a custom survey creation and management platform, I initially sought to build Keep Asking on top of an existing popular survey system and focus my development efforts on the recurring aspect of the problem. This would have eliminated the need to develop a custom system for creating surveys, responding to surveys, and viewing the responses. I sought a free (and ideally open-source) survey system with a comprehensive public API that would allow me to programmatically create and modify existing surveys, and access the submitted survey responses.

However, I was unable to find an existing survey management system that was sufficiently extensible to allow me to achieve my goals for Keep Asking. Google Forms has no proper public RESTful API, and programmatic access is only possible through custom Google Apps Scripts, which would require significant user hassle and which are not designed for use by third-party

apps, such as Keep Asking.[13] While SurveyMonkey has a comprehensive public API, the company forbids the use of the API to "replicate functionality already offered by SurveyMonkey's survey tools".[14] As previously discussed, SurveyMonkey already has support for recurring surveys (albeit with many flaws), so Keep Asking was not able to use this. Survey Sparrow's public API is minimal and does not allow for creating or modifying surveys.[15]

Even if a suitable survey system with a sufficiently comprehensive public API existed, given the importance of the user experience to this process it seemed risky to leave the implementation of the survey experience to a third-party provider. This could be an interesting area of development in the future, though we must be mindful to avoid core aspects of the service to third-party providers in the absence of standards, lest their codebase evolve and cease being interoperable.

In light of there existing no system that supports the necessary API methods and permits the development of recurring survey applications, I set out to build a survey system from scratch.

Server-Side Architecture

I implemented the server-side app using the Model-View-Controller software design pattern. This allows for clear separation of data, business logic, and presentation within the application.

Models

I created models for the following classes of elements within the Keep Asking system: cohorts, events, respondents, responses, surveys, survey sets, and users. Each of these models has a defined schema which is used by Mongoose to enforce consistency within the database.

Controllers

The controllers manage database access, authentication, sending emails, and routing requests for both views and the public API.

Keep Asking sends emails using the Mailgun email delivery service via the Nodemailer email module. Keep Asking contains HTML and plain text email templates for sending survey request emails to cohort members and for inviting people to join a cohort co-owners. Contextual information is merged into these email templates using the EJS templating engine. Every ten

[13] An issue (https://issuetracker.google.com/issues/36754753) was created in Google's issue tracker in early 2009. This issue has received over 100 upvotes over the ensuing years, but is currently labeled as "Won't Fix". The API documentation for the Google Apps Script Form Service, which is not designed for use by external applications, is here: https://developers.google.com/apps-script/reference/forms/

[14] SurveyMonkey API Approval Checklist: https://developer.surveymonkey.com/build-a-public-app/

[15] Survey Sparrow API documentation: https://surveysparrow.com/developer/

minutes, a scheduled process runs to send any survey request emails that are due to be sent. Keep Asking has complete time zone support, and automatically adapts to the user's local time zone.

Views

Keep Asking dynamically renders views using the EJS templating engine based on data from the relevant models. Views have a hierarchical URL structure, such that the results of the survey set *surveyID* and cohort *cohortID* would be located at the resource address `/cohorts/cohortID/surveys/surveyID/results`. Because every cohort and survey within the Keep Asking ecosystem has a unique, canonical URI it is easy for co-owners of a cohort to reference specific cohorts, surveys, and results by sharing the URL of the page they are viewing. All pages except the external-facing brochure page and the survey response page require authentication to access.

Public API

Keep Asking exposes a comprehensive public RESTful API for interacting with the recurring survey system programmatically. This is the same API that the Keep Asking front-end website uses to interact with the server application. All requests must be authenticated using an API key, which can be retrieved and regenerated from a user's profile page on the Keep Asking website.

The API permits users to create new cohorts, surveys sets, and add co-owners to cohorts using HTTP POST request method. Existing cohorts and survey sets can be modified using the HTTP PATCH request method, while co-owners can be deleted using the HTTP DELETE request method. Using the HTTP GET request method, a user can programmatically retrieve a list of cohorts, a specific cohort, a list of survey sets within a cohort, and the responses to survey set.

This API not only allows for programmatic access to an existing user's existing cohorts but also allows for programmatic agents ("bots") to create cohorts and invite people to collaborate as a co-owner of the cohort. This design makes it possible to integrate Keep Asking into existing systems while limiting the overhead effort users typically associate with having to manually import data from their prior systems or central databases. For example, an academic institution could write a simple script that automatically creates cohorts of the students in every course and automatically update this cohort as students join and leave the course. The instructors of each course could be invited to collaborate as co-owners.

This implementation delicately balances the security of the platform with making it easy to integrate Keep Asking with a third-party system. In the example of the academic institution, above, only one system administrator needs to think about using the API. From the perspective

of the instructors, the system "just works" and they are never required to even touch an API key – all as a result of the co-ownership system.

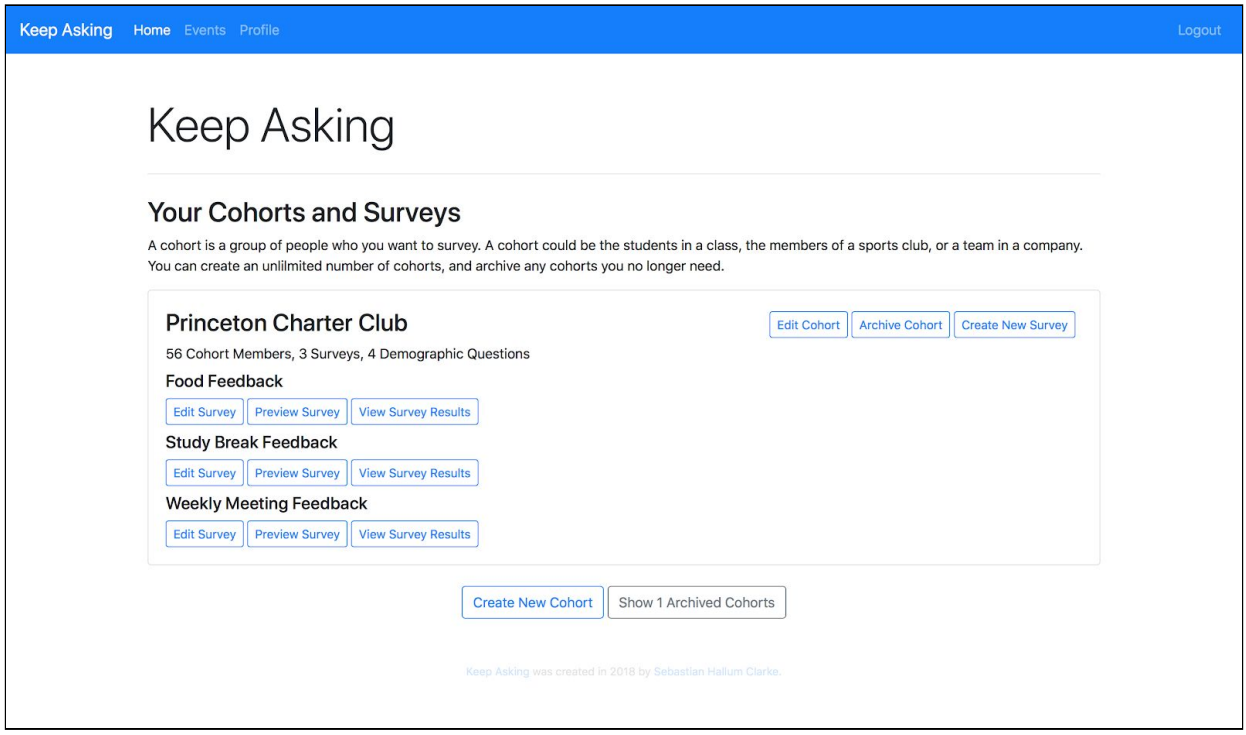
Front-End Architecture

Because of the relatively low level of interactivity in the interface, Keep Asking uses server-side rendering. This design choice, along with the use of the Bootstrap front-end component library, greatly simplifies the client-side development of Keep Asking. The system uses essentially no custom CSS for styling; all components and styling is performed using Bootstrap classes.

Keep Asking uses front-end JavaScript to power the modest interactivity in the web application, namely gathering the required information from surveyors and respondents for creating and responding to surveys. This front-end code is packaged using the Browserify module, which allows for modular development and assists with including front-end dependencies.

Many online survey systems have very complicated front-end administration interfaces, with important settings buried beneath layers of nested menus. Equally, the relationship between different objects within a system is often not obvious. Keep Asking attempts to avoid these pitfalls by using simple, hierarchical data models and faithfully representing these models in the interface. For example, as shown in figure 2, the "Food Feedback", "Study Break Feedback" and "Weekly Meeting Feedback" survey sets are visually contained within the "Princeton Charter Club" cohort, which reflects that survey sets are subsidiaries of cohorts. Similarly, the user interface for creating a survey set very closely maps to the data schema for the survey set model (see Appendix B for a comparison).

Figure 2. Keep Asking's Dashboard Page



Evaluation

The basic success criteria I defined for this project was that a user be able to administer a recurring survey using Keep Asking, accept responses, and analyse the results. My additional metric for measuring success was the average survey response rate.

Real-World Testing

Professor Jérémie Lumbroso[16] performed a real-world test of Keep Asking with students in Princeton's *Advanced Programming Techniques* (COS 333) course. The students in this course spend half of a semester in a team building a software product. Each group is assigned a faculty advisor, who meets with the students weekly, in Professor Lumbroso's case these meetings occurred on Fridays.

Professor Lumbroso used Keep Asking to survey his advisees in this course to track their self-assessed degree of preparation for the meeting, monitor project progress, and assess how confident students in each group are about their next steps. While Professor Lumbroso did not collect individual students' names, he used a demographic question that asked students to

[16] Department of Computer Science, Princeton University

identify their team name. This allowed Professor Lumbroso to filter the survey results by group. (Although the identity of respondents is known internally through their email address, this information is currently not displayed in the survey results interface.)

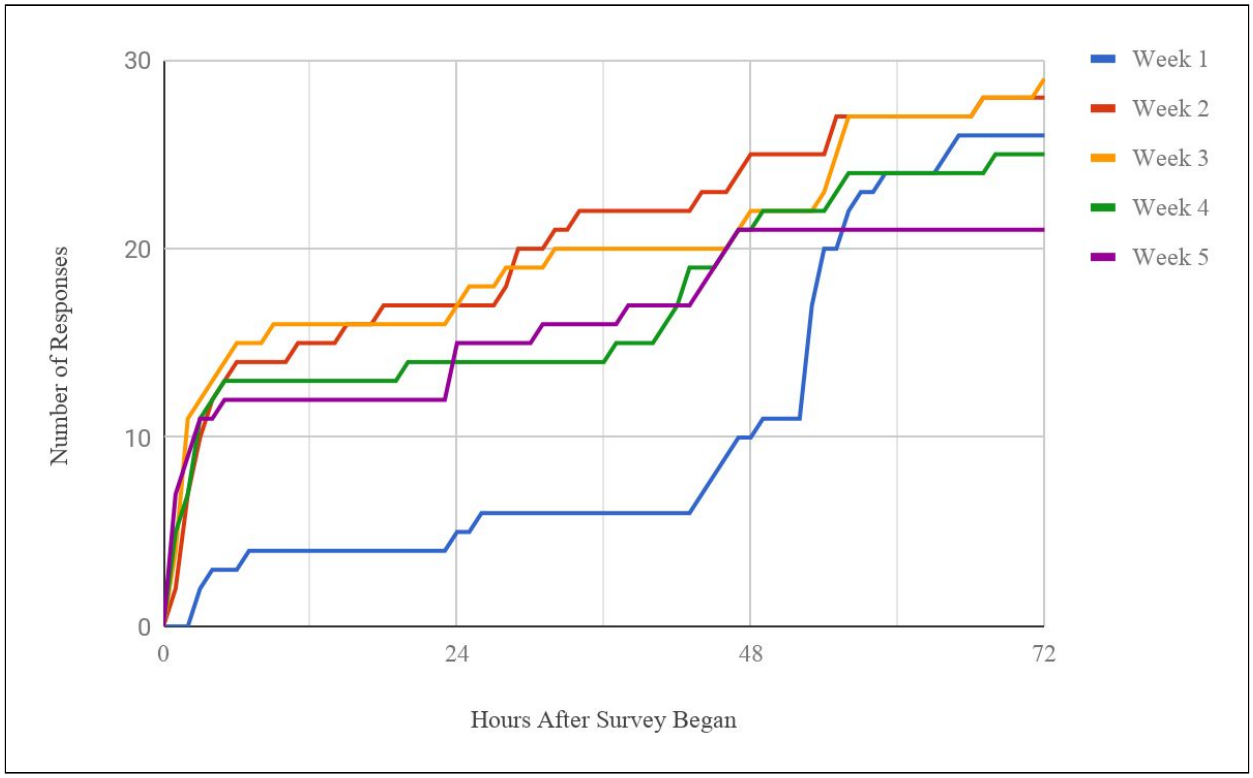
Professor Lumbroso scheduled his surveys to be sent out to his advisees at 16:00 every Friday, shortly after the weekly meetings, for five weeks. He told his 30 advisees in advance that they would soon receive a survey. In this survey, Professor Lumbroso asked four questions.

Two of these questions were single-line text questions. These turned out to be difficult to qualitatively or quantitatively exploit within the Keep Asking web interface because the system does not display respondent identities and there is not yet have any mechanism for aggregating textual responses. This indicates an area for future research on the interface design. Fortunately, for now, the data can conveniently be exported for further analysis.

The other two questions were respectively a boolean choice (Yes/No) and a scale question (confidence with next steps on a scale of 1 to 5) which proved easy to visualize in a generic way. In particular, the scale question provided a convenient way to evaluate the confidence levels of the entire cohort, and how they evolved throughout the week. This provided Professor Lumbroso with an important, desirable, and beneficial insight into his overall cohort's progress.

Now, considering the response times for this sample: Figure 3, below, illustrates the cumulative frequency of responses to Professor Lumbroso's survey. This graph can be used to understand how different types of students interacted with the Keep Asking system in different ways over the survey period.

Figure 3. Cumulative Frequency of Survey Responses by Week[17, 18]



This visualisation tells us that a significant proportion of Profesor Lumbroso's students consistently respond to the survey within a few hours of the delivery of the email. On average, 45% of students responded within six hours of receiving the email.

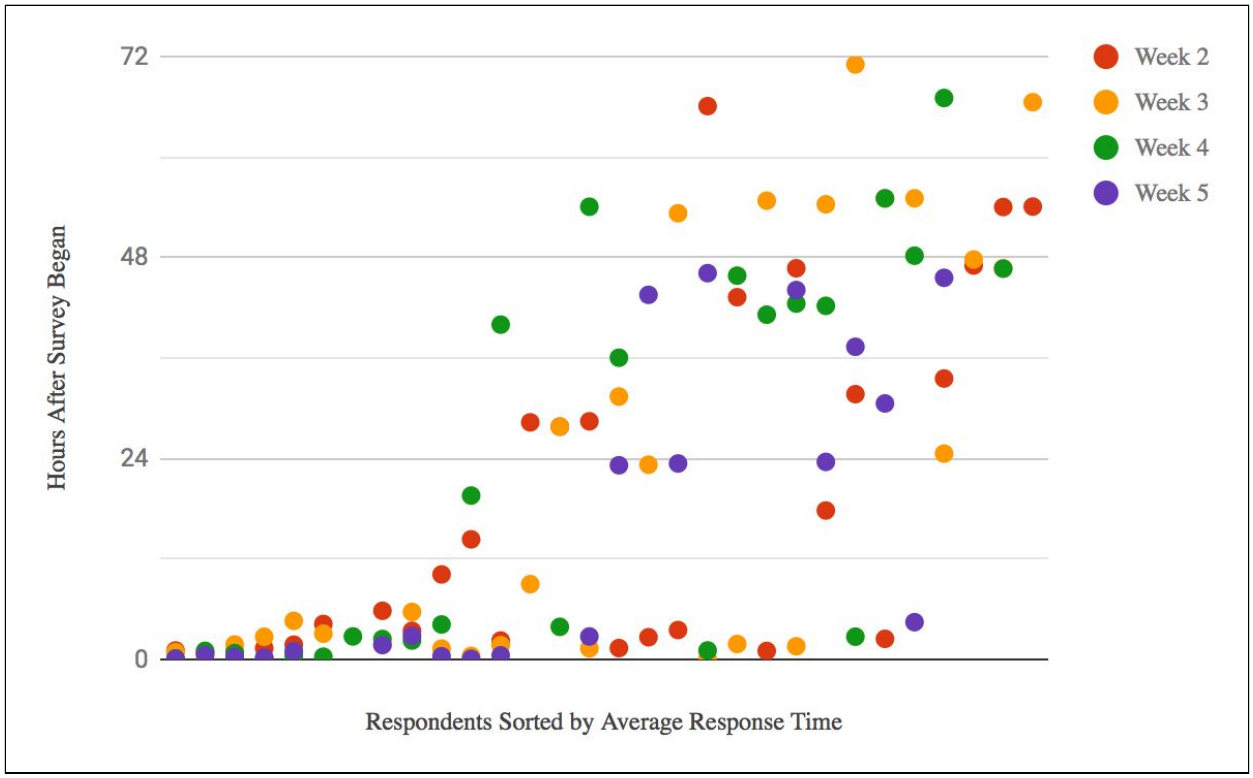
After the submissions of these eager students, the response rate slows significantly. Keep Asking's reminder emails help to incite procrastinating students to respond. While we did not conduct an A/B test to definitively assess whether sending a reminder email increases the likelihood of a cohort member responding, the data suggests there is an impact. The response rate in the 18 hours after the reminder email was sent (hours 36 to 60) was twice the response rate in the 18 hours preceding the distribution of the reminder emails. This affirms the positive impact of the reminder emails.

[17] Many of the survey invitation emails for the Week 1 survey were caught by respondent's spam filters; it was only after two days of improvements to the email infrastructure and message content that the redesigned emails were permitted to pass by the spam filters. This is reflected in the sharp increase in survey responses for Week 1 at around 50 hours.

[18] Week 5 results are likely reduced in the 48 to 72 hour period by house parties and Lawnparties, which are campus-wide social events which many Princeton students would have been attending during this time range.

Figure 4, below, illustrates the time at which each of the 30 cohort members responded to each of the surveys. Respondents are sorted by their average response time, which highlights the habitual differences among the students in the cohort. Analysing these habits is important because it helps me to understand how Keep Asking can better tailor email messages to the specific personality of each respondent.

Figure 4. Response Times By Respondent, By Survey[19]



The students to the far left of the graph (respondents one through nine) consistently respond very quickly to the survey invitation emails. Many students in the centre of the graph have responded in less than six hours to at least one of the surveys, but have more variation in their response times. The students at the far right of the graph have much greater response times.

Because people's memory fades quickly after an event, it is important to gather feedback as soon as possible. To reduce the survey response time and improve the quality of responses collected, Keep Asking could use online learning to dynamically adapt the frequency of automatic reminder emails based on each respondent's habits.

[19] Week 1 is omitted because many of the survey invitation emails were caught in spam filters.

Future Work

I built Keep Asking over the course of around nine weeks. Because of this compressed timeline, I prioritized the roadmap for this service to focus on reaching a state where the service is a sufficiently viable tool that implements the important interface details mentioned in the motivation for this project. At times this led me to focus on features with less visibility. For instance, I prioritised the development of public API because it was essential to enabling integration within existing systems and thereby was essential for minimising administrative overhead.

That said, we have clearly identified several important improvements that would significantly enhance Keep Asking's value proposition. In what follows, I will mention a non-exhaustive set of such ideas.

Cohort and Survey Management

Keep Asking currently requires that all new cohorts and surveys be manually created (or constructed programmatically through the API). There is no facility for duplicating an existing cohort or survey (regardless of the owner). It would be useful to allow surveyors to clone existing cohorts or surveys. For example, every semester a professor could clone their previous semester's course cohort (including the cohort's surveys), and she would only need to update the cohort members list and the change the dates on which the surveys should be sent to be ready for the next semester.

An additional powerful feature would be a crowdsourced gallery of high-quality survey templates that surveyors could import. This would make it easier for new users to configure their use of Keep Asking, provide inspiration about potential ways of using the system, and promote best practice in survey management.

The Keep Asking user experience could be further enhanced by providing some degree of integration with existing survey management systems. The onboarding process for new users could be made significantly easier if Keep Asking offered to automatically import a user's existing forms from another service, such as Google Forms. Through web scraping, this could be performed even in situations where the user's former survey system lacks a cooperative public API.

Improving Emails

Although the survey request emails were sufficient to receive a high response rate in the test with Professor Lumbroso, in a different situation with less social pressure to respond (such as when targeting a large group for feedback) it is likely that the response rate would be lower. The text of the survey invitation emails (Appendix A) contains the basic information about the cohort and survey topic, but the messages not customisable beyond this. I would like to allow surveyors to write their own email messages. This would empower surveyors to specifically explain how the respondent's participation in the survey is in each respondent's best interest, increase the sense of social pressure and play on the cohort members' desire for social cohesion.

It would also be useful to track when cohort members open the survey invitation email. This would be another important event to be logged in a user's Events page that would help surveyors to better understand how their cohort members are interacting with each survey. This tracking could be discreetly performed using a tracking pixel image embedded in the email.

Survey Participation Incentives

In situations where there is little obvious incentive for a cohort member to respond to a survey, such as when they do not feel strongly about a topic or there is little social pressure to participate, additional incentives may be required to gather sufficient survey responses. These incentives could be both internal and external to the Keep Asking system.

Internal incentives could include gamification techniques, such as unlocking "badges", leaderboards based on the speed of a cohort member's response, or similar. More directly related to the subject matter, survey administrators could be given the option to share their survey results with all cohort members who have already responded to the survey. This gives cohort members an incentive to submit their thoughts in order to see how their opinions compare with the rest of the cohort members. As previously discussed, Iarossi (2006) found that response rates are greatly impacted by whether respondents believe that they have a social responsibility to participate. Gamification would help people to quantify the scale of their contribution, and compare their participation with others (thereby playing on a desire for social cohesion).

Improving Survey User Interface

As previously mentioned, Keep Asking's front-end was almost entirely constructed using the Bootstrap component library. While this resulted in a shorter development time for this part of the application, it does mean that the interface is relatively simple. I would like to do more work on improving the interface of Keep Asking, particularly the page that cohort members use to

respond to surveys. Working to make this page more mobile-friendly, with larger tap targets, would be an important improvement.

I think there would also be value in experimenting with different ways of presenting the survey. I would like to test whether a system that shows only one question at a time (like Typeform[20]) or uses a conversational chatbot interface (like acebot.ai) might improve user engagement with the surveys and increase response rates.

Accessibility

Keep Asking has only been tested with users who have no computer interaction impairments. Making Keep Asking more accessible to users with impairments would broaden the range of people who can use the system, and improve the experience for all users. Examples of improvements that could make the system more accessible include making the system fully keyboard-navigable, supporting screen readers, and ensuring that all text is sufficiently legible (both in terms of size and colour contrast).

Conclusion

Keep Asking is a powerful system that allows people to administer recurring surveys with ease. Because we designed the system to focus primarily on this type of survey it is easier to administer and analyse the results than with the recurring survey features of prior online survey systems.

This clarity of purpose is reflected in Keep Asking's simple web interface and easy workflow: in just five minutes a surveyor can create a new cohort and survey set, complete with automatically scheduled survey invitations and reminder emails. For complete peace of mind, the surveyor can track every interaction his or her cohort members have with the system through the Events page. Once survey responses arrive a report is automatically generated, allowing the surveyor to immediately understand the opinions of the cohort at each surveyed point in time. The entire process is a constant-time operation with respect to the number of times the survey is run, which makes it easy for the surveyor to feedback as often as he or she desires.

This is in stark contrast to Google Forms, the most popular existing system, in which the surveyor would need to manually invite cohort members to respond to each round of the survey, manually follow-up with people who had not responded, and manually separate the results from different rounds of the survey within a spreadsheet. Only then could the surveyor (manually) graph the results and seek to gain insights from the responses.

[20] Typeform: https://www.typeform.com

What is more, Keep Asking is intentionally a *generalised* system for collecting feedback on a recurring basis. It is suitable for almost any situation in which a person (or small group) seeks to gather feedback from a cohort of respondents any size. The system is agnostic to its use case and can be equally powerful in enterprise, educational, or personal uses.

Bibliography

Basnov, Maja, et al. "Reliability of Short Form-36 in an Internet- and a Pen-and-Paper Version." *Informatics for Health and Social Care*, vol. 34, no. 1, 2009, pp. 53–58., doi:10.1080/17538150902779527.

Halvorsrud, Liv, and Mary Kalfoss. "Quality of Life Data in Older Adults: Self-Assessment vs Interview." *British Journal of Nursing*, vol. 23, no. 13, 2014, pp. 712–721., doi:10.12968/bjon.2014.23.13.712.

Hattie, John, and Helen Timperley. "The Power of Feedback." *Review of Educational Research*, vol. 77, no. 1, 2007, pp. 81–112., doi:10.3102/003465430298487.

Iarossi, Giuseppe. *The Power of Survey Design a User's Guide for Managing Surveys, Interpreting Results, and Influencing Respondents.* The World Bank, 2006.

Kongsved, Sissel Marie, et al. "Response Rate and Completeness of Questionnaires: A Randomized Study of Internet Versus Paper-and-Pencil Versions." *Journal of Medical Internet Research*, vol. 9, no. 3, 2007, doi:10.2196/jmir.9.3.e25.

Skonnord, Trygve, et al. "Survey Email Scheduling and Monitoring in ERCTs (SESAMe): A Digital Tool to Improve Data Collection in Randomized Controlled Clinical Trials." *Journal of Medical Internet Research*, vol. 18, no. 11, 2016, doi:10.2196/jmir.6560.

Sunstein, Cass R., et al. *Nudge: Improving Decisions About Health, Wealth, and Happiness*. Yale, 2014.

Appendices

Appendix A: Sample Survey Request Invitation Email

This is a sample email that is sent to cohort members inviting them to complete a Keep Asking survey.

Hello,

I am conducting a periodic survey about Social Events (Cuba Libre Night) as part of Princeton Charter Club. I would like to learn about your experiences and opinions with respect to this topic. Your response is important for identifying areas of improvement and tracking progress.

Please complete the Social Events survey within the next 3 days.

Thank you,

John Smith

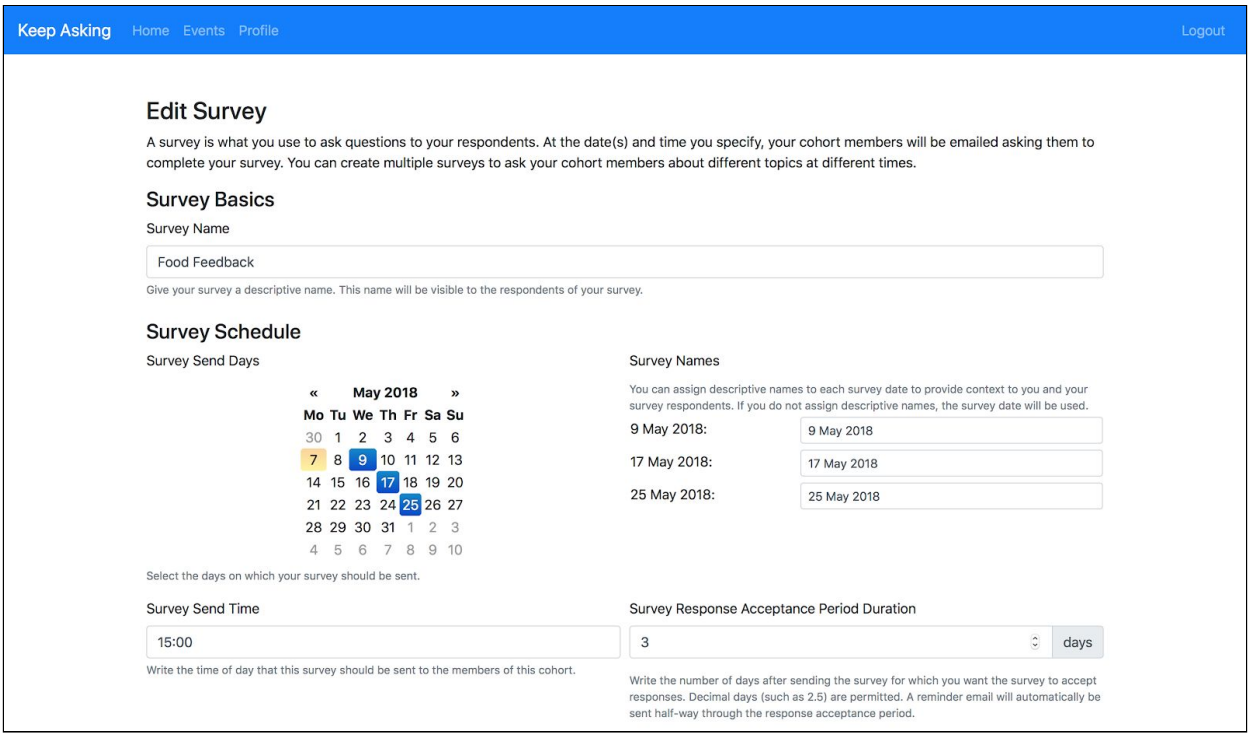
(You are receiving this email because John Smith added you to a Keep Asking survey cohort. To stop receiving these emails, please contact John Smith. Keep Asking is located at 35 Olden Street, Princeton, New Jersey, USA.)

Appendix B: Similarity Between Back-End Data Representation and Front-End Interface

The following block of code is the schema that Keep Asking uses to internally represent a survey set object. This is a simple schema with explicitly defined types and relatively little nesting.

```
{
  cohort: ObjectId,
  name: String,
  surveys: [{
    date: Date,
    name: String
  }],
  responseAcceptancePeriod: Number,
  questions: [{
    title: String,
    id: String,
    kind: {
      type: String,
      enum: ['text', 'scale', 'choice', 'rank']
    },
    options: [String],
    textAreaSize: {
      type: String,
      enum: ['small', 'large'],
    },
    multipleChoice: Boolean
  }]
}
```

This schema very closely maps to the front-end interface that Keep Asking uses when creating or editing a survey set, shown below. This clarity of data representation simplifies the task of collecting and displaying the information on the front-end.



Appendix C: Keep Asking Source Code

The source code of Keep Asking can be viewed at https://github.com/sebthedev/keep-asking.