Durham University

Advanced Research Computing

Requirements Analysis

Research Software Engineering Technical Specification

PROJECT

Building novel architecture to measure strength and variation of international scientific community opinion

DEPARTMENT & PROJECT STAFF

Department of Philosophy, Durham University
Prof. Peter Vickers
peter.vickers@durham.ac.uk

AUTHOR

Dr. Samantha Finnigan
samantha.finnigan@durham.ac.uk

DATE

17/07/2023

Advanced Research Computing
The Lodge, Durham University, DH1 3LE, United Kingdom

Table of Contents

Executive Summary

Humanity needs a way to pool scientific community opinion quickly and efficiently on a given statement of interest. This should be on a very large scale, such that one can have confidence that the result reflects international scientific opinion. For this pilot project (2022-23) Peter Vickers has built a network of 30+ academic institutions around the world. Personal, one-to-one emails are to be sent locally to all relevant scientists at those institutions, asking for a yes/no answer to a given question. The scientist answers by hitting a button embedded in the email, and confirming the response in a second step. Each scientist on our list should have one vote only, and nobody else gets to vote. Voting should be anonymous, with any 'token' linking the scientist to his/her vote being destroyed after approx. two weeks. Votes should be tagged to academic department, and institution, for subsequent data analysis. Originally the project was set up with Word, Excel, and Microsoft Forms, but several problems were encountered; tailored architecture is needed.

Glossary

Term	Definition
API	Application Programming Interface: a mechanism using well-defined function calls for data exchange between programs or within a single application
LAMP	Linux, Apache, MySQL and PHP – a standard webserver technology stack
Python	Python is a high level, general purpose programming language
Git	"Global information tracker". Named for its creator, a distributed version control system (VCS) or content tracker, used to track changes in a set of files
GDPR	General Data Protection Regulation, a legal framework regulating the processing of personal data
MITM	The "Man In The Middle", an attack vector where a bad actor inserts themselves between sender and receiver to modify or intercept data
MVP	Minimum Viable Product, the absolute minimum set of features required to create the first viable version "1.0" of a program or product.
ORM	Object Relational Mapper, an abstraction layer handling database structure and connectivity from a higher-level language (like Python, see above)
SAR	Subject Access Request, a process by which an individual can request data pertaining to them from a public organisation
WSGI	Web Server Gateway Interface, a web server communications protocol
ZKP	Zero Knowledge Proof, a cryptographic method of proving that information is true without knowledge of the information itself or any other knowledge

Project Overview

The project is envisioned as providing a large network within the scientific community, with contact managed through institutional contacts. Currently, the methodology involves the use of an ad-hoc arrangement where these contacts create a mail-merge email using a Microsoft Excel spreadsheet of contacts, which is emailed to the wider network.

In the envisioned system, these emails would contain a unique link which when clicked by the contact records their vote at a centralised location, in this case Durham University. However, there are a set of key requirements in this:

1. Votes should be anonymous. It should not be possible to connect a vote to a voter, and thereby make assertions about the opinions of a single individual

This is of particular importance where the question topic is controversial and may dissuade some contacts from participating if they thought that their vote could be tied to them.

2. Votes can only be cast once by any given individual. It should not be possible to double-vote (i.e., via replay attack) through which an individual could skew the result in favour of one answer.
3. It must be possible to retain some key pieces of demographic information. Responses are tagged with the type of Scientist, and their institution. For example, that the vote-caster was a chemist, residing at Durham University.

It is important to note that the more pieces of metadata (e.g., demographic information) which are collected, the greater the possibility that an individual's identity could be discovered by combining those key pieces of information. Rocher *et al*. (2019) were able to de-anonymise 99.98% of individuals using just 15 pieces of metadata. Within the restricted cohort of individuals in this study, it is highly likely that fewer pieces of data would be required. For example, just the three categories "female, research software engineer, at Durham University" could lead to the linking of an opinion with an individual with high certainty. Per Rocher *et al.*: "*The rule of thumb is the more attributes in a data set, the more likely a match is to be correct and therefore the less likely the data can be protected by "anonymization.""* For the purposes of this system, scientists are put into 5 categories: Physics, Chemistry, Biology, Earth Sciences, and Health Sciences.

4. Responses are made on a five-point Likert scale, from "Disagree strongly" through "Agree strongly".

Likert scales are vulnerable to two biases: acquiescence bias and social-desirability bias. The acquiescence bias relates to our tendency to agree with others' viewpoints, and the social-desirability bias leads us to report views that are regarded favourably by others. To this end, we should not personalise the response page based on the voting link: these biases can be reduced if an individual is not visibly asked to provide details or be linked to an identity.

Further, the inclusion of a central point on the Likert scale is important, as it can demonstrate ambivalence (Clear *et al*. 2018). An example of a question which could be asked through such a system could be, "*Do western and non-western scientists agree that climate change is caused by human activity?*" Naturally, care must be taken with the framing of the research question, but the system itself is not concerned with this, acting instead as a platform for research.

5. Responses should not be vulnerable to exfiltration from the organising institution using legal processes such as subpoena.

We will need to balance aggregation of data with the necessity to satisfy statistical queries about voter demographics for a given response. A short overview of the legal constraints follows.

Legal Frameworks and Data Exfiltration

A subpoena is a coercive summons issued by a court requiring an individual or organisation to produce evidence. Subpoena are used in the United States to compel testimony. The 1970 Hague Evidence Convention would be the fallback for a lawyer in the USA attempting to exfiltrate data from a British company[1], using a witness summons which performs a much narrower discovery function than the US subpoena.

A witness summons is the equivalent UK process for compelling an individual to produce evidence at trial: the word 'subpoena' is no longer used in UK law. A witness can refuse to comply with a witness summons on the grounds that they cannot produce the requested evidence, or that they have a duty of confidentiality to any person to whom the evidence relates, so long as that duty outweighs the reasons for the court summons, warrant, or order[2].

There are other legal routes which do require a response, which are accessible without the intervention of a court of law. A Freedom of Information Request (FOI) can be issued to a public institution, such as a university, by anyone at any time. However, various exemptions are listed in the Freedom of Information Act: for example, a request can be refused if it would disclose personal information in contravention of the UK Data Protection Act 2018, or the General Data Protection Regulation (GDPR).

However, this must be balanced against the Public Interest Test, or PIT: an exemption can only hold if it is not in the public interest to release that data[3]. GDPR would provide protection against this, as the responses of individuals could be argued to be their personal data, which as the data controller we would be unable to release. One can imagine that it would be difficult to argue that it is in the public interest to reveal information about how any given individual voted, and it would likely be allowable to release summary information (demographics, etc) at an aggregate level to satisfy a FOI request.

[1] https://en.wikipedia.org/wiki/Hague_Evidence_Convention
[2] https://www.defence-barrister.co.uk/witness-summons
[3] https://ico.org.uk/for-organisations/guide-to-freedom-of-information/refusing-a-request/

Further, an individual can make a Subject Access Request (SAR) to access data pertaining to them personally. For example, a participant might request for us to provide records on how they, individually, voted. Such requests could be time-intensive to satisfy, but do not provide a route through which broad-category voting data relating to others could be exfiltrated.

While we therefore do not anticipate that it would be possible to reveal the responses (personal data) of participants using the legal framework within the UK jurisdiction, the author is not a barrister or legal practitioner. It is therefore sensible to design the system in such a way that it is not computationally feasible for anything more than summary data to be returned in response to the various legal routes. The best protection against sensitive data exfiltration is not to hold sensitive data: if data is anonymised, or aggregated, at the point at which a vote is recorded, then it will be more difficult to link an identity to a vote except in a small number of edge-cases[4].

Cryptographic Voting

Electronic voting has been investigated as a research problem within the computing sciences, especially with regards to building cryptographic methods to enforce the voting primitives of anonymity and verifiability: which per the above listed requirements are of significant concern for this project. The cryptographic primitives underpinning such voting schemes have existed for several decades, with the major contributions to the field relying on Zero Knowledge Proofs (ZKPs)[5]. A thorough primer to this work can be found in (Bernhard and Warinschi 2016).

The underpinning motivation in the development of such protocols is that a reliance on a so-called “trusted third party” to tally votes and provide the voting infrastructure leaves voting systems open to manipulation by the third-party or a motivated attacker relying on that single point of failure. Cryptographic protocols have been developed to remove that point of failure, with examples including the open vote network (OV-net) protocol (Hao et al. 2010). More recently, such protocols have been implemented using blockchain technologies including Ethereum (McCorry et al. 2016) as a base, making use of the distributed ledger to ensure that records are immutable and computationally infeasible to modify or falsify.

Within this project, our main concern lies in getting an initial implementation running in a short amount of time. Therefore, while cryptographic methods should be considered the ‘gold standard’ for truly allowing users of the system to trust that the results it produces are accurate, in the short term a centralised solution is acceptable.

There is a balance to be struck in terms of the time available to implement this project, against the stated requirements of anonymity, and against the need to provide useful metrics. This is a non-trivial problem to solve, and as such, the following system specification outlines a minimum viable product to address the research need. As

[4] For example, in the case where only one individual has voted, if stored in plaintext aggregate data would not protect their response from being identifiable.
[5] https://github.com/kantuni/ZKP

such, the anonymity requirement (1) outlined above will not be truly solved within the constraints of this pilot project.

System Workflow

The existing manual workflow to survey participants is as follows:

- A list of participants is prepared by the project team. This list includes email addresses and demographics of the participants.
- A co-ordinator at each institution is instructed to email participants and then collect replies.
- Replies are emailed back to the host institution.

In the proposed system, a web interface will collect votes. A back-end system is required to set up and generate a survey, including a list of generated codes which link a response to an individual. There are a set of three workflows (WF1,2,3) which make up this survey process:

WF1 / Workflow 1: Survey Distribution

1. The survey administrator poses the question by entering it into the system.
    - **Requirement**: administrator access to create new survey; enter questions
2. They generate a list of emails from the database of participants, alongside a list of voting links.
    - **Requirement**: system generates **unique voting links** per participant.
3. Separate lists are returned by institution, containing only participants within that location.
    - **Requirement:** participant list formatted as CSV or Excel spreadsheet
4. These lists are sent out to the organisational contacts.
    - **Option:** this can happen automatically (system sends out emails) or manually (list returned to project admin who attaches lists to emails and sends to organisational contacts).
5. Organisational contacts perform a mail-merge to send out survey template to list of participants.

Following the distribution of the survey, a second workflow (WF2) describes participant interaction with the system.

WF2 / Workflow 2: Response Capture

1. Participants, receiving the email, click their personal link to cast their vote.
    - **Requirement:** The link contains a **unique voting link**, a participant identifier used in the database to record that a participant has voted. The link cannot be used more than once.
2. The participant goes to the voting page, which contains the question and the Likert scale of options, e.g.:

"*The currently observed changes to worldwide climate are a direct result of human activity"*

Strongly Disagree ○ Disagree ○ Neither Disagree nor Agree ○ Agree ○ Strongly Agree ○

3. The voter clicks their option and then clicks "Submit".
4. Their response is sent to the server in combination with their unique ID.
    - **Requirement:** Encrypt the response with an *https://* secure connection.
5. The server records that the voting link was used, and inserts the response into the database.
    - **Option:**
        a) Aggregation is done automatically as responses come in, or:
        b) Responses are temporarily stored linked to the unique ID, then following closure of the survey the responses can be aggregated

Following closure of the survey, it is necessary to return the results for analysis by the project team.

WF3 / Workflow 3: Results Retrieval

1. The survey administrator closes the survey.
    - **Option**: This could optionally be handled by setting a date/time for the closing of the survey at creation-time. The default longevity of a survey will be two weeks.
2. The results of the survey are retrieved by the survey administrator through the administration panel.
    - **Requirement:** Per requirement (1), results cannot identify voters.
    - **Option:** Data can be
        a) returned as anonymised records in an Excel spreadsheet, or
        b) pre-aggregated on the server
    - In the case of (b), we will need to identify the aggregations to be performed in advance. This will be inflexible, but would provide a better level of anonymity for participants in the event of data exfiltration, as described above.

Attacks on this System

Attack 1: MITM. Trusted organisational survey administrators are effectively a 'man-in-the-middle'. Bad actors in such a system could cast votes on behalf of individual participants, and it would not be possible to prove that this had occurred. We must trust survey administrators to act in good faith in this system design.

Attack 2: Data Exfiltration. Bad actors with access to the server would be able to exfiltrate data as it is being collected, as unique IDs would link directly to participant records unless these are somehow kept separate. Direct Aggregation (i.e. never storing identifiable data, only tallies) would be one mitigation to this attack.

System Architecture Design

The application is formed of a front-end and a back-end, with the front-end written in HTML5 and Vue.js, and the back-end software written in a server-side language (in this case, Python), running as a WSGI gateway behind the *nginx* web server. Database activity will be handled using MySQL, a relational database, with structure deferred to a Python ORM such as Django ORM or SQLAlchemy.

Data Model

The survey database design will be formed of four primary tables: a table containing participants and their demographics, a table containing active surveys, a table containing active user-survey unique links, and a table containing survey responses. Further tables will be required to handle administrative user management and scheduled tasks.

Tables must be UTF-8 formatted to account for special characters, for example in participant names. The InnoDB storage engine is preferred. No direct SQL should be required as queries will be handled through the ORM. If SQL queries are required, they should be parameterized to avoid the possibility of injection attacks.

PARTICIPANTS Table

Contains details of participants. Data to be imported from existing Excel spreadsheets.

Field	Type	Primary/Foreign Key? Index?
ParticipantID	INTEGER	PRIMARY KEY
Name	VARCHAR(255)	
Title	VARCHAR(32)	
Email	VARCHAR(255)	
Institution	VARCHAR(255)	INDEX
Profession	VARCHAR(64)	INDEX
(Further rows for demographic information as required)		

Demographic information will match the demographics already collected by the project team. It will be necessary to import data from the existing Excel spreadsheets, and provide functionality to add and delete records.

SURVEYS Table

Contains active surveys. There may be more than one survey at any given time:

Field	Type	Primary/Foreign Key? Index?
SurveyID	INTEGER	PRIMARY KEY

Question	TEXT (limit 64kb)	
Active	BOOLEAN	
Kind	ENUM(LIKERT)	
Expiry (**optional per WF3.1**)	DATETIME	

NB: There are several extensions to this table which could be added in future. The 'Kind' field can be extended to allow for different survey types. In this instance, we handle only one survey type. Alternately, we could introduce a 'Fields' table which would allow form-builder functionality by storing multiple question and answer types along with the exact structure of the form against the survey ID.

Regarding Expiry, it may make sense to build the app in Django rather than Flask, even if we only want to support this in future, as it has a range of task-scheduling add-on packages which can perform the necessary aggregation and erasure operations automatically.

ACTIVE_LINKS Table

Field	Type	Primary/Foreign Key? Index?
ParticipantID	INTEGER	FOREIGN KEY (PARTICIPANTS)
SurveyID	INTEGER	FOREIGN KEY (SURVEYS)
UniqueLink	VARCHAR(128)	
VoteUsed (**optional WF2.5b**)	BOOLEAN	

The Active Links table serves to temporarily link participant responses to identities in the database, while the survey is running. Dropping entries for the survey from this table serves to effectively un-link the participant from their response. Ideally, this would happen as the vote is cast: the participant demographics are copied into the results table with the UniqueLink as the key, then the entry for that link is dropped from this table.

If option WF2.5b is chosen, then the VoteUsed column will be necessary to track that a participant has voted. In the case of WF2.5a, the link will be removed, and the absence of a link in the table when a voting link is clicked by a participant will mean that they have already cast their vote.

RESULTS table

Field	Type	Primary/Foreign Key? Index?
ResultID	INTEGER	PRIMARY KEY
UniqueLink (**optional WF2.5b**)	VARCHAR(128)	FOREIGN KEY (ACTIVE_LINKS) ON DELETE SET NULL
Vote	JSON	

Institution	VARCHAR(255)	INDEX
Profession	VARCHAR(64)	INDEX
(Further rows for demographic information as required)		

The Results table stores voting results. It is populated when a vote is cast. Demographic information is copied from the Participant table, but is not linked directly. References to identities must be broken when an entry in the Active Links table is deleted. We therefore copy the demographic data into the Results table at the time when that link is broken (i.e., when a vote is cast).

Back-End

The back-end will be written in Python, using a suitable web framework such as Flask or Django. Both are modern, performant server-side frameworks. Django offers a more complete feature-set, but Flask offers greater flexibility. In Django, an admin panel is included as standard, but the generic approach taken often means that usability suffers. For a comparison between both frameworks, see https://www.trio.dev/blog/django-vs-flask.

The front-end webserver will be nginx, which will reverse-proxy requests to the Python application server. Server-side logging will be used to track requests and to enable debugging. An API will be designed to handle the following functions:

- Vote submission using unique link
- Administrative functions (behind a user login) including:
  - Create survey with a question and (**optional WF3.1**) expiry
  - Cancel/retract a survey
  - Upload participant data from Excel Spreadsheet or CSV
    - In the short-term MVP, uploading participant data will overwrite all data. If there are active surveys, the option will be given to cancel them.
  - Retrieve existing participant data (tabulated and paged)
  - Retrieve surveys (and metadata incl. active/inactive status and expiry)
  - Retrieve survey results
  - **Optional WF3.2b**: aggregate results
  - **Extension**: basic data visualisations for aggregated results
  - **Extension**: view administrative login attempts

The server will also return web-page content. The use of server-side rendering for page templates may be desirable (https://vuejs.org/guide/scaling-up/ssr.html) but would require running a node.js server alongside the Python backend.

The data format for the existing Excel spreadsheets will need to be determined from an example of the data provided by the project team. Once implemented, this will determine the format for all future data uploads. **Optionally**, we could integrate

Microsoft OneDrive functionality to allow storage of anonymised survey results directly in the survey administrator's OneDrive account.

Front-End

The web front-end will be built using Vue.js, a Javascript framework for building user interfaces. Vue uses standard HTML, CSS and Javascript with an intuitive API, and offers many addon libraries to enhance page functionality without large amounts of custom code.

The pages required within the web application are:

- User-facing:
  - Index page
  - Voting page
- Administrator-facing:
  - Login page (**question:** handle this with Shibboleth or MS365 OAuth?)
  - Set a survey
  - View existing surveys
    - Filter for active and inactive surveys
    - Cancel an active survey with a button and confirmation dialog box
  - Upload new participant data
    - **Extension:** filter uploaded data for new records and ask for approval, instead of just overwriting everything.
  - View existing participant data
  - Download survey results

A link to the institutional ethical approval should be included within the project website on the voting and index pages.

Project Management

The project will be managed according to the tasks listed in the Milestones and Reporting section below. All code will be stored in GitHub, an online version control system (VCS). It is the preference of the RSE team that code is open-source and licensed under an appropriate license requiring attribution, for example the BSD or MIT licenses.

Documentation will include guides on the user and administrative workflows within the application. Code will be commented to a self-documenting standard, with deployment details included in a Readme.md file on the GitHub site, following the RSE team template.

Milestones and Reporting

Milestone	Tasks	Reporting	Days
M1 - Analysis (COMPLETED)			
1.1	Analysis and design stage, perform requirements analysis of system workflow	Email report, COB 24/01/2023	3
1.2	Architecture design	Email report	2
1.3	Design work plan (distribution of tasks)	Meeting to review work plan	1
M2 - Development			
2.1	Implement back-end frameworks and automated test infrastructure	None required	2
2.2	Create app on Azure AWH platform	None required	2
2.3	Implement Survey Administrator workflow (WF1)	Email report	
	Back-end (Samantha): • Login functionality • Create survey in DB • List active surveys • Cancel a survey • Upload Participant Data • View Participant Data	 1 2 0.5 0.5 4 3	11
	Front-end (Joanna): • Login page • Create Survey page • List + cancel active surveys • Upload form for participant data • Tabulated view for participant data • Framework learning time (+50%)	 1 2 3 1 3 5	15
2.4	Implement participant survey workflow (WF2)	Email report	
	Back-end: • Generate Unique Links • Receive Survey Votes • Store Vote in Database • Un-link data records • Export per-institution survey links	 2 1 0.5 0.5 2	6
	Front-end: • Survey page (by SurveyID) • Submit survey form data • Download of survey links sheet • Framework learning time (+50%)	 3 1 1 3	8
2.5	Implement results retrieval workflow (WF3)	Meeting to review interaction flow	

Milestone	Tasks	Reporting	Days
	Back-end: • Close survey following expiry • Close survey manually • Aggregations (**option WF3.2b**) • Tabulate and return results	 2 0.5 3 2	7.5
	Front-end: • View expiry for surveys • Close survey manually • Download completed surveys • Erase completed surveys • Framework learning time (+50%)	 0.5 0.5 0.5 0.5 1	3
2.6	Populate database with participant data	None required	2
2.7	Finalisation of development and bugfixing	None required	5
M3 - Testing			
3.1	Design mail-merge templates	Email report	2
3.2	Perform initial survey with a limited group of participants	Meeting to discuss results of initial survey, successes/failures, and content of documentation	1 (gap for deployment time not included)
3.3	Develop basic documentation to assist organisational contacts, and to outline workflows and administrator functionality		10
3.4	Finalise project (slack / overflow time)	Email report and approval	2
		Total Hours:	82.5 days total
		Hourly estimate at 80% FTE (6.4 hours / day)	528 hours
NB: 6 (38.4 hours) days of Project analysis (Milestone M1) included in total hours, but work complete.			

Time Breakdown By RSE

Research Software Engineer	Milestone (Mx) / Split	Days (Hours)	Hours
RSE1: Samantha Finnigan	M1 / 100%	6	38.4
	M2 / 49%	30	192
	M3 / 50%	7.5	48
RSE2: Joanne Sheppard	M2 / 51%	31.5	201.6
	M3 / 50%	7.5	48
	TOTAL	82.5	528

RSE time overlaps. The length of this project is **39 days** with 2 RSEs at 80% FTE.

References

Bernhard, D. and Warinschi, B. (2016). "*Cryptographic Voting — A Gentle Introduction*". Available at: https://eprint.iacr.org/2016/765.pdf

Clear, A. K., Mitchell Finnigan, S., Olivier, P., Comber, R. (2018), "*ThermoKiosk: Investigating Roles for Digital Surveys of Thermal Experience in Workplace Comfort Management"*, In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI'18. 382. ACM, Montréal, Québec, Canada. https://doi.org/10.1145/3173574.3173956

Hao, F.; Ryan, P.Y.A.; Zieliński, P. (2010). *"Anonymous voting by two-round public discussion"*. IET Information Security. **4** (2): 62. doi:10.1049/iet-ifs.2008.0127. Available at: http://homepages.cs.ncl.ac.uk/feng.hao/files/OpenVote_IET.pdf

McCorry, P. Toreini, E. and Mehrnezhad, M. (2017). "*Removing Trusted Tallying Authorities*". Financial Cryptography. Available at: https://www.economist.com/sites/default/files/newcastle.pdf

Rocher, L., Hendrickx, J.M. & de Montjoye, YA. (2019). "*Estimating the success of re-identifications in incomplete datasets using generative models.*" Nature Communications 10, 3069. https://doi.org/10.1038/s41467-019-10933-3

Appendix A: Open Source Survey Systems

Name	Features	Pricing / Self-host option?
LimeSurvey https://www.limesurvey.org/	Full-featured survey platform Encrypted responses	30% discount for universities[6] Self-hosting[7] from 99€/year.
JD ESurvey https://www.jdsoft.com/jd-esurvey.html	Java form builder	Self-hosted
ngSurvey https://github.com/ristemingov/ngSurvey	AngularJS survey form builder. Last update 2021. Not actively maintained?	Self-hosted
Quick Survey https://github.com/simonv3/quick-survey/	Question-and-answer or multiple choice list surveys. Run on https://sandstorm.io/	Self-hosted
TellForm https://www.tellform.com/	Free, opensource form builder similar to Google Forms or TypeForm	Self-hosted No monthly fee
SurveyProject https://www.surveyproject.org/	Microsoft-compatible C# .NET online webforms tool. Dated interface.	Self-hosted

[6] https://www.limesurvey.org/solutions/universities

[7] https://www.limesurvey.org/blog/20-blog/120-hosting-solutions-two-sides-of-the-same-coin?highlight=WyJzZWxmLWhvc3RlZCJd