CSCI 370 Report

Jimmy Baldwin
Camden Lyles-Smith
James Vongphasouk

Revised June 13, 2023

CSCI 370 Summer 2023

Dr. Paone

Table 1: Revision history (for submissions)

Revision	Date	Comments
New	May 17, 2023	Completed Sections: I. Introduction II. Functional Requirements III. Non-functional Requirements IV. Risks V. Definition of Done XI. Team Profile
Rev – 2	May 19, 2023	Updated Sections: I. Introduction - added background information on Orderly and need for product II. Functional Requirements - separated requirements into hard and nice-to-have requirements III. Non-functional Requirements - added subsections for qualities that product should have IV. Risks - labeled table V. Definition of Done - revised to be echo of hard requirements Completed Sections: Appendix A - Key Terms Appendix B - Tables and Figures
Rev – 3	May 22, 2023	Completed Sections: VI. System Architecture
Rev - 4	May 26, 2023	Updated Sections: VI. System Architecture - added more Figures to describe both back-end and front-end components
Rev - 5	May 31, 2023	Completed Sections: VII. Software Test & Quality VIII. Project Ethical Considerations Updated Sections: I. Introduction - changed all Figures in section to tables and explained % correct range II. Functional Requirements - changed all Figures into tables

		IV. Risks - updated label name VI. System Architecture - added keys to Figures Appendix B - updated label names and orders
Rev -7	June 7, 2023	Updated Sections: VII. Software Test & Quality - added software quality tests table and paragraph about the ethical considerations if Software Quality plan fails. Need to add unit/integration tests and get the results from them to put in the next section. XIII. Team Profile - all profiles same size and format Completed Sections:: IX. Results - need to finish getting testing framework X, Future Work - flush out ideas more and be more concise XI. Lessons Learned - expand on the lessons XII. Acknowledgments - format text
Rev - 8	June 13, 2023	Updated Sections VII. System Architecture - Updated to match final design and system VIII. Software Test and Quality - added tests for the backend X. Results - added results of the tests for backend and frontend Appendices - updated table of Figures and added extra product setup instructions

Table of Contents

I. Introduction

Orderly Health provides customers with accurate and accessible provider data through their service: Orderly Provider Directory (OPD). OPD has up-to-date data about a provider such as their National Provider Identifier (NPI), specialty, primary address, and primary fax information. To make sure that this data is high quality, Orderly trains machine learning (ML) models on this data and tests their models against ground-truth data or confirmed attested data. The data is phone attested through data attestors. Each data attester follows a script provided by Orderly to confirm practitioner/provider information and if incorrect, gets the right information. The process of getting the right information may involve more phone calls.

Currently, the attested dataset sent back from data attestors requires additional formatting to fit with Orderly Health's schema so it is hard for their data science team to update data points in their models. Information like address should be validated during the phone call because an invalid address is useless, along with invalid phone numbers, NPI, etc. Additionally, there were other areas to make the process more streamlined to be more efficient and accessible. Orderly also uses an address Application Programming Interface (API) to verify addresses which the team members of the group (TMs) have considered using in the end product.

A snippet of the attested data with a practitioner's primary specialty that is sent back to Orderly is provided in Table 2. This is an example template of attestation data that data attestors would return to Orderly. There are 3 other templates for address, primary address, and fax that are not included here, but look similar. This method is efficient to get the attested data for each of the data fields, but Google Sheets does not have strict restrictions on the data to be entered that make the data harder to ingest for the data science team. For example, the "Validation Completion Date" is in the form,yyyy/mm/dd, but not every data attestors agent will follow that standard and may input date in the form, mm/dd/yyyy.

Table 2: Example Monthly Attestation Template sent back to Orderly from data attestors

Willing to participate?	**If no, is there a better time to call back?**	**Name**	**Do you have practitioner(s) by the name of ________?**	**Fax**	**Is this the correct FAX # for this location?**	**If NO, can you provide the correct fax #? [record new fax]**	**Call Quality**	**Validated**	**Validation Completion Date**	**Comments**	**# of Calls**
Yes		John Doe	Yes	5555555557	No	5555555558	Good	Yes	2022/08/19	reached the practitioner	2

Yes		Jane Doe	No	5555555559			Good	Yes	2022/08/19	the location was no longer associated with this practitioner so we did not ask about them about the fax number	3
No		Jim Doe		5555555561				Yes	2022/08/19	line no longer in service	3

The main stakeholders are data attestors' agents because they are directly using the system to create the attested data set, and Orderly's Data Science team because they use the attested data set to create better products. After the TMs has created a product, Orderly's team of software engineers and data scientists/engineers will take over and integrate it into their system.

II. Functional Requirements

Orderly requests an improved attestation system where the "ground truth" becomes consistent and accurate leading to better ML models. The current system being used (spreadsheets and email) has some benefits like having multiple agents editing one spreadsheet at the same time which is very productive and that the agents are able to edit responses to questions "freely" as the phone conversation may direct to different parts of data needing to be attested in nonlinear fashion. To summarize, the "spreadsheets and email" system is easily accessible and flexible which the TMs have kept in mind as nice-to-haves.

The next two sections are the hard requirements that Orderly have accepted as a minimum viable product and nice-to-haves which are not required in the product but nice to have.

a. Hard Requirements

A summary of the improved attestation system is that it is able to guide the agents through the call, validate information at the time of the call, standardize the data collected, and create the attested dataset so it is easily accessible for Orderly's system. Orderly uses the Google Cloud Platform (GCP) for their applications and they primarily use BigQuery as their data warehouse. The hard requirements are as follows:

- The improved attestation (front end and back end) system must work in Orderly's system
- There must be documentation on the process to load the data to be attested
- data attestors must be able to sign in to keep track of who is attesting what data
- Any Application Programming Interface (API) secrets shared are to only be used in the backend
- Front-end specific
    - The data attestation user interface (UI) must allow data attestors agents to attest 4 points of data associated with the practitioner:
        - Primary Specialty

        - Fax
        - Address
        - Primary Address
    - data attestors should be able to answer the fields in the attestation templates (example template shown in Table 2) through the UI
    - There must be synchronization for showing which calls are being taken, and which call is being handled by which data attestor
    - data attestors must have the ability to jump around between questions on the portal
    - Addresses, phone numbers, and zip code entries must be validated in real time on the portal
- Back-end specific
    - Local database stores data (attestation entries) from front end
    - Attested data must be compatible with the bq_snapshot schema and able to sync to Orderly's BigQuery
    - Attested data must be standardized for orderly's ML algorithms
    - Attested data includes update timestamp for when data is validated for each practitioner in the database
    - Orderly must be able to put together a new attestation document every month for data attestors using the data stored in BigQuery and gathered from the data attestation UI

b. Nice-To-Haves

These features fall under the nice-to-have category as they are not required to satisfy Orderly's request. There may be more nice-to-have features that the TMs encounter while delivering the minimum viable product, but the current list is as follows:

- Data attestors are authenticated through Google
- Data attestors can grant and revoke access to the portal to varying amounts of data attestors throughout the day
- Backups of the local attested database exists

III. Non-Functional Requirements

These requirements are quality requirements to ensure that our minimum viable product follows a set of guidelines to ensure Orderly is satisfied. These are separated into 4 categories of non-functional requirements: performance and reliability, quality, and security. There are no non-functional requirements involving cost, but the TMs were advised to reduce implementations that require using subscription-based services or paid GCP services.

a. Performance/Reliability

These requirements determine how the product should respond throughout its lifecycle:

- Live user data displayed on UI (knowing which users are on) should be accurate and near real-time
- Website should be responsive and handle a large set of users at the same time

b. Quality

These requirements determine how the user experience should feel when using the product:

- data attestors should enjoy working with the UI as much as Google Spreadsheets

- Representative data attestor has told us that he likes the SalesForce Lead's System UI design.

c. Security

These requirements determine how the product stores sensitive information and adheres to standards:

- Data attestors should be able to record call results without sending links or permissions through email
- Database access should be flexible enough that client and data attestors can access different versions of it

IV. Risks

For a five-week project, there are many risks associated with having enough time to learn the skills needed to complete all the necessary features. The TMs have identified 7 possible risks and mitigation strategies to prevent them in Table 3. If the risk does occur, the TMs have also included contingent actions to help reduce the impact if the risk occurs.

Table 3: Risk Analysis Matrix

Risk description	**Likelihood of Risk Occurring**	**Impact If the Risk Occurs**	***Severity Rating***	**Mitigating Action**	**Contingent Action**
The team will be unable to implement a solution that meets all the requirements of back end updating and processing	LOW	HIGH	HIGH	Maintain constant communication with Orderly, and keep them updated on the team's progress delivering code periodically to ensure the client's needs are being met	If the project begins to experience scope creep and exceed the team's capabilities as the team develops more and runs into more issues, consult the storyboard to determine what functionality is the most critical, and consult the client with a solution with reduced scope
Linking BigQuery and the Structured Query Language (SQL) database will not be possible, or will be substantially harder than anticipated	LOW	MEDIUM	MEDIUM	Perform small instance testing early on in the development process to determine early on if the desired solution will be unattainable, and have additional methods of linking local data with BigQuery prepared	Consult the client and inform them that the desired solution will not be feasible, and offer alternative solutions from the methods prepared in contingency
The front-end UI will not work with the desires of data attestors and will need to be overhauled to better suit their wants.	MEDIUM	LOW	LOW	Provide preliminary mockups to data attestors to illustrate the planned UI, and continually gather feedback on the progress of the UI being flexible	Should data attestors be unhappy with the UI, the team can modify the looks or functionality very easily, since it will behave the same with the back end regardless

				to changes in the desires of data attestors	of what it looks like - modify the plans for the front-end
The client will introduce new requirements for the project that will increase scope beyond the capabilities of the team	MEDIUM	MEDIUM	MEDIUM	Maintaining communication with the client and ensuring both the team and the client are on the same page about expectations, primary goals, and stretch goals for the project will be key in reducing scope creep	Should scope creep occur, two options will be present for the team: 1. adjusting schedule to account for new or changed feature 2. the new feature is too great to be included in the project, so negotiate with the client a middle ground between full functionality of the feature and the team's ability to deliver the finished product
The database schema the team will be loading data into in BigQuery will change or be altered during the lifetime of the project	LOW	MEDIUM	LOW	Keep track of the schema and be aware of when it gets updated or modified	Modify the local schema and its communication with BigQuery to integrate with the modified cloud schema
TMs will be unable to fully learn and grasp new tools and languages required to complete the project (one member currently has extensive knowledge of front-end JavaScript frameworks, the others only have rudimentary knowledge of back end)	LOW	HIGH	MEDIUM	Hold check-ins at the daily standups to ensure TMs are able to learn their tools, or if they require additional support from other TMs to expedite the learning process	Pull TMs from the other team into the team struggling (front end to back end, or vice versa) and collaborate fully on the functionality causing struggles

V. Definition of Done

To complete the project, the TMs have created a minimum viable product (MVP) that aligns with the hard requirements specified by Orderly. The MVP is the product that the client will accept with no additional features, which will be restrained by time or complexity. The following features define the minimal useful feature set which were summarized from the hard requirements:

- Compatibility with Orderly's system
- Documentation for the process on loading attestation data
- Sign-in for data attestors to track who is actively making calls
- Application programming interface (API) secrets used ONLY in the back end
- Agents can attest primary specialty, fax, address, and primary address
- Agents can answer the fields in the attestation templates through the portal
- Synchronization for the calls being taken and which calls are handled by which agents

- Can navigate through questions nonlinearly
- Addresses, phone numbers, and zip code entries are validated in real time
- Attestation entries are stored in local database
- Attested data is compatible with the bq_snapshot schema and able to sync to Orderly's BigQuery
- Attested data is standardized for Orderly's ML algorithms
- Attested data includes update timestamp for when data is validated

The client will run a series of tests upon taking delivery of the product to ensure it meets the hard requirements they have specified for the team, and to ensure no issues are present with the product that would decrease its usability and functionality. Tests have primarily revolved around the features defined in the definition of done:

- information is automatically populated when opening a record from the agent view
- confirming information updates the SQL database of attestations and records, if appropriate address field autocompletes and suggests valid addresses
- date field is normalized and autocompletes to store the correct normalized value in the SQL database
- information entered in front end is received and stored in SQL database
- records can be updated in BigQuery based on attested data stored in SQL database
- practitioner data can be pulled from BigQuery into SQL database to be attested

The product has been delivered incrementally to the client in tech demos each week every Friday displaying functionality developed and expanded upon that week, and has been open to feedback and suggestions from the client. Code has been checked into GitHub repository before each tech demo, and whenever large modifications are made to ensure proper version control and the ability for rollback if needed. The completed product has been delivered in a final code check-in to the repository after extensive testing and quality assurance.

VI. System Architecture

In our product, there are two main components (front and back) that interface with each other. Our front end handles the user interface/user experience, user input to attest data, and user sign in. The front end is built using JavaScript React, Firebase Realtime Database, and Google authentication.

Our back end handles the data processing from BigQuery to the GCS buckets, stores local attestation data in a Cloud SQL database, and sends the local attested data back to GCS buckets for data analytics. The Cloud SQL database is a MySQL transactional database that is used for the operations demanded by the front end such as displaying the dashboard of records, information about those records, and sharing permissions. The backend is built using the Java Spring Boot Framework.

In Fig. 1, the product is separated into server-side (back end) and client-side (front end). The backend is composed of 5 major parts. These parts are the BigQuery Data Warehouse that has Orderly's attestation data, GCS buckets to extract the attestation data, MySQL database to store the attestation data to send to/receive updates from the front-end, Google's authentication services and Firebase Realtime Database. The front end is composed of 3 major parts: login, dashboard, and questionnaire. Every user has to sign-in/authenticate themselves before getting to the main dashboard. From there, the user can attest data in the questionnaire which gets sent to the backend.

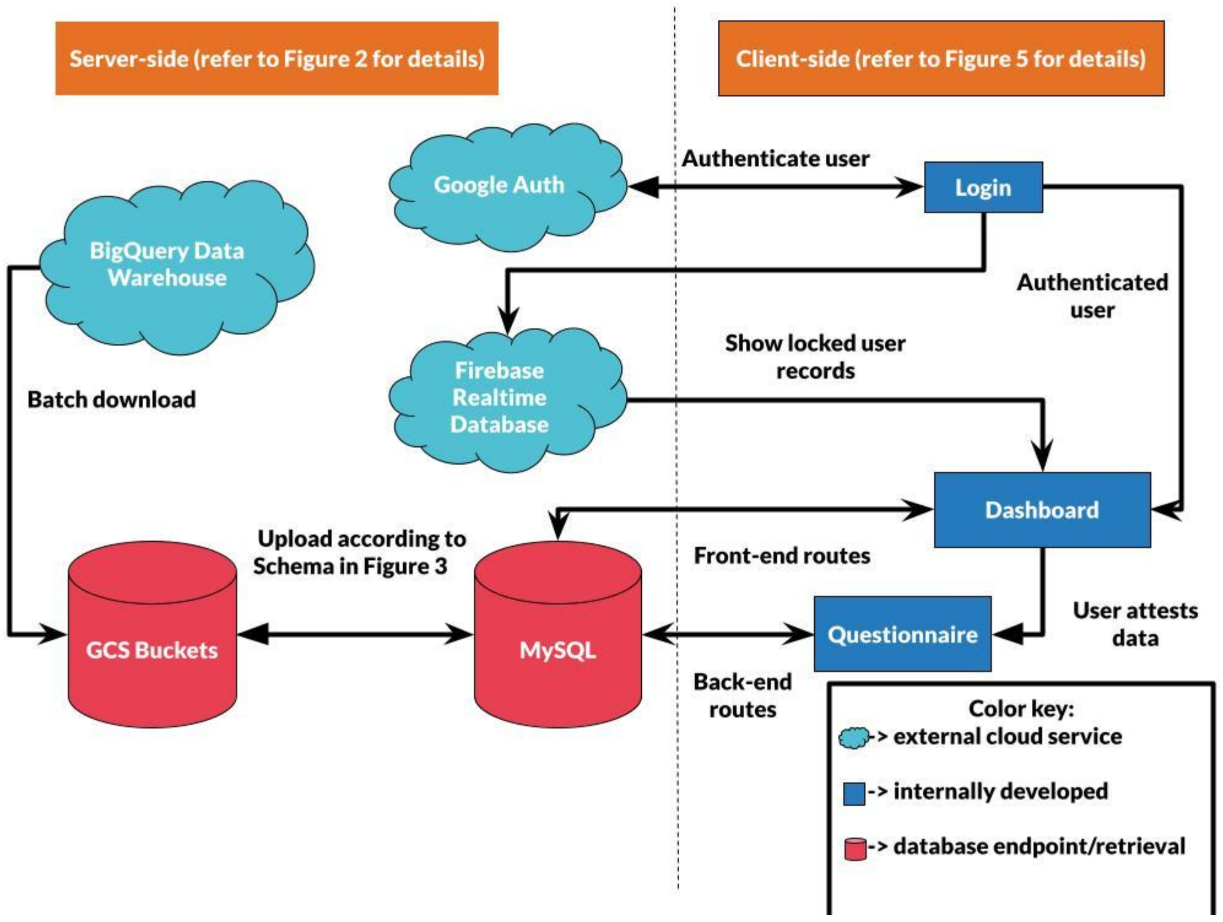
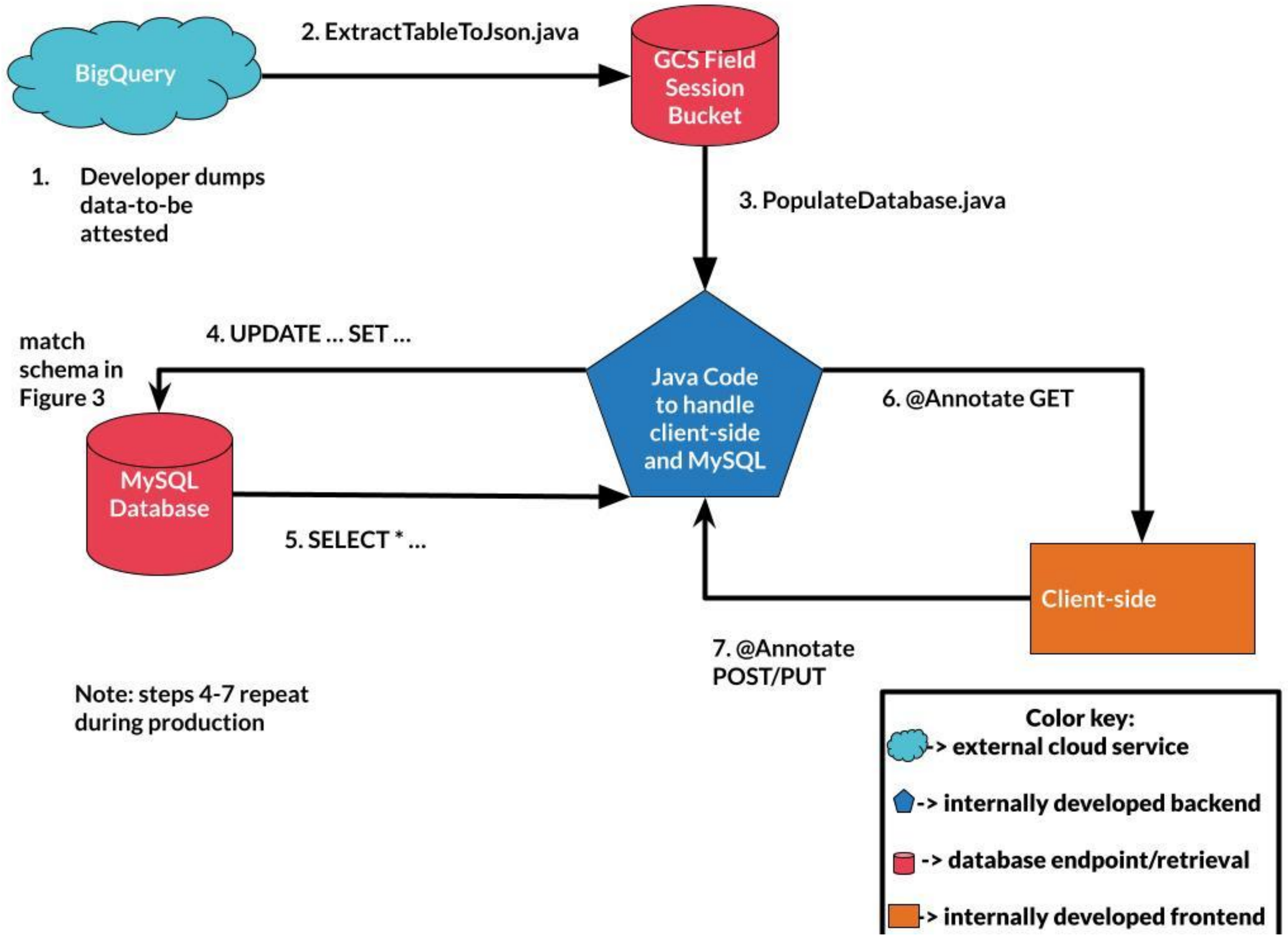


Fig. 1: Initial Front-end and Back-end Design

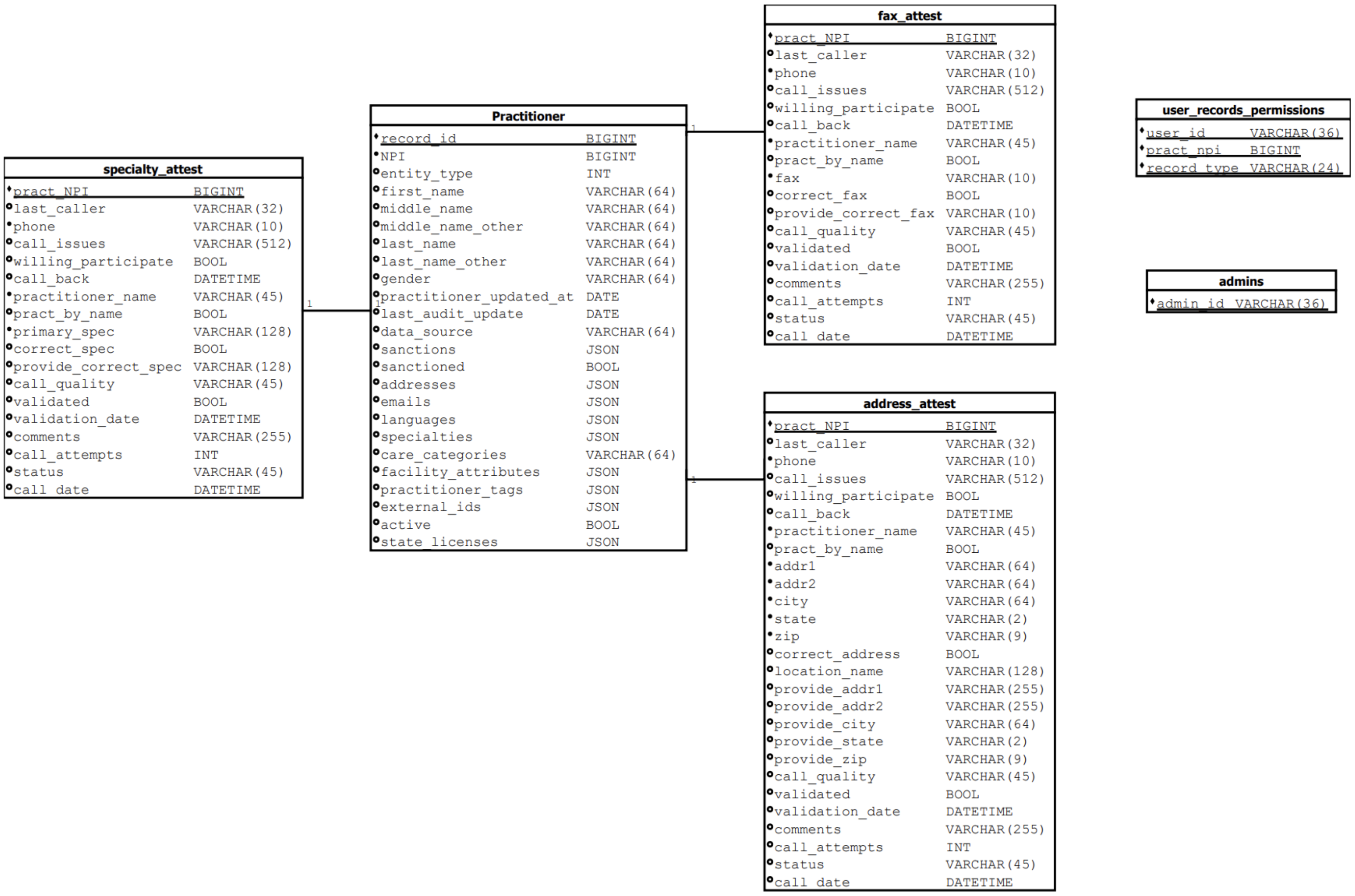
In Fig. 2, the back end contains the low-level flow of how the code makes the data flow. Initially, the Orderly Health Developer dumps attestation data into Bigquery. Then, by using the Java Google Cloud Storage Client, the attestation data goes from Bigquery to the GCS field session bucket to the backend application. Next, the backend stores this data in the MySQL database sure to match the schema in Fig. 3, and sends it to the client-side to attest using Java Spring Framework annotations. Finally, the back end receives the attested data back from the client-side. To make this clear, the TMs have included numbered steps in Fig. 2 as well. The code/methods next to the numbers are abstracted but they are the main methods that handle the data in the backend. Fig. 2 does not include Google Authentication of Firestore because these are products that are only accessible by the front end and are cloud services.



BigQuery
2. ExtractTableToJson.java
GCS Field Session Bucket
1. Developer dumps data-to-be attested
3. PopulateDatabase.java
match schema in Figure 3
4. UPDATE ... SET ...
Java Code to handle client-side and MySQL
6. @Annotate GET
MySQL Database
5. SELECT * ...
Client-side
7. @Annotate POST/PUT
Note: steps 4-7 repeat during production
Color key:
-> external cloud service
-> internally developed backend
-> database endpoint/retrieval
-> internally developed frontend

Fig. 2: Data pipeline from BigQuery to Client-side to GCS Updates

The schema for the SQL database in Fig. 3 contains 6 tables that are used to store information to work with, where practitioner information is stored in a single source of truth, and each practitioner record owns an attestation record of each type - address, specialty, and fax. For ease of use and simplicity in the MySQL database, the schema has been flattened to contain JavaScript Object Notation (JSON) data types for the columns that contain arrays in BigQuery (such as addresses, specialties, and state licenses). Two separate tables contain information for permissions checking to ensure that users only have access to records they have shared. The 'user_records_permissions' table contains rows consisting of a user's uniquely generated identifier, a practitioner's NPI, and the type of attestation record that has been shared with this user; each row represents a record and a user. The 'admins' table contains the unique identifiers of users who are granted access to all records, such as our client at orderly, management at data attestors, or the TMs during testing.



specialty_attest

pract_NPI	BIGINT
last_caller	VARCHAR(32)
phone	VARCHAR(10)
call_issues	VARCHAR(512)
willing_participate	BOOL
call_back	DATETIME
practitioner_name	VARCHAR(45)
pract_by_name	BOOL
primary_spec	VARCHAR(128)
correct_spec	BOOL
provide_correct_spec	VARCHAR(128)
call_quality	VARCHAR(45)
validated	BOOL
validation_date	DATETIME
comments	VARCHAR(255)
call_attempts	INT
status	VARCHAR(45)
call_date	DATETIME

Practitioner

record_id	BIGINT
NPI	BIGINT
entity_type	INT
first_name	VARCHAR(64)
middle_name	VARCHAR(64)
middle_name_other	VARCHAR(64)
last_name	VARCHAR(64)
last_name_other	VARCHAR(64)
gender	VARCHAR(64)
practitioner_updated_at	DATE
last_audit_update	DATE
data_source	VARCHAR(64)
sanctions	JSON
sanctioned	BOOL
addresses	JSON
emails	JSON
languages	JSON
specialties	JSON
care_categories	VARCHAR(64)
facility_attributes	JSON
practitioner_tags	JSON
external_ids	JSON
active	BOOL
state_licenses	JSON

fax_attest

pract_NPI	BIGINT
last_caller	VARCHAR(32)
phone	VARCHAR(10)
call_issues	VARCHAR(512)
willing_participate	BOOL
call_back	DATETIME
practitioner_name	VARCHAR(45)
pract_by_name	BOOL
fax	VARCHAR(10)
correct_fax	BOOL
provide_correct_fax	VARCHAR(10)
call_quality	VARCHAR(45)
validated	BOOL
validation_date	DATETIME
comments	VARCHAR(255)
call_attempts	INT
status	VARCHAR(45)
call_date	DATETIME

user_records_permissions

user_id	VARCHAR(36)
pract_npi	BIGINT
record_type	VARCHAR(24)

admins

admin_id	VARCHAR(36)

address_attest

pract_NPI	BIGINT
last_caller	VARCHAR(32)
phone	VARCHAR(10)
call_issues	VARCHAR(512)
willing_participate	BOOL
call_back	DATETIME
practitioner_name	VARCHAR(45)
pract_by_name	BOOL
addr1	VARCHAR(64)
addr2	VARCHAR(64)
city	VARCHAR(64)
state	VARCHAR(2)
zip	VARCHAR(9)
correct_address	BOOL
location_name	VARCHAR(128)
provide_addr1	VARCHAR(255)
provide_addr2	VARCHAR(255)
provide_city	VARCHAR(64)
provide_state	VARCHAR(2)
provide_zip	VARCHAR(9)
call_quality	VARCHAR(45)
validated	BOOL
validation_date	DATETIME
comments	VARCHAR(255)
call_attempts	INT
status	VARCHAR(45)
call_date	DATETIME

Fig. 3: Schema for Attested Data

The following Figures describe a functionality map from point A to point B to describe the flow of the product. We used Dia, a structured diagram designing tool, to help communicate different features in the design. In Fig. 4, there is a separate key to show what each shape represents in the following Fig. 5-9. For example, circles indicate the start of a section of each functionality map.

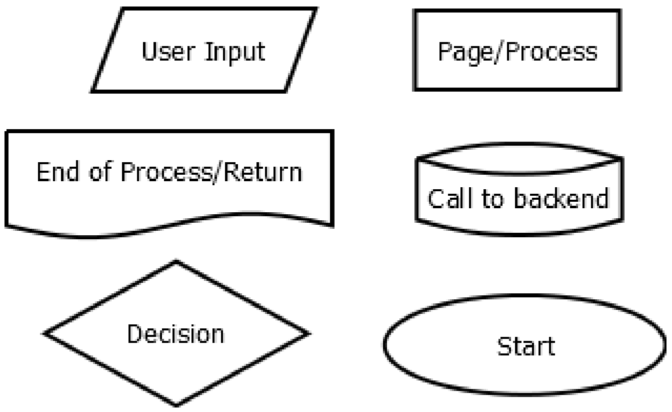


Fig. 4: Key for Fig. 5-9

In Fig. 5, the user has to log in and be authenticated by Google authentication. After that, the user is presented with a dashboard and can do 5 different options: Select Record and Initiate Questionnaire, Get CSV, Share Data, and Filter Records. Fig. 5 has abstracted these options, but their details can be found in Fig. 6-9 respectively.

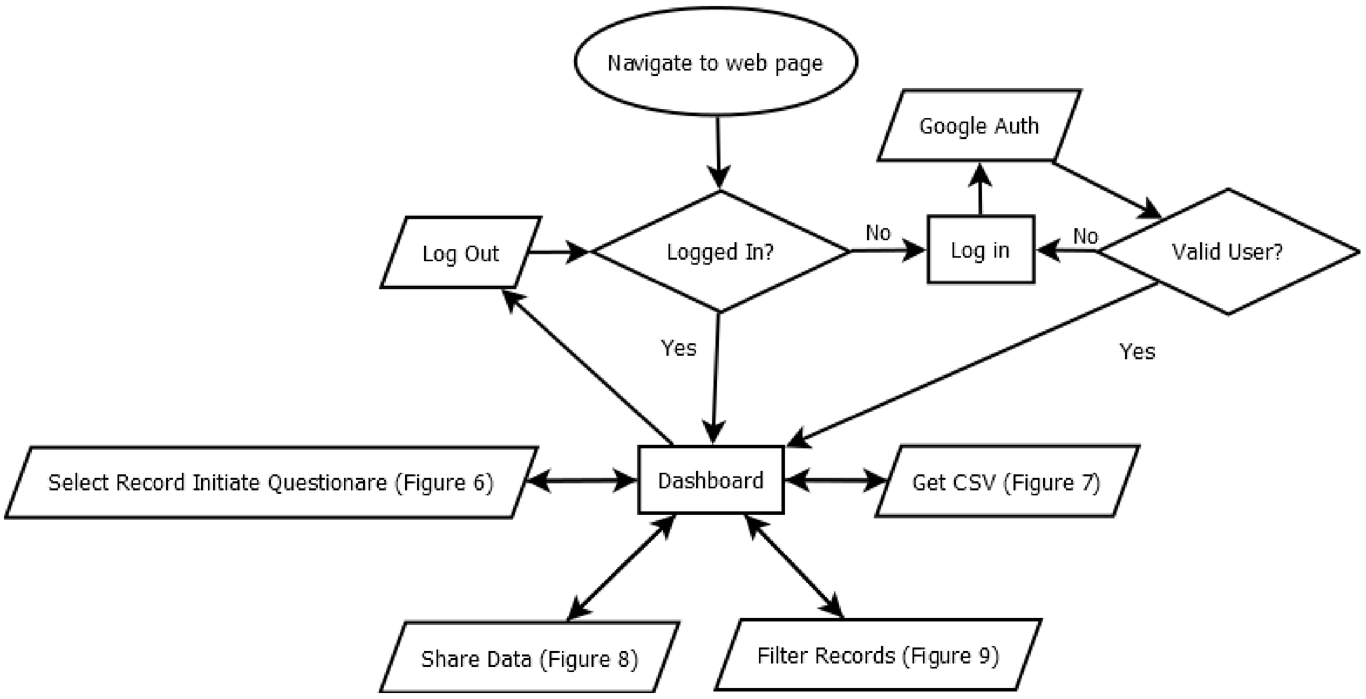


Fig. 5: UI Flow to Login to Dashboard

In Fig. 6, a user can attest a record (practitioner information) after making sure the front end locks the record in Firestore. This is to make sure no one else can edit the data at the same type to prevent duplicates or corrupt data. After that, the data is loaded to the questionnaire from the backend and the user can attest the data by filling out the form. This data is finally sent back to the back end and the Firestore record is unlocked. There is also a sketch of what this looks like in Fig. 16.

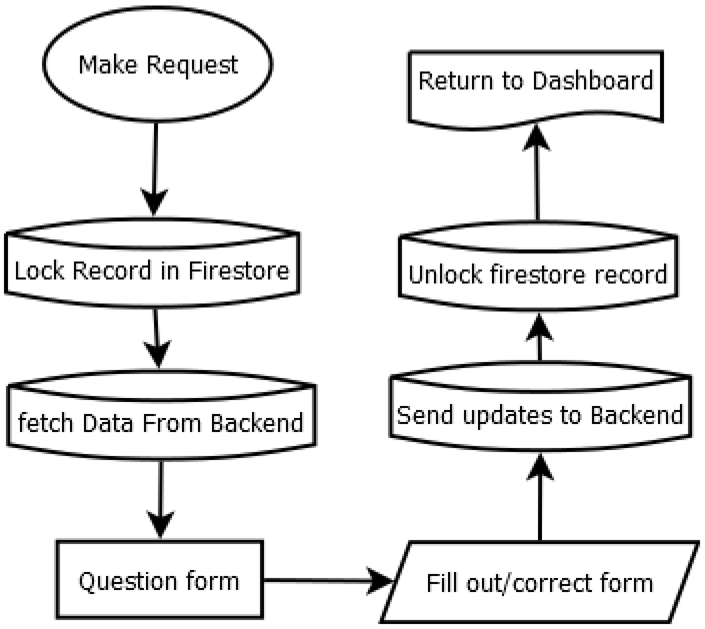


Fig. 6: Questionnaire UI Flow

In Fig. 7, a user can also choose to download a comma-separated values (CSV) file to their local filesystem for data lineage or validation of attestation records. The data from the back end is sent as a JSON file, where the front end converts that data to a CSV file. Finally, the CSV file will be available for download to the user.

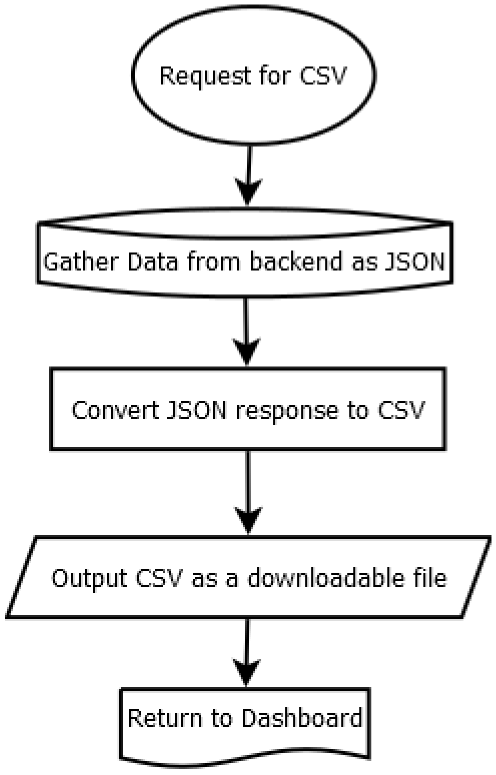

Request for CSV
Gather Data from backend as JSON
Convert JSON response to CSV
Output CSV as a downloadable file
Return to Dashboard

Fig. 7: CSV UI Flow

In Fig. 8, the user can share data with other users and allow them to access that data. The user can filter the shown users after fetching all users from the back-end database. The front end shows all users to the current user and there are options to filter them.

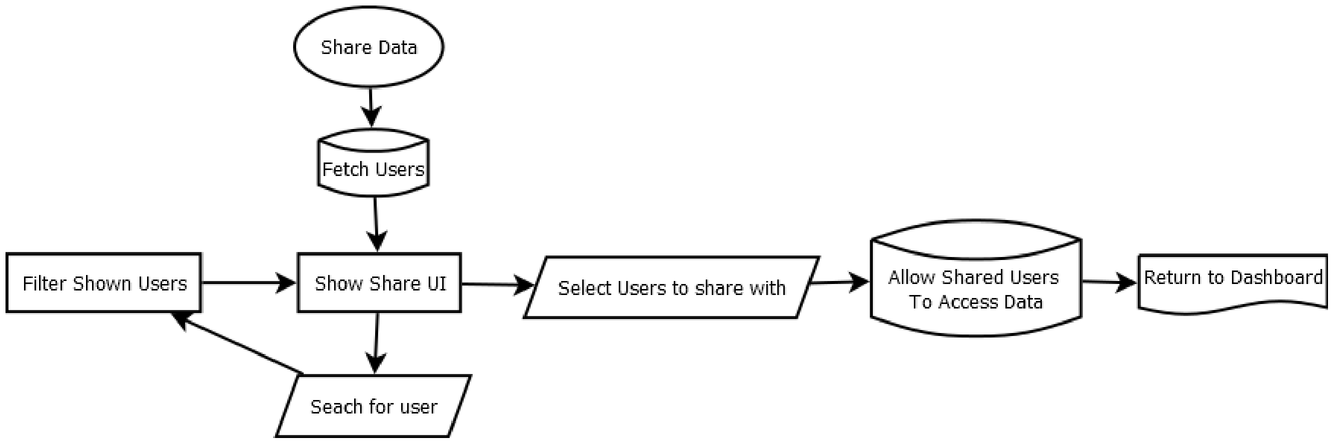


Fig. 8: Share UI Flow

In Fig. 9, the user is able to filter the records by some data attribute of the record. This is to help the user search for a particular record while on the dashboard. The filters array can have many filters at once and, once the user is finished applying filters, the dashboard updates to show filtered records.

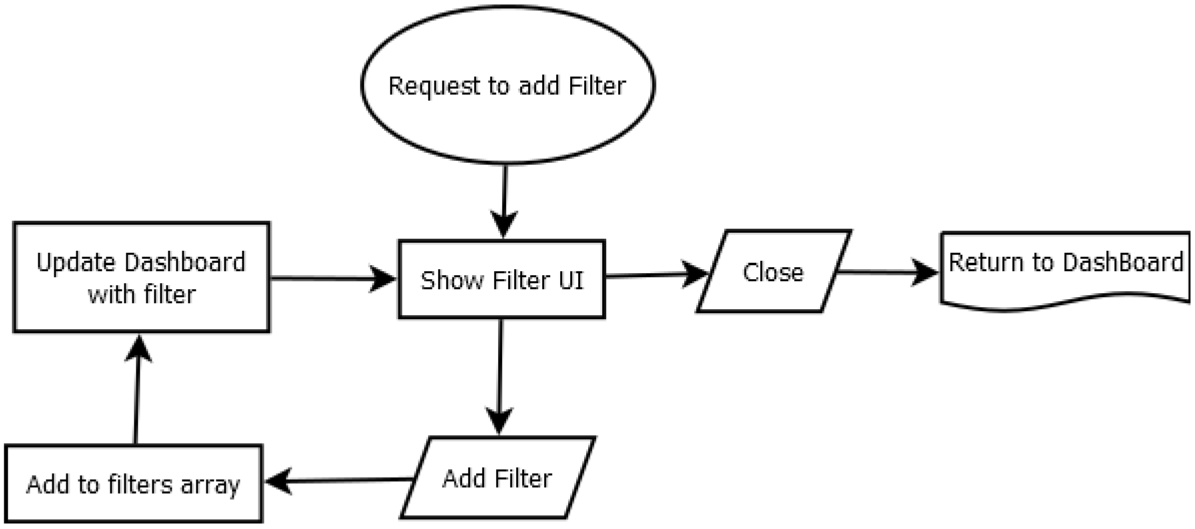


Fig. 9: Filter UI Flow

In the next section, these functionality maps are implemented into the actual UI.

VII. Technical Design

This section focuses on the technical design of the front end because the UI and user experience (UX) is fundamental for any successful web application. In the words of the late Steve Jobs, " Design is not just what it looks like and feels like. Design is how it works" [1]. This section of the report aims to answer the non-functional requirement under product quality. , "data attestors should enjoy working with the UI as much as Google Spreadsheets."

The next 7 Figures are created using Figma which is a software used to design interactive UI's. To convey the interactions between the different screens, the TMs decided to use a website wireframe to map out the main features and navigation seen in Fig. 10. It takes the user from the login screen to the dashboard/dash-open menu where the share popup, filter popup, questionnaire, and download CSV can be reached. These features can be seen in more detail in Fig. 11-16 respectively.

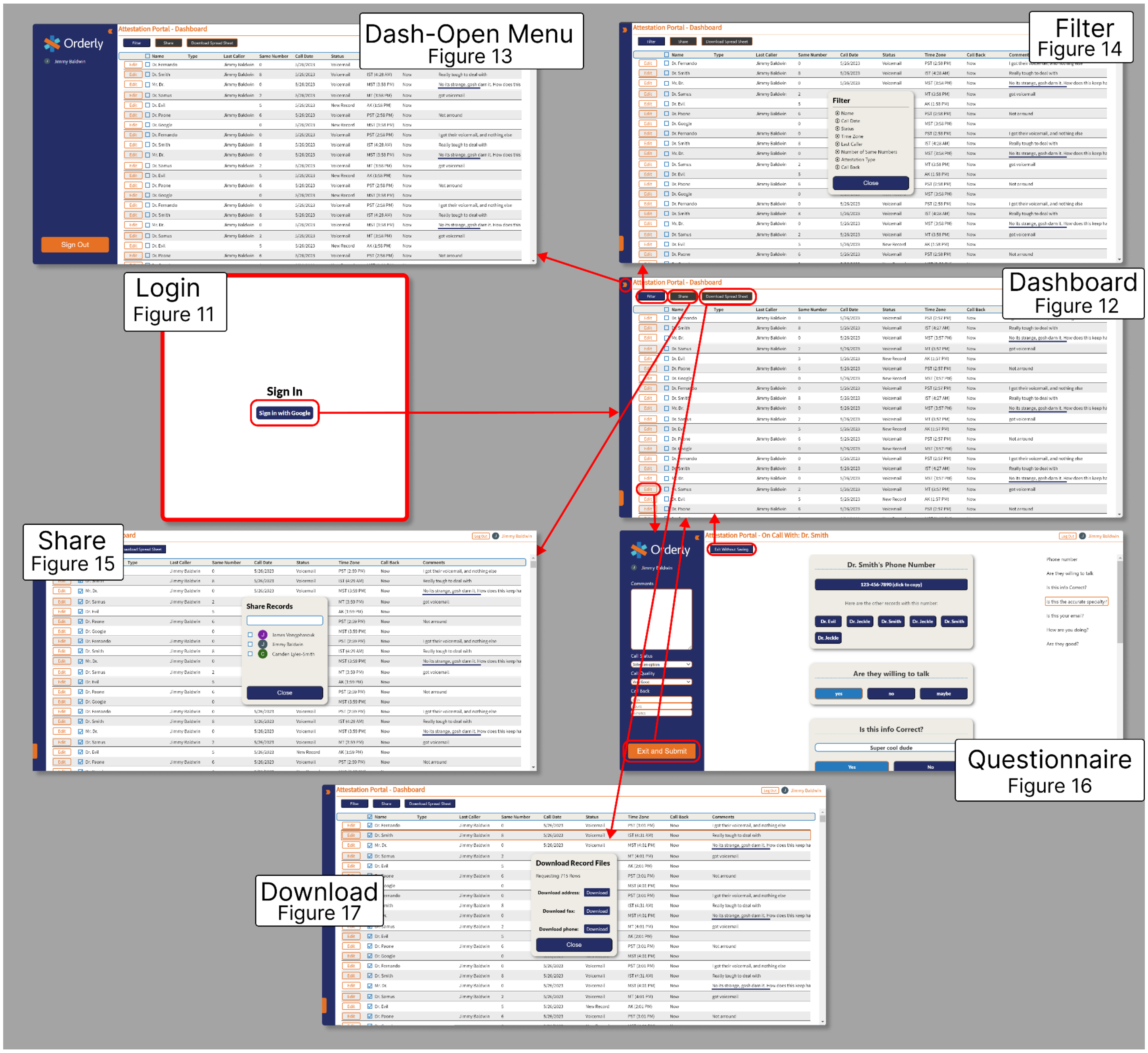


Fig. 10: UI Wireframe

Fig. 11 is a simple login screen for users to sign in with a click of the blue button. The authentication procedure uses Google authentication as a service by Firebase.

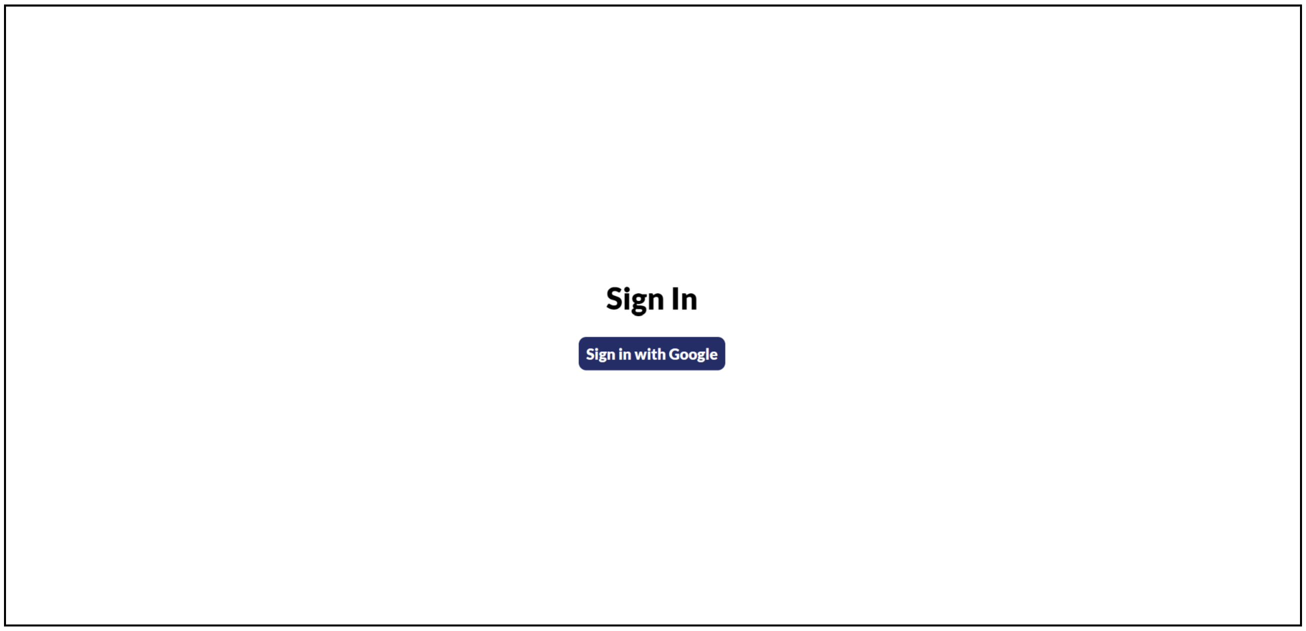
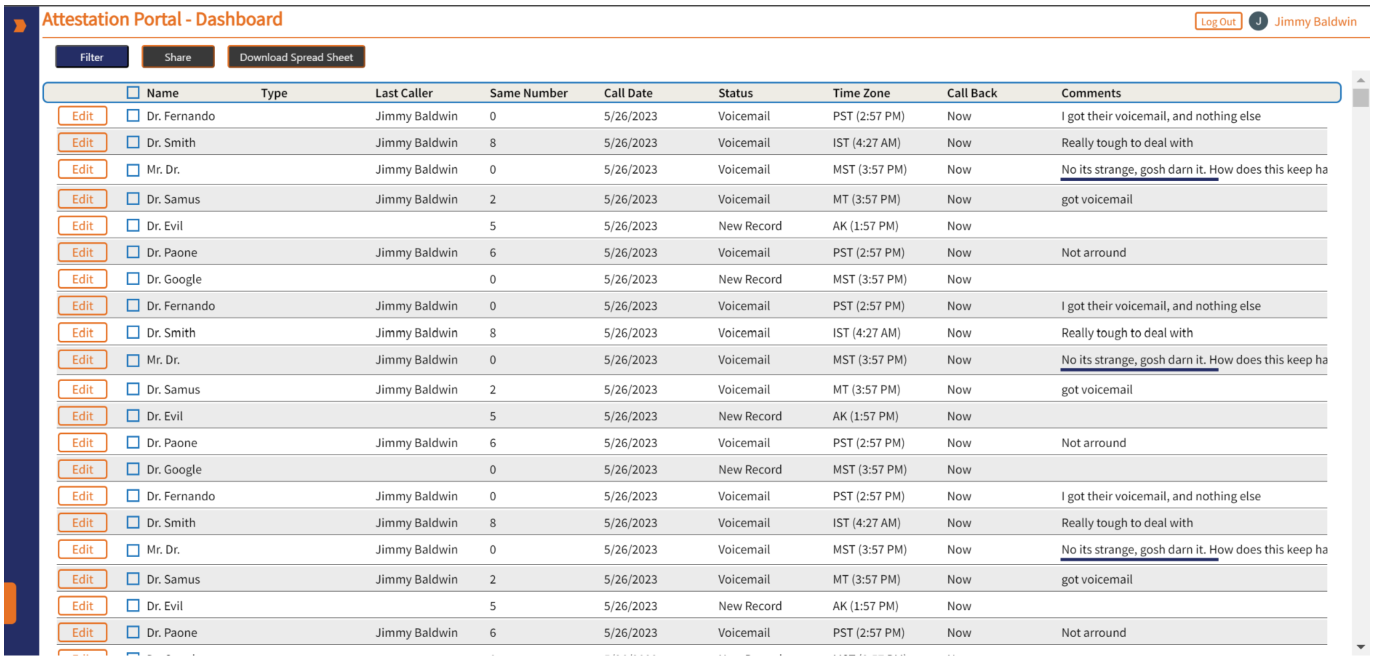


Fig. 11: Login screen

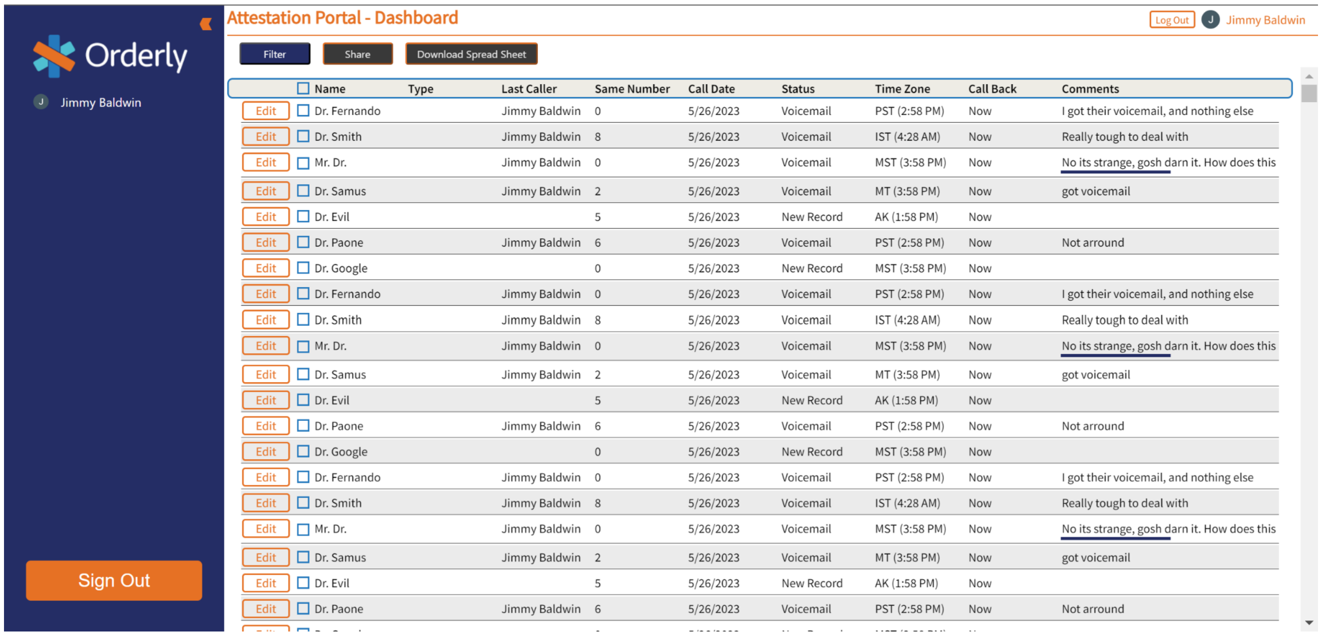
Fig. 12 shows how the dashboard presents the user with all the pertinent information for each of the calls that they are assigned. By presenting the data in a SalesForce “lead” format, the user can glance at all records like a spreadsheet. This also allows the users to jump between records with ease.



	Name	Type	Last Caller	Same Number	Call Date	Status	Time Zone	Call Back	Comments
Edit	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (2:57 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:27 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (3:57 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	Dr. Samus		Jimmy Baldwin	2	5/26/2023	Voicemail	MT (3:57 PM)	Now	got voicemail
Edit	Dr. Evil			5	5/26/2023	New Record	AK (1:57 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6	5/26/2023	Voicemail	PST (2:57 PM)	Now	Not arround
Edit	Dr. Google			0	5/26/2023	New Record	MST (3:57 PM)	Now	
Edit	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (2:57 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:27 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (3:57 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	Dr. Samus		Jimmy Baldwin	2	5/26/2023	Voicemail	MT (3:57 PM)	Now	got voicemail
Edit	Dr. Evil			5	5/26/2023	New Record	AK (1:57 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6	5/26/2023	Voicemail	PST (2:57 PM)	Now	Not arround
Edit	Dr. Google			0	5/26/2023	New Record	MST (3:57 PM)	Now	
Edit	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (2:57 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:27 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (3:57 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	Dr. Samus		Jimmy Baldwin	2	5/26/2023	Voicemail	MT (3:57 PM)	Now	got voicemail
Edit	Dr. Evil			5	5/26/2023	New Record	AK (1:57 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6	5/26/2023	Voicemail	PST (2:57 PM)	Now	Not arround

Fig. 12: Dashboard

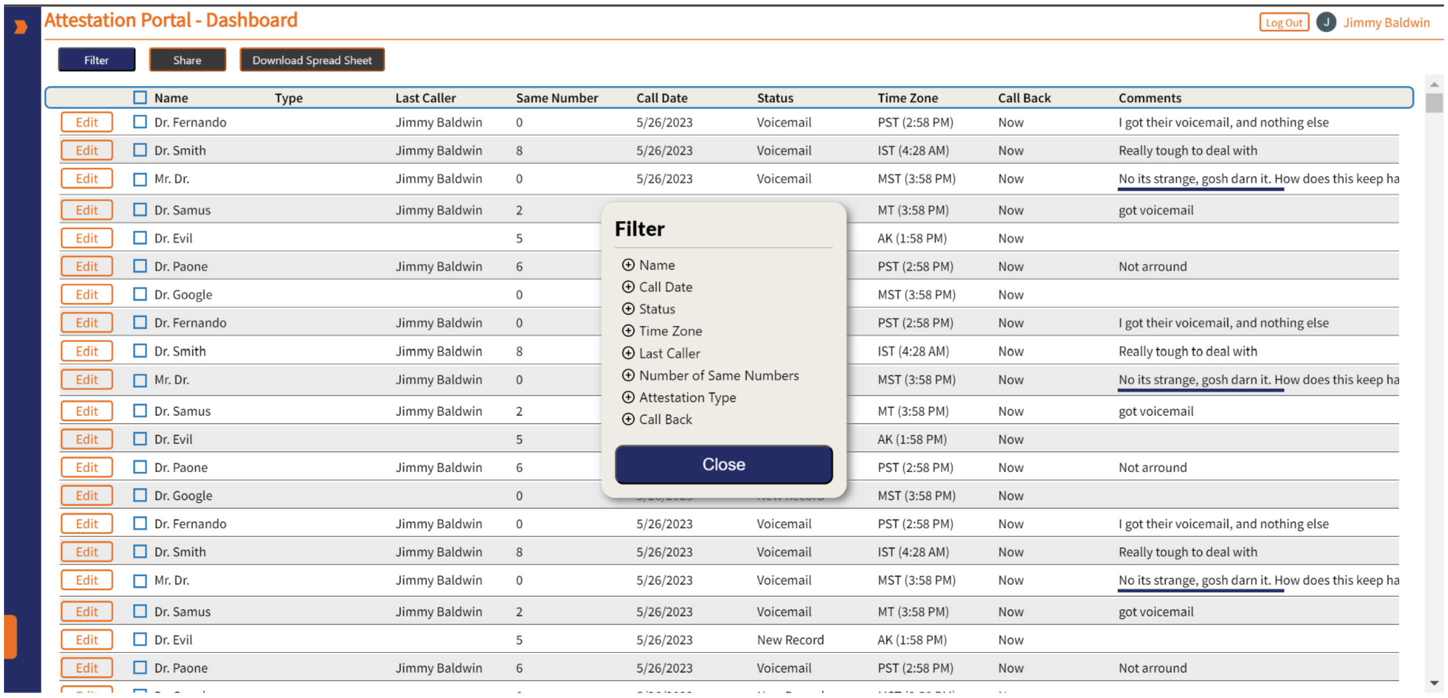
Fig. 13 shows what happens to a user’s dashboard when the user clicks the orange arrow in the top left corner. It shows a side panel view where a user can sign out.



	Name	Type	Last Caller	Same Number	Call Date	Status	Time Zone	Call Back	Comments
Edit	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (2:58 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:28 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (3:58 PM)	Now	No its strange, gosh darn it. How does this
Edit	Dr. Samus		Jimmy Baldwin	2	5/26/2023	Voicemail	MT (3:58 PM)	Now	got voicemail
Edit	Dr. Evil			5	5/26/2023	New Record	AK (1:58 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6	5/26/2023	Voicemail	PST (2:58 PM)	Now	Not arround
Edit	Dr. Google			0	5/26/2023	New Record	MST (3:58 PM)	Now	
Edit	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (2:58 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:28 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (3:58 PM)	Now	No its strange, gosh darn it. How does this
Edit	Dr. Samus		Jimmy Baldwin	2	5/26/2023	Voicemail	MT (3:58 PM)	Now	got voicemail
Edit	Dr. Evil			5	5/26/2023	New Record	AK (1:58 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6	5/26/2023	Voicemail	PST (2:58 PM)	Now	Not arround
Edit	Dr. Google			0	5/26/2023	New Record	MST (3:58 PM)	Now	
Edit	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (2:58 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:28 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (3:58 PM)	Now	No its strange, gosh darn it. How does this
Edit	Dr. Samus		Jimmy Baldwin	2	5/26/2023	Voicemail	MT (3:58 PM)	Now	got voicemail
Edit	Dr. Evil			5	5/26/2023	New Record	AK (1:58 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6	5/26/2023	Voicemail	PST (2:58 PM)	Now	Not arround

Fig. 13: Dash-Open Menu

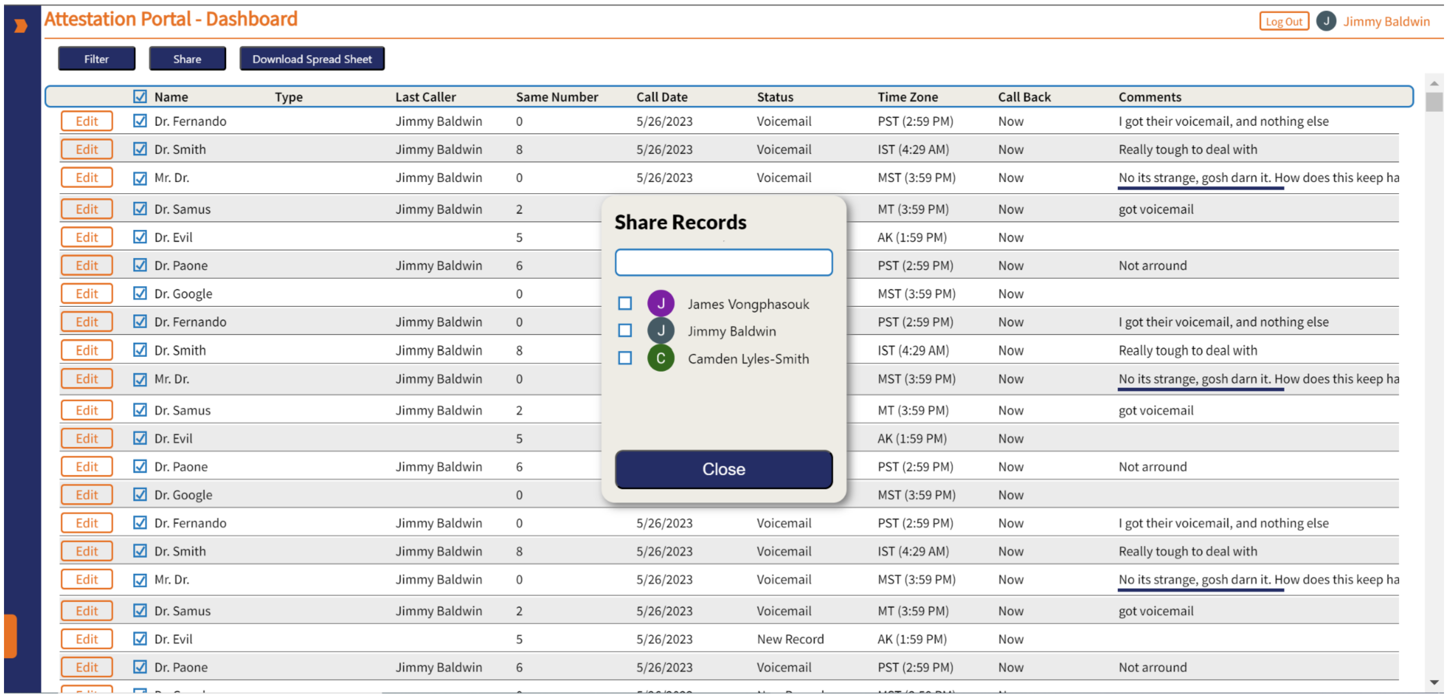
In Fig. 14, the user can see this popup when clicking the “filter” text. Some initial filters to help users comb through the information are name, call date, status, etc. Once the filters are inputted, the user can click the close button, and the dashboard is appropriately updated.



	Name	Type	Last Caller	Same Number	Call Date	Status	Time Zone	Call Back	Comments
Edit	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (2:58 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:28 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (3:58 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	Dr. Samus		Jimmy Baldwin	2			MT (3:58 PM)	Now	got voicemail
Edit	Dr. Evil			5			AK (1:58 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6			PST (2:58 PM)	Now	Not arround
Edit	Dr. Google			0			MST (3:58 PM)	Now	
Edit	Dr. Fernando		Jimmy Baldwin	0			PST (2:58 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8			IST (4:28 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0			MST (3:58 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	Dr. Samus		Jimmy Baldwin	2			MT (3:58 PM)	Now	got voicemail
Edit	Dr. Evil			5			AK (1:58 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6			PST (2:58 PM)	Now	Not arround
Edit	Dr. Google			0			MST (3:58 PM)	Now	
Edit	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (2:58 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:28 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (3:58 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	Dr. Samus		Jimmy Baldwin	2	5/26/2023	Voicemail	MT (3:58 PM)	Now	got voicemail
Edit	Dr. Evil			5	5/26/2023	New Record	AK (1:58 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6	5/26/2023	Voicemail	PST (2:58 PM)	Now	Not arround

Fig. 14: Filter Popup

A user can check the boxes next to the records that a user wants to share with other users. Once a user checks at least one box, the user can click on the share button. After it is clicked, the user sees a popup such as in Fig. 15. Next, a user can check the boxes next to the users that the user wants to share. Finally, the user can hit the “close” button to share the checked records.


Attestation Portal - Dashboard

Log Out | Jimmy Baldwin

Filter | Share | Download Spread Sheet

		Name	Type	Last Caller	Same Number	Call Date	Status	Time Zone	Call Back	Comments
Edit	☑	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (2:59 PM)	Now	I got their voicemail, and nothing else
Edit	☑	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:29 AM)	Now	Really tough to deal with
Edit	☑	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (3:59 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	☑	Dr. Samus		Jimmy Baldwin	2			MT (3:59 PM)	Now	got voicemail
Edit	☑	Dr. Evil			5			AK (1:59 PM)	Now	
Edit	☑	Dr. Paone		Jimmy Baldwin	6			PST (2:59 PM)	Now	Not arround
Edit	☑	Dr. Google			0			MST (3:59 PM)	Now	
Edit	☑	Dr. Fernando		Jimmy Baldwin	0			PST (2:59 PM)	Now	I got their voicemail, and nothing else
Edit	☑	Dr. Smith		Jimmy Baldwin	8			IST (4:29 AM)	Now	Really tough to deal with
Edit	☑	Mr. Dr.		Jimmy Baldwin	0			MST (3:59 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	☑	Dr. Samus		Jimmy Baldwin	2			MT (3:59 PM)	Now	got voicemail
Edit	☑	Dr. Evil			5			AK (1:59 PM)	Now	
Edit	☑	Dr. Paone		Jimmy Baldwin	6			PST (2:59 PM)	Now	Not arround
Edit	☑	Dr. Google			0			MST (3:59 PM)	Now	
Edit	☑	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (2:59 PM)	Now	I got their voicemail, and nothing else
Edit	☑	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:29 AM)	Now	Really tough to deal with
Edit	☑	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (3:59 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	☑	Dr. Samus		Jimmy Baldwin	2	5/26/2023	Voicemail	MT (3:59 PM)	Now	got voicemail
Edit	☑	Dr. Evil			5	5/26/2023	New Record	AK (1:59 PM)	Now	
Edit	☑	Dr. Paone		Jimmy Baldwin	6	5/26/2023	Voicemail	PST (2:59 PM)	Now	Not arround

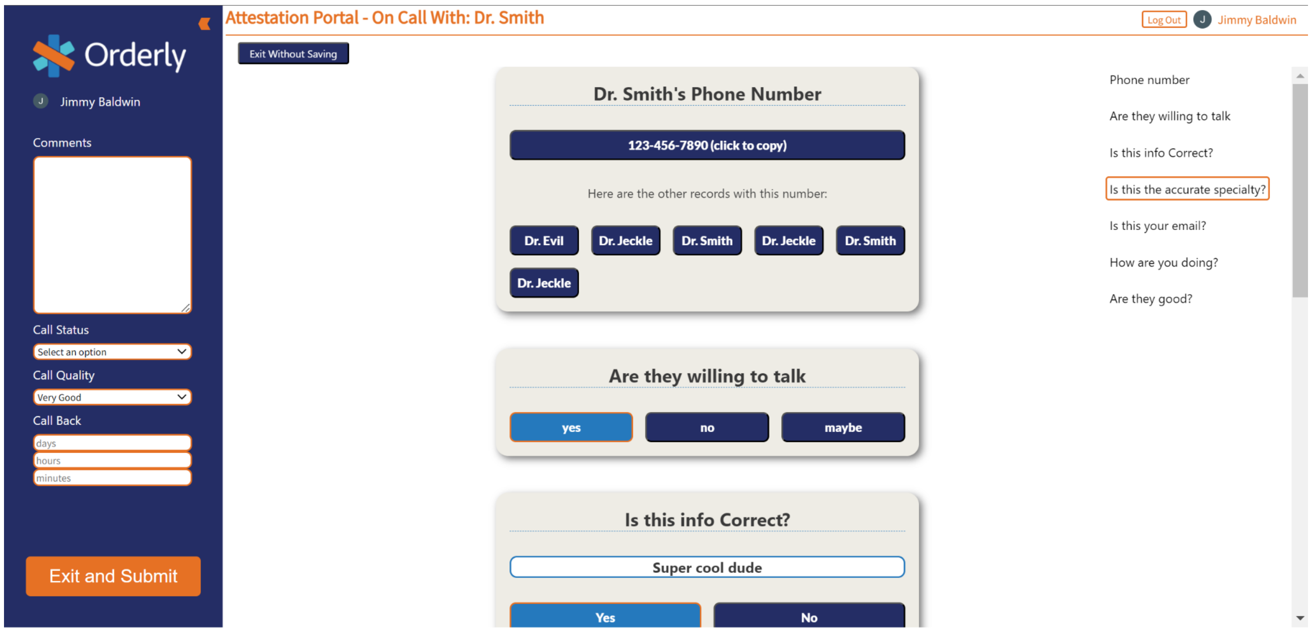
Share Records

- ☐ J James Vongphasouk
- ☐ J Jimmy Baldwin
- ☐ C Camden Lyles-Smith

Close

Fig. 15: Share Popup

Fig. 16 shows the questionnaire form which mirrors the script of the attestors. From here, the attestors can validate all information gathered. This format allows for an easy workflow, and allows for the data gathered on calls to be uniformly formatted and validated to the backend.


Orderly

Jimmy Baldwin

Attestation Portal - On Call With: Dr. Smith

Log Out | Jimmy Baldwin

Exit Without Saving

Comments

Call Status: Select an option

Call Quality: Very Good

Call Back: days / hours / minutes

Exit and Submit

Dr. Smith's Phone Number

123-456-7890 (click to copy)

Here are the other records with this number:

Dr. Evil | Dr. Jeckle | Dr. Smith | Dr. Jeckle | Dr. Smith | Dr. Jeckle

Are they willing to talk

yes | no | maybe

Is this info Correct?

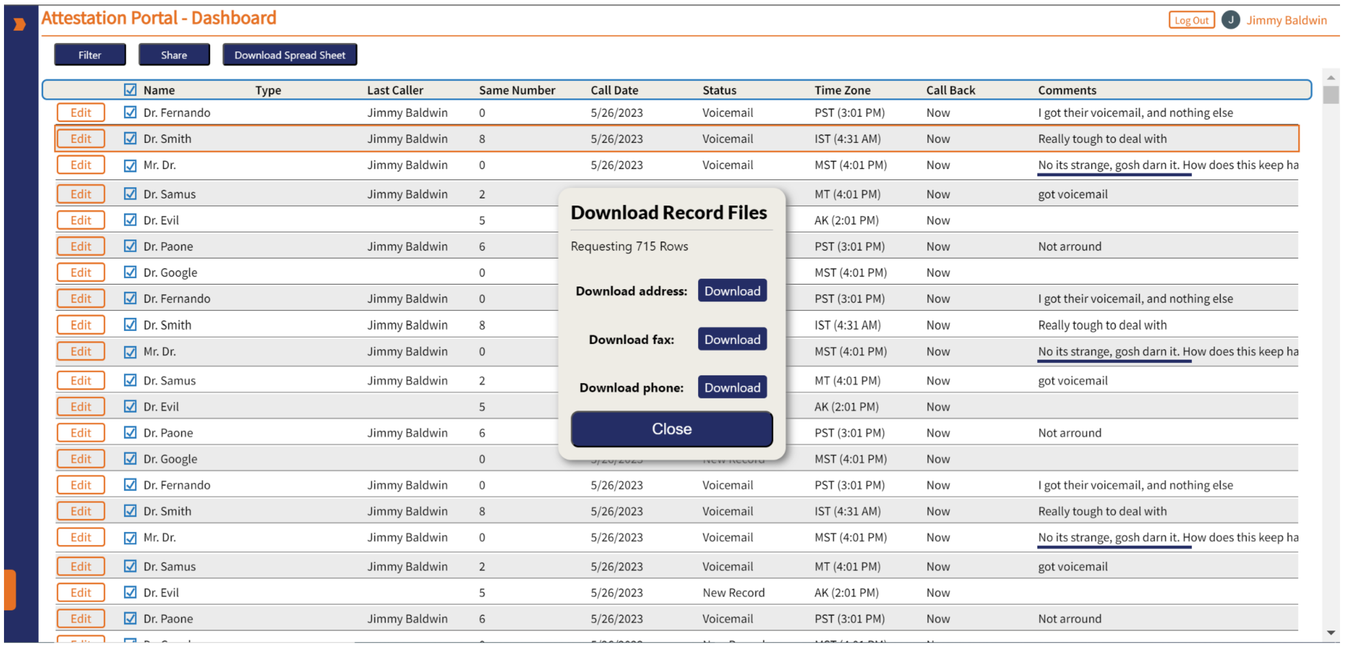
Super cool dude

Yes | No

Phone number
Are they willing to talk
Is this info Correct?
Is this the accurate specialty?
Is this your email?
How are you doing?
Are they good?

Fig. 16: Questionnaire Screen

The last button the user can click on is the Download Spreadsheet button. Upon clicking, a user is presented with Fig. 17 and can download address, fax, or phone attestation records as a CSV file for further validation. There is currently no button to upload the validated attestation records, but it is noted in the Future Work section.



	Name	Type	Last Caller	Same Number	Call Date	Status	Time Zone	Call Back	Comments
Edit	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (3:01 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:31 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (4:01 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	Dr. Samus		Jimmy Baldwin	2			MT (4:01 PM)	Now	got voicemail
Edit	Dr. Evil			5			AK (2:01 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6			PST (3:01 PM)	Now	Not arround
Edit	Dr. Google			0			MST (4:01 PM)	Now	
Edit	Dr. Fernando		Jimmy Baldwin	0			PST (3:01 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8			IST (4:31 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0			MST (4:01 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	Dr. Samus		Jimmy Baldwin	2			MT (4:01 PM)	Now	got voicemail
Edit	Dr. Evil			5			AK (2:01 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6			PST (3:01 PM)	Now	Not arround
Edit	Dr. Google			0			MST (4:01 PM)	Now	
Edit	Dr. Fernando		Jimmy Baldwin	0	5/26/2023	Voicemail	PST (3:01 PM)	Now	I got their voicemail, and nothing else
Edit	Dr. Smith		Jimmy Baldwin	8	5/26/2023	Voicemail	IST (4:31 AM)	Now	Really tough to deal with
Edit	Mr. Dr.		Jimmy Baldwin	0	5/26/2023	Voicemail	MST (4:01 PM)	Now	No its strange, gosh darn it. How does this keep ha
Edit	Dr. Samus		Jimmy Baldwin	2	5/26/2023	Voicemail	MT (4:01 PM)	Now	got voicemail
Edit	Dr. Evil			5	5/26/2023	New Record	AK (2:01 PM)	Now	
Edit	Dr. Paone		Jimmy Baldwin	6	5/26/2023	Voicemail	PST (3:01 PM)	Now	Not arround

Fig. 17: Download Record Files as CSV File

VIII. Software Test and Quality

Testing of the software is integral in ensuring the product's quality, and delivering a well-rounded, completed product to the customer at the conclusion of the project. Tests that are written and completed before product delivery fall into three main requirement-based categories: testing of front end features and functionality, backend data pipeline testing, and security testing.

Front end tests mainly consist of following the process of a data attestors through an attestation to ensure each feature functions properly and the flow of the application allows the agent to complete the attestation similarly to how they currently complete an attestation through the spreadsheet.

Testing of the back end data pipeline uses JUnit and integration testing to check values being passed through the pipeline at each stage to ensure both its integrity and accuracy through the data acquisition and upload process.

From the functional and non-functional requirements, the TMs have created tests to ensure they are met in table 4. Some of the requirements are not necessarily testable such as "There must be documentation on the process to load the data to be attested." Our main categories of testing are Manual UI/Command Line tests and code reviews.

Manual UI tests are to ensure that the product's UI is friendly and functional, Unit/Integration tests are to ensure that the code is bug free and handles edge cases of data, and code reviews are to keep the code from smelling to Orderly’s engineering team. Note that these tests are not exhaustive, but helps to improve the product as it gets closer to the definition of done.

Table 4: Product Tests

Test name	**Category**	**Addressed Requirement/Quality**	**Setup**	**Action(s)**	**Expected Result**
Invalid Email Sign in	Manual UI Testing	data attestors must be able to sign in to keep track of who is attesting what data. Having a valid sign in process also ensures basic security of data.	The front end is running and reachable from the user’s computer.	1. User clicks on the sign in button. 2. User is redirected to Google Sign in and signs with invalid email address.	User is redirected back to the sign in page and is presented with, “Error: Must use a valid email address within this organization.”
Nonlinear Edit Record	Manual UI Testing	The data attestation user interface (UI) must allow data attestors to attest 4 points of data associated with the practitioner: ● Primary Specialty ● Fax ● Address ● Primary Specialty Editing a record should also be as clear/nonlinear as editing a cell in Google Spreadsheets and it should involve no sharing of permissions or data through email.	The front end is running and reachable from the user's computer and the back-end REST server is reachable from the front-end.	1. User signs into dashboard view. 2. User clicks on the “edit record” text for a record. 3. User sees the correct information for that record. 4. User attests the information displayed by inputting into form nonlinearly. 5. When the user finishes attesting, the user clicks the “exit and submit” button.	User is able to finish the call and attest the information even if the call does not follow the left-right or top-down question flow.
(near) Real-time Validation	Manual UI Testing	Addresses, phone numbers, and zip code entries must be validated in (near) real time on the portal.	The front end is running and reachable from the user's computer and the back-end	1. User signs into dashboard view. 2. User clicks on the “edit record” text for a record. 3. User sees the correct	When User gets to attesting addresses, phone numbers, and zip code entries, the information is

			REST server is reachable from the front-end.	information for that record. 4. User starts attesting the address, phone number, and zip code entries.	validated in (near) real-time.
Locked Records	Manual UI Testing	There must be synchronization for showing which calls are being taken, and which call is being handled by which data attestors agent. This means that the user data should be accessible at near-real time of us	The front end is running and reachable from 2 users' computers.	1. Two users sign into the dashboard view. 2. User one clicks on the "edit record" text for a record. 3. User two attempts to click on the same "edit record" text for the same record.	User two is not able to access the record until User one has done one of x options: ● clicked "exit and submit" button ● closes the window ● back arrow button ● select a new record
Correct data sent from back end to front end and vice versa	Manual UI Testing	Local database stores data (attestation entries) from the front-end.	The front end is running and reachable from the user's computer and the back-end REST server is reachable from the front-end.	1. User signs in and waits for the dashboard view to load. 2. front end requests records from records/dashboards endpoint. 3. Backend sends the records to display. 4. User edits the first record and hits the "exit and submit button"	Dashboard view should display all backend data that is not null at step 3. After step 4, User should see his edited record in the dashboard view.

Data to attest can be uploaded to web portal	Manual Developer Command Line Test	Orderly must be able to put together a new attestation document every month for data attestors using the data stored in BigQuery and gathered from the data attestation UI.	The front end is running and reachable from the user's computer and the back-end REST server is reachable from the front-end. Developer can import data using the PopulateDatabase.java file	1. All users logout of web portal 2. Developer runs the cloud-sql-proxy with correct parameters 3. Developer runs the PopulateDatabase.java file to extract table data to a bucket as a newline delimited JSON file, download the JSON file to their local filesystem and upload that file to GCS MySQL. 4. Developer runs the back end. 5. Developer starts up the front-end. 6. Users sign in and see the data that a developer uploaded.	Developers should be able to extract table data into a GCS bucket as a newline delimited JSON object. From the GCS bucket, the developer should be able to download the JSON object onto their local filesystem and into GCS MySQL where the backend can display the dashboard data.
Data that is attested can download into CSV for validation OR can be downloaded from GCS	Manual UI Test	Orderly must be able to put together a new attestation document every month for data attestors using the data stored in BigQuery and gathered from the data attestation UI.	The front end is running and reachable from the user's computer and the back-end REST server is reachable from the front-end. Connection to MySQL exists for backend	1. All users are done attesting. 2. User clicks checks boxes of records that the user wants to be downloaded into a spreadsheet. 3. User hits the download button next to the attestation type that the user desires. OR	A user can export the rows that were checked off to be downloaded to a CSV file. A developer can export the data from a MySQL instance into a GCS bucket for machine learning.

				1. All users are done attesting. 2. Developer logs in to GCP and navigates to SQL instances 3. Developer clicks on "field-session-db" which is the MySQL database. 4. Developer clicks on the export icon. 5. Developer specifies format, data to export, and destination GCS bucket, and clicks the export button. 6. Developer navigates to the GCS bucket by typing cloud storage in the search bar and clicking on the bucket with the exported data. 7. Developer clicks on the download icon in the same row as their exported data.	
Code Reviews	Code Quality	There must be documentation on the process to load the data to be attested. The improved attestation (front end and back end) system must work in Orderly's system.	Latest code is accessible to all TMs through the GitHub portfolio.	1. One TM will write code for a feature. 2. Another TM will review code for comments, logic, and readability.	Code is maintainable / readable / bug free for Orderly's engineering team.

The TMs decided to omit the unit tests from this table as they are testing basic functionalities that are expected from a working product. It is important to note that these tests lie primarily in the backend and are implemented using JUnit testing framework. The results of these tests and the tests in the table are in the next section. The TMs did not have time to thoroughly conduct integration testing, but the many manual UI tests achieved the same effect because it would be obvious when the data passed through our product was incorrect.

IX. Project Ethical Considerations

The abundance of PII has become a goldmine for both individuals and malicious actors alike. PII encompasses a wide range of sensitive data that, when mishandled, can have far-reaching consequences. There are ACM and IEEE principles that are relevant in the delivery of our product. Then, the TMs consider Michael Davis questions as it relates to our product. Finally, the TMs consider the ethical considerations for the project if the software quality plan is not enough.

In the ACM Code of Ethics, principle 1.6 states, "... allow individuals to understand what data is being collected, … , to give informed consent for automatic data collection, …" [2]. This relates to PII as these providers have their information publicly available, but, as Orderly streamlines it, the provider may experience unwanted privacy breach. For example, a provider may list their personal phone number as their contact information and they are at risk of more people getting access to it. Also, in the ACM Code of Ethics, principle 1.7 states "... protect confidentiality except in cases where it is evidence of the violation of the law, … information should not be disclosed except to appropriate authorities." [2] . Similarly with privacy, confidentiality of the data is important so that malicious parties do not obtain the data. Since our product is using Google authentication to authenticate data attestors, these attestors need to provide the product with their Google email accounts. Our product should keep this information confidential.

Additionally, in the IEEE Code of Ethics, principle 1.03 states, "Approve software only if it … does not … diminish privacy" [3]. Our product which handles the direct privacy and confidentiality of PII data stores the data directly in Orderly's MySQL database. Only Orderly is allowed to access the portal and view the PII, so our product does not diminish privacy.

The team thinks that the Harm Test and the Common Practice Test are the most appropriate when handling PII. The Harm Test states, "does this option do less harm than any alternative? Do the benefits outweigh the harms?" [4]. Our product which eliminates the need for email exchange of PII is better than the current alternative because the current alternative uses emails and spreadsheets to exchange PII. The TMs know that this project is better than the alternative because the product securely exchanges PII in the cloud. The TMs also conclude no foreseeable harms that outweigh the benefits. The Common Practice Test states, "What if everyone behaved this way?" [4]. Our product uses a friendly UI to input user info which the TMs believe is what every software engineer would do.

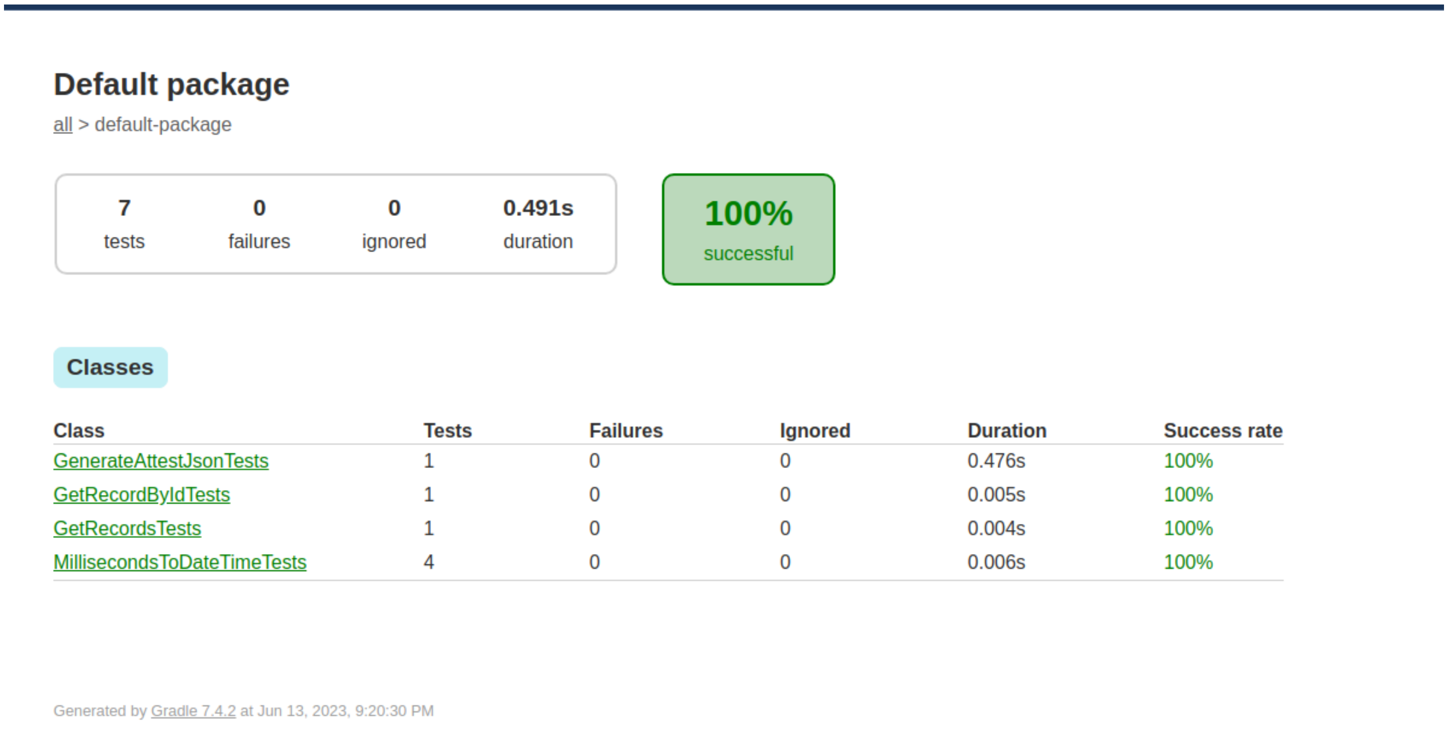
If the software quality plan tests fail, then there are two major risks. PII can be compromised and Orderly Health Google email accounts are leaked. The team sees these as preventable risks as long as we make sure to have code that prioritizes security and privacy. We also need to make sure that our code passes the tests in Table 4 as they ensure functional requirements are met.

X. Results

Our product passed each of the manual UI tests that were listed in Table 4. However, there is much more testing to be done in terms of automated UI tests. Cypress, an automated JavaScript testing framework, is one area that can improve the quality of our front-end functionalities. It can query for elements on a page with a .contains() which can be very helpful for how many elements the dashboard view contains and it can also make sure that a DOM (document object model) element is clickable with .click().

The manual developer command line test was also passed by our product which hopefully makes it easier for Orderly to upload their attestation data to the web portal for data attestors to attest it every month. The documentation of the data pipeline and its usage is a hard requirement so that Orderly can adjust the data pipeline for different kinds of attestation.

As for the unit tests, Fig. 18 documents their results. The product used Gradle, a build automation tool, to run the web portal and its tests. It also came with a convenient summary display to show the names of the test classes, number of tests, number of failed tests, number of ignored tests, duration of testing time, and a success rate for that test class. For example, the MillisecondsToDateTimeTests class had 4 tests that passed and took 0.006 seconds to complete.



Default package

all > default-package

7	0	0	0.491s	100%
tests	failures	ignored	duration	successful

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
GenerateAttestJsonTests	1	0	0	0.476s	100%
GetRecordByIdTests	1	0	0	0.005s	100%
GetRecordsTests	1	0	0	0.004s	100%
MillisecondsToDateTimeTests	4	0	0	0.006s	100%

Generated by Gradle 7.4.2 at Jun 13, 2023, 9:20:30 PM

Fig. 18: Back-end Testing Results

As for code reviews, the TMs reviewed each other's code as discussed in Table 4 and commented confusingly nested for loops, truncated long one-liners, and refactored repeated code into their own classes. This process made the product contain self-documenting code, as well as maintainable code for Orderly's Engineering Team to take over.

XI. Future Work

All of the hard functional and nonfunctional requirements were passed, but there were some nice-to-haves that came up during the development process that the TMs could not get to with the time allotted. Given that Orderly has their own engineering team, they can possibly work out some of these nice-to-haves. These include an admin panel/view for Orderly engineers to enable/disable certain features depending on the user, a UI overhaul that would make the product read more in a left-to-right direction, in-depth security of user credentials, backups of MySQL database, live users in the side panel, and user statistics. These components can be easy or medium difficulty to implement since most of these are front-end fixes in JavaScript React Framework. Having backups of a database is a GCP issue where Orderly is well-versed in. The team regularly asked Orderly's engineers for assistance regarding GCP, so any GCP addons or tools that can replace our components should also be considered as future work.

In terms of components that can be improved, we talked to one of Orderly's engineers that work on the backend and their main development language is in Python. The product's backend is built on Java SpringBoot Framework that works well as is, but the team would like to make it as easy as possible for Orderly's engineers to deploy our web portal. Django, Flask, or FastAPI are all great Python backend frameworks that can replace our Java backend. The team's recommendation is Flask since it is easy to start, well-documented, and popular so that StackOverflow answers can be found. The front-end framework used can be swapped for other frameworks as needed, but adding React components would be easiest if adding more features.

Last but not least, the feature to upload the validated attestation records back to the web portal for storage in the MySQL database was not implemented. The attestation results are stored in MySQL and available for download using the Download Spreadsheet button, but the validation columns are not. This leaves data attestors to continue using Excel software for their validation. Alex, a representative for data attestors mentioned that he would like to keep using that software to validate the data. Orderly can implement a way for data attestors to upload the CSV such that the web portal parses the CSV and updates the MySQL database based on the validation.

XII. Lessons Learned

Many lessons such as communicating our needs to the client and each other, delivering quality work, and following an Agile development process were experienced during the duration of this project. All of the lessons have one common crux and that is, to develop early, but ask questions even earlier because most of the problems stemmed from not asking enough questions when the TMs had the chance.

For example, the TMs knew they were going to develop locally on our machines and not on GCP to make it easier for all of us since the TMs know their own development environments. However, it is much harder to deploy the product to the client since GCP has a lot of dependencies that the TMs did not consider. Another example is when the TMs were implementing the backend infrastructure in Java and found out from one of their engineers later that Python was the main language of choice for their backend. The TMs had assumed that Python was not one of the options because the client had developed in Java prior, so the TMs used a rather difficult language when the TMs could have used a higher level language or framework.

The TMs also learned that testing is very hard to do at the last minute as developing tests can sometimes be even harder than developing the actual code. Test Driven Development (TDD) was recommended to us by our Field Session lectures and advisors, but the TMs were too excited to just start coding. If the TMs created failing tests to go with the features, the TMs could ensure quality much easier.

The two main takeaways from the many lessons learned are:

- TMs need to ask more questions before coding to ensure a smooth development process and deployment process
- TMs need to do TDD to ensure quality every step of the development process

XIII. Acknowledgments

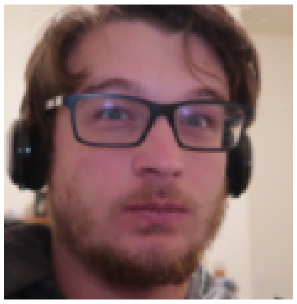
Dr. Paone is a Professor at the Colorado School of Mines that helped us facilitate ideas, learn the important elements of software development in the field, and helped finalize this report document. Thanks to Dr. Paone for giving up half his summer to progress in our academic careers. We learned a multitude of Agile methodologies, ACM/IEEE ethical principles, software quality insurances, and overall software engineering advice from Dr. Paone.

Mark Barkmeier is one of the clients from Orderly Health that also worked as a software engineer. Mark answered all of our random questions throughout the project and connected us to the rest of the Orderly team. Daniel Lockman is the other client and Vice President of the Engineering team at Orderly Health that helped advise our project and facilitate demos with the company. He also helped us connect with the rest of the company and gave feedback on our UI. Thanks to the clients for allowing us to learn from their tech stack and professionally develop ourselves.

Another thanks to some of the engineers, managers, and designers who also gave us feedback on our product and taught us software development skills which includes Sylvester Meighan, Micahel Cunha, Arielle Kahn, and Megan Hart.

One final special thanks to Alex Wynn who was the data attestors representative that helped determine how the end UI of the product would look since his agents would be the ones using our product.

XIV. Team Profile



Name: Jimmy Baldwin

Year: Senior

Discipline: Web Development

Hometown: Incline Village, NV

Work Experience: Freelancing Web Development, Marketing Web Development

Hobbies: Minecraft, Skiing, Painting, Playing Music

I wanted to build something that would be used to make an impact in people's lives. One of the best ways to do that is through building out health care products. I love web development, so I can not wait to get started!

Name: Camden Lyles-Smith

Year: Senior

Discipline: Cyber Security

Hometown: Denver, CO

Work experience: landscaping, Whole Foods E-commerce, Stagehand, Certol Warehouse

Hobbies: Hiking, camping, playing guitar, working on cars

I'm looking forward to working with the team to help create a more streamlined system for Orderly Health's data attestation pipeline and working more with web applications!

Name: James Vongphasouk

Year: Senior

Discipline: Data Science

Hometown: Thornton, CO

Work Experience: Data Analytics, Data Engineering, Home Depot, Amazon Warehouse

Hobbies: video games and learning Natural Language Processing

I like health data and Orderly Health has a ton of it, so I'm very excited. Also I have no web development experience, so this will be fun to be outside my comfort zone!

XV. References

[1] P. Gangadharan, “The importance of User Experience Design,” Medium, https://uxplanet.org/the-importance-of-user-experience-design-988faf6ddca2 (accessed Jun. 14, 2023).

[2] D. Gotterbarn et al., “The code affirms an obligation of computing professionals to use their skills for the benefit of society.,” Code of Ethics, https://www.acm.org/code-of-ethics (accessed Jun. 1, 2023).

[3] D. Editor, “Code of ethics,” IEEE Computer Society, https://www.computer.org/education/code-of-ethics (accessed Jun. 15, 2023).

[4] M. Davis, “A General Approach to Ethical Problems ADAPTED FROM ETHICS ACROSS THE CURRICULUM MATERIAL.” The Daniels Fund Ethics Initiative, https://cs-courses.mines.edu/csci370/Slides/EthicsFrameworks.pdf (accessed June. 6, 2023).

XVI. Appendix A – Key Terms

Includes a descriptions of technical terms, abbreviations and acronyms

Term	Definition
API	Application Programming Interface - Mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.
Attestation	An official verification of something as true or authentic
BigQuery	Severless (no need to manage worker servers) fully-managed (by Google) data warehouse tool to store “Big” data and query “Big” data.
Bucket	A basic container to store data on the cloud
Cypress	A JavaScript front-end automated testing tool
Django, Flask, FastAPI	Backend libraries similar to Java Spring Boot but for use in Python programming language
DOM	Document Object Model - programming API for HTML and XML documents or the logical structure of documents and how the elements inside the document are accessed and manipulated
Firebase Realtime Database	Google Cloud service for simple authentication and database integration into front-end applications
FireFox	Web browser created by Mozilla
GCP	Google Cloud Platform - The cloud product Google owns and sells different cloud services on.
GCS	Google Cloud Storage
Gradle	An automated build tool used to build and run tests
Ground-Truth	The confirmed attested data used to test ML models
Java Google Cloud Storage Client	A client for GCS to access the storage using Java
Java Spring Boot Framework	Programming model for modern Java-based backend applications
JSON	JavaScript Object Notation - JavaScript-compatible file format
Lead	A person who is interested in the product or service you sell
ML	Machine learning- The process of building a system to learn from historical data using statistics and computer science.
MVP	Minimum viable product - The end product which will meet all of the client’s hard requirements
MySQL	An open-source relational database management system
NPI	National Provider Identifier - A unique 10-digit identification number for covered health care providers.

OPD	Orderly Provider Directory - Orderly’s product to provide accurate provider data.
PII	Personally Identifiable Information - Information that can indirectly or directly identify a unique person e.g: social security number (SSN), passport number, driver's license number, taxpayer identification number.
Python	High level programming language
SalesForce	Salesforce is a cloud-based customer relationship management (CRM) platform that helps businesses manage their sales, marketing, and customer service operations
SQL	Structured Query Language - A domain specific language used for managing data held in a relational database management system
TDD	Test Driven Development - a software engineering practice where developers continuously write failing tests for feature and write features such that it passes the tests
TMs	Team members of the group - The three members of the team working on this project
UI	User Interface - What the user sees when interacting with the portal.

XVII. Appendix B – Tables and Figures

a. Tables

a. Figures

XVIII. Appendix C - Development Setup
https://github.com/OrderlyHealth/csm-field-session-23