fEMR Coder Portal
Tech Spec

Prepared by team SkeleTubbies:
Jaime Aguilar, Naomi Donato, Chloe Lo, Alexis Ramirez, Garrett Ravatt

Created on: November 11, 2022
Last updated: March 23, 2023
Task tracker reference
Github repository: Frontend
Backend
Beta Site: GitBook

Table of content

Introduction

a. Problem Description

fEMR is a non-profit organization employing, in part, inexperienced volunteer medical coders using a non-standard medical coding workflow in an effort to accurately process raw data obtained from their kits into useful statistical information.

Today, these volunteer coders use google sheets alongside external resources to process encounter data. There is currently little or no automation on the process of assigning encounters and collecting data from coders, and the tools coders use are not specialized for their task.

b. Glossary

Encounter: a single visit to a kit operator for a single patient

Medical Code: a brief string (e.g. AB123) that identifies significant information about an encounter

ICD-10: The tenth revision of a set of international medical codes set forth by the WHO to categorize diseases, injuries, and other patient conditions. A code from a different ICD encoding will append as '.<version>,' ie, an ICD-9 code will be something like AB123.9.

CPT: A set of medical codes used to categorize the treatment a patient received

Coder: medical coder; a person who interprets medical codes for encounters; refers to both standard coders and arbiters

Standard Coder: a less experienced medical coder; each encounter is read and interpreted by two standard coders

Arbiter: experienced medical coder; a coder that resolves disputed code interpretations between standard coders

Coder Admin: a manager for coders; responsible for creation of coder accounts and promotion of standard coders to arbiters

Site Admin: a manager for coders and portal admins; responsible for creation of portal admin accounts

Campaign: a period of deployment for a fEMR kit

c. Background

“Team fEMR aims to build continuity between short-term medical relief efforts in low resource settings… We think that data-driven communication is the key to improving the quality of care that transient volunteers provide.” -Quote obtained from fEMR Coders Guidelines

Our meetings with fEMR staff have also revealed that this operation serves a role in the medical coding job market. Nurses who wish to become medical coders are expected to have experience with coding when applying to many coding jobs, but there are few opportunities for them to do so. fEMR's volunteer program is one such opportunity.

d. Goals and Technical Requirements

User Stories:

As a site administrator…

- I would like to log into an account with an assigned role (site admin) so that I can access the portal.
- I would like to assign and manage coder admins so that I can give them access to manage the coders.
- I want to deactivate the role of coder admin when the coder admin has resigned so that they no longer have permissions.
- I would like to see the metrics (activity, accuracy, and velocity) of standard coders so that I can evaluate their performance.
- I would like to upload or download campaign data in my preferred file format so that I can reference and update the data accessible through the portal.

As a coder administrator…

- I would like to log into an account with an assigned role (coder administrator) so that I can access the portal.
- I would like to create coder accounts so that I can onboard medical coders.
- I want to deactivate standard coders' accounts so that when they resign I can remove their access.
- I want to reactivate the role of an arbiter and standard coder if they reach out so that they can work with us again.
- I would like to change the role for an existing account so that standard coders can be eventually promoted to an arbiter.
- I would like to see the metrics (activity, accuracy, and velocity) of standard coders so that I can evaluate the performance of these standard coders.
- I would like to sort ICD-10 codes that have been assigned by category (infectious diseases) so that I can get data on the number of patients with similar diagnoses.

As an arbiter…

- I would like to log into an account with an assigned role (arbiter) so that I can access the portal and begin resolving a code conflict for an encounter.
- I would like to see all the data for an encounter on the same screen so that I can cross-reference information quickly.
- I would like to view standard coders' codes so that I can give them feedback.
- I would like to make the final decision on the code for an encounter so that I can resolve differences between the standard coders.
- I want to see the red color of the code if there is a difference between two coders, so that I know that code need to be reviewed
- I would like to see a green color for the code if the two standard coders assigned the same code to a patient so that I know there is no need to review the encounter.
- I would like to see a history of arbited medical codes in the patient's information, so that I can apprehend the patient's conditions when providing accurate codes for this encounter.

As a standard coder…

- I would like to sign up for an account so that I can access the portal.
- I would like to log into an account with an assigned role (standard coder) so that I can access the portal, and begin coding for an encounter.
- I would like to submit an encoding for an encounter so that I can practice my coding skill
- I would like to see all the data for an encounter on the same screen so that I can cross-reference information quickly.
- I would like to receive an email when I am approved to contribute to team fEMR so that I know when to start coding
- I would like to see a list of coders in a table so that I know the names of the coders I will possibly be working with.
- I would like to see my personal metrics (activity, accuracy, velocity) so that I know my current performance and the area that need to be improved
- I would like to see results from the other standard coders and the arbiter's decision for feedback so that I can learn from them
- I would like codes to be inputted with similar syntax (or for whitespace, dashes, etc. to be removed from codes after submission) so that I won't be flagged for minor typos in codes.
- I would like to receive a notification if my code was approved or rejected.
- I would like to see a history of arbited medical codes in the patient's information, so that I can apprehend the patient's conditions when providing accurate codes for this encounter.

As a client…

- I would like to submit a request through the self-registration portal so that I can access the parts of the portal I need to.
- I would like to reference publicly available campaign data to use in my research.

- I would like to create a campaign for coders using data from my fEMR kit so that the data is backed up and others can benefit from it.

Technical Requirements:

- The system shall be compatible with Chrome.
- The system shall not allow any users unnecessary access to patient data in conformance to HIPAA guidelines.
- The system shall exist as a component of a larger fEMR central web service:
    - frontend shall be a unified React web app with components maintained by each team.
    - backends for services will be separate and maintained by individual fEMR development teams using Django.
    - central database shall be shared across all services and teams.
- The system shall enable interactions for self-registration forms provided by frontend web services maintained by Team Fibula.

- The system shouldn't allow more standard coders to encode an encounter than needed (eg 3 coders active on an empty encounter or 2 coders active on a half-coded encounter).
- The system shouldn't allow coders to choose a specific encounter to code, only the choice to code an encounter from a campaign.
- The system shouldn't allow coders to 'reserve' an encounter to encode. If the coder breaks connection with the portal, the encounter they were coding may be given to an active coder.

- The system shall be able to interpret campaign data by communicating with a remote database.
- The system shall be able to output or populate campaign data in csv or HL7 FHIR file formats.

f. Future Goals

- As a coder, I would like all encounter data to be translated into my preferred language so that I can do my job with fewer tabs open.
- As a coder, I would like to look up the meaning of a medical code by mousing over it.
- I would like to message the other standard coders so that I can discuss with them regarding the code we write
- As a coder, I would like to sort the codable campaigns by priority so that I can quickly find the campaign I want to code for.

g. Assumptions

- A shared database between all fEMR-central teams is an optimal solution
- A specialized portal will improve the performance of coders
- fEMR OnChain API is functional and supports our needs
- Legacy fEMR data is uploaded to the central database

Solutions

a. Current Solution

An admin with access to the identified data runs a script that draws 100 rows of deidentified data at a time.

That segment is uploaded as a sheet to Google Sheets, and is assigned to two standard coders. Those standard coders read the sheet, referencing third party tools to lookup medical codes and verify their own work.

Once they are done, an arbiter reads the sheet. Wherever there is a row where the standard coders got different answers, the arbiter inputs the final set of medical codes.

There is no current database for this final data, meaning it is only accessible through accessing Google Sheets directly.

This process is repeated, assumedly until all the data is processed.

pros:

- Coders are familiar with the current solution
- Solution uses common tools (spreadsheets)

cons:

- Admins have to create new pages for every hundred encounters that are processed
- Arbiters have to search for conflicts instead of seeing only conflicting data

b. Proposed Solution

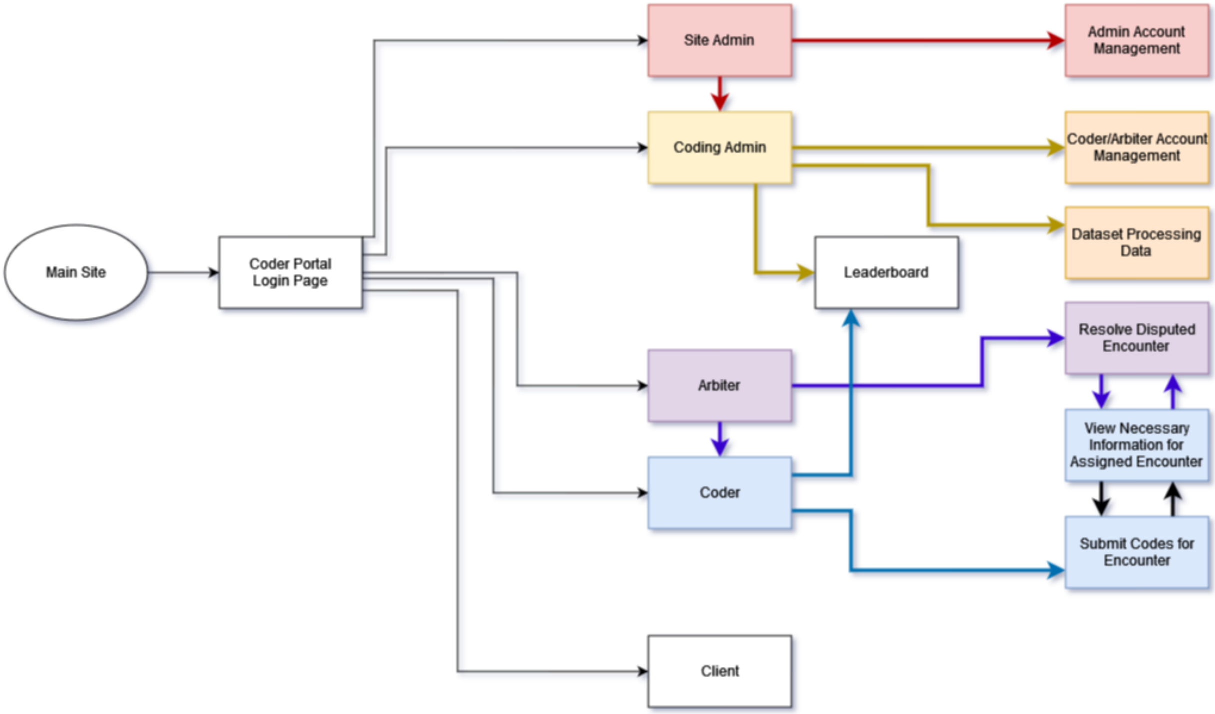

Site Admin
Admin Account Management
Coding Admin
Coder/Arbiter Account Management
Dataset Processing Data
Main Site
Coder Portal Login Page
Leaderboard
Resolve Disputed Encounter
Arbiter
View Necessary Information for Assigned Encounter
Coder
Submit Codes for Encounter
Client

Diagram 1: Role-Based Web Service Access Diagram

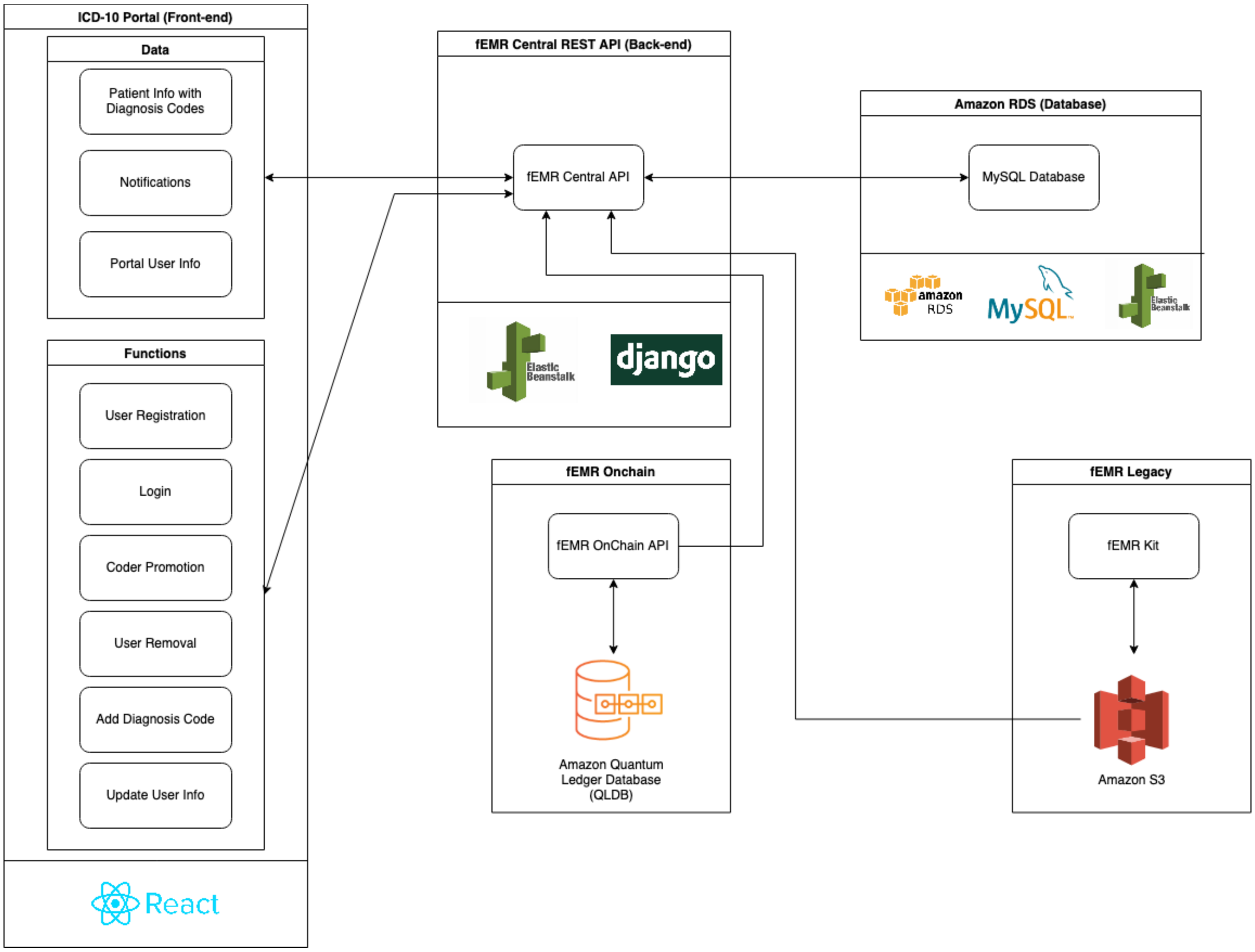


Diagram 2: Architecture Diagram

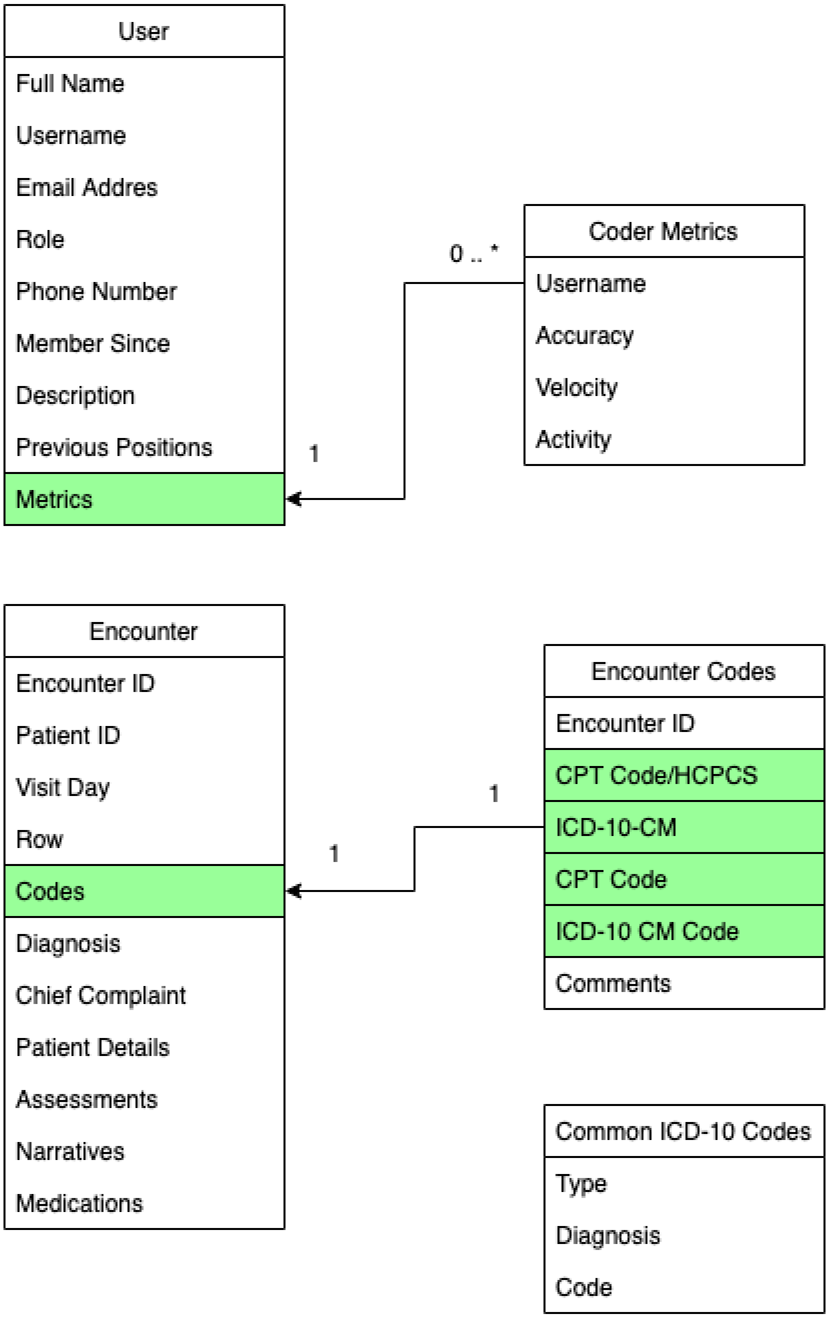
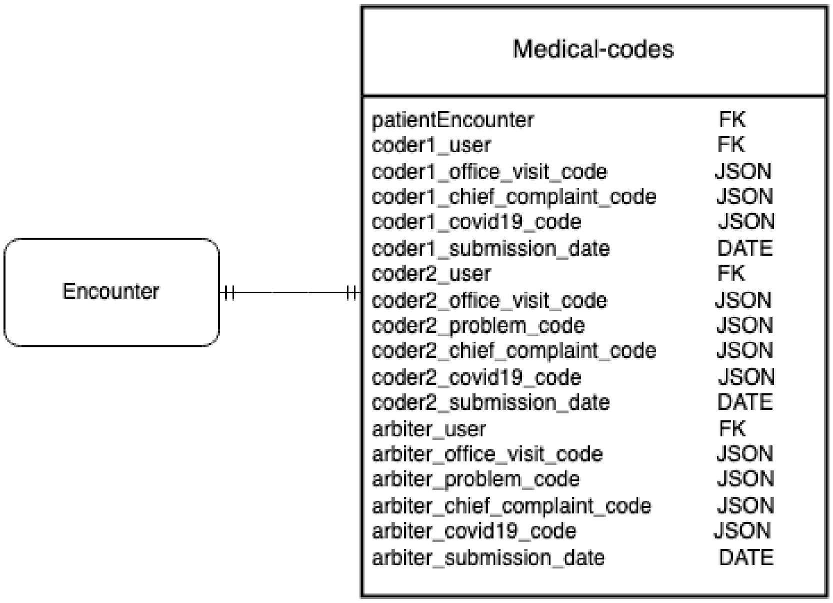


Diagram 3: Backend UML Diagram

c. Data Schema

- Medical-codes schema: This schema is for creating a set of codes from coder1, coder2 and arbiter for each encounter



Medical-codes	
patientEncounter	FK
coder1_user	FK
coder1_office_visit_code	JSON
coder1_chief_complaint_code	JSON
coder1_covid19_code	JSON
coder1_submission_date	DATE
coder2_user	FK
coder2_office_visit_code	JSON
coder2_problem_code	JSON
coder2_chief_complaint_code	JSON
coder2_covid19_code	JSON
coder2_submission_date	DATE
arbiter_user	FK
arbiter_office_visit_code	JSON
arbiter_problem_code	JSON
arbiter_chief_complaint_code	JSON
arbiter_covid19_code	JSON
arbiter_submission_date	DATE

d. Test Plan

We plan to have frontend testing using Jest and Taiko, as well as backend API testing with an emphasis on request permissions. We plan to emphasize code coverage on sensitive backend elements.

Jest will be used by itself for unit testing of internal utilities and React components. React's testing library can

Jest and Taiko will be used to implement unit and end-to-end testing, and Django's testing module will be used for backend unit testing.

We will have a goal of 85% code coverage for core backend functionality, e.g. account management, encounter code submissions. We will only require 65% code coverage on non integral functionality, e.g. coder self-status notifications.

e. Alternate Solutions

An alternate solution would be to manipulate the currently implemented google sheets workflow to allow only administrators the ability to approve new deidentified information creation and assignment, and for them to view coder statistics derived from google sheets activity.
This solution would not provide a centralized database solution for the data the coders produced. For better or for worse, no change would be apparent to the coders.

Further Considerations

a. Security Considerations

Potential threats:

- Any scenario involving the database responding to a query with identified patient data

Mitigations:

- Database will only store deidentified data, which can only be identified
- Database will only give patient and encounter data to coders

The proposed solution will offer a lot more granular control over what data is seen by who and how often. For instance, coder admins will no longer need to create the deidentified data the coders are accessing, and arbiters will only view encounters they need to resolve

b. Privacy Considerations

- How is patient information de-identified in the application?
    - The patient is not identified by their name and credentials, but with an identification number.
- How is patient information protected in our application?
    - Only standard coders and arbiters can view patient history as they should be HIPAA compliant.
- How is privacy given to standard coders and arbiters?
    - Standard coders and arbiters are not allowed to track each other's real time activity.

c. Regional Considerations

- Content of portal is written in English
- Encounter data may be written in languages other than English

d. Accessibility Considerations

- Consideration for color blindness and sight impairment

    - Distinguishing of different colors and what they mean
        - Red if standard coders disagree
        - Green if disagreement of code is resolved

- Considerations for collaboration over Zoom
    - Having icons and buttons that are distinct and easily described

e. Operational Considerations

- How will the cost be kept low?

f. Risks

- What are the risks that our system may undergo?
    - Storing sensitive data makes you a hacking target, but we are storing de-identified patient data
    - Since we need to collaborate with the other teams, and it is a continuous project, other teams or any team in the future may take our work. Therefore, we need to ensure code and system documentation will be created and readable.

- What if our application is not considered completely finished?
    - We plan on making the main functionalities of our application work, which is the standard coder and arbiter interaction of applying codes to de-identified patients. Our other main functionality is having the coder administrator assign a standard coder to be an arbiter. Everything else such as the leaderboard, built in translation functionality, and notifications should be added on to enhance our core functionality.

g. Handover

- OnChain-to-Legacy Doc: https://docs.google.com/document/d/1_56bjLrpztMFXaZvrcZjrtDgAMBGTKiNjo1UgxdeOC0/edit?usp=sharing
- Models and Fields Needed to Obtain a Single Patient Encounter: https://drive.google.com/file/d/1YT5C_WJfvxTJIAV7xoL27DGi6JuPrD8f/view?usp=sharing

Deliberation

a. Discussion

We are still developing a shared understanding of code documentation and regulating pair programming within the team.

b. Open Questions

How will the database be hosted?

References and Acknowledgements

- Acknowledgements
    - Sarah Draugelis, Andi Mastie - Team fEMR
    - Professor BJ Klingenberg - California Polytechnic State University CSC/SE Department