1. Front matter

- Team fEMR
- Collin Leslie, Noel Treese, Nick Baggett, Joon Lee, Omar Shorab
- Team Lemur
- Created on November 11, 2020
- Last update : March 10, 2021
- http://platinum.cscaws.com:8080/secure/RapidBoard.jspa?rapidView=38&view=planning&selectedIssue=LEMUR-17&issueLimit=100
- https://github.com/CPSECapstone/lemur-femr.git
- https://github.com/CPSECapstone/super-femr

2. Introduction

a. Problem Overview

Mobile teams travel to remote areas to diagnose illnesses, treat injuries, and administer drugs to support the sick and injured. However, due to lack of internet connectivity, medics must often rely on stored information that may not have been updated for days. Furthermore, details about new patients will remain on local databases until the medical teams return to areas with some form of connectivity. The suggested solution is to implement two main features:

1. Enhance fEMR kit to be able to upgrade software, update the database model, migrate the data, and synchronize the database to a remote repository. This should happen whenever a kit is re-connected to the internet.
2. Add the capability to merge data from multiple kits into one database, to address the issue of overlapping patient data coming from different kits.

The primary stakeholders in this project are the fEMR team members and CEO. Second are the developers working on this project. But there are also stakeholders outside both of these organizations. Every doctor, nurse or anyone else using this kit is ultimately a stakeholder, even if they are not the primary one.

b. Glossary

- Kit - fEMR product used to enter medical records
- Patient - general term for visitors to the pop-up clinic
- Triage - check-in form used to enter patient's medical information
- Encounter - when a patient sees a doctor or nurse
- Pharmacy - a tab to check if a patient were prescribed any medication
- Nurse - provide and monitor patient care, educate patients and family members

- Doctor - qualified practitioner of medicine. Responsible for treating patients at pop up clinics
- Researcher - trip attendees conducting medical or other research alongside doctors or nurses
- Trip Administrator - person in charge of conducting the popup clinic and ensuring all facilities are in place to provide medical care to patients

c. Context or Background

This problem is worth solving because there are people living in remote areas without access to hospitals and health care that need the assistance from medical professionals. This product enables doctors, physicians, and nurses to create "Pop-up" clinics that can serve many patients anywhere in the world.

Prior to the fEMR kit, medical professionals were limited by the constraints of physical records. Not only were they harder to organize and maintain, the biggest issue was not being to see a patient's previous records. This is an issue since some patients depend on medicine and treatment that is life-saving, however, if the team does not know such supplies are needed prior to making their trip, it can cost lives.

Currently, fEMR kits help solve that issue, but the product has its own limitations. The data collected at a remote location can only be accessed and stored onto the central database once the kit is physically returned. Team fEMR's goal is to have data accessible as soon as possible to allow for a patient history to be available. Many of the patients treated are refugees that often need to relocate due to circumstances. For example, if a refugee is treated in location A, then moves to location B to get treated again, their medical record will not be accessible if the kit at location A has not been returned yet. For this reason, the main goal of team fEMR is to implement a data syncing feature that can merge a local kit's data to the central database.

d. Goals or Product and Technical Requirements

- Product requirements in the form of user stories
    - *As a doctor in a remote "Pop-up" clinic I want to store a patient's data so that I can reference it in the future.*
    - *As a physician in a remote "Pop-up" clinic I want to merge patient data collected locally when good enough internet connectivity is found.*
    - *As a user of the fEMR kit, I want to perform any software updates on the kit when internet connectivity is found.*
- Technical requirements
    - *The system shall not allow a local kit to overwrite data from the central database when merging*
    - *The system shall not allow for duplicates to exist in the central database.*

Note on terminology: In this template, *product requirements* are functional requirements, whereas *technical requirements* are non-functional requirements.

e. Non-Goals or Out of Scope

- Enhancing the fEMR kit so that a physical piece of hardware is not needed for functionality.

f. Future Goals

- Make femr a runnable application instead of requiring a java project to be compiled and run
- Allow femr to work anywhere in the world by providing a software download instead of shipping out prebuilt kits
- Combine the software and database update into one consolidated download
- Allow admin to trigger internet test script and software update

g. Assumptions

- Internet connectivity is required for the local kit to update software.
- Strong internet connectivity is required for the kit to sync local data to the central database.

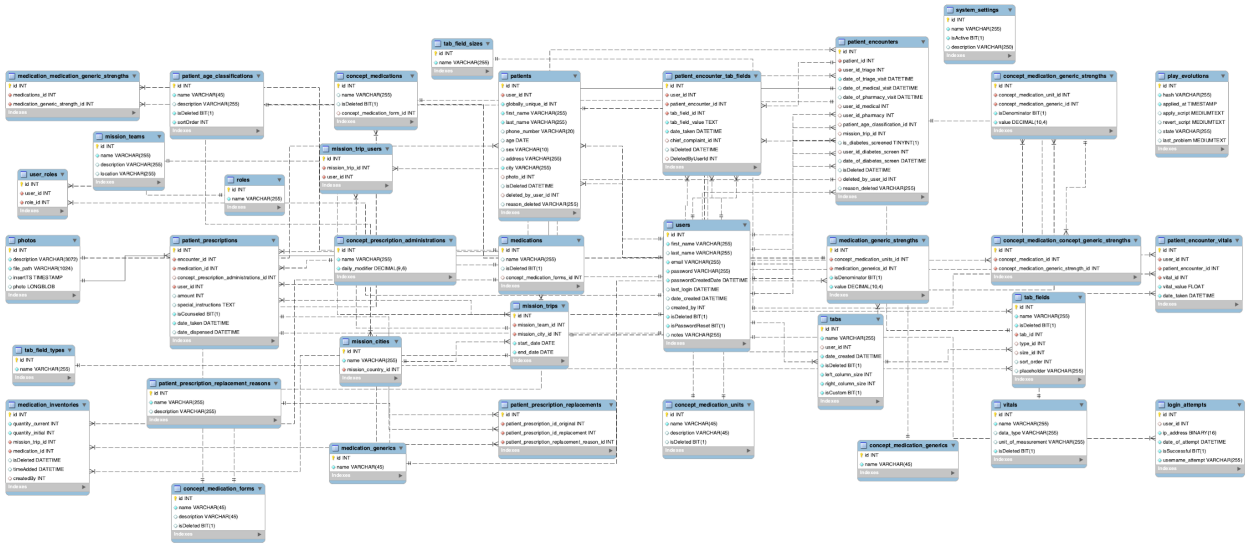
3. Solutions

a. Current or Existing Solution / Design

- The fEMR software provided currently is able to perform well on local instances (kits). It has the capability to set up the necessary MySQL tables provided that there exists a database that it can work within. It also has the capability to perform all the CRUD operations necessary for a kit to function locally. There is no master database, no globally unique identifier, and no ability to merge local databases with a master db.
- Pros of current solution
    - The fEMR software functions as expected when being run locally
- Cons of current solution
    - The fEMR software has no capability to work with a remote master db

b. Suggested or Proposed Solution / Design

- The solution that we are designing will interact with a remote master db hosted on AWS using RDS to maintain a permanent and consolidated record of the data that is currently only stored locally
- The current solution requires MySQL and an existing database that it can work with.
- Pros of the proposed solution
    - It will provide a permanent storage solution for all the data collected on local kits.
    - Data will be merged automatically when possible.
- Cons of the proposed solution
    - Depending on the quality of the globally unique ID for patients, duplicate patients may be entered into the master db.



- **Data Model / Schema Changes**
    - See the table above for the entire database schema
    - Changes to the original schema are minimal. We added a globally_unique_identifier (guid) field to the patients table in order to identify specific patients across different local instances.
    - Validating that a guid is good is difficult. Many things might change between patient encounters that make identifying this patient difficult. We will try and generate the guid based on information that will not change and is unique to that patient.
- **Business Logic**
    - API changes
        - An endpoint may be added to trigger a merge with the master database.
    - Error states
        - Unsuccessful merge.
        - Db with duplicated data.
        - Unable to connect to a strong enough network.

        - Network disconnect mid merge.
    - Failure scenarios
        - Unable to complete a successful database merge.
        - Database allows for duplicate patient data.
    - Conditions that lead to errors and failures
        - Loss of internet connectivity.
        - A difference in versions between the local and remote database.
        - Duplicate patient records in either database.
    - Limitations
        - Heavily limited by the availability of a strong enough internet connection.
- **Presentation Layer**
    - UI changes
        - Align text field boxes for login/change password screen.
        - Update navigation bar.
    - Web concerns
        - Rendering components on various screen sizes.
- **Other questions to answer**
    - How will the solution scale?
        - Since the merge happens between two databases, the only element of scale is the amount of data.
        - The length of the merge will scale with the amount of patient data.
        - Once multiple kits attempt to merge, we will need to create a blocking queue which will not allow other ktis to merge until the kit currently merging finishes. This will prevent data conflicts and anomalies.
    - What are the limitations of the solution?
        - Limited by internet connectivity.
        - With the above queue solution, only one kit can sync at a time which may prevent other kits from being able to sync if internet access is limited. Synching data should be a quick task though since local databases tend to be small.
    - How will it recover in the event of a failure?
        - Attempt to remerge data. Offer users a manual merge if the automatic one fails.
        - The remote database will most likely take advantage of database snapshots. A snapshot should be created before a database merge so that a rollback is possible in the event of data loss or other issues.

c. Test Plan

Explanations of how the tests will make sure requirements are met and whether there is a minimum bar for a standard test coverage.

- Unit tests

    - Unit tests are already in place.
    - We can add more unit tests to this testing file.
    - Achieve at least 60% test coverage.
- Integrations tests
    - Continuous testing is already in place.
    - We can create more tests to be run by the CI server.

d. Alternate Solutions / Designs

- Short summary statement for each alternative solution
    - In the event that the AWS RDS database will not suffice, we could create the central repo using an Azure server or building our own server.
- Pros and cons for each alternative
    - Possible Cons:
        - API integration
        - Server availability.
    - Possible Pros:
        - May be easier to set up.
        - May be cheaper than AWS
- Reasons why each solution couldn't work
    - We may be unable to build a MySQL server from scratch on our own server.
- Ways in which alternatives were inferior to the proposed solution
    - AWS tends to be easier to set up than most other hosting services.
    - Hosting our own server would require a dedicated computer to run a MySQL server.

4. Further Considerations

a. Privacy considerations

- We have considered making our solution HIPA compliant. The fEMR team has informed us that this is a future goal of theirs and if our solution remained HIPA compliant they would not need to modify it to become compliant. The downside of achieving HIPA compliance with our solution is that there are a lot of restrictions on data processing and management and we would need to weigh each restriction carefully.
- Also, since the fEMR kits are globally deployed we would need to consider local laws on patient data. In some countries, it might be illegal to send patient data to a main repository.

b. Accessibility considerations

- Our solution will be extremely accessible. The goal is for it to fit right in with the current fEMR software and for it to simply run in the background with little to be done on the user's end.
- In order to evaluate accessibility, we will evaluate using a standard or a checklist such as this one from the DOJ.

5. Deliberation

a. Discussion

- There are currently no elements of the solution that members of our team do not agree on and need to be debated further to reach a consensus.

b. Open Questions

- When executing a software update, is it possible to keep the application open or can we perform an automatic restart?

8. References and Acks

- Official Documentation: https://github.com/FEMR/documentation\
- Femr Front End/Testing: https://github.com/FEMR/femr-frontend
- Problem Codes: https://github.com/FEMR/data-code
- Credit people who have contributed to the design that you wish to recognize.
- Sarah Draugelis, President, Team fEMR
- Andy Mastie, Chief Technical Officer, Team fEMR