TEAM

femr

fEMR Capstone Project Technical Specification

Authors:
Jacky An
Mirei Machiyama
Yash Satyavarpu
Siriwan Sereesathien
Kristina Tran

Team Feather

Reviewers: BJ Klingenberg, Sarah Draugelis, Andy Mastie
Created on: October 29, 2020
Last updated: March 9, 2021
Epic, ticket, issue, or task tracker reference link: JIRA
Repository: https://github.com/CPSECapstone/feather-femr

1. Introduction

a. Overview, Problem Description, Summary, or Abstract

The Fast Electronic Medical Record (fEMR) system is an EMR system specifically designed for "pop-up" clinics. The situations differ from EMR systems in traditional medical facilities because there may be a lack of internet connectivity and lack of personal identification for patients. The existing fEMR implementations are self-contained "kits" that store data only locally.

Team fEMR has requested Team Feather to improve their product to expand their data access and reliability. This project has largely two parts: the functionality to interact with a central remote database and the functionality to safely merge this data with existing data in the central database.

Team Feather is working with Team Lemur and ZZS to realize the goals of Team fEMR. Team Feather has taken on the responsibility of handling data migration.

b. Glossary of Terminology

Please refer to our Github wiki glossary found here.

c. Context or Background

fEMR's current architecture requires manual maintenance when kits are returned to the organization. Data is stored locally on a hard drive and the kits must be wiped each time they are deployed. Making such processes automatic can speed up the turnaround greatly. Implementing a central AWS server allows for unique patient keys, reduced risk lost data, and an easier way of loading past patient data. By having previous patient data, medical teams can stock up on appropriate medication and resources before arriving at the site.

Our solution must assume and accommodate for the technical requirement of having only sporadic chances to connect to the internet.

d. Goals or Product and Technical Requirements

- Product requirements in the form of user stories
    - As an administrator, I want to access patient data from previous trips.
    - As an administrator, I want to choose whether to update software now or to defer.
    - As an administrator, I want to regularly merge my data to the central database, so I can access them on future trips.
- Technical requirements
    - The system shall be compatible with Chrome's latest versions.
    - The system shall not assume a constant, stable internet connection.

e. Non-Goals or Out of Scope

- Handling real patient data (compliance reasons).

f. Future Goals

- HIPAA compliance.

g. Assumptions

- User is running fEMR on Chrome.

2. Solutions

a. Current or Existing Solution / Design

- The current fEMR approach and solution involves dispatching kits to urban areas that provide a Electronic Medical Records system for transient medical teams in order to promote data driven communication in low resource settings. The records are kept in a "kit-by-kit" basis - as such the patient records are non-transferrable to other kits. The data kept on an individual kit are preserved and accessible for the next trip.

- Benefits: Kits are portable and an all-in-one service. In addition, all patient data is kept offline. The user interface is kept relatively simple and non intrusive to provide quick and easy access to adding, editing, and looking up patient data.
- Limitations: Patient information and details are kept on an individual kit with no current design to back up the data. As such, if any loss of data occurs, there is an absence of backup data (current implementation in progress). In addition, the transfer for patient data to other kits is restricted due to the HIPPA compliance.

b. Suggested or Proposed Solution / Design

We will be using an external AWS database as the central database for all patient information from the different kits. We need to merge all the current data we have from the local SQL database into the central, but we don't have access to see the data due to the HIPPA compliance restriction. We will be adding an observer/scheduler to notify the kit for an update to the central database for the kit once it has access to the internet. The admin/kit user will have the option to manually update or accept the observed/scheduled update.

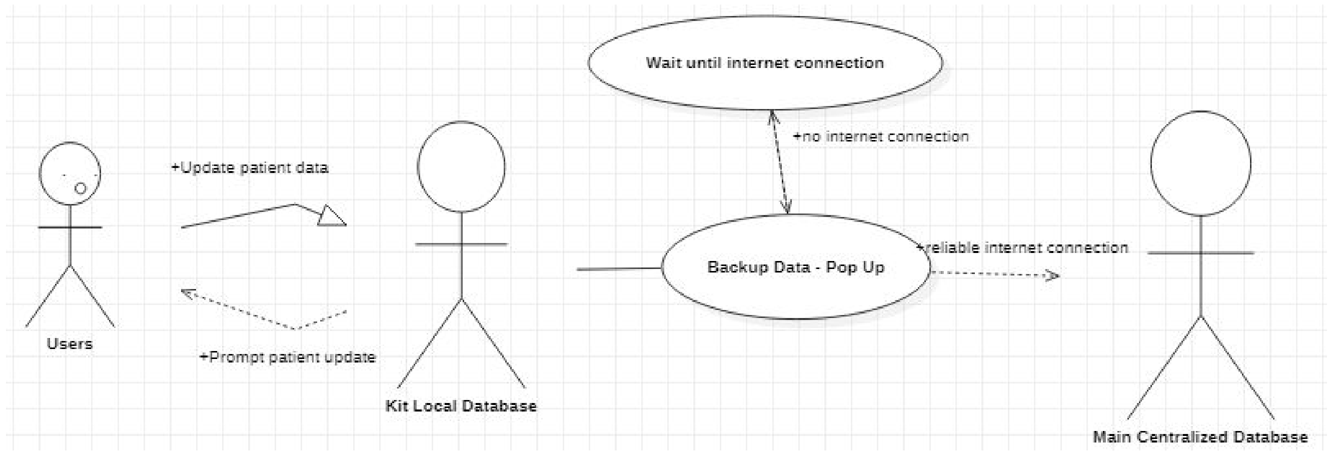
Pros:

- All patient data will be accessible from one location
- No risk in losing data while transporting the kit

Cons:

- Model updates in the database could result in loss of information
- Require internet access at remote sites that may or may not be available

Overview solution design:



Data Model / Schema Changes

- Need to add unique identifier
- PatientID = unique identifier
  - Will be hashed using Patient's information:
    - Name
    - Birthday
    - Kit ID (Admin ID?)
  - Collision of the hash will be taken care with a probing technique that we will decide on later
- **Business Logic - *pending…*** (working on it as we make progress)
  - Flowcharts
  - Error states
  - Failure scenarios
  - Conditions that lead to errors and failures
  - Limitations
- **Presentation Layer**
  - UX/UI changes regarding patient matches and software/database updates
  - A pop up will appear when there seems to be a conflict in users
  - A pop will appear when it's connected to the internet to have all data backed up to the server
- **Other questions to answer**
  - How will we update the data model of the kit when we have to perform a software rollback?
  - How will the solution scale?
  - What are the limitations of the solution?
  - How will it recover in the event of a failure?
  - How will it cope with future requirements?

c. Test Plan

- Unit tests -- JUnit; for future code changes, we should at least get 50% coverage

- Integrations tests -- Sandwich testing
- Selenium tests -- E2E for cases such as user input that seem to collide with existing user

d. Alternate Solutions / Designs

- None currently

3. Further Considerations

Security considerations

- What are the potential threats?
  Potential threats could be a third party gaining access to the patient data, therefore violating HIPAA.
- How will they be mitigated?
  This could be mitigated by making sure that end points for transferring patient data are secure by using SSL encryption as well as designing a system that doesn't allow for SQL injections
- How will the solution affect the security of other components, services, and systems?
  The solution provides a more secure and reliable system now that the main patient data is backed up online in an AWS RDS rather than in a physical hard drive.

Privacy considerations

- Does the solution follow local laws and legal policies on data privacy?
  Yes, the solution should be HIPAA compliant.
- How does the solution protect users' data privacy?
  The solution protects user data by ensuring that other kits, especially those in other countries, do not have access to other kits' data.
- What are some of the tradeoffs between personalization and privacy in the solution?
  The solution doesn't provide any sort of personalization from user to user, therefore not compromising the privacy/security of the system.

Regional considerations

- What is the impact of internationalization and localization on the solution?
  The impact on internationalization/localization of the solution should be minimal, the biggest concern is the translation of certain characteristics between the languages, but this can be solved with enums.
- What are the latency issues?
  This is remedied with the admin. In the current solution, we set up a cron task which will prompt the admin to upload the most recent patient data each week. The admin will make the judgement call on when the best time to upload the data.
- What are the legal concerns?
  Legal concerns could arise with how secure it would be to store patient data in AWS's RDS.

- How will data transfer across regions be achieved and what are the concerns here?
  Data transfer will not be a thing between different kits, but the primary transfer will occur between the kits and the central database

4. Deliberation

a. Discussion

- We have agreed on all the elements of our solution thus far. However, there are still some areas we have yet to design our solution, such as how we will support upgrading legacy versions of fEMR that are in the field.

b. Open Questions

- How should we support the ability to upgrade old versions of fEMR without losing any data?
- Should we utilize Docker to upgrade/downgrade software using a private registry and swapping out images?
- Or should we just use git's pre-existing version control system to update and rollback versions of the fEMR software?
- Or should we use the custom script written by Team Lemur that downloads all updated files and replaces them?
- Will the kits ever be in a situation where they can lose power? Should the kits ever be deployed to a location without power?

c. References and Acknowledgements