Technical Spec

Team trEMR
Bat Battur, Skylar Kurth, Rey Ortiz

Created on 11/4/2021

https://platinum.cscaws.com:8443/secure/Dashboard.jspa
https://github.com/FEMR/femr/tree/tremr-branch

Last Revised on 06/7/2022

TABLE OF CONTENTS

2. Introduction

a. Overview, Problem Description, Summary, or Abstract

- With the current refugee crisis and an increasing amount of disasters occurring worldwide, many of the first responders who use fEMR have an increased need for an EMR that suits trauma responses. Trauma care is unique in that patients may be unidentifiable; triage decisions need to be made rapidly; and pertinent medical information needs to be recorded much quicker than in a primary care visit. Our vision is to add a module to legacy fEMR that accommodates the trauma response.

b. Glossary / Terminology

- Trauma Sheet - Digital version of the trauma flow sheet.
- Anonymous Patients - patients that can be added with minimum information, such as automatically assigned patient id number.
- Color Coding - Medical Triage Color Spectrum for different pain levels.
- Mission Trips - the place where disaster/trauma event occurred and where the trauma intake process is occurring.

c. Context or Background

- Currently, the trauma response is recorded only in paper form which can be inconvenient during rapid and fast paced environments. The paper documents can be lost or damaged, therefore, the important information about the patients can be lost.

- Thus, creating a digital version of EMR that suits trauma responses would be ideal so all of the information can be centralized and stored effectively. This will enable intake nurses and physicians to document critical information about the patients faster and track them in an efficient way.

- In our research, we found that there are no trauma sheets in digital form currently available. Therefore, we have implemented the first iteration and the design of our project. We planned to develop the user interface and the backend functions simultaneously to store the trauma patient information. Then, we planned to do usability testing with actual users that will be using our product in an actual trauma setting to further improve the trauma sheet.

d. Goals or Product and Technical Requirements

- Product requirements in the form of user stories:
  - **As an** intake nurse/physician in a trauma setting, it is important that I can quickly document the injuries and needs of my patients,
    I want to spend as little time as possible doing this,
    So that I can get straight to getting our patients care with the best flow and interaction will help us succeed in the field.
  - **As an** onsite nurse or physician in a trauma setting
    I want to be able to draw on the image of human body on the trauma sheet,
    So that I am able to note the areas of injury in a fast and efficient way.
  - **As an** onsite nurse or physician in a trauma setting
    I want to be able to view the different patients receiving care
    So that I am able to keep track of my patients and their care
  - **As a** nurse/physician working with fEMR kits
    I want to be able to zoom in on unclear parts of the trauma sheet
    So that I can quickly and correctly gather critical information on the patient I am treating
- Technical requirements:
  - GUI Interface to support use of graphical body model patient intake.
  - The trauma sheet should be able to add anonymous patients by number only, where details can be added later.
  - The trauma sheet should integrate Color Coding for trauma level - showing different pain levels.
  - The trauma sheet interface will support Google Chrome on both PC, and tablet systems.

e. Non-Goals or Out of Scope

- Mobile compatibility and responsiveness for the trauma sheet.
- Language picker for translation / internationalization.

f. Future Goals

- Pinch zoom/unzoom or mouse wheel or zoom select depending available client tech.

g. Assumptions

- The trauma sheet page should be loaded on Google Chrome either on PC or tablet device.
- Intake physicians/nurses should be able to draw on the body chart.
- Trauma patient information can be stored without required fields.

3. Solutions

a. Current or Existing Solution / Design

- Current solution description

Currently in high intensity disaster situations the solution for a quick evaluation is a physical trauma sheet. The femr system is built to handle regular patient intake, but not trauma situations yet.

- Pros and cons of the current solution

A tangible, analog solution is beneficial because it is usually an intuitive experience and there is no chance of crashing/program failure. However, it is much harder to organize and sync up, and there are much more restrictive limitations in regards to physical resources and equipment.

b. Suggested or Proposed Solution / Design

- External components that the solution will interact with and that it will alter

The software will be run on individual femr kits providing their own local wifi network, until the kits are reconnected and the information is synchronized.

- Dependencies of the current solution

The current solution is written in Java and deployed using mySQL and the Play Web Application Framework. The proposed trauma solution is getting built off of this stack as well as relying on the Javascript library to power the graphic interface. For the drawing component, we are using Javascript based Konvajs/react-konva library.

- Pros and cons of the proposed solution

Digital records provide the opportunity for increased organization, which makes the maintenance process of keeping all patient information up to date far easier. This way, the need for specific drugs/resources can be brought to new teams arriving subsequently. However, it should be noted that the reliability of software presents new limitations and challenges that the development team will need to consider thoroughly and carefully.

- Data Model / Schema Changes

Currently the trauma sheet requires that a user enter the first name, last name, and the city before being able to submit the patient. The way that we worked around this was to generate dummy data if a user does not provide that information. For the first and last names, we generate “John Doe” (m) and “Jane Doe” (f), “Not given” for the city. The plan for the future will be to modify the database so that it does not require those input fields.

The database currently does not store input fields from the dispo, medical/surgery, and the problem cards. Below we have added the schemas for each table that have to be added in order to have full functionality.

    - Schema definitions
        - Patient

        - ID - required (assigned by backend on intake of patient)
        - Sex at birth - required
        - Vitals
            - Temperature
            - Blood Pressure
            - Height
            - Weight
            - O2
            - Weeks Pregnant
            - Glucose
            - BMI
        - Name
        - City
        - Address
        - Photo
        - Previous medical history
            - Diabetes, cholesterol, hypertension, etc
        - Problem list
            - Problem Name
            - Medical Plan to treat
        - Trauma Rating

- Schema table for new required input fields
    - Medical_surgical_history
        - id (unique)
        - value
        - patient_encounter_id (foreign key to patient_encounters.id)
    - Dispo
        - id (unique)
        - Value
        - patient_encounter_id (foreign key to patient_encounters.id)
    - Trauma_severity
        - id (unique)
        - severity

            - patient_encounter_id (foreign key to patient_encounters.id)
        - Trauma_encounter_photos
            - patient_encount_id (foreign key to patient_encounters.id)
            - photo_id (foreign key to photos.id)
        - Problem_card_steps
            - id (unique)
            - patient_encounter_id
            - order
            - Value
            - substeps (list of problem_card_steps_substeps.id)
        - Problem_card_steps_substeps
            - id (unique)
            - patient_encounter_id
            - problem_card_steps (foreign key problem_card_steps.id)
            - order_id (order of the substeps)

    - Data validation methods
        - Patient
            - fields are meant to be flexible in order to maintain a quick triage in a trauma setting
            - Required fields: ID, Sex, Age
            - Many optional fields

- Business Logic
    - API changes
      We are working with the internal fEMR database local to each machine (for now). API use for us are simply just calls to the software pages and selections receiving the data from the database.

- Pseudocode
  Key components of our build for the next team:
  1. Implement the trauma sheet digitally on its own page that will be interactive via touchscreen technology. Accepting the same information as the form.
  2. Trauma sheet must be built with components, all are intuitive and use scroll functionality or expandable (not using tabs).
  3. Data must be saved in a different SQL table to ensure it is possible to save patient information without previously important information (ie: name, birthday).
  4. Use such data for the hierarchy log to help intake nurses/physicians make quick decisions with less thought.
- Error states
  There should be minimal errors with this side of the program. There are less requirements for the data input fields. There will be issues if the drawing software doesn't compile and save appropriately, and unpackaged when hoping to look at it again.
- Failure scenarios
  The portion we are adding to the software will fail if the patient data is (1) not correctly stored in the database, and (2) not brought up for the hierarchy log. If either of those two scenarios happen the patient information will not be retrievable causing real-life consequences. Additionally, if we add anything that causes a big bug in the software generally speaking, we assume another failure scenario is that the version we release would prevent the entire working program from being launched on-site, another real-life consequence scenario. From a business perspective this is financially damaging as well as productively horrible. We want to ensure a good, working product that does not threaten lives, but ONLY benefits the physicians and patients in a positive way.
- Conditions that lead to errors and failures
  Conditions that would lead to this are rushing developers leading to code that is not clean. Overloading developers with tasks they were not skilled for or too busy to take on. Unclear direction from the sponsors, or limited resources to create the desired outcome.
- Limitations
  Limitations on the success of this project will be dependent on the student's time allotment to development, the power of their machines (similarly to the on-site physicians), access to Andy and Sean for correctional help on the technical aspect, and likewise Sarah for overall purpose and direction.

- Presentation Layer
    - User requirements

      The interface needs to be highly intuitive, in the case that english is not the user's primary language. This means avoiding pop-ups, tabs, or any other confusing design patterns as well as avoiding text labels in favor of visual cues.
    - UX changes

      The trauma sheet will have its own page accessed via the navbar. The main interactive components that must be included are; the body diagram, the problem list, and the required information.
    - UI changes

      Ideally, the color scheme and layout of the fEMR software will be updated for a cleaner, more high contrast look.
    - Links to Wireframes:
        - LoFi Wireframe Prototype
        - HiFi Wireframe Prototype
    - Mobile vs Web concerns

      The only time mobile design must be accounted for is when it will be accessed via chrome on an ipad when computers are not accessible. This does not present too many design challenges, but all interactive choices take into consideration that the user may be using a mouse, trackpad, or finger/stylus. This means all design choices must be focused on ease of navigation.
    - UI states & Error handling - Will be clarified during the high fidelity design phase
- Other questions to answer
    - What are the limitations of the solution?

      The solution will be bound by the limitations of technology, compared to an analog solution. As opposed to having the freedom to write anything anywhere on paper, the digital trauma sheet input will be limited to what we allow as developers.
    - How will it recover in the event of a failure?

      Worst case scenario the users can always revert to using an analog trauma sheet. However, we would like to avoid this, so we will be designing our 3 elements to avoid interdependence as much as possible.
    - How will it cope with future requirements?

Our strategy for requirements is to determine them as early as possible. If something comes up late into the development process we will accommodate it, but hopefully by using prototypes and performing regular user testing we can uncover requirements as early on as possible. By doing this and writing clean code with high cohesion/low coupling we should be able to minimize the amount of extra work new requirements create.

c. Test Plan

- How the tests will make sure user requirements are met for the next team: The next team should be making test specifications for the user requirements and writing integration tests to test out all of the features we add in the femr app.
- Unit tests: For every component, the next team should plan to make unit tests to make sure the components work as expected. Since the project has modular components, we expect that it would be easy/moderate to write the unit tests.
- Integrations tests: For the integration tests, the next team will want to test out the integration of the trauma page with the rest of the application. Also the next team will want to test integration of our components on the trauma sheet and make sure it works as expected when all of them are tested at the same time.
- QA: The next team should plan to test out every pull request before they merge it to the tremr-branch branch once they begin implementing backend functionality. That way they will have more sets of eyes and fewer bugs in our code.

d. Monitoring and Alerting Plan

- Logging plan and tools

  When working on the testing portion of the project, in order to be successful, the team will log any testing tasks they do on Jira. This will allow them to see how much time they are spending on certain testing tasks. Logging the tasks will allow the team to see the estimates and the actual time spent.

- Monitoring plan and tools

  The monitoring plan for the project will include evaluating and gathering feedback on the status of the testing during a sprint. Such information will be crucial in the development process of the project. This will allow the team to see which areas we can improve on and what parts we are successful at.

- Metrics to be used to measure health

The metric that will be used to measure the overall health of the project from a testing standpoint, is making sure that all tests that have been written are passing before pushing any code. While the code was not very testable, as it was mostly a visual facade, testing will be a focus for the next team that picks up the project.

- Alerting plan and tools

  We used our current communication channel as a means to alert others of our plans or any issues that may arise. We did not look into any specific alerting tools but it is something the next team can look at in the future.

e. Release / Roll-out and Deployment Plan

- Deployment architecture:

  Our deployment uses AWS Codebuild service to build and deploy the application. We code reviewed every pull request on github and then safely merged them to the tremr-branch branch.

- Deployment environments:

  The deployment environment consists of Github, AWS Codebuild, and everything required for AWS Codebuild which includes the buildspec.yml file and Docker.

- Phased roll-out plan e.g. using feature flags:

  We worked on the tremr-branch branch which was a development branch and merging our pull requests into that branch. Since it is a development branch, we used that for testing and merging our pulls more frequently. When it is time to merge the tremr-branch branch to super-femr branch, the team will be communicating with the official femr team and the other student teams.

- Plan outlining how to communicate changes to the users, for example, with release notes:

  We were following the same template with current femr's release notes to communicate changes to the users. We were communicating with the fEMR team to get permission to publish a new release.

f. Rollback Plan

- Detailed and specific liabilities:

  Since we worked with the tremr-branch branch, we expected that there will be no production level errors, but in case of bug or error release in the tremr-branch branch, we planned to revert the commits with bugs using git bisect and debugging. However, in case of bugs slipping into femr master branch, we planned to rollback as soon as possible before the kits are deployed to disaster sites.

- Plan to reduce liabilities

We planned to not make tech debts so that way we had clean code and avoided code smells. Also by following clean coding principles and testing strategies, we planned to reduce any problems and liabilities.

- Plan describing how to prevent other components, services, and systems from being affected

  Ideally for every component, plan to make unit tests and integration tests to test the whole trauma intake system. Also we planned to make the trauma components modular, so in case any of the trauma features run into errors, it will not take the whole kit down.

g. Alternate Solutions / Designs

- Short summary statement for each alternative solution

There are three main options: Analog documentation (the current solution), building personalized trauma response software within the femr ecosystem (the proposed solution), or paying for pre-existing medical tracking software. Some of the top EMR system solutions on the market right now are Epic, Cerner, and Meditech.

- Pros and cons for each alternative

While externally sourced software solutions would likely be much more reliable and developed and have a more in depth support team.

- Reasons why each solution couldn't work

However, the alternatives could not handle the specific requirements necessary for integration into the femr environment/kit structure.

- Ways in which alternatives were inferior to the proposed solution

Our proposed solution will be built within the framework of the existing femr software for flawless integration. We will have the ability to design a solution with the exact interactive and technical requirements that the femr team needs.

4. Further Considerations

a. Security considerations

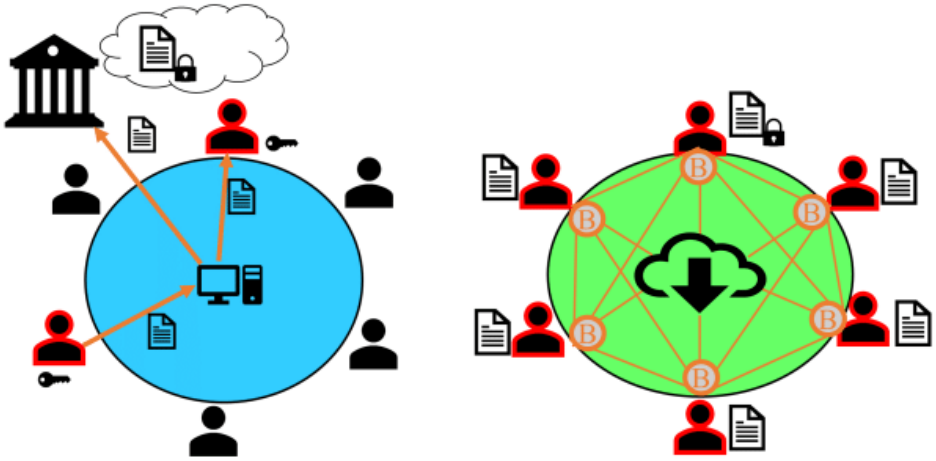
- What are the potential threats?

One of the potential threats is that the database of patients is a likely target for hackers to steal this information from fEMR. Hackers can also gain access to the nurses or first responders information that was logged when they created an account.

Another potential threat is for hackers to completely down the whole system which would disrupt the first responders ability to log the patients information.

- How will they be mitigated?

The way that such threats will be mitigated is simply how fEMR will be built. fEMR will be built using the Fennel Protocol (fEMR's "on-chain") that will allow fEMR to define their own blockchains. The benefit of using blockchain is that information is not kept in a single location, instead the blockchain is spread across different networks (see diagram below). When information is spread across different/multiple networks, hacking becomes much more difficult. If a hacker does get a hold of a blockchain, only that particular copy of the blockchain would be compromised instead of the whole system.



Centralized Framework Decentralized Framework

5. Success Evaluation

a. Impact

- Security impact
  - By having a digital trauma sheet used by devices such as tablets and computers, patient intake is stored on local femr kits then to the server when possible. By storing digital records, it greatly reduces the security risks that can occur from the loss or damage of the physical paper records.
  - By transitioning to the digital storage of records, the possibility of unauthorized access from threat actors and other attacks also becomes possible.

- User performance impact
  - There is a great boost in performance that will occur by transitioning to digital records. Physicians and nurses can intake patients in a quicker more accurate manner which is especially important in trauma response thereby putting more focus on the wellbeing of their patients. Beyond just patient intake, medical staff will be able to look up patient info quickly without the risk of not finding the correct file in time.

- Cost impact
    - Costs can include
        - hardware (femr kits, computers, tablets, etc)
        - developer costs
        - server costs
- Impact on other components and services
    - By storing patient records digitally, the pharmacy can be able to access the patient's file and see which medicines have been prescribed.

b. Metrics

- List of metrics to capture
    - Which fields are used the most (should then be prioritized in visual hierarchy)
    - Avg amount of time it takes first time users to understand the diagram/problem list without external explanation
    - avg time it takes to intake a patient
- Tools to capture and measure metrics
    - measure completion rate by assigning tasks to a test team
    - user testing
    - user surveys

6. Work

a. Work estimates and timelines

- Timelines and estimates were set by stand-up meetings prioritizing both our backlog tasks as well as ideas and features to implement as requested by the customers with an MVP-first way of thinking.
- These tasks were chosen based on sprint timeline and estimation.
- To ensure we completed a task it was critical we have mid-sprint deadlines (at least when necessary) to assure we are making progress soon enough with time to ask questions.

b. Prioritization

- Tasks are categorized by urgency to create a MVP each viable sprint, as well as the thinking of "what is the highest reward and lowest cost?" To do these tasks first (as long as they're not dependent on being sequential), then the less-necessary but feature-rich tasks after.

c. Milestones

- Significant checkpoints seemed to align with our sprint deadlines.. This is documented in JIRA with our burndown charts. Deadlines are hard until something comes up.

- Metrics to indicate milestone passing were (a) did the priority tasks assigned get accomplished. Each person got 1-2 ‘highlighted’ tasks to definitely finish. We assigned ‘reach’ goals thereafter, which several of them we discuss if they need to be done (just later), or if they’re stretch goals.

d. Future work

- We were able to get the front-end aspect of the trauma implemented, the future team that continues with this project will have to connect a few of the input fields and save the image of the drawing canvas to the database.
- Make the current trauma sheet print friendly so that a nurse can print the information they logged of the patient and hand that information to a hospital that the patient is being transferred over.
- Create a separate page that can display the patient information from the trauma sheet. This could also possibly be displayed on the “Medical” tab.
- Make the trauma sheet responsive so that if a user resizes the page it resizes everything correctly.
- User testing & subsequent UX product revisions
- Finish any remaining tasks from the current year.

7. Deliberation

a. Discussion

- Our team has an understanding on how the trauma sheet should function and a high fidelity prototype to base our work off of as we continue to develop our product. Moving forward the discussion with the next team needs to be focused on linking frontend and backend, and optimizing database alterations.

8. End Matter

A. References

- https://stackoverflow.blog/2020/04/06/a-practical-guide-to-writing-technical-specs/

B. Acknowledgments

- Professor BJ Klingenberg - Cal Poly SLO CSC/SE Department
- Professor Bruno Da Silva - Cal Poly SLO CSC/SE Department

C. Useful links

- Readme
- Gitbook
- JIRA Backlog
- Figma: Lofi, Hifi
- Github
  - Base branch: super-femr
  - Head branch: tremr-branch
  - All changes