CSCI 370 Final Report

Keeping Labor Safe 1

Zane Beasecker
Rachel Jones
Delia McKee-Gresham
Mark Vosseteig

Revised December 3, 2023

CSCI 370 Fall 2023

Dr. Christopher Painter-Wakefield

Table 1 - Revision history

Revision	Date	Comments
New	9/2/2023	Added introduction, functional requirements, non-functional requirements, risks, and definition of done.
Rev – 2	9/19/2023	Added team profiles and system architecture.
Rev – 3	10/18/2023	Added software test and quality and ethical considerations.
Rev - 4	11/9/2023	Added future work and lessons learned.
Rev - 5	12/4/2023	Added project completion status, acknowledgements, and finalized report for review.
Rev - 6	12/8/2023	Updated report with peer feedback, final submission

Table of Contents

I. Introduction

In 2019, UNICEF reported 2.4 million neonatal deaths and 2.0 million stillbirths. However, 75% of newborn deaths, plus the majority of maternal deaths and stillbirth, can be prevented with high-quality care [1]. The current methods of screening fetal and maternal health are inadequate and are not diagnosis tools. With an improved tool for health screening, used before, during, and after labor, diagnosis will improve, and with that, infant and maternal mortality rates will decrease.

Keeping Labor Safe (KLS) is a start-up medical technology company that is developing a technology called the Fetal Reserve Index (FRI). This technology will allow medical professionals to monitor fetal health in real time, which will be an improvement in Electronic Fetal Monitoring (EFM). This new approach, developed by Dr. Mark Evans M.D., will be able to take in data from the monitoring device and quantitatively score the health of the fetus based on maternal, fetal, and obstetrical risk factors in order to obtain a better picture of the health of a pregnancy than the current method of screening [2].

In its initial state, the web application front end had been created and built to accept CSV files in place of EFM monitoring. These CSV files contain data that an EFM would produce, such as fetal heart rate and uterine activity. Our main goal for this project was to expand this EFM simulation flow. We set out to store the CSV data as it was not saved previously, assign each case to the appropriate patient, visualize the data in a graph on both the “Monitoring” and “Reports” tabs, add more roles for different levels of access, and allow the manual addition of fetal, maternal, and obstetrical risk factors during data collection. With these additions to the existing application, the FRI will allow medical professionals to improve the safety of labor and delivery across the globe.

II. Functional Requirements

1) Create an API that connects all relevant components of the application
   a) Record the tracing
   b) Compute the score
   c) Generate and store reports
2) Add patient functionality and additional usability features
   a) Add new patient through UI
   b) Add patient history and risk factors
   c) Associate report with patient
3) Add features and capabilities listed in “Application Requirements” below to the web app created by the CS Mines fall 2022 and summer 2023 field sessions. This will turn the Fetal Reserve Index algorithm into an Intelligent Cloud Platform and Bedside Web Application. Currently the application has 3 user types (Administration, Clinical, and Decision Support) and will work on the current releases of the major web browsers.

   Application Requirements:

   - Refactoring the code to make it easy to internationalize the product.
   - Dynamically update on-screen reporting as data is input.
   - Standardized Reports
   - Fluid Graph with Trendline
   - Front End Improvements
     - Improve the design
     - Improve the Intuitiveness
     - Add reports
   - Reset/Forgot Password capabilities
   - Data input through intuitive UI
     - Live/Manual/Automated Input of EFM Data
     - Live/Manual/Automated Input of EHR/Risk Factor Data
     - Live/Manual/Automated Input of Notable Events

    - Drop Down Menu for Maternal, Obstetrical and Fetal Risk Factors
    - UI automatically prompts for inputs at predetermined and variable intervals
    - Dynamically update on-screen reporting as data is input
    - Multiple Concurrent viewing capabilities: Real Time Sharing of Data
    - Remote Administrative access for diagnostic support

III. Non-Functional Requirements

- The software and database must be HIPAA compliant to the best of our ability. This will have to be fully completed and certified by a professional in that field.
- Must be built to allow for expansion.
- Must be compatible with all major web browsers such as Chrome, Firefox, Safari, and Edge on Windows.
- Back-end must be built in Python Flask.

IV. Risks

- Data Security
    - Mitigation Plan: We will try to secure the databases to the best of our ability, but KLS will have to hire a consultant to certify that the program is HIPAA compliant.
- Test Data
    - Mitigation Plan: Since the amount of test data we have access to is very limited, we will have to work within the bounds of the provided data and create our own program to simulate reading from an EFM.

V. Definition of Done

- Minimal useful feature set
    - API sends web requests at specified increments of time
    - Front-end displays new data from the EFM
    - Passes all unit tests
- Tests the client will run before accepting software
    - Connection to the EFM
- How the project will be delivered
    - Source Code (GitHub)
    - React App
    - Azure Repository
    - Environment Variables

VI. System Architecture

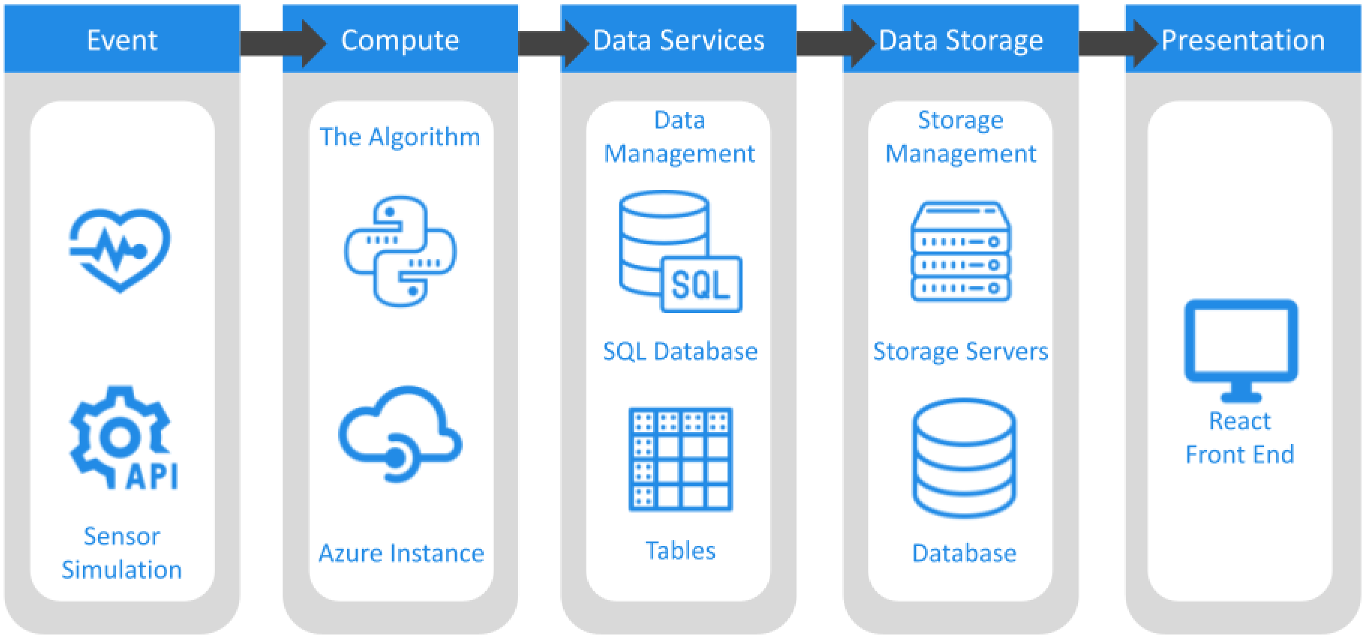


Figure 1A - Logic Flow

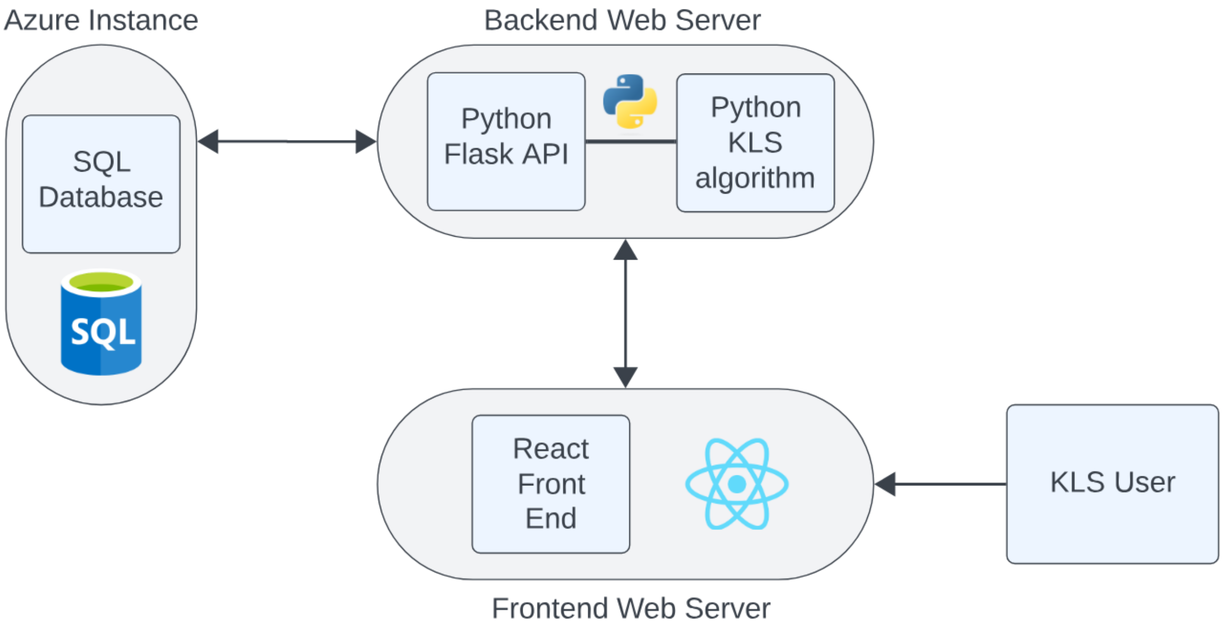
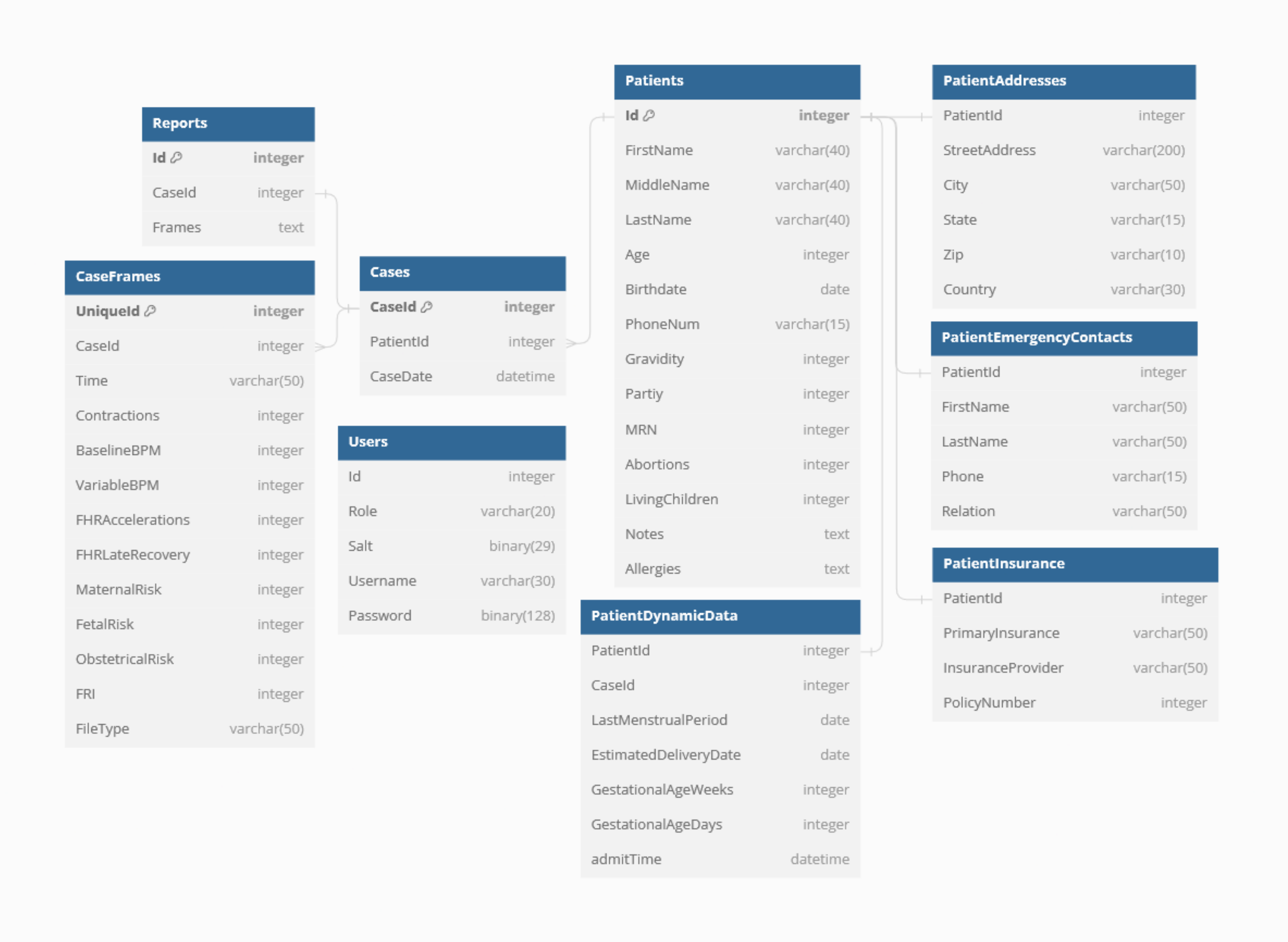


Figure 1B - Architecture Diagram

Figures 1A and 1B depict the logic flow and architecture of the application. The user will interact with the frontend web server, which is supplied with data from the backend web server. The backend of the application connects to the SQL database through the Flask API. All of these components communicate with each other to ensure the best possible user experience.



Reports	
Id	integer
CaseId	integer
Frames	text

CaseFrames	
UniqueId	integer
CaseId	integer
Time	varchar(50)
Contractions	integer
BaselineBPM	integer
VariableBPM	integer
FHRAccelerations	integer
FHRLateRecovery	integer
MaternalRisk	integer
FetalRisk	integer
ObstetricalRisk	integer
FRI	integer
FileType	varchar(50)

Cases	
CaseId	integer
PatientId	integer
CaseDate	datetime

Users	
Id	integer
Role	varchar(20)
Salt	binary(29)
Username	varchar(30)
Password	binary(128)

Patients	
Id	integer
FirstName	varchar(40)
MiddleName	varchar(40)
LastName	varchar(40)
Age	integer
Birthdate	date
PhoneNum	varchar(15)
Gravidity	integer
Partiy	integer
MRN	integer
Abortions	integer
LivingChildren	integer
Notes	text
Allergies	text

PatientDynamicData	
PatientId	integer
CaseId	integer
LastMenstrualPeriod	date
EstimatedDeliveryDate	date
GestationalAgeWeeks	integer
GestationalAgeDays	integer
admitTime	datetime

PatientAddresses	
PatientId	integer
StreetAddress	varchar(200)
City	varchar(50)
State	varchar(15)
Zip	varchar(10)
Country	varchar(30)

PatientEmergencyContacts	
PatientId	integer
FirstName	varchar(50)
LastName	varchar(50)
Phone	varchar(15)
Relation	varchar(50)

PatientInsurance	
PatientId	integer
PrimaryInsurance	varchar(50)
InsuranceProvider	varchar(50)
PolicyNumber	integer

Figure 2 - Database Schema

Figure 2 shows the architecture of the SQL database used to store the application data. The Patients, PatientAddresses, PatientEmergencyContacts, and PatientInsurance tables are connected through the ID assigned to each patient at creation. These three tables store related information for each patient, simplifying the main Patients table. Additionally, this ID links the Patients table to any relevant entries in the Cases table. The Cases table is connected to the CaseFrames and Reports table through a separate ID, the CaseID. The CaseFrames table stores each "frame" of data collected on the monitor tab. The CaseID then links all of these frames together in the Cases and Reports tables. The Users table stores all relevant information for a User, and is therefore not linked to any of the other tables.

VII. Technical Design

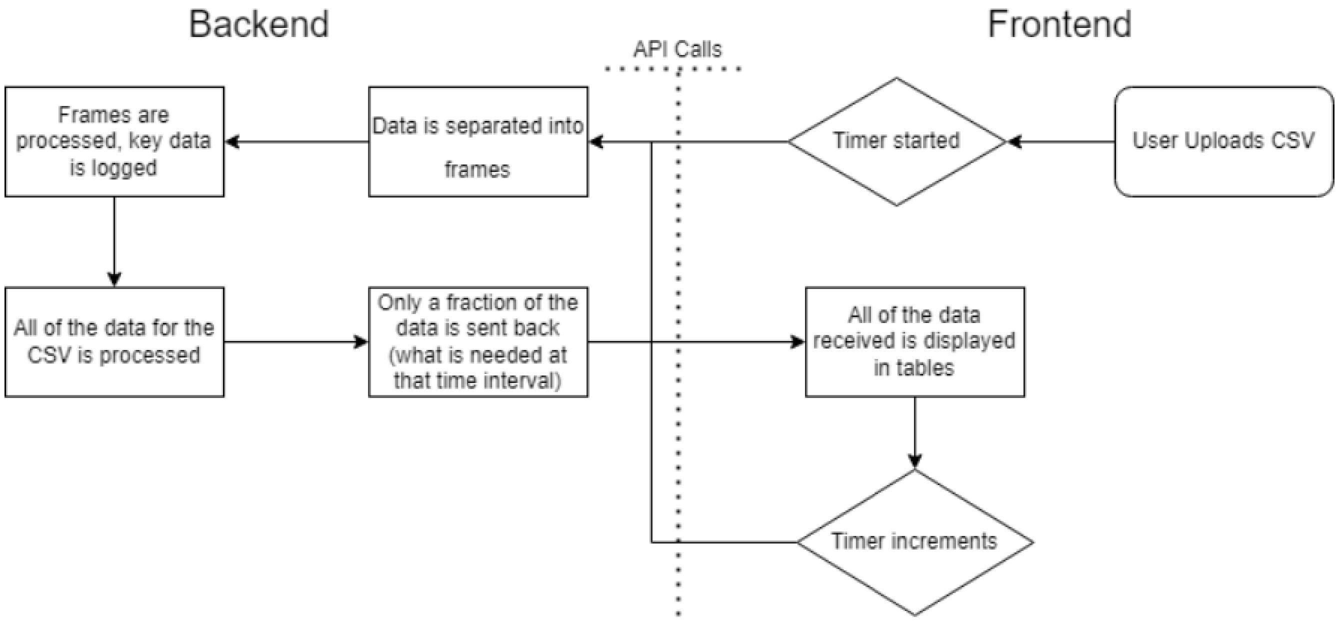

Backend
Frontend
API Calls
Frames are processed, key data is logged
Data is separated into frames
Timer started
User Uploads CSV
All of the data for the CSV is processed
Only a fraction of the data is sent back (what is needed at that time interval)
All of the data received is displayed in tables
Timer increments

Figure 3 - Connection Diagram

Figure 3 shows the processing flow for the csv upload on the live monitoring tab, as implemented by the previous team. The old live monitoring system did not have any saving functionality and was not capable of receiving updated risk factors mid-simulation. With this flow, it would not have been possible to implement these features, since all of the csv data is processed between each increment of the timer, even though only a very small fraction is sent back. To avoid this, the whole live monitoring flow was reworked by our team. The new flow diagram is depicted below.

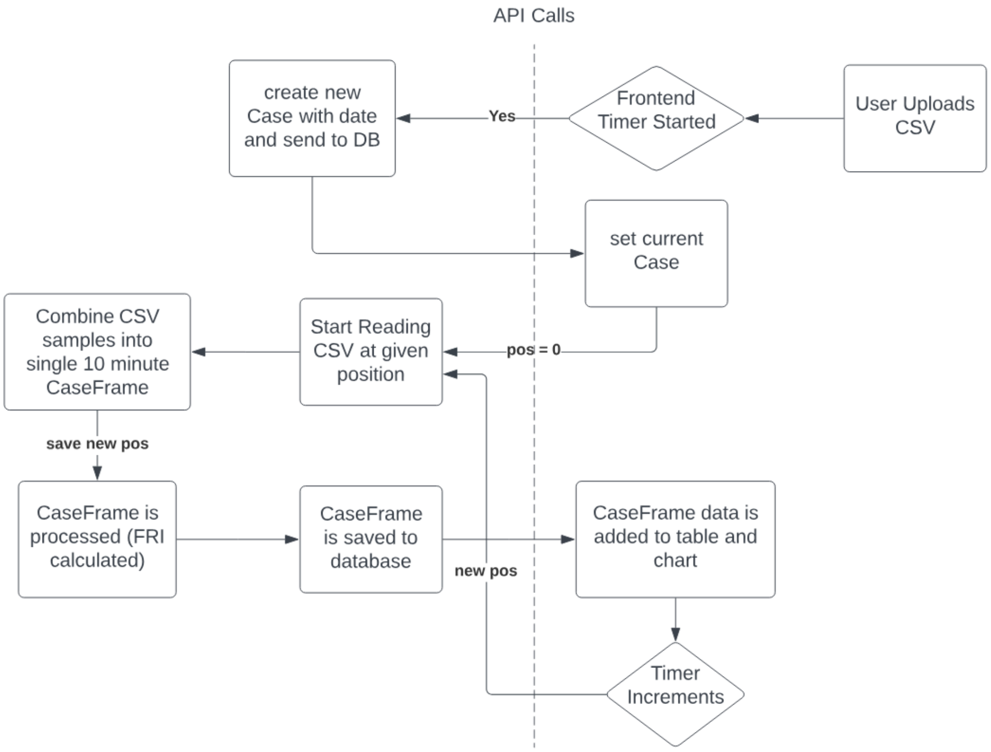

API Calls
create new Case with date and send to DB
Yes
Frontend Timer Started
User Uploads CSV
set current Case
Combine CSV samples into single 10 minute CaseFrame
Start Reading CSV at given position
pos = 0
save new pos
CaseFrame is processed (FRI calculated)
CaseFrame is saved to database
new pos
CaseFrame data is added to table and chart
Timer Increments

Figure 4 - API Calls

The figure above demonstrates the new logic flow for the live monitoring system. Now, after the frontend timer has been started, a Case is created and saved to the SQL database. The caseId is then stored in the frontend. After that, the backend starts reading the uploaded CSV at the start of the file. Instead of reading the entire CSV, the backend processes only enough samples to create a single 10 minute CaseFrame. Once this CaseFrame data has been collected, the data is processed and the CaseFrame is saved to the database with the corresponding caseId. Finally, the CaseFrame data is sent to the frontend for display and the timer increments, causing the next frame to be read starting at the position in the CSV where the last frame finished. This ensures that only the data that is needed is processed between each timer increment, and previous data is not recalculated with any new risk factor data.

VIII. Software Test and Quality

Testing

User interface testing will be tested on different devices using various web browsers by all members of the team. However, user acceptance testing will be difficult to implement during this phase of development. We are unable to certify HIPAA compliance and unable to connect to a real EFM in order to test the transfer of data. We will work to make sure that our implementation protects the data to the best of our ability and ensure that the testing program is as similar to the EFM as possible.

Throughout the development process, in-depth testing was conducted on both the frontend and backend applications. The findings indicate that, in nearly all instances, frontend pages load almost instantly, providing accurate information. Notably, the FRI application demonstrates operational functionality on popular browsers such as Chrome, Firefox, Edge, and Safari. The errors that were initially identified on the frontend application at the project's outset have been successfully addressed and resolved. However, it is important to note that true usability tests could not be performed due to the unavailability of an Electronic Fetal Monitor, preventing the transmission of real data to the web application. Additionally, the application lacks HIPAA compliance, rendering it unsuitable for use in medical settings. Despite these limitations, the overall performance testing results showcase the efficiency and effectiveness of the application's frontend and backend components.

Table 2 - Testing

Test Purpose	Test Description	Environment, Setup, and Tools	Acceptability Threshold	Edge Cases
Application is compatible with major web browsers	The web app functions as expected when using the web browsers specified by the client	Google, Firefox, Edge, and Safari	Pass/Fail	Older versions of browsers may be unsupported
Dynamically update on-screen reporting as data is collected	The API will send web requests every 20 seconds which should update the reports page with the newly collected data	Platform UI, Azure SQL Database, Python Flask Test	Pass/Fail	
Standardize reports	The reports page	Platform UI, Azure		

	should display the data contained in the database to the correct users	SQL Database		
Role implementation	Users can see only the information they should see as specified by their role	Platform UI, Azure SQL Database		
Reset/forgot password flow	Assure screen looks appropriate on desktop, tablet, and mobile phone, and password value changes after flow is complete.	Platform UI, Azure SQL Database, Python Flask Test	Text does not warp, all components are visible, and website flow is smooth on all devices, and password value changes after flow is complete	
Risk factor dropdown	Assure screen looks appropriate on desktop, tablet, and mobile phone,	Platform UI, Azure SQL Database	Text does not warp, all components are visible, and website flow is smooth on all devices, only one risk factor per category	
API integration		Azure SQL Database		

Quality

For each change made by any member of the team or pair of members of the team, the changes have been accompanied by an explanation of the changes, the problem they solve, and why the changes are necessary. The changes were then approved by all members of the team who were not directly involved with the implementation of the changes. This ensures code quality through code reviews. These code reviews keep in mind code metrics such as readability, reliability, testability, clarity, efficiency, extensibility, and documentation in order to keep the code that we will deliver up to the standards of Keeping Labor Safe.

In order to simulate the data received from the sensors of the EFM, a program will be created by another team that will take the place of the upload of data in the format of a CSV file. This will connect to the API, which we will also build and test. To make sure the application, the databases, the API, and the testing program all work well together, our team used integration testing along with the individual testing of each component.

IX. Project Ethical Considerations

As software engineers, we must work to ensure that this application is of good quality and meets the expectations of the client. Since sensitive medical data will be handled in the application, security is a main concern for this project. As this application has been developed, we have tried to ensure to the best of our ability that patient data is encrypted and stored securely. These are some ACM/IEEE principles that are pertinent to this project:

- 1.01 Accept full responsibility for their own work

- 2.03 Use the client’s authorized property with the client's knowledge and consent.
- 2.06 Identify and report to the client or the employer if, in their opinion, a project is likely to raise issues.
- 3.08 Ensure that specifications for software on which they work have been well documented, satisfy the users’ requirements and have the appropriate approvals.
- 3.10 Ensure adequate testing, debugging, and review of software and related documents on which they work.
- 3.11 Ensure adequate documentation, including significant problems discovered and solutions adopted, for any project on which they work.
- 3.12 Work to develop software and related documents that respect the privacy of those who will be affected by that software.
- 3.14 Maintain the integrity of data, being sensitive to outdated or flawed occurrences

One of the requirements of this project, however, is for the Fetal Reserve Index to be HIPAA compliant. Our team was unable to provide true HIPAA compliance, though we have tried to encrypt the data to the best of our ability. To that end, these are some ACM/IEEE principles that were most in danger of being violated:

- 3.12 Work to develop software and related documents that respect the privacy of those who will be affected by that software.
- 3.14 Maintain the integrity of data, being sensitive to outdated or flawed occurrences

Applying Michael Davis’s harm test and professional test to the project, the Fetal Reserve Index should pass both. The application poses the threat of exposing sensitive medical data, but given that our team developed with good practice and that HIPAA compliance will be certified by a professional, the data should be protected and the public, patients, and medical professionals will not be compromised by the app. Additionally, the application’s purpose is to provide more accurate readings of fetal health. The computations that will be used in the FRI have been tested by medical professionals, and assuming sound implementation, the application should not cause harm to any of the stakeholders. As for the professional test, so long as we have done our best to be transparent with our client and deliver a well-developed and well-tested product, the Software Engineering code of ethics will have been followed.

If the software quality plan was not implemented properly, the health of the patients could be at risk. Improper readings of fetal health could negatively impact fetuses, mothers, and even the medical professionals who make decisions based on the readings. Additionally, sensitive medical data will be used in the app, and if we do not encrypt the data and test data transfer properly, this data could be leaked, or at the very least, it could make it difficult for anyone trying to certify HIPAA compliance.

X. Results

While most of the functional requirements that were given to us were completed, some were unable to be completed. Specifically, HIPAA compliance was unable to be implemented since we lacked the time, skills, and should not undertake the legal responsibility. The application will have to be sent to a HIPAA compliance specialist in order to complete this requirement. Once this is done, the application can be connected to an EMR.

The app will also have to be able to be connected to an EFM and the CSV upload deprecated. This will require access to an EFM for proper testing.

The last items to be completed are new user role verification, meaning that users will have to provide evidence that they can legally view the requested data, e.g. ensuring that users who register as doctors are truly practicing doctors in the state they register in. Additionally, we lacked the time to allow risk factors to be viewed and modified on the patient records page.

XI. Future Work

Currently, the application uses a CSV file upload to generate data reports for patients. When the application is ready for production, the reports should be generated directly from the Electronic Fetal Monitor. Keeping Labor Safe 2 has been working on sending data to the application using bluetooth, but their EFM bluetooth data simulator has not been integrated into the FRI. Eventually, that program will also be replaced by a true EFM which will require attaining an actual EFM for testing. Additionally, all of the features listed below will need to be implemented before the FRI will be ready for use in a medical setting.

- Create new user role verification
  - When creating a new account, the user will have to prove that they are a working professional within the healthcare industry. We did not have the time or skills to be able to take on this task.
- Connection to an Electronic Fetal Monitor
  - We do not have access to an EFM, but previous teams have implemented testing by uploading CSV files containing test data that mimics an EFM. However, the data is input all at once in a file upload, which is not how data collection from the EFM will function. Keeping Labor Safe 2 has been working on creating a program that will send data from a raspberry pi to the web application that will simulate EFM data until such time that an EFM can be connected.
- HIPAA Compliance
  - HIPAA compliance will need to be certified by an expert or team of experts on HIPAA compliance. We are unable to certify HIPAA compliance due to time, skills, and legal responsibility.
- Connection to Electronic Medical Records
  - Connection to an EMR will require HIPAA compliance.
- Allow risk factors to be added on patient records page

XII. Lessons Learned

It is very difficult to add new features when existing documentation is lacking. Our team initially struggled to understand how the system currently functioned, because the previous groups had not created much documentation. The team encountered errors when trying to run the existing code, which had not been documented. It would have been very helpful if the previous teams had documented these issues. Information from previous groups also seemed to be conflicting. Our team had to spend a great deal of time trying to work out how the backend and frontend actually functioned.

We also learned that your client contact may not have a technical understanding of the project. Our team struggled to get access to the correct resources for a long period of time at the start of the project, mainly because our client contact did not fully understand what we needed to get started on the project. Our team realized over time that we needed to be very specific with our client contact and explain everything, instead of assuming that the client contact would understand.

Trying to avoid temporary fixes that will be changed is another lesson learned. A few times, our team quickly implemented solutions to show to our client that we knew we would have to change in the future. However, later down the road this made it more difficult to implement new features. Unless it is absolutely necessary, it is better to spend the time creating the right solution instead of a quick one that works for the time being. This also makes it much easier for other teams in the future.

Lastly, maintaining consistency throughout your codebase is extremely important. The previous teams that had worked on our project did not follow this rule. Communication between the backend and frontend was not done through a standard method in all places. This made it difficult to create new data types and methods for transferring information.

XIII. Acknowledgments

We would like to thank Dr. Christopher Painter-Wakefield for his help as our advisor on this project. Without his help, we would not have been able to accomplish the completion of our project. We would also like to extend our gratitude to the Keeping Labor Safe 2 and Keeping Labor Safe 3 teams for their collaboration.

XIV. Team Profile

Zane Beasecker

Class: Senior

Major: Computer Science + Robotics and Intelligent Systems

Hometown: Piedmont, OK

Work Experience: Development Intern at Lopez/Dorada, Mines CSCI TA

Hobbies: Music Production, Anime, Piano

Rachel Jones

Class: Senior

Major: Computer Science + Robotics and Intelligent Systems

Hometown: Thornton, CO

Work Experience: Application Development Intern at U.S. Engineering

Hobbies: Ballroom dance, learning languages

Delia McKee-Gresham

Class: Senior

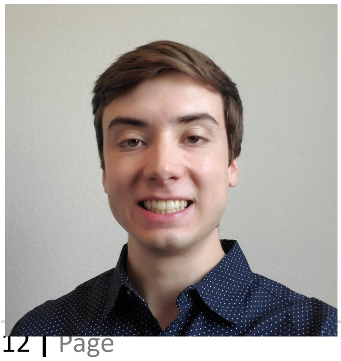
Major: Computer Science + Robotics and Intelligent Systems

Hometown: Boulder,CO

Work Experience: Software Engineering Intern at NCAR

Hobbies: Gaming, fencing

Mark Vosseteig



Class: Senior

Major: Computer Science + Space

Minor: Aerospace Engineering

Hometown: Fort Collins, CO

Work Experience: Lockheed Martin Intern, CS Research Assistant

Hobbies: Piano, gaming, snowboarding

References

[1]“New UN report highlights child mortality rates for the Pacific,” *www.who.int*. https://www.who.int/westernpacific/about/how-we-work/pacific-support/news/detail/23-10-2017-new-un-report-highlights-child-mortality-rates-for-the-pacific

[2] Mark I. Evans, MD; David W. Britt, PhD; Shara M. Evans, MSc, MPH; Lawrence D. Devoe, MD, *“*Improving the interpretation of electronic fetal monitoring: the fetal reserve index,” *American Journal of Obstetrics and Gynecology*, vol. 228, no. 5, May. 2023. Accessed: Dec. 4, 2023. doi:https://doi.org/10.1016/j.ajog.2022.11.1275. [Online]. Available: https://www.ajog.org/article/S0002-9378(22)02163-9/fulltext

Appendix A – Key Terms

Include descriptions of technical terms, abbreviations and acronyms

Term	Definition
KLS	*Keeping Labor Safe - A medical technology startup company and our client.*
FRI	*Fetal Reserve Index*
EFM	*Electronic Fetal Monitor - A medical device used to monitor the fetal heartbeat and uterine activity.*
EMR	*Electronic Medical Records - Medical records that are stored digitally.*
HIPAA	*Health Insurance Portability and Accountability Act*
UI	*User Interface - the means by which the user and a computer system interact, in particular the use of input devices and software*