Kaleida Health Application: A Platform for Informed Patient Care and Doctor Collaboration

Christopher Dushkoff
State University of New York at Buffalo
crdushko@buffalo.edu
Advised by Dr. Alan Hunt
State University of New York at Buffalo
ahunt@buffalo.edu

Abstract – **The internet is often the first place patients go to for medical information, but the sheer volume of unverified or inaccurate content can lead to misunderstandings, unnecessary anxiety, or even harmful decisions. Many patients leave medical appointments with unanswered questions, which may lead them to seek answers online.**

The Kaleida Health Application addresses this challenge by providing patients with a trusted source of curated information directly from healthcare professionals. It also offers direct communication between patients and their doctors which allows them to clarify any doubts or seek additional guidance for their conditions. This paper outlines the design, implementation, and functionality of the application. It highlights its features, issues, and direction for future development.

I. INTRODUCTION

In an era where access to information is instantaneous, patients often turn to the internet for answers to health related questions. While this can be informative, there are also significant challenges. The accuracy and reliability of online medical information can often be questionable. The trend of "Dr. Google" has led to widespread misinformation, unnecessary anxiety, and improper self diagnosis, which results in poor healthcare decisions.

II. IMPORTANCE OF THE PROBLEM

The internet is full of unverified medical content and misinformation. By delivering curated and accurate resources directly from healthcare providers, the Kaleida Health Application serves as a reliable alternative to the overwhelming content available online. This will help patients improve their understanding of their conditions and how to treat them.

III. FEATURES AND FUNCTIONALITIES

The Kaleida Health Application offers the following main functionalities for patients:

- **Viewing Diagnoses**: Users can view all the diagnoses that have been assigned to them by their doctor and all information related to their conditions
- **Sharing Diagnoses**: Patients can share information about their diagnoses with a family member or trusted individual so that they can assist with taking care of them.
- **Chatting with Doctor**: Users can reach out to their doctor directly through a built in chat if they have any questions about their conditions. The chat is also automatically translated to the user's preferred language.

Doctors will have these main functions:

- **View Patients**: Doctors will be able to view all patients assigned to them. They can also search and find patients by location.
- **Assign Diagnoses to Patients**: Doctors can assign diagnoses to patients which will include accurate information regarding their condition in the form of documents and videos.
- **Chat with Patients**: Doctors will be able to chat with patients and answer their questions. Their responses will automatically be translated to the patient's preferred language

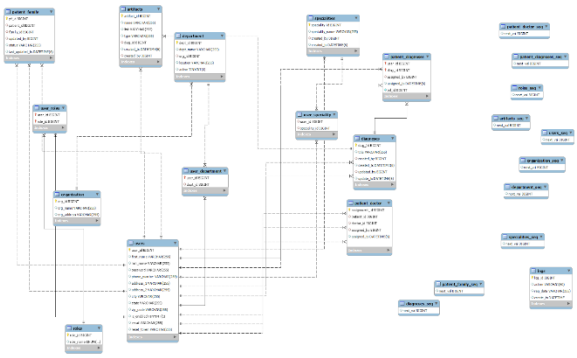
Admins will be able to do the following:

- **Create and Manage Accounts:** Admins will be able to create new accounts and make changes to existing accounts such as updating their name or location
- **Create and Manage Diagnoses:** Admins can create new diagnoses and attach relevant documents or videos. They can also update or delete existing diagnoses.
- **Assign Doctors to Patients:** Admins will be able to assign doctors to patients or remove doctors that are already assigned to a patient
- **Manage Doctor Groups:** Admins can add or remove doctors from doctor groups for chat

IV. SYSTEM ARCHITECTURE

- Frontend: Built with React for the web interface
- Backend: Spring Boot handles all API and authentication requests.
- Database: MySQL database is used for storing account information, diagnoses, chat messages, etc.
- Deployed on AWS Elastic Beanstalk
- S3 Bucket for storing documents for Diagnoses

V. DATABASE STRUCTURE



VI. USER INTERFACES

The Kaleida Health Application includes many different interfaces between the 3 different user types: patient, doctor, and admin.

Patient View:

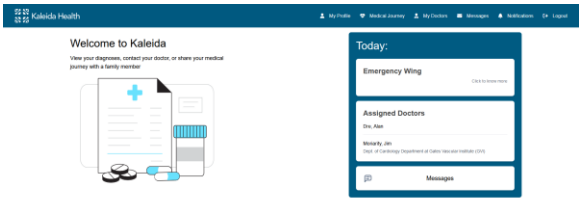


Fig. 1. Patient Dashboard

From the dashboard patients can see their location, assigned doctors, and messages. They can click on 'My Profile' to edit their profile such as their preferred language or contact information.

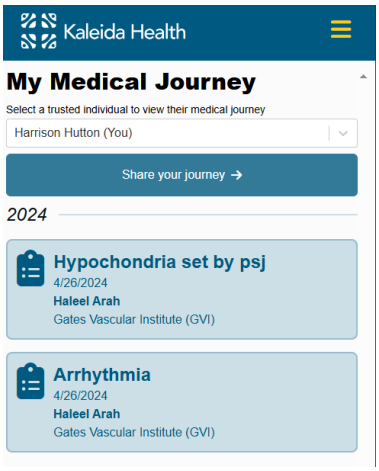


Fig. 2. My Medical Journey

From the 'My Medical Journey' page, patients can view their diagnoses and share information with a trusted individual or family member.

Doctor View:

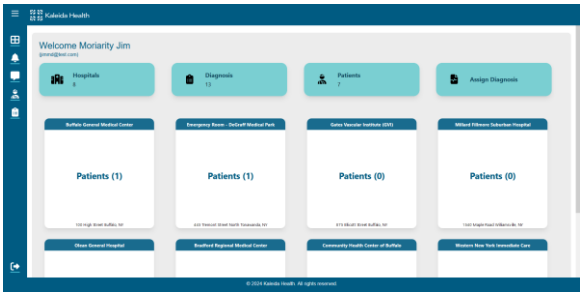


Fig. 3. Doctor Dashboard

The doctor's dashboard displays information about hospitals and patients. From there they can view patients by location or assign a diagnosis directly to one of their patients. There are links on the side for notifications, chat, patient list, and diagnosis list.

Admin View:

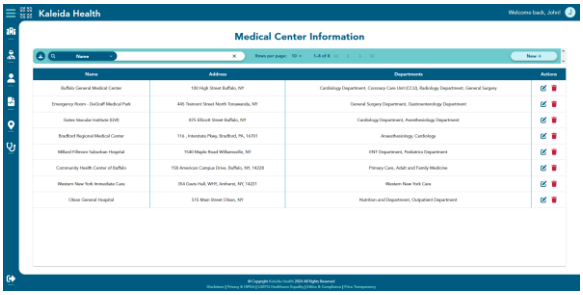


Fig. 4. Admin Dashboard

Admins are responsible for maintaining user accounts for both patients and doctors. They also manage medical centers, locations, doctor groups, and diagnoses.

Editing existing data:

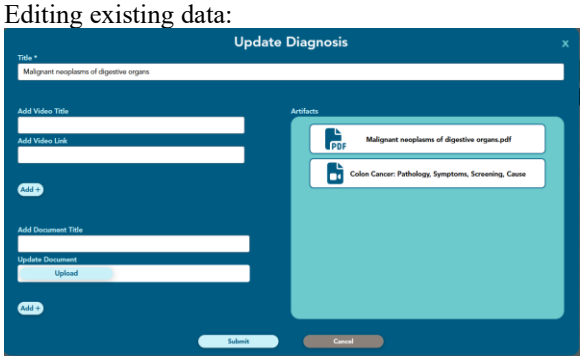


Fig. 5. Updating Diagnosis

Clicking on the pencil icon in the doctor or admin view will allow them to edit the diagnosis. They can also create new ones by clicking the 'new +' icon.
Doctor groups, locations, patient and doctor accounts can also be edited by the admin in a similar way.

Chat:

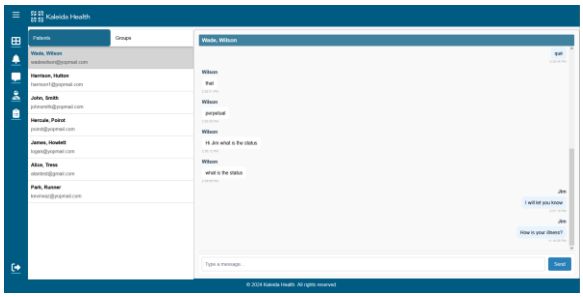


Fig. 6. Chat

Both patients and doctors are able to chat with one another. Chat messages are automatically translated to the patient's selected preferred language.

Notifications:

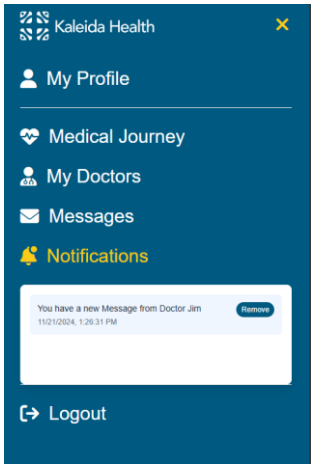


Fig. 7. Notifications

Patients will get notifications for new chat messages, if they are assigned a new doctor, or their location changes. Doctors will also get notifications for chat messages or if they are assigned a new patient.

Each interface in the Kaleida Health application has a consistent design that follows the organization's stylistic guidelines that were provided. The views are able to adapt to different screen sizes. Both patient and doctor views are optimized for mobile as well.

The interfaces were designed to be intuitive and easy to navigate for a smooth user experience.

VII. CONCLUSION

The Kaleida Health Application provides a system that allows doctors to provide their patients with the necessary information about their conditions and direct communication if patients need any further clarification. Patients are able to view the information doctors provide and chat with them directly. They can also share information with a trusted individual so that person can assist with care.

VIII. FUTURE WORK

Potential improvements could be having an option for a live translator to join chats in case there is some miscommunication with the auto translation. Eventually the application could be integrated with Kaleida's actual database and other systems.

There are currently issues with the deployment due to conflicting routing between react and spring boot. For future development I would recommend separating the react frontend and use spring boot as a rest API backend. To help with this transition I have added a variable for the backend server that precedes most of the API calls in the frontend code. I have also created a CORS configuration for the backend. Adjustments will also have to be made to the security configuration class as well.

ACKNOWLEDGEMENTS

This report is part of a group project that has been in development for over a year now across three separate teams. The project was developed collaboratively by Chaoping Lin, Franky Naidu, Hon Ching, Harrison Hutton, Gursimrat Tiwana, Mokshita Gupta, Jennifer Tsang, Naga Jaswanth Gandi, Pavan Kulkarni Bedikhanna, Phani Visweswara Sandeep Chodavarapu, Siddharth Cilamkoti, Venkat Amballa, and myself.

CONTRIBUTIONS

The following are my major contributions to the project:

- Added swagger for testing endpoints and automatic API documentation
- Added encryption for protecting patient information
- Added security for all endpoints using role based enforcement
- Aided with implementation of chat using spring web sockets and database structure for chat messages
- Numerous bug fixes, including one that prevented the application from running locally on computers with Windows operating systems
- Created a test plan for making sure all features are working properly before deploying
- Added CORS configuration and other changes to allow frontend and backend to be deployed separately
- Implemented a better solution for deployment and provided recommendations for future development

REFERENCES

[1] React Documentation. Meta Platforms, Inc., https://react.dev/
[2] Spring Boot Documentation. Pivotal Software, https://spring.io/projects/spring-boot
[3] Apache Maven Documentation. Apache Software Foundation, https://maven.apache.org/
[4] Amazon Web Services Documentation. Amazon, https://aws.amazon.com/