

Analytically Project Submission

HackSwiftly Submission

Created by Rayan Garg and Veer Doshi

April 18, 2020

The Problem

According to a study by the American Medical Association, primary care physicians, or people qualified to practice medicine, spend almost **6 out of their 11 hour workday** on data entry. Another study found that for every hour physicians were spending seeing patients, they were spending an additional two hours doing paperwork. The study also found that EHRs that after doctors go home, they then spend another 1-2 hours each night on EHR tasks. In short, the amount of documentation doctors have to do is outrageous.

All this paperwork leads to two primary issues: doctors burning out and less time with patients. According to the Harvard School of Public Health, **78% of doctors reported feeling burned out**, with research done by the American Medical Association identifying EHRs as the leading cause of this burnout. This burnout has a tangible impact, as it costs the medical industry **\$4.6 billion dollars** each year (NPR). There is also projected to be a shortfall of over 100,000 physicians by 2030. Reducing the time doctors have to spend doing paperwork means far less doctors would be needed, and at the same time, would prevent doctors from feeling burned out. Meanwhile, only 11% of patients feel that they get enough time with their doctors, an alarming number considering that a doctor's primary job is to help patients. These issues also have an impact on patient lives, as 400,000 people die each year from medical errors, something these problems largely are responsible for.

Additionally, most doctors struggle to actually use their EHRs(the electronic systems they use for paperwork), further hindering them on a daily basis. According to a recent Medical Economics report, 57% of doctors said they **would not** recommend their system to fellow physicians, showing that a clear need exists to make EHR's easy to use. Doctor Karla Dick elaborates in an article, "The interfaces are just so confusing and clunky. They invite error." A study published in the journal Health Affairs found that 36% of the medication errors at pediatric hospitals they studied were due to EHR usability issues. The other part of the issue is software glitches. An investigation by KHN and Fortune has found that alarming reports of patient deaths, serious injuries, and near misses—thousands of them—tied to software glitches, user errors, or other flaws have piled up.

Overall, according to a national survey by Medical Economics, more than 65% of doctors say EHRs have actually resulted in financial losses, and less than 11% of doctors reported savings. In the end, a system meant to save doctors time and simplify the process has instead done the exact opposite.

The Solution

Our app, **Analytically**, is a mobile EHR that solves the problem doctors face by reducing the time doctors have to spend with EHRs and providing an easy-to-use interface that contains no errors, making EHRs a tool that empowers doctors instead of hindering them.

Firstly, Analytically reduces errors and EHR difficulties by employing a clean interface without any system errors. Our modern interface ensures the doctors will be able to access the information they need quickly and without problems. Secondly, Analytically saves doctors time in two key ways. As of now, typing clinical notes, which doctors have to do each meeting with a patient, accounts for over 50% of the time doctors spend on their EHRs. Analytically reduces this time by allowing doctors to dictate their clinical notes instead of having to type it out. Studies have found speaking to be 3 times faster than typing, meaning that we save doctors more than one hour each day just with this feature. Analytically also saves doctors time by reducing the time doctors spend on creating a plan for patients. One of our most defining features, we use ML and AI to detect the disease patients have based on the data doctors put in. We take the data, send it to a server, run the python program on it, get it sent back to the server, then have it determine the likeliness of the patient having that specific data. Aside from saving doctors time, this feature also makes it possible for doctors to know immediately what issues the patient is having and how to treat it. These various features massively improve the EHR experience for doctors, and is something doctors have definitely stated they would be interested in.

Future Plans

In the future, we hope to implement a variety of other features that reduce the time doctors have to spend with EHRs. We have already written python code that would take the clinical note the doctor generates and then isolate the prescriptions the doctor wants to order, the scans the doctor wants to order, along with the next appointment the patient wants to set. We also have the code written to go ahead and automatically fill out the form in our app based on the data extracted. With this itself estimated to completely eliminate doctors 30% of the total time doctor's spend on EHRs, this would be transformational. No other EHR anywhere in the world has done this. The sole reason we were unable to implement this feature is because we did not have enough time to tie the python program into the server connecting it to the app. We would make this a key staple of our app in the future. Even without this feature, as presently constructed, our app is extremely unique. Not many mobile EHRs exist, and none of them automatically detect the disease the patient has, making our app completely original.

General Application Features

- The Analytically application provides a seamless user interface consisting of an elegant login page and several valuable features simplifying doctors' daily tasks.
- Login: The login page consists of several identifiers unique to every doctor, including hospital identification and biometric authentication through Face ID technology.
- Calendar: Three concise views are provided for the doctor after login authentication. The calendar view is designed to mimic the standard Apple calendar and also provides information about the patient when a meeting with the patient is tapped.
- Inbox: The second page, Inbox, allows the doctor to record his or her voice into the iPhone, which automatically transcribes the message through speech-to-text technology. Although actually sending emails hasn't been implemented due to the limited hackathon duration, the primary feature of this page is the speech-to-text technology implemented.
- Patient Interface: The third page, Patients, allows doctors to access the patient interface for each individual. Pressing on a patient (in the application, the interface is limited to the patient 'Jerry Saunders') results in the display of the patient home page, which consists of a brief description of the patient and his or her conditions. The patient interface consists of a seamless navigation menu to switch between tabs. The second view, Notes, allows the doctor to transcribe medical notes for the patient, utilizing the speech-to-text technology. Two more views exist within the interface: Prescriptions and Statistics.
- Statistics: The statistics page employs PUT and GET requests between the iOS application and the Heroku-driven REST API. After inputting the patient's cardiovascular values into the view, the API implements machine learning (Random Forest Classification) to calculate whether or not the patient has risk of heart disease. This boolean value is returned to the iOS application in the form of a GET request once the doctor refreshes the Statistics page.

Technologies used


- Heroku's cloud services were used to host the REST API carrying patients' heart information (database managed by PostgreSQL) and outputting the machine learning-driven data regarding patients' risk of heart disease.
 - The pandas and scikit-learn Python libraries were utilized to implement Random Forest Classification on the heart data to output risk information.
 - Apple libraries were implemented for biometric authentication (Touch ID, Face ID) and speech-to-text technology.
 - Several third-party libraries, including iOSDropDown and FoldingCell, were utilized to improve user experience within the iOS application.
-

Installation

Prerequisites

- Xcode
- Mac OS
- An iPhone 11 (or iPhone 11 on Simulator)

Usage

- Open Xcode, select 'Open another project', and open the file titled 'Speakably.xcworkspace'
 - Click on the blue file at the top left, titled 'Speakably', and go to Signing and Capabilities
 - Change the provisional profile to your developer account and edit the bundle identifier
 - If there are further issues, follow these instructions.
 - On the top left corner of Xcode, select 'Speakably' as the scheme and 'iPhone 11' because the app has been specifically designed for this device.
 - Press  + R and wait for Simulator (or the iPhone 11 device connected) to load.
 - For optimal user experience and application design, test the app on dark mode. Do this by accessing Settings > Developer > Dark Appearance.
 - For the first login page, select California as the hospital state, as hospital options have only been listed for California.
 - Enter any credentials for the three login pages. (For Simulator) Before pressing 'Continue' on the third login page, go to Features > Face ID and press 'Enrolled'. When prompted with the Face ID animation, go to Features > Face ID and press 'Matching Face'.
 - On the 'Patients' page of the application, select the patient 'Jerry Saunders,' as the remainder of the user experience has been designed for this particular patient.
 - On the 'Statistics' page of the patient UX, press 'Refresh' to view updated heart-disease risk after inputting cardiovascular data.
 -
-