Apollo - Product Requirements Document

About the Team

Team Name: Minimum Viable Team
Project Name: Apollo
Company: WELL Health
Lead: Ekta Shahani
Scribe: Terrell Marshall
Members:
Chris Lianides (chrislianides@gmail.com)
Michelle Nguyen (michellehnguyen@gmail.com)
Terrell Marshall (tbmarshall97@gmail.com)
Aditya Nadkarni (nadkarniaditya@gmail.com)
Ekta Shahani (shahani.ekta@gmail.com)

TL;DR

A telemedicine platform to make virtual doctor's visits effective and efficient by enhancing patient-doctor communication.

Background

Problem

Doctor visits are inconvenient. Patients must worry about the commute and then actual waiting time despite getting there early for your appointment - and that's if you even made an appointment. Health care is a fundamental right yet it is so difficult to access.

Commuting, wait times, costs, privacy and scheduling are significant barriers to accessing healthcare. Telemedicine seeks to address these problems; however, virtual appointments today diminish the quality of communication between patients and doctors.

Motivation

Expanding digital access to healthcare will increase patient satisfaction, promote patients to access healthcare, and help grow medical organizations.

- 84% of healthcare executives felt that the development of telemedicine services is important to their organizations.
- 74% of patients are comfortable with communicating with their doctors using technology instead of seeing them in person.
- 67% of patients say that using telemedicine would increase their satisfaction with medical care.

Existing solutions

- PatientAccess Mobile App

    - Connect patients to healthcare services (book GP appts (remote or in person), order prescriptions, explore pharmacy services, symptom information, view medical record (immunizations, test results, allergies, etc))
- Teladoc, DoctorOnDemand
    - Patients enter symptoms before video appointment and video conference with doctors affiliated with the platform.
- SutterHealth Patient Portal
    - Centralizes messages, health records, appointments, and billing for patients.
    - Live chat support
    - Example of technology behind large healthcare providers

Strengths of existing solutions

- Reporting symptoms is simple and can be done online by the patient.
- For larger providers, information like health records and past prescriptions is easily accessible in a central location.
- Virtual visits save patients time overall.

Gaps in existing solutions

- The actual virtual visit experience does not provide value beyond video conferencing.
    - 41% of millennials trust physicians as the best source of health information.
    - Lack of in-person communication due to virtual visits could negatively influence this already low number.
- Doctors cannot record patient vitals before visit
- Types of visits that can be conducted virtually are limited to common conditions that can be easily described and recognized (e.g. fever, allergies, flu) and addressed with a prescription.
- The post-visit experience is largely dependent on the information put forth by healthcare providers.
    - Less than 25% of millennials agree that doctors and pharmacists give them the information they need to make decisions

Note: Our target user group consists of people trying to save time through virtual visits. This group is comfortable with technology and trusts to engage with a telemedicine platform. Thus, millenials are a key target demographic.

Core Technical Advance

One of the project's ambitions is to extend the types of ailments that can be treated virtually, by using computer vision to guide patients through physical therapy. From the patient's view, they will be prompted with different illustrations of physical therapy poses to follow. The program will be automated to notify the patient when they do the exercises correctly. This will visually enhance the telemedicine experience for physical therapy treatments.

Language barriers diminish the quality of healthcare services to the patient. To address this, Apollo will provide patients with real-time multilingual speech-to-text transcription. We will use Google's Cloud Translation API to translate the doctor's English audio into text and then translating it to the patient's language. The text will be displayed as a caption for the patient to

read. The patient's audio data will be processed similarly except the API will transcribe their foreign audio data into the text of their native language and then translating that text into English for the doctor to read.

Another issue with virtual doctor visits is that doctors are unable to retrieve their patient's vitals. Without the vitals, doctors will not have a baseline to judge their patient's condition. To solve this problem, we plan to use Fitbit smartwatches to send live heart rate data directly from the patient to our application, for the doctor to analyze.

Goals

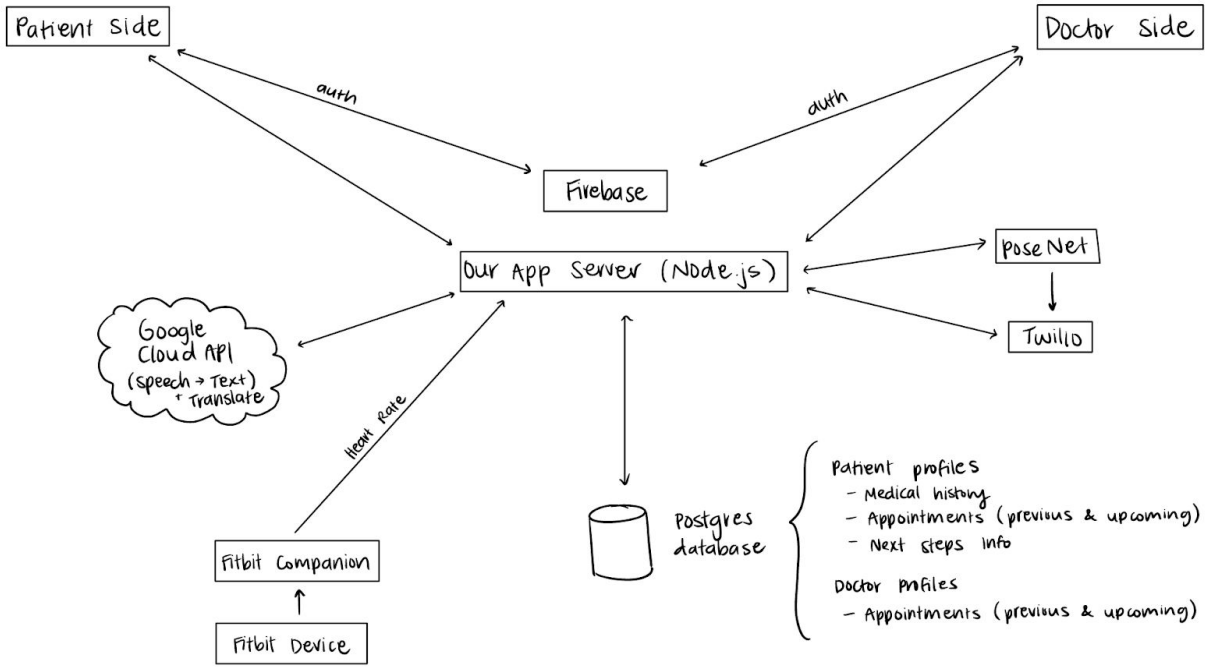
Existing telemedicine solutions utilize video conferencing to connect patients and doctors. Our goal is to augment the patient experience of virtual doctor's appointments and make virtual visits as effective as in-person visits by:

1. Improving virtual patient-doctor communication by allowing patients to communicate with any doctor in the language of their choice without a translator
2. Collecting patient data from wearables to better inform doctors about patient's current state during virtual visits.
3. Target a new use case for virtual visits by allowing doctors to demonstrate physical therapy exercises and have patients follow along with a virtual guide.

Assumptions

1. Doctors and patients using this platform are associated with a particular clinic.
2. Clinics using this platform already have access to detailed patient information and medical records.

Architecture Diagram



User Stories

End-To-End Pipeline [1] - Prioritization

User Story: As a user, I can send a message to the server through a client app, which is added to a database, and retrieve the message afterwards through the client.

Acceptance test: Given the client and server are running, the described behavior accurately updates the frontend and database and the server gets and sends the correct API calls.

Github commits:

Initial skeleton setup

Client and server communication

Client input form

Docker container set-up + Server->Database Communication

Logging in

User Log In [2] - Prioritization

User Story: As a user, I will be able to input my credentials to log into my existing Apollo account or register as a new user so that I can store and access health data.

Acceptance test:

On the sign-in page user can enter their gmail address and password and submit it for authentication. User password needs to follow the company's password policy. Upon successful login, user is redirected to the homepage. If user enters an incorrect username and or password they should see an appropriate error message indicating incorrect credentials. User must also see a link which allows them to reset their password. User must also see a link which allows them to register as a new user.

Github commits:

Firebase authentication

Scheduling an appointment

Patient scheduling - [6] - Prioritization

User Story: As a patient I can view all my doctor's available time slots so that I can schedule an appointment that works for both of us.

Acceptance Test: Given patient has logged in, when they navigate to the 'Schedule Appointment' page then they can see their doctor's available time slots, and schedule an appointment in that time slot. Then the slot is marked as unavailable in the doctor's calendar, and both the patient and doctor can see the appointment set up at that time.

Doctor scheduling - [6.5] - Prioritization

User Story: As a doctor/medical staff, I am able to view and edit a google calendar of my patient appointments, so that I can have full control over my work day calendar.

Acceptance Test: Assuming log-in was successful and user has opened their calendar, when a modification or new event is made on the calendar by the user, the change is updated in real-time to everyone who has access to view the event.

- An event card will contain the event's start time, end time, title, and description.
- A start time must be before an end time

- Title cannot be blank
- Description is optional
- Each event card will have a delete button

Medical Information

***Use Case**: Enter/Update Symptoms - [9] - Prioritization*
Actors: Patient, Health System, Patient Information Database
Precondition: Patient has logged into the system and entered their profile. They click on 'Enter Symptoms' during/after they make an appointment. This happens before the actual appointment.
Flow of Events:
Basic Path:

1. System retrieves patient account from the database
2. System asks the patient to choose one option from a general list of grouped symptoms.
3. Based on the selection, system asks a followup question with more specific symptoms based on the original question. Patient can select more than one option here.
4. After two rounds of questions, the patient clicks, 'Submit Symptoms'.
5. System updates the database with the new Patient information.

Alternative Paths:

1. In the case that the symptom is not listed, there will be an 'Other' option and the followup question will consist of a free-response section.
2. Patient logs out, clicks cancel, or leaves the page at any point before hitting 'Submit Symptoms'. System redirects to the Patient Portal.

Postcondition: Patient's symptoms data has been updated in the Patient Information Database
Github commits:
Symptoms stub page

Access patient records and symptoms
User Story: As a doctor, I am able to access my patient's profile through their appointment slot, so that I know what to expect before our appointment.
Acceptance Test: Given that the patient has submitted their symptoms, when the doctor opens up GCal to see their upcoming appointments, the doctor can view the patient's profile, which will contain their reported symptoms.
Github commits:
Database route to GET users

Live Appointments

Video call [3] - Prioritization
Story: As a patient or doctor, I can video conference via the web app for a virtual appointment.
Acceptance Test: Given an appointment is in session, when both the patient and doctor have joined the video call, then they can see and hear each other live.
Github commits:
Video calling integrated into project

Real-time content sharing [8]
Story: As a patient, I can share and annotate pictures, links, and documents with my doctor in real-time during a video call so that I can share specific concerns with my doctor.
Acceptance Test: Given a picture, document, or link on a patients' laptop, when selected by the web app, then the patient and doctor can see the content and draw on the item.

Real-time system information sharing [8.5]
Story: As a doctor, I can view or share and annotate a patients' records or notes from previous appointments with my patient in real-time so that I can clearly explain my observations and diagnosis as needed.
Acceptance Test: Given a patient information already contained in the system, when selected by the web app, then the patient and doctor can see the record and draw on it.

- Note: The system will not connect to a real EMR system.

Use case: *Live translation - [4] - Prioritization*
Actors: Patient, Doctor, system
Precondition: Patient and doctor are both in a video call. Doctor is speaking in English. Patient is speaking in a non-English language that is supported by the system.
Flow of Events:
Basic Path:

1. Doctor questions patient in English.
2. Patient sees captions in their language of Doctor's speech in realtime.
3. Patient answers questions and describes problems in their language.
4. Doctor sees captions in English of Patient's speech in realtime.

Alternative Paths:

1. If captions do not appear or are significantly delayed, the patient or doctor can indicate the problem through a chat box.
2. The patient / doctor can communicate through the chat box, which will translate their text appropriately, or reiterate until captioning resumes.

Postcondition: Any post-visit information generated will be available in both languages for both the doctor and patient.
Github commits:
Example of translating speech to text output
Example of using Twilio DataTrack API

AR Physical Therapy (Patient Side) - [5] - Prioritization
User Story: As a patient, I am able to show my body on the screen so that the system can recognize my actions and I can follow along.
Acceptance Test: Given the patient's body is visible through the webcam, the patient can listen to the doctor's instructions and move their body according to the virtual lines displayed by their screen. The virtual line with first outline the patient's body in a greyed out line. The correct position of the therapy movement will be shown in yellow and will turn green once the patient has formed the correct position.

Share Live Heart-rate *[7] - Prioritization*

Story: As a patient, I can open the Apollo Fitbit app on my watch and view my live heart rate on my Fitbit.
Acceptance Test: Given the app is running, when the patient's heart rate changes, then the heart rate displayed on the watch changes.

Story: As a patient, I can open the Fitbit app on my phone to send my heart rate to the Apollo web app.
Acceptance Test: Given the Fitbit app is running, when the patient is in a live appointment, then both the doctor and patient can view the patient's live heart rate in the Apollo app.

Get next steps

Access information from completed appointment
User Story: As a patient, I can review my appointment details which contains an automatically generated appointment summary, action items and documents referenced during the appointment via email/webapp, so that I remember exactly what happened during an appointment and I know what to do next to best treat my symptoms.
Acceptance Test: Given the patient has completed a virtual appointment with their doctor and the patient is in the webapp, when the patient interacts with that appointment, then Apollo displays to the patient a pop-up that shows (in this order) an appointment summary, next steps, and documents shared during the appointment (this flow can also be completed similarly in an email that would link the user to the webapp).

Get a second opinion
User Story: As a patient, I can request a "second opinion" while viewing an appointment summary, so that I can feel confident that I am receiving the best, peer-reviewed treatment possible
Acceptance Test: Given the patient has completed a virtual appointment with their doctor and the patient is in the webapp, when the patient is viewing their appointment summary and requests a second opinion, then Apollo suggests doctors that specialize in their symptoms for the patient to choose from and proceed to schedule another virtual appointment with that doctor.

Follow referral
User Story: As a patient, I can be notified via email/webapp notification when I have been referred by my doctor and be guided to an appointment set-up flow, so that my referral process is as quick and as seamless as possible
Acceptance Test: Given the patient is activated on Apollo, when the patient is referred by their doctor, then Apollo sends a push notification and/or email to the user which will notify them that they have been referred to [Doctor name] and offer an option to the user from the notification to quickly launch an appointment set-up flow. This flow can take one of two forms:

- **Doctor is activated on Apollo -** this will simply trigger the default make appointment flow and notify the doctor within the webapp when the appointment has been created
- **Doctor is not activated on Apollo -** from the patient's side, this looks the same as the previous option, but after finishing the appointment setup flow, this will create a

templated email with the patient's preferred times, symptoms, record, etc, and allow the patient to review the email in-app before sending it out

Recommend next steps / follow up appointment

Automatically generated "next steps"/appointment summary - [10] - Prioritization
User Story: As a doctor/nurse, I can optionally review/modify Apollo's automatically generated "next steps," so that I can quickly ensure my patient is receiving the best treatment possible.
Acceptance Test: Given the doctor is engaged in a virtual appointment with their patient, when the appointment is about to end/has ended, then Apollo displays automatically generated "next steps" on the screen, and the doctor has 4 choices to deal with the pop-up: approve, edit, assign, or dismiss

- **Approval** will send the next steps to the patient through the webapp and potentially via email
- **Edit** will allow the doctor to manually fix any errors in Apollo's auto generated "next steps" and/or attach any relevant documents
- **Assign** will pop-up a list of the doctor's assigned nurses to quickly assign-to so they can review the "next-steps" and the doctor can move on with their day
- **Dismiss** the pop-up which will not send patient "next steps" at all

Refer the patient to another doctor/specialist
User Story: As a doctor/nurse, I can refer my patient to another doctor directly within Apollo, so that I can quickly refer the patient without having to waste time finding that doctor's contact information through external services and I don't have to waste time reaching out to that doctor myself
Acceptance Test: Given the doctor has had one appointment scheduled with a given patient (could be an appointment in the past or upcoming), When the doctor visits that patients profile and clicks "Refer patient," Then Apollo automatically suggests doctors to refer to the patient depending on the symptoms the patient entered into the appointment details or the doctor can manually search for the doctor And after clicking on a doctor, there are 2 possibilities:

- **Doctor is activated on Apollo -** this will simply notify the "clicked-on" doctor's practice that this given patient has been referred to them from within the webapp. There will be an additional experience on the patient's side as well
- **Doctor is not activated on Apollo -** this will automatically generate a template email based on the symptoms the patient entered on their most recent appointment creation. This experience will be continued on the patient's side

Appendix

Technologies:

- Frontend: React, react-bootstrap
- Backend: Node.js, Koa, AWS S3
- APIs: Twilio, GCP Speech-To-Text, Google Calendar, PoseNet
- Database: Postgres

- UI Design: Figma

Resources
https://www.forbes.com/sites/forbestechcouncil/2019/04/09/five-ways-millennials-do-health-care-their-own-way/#66e4003220c5
https://wellapp.com/blog/millennials-approach-to-health/