

# HEBO

---

**NILA BANERJEE, SYDNEY CHOI, DANIELLE HU,  
STEPHANIE TRUONG, KATIE WILLIAMS**

HCI CAPSTONE FINAL REPORT, SPRING 2018

# CONTENTS

---

3	Introduction
3	Executive Summary
4	Research and Prototyping
7	Hebo
12	Next Steps
17	Tools
17	Overview
18	Logistics
19	DialogFlow
28	Mobile App
34	Appendix

# INTRODUCTION

## Problem Space

After undergoing Mohs surgery, patients often have questions and concerns about their post-operative treatments. Most of the post-operative instructions are given right after the surgery and most of the time they are rushing with adrenaline and at times slightly disoriented. Therefore, they cannot retain all the information that the nurses tell them at the time of the procedure. Therefore, patients require support from clinical staff over the phone or in the clinic to address common concerns. Moreover, staff members often have to dedicate significant portions of time to communication information already made available to patients.

Our team has been working to find a solution that will better assist patients during this post-operative period. Our research was first geared towards understanding the perspectives of both the hospital staff and skin surgery patients, while uncovering what the post-operative care process entails.

## Hunt Statement

In order to successfully address the post-operative concerns of skin cancer patients, we will work to understand the patient mindset, identify the challenges that patients face, and explore media that are optimal for communicating relevant post-operative solutions. This will help us create a platform that will convey medical instructions in a personalized and comprehensible manner, thereby reducing the volume of low priority calls taken by the hospital staff and streamlining emergency concerns.

# EXECUTIVE SUMMARY

Patients recovering from skin cancer surgery often do not completely understand and have numerous concerns about their post-operative care, which results in clinicians receiving a high volume of redundant and low-priority phone calls. Our team was tasked with developing a post-operative care assistant that could handle these concerns and act as a constant companion throughout a patient's recovery. We conducted numerous stages of research and prototyping to finally create Hebo, an app-based chatbot that uses voice, text, and images to convey care information. Here, we overview how we arrived at our solution, as well as how it serves as a proof of concept to convert confusing and overwhelming medical information into easily comprehensible pieces of knowledge.

# RESEARCH AND PROTOTYPING

In scoping our problem, we found that there were two major groups of stakeholders: hospital staffs. To find a solution that would balance their needs, we need to understand the perspectives on post-operative care for both of these groups. Therefore, we chose research methods that would best convey not only the breakdowns in the care process, but also the mindsets of both patients and nurses.

We started with a comprehensive literature review to not only understand the current post-operative care process, but also to gain knowledge on pain management and the relationship elderly people have with current technologies. Information from our literature review helped supplement our first-hand findings and filled in gaps on aspects we could not directly test. Next, we conducted a heuristic analysis, evaluating a prototypical app Dr. Carroll previously developed to guide patients through their post-operative care using Nielsen's principles. To gain a more thorough understanding of what patients and clinical staff undergo, we performed numerous contextual inquiries at the UPMC Presbyterian and St. Margaret clinics, observing surgery, lab work, patient guidance, and more. Our team shadowed patients, nurses, and doctors as they went about their tasks to identify current processes and breakdowns. As a part of our diary study, we provided each surgical patient within our study's timeframe with a notebook to record their post-operative journey, which they were asked to fill out from the evening of their surgery until their one week follow-up appointment. We intended to discern what problems patients commonly face after surgery and what actions they take to address any concerns they have. Finally, we conducted interviews with patients, caretakers, hospital staff, and doctors to recognize the perspectives of each stakeholder and gather information from all relevant parties. For more details on our research process, please refer to our research report.

From our initial research, these were our insights:

1. Doctors **think** that they are providing patients with all of the information they need, but this is not always the case.
2. Patients look for **greater reassurance** and **personalized care** from the post-operative process.
3. Nurses feel that some post-operative calls are **redundant**.
4. The predominant demographic of skin cancer patients, the elderly population, calls for an emphasis on **accessibility** and **simplicity**.

From our research insights, we had multiple brainstorming sessions to come up with a list of solutions. We narrowed down our ideas to 2 possible solutions which was the chatbot and the interactive tutorial on post-operative care where we parallel prototyped. From there, we took the best features from each prototype to create Hebo, a chatbot with limited conversation topics designed to answer questions specific to the patient's surgery experience. We had multiple user testing sessions and here are the consolidated list of our insights:

*Voice is good, but can't have just voice only. Visual elements aid patient understanding of the instructions.* From our user testing with our prototype chatbot, it was revealed that users became confused when trying to remember and understand lengthy directions given to them by a chatbot. Therefore, it is important to keep the instructions short and easy to understand. Whenever it is appropriate, having visual answers will help patients understand the instructions.

*Patients asked questions that we didn't anticipate and they want quick answers to their questions when they arise.* It is unpredictable when patients will want to know certain information about their post-operative care. They do not feel like information is relevant until it becomes a pressing issue or problem for them. Therefore, information needs to be accessible whenever the patient needs it. If Hebo cannot answer the questions, it must provide a way for patients to find their answers.

*Patients trust tools that comes from their personal doctor's office.* Patients are more likely to trust a tool that is more closely linked to their doctor or comes directly out of their office. Our users requested confirmation and reassurance that the answers coming from the chatbot were coming from their own doctor. Therefore, the introduction and delivery of Hebo must come from the doctor's office to ensure trust in Hebo.

*Hebo's knowledge base should extend beyond the post-operative sheet.* Through our testing sessions, users find Hebo more valuable if its knowledge base cover common questions that extend beyond the post-operative instructions. Another approach to extract value from Hebo that differentiates from the post-operative sheet is to make Hebo more dynamic and interactive than the static state of the sheet. With a dynamic interaction, Hebo can help the user parse through uncertainties that might not be addressed from their initial read of the post-operative care instructions.

*Answers need to be more targeted and personalized to improve accuracy and engagement.* Inaccurate responses can be addressed with more thorough programming of Hebo. Currently we use Dialogflow to build out Hebo; so in this case, we need to be more extensive and detailed with our training phrases for each intent. The more we train our responses, the less Hebo will get confused about matching the user's question to the appropriate response. Both tangential and generic responses can be

addressed if we add more specificity to our chatbot. Solving for specificity should also solve issues with inaccuracy: the more we know what the user is asking, the better we can answer his or her question.

*Personalizing care instructions for the user creates engagement.* For future iterations of Hebo, it is important to focus on including more recommendations or insights that are personalized to the user. Knowing what types of information to store may not be trivial. Storing basic information about the surgery is a good starting point, but information revolving medical histories (i.e. allergies, medication, etc.) may not be feasible to program into Hebo since they are more serious recommendations. Looking into the complications of such recommendations and scoping these down to a useful subset will be a large part of this design process. In addition, it might be necessary to design the interaction of inputting user data into the application. This will likely either be the user putting this information in themselves, or a healthcare provider inputting this information for the patient during the onboarding session. Understanding the willingness of both stakeholders in this input phase is crucial in deciding what would be the best solution for this interaction.

*Development should focus on usability improvements to minimize awkwardness and confusion.* For the voice aspect of Hebo, we should make the listening and talking interaction between the user and the agent as smooth as possible. This means making Hebo more patient so that the user is not cut off early, making Hebo better convey to the user its state (i.e. listening), and maybe even helping the user feel more natural when they ask questions. One way to solve some of these issues would be to have a walkie-talkie interaction: instead of the user pressing the microphone icon once and having Hebo decide when the user stopped talking, the user can press and hold the microphone icon to indicate that he or she is speaking. In general, we should prioritize testing, designing, and developing for usability, especially with our target demographic.

***Please refer to our previous reports for more detailed information regarding our research insights, low-fidelity prototype, and mid-fidelity prototype development.***

# HEBO

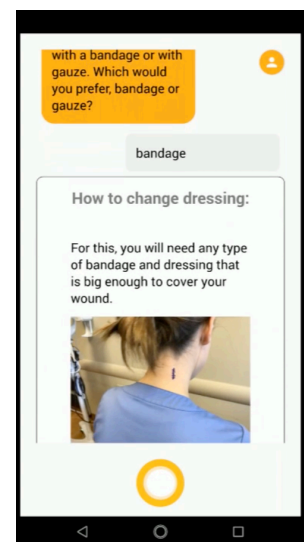
Our chatbot is designed to be a personal healthcare companion and assistant that answers any questions the patient may have about bleeding and wound care. We found in our research that patients need reassurance to questions that they may already know the answers to. Hebo helps to reassure patients by being available to answer any questions the patient may have at all times and thereby simultaneously reduces the call load of the nurses. In order to accommodate our limited time in the project, we narrowed the scope to questions that concern bleeding, wound care, and swelling on surgery sites on the neck and scalp. Based on conversations with nurses and looking at a few nurse call logs we estimate that although small this scope will address around 25% of the post-operative calls from patients within 48 hours of their surgery.

## *Improvements for High-Fidelity Prototype*

---

### Visual Answers

As outlined in our research insights, visuals were found to be extremely helpful in communicating certain ideas to patients. However, we wanted to maintain a balance of including simple and quick verbal answers and introduce more detailed visual answers when it was helpful. We also realized that having real photos was more effective than having drawings since patients are able to connect more with the visuals. As per the recommendation of our client, we determined that within our scope the questions that required a visual answer was how to change your dressing, how much vaseline to use, and how to apply pressure on a wound. In order to construct these answers, we went to the clinic and had the nurses perform a think-aloud of how they would instruct a patient normally and we documented the process with pictures along the way. We found it was helpful to record audio of the nurses' voices and transcribe them later into steps that were easy to follow.



### Specific and Customized Answers

Effective medical recommendations require at least basic knowledge of the patient's history. We knew that the answers Hebo gave must be personalized to the patient in

order to provide the best care. To emulate this, Hebo takes into consideration the date, time, and surgery site of the patient's procedure when responding to questions. For example, when the patient asks, "When should I change my dressing?" Hebo will calculate the appropriate time based on when the surgery was initially performed.

## Expand Q/A from the Post-Op Sheet

To make a robust solution we knew that Hebo's knowledge base must extend off of the post-operative care sheet. To generate a pool of possible questions and answers that were in our scope we first compiled a list of possible questions based off of the post-op sheet for dissolvable sutures as a starting point. Then, we consulted the nurses and doctors in addition to what we found in our diary studies to identify common questions to expand Hebo. This was an iterative process, and we had our client and nurses check on our progress and advise if they thought any additional questions should be included. To summarize our question and answer database, we made a logic tree using post it notes to identify the main branches and streamline the questions that pointed to the same question (Appendix #). Finally, we imported the questions and answers to DialogFlow which is outlined in further detail in a later section of this document. In addition to expanding Hebo's knowledge base beyond the sheet, we also incorporated a useful feature that helps the patient put the appropriate amount of pressure on their wound by incorporating a built in timer. Features such as these help to make Hebo a solution that bridges the gap between the nurses and doctors and provide more comprehensive care.

## Verify Answers with the Clinic to Ensure Hebo's Recommendation Accuracy

Of course, every recommendation Hebo makes must be thoroughly reviewed by a doctor to ensure accuracy. We gave access to our digital logic tree to our client for review in addition to meeting in person to talk through the recovery process and ensure Hebo's answers were appropriately worded and accurate. Furthermore, we verified with nurses and doctors that our pool of questions were low priority and easy to answer. While our goal is to allow Hebo to help reduce some of the nurses' call log, we did not want Hebo to address high risk questions that require a doctor's attention.

## Make Interactions more Dynamic by Introducing Follow-up Qs

One problem we encountered in earlier stages of development was that Hebo would not correctly answer the patient's more specific questions. We wanted to fix this to not only provide the patient with the answers they need but also make the conversation more fluid and dynamic. In order to increase the specificity of answers, we added more



content into DialogFlow and also trained Hebo intensively using DialogFlow's internal training tool. Additionally, we introduced follow up questions to make sure Hebo got at the core of what the patient was asking and ensure that Hebo was providing the right recommendations. For example, after a patient asks something like "Should I see blood on my bandage?" Hebo will first respond by asking "Can you confirm whether or not you are bleeding right now?" before answering the patient's original question.

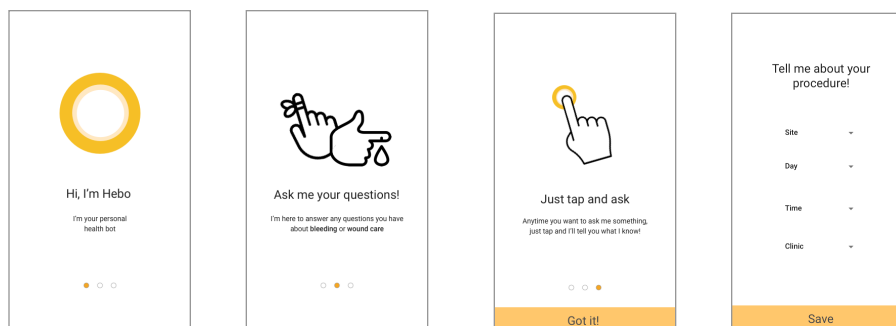
## Design

---

The design of Hebo is very simple. To accommodate for the lack of technological expertise in the elderly population, we focused on making the UI as simple as possible. As such, all the navigation is what android design guidelines refer to as in-app navigation in which there are no components such as navigation drawers or tabs. Like the calculator app, there are very few buttons and virtually no need to navigate through different components. There are three main components of Hebo's design: onboarding, the conversation page, and the patient information page.

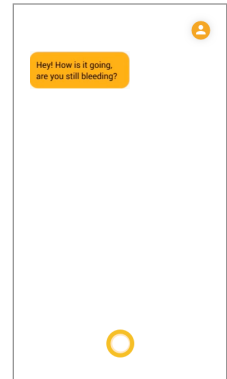
### Onboarding

For Hebo's onboarding, we knew it was extremely important to establish trust and also educate the user on how to use Hebo since most users are unfamiliar with using this type of technology. We incorporated three simple screens that introduced the user to Hebo, explained the scope of Hebo's knowledge bank, and instructed the user how to interact with Hebo respectively. Additionally, we automatically prompted the user to populate their procedure site, date, time, and clinic on the patient information page and also agree to a consent form. Due to the technological difficulties of creating Hebo's knowledge base and building out a scalability plan for Hebo's future, the team did not fully flesh out Hebo's design. We recommend future teams spend time iterating on the overall design, especially the onboarding process to make sure that it is effective in teaching users how to use the app and what questions to ask.



## Conversation Page

The conversation page serves as the hub of the app and where Hebo lives. Users who have already gone through the onboarding stage will be directed to this page as their home screen and can immediately start speaking to Hebo. Users simply tap on Hebo's icon to begin talking. We recommend that future iterations of the design include an animation such that the user knows when Hebo is listening and when Hebo is processing and preparing an answer. In addition to the back and forth conversation bubbles that appear, visual answers are also embedded within the conversation and the user simply needs to scroll up to view the content.



## Patient Information Page

The patient information page is introduced to the user as part of the onboarding and then appears as a button on the top right corner of the conversation page. Its purpose is to give the user the ability to update their patient information whenever necessary so Hebo can update their answers accordingly. This is especially important for patients who undergo multiple procedures.

*We made Hebo an android app since our client has an Android device. Ideally, Hebo could be optimized for both iOS and Android platforms to reach a broader audience in the future.*

A vertical rectangular mockup of a mobile app screen. At the top, the text "Tell me about your procedure!" is centered. Below this are four rows, each with a label and a dropdown arrow: "Site", "Day", "Time", and "Clinic". At the bottom, there is a solid orange bar with the word "Save" in white text.

## User Testing and Results

---

We performed more focused users testing with our high fidelity prototype. Our target audience included patients from the clinic, in addition to the nurses and other potential users who were above the age of 60. Participants were invited to spend some time interacting with Hebo while performing a think-aloud and then follow-up in a brief interview. The main insights we gained from testing are as follows:

*Participants sometimes forget how to interact with the app.* We found that although the interaction design for Hebo is very simple, users still had difficulty remembering to push the button before they started to speak. Especially for users who did not have smartphones or were not accustomed to using such technology, we found that simple gestures such as scrolling up to see the remainder of the visual answers were not

intuitive to our user population. We recommend incorporating popup messages and additional scaffolding to help remind the user how to interact with the app.

*Participants asked questions we didn't anticipate.* As we had anticipated, sometimes the user would ask questions that Hebo was completely unprepared for. These questions were either highly specific such as asking Hebo is a certain brand of soap was acceptable to shower with or questions that did not pertain to bleeding, wound care, or swelling. In order to combat this, Hebo has a default message when it doesn't understand. Hebo will ask the user to rephrase the question or simply say that he does not know how to answer that question.

*Accessibility should remain a primary concern* As the target population is mostly above the age of 60, we found it was important to continuing iterating on the design and overall functionality to compliment accessibility features. Examples include font size, the onboarding process, and speed of speech. No matter the future iterations of Hebo that should occur, accessibility should remain a high priority.

*Participants see value in Hebo.* Finally, we found that overwhelmingly our users found significant value in Hebo. Actual patients thought Hebo was helpful and easy to use and the nurses and doctors thought Hebo was accurate and could help reduce the volume of low priority calls. With this in mind, we can justify further development of Hebo past the lifecycle of this team's work and hopefully impact patients' recovery process in multiple clinics.

# NEXT STEPS

Our team has worked to create a proof of concept project that provides evidence that a personal health bot, Hebo, will help decrease the amount of patient calls. We believe that in order to successfully carry out the remainder of this project, our client will need to pursue work with future student consultant teams to focus on further refining Hebo, to expand the scope of the Hebo functionalities, and to create a feasible scalability plan that allows the introduction of Hebo to multiple different clinicians serving Mohs surgery patients. Figure A depicts a one-year projection of the project given this timeline.

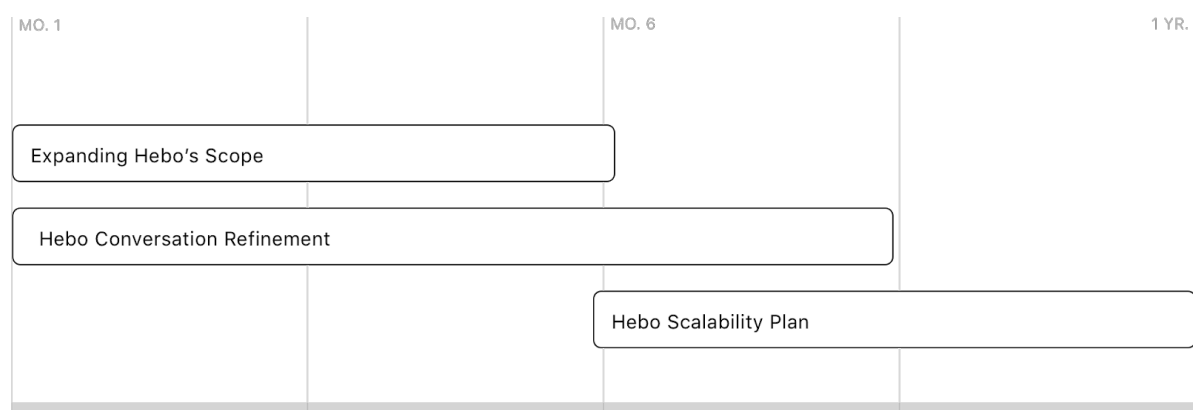


Figure A

## Expanding Hebo's Scope

Currently, we have limited Hebo's scope to Bleeding, Swelling, and Wound Care. This has allowed us to focus our proof of concept on the most common problems that patients face following their Mohs surgery. In doing so we have been able to limit the amount of potential questions Hebo answers and refine the interactions and logic that Hebo uses to answer these questions. In order to build out a successful and comprehensive chatbot interface for patients, Hebo's scope must be expanded to all aspects of patient care.

In our research, we discovered a set list of topics that cover the majority of the questions asked by patients at the UPMC Presbyterian and UPMC St. Margaret Dermatology offices. The following is a list of question categories that cover most of the patient concerns identified.

## Question Categories

- Bleeding
- Swelling
- Wound Care
- Infection
- Pain
- Bruising
- Redness
- Medication
- Diet
- Activity
- Spitting Stitches
- Emergency

Expanding Hebo's scope will mean covering all of the questions that we found in these question categories. Many of the documents from our research, including our call logs, diary studies, and interview notes, already have many example questions and concerns from patients. Future teams can use these resources to continue this research and discover more questions that fall into these Question Categories.

As our research progressed, we discovered that some patient questions were not fully addressed by our set of Question Categories. Particular issues such as dental concerns or travel concerns fell into an *Other* category that we were not able to fully cover in our research and prototyping. This forced us to recognize that Hebo may be asked quite a few questions that don't fit into our standard question model. We recommend that future teams continuing work on this project take advantage of the following tools to expand Hebo's scope and discover all concerns that patients may have:

## Patient Call Logs

Many of the patient call logs held by the UPMC Dermatology office are confidential; however, we were able to create a collection of patient call topics by querying staff about the general questions, topics, and concerns that patients had called in about over the past few weeks. This allowed us to build up a more realistic list of calls that patients have been making to the office.

## Contextual Inquiries with Nurses

We used contextual inquiries to observe nurses going about their routine. These inquiries gave us the opportunity to prompt nurses to recall their most recent patient calls and what questions were asked. We were also able to observe how nurses dealt with the patient calls they received and the process they followed to answer patient questions. A particularly useful resource we found during this study was the triage nurse. Many clinics have a *triage nurse* who focuses on answering patient calls throughout the day. We found that he or she is usually able to recall more details about the post-operative calls made to the office due to the fact that the triage nurse's

primary role revolves around answering patient questions and resolving any post-operative complications.

## Diary Studies

As described in our preliminary research report, our diary study was a significant way that we collected patient concerns and questions following the day of surgery. We believe that this method helped us discover questions that our client previously was not aware that patients had. This method also engaged patients and we found that our subjects were excited about participating in a study where they felt that their opinions were being heard.

Diary Studies can be used as a way to build out the patient question database. To help future members of this project use diary studies to gain this insight in an efficient manner, we have supplied a template for the diary study notebooks (Appendix A). During our research phase, these diary studies took roughly two weeks to complete per subject. Our timeline for the research worked as follows:

- Before the Study: Create the diary using small 5' x 3' notebooks and the templates included with this report.
- Day of Patient Surgery: Distribute the diary to the patient along with a brief walkthrough of how to use this diary during their post-operative recovery period. Remind the patient to bring the diary study back to the office during their follow-up visit.
- Patient Follow-up Visit: Retrieve the diary study with the patient. Look through the diary. If time allows, clarify any points of extreme pain, concern, or confusion with the patient.
- Upon Receiving the Diary: write any clarification points concerning the patient's diary study, take note of any comments made during the follow-up visit, take note of the patient's general demographic.

In our experience, the rate of diary returns depended mainly on the onboarding process between doctor or nurse and patient. Handing out the diary to patients who appear interested in the study and understanding of the process yielded a higher return rate of the diaries.

## ***Hebo Conversation Refinement***

---

Our user testing found Hebo to be comprehensive, but somewhat awkward in conversation. Here are some suggestions that we believe can be used to enhance user interactions with Hebo and make it a more personal companion.

### **User Testing All Possible Conversations**

As teams are able to expand Hebo's scope, they can use Dialogflow to input new intents into the system. Each new addition and any revisions to the scope should be reviewed with the client. Following review and addition of new topics that Hebo can address, we recommend further user testing focused on these new additions. This will allow the team to understand how any updated version of Hebo reacts to the various ways that a user can ask a question and maintain a dialog. Our client was able to give us access to many potential users in order to carry out this user testing. We also recommend pulling from other potential user bases around the same age range. While it is best to test with potential users of the product, many of the flaws in the system can be highlighted with general users. We made an effort to provide breadth and depth in our user testing in order to increase the possible outcomes and conversations that we could observe between users and Hebo.

### **Quick User Testing Sprints**

In our experience, performing rounds of iterative user testing and development was also a successful step in refining the conversation experience with Hebo. Bugs could be noticed fairly quickly with two or three 15 minute user tests. By performing short sprints of user testing and then fixing any bugs identified before the next round of user tests, we were able to have a more efficient user testing process.

### **Create 3-5 Training Phases per Intent**

With each intent added to dialog, a new conversation tree must be created. This conversation tree must account for all possible interactions the user has with Hebo. Our team noticed that the more we increased the scope of Hebo, the more overlap there could be in the training phases used for each intent. As teams work to increase the scope of Hebo, we also recommend to limit the amount of training phases possible for each intent. In our research we found that three to five training phrases was ideal for each intent. A more effective way of ensuring Hebo's comprehension of the user was to focus on entity creation and the use of synonyms in Dialogflow.

## Emphasizing the Personal Aspect of Hebo

A final recommendation we have in refining Hebo's conversation skills is to increase the companionship, sympathy, and emotional support that Hebo provides to users. Our initial research showed that many users make postoperative calls because they are looking for greater support and reassurance in their recovery period. A more personalized Hebo can be created by focusing on two different aspects of the conversation: positive phrases and personalized check-ins. The use of positive phrases and words of encouragement is practiced by many nurses and doctors in their phone calls already. Providing personal statements of reassurance like "you can do it" and "I'm sorry you're uncomfortable" emphasize Hebo's focus on personal comfort. These statements should be incorporated with any responses programmed into Hebo's Dialogflow interface. As the scope of the chatbot grows, this same personality should be incorporated into all possible Hebo responses. Ultimately, Hebo can truly become a personal health bot by knowing and anticipating the needs of patients. If Hebo can remember to follow up on concerns patients have had or monitor the kinds of questions the patient has, we can understand what interactions the patient is looking for during their recovery period. Hypothetically, if a team were to focus development on this personalization, Hebo would be able to predict complications before they happened and address questions before they were asked by closely monitoring the progress of the patients post-operative period. Each interaction with Hebo would be logged and would trigger a reminder for Hebo to follow up with the user in a few hours or the next day.

Our diary studies also proved to be effective in creating an outlet for patients to voice their current opinions and emotions. While this action is not only cathartic for the patient, it also provides meaningful data to doctors and nurses regarding the status of their patient. Our research proves that the incorporation of a journaling component to Hebo's conversational skills would be beneficial to both parties.

Ultimately, Hebo's conversational skills should mirror that of a doctor or nurse. Through the use of contextual inquiries, a team would be able to understand these interactions in order to better mirror them in Hebo.

## ***Hebo Conversation Refinement***

---

Ultimately, Hebo's success relies on its scalability. In order for this to be a tool that can be adopted by any doctor practicing Mohs surgery, Hebo needs to be scalable and customizable to a certain extent. We have foreshadowed an abstract scalability plan that could potentially be incorporated into a scalable prototype of the Hebo platform.






Hebo's conversation logic can be broken down into categories and fill-in-the-blank statements. The final, perfect version of Hebo will know all the possible questions a patient may ask. The answers to these questions may not necessarily be standard across all dermatology practices. For each question, there must be an answer; however, the variability of that answer usually only ranges in terms of differences in medication recommended, time periods, etc. We believe that each answer that Hebo supplies can be broken down into fill-in-the-blank statements. Future teams have the power to create an onboarding survey that allows doctors and nurses to answer questions in a survey about the specific recommendations that their practice gives to patients. With each question answered, Hebo learns more about the preferences of that practice and is able to answer a greater percentage of patient questions. After amassing a significant amount of data from different practices, Hebo would also be able to provide suggestions on what doctors should recommend based on the most common answers to these survey questions.

We believe that this scalability plan and the data amassed by Hebo as it is used by more and more practices is the most valuable component of Hebo. Once Hebo becomes scalable, it can collect data that can help physicians give better recommendations and better understand their patients' needs.

## TOOLS

### Overview

---

		
Mobile application (Android)	Dialogflow	Webhook (Google Cloud Platform)
<ol style="list-style-type: none"> <li>1. Sends requests to Dialogflow with correct parameters</li> <li>2. Handles all mobile application interactions and features</li> <li>3. Keeps track of user information</li> </ol>	<ol style="list-style-type: none"> <li>1. Receives requests, chooses appropriate Intent, and returns an appropriate response back (handles natural language processing of request to choose the correct response)</li> <li>2. Contacts webhook if intent uses Fulfillment to respond to request</li> </ol>	<ol style="list-style-type: none"> <li>1. Receives POST requests from Dialogflow, and returns the appropriate response</li> <li>2. Uses parameters to return a more personable / customized answer</li> </ol>

### Overview of how it works

The intricacies of understanding natural language is not an easy task. In order for a program to determine the difference between the phrases "how do I change my bandage" and "when can I change my bandage," a computer needs to understand the various nuances to how users phrase their questions and requests. Dialogflow is an online tool that uses machine learning to help with natural language processing. As a user of Dialogflow, we can create an agent (Hebo) and map out all of the conversations a user can have with the agent. In our case, conversations would center around a patient's question about his or her post-operative care, and Hebo's responses to those questions. You can read more about Dialogflow below or on their official documentation.

To create these conversations, we use Dialogflow's interface to create intents, entities, contexts, and fulfillments:

- **Intent** - every possible response that we wish Hebo to respond with will need an intent. So a question of "How do I change my bandage?" that needs a visual response that steps through how to change a bandage will be an intent. Another intent will be "When can I change my bandage?" which will receive a textual answer of something along the lines of "you can change your bandage as frequently as you need, but we recommend at least once a day." Within each intent, you can program multiple training phrases, which is a simple way of programming multiple ways to reach an intent. So a training phrase for "How do I change my bandage" can be "Help me change my dressing" since both phrases should lead to the same answer.
- **Entities** - since words have synonyms, it can help to use entities to make the job easier. Entities are a way to create synonyms and variables that can be recognized by the agent. Since "dressing," "bandage," and "gauze" are all similar, you can create an entity that recognizes this relationship. When you create training phrases, you can create a single training phrase that answers "how do I change my bandage" instead of three different phrases that each use a different synonym. Entities can also be created to more easily recognize the important variables in a phrase. If knowing that a user has "bandage" versus "gauze" is important to recognize, the use of entities will help classify this distinction.

- **Contexts** - for Hebo to use follow-up questions or be more conversation-like, the use of contexts is needed. Contexts allow Hebo to understand more about the conversation and respond accordingly. If Hebo needs to ask a follow-up, such as "do you have bandage or gauze?" The user can respond with the phrase "bandage" or "gauze." This conversational flow makes sense because of the context of the follow-up. However, if a user opens up the application for the first time and just says "bandage," Hebo should actually respond with something along the lines of, "I'm sorry, I don't understand." This response comes from the fact that a conversation that begins with just the phrase "bandage" makes no sense. What about the bandage? Would the use like to know how to change it, when to change it, etc. Therefore we need to keep contexts in mind when we create conversations in Dialogflow.
- **Fulfillments** - the majority of conversations for Hebo can be done under the Responses tab in an intent. This is because most conversations have a standard textual response that can be programmed directly using Dialogflow's web application. However, for more flexible conversations that allow for more customization, we use fulfillments. Dialogflow allows for fulfillments with a webhook. You can think of a webhook as a computer somewhere in this world that you can poke every once in a while with a request. The computer will then take your request and use its program to return an appropriate response. In this program, you can use standard API calls, mathematical operations, and any other computational power to craft your response. For our case, Hebo uses information such as when a patient's surgery was to choose between two responses to return. For example, those who had the surgery within the last 48 hours should be told to not get the dressing wet, but those who are at the point of 48 hours later can freely get the dressing wet.

## How to use Dialogflow for Hebo

We will step through the creation of an intent in this section. Our intent is related to the question of whether or not you can take a shower. For Hebo, we have been programming all of our responses in a particular format:

```
(@body-parts:body-parts) (@sys.date-time:date-time) [phrase]
```

@body-parts and @date-time are entities that are used as parameters to tell our webhook more about the patient. This is an unconventional way of using Dialogflow that we used because our prototype does not utilize a database. This was therefore the only way to pass information from our mobile application (which houses information about the user) to the webhook (which uses this information to customize the

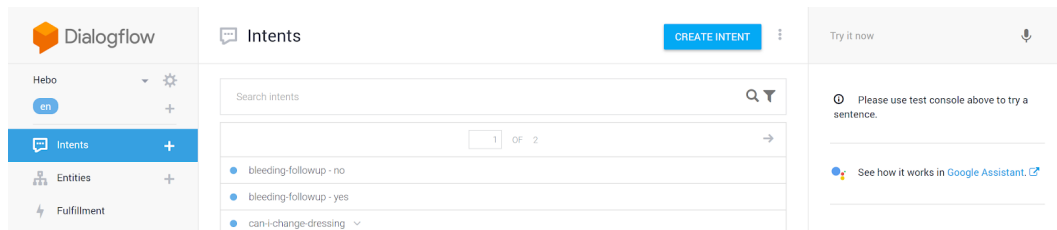
response). In a future development, the use of a database can eliminate this formatting of training phrases, where the webhook would call an API or query the database to find out about the user. [phrase] would be a training phrase that links to the intent. A few examples for our case would look like this:

```
(neck) (Feb 2, 2018 at 9 AM) Can I shower  
(eyelid) (Feb 2, 2018 at 10 AM) Am I allowed to shower  
(cheek) (Feb 9, 2018 at 9 AM) Can I get the dressing wet
```

Note: Dialogflow handles the matching and formatting of entities for you, so listing various types of @body-parts is the same as listing one for all of the training phrases (meaning we could just use "neck" for all of these examples as long as we tag it properly with the @body-parts entity). The format for @date-time is also handled by Dialogflow, so we could have just as easily written it like "(2/2/18 at 9am)."

Example: "can-i-shower"

1. Go to Dialogflow's website and click on "Go to Console." Log into Dialogflow with the hebochatbot Gmail account (in the Tools:Logistics section of this report).
2. Logging in should bring you to the "Intents" tab, which should show a list of Hebo's current intents.
3. We want to create a new intent that involves showering, so click the "Create Intent" button at the top right corner.



4. We start by naming the intent (Intent Name), since our question is involved in showering, we can name it "can-i-shower"
5. Now we should think of as many examples of questions or phrases a user can ask to get to this intent. Add these examples in the "Training Phrases" section, but also for our case, don't forget to add the parameters about the user's surgical area and date of surgery. In order to add entities to your training phrases, double click or highlight the part of the phrase that is to be coded as the entity. In our case, we first type out an example such as "(neck) (4/18/18 at

10:28 PM) Can I wash my hair?" Then we would highlight "neck" and wait for Dialogflow to prompt for us to select the type of entity. Do this for every entity and training phrase.

can-i-shower

SAVE

Training phrases ?

Search training phrases

” Add user expression

” (lips) (4/18/18 at 10:28 PM) Can I wash my hair?

” (lips) (4/18/18 at 10:28 PM) can i shower?

” (lips) (4/18/18 at 10:28 PM) I am trying to shower but I don't know how to without keeping my wound dry

” (ear) (4/11/18 at 3:32 AM) What is the best way to shower?

” (mouth) (3/22/18 at 6:43 AM) How should I be showering?

” (scalp) (4/16/18 at 9:19 PM) How can I shower with my wound?

” (nose) (4/09/18 at 8:55 PM) What should I do when showering?

” (ear) (4/11/18 at 8:56 PM) Am I allowed to shower?

” (back of neck) (4/25/18 at 5:47PM) Can I bathe now?

” (back of neck) (4/25/18 at 5:47PM) Can i wash my hair?

1

OF 2

→

The “Actions and parameters” section will start to populate with the different entities that are used in your training phrases. We don’t need to touch this portion for Hebo, but know that Dialogflow does have extra features that can be useful to development in the future (such as requiring a parameter). You can read more about these features here: <https://dialogflow.com/docs/actions-and-parameters>.

- At this point, we have programmed how Dialogflow will reach this answer, but we still need to decide how we should respond to this intent. There are three different types of responses that we can return:

22

## Types of Answers

**Textual** - Simple text response that answers the intent directly. These responses are general enough that no further action needs to take place.

To do this, just fill in appropriate responses in the Response section of the intent. If there are multiple responses, Dialogflow will randomly select one to return. You can also program in parameters into responses using the '\$'

**Responses** ?

DEFAULT GOOGLE ASSISTANT +

Text response ?

1

Yes you can shower!

2

Definitely, please do shower.

3

Enter a text response variant

ADD RESPONSES

☐ Set this intent as end of conversation ?

**Customized** - This response needs to reach the webhook for some backend processing, meaning it likely needs to use information about the user's surgery to respond.

To do this, scroll to the bottom and turn on the 'Enable webhook call for this intent' option. If this switch is enabled, whenever the intent is selected, Dialogflow will send a request to the webhook that is connected with the agent. We will continue with this example in the next steps to understand how to create a customized response.

**Fulfillment** ?

☒ Enable webhook call for this intent

☐ Enable webhook call for slot filling

**Follow-Up** -To dive more into specifics of an intent, Dialogflow supports follow-up questions. To create a follow-up, simply put the follow-up question in the Response section (similar to the Textual answer), but make sure to include a question mark at the end of the response '?'

Then, return to the Intents tab and hover over your intent. On the right side of the intent, there should be an 'Add follow-up intent' button. If you click it, it will let you create a follow-up intent to address the follow-up question. There are pre-populated responses that Dialogflow provides. See their documentation for more info.

The screenshot shows the 'Intents' tab in the Dialogflow console. At the top right is a 'CREATE INTENT' button. Below the header is a search bar labeled 'Search intents'. A list of intents is displayed, including 'bleeding-followup - no', 'bleeding-followup - yes', 'can-i-change-dressing', 'can-i-shower', 'do-i-do-the-same-dressing-as-my-doctor', 'dry-blood-stuck-to-dressing', 'Error (Fallback) Intent', 'help', 'how-can-i-stop-my-bleeding', 'how-do-i-change-my-dressing', 'how-do-i-keep-wound-from-drying-out', and 'how-do-i-put-dressing-on-unseen-area'. A dropdown menu is open over the 'can-i-shower' intent, showing options: 'custom', 'fallback', 'yes', 'no', 'later', and 'cancel'.

*Important: You do not need to add the parameters to the Training Phrases of the follow-up. Our mobile application does not send parameters if the previous response it received from Hebo was in the form of a question (it checks for the question mark at the end). This modification lets us easily use their pre-programmed follow-up intents (meaning we don't have to add '(neck) (1/1/18 at 9AM)' to each training phrase). There are certainly bugs that can be introduced to this format, so ideally a future version of Hebo accesses parameter information from a database, rather than passing information through Dialogflow.*

7. For the question 'Can I shower?' we want to respond to the user 'Yes' if the surgery took place over 48 hours ago, and 'No' if it has not been 48 hours since the surgery. We therefore need to customize their response in the webhook (if not, we would have to create two separate intents in Dialogflow, one with each response). Using the webhook requires some minimal coding experience. The current system is hooked up to the webhook, which means we only need to access the index.js file (<https://github.com/hebochatbot/hcii-postop/blob/master/Chatbot/index.js>) that houses the code. This file is found in the Github folder 'Chatbot'. The webhook takes the request and extracts relevant values to help determine an appropriate response. For our case, we mostly look at the timeOfSurgery variable. The webhook looks at the intent's name (intentName) to determine how to handle the response.

```
72 // DIALOG:
73 // Can I shower?
74 case "can-i-shower":
75     if (isAfterFortyEightHours(timeOfSurgery, currentTime)) {
76         response = speech = "You can let water run over your " + surgeryArea + ", but you should avoid putting it" +
77             " directly in a stream of water. I always recommend a bath over a shower so that you can avoid getting your " + surg
78     } else {
79         timeOfSurgery.setDate(timeOfSurgery.getDate() + FORTY_EIGHT_HOURS_LATER);
80         response = speech = "Please avoid getting the wound and dressing wet for now. Wait until 48 hours after your surgery, " +
81             timeOfSurgery.toDateString() + " at " + formatTime(timeOfSurgery) + ". When that time comes, let me know if you nee
82     }
83     break;
84
85 case "do-i-do-the-same-dressing-as-my-doctor":
```

In this example, we check if it has been 48 hours after the surgery. If yes, return one response that encourages showering; if no, return the other response where it recommends not showering. Hebo in this case also uses the surgery time to customize its response to the user. It will give the exact time when the patient can shower by calculating when 48 hours after the surgery is.

Once you add this case to the index.js file, you can save it and deploy it to the cloud so that the service uses the most updated version of the webhook. We currently use Google Cloud Platform's service to host our webhook. Follow this tutorial: [https://dialogflow.com/docs/getting-started/basic-fulfillment-conversation#setup\\_google\\_cloud\\_project](https://dialogflow.com/docs/getting-started/basic-fulfillment-conversation#setup_google_cloud_project) to help setup the environment for deployment.

**We use** `hebo-1fd69.appspot.com` **as our** `[BUCKET_NAME]`:

```
> gcloud beta functions deploy heboHttp --stage-bucket
hebo-1fd69.appspot.com --trigger-http
```



Make sure you are in the same directory as the index.js file when you are using the command in the Google Cloud SDK Shell (or else it won't be able to find heboHttp).

8. Once deployed, your intent should be ready to use! You can test it through Dialogflow's interface or even the mobile application. Remember to include the parameters since this is how our mobile application serves the requests to Dialogflow.

The screenshot displays the Dialogflow 'Intents' management page. On the left, a list of intents is shown, including 'bleeding-followup - no', 'bleeding-followup - yes', 'can-i-change-dressing', 'can-i-shower', 'do-i-do-the-same-dressing-as-my-doctor', 'dry-blood-stuck-to-dressing', 'Error (Fallback) Intent', 'help', 'how-can-i-stop-my-bleeding', 'how-do-i-change-my-dressing', 'how-do-i-keep-wound-from-drying-out', 'how-do-i-put-dressing-on-unseen-area', and 'how-much-vaseline'. A 'CREATE INTENT' button is visible at the top right of the list. On the right side, a simulation panel is active, showing a user input '(head) (1/1/18 at 9 AM) can i shower' and the corresponding default response: 'You can let water run over your scalp, but you should avoid putting it directly in a stream of water. I always recommend a bath over a shower so that you can avoid getting your scalp wet.' The simulation also shows the current context as 'can-i-shower'.

9. That's it! You can repeat these steps to add or edit conversations for Hebo. If you would like to read more about how to use Dialogflow, see this tutorial or feel free to explore their other documentation for help. We did not cover contexts (<https://dialogflow.com/docs/contexts>) in this tutorial, but know that this is used to keep conversations more natural with Hebo. Adding a follow-up intent sets up contexts for you automatically.

## Other Features of Dialogflow

It is also worth mentioning that Dialogflow also supports easy export/import of agents. This essentially means chatbots can be modular, which allows for Hebo and other future post-operative care chatbots to scale well. Imagine starting with a baseline amount of intents by importing a chatbot that knows about bleeding and swelling. From there you can build out the chatbot more by importing another chatbot that knows intents for infection.

## Training

Dialogflow is also a machine learning tool, which means training the machine can help the performance of the chatbot even more. The Training tab for Dialogflow allows anyone using the dashboard to view common intent matches to validate or invalidate them. This training helps Hebo determine whether it made a mistake or correctly identified a user's response.

When you click on a conversation, you can see everything the user says to Hebo. You can see the Intent that the phrase was matched to. If the question is correctly matched to the intent, you can click on the check mark to "Add to the intent..." and this will be added to the training model. If it does not match up the correct intent, you should click on the trash. As of right now, you cannot change the intent it was matched to so the best way is to click on the trash icon and add that phrase directly to the correct intent's training phrases. If you were to click on the circle-backslash symbol, it will add this intent to the Default Setback Intent in which it will answer "Sorry, I didn't understand that. Can you try asking again?"

The screenshot shows the Dialogflow Training interface. On the left is a sidebar with navigation options: Intents, Entities, Fulfillment, Integrations, Training (selected), History, Analytics, Prebuilt Agents, Small Talk, Docs, Forum, and Support. The main area displays a list of training examples. Each example shows the user's input, the matched intent, and the context out. The first example is "(head) (5/7/2018 at 12:25PM) can I take a shower" matched to the "can-i-shower" intent. The second example is "(head) (5/7/2018 at 12:26PM) how do I change my bandage" matched to the "how-do-i-change-my-dressing" intent. The third example is "bandage" matched to the "how-do-i-change-my-dressing - bandage" intent. Each example has a table of parameters and a set of action icons (checkmark, trash, and circle-backslash) on the right.

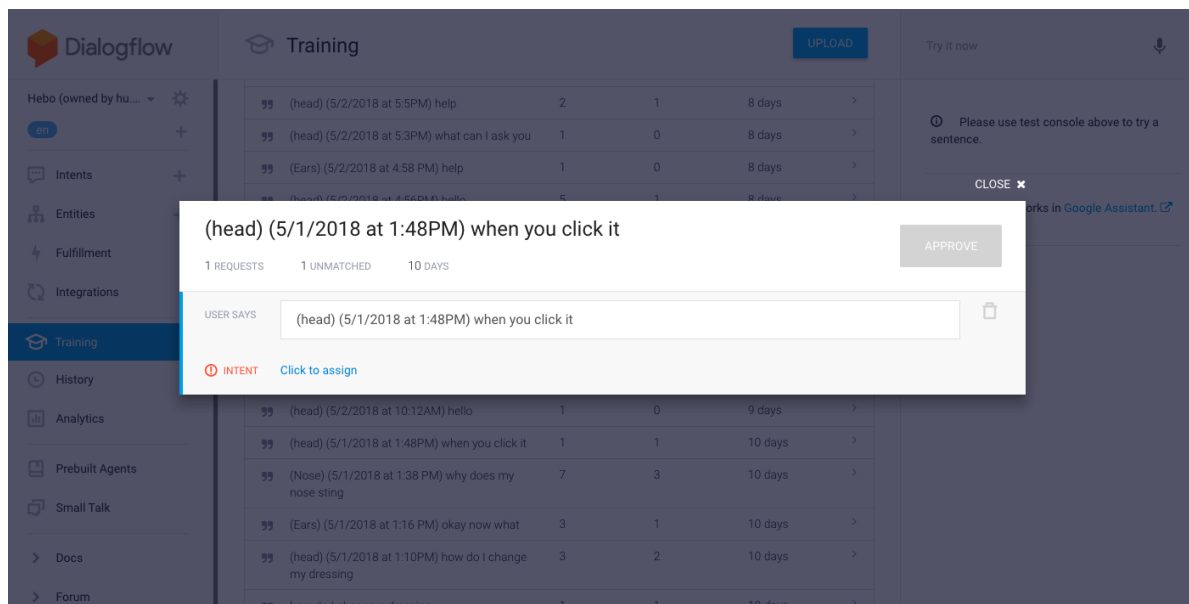
PARAMETER NAME	ENTITY	RESOLVED VALUE
body-parts	@body-parts	head
date-time	@sys.date-time	5/7/2018 at 12:25PM

INTENT: can-i-shower  
CONTEXT OUT: can-i-shower can-i-shower-followup

INTENT: how-do-i-change-my-dressing  
CONTEXT OUT: how-do-i-change-my-dressing how-do-i-change-my-dressing-followup

INTENT: how-do-i-change-my-dressing - bandage  
CONTEXT OUT: how-do-i-change-my-dressing-custom-followup

Sometimes users will ask questions that Hebo will not understand and it will show up as training like the image below. This means that the Hebo was not able to match the question to any intent. Therefore, you can click on "Click to assign" to map it to an existing intent or you can create a new one. In the future, you can come back to see which questions were not answered and create new intents based on these unanswered topics.



## Training

There is also a History section in Training that lets you view all of the requests from users. These tools provided by Dialogflow can help practitioners understand their users' questions, and whether those questions are successfully being fulfilled by Hebo. If they are not, the practitioners can correct the mismatches in an easy-to-use interface.

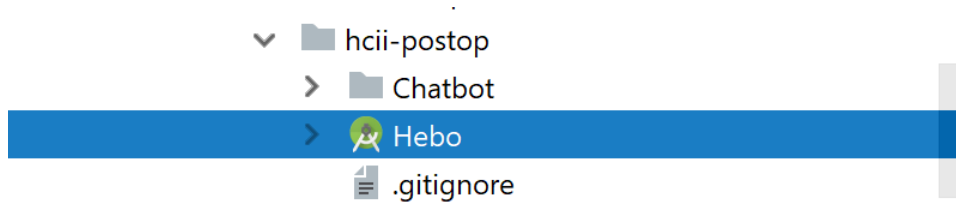
## Analytics

Under the Intents section, you can see the top intents that was triggered. This can allow the doctors to know which type of questions is being asked the most. You can also see the session flow that will map out the common conversation path. The exit % show what percentage of the interaction with Hebo ends with a specific phrase.

## Mobile Application

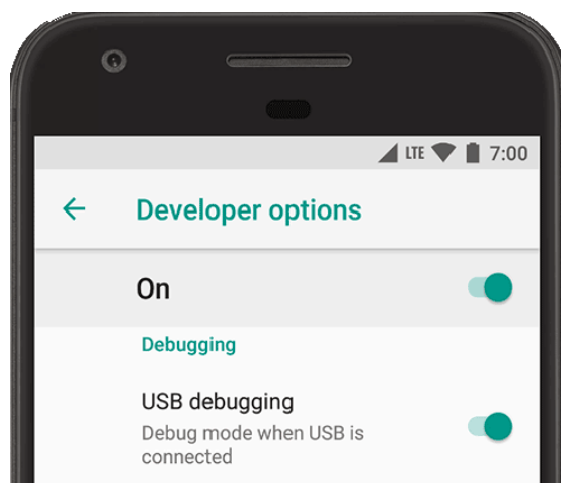
### How to setup environment and install Hebo

The repository for the mobile application is located here: <https://github.com/hebochatbot/hcii-postop/tree/master/Hebo>. You can clone or download the repository to start editing the source code. The only setup that is needed is Android Studio (or related Android-friendly IDE). Once Android Studio is downloaded, you can open the project (File > Open > [Locate Hebo directory] > OK)

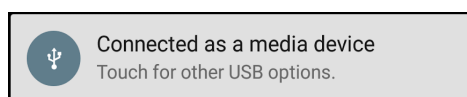


Once the project is loaded, we can deploy the program to a connected mobile device. Before this happens, you need to enable your mobile device to allow for USB debugging. To do this, on your Android phone that you want to install Hebo:

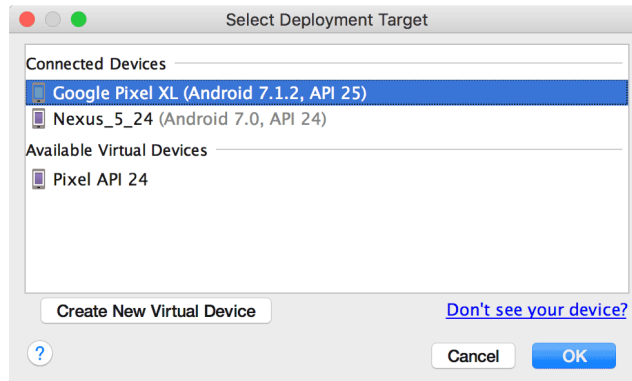
1. Open up Settings > System > About phone
2. Scroll to Build Number and tap it 7 times
3. Return to the System page and you should now see Developer options as a tab
4. Scroll down to Debugging and enable USB debugging



Your mobile device should be ready to install Hebo, but if you have issues later, you may need to follow this guide (<https://developer.android.com/studio/run/device>) for further steps. Return to Android Studio on your laptop and plug in your phone. Your phone should identify this connection in some manner:



On the top right corner of Android Studio, click the ► button. A window should pop-up that identifies your Connected Device. Select your phone and click 'OK.' Android Studio will start to build and upload Hebo to your device unless an error is identified. If a build or compilation error occurs, there is something wrong with the code. Check the build error(s).



If there were no errors, and Android Studio identifies the Build as successful, check your phone to see if Hebo is now in your applications list. It should be there, so happy demoing!

Feature	File (.java)	Description
General	MainActivity	Main file of the application: this activity is the <code>onCreate</code> function is the first that gets fired, so it's a good place to start tracing the code
	Config	Where static variables are located. If there are general values that are used, include them here to avoid "magic numbers" and hardcoding
Onboarding	Onboarding	Activity that is fired if the user has never used this application before ( <code>isUserFirstTime true</code> )
Profile (saving user information)	ProfileActivity	Activity that is fired after Onboarding and if the top right profile icon is tapped. Allows user to update information about his/her surgery
	DatePickerFragment	Used to create date and time pickers for the user's profile
	TimePickerFragment	

Chatbot (see below for more details)	Message	Java class used to classify the types of messages Hebo can respond with
	MessageListAdapter	Adapter class that inflates the appropriate Message
	Response	Java class used to classify the response
	ResponseListAdapter	Adapter class that inflates the appropriate Response
	Timer	Custom timer used by Hebo. Defines what happens when timer starts, is canceled, finishes, etc.

### List of Implemented Features

Below is a list of features that were implemented for the Hi-Fi prototype. There are likely still bugs and informalities in the code due to the time constraint of the project:

- Conversation with Hebo (Dialogflow): both text and voice are used
- Visual answers that provide more information
- Error recovery: if an Intent is repeated consecutively, Hebo assumes it is misinterpreting the user's request
- Consent and permission forms
- Profile of user: surgery site, date and time of surgery, and clinic
- Onboarding screen to introduce the application to users
- Hebo can set a timer and check back on the user (used for bleeding)

### Overview of Chatbot

Hebo's primary feature is being a chatbot, so the core focus of the MainActivity.java file is establishing this flow. The interaction starts when the user clicks the Hebo icon button (listenButtonOnClick). If user consent has been given, the app will start the microphone and listen to vocal input. Once the SpeechRecognizer stops listening, it fires the onResults function. onResults will either send the speech in its raw form (if this is a response to a follow-up question) or prepend the parameters of '(body\_part) (date\_time)' to the raw speech. This text is sent to Dialogflow through the sendRequest function. Once Dialogflow returns its response, the onResult function is called. This function is where the response from Dialogflow gets parsed and the appropriate action is taken to display the response to the user.

As of right now, there are four possible objects to display to the user. We classify these ways with the Message class and MessageListAdapter:

Type	Description
MESSAGE_SENT > addTextMessage(text, false)	Message user sends to Hebo (Dialogflow)
MESSAGE_HEBO_TEXT > addTextMessage(text, true)	Text response from Hebo. These messages just contain a single item in the ResponseListAdapter
MESSAGE_HEBO_VISUAL > addVisualMessage(stringArr)	Visual response from Hebo. These messages have an arbitrary amount of Text and Image items in the ResponseListAdapter. <code>stringArr</code> refers to the name of the visual answer that will be looked up and displayed. In the webhook the response should be in the format ( <code>stringArr</code> ) and in the <code>visual_answers.xml</code> file, there should be a string-array with the same name
MESSAGE_HEBO_TIMER > addTimerMessage()	Message with a countdown timer. Used to check back on the user (e.g. stopping bleeding)

## How to add Visual Answers

Visual answers are easy to add to Hebo, but may not be the most intuitive at first. In order to summon a visual answer, we have to program the webhook to respond with a specific text to trigger the mobile app program to go down the visual answer path (rather than plain text). We currently use the webhook to do this because we separate the `displayText` from the speech in visual answers; meaning we want Hebo to read out something different from what it will display. We do this by setting the speech to what Hebo will read out, and encoding `displayText` in the following format:

(VISUAL\_ANSWER\_NAME)

Where `VISUAL_ANSWER_NAME` is the variable name that you want to lookup in the mobile app to figure out which visual answer is returned. The '(' and ')' is what tells our mobile app to set-up a visual answer, while the string inside is used to look up the content of the visual answer. Once the webhook is set up, go to Hebo > app > src > main > res > values > `visual_answers.xml` to add in your visual answer. You can see that each visual answer is a `<string-array>` with `name=VISUAL_ANSWER_NAME`. The first `<item>` is always the title of the visual answer. Every other `<item>` within the array is either normal text or an image file name. The normal text is what text instructions the visual answer is providing. The image file name is the image file that will be looked up and loaded. So to create a new visual response, you mostly just have to fill in the content of the visual response in the `visual_answers.xml` file in its linear order.

Note: Image files must start with '\_' to identify it as an image response (rather than text). The program currently looks online in the Github folder (<https://github.com/hebochatbot/hcii-postop/tree/master/Hebo/app/src/main/assets>) to load the images. This means that adding an image requires:

1. Naming the file to start with '\_'
2. Dragging the image file to Hebo > app > src > main > assets
3. Pushing to Github the file

If the images are not loading properly, check that the URL of the image matches Config.VISUAL\_IMAGES\_URL

```
163 // DIALOG:
164 // You should put a lot of vaseline to minimize scarring. Would you like me to show you how? Yes
165 case "how-to-reduce-scarring - yes":
166     speech = "This is how much vaseline you should apply."
167     response = "(VASELINE_RESPONSE)";
168     break;
```

```
<string-array name="VASELINE_RESPONSE">
  <item>How much Vaseline to apply:</item>
  <item>You should apply this much Vaseline! The more Vaseline, the better it is for the healing process.</item>
  <item>_vaseline.jpg</item>
</string-array>
```

## TOOLS



# APPENDIX

## Helpful Links

*Request Handle:* <https://dialogflow.com/docs/reference/api-v2/rest/Shared.Types/WebhookRequest>

*Response Handle:* <https://dialogflow.com/docs/reference/api-v2/rest/Shared.Types/WebhookResponse>

*Android SDK:* <https://github.com/dialogflow/dialogflow-android-client#android-sdk-for-dialogflow>

*Official Documentation:* <https://dialogflow.com/docs/reference/v1-v2-migration-guide>