Rising Risk GA Product Documentation

What is Rising Risk?

Rising Risk is a workflow software product powered by predictive analytics.

The Problem

- Care management teams typically rely upon claims data to help prioritize patients for expedited outreach. This means that those decisions are based on delayed and incomplete data because of the claims lag. This leads to suboptimal decisions.
- Additionally, these teams usually have a fragmented, challenging workflow based on outlook calendars and disparate excel sheets.

The Value

One of our customers saved themselves $77.5 MM in a single year by focusing on 600 of their ~140,000 patients and cobbling together a similar solution using claims and Pings. This works out to over $100K per patient, per year. We conservatively estimate that our customers can save themselves at least $25,000 per enrolled* patient per year.

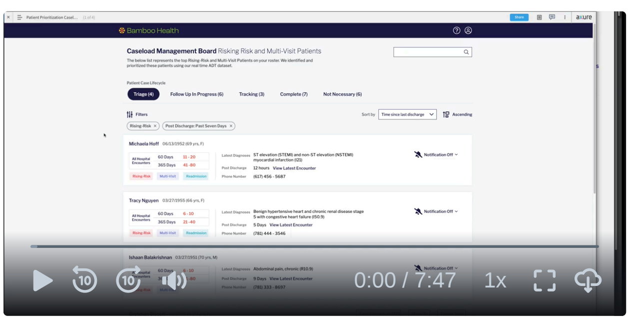
*Enrolled here means that the user identified the patient as priority, contacted them, and convinced them to participate in complex care management or another appropriate intervention.

The Solution

A workflow software product powered by analytics which learns from user input. We look at the aggregated history for each patient, make recommendations, and then look at user actions to inform the next iteration of predictions.

The goal is to optimize not only the patient identification workflow but also the patient outreach workflow in the same tool.

Preview Video



General Docs and FAQ

Customer Facing Pitch Deck

Customer Facing Preview

One Pager

Predictive Input One Pager

FAQ (Internal Only)

Support Docs

Rising Risk Key Definitions.docx (linked in app)

Rising Risk Support FAQ

Product and Technical Docs

- Product Requirements Document
- Technical Requirements and System Design
- **Internal** Pitch Deck
- Rising Risk: Addressing Health Equity Concerns
- DR for Data Integration into Ping

All Related Documentation

- Rising Risk: Analytics Pipeline Improvements
- Patient Prioritization Roadmaps
- Rising Risk: Data Integration
- Rising Risk Design V2
- Rising Risk: Redshift Tables and Training Set
- Rising Risk Recommendation Ordering
- Rising Risk Alpha Retro Document
- Rising Risk Pricing Thoughts
- Rising Risk Failure Points
- Rising Risk Analytics Implementation - Production
- Rising Risk - SMS Integration + Contact Cards
- Rising Risk -Set Next Action Reminders
- Rising Risk - Complete and Not Relevant Reason
- Rising Risk Infrastructure
- Rising Risk Application Indicators
- Rising Risk Categorical Variables
- Rising Risk Experiment Design
- Rising Risk Production Implementation and Deployment Checklist
- Rising Risk Support FAQs
- Rising Risk - Ping Feed Interaction
- Rising Risk: Module & Model Release Process
- Rising Risk Beta Feedback Sessions
- Rising Risk - Patient Specific Notifications
- Rising Risk - Leaving User Context (fka Leave a Note)
- Rising Risk Permissions
- Rising Risk Training Set and Model Training
- Rising Risk - Inclusion and Exclusion

Rising Risk: Analytics Pipeline Improvements

The purpose of this page is to outline improvements necessary to the continued operation of our analytics pipeline. PatientPing end users rely on this system to help them articulate the value of our products and services to customers and prospects. Spotlights also uses it as a primary data source, as will any efforts coming out of the Patient Prioritization Alpha. We have not made meaningful improvements to this system since 2019, but we still rely upon it every day.

Notes from 9/23 Call:

Participants: Pearson, Heidi H, Dan D, Zeek, Owen M

- This system will be transitioned to Justin Manning and team in 2022
- Nishant and team will be responsible for operating and improving the system until handoff.
    - Zeek to schedule kickoff with Pearson, Nishant, and Krista.
    - We need to begin improvements ASAP to support development / operation of the new rising risk product
- Heidi H to begin the process of rolling out change control across PP - that should help us break this system less once fully implemented.
- Heidi H to schedule knowledge sharing with Nishant, Justin Manning, et al in October to prepare them for early 2022 hand-off
- Justin likely to consult on 2021 improvements
- Longer term improvements:
    - “single pane” monitoring / alerting for PP
    - select and migrate to a single ETL tool for Bamboo Health

What Tools are we using?

- Aurora - primary application DBs
- Airflow - orchestration
- Redshift - Data warehouse
- Looker & Tableau - BI layer for Spotlights users and internal users, respectively.
- S3 - staging area between primary and secondary extraction.

What is the current state of this system?

The primary problems to solve are: Pearson [these sound like challenges that Lukasz Chlosta is uniquely positioned to solve.]

1. **Performance**: the jobs have gradually grown to 5-6 hours as the size of our data has increased. This is especially problematic since we execute the jobs twice every night.
2. **Table Lifecycles:** There are a few very expensively assembled tables which do not need to run every night. We need to get them to a different schedule, but this is more difficult than it should be.
3. **Fault Tolerance**: Changes in upstream data, especially those related to table granularity, identifier uniqueness, and incremental keys break this system in ways which are not always easy to resolve.
4. **First Class Citizen status**: We very often find out about changes only because they broke something. This is not a desirable state for a system which will support plural user facing, revenue generating products.

   a. Pearson: Propose we rollout Change Management/CAB processes across PatientPing. This will help prevent these from occurring. We faced the same in AH and this approach (it's an explicit part of the CAB process) has helped greatly.
5. **Modularity**: This codebase is configuration heavy and monolithic, which means that changes which sound easy conceptually are often challenging in practice.

What work is required for this system - short term

1. **Performance & Table Lifecycles**
   a. **Primary Extraction >> Glue:** Primary extraction from Aurora >> S3 is carried out over a cursor which is very slow. This needs to be migrated to a parallelized `mysql dump` type functionality. We have explored Glue in the past since we already have a BAA with AWS and the tool seems fit for purpose. The scope here is about 100 discrete tables.
   b. `monthly_group_roster_sizes` to monthly build instead of daily. This table is very inefficient (~6hrs) but is required for billing on a monthly cadence.
   c. **Capacity Planning** - We have been scaling our RS clusters 1-2 nodes at a time. We need a better long term plan.
      i. Pearson - [analysis by Justin and then a proper RI purchase with input by Justin/Zach/Jerimy]
2. **Fault Tolerance and First Class Citizenship**
   a. We need a better way to identify and handle (or at least isolate) breaking changes so that they don't bring the entire build to a halt.
   b. We need monitoring and alerting befitting a user facing, revenue driving system. We should get this from Glue as well.
      i. Pearson - [bigger discussion here. We need single pane of glass monitoring across our ecosystem. It's on the list, but lower than other high-value items.]
   c. This system needs a dedicated owner, now that the team which formerly supported it is part of the Data Science group.
      i. Pearson - [Nishant's team will continue to own it until the backfills we just had approved are hired, up to speed, and transition is completed.]

What work is required for this system - after the short term

1. Modularity: We need changing this system to be less complex, with less configuration and more modularity.
2. Investigate Migration from existing codebase
   a. If we could migrate the primary (Aurora >> S3) and secondary (S3 >> Redshift Staging) extraction phases to Glue or similar, there may be an opportunity to migrate the business logic from the existing code base to something modern such as DBT.
3. Get away from EC2 - if we elect to continue self-hosting, it may make sense to explore a containerized implementation that can be more easily taken down when inactive and scaled horizontally.

Diagrams and Links

These diagram shows how this system works today. Note that the Airflow machines and Redshift clusters are all hand deployed, not governed by IAC.

This system is documented here: Deployed Environments – note the many child pages as well.

It runs from the repository here: https://github.com/PatientPing/etl_analysis Connect your Github account

Deployment: We build using the jenkins jobs linked above but deploy manually by executing Ansible on the AF machines directly.

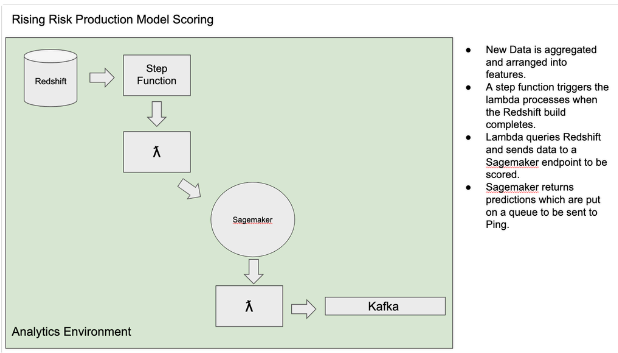
This diagram shows how a single slice of the above diagram works in more detail:

This diagram describes the schema swapping which happens with each build:

Rising Risk Analytics Implementation - Production

Building on the prototypes which we roughed out here, we have settled on the following for the production implementation of the analytics workflows required for Rising Risk.

See documentation on the data model and training set here: Rising Risk: Redshift Tables and Training Set



Toolset

AWS Sagemaker

- We created a Sagemaker endpoint which we will call to execute model scoring on a daily basis. The endpoint takes in new data, scores it using a Logistic Regression model, and returns predictions.

AWS Lambda

- Lambdas are the glue which hold the process together. They will query our Data Warehouse for new data to be scored, read the scoring back from Sagemaker, and move the predictions to the queue which will take them over to Ping.

AWS Step Functions

- Step functions are an orchestration layer for Lambdas. In this case, a step function will be the trigger to kick off the scoring process. It will itself be triggered by completion of the Airflow jobs which populate the Data Warehouse.

Key Facts

- This is a batch process and it will execute once every 24 hours.
- The python application executing within the Lambdas is test driven and deterministic

Rising Risk: Redshift Tables and Training Set

The purpose of this page is to serve as the documentation home for our Rising Risk product as it progresses through the Product Development Life Cycle

An overview of the project and the business case is here.

Analytics Engineering and Business Logic

We created a prototype data system for this purpose, with the following tables. These tables are managed by our Airflow processes and live in our Redshift Data Warehouse. They are each refreshed once every 24 hours. For clarity - our goal for this prototype was to create working data system. We do not believe that this is the final or optimal system in any way. Instead, it is something basic that will help us answer the questions required to move to the next stage of development. These tables will eventually be dropped once the product is fully developed.

Prototype Tables

Table	Link	Description
`reporting_pdf_mvp_prototype`	https://github.com/PatientPing/etl_analysis/blob/master/python/pp_airflow/redshift/public/reporting_pdf_mvp_prototype.sql	Intermediate data model which performs basic aggregations.
`pdf_prototype_training_set`	https://github.com/PatientPing/etl_analysis/blob/master/python/pp_airflow/redshift/public/pdf_prototype_training_set.sql	Our recommendations are powered by this table. It consumes aggregations from `reporting_pdf_mvp_prototype` and does the scoring of each PMPI.
`pdf_alpha_discovery_report`	https://github.com/PatientPing/etl_analysis/blob/master/python/pp_airflow/redshift/public/pdf_alpha_discovery_report.sql	I created this table to house the PII of the patients in our model, so that we can derive our alpha files easily from it and keep PII out of the scoring processes. In practice this table is combined with `pdf_prototype_training_set` in order to generate the reports. Some enhancement is needed here.

Once we were relatively confident that we would in fact make this product, we created production tables similar to but more robust than the prototype tables:

Table	Link	Description
`rising_risk_basic_aggregations`	https://github.com/PatientPing/etl_analysis/blob/master/python/pp_airflow/redshift/public/rising_risk_basic_aggregations.sql	As above, an intermediate data model which performs basic aggregations.

`rising_risk_feature_store`	https://github.com/PatientPing/etl_analysis/blob/master/python/pp_airflow/redshift/public/rising_risk_feature_store.sql	Similar logic to `pdf_prototype_training_set` but looks at all patients with >1 encounter in the last 2 years.
`rising_risk_encounter_timeseries`	https://github.com/PatientPing/etl_analysis/blob/master/python/pp_airflow/redshift/public/rising_risk_encounter_timeseries.sql	This table tracks a time series of all patient encounters since 2018-01-01. This will help us prove the efficacy of Rising Risk by looking at changes in utilization for different groups.
`rising_risk_training_pmpis`	https://github.com/PatientPing/etl_analysis/blob/master/python/pp_airflow/redshift/public/rising_risk_training_pmpis.sql	This table contains PMPIs which represent relevant recommendations in the alpha. That is, our alpha customers have told us that they were NOT managing these patients but WOULD DO SO based on our recommendations. We will use this information to train our predictive model.
`rising_risk_training_set`	https://github.com/PatientPing/etl_analysis/blob/master/python/pp_airflow/redshift/public/rising_risk_training_set.sql	we unify features from the feature store with data on user interactions here. This will form the basis of model re-training.

Model Scoring

With the "working data system" goal in mind, we created a very basic rules based model using the following features. Each feature is weighted on a 5 point scale, and the points are simply summed at the end to supply the model scoring. In production, we will use the same features but a predictive model will do the scoring.

Feature	Definition	Thought Process
`Hospitals_last365`	The number of hospitals on the PP network at which the patient was seen in the last 365 days	Higher number of hospitals = increased complexity and need for care coordination.
`Insurance_Value`	High Level Insurance classification. (Commercial, Medicare, Medicaid, Self-Pay)	We use this only to identify and give extra points to Medicaid patients as part of our focus on health equity.
`Age_Value`	Numerical segmentation by patient's age. • <30 = 1 • 30-49 = 2 • 50-69 = 3 • 70-89 = 4 • 90+ = 5	We boost to 3s and 4s, thinking that the younger patients have much more time to consume health care and are already quite sick.
`gender`	Patient's gender as communicated to us in the ADTs. We don't handle unknown and trans patients that well.	We boost female patients as part of our focus on health equity.

`States_Last365`	The number of states in the PP network at which the patient was seen in the last 365 days	Higher number of states = increased complexity and need for care coordination.
`Readmissions_last60`	The number of inpatient readmission encounters in the last 60 days	This is a clear sign that this is a patient in need of prioritized follow up as multiple inpatient readmissions indicate the systems are not working as intended.
`adjusted_encounters_last60`	The number of hospital encounters in the last 60 days, adjusted for DQ issues	We give extra points for higher levels of utilization in the last 60
`adjusted_encounters_last365`	The number of hospital encounters in the last 60 days, adjusted for DQ issues	We give extra points for higher levels of utilization in the last 365
`rising_risk_ratio`	HOS Utilization last 60 / HOS utilization last 365. Controlled for Low N by discounting the ratio when there are fewer than 6 HOS encounters in the last year.	This ratio is really the core of this work because we are trying to identify newly vulnerable and expensive patients before they become MVPs.
`ed_inpatient_ratio`	The ratio of ED::Inpatient encounters. Controlled for Low N by discounting the ratio when there are fewer than 6 HOS encounters in the last year.	ED encounters are more frequent and less costly than Inpatient encounters, want to make sure that difference is reflected in the scoring.
`dx_we_care_about`	A flag indicating an encounter with the presence of a chronic disease or behavioral health diagnosis code. Specifically: CHF, COPD, ESRD, BH diagnoses , type 2 diabetes	These are chronic diseases which typically result in high levels of hospital utilization. NOTE: This information is used in scoring ONLY and is not revealed to alpha users. It will be revealed in the UI following all existing rules.
`zip_healthcare_access`	whether a given ZIP is above the 75th %ile for “lack of health insurance” according to the CDC	These fields are the result of some great work @Heidi Schmidt (Deactivated) did to bring in “geographical determinants of health” information from the CDC, linked at the ZIP level. They will help inform our model and help us ensure that we do not introduce survivor bias into the model. See the following page for more detail: Geographical Determinants of Health: Requirements and Initial Design
`zip_chronic_heart_disease`	whether a given ZIP is above the 75th %ile for prevalence of CHD according to the CDC	
`zip_copd`	whether a given ZIP is above the 75th %ile for prevalence of COPD according to the CDC	

`zip_diabetes`	whether a given ZIP is above the 75th %ile for prevalence of Diabetes according to the CDC	
`zip_ckd`	whether a given ZIP is above the 75th %ile for prevalence of Chronic Kidney Disease according to the CDC	

Infrastructure

Rising Risk: Data Integration

Janus & Front End

Rising Risk Training Set and Model Training

The purpose of this page is to outline the steps for creating a training set for the Rising Risk predictive model. As we collect more data and implement more automation, this process will become smoother, but for now it is somewhat manual. First, some basics:

What is a training set?

A training set is a dataset which helps a predictive model learn. To do this, there are two required elements:

- Labeling
  - Also known as “ground truth”, this is a (usually) boolean indicator of positive and negative examples applied to each row of the training set.
  - Example: Did a customer respond to a specific marketing campaign? Does a picture contain a car / plane / bus? etc
- Features
  - Features are columns in the dataset which contain information about each training example.
  - In the case of Rising Risk, the features are counts such as the number of hospital encounters last 60 / last 365, number of readmissions, insurance etc. Each specific to an individual patient.

Why do we need to create a training set if we already have a trained model?

Rising Risk is a “continuously learning system”, which means that it is always learning from user input. In order to realize the learning contained in the data we collect, we need to periodically re-train the model.

How will we know we are improving?

We will maintain a model scorecard to ensure that we are continuously improving accuracy and not introducing bias.

So How do I create a training set?

As we collect more data and implement more automation, the process of creating a training set will get easier. See the table `rising_risk_training_set` in redshift. This table contains all the patients who were shown in the Rising Risk UI, and it maintains a set of features** for each. It also captures the user actions and patient outcomes from the product.

<u>**Very Important: The features in the training set can change each day. If you need to retrain based on labeling from more than ~60 days ago, you will need to recreate the features as well. See below.</u>

Until we have enough examples in `rising_risk_training_set`, we need to follow the process below for retraining:

Part 1

Access the master pmpi list and training script here: s3://rising-risk-trained-model/rising-risk-training-tools/. As you can see in the file, this labeling comes from our alpha and beta customers.

Load s3://rising-risk-trained-model/rising-risk-training-tools/rising_risk_master_pmpi_list_220818.csv as a table in the scratch schema of your choice in the staging redshift cluster if it isn’t already.

Part 2

If you need to create “as was” or historical features, recreate both `rising_risk_basic_aggregations` and `rising_risk_feature_store` with the relevant time frames by modifying the “change me” dates in the training generator script. If you are using labeling from different time periods, for example the alpha and the beta, you’ll need to create two different sets of features. For reference, the Alpha took place during summer/ fall 2021 so we have been using 2021-09-01 as the cut off date. The beta launched in May 2022 and ran through August.

Part 3

Join the features to the labeling on pmpi to create a training set. Note that there may be duplicate pmpis in the labeling - this is expected as these labels come from different time periods and different customers / users.

Note: For the training set to work in Sagemaker, the file cannot have headers and the labels need to be in the first column. I'd suggest saving the file with headers and then creating another version for implementation.

Upload your training set to s3://rising-risk-trained-model/rising-risk-training-tools following the naming convention rising_risk_training_set_YYMMDD.csv

Rising Risk - SMS Integration + Contact Cards

What are we doing & why are we doing it

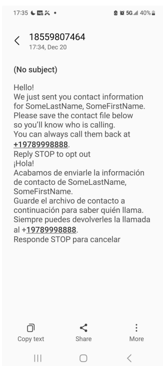
At the highest level, we are building workflow tools into our applications so that users can take valuable, ROI supporting actions.

More specifically, this feature will support user outreach to their most vulnerable and costly patients. That is - simply identifying patients our customers may want to manage via the Rising Risk platform is necessary but insufficient. The true success measure of the product will be successfully contacting and enrolling the patients in question in interventions appropriate to their situation.

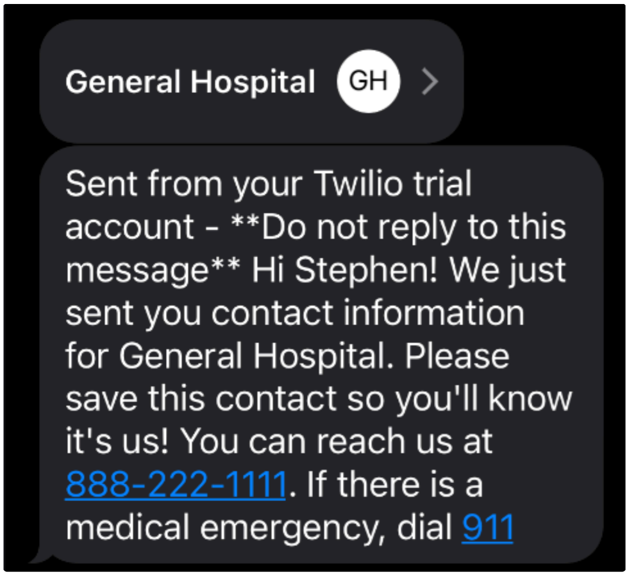
For these reasons, we have chosen to explore a text messaging feature. Specifically, the feature will allow users to send a contact card to a patient. The contact card will include all the numbers from which the customer might try to call the patient, and will include a message to the effect of “Hello, we will be contacting you from these numbers, please save this contact card so you know it’s us!”

Hypothesis: Patients will answer calls from known numbers at a higher rate than calls from unknown numbers.

Sample text message from our development environment:



<u>The example below is ***Illustrative only***</u>



How might it work

We implemented a quick POC using the following documentation: How to Send a vCard with Twilio Programmable Messaging

We anticipate that the production feature will work similarly, with the obvious enhancements to robustness, usability, and compliance.

What is required

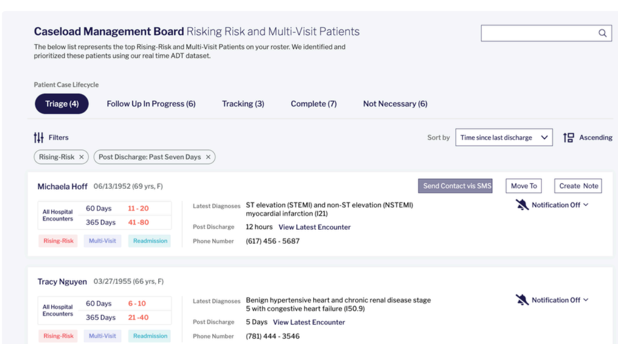
- An active Twilio account with at least one phone number to send from.
- A means of collecting the numbers from Rising Risk customers - this will be managed by Launch Services but they are likely to need some tooling.
    - There may be many numbers from which a customer might call a patient. The VCF cards can handle multiple numbers.
- A single callback number for each customer.
    - There should be only one callback number for each customer.
- An implementation of the python script which creates the .vcf file
- A place to host the completed .vcf files which is accessible to the Rising Risk application as well as an easy way for us to know which file to send.
- Compliance:
    - We have been advised not to use any identifying information in the message itself, not even first name.
    - Patients must have the ability to opt out of further text messages.
- Accessibility: We should support native Spanish speakers by either 1) defaulting to English + Spanish in our messages 2) somehow supporting language configuration.
- Message Status: We should show users message success / failure in the UI
- Related, do we have a way of knowing whether a phone number is a mobile vs. landline?
    - **What is the workflow for this**
        - When does the Modal appear in the user workflow (i.e. on Triage, on Follow Up in progress/Tracking, after the user has attempted to contact the patient once)?
        - What is the response time to validate a mobile/landline
        - Do we determine mobile vs. landline prior to the user clicking send/save
            - Do we only check once the patient is identified as Rising Risk (i.e. once a day)
        - Is there a cost for the number of calls we can make to Twilio
    - Question for Attributions: how accurate is distinguishing mobile numbers from the Roster?
    - Twilio does have a Lookup API we should leverage for this

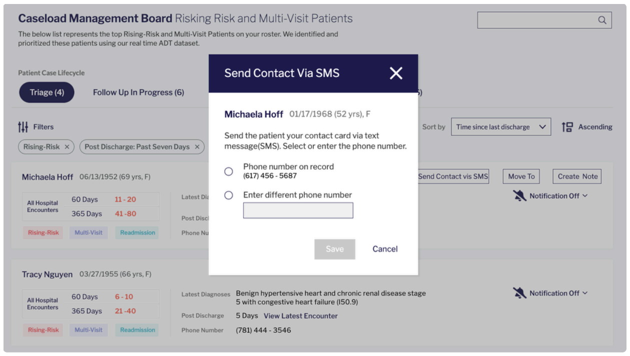
    - [Determine carrier, phone number type, and caller info | Twilio]
- The software which will create and send the requests to Twilio
- Patient Safety: Are there any non-obvious patient safety concerns here?
- The required UI elements:
  - Modal
  - Phone number lookup and “insert other phone number here” field
  - Indication on the patient card that a message has been sent, as in the clickable prototype.
  - Success / Failure toasts
  - Mixpanel instrumentation

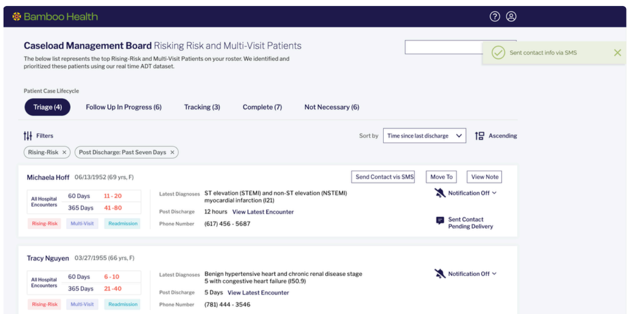
Copy

(we’ll need to validate this with customers and with legal)

```
Hello!

Hello. We just sent you contact information for {practiceName}. Please save this contact information so
you'll know it's us. You can reach us at {practicePhone ? formatPhoneNumber(practicePhone) : 'xxx-xxx-xxxx'}. If there is a medical emergency, dial
911.

OPT OUT BOILERPLATE

¡Hola!

Hola. Acabamos de enviarle la información de contacto de {practiceName}. Por favor, guarde esta información de contacto para
que sepas que somos nosotros. Puedes ponerte en contacto con nosotros en {practicePhone ? formatPhoneNumber(practicePhone) : 'xxx-xxx-xxxx'}. En caso de emergencia médica, marque
911.

OPT OUT BOILERPLATE EN ESPANOL
```



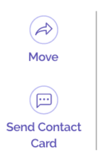




Workflow

I, as a Rising Risk User, should be able to send a phone number contact card to my patient.

1. When I view a Rising Risk Patient Card, I should be able to see an icon called “Send Contact Card”



2. Once I click on the link, a modal will pop up with the following fields:

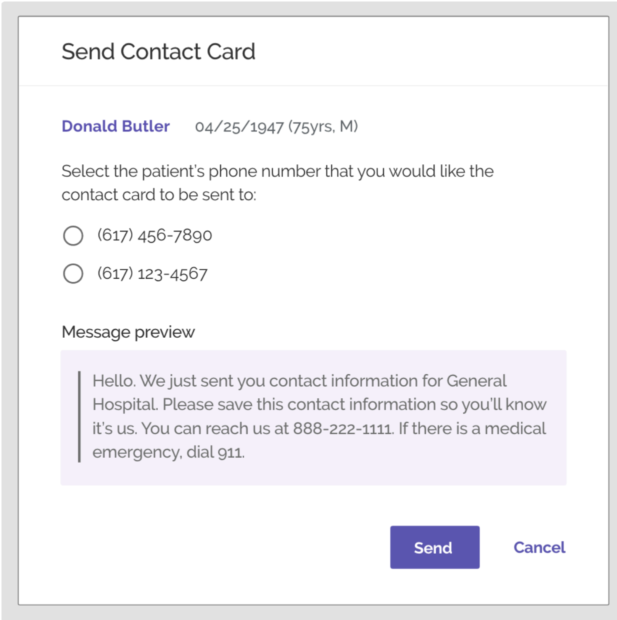
a. Patient Name, DOB, Gender

b. “Send the patient your contact card via text message. Select the phone number below”

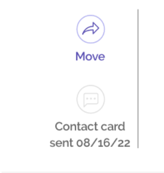
ii. All mobile patient phone numbers listed from the patient profile display

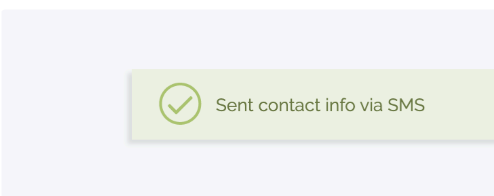
iii. A message preview will also display

1. Message preview should only display in English



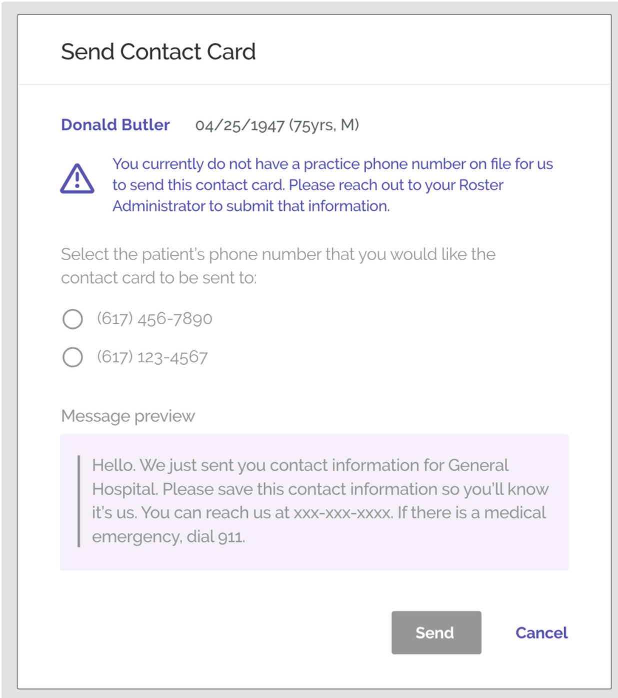
1. Once a phone number is selected, user can click “Save”. An alert will appear at the top right indicating that an SMS was sent and should be logged with the date. The icon will also update to reflect that the SMS was sent.



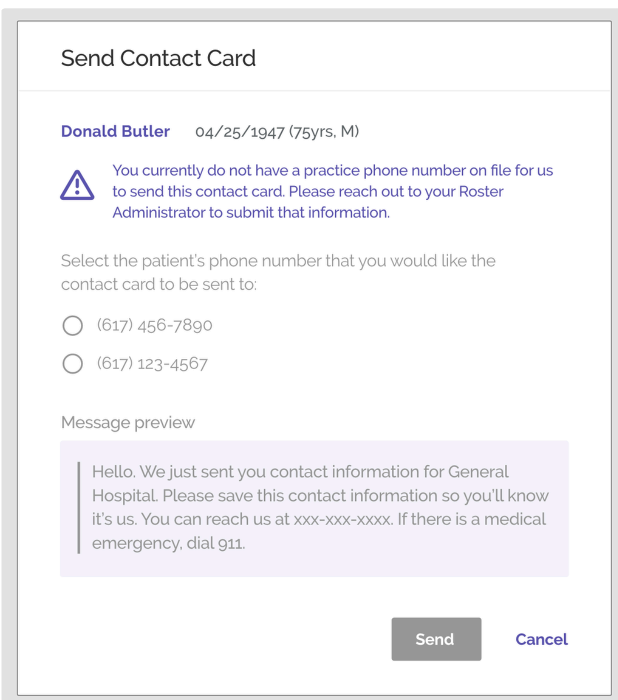


1. If no mobile phone numbers are available for the patient, the modal should have an alert/greyed out that.

Text should read: “There are currently no mobile phone numbers available for this patient to send your contact card.”



1. If no contact phone number is available, we can display an alert asking the user to contact their Organization's Administrator to update their roster with a practice phone number.



Text: "You currently do not have a practice phone number on file for us to send this contact card. Please reach out to your Roster Administrator to submit that information."

Demo Environments:

We will use a dummy phone number for the modal but will not actually send a contact card through twilio

Rising Risk Design V2

The purpose of this page is to lay out ideas for the second design iteration of our rising risk alpha. For context on the product, see the following links:

- Product Requirements Document
- Technical Requirements and System Design
- **Internal** Pitch Deck
- Rising Risk: Redshift Tables and Training Set

What are we trying to achieve?

Briefly, we are attempting to build a workflow product which will help care coordinators identify and address the needs of their costliest and sickest patients more quickly and efficiently. We are doing this because it plays to our data advantages and solves an expensive and painful problem for our customers.

What have we proven so far?

So far we have 2 live alpha clients, one more contracted, and two very warm leads. We have identified ~100 patients about whom our customers have told us two key things:

1. They are not now care managing the patient
2. They would do so based on the information we presented.

Our accuracy rate is 56%. Our goal for this prototype was >50% and we believe we can increase the accuracy with additional model development.

What feedback have we collected?

In addition to the above, we have learned:

- Consensus that this is indeed a problem and a high priority one.
- Typical workflow for this sort of task is a spreadsheet downloaded from the EMR and other data sources. Data combination issues.
- Each nurse typically carries a separate case load of ~100 pts - this means the value of a “user assignment” feature is less clear pending future discovery.
- The workflow is divided into different stages, each of which could correspond to a workflow step: *(we’ll need to think through the business logic for state changes here)*
  - **Triage** - A new patient
  - **Follow Up in Progress** - Active follow up, often several calls. *(wonder if scheduled follow up is a fit here) + notifs*
  - **Tracking** - Follow up completed, watching for events *(notifs are big here)*
  - **Complete**
  - **Follow up not necessary** *(its important that we have some kind of feedback mechanism here)*
- We will need to manage a certain level of customization for each customer, particularly around which populations are covered by the tool.
  - The most generalizable example is cancer patients, who typically have elevated levels of care coordination by default.
  - Experience tells that any given customer is likely to have at least one population which they will want to exclude. We hope to do this by grouping exclusions into roster programs and developing exclusion workflows operable by the launch teams.

- Our tech can be the best in the world and we are still depending on nurses to reach patients by phone. We would do well to develop solutions which make that workflow either easier or less necessary.
- The “leave a note” and “schedule follow up” features were called out as useful parts of the workflow.

What are our ideas for design V2?

- Incorporate SMS as we have discussed.
- Add the different stages and have a couple of patient cards in each.
- Otherwise the V1 design is pretty great and we can keep most if not all of it, we just need to flesh it out some. It definitely resonates with our users and buyers so far.
- Note: Do not take anything here literally, just trying to get our thought process across clearly - you are the designer, not us.
- Using colors here to denote different states - green for something a user can do, yellow for things that are complete, red for potential problems or inactive states. I’m sure this is not the best way to do this, please adjust as needed.