Project Name: AWS Team Safe

Problem Statement

Amazon Pinpoint did not fully support original design considerations for the project regarding two-way messaging and tracking responses of team members to each safety event. A long code must be requested through Amazon support and not available programmatically (AWS JavaScript SDK) to map a new long code to each event created for two-way alerting.

Success Criteria

1. Achieve correct mapping of responses using Amazon Pinpoint's capabilities.
2. A user-friendly SMS experience to respond to events distributed by the application.

In Scope

1. Leverage use of AWS serverless technologies
2. Set up pre-established long code to handle all app communication
3. Designate keywords for specific tasks (opt-ins, opt-outs)

Out of Scope

1. Use of third-party Communications API technology (i.e. Twilio, Bandwidth, etc…)
2. Obtaining a dedicated short code. ($900+ a month)

Use case

Manager creates a required response safety event:

1. Manager creates event and designates target team.
2. Target team members receive SMS message from application's long code and respond.
3. Team member responses are visible and tracked in event dashboard.
4. Manager receives SMS text regarding a response that needs attention.

Implementation

A new design was implemented providing the team members receiving an event. Three SMS messages are sent to each team member, the first of which contains the team manager's safety information to be distributed and instructions on how to respond to the event. Following the information message, the team member receives two response messages that include a response type and event ID. The team member can select one of the response messages, copy it, paste it back into their phone input and send.

Front End:

After each event is created using the AWS AppSync API, a callback of the API call returns the new data attributes submitted to DynamoDB for each event. The partition key is stored in the applications front end. Amazon Pinpoint API is called three times to send an information message, and two response messages that include the event's partition key.

Backend:

Each response from a team member triggers an AWS Lambda function through an SNS topic. This SNS topic is hooked to the applications singular long code. This Lambda function filters each response received by parsing the SMS message contents and maps the response to the event identified by the partition key in the message, allowing the response to show up in the correct event's dashboard.

Open Questions

Is this implementation over-engineered? Several options are implementation of multiple rotating long codes, mapping incremented responses to events such "yes1", "yes2" to solve the stated problem.

References

https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-awssupport-long-code.html