Michigan State University

Microsoft Intune Company Portal Helper Bot

Project Plan

Spring 2017

Microsoft Staff:

Kurt Seippel

Team:

Lefan Zhang

Nicholas Bunton

Ramon Niebla

David Saksa

Anh Nguyen

Table of Contents

1. Executive Summary

Since 1975, Microsoft has become a prominent leader of innovative software, hardware, and services. Its headquarters in Redmond, WA and satellite offices all over the world thrive on the idea of productivity and the empowerment of businesses and individuals to achieve more.

Microsoft Intune is a cloud-based enterprise mobility management service that aims to help workforces manage their mobile devices. With the Intune service, clients are able to control which devices are able to access company data, manage the applications installed on the devices, and make sure that the devices are compliant to company security requirements.

Intune's client base is growing at a rapid pace. The Company Portal app allows clients to access their Intune account and manage their devices on-the-go. The company portal android team actively receives bug reports from users without much information about what they are experiencing. These experiences can vary from application crashes to not knowing how to use a specific feature of the application. This leads to unorganized and cluttered data that can be avoided if the user was presented with appropriate documentation on the issue, or a way to report detailed bugs to Microsoft engineers.

A solution to fix these issues would be to implement a helper chat bot that could understand if the user is trying to get support for the Intune service or if they are trying to report a bug or give feedback.

2. Functional Specifications

The Intune Company Portal Helper Bot is an advanced natural language bot capable of diagnosing, reporting, and resolving problems customers are having with the Company Portal experience. Many Company Portal and Intune customers are experiencing problems with the application and are unable to resolve their problems. It’s difficult and time consuming for the customer to search for help via Google or the Documentation. In addition, searching via search engines proves to include many unrelated answers. The current bug reporting system is also a key problem. To submit a bug, the user clicks a button which will email the Microsoft team with log files. The log files are long, complicated files which include tons of noise making them quite useless in many cases. It does not have any more information about the bug or have any information about how to reproduce the bug. These are our two main problems.

Our primary goal is to create a bot capable of resolving problems without the need for human interaction. The first step is to create an intelligent bot capable of finding, linking, and distributing relevant articles and links to the user when they ask a question. This is done via a messaging application on a mobile device. When a user asks a question, the bot will do several things. First, it will query API.AI to parse out keywords from the question. Then, the bot takes these keywords and queries the GitHub Search API on the Intune Documentations to look for articles relating to the issue. Finally, the bot will check the response and send back articles it received. The bot gains knowledge on previous experiences to provide even more accurate results for future users. In addition to linking users to relevant articles, another main goal is to provide a better bug reporting experience. As stated above, current bug reports consist primarily of log files which do not help the user nor the Microsoft team. Our team plans on implementing a new system to report troublesome bugs back to the Microsoft team. The new process would simply be integrated into the chat bot. Whenever the bot is unable to find relevant or helpful articles the user can send a bug report instead. The bot will ask questions relating to how they experienced the bug, how to reproduce the bug, as well as being able to attach relevant screenshots to demonstrate the problem the user is having.

The third goal of our project is to make the bot more intelligent over time. To do this, the bot will need a sophisticated knowledge bank capable of teaching itself to predict the user's intentions as well as learn from past experiences. In the early life of the bot, the bot has a difficult time predicting which articles are most helpful to a user. As time goes on, the model will continually update itself in a variety of ways. First, as Microsoft adds more articles and documentation to help users, the bot will be able to use these articles to provide more accurate results. The bot is constantly keeping track of how often certain links appear, how often they are clicked, and how frequently a specific question is asked. Using a combination of this information, the bot will be able to provide more accurate results for future users. In addition, during the conversation the bot will ask questions relating to the articles. Based on this feedback the bot will update its article bank and knowledge model to provide better results for future users.

The bot will first be available for Android. The Android app is the primary goal of the project and will be the main focus for the project. Once the Android app is functioning completely, the iOS app will also be built. Because the bot must work for multiple platforms, it is important for the team to build an extensible app.

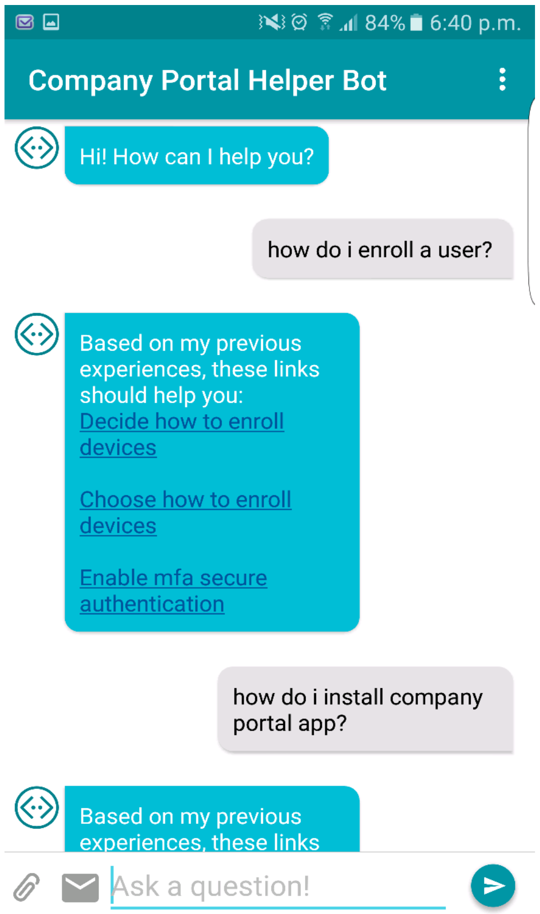
3. Design Specification

3.1 Overview

The Intune Company Portal Helper bot is designed for the user to interact and talk to the bot in a question-answer manner. Therefore, the user interface allows the user to input messages and expect the bot to return a sophisticated and programmed response. Like most messaging applications, users are able to see message history until they end the session. When the bot responds with a link that it thinks might be helpful to users, users are then able to click on the provided link to open it in their default web browser application.

3.2 User Interface

Since the Intune Company Portal Helper bot is an add-on feature to Microsoft's Intune, its user interface is kept as simple and functional for a messaging application as possible.



Company Portal Helper Bot

Hi! How can I help you?

how do i enroll a user?

Based on my previous experiences, these links should help you:
Decide how to enroll devices

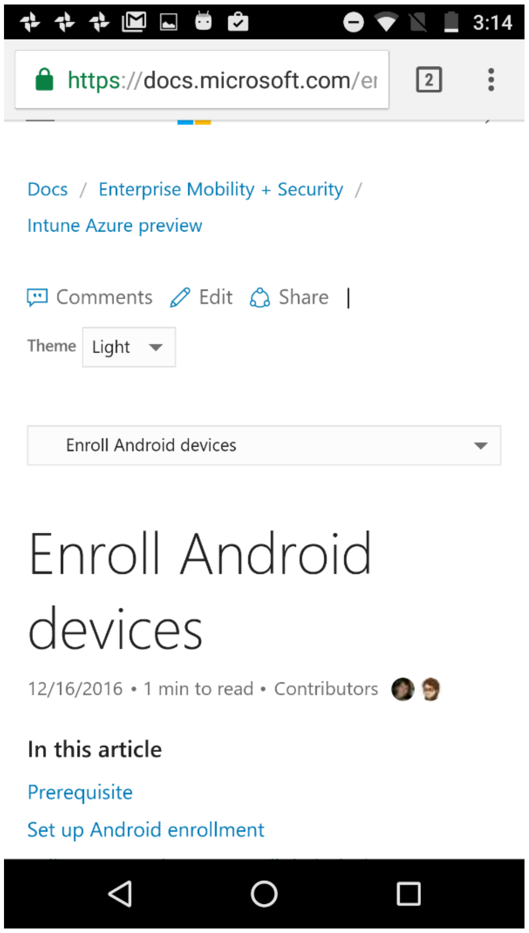
Choose how to enroll devices

Enable mfa secure authentication

how do i install company portal app?

Based on my previous experiences, these links

Ask a question!



https://docs.microsoft.com/e

Docs / Enterprise Mobility + Security / Intune Azure preview

Comments Edit Share

Theme Light

Enroll Android devices

Enroll Android devices

12/16/2016 • 1 min to read • Contributors

In this article

Prerequisite

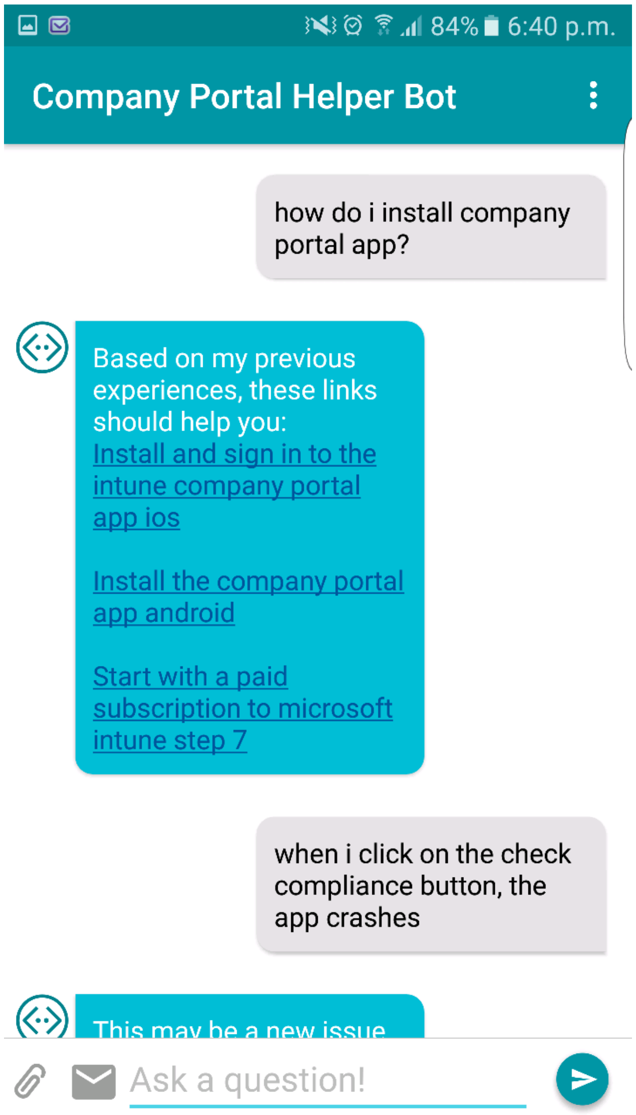
Set up Android enrollment

Figure 1: Bot responds with relevant link to appropriate support article

Figure 2: WebView that is displayed when user clicks on link

The above mockups demonstrate a basic user flow where users open the Intune Company Portal Helper Bot application and ask a question of how to enroll a user on Android platform. In return, the bot programmatically searches and responds with a link it thinks is helpful. Users can then click on the link to have it open in WebView client supported by Android as demonstrated in figure 2.

In the case where the bot is unable to retrieve a helpful answer, it continues to ask if users could provide more details. If users agree to proceed seeking for help, they can do so by rephrasing the previous question. The bot then attempts to find a sophisticated answer to return to users. Figure 3 depicts this scenario.



how do i install company portal app?

Based on my previous experiences, these links should help you:
Install and sign in to the intune company portal app ios

Install the company portal app android

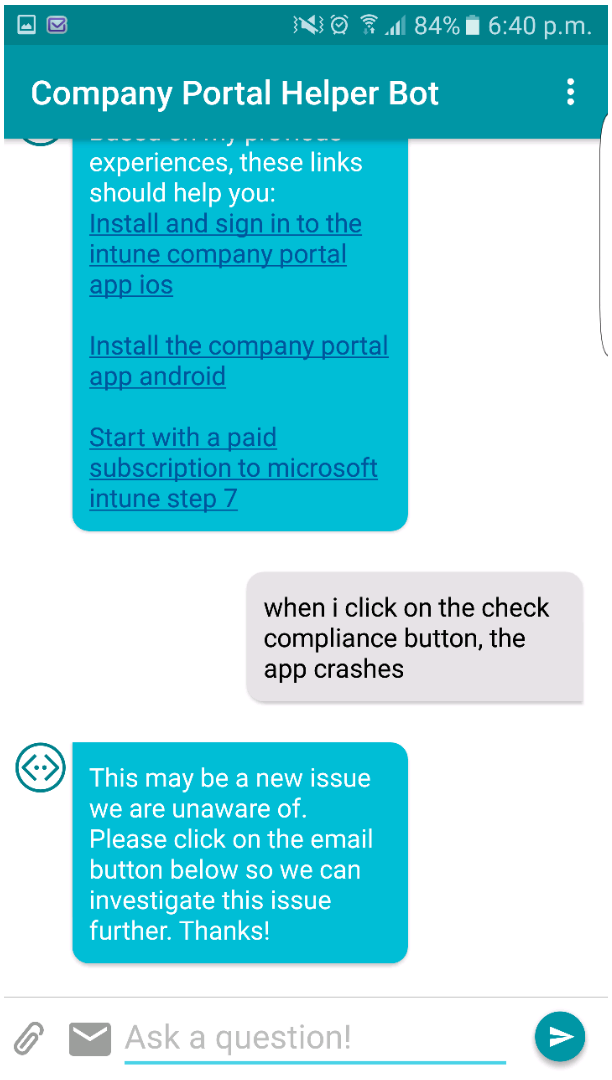
Start with a paid subscription to microsoft intune step 7

when i click on the check compliance button, the app crashes

This may be a new issue

Figure 3. Bot reattempts to find an answer.

When the bot fails to retrieve an answer the first time, it asks the users if they want to try again. If users deny, they are asked whether they would like to file a bug report. Users can continue to converse with the bot by giving more details on the issue, so the bot can store the transcript to send to Microsoft engineers, as shown in Figure 4 and 5.



84% 6:40 p.m.

Company Portal Helper Bot

experiences, these links should help you:
Install and sign in to the intune company portal app ios

Install the company portal app android

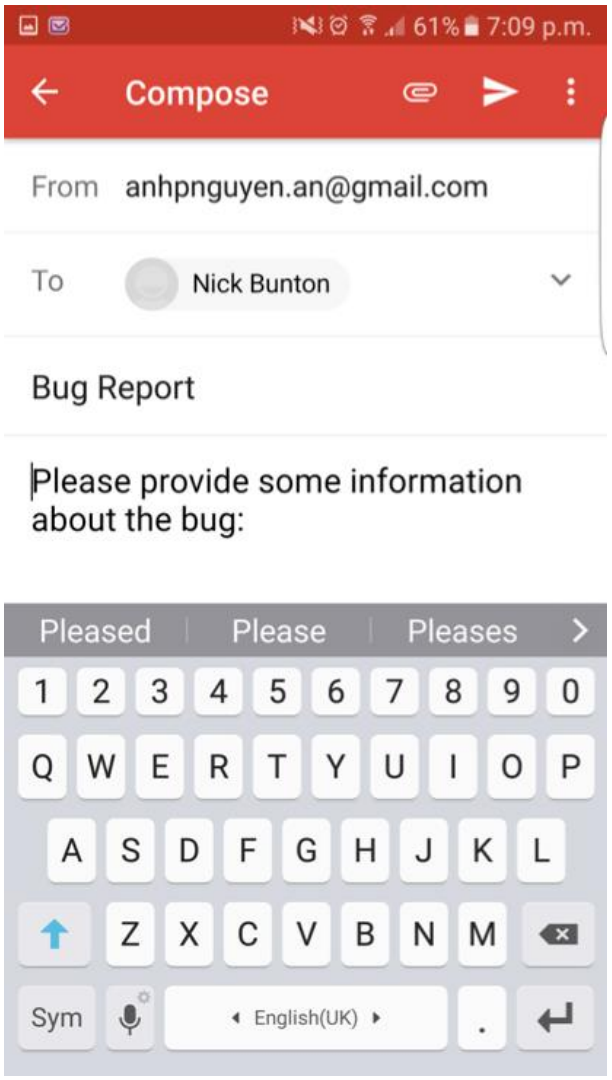
Start with a paid subscription to microsoft intune step 7

when i click on the check compliance button, the app crashes

This may be a new issue we are unaware of. Please click on the email button below so we can investigate this issue further. Thanks!

Ask a question!

Figure 4. Bot asks for bug report.



61% 7:09 p.m.

Compose

From anhpnguyen.an@gmail.com

To Nick Bunton

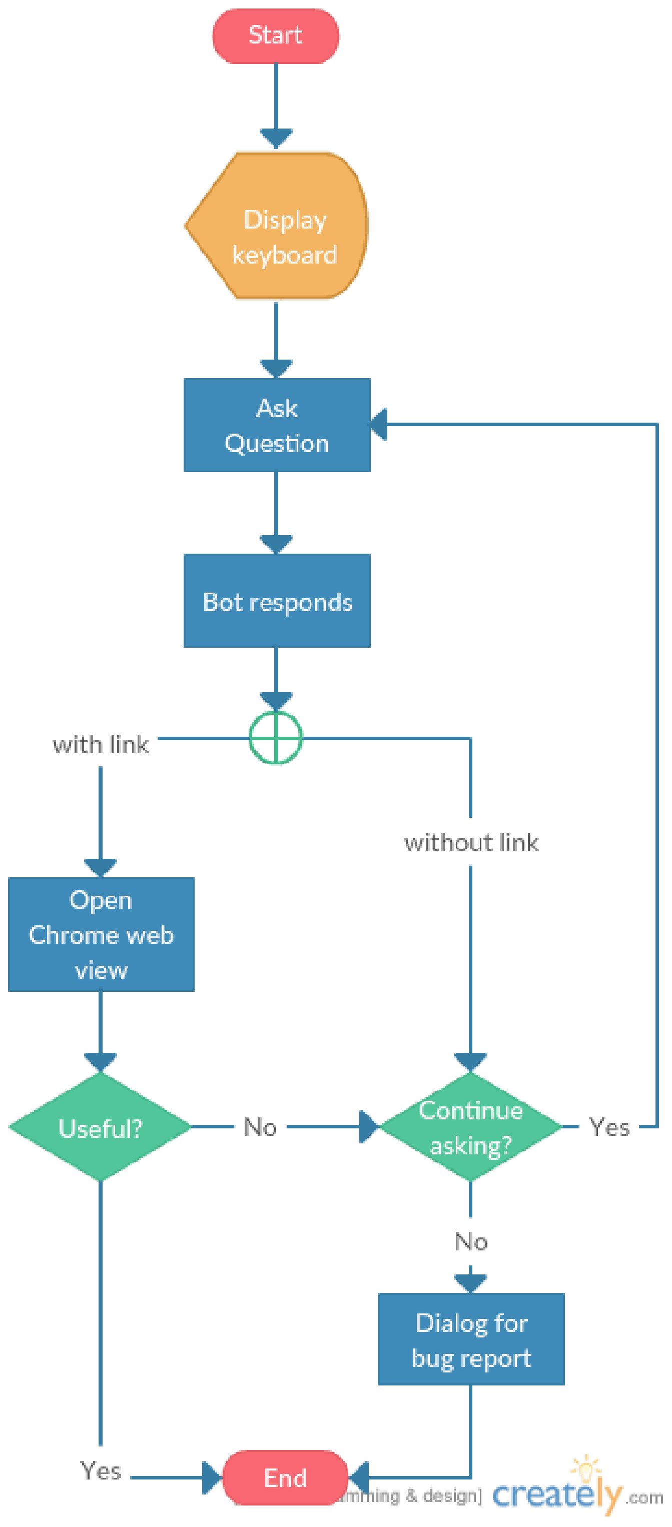
Bug Report

Please provide some information about the bug:

Figure 5. User gives feedback.

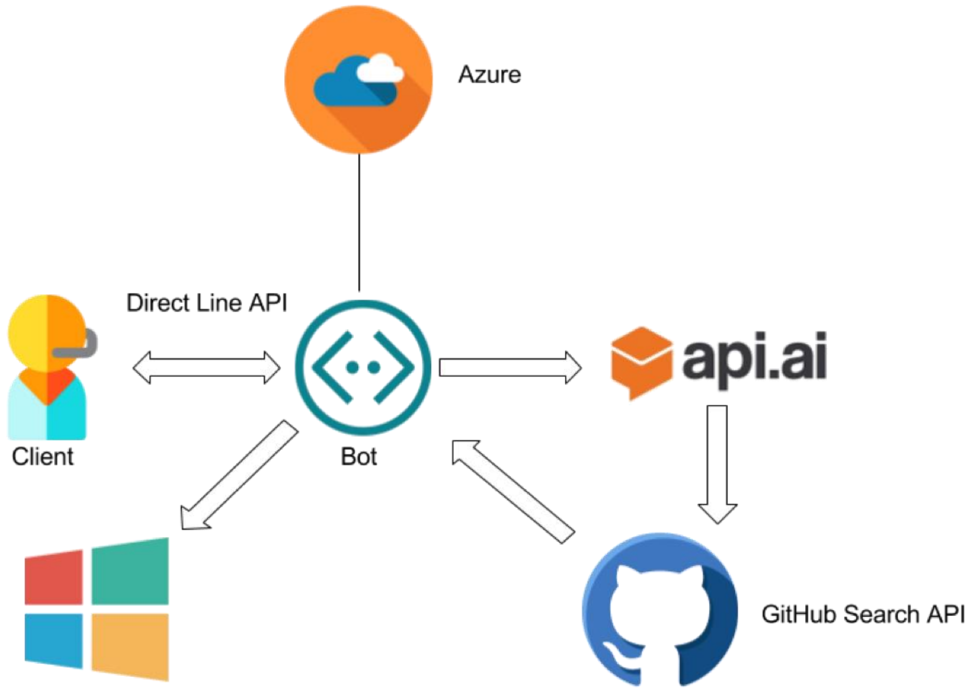
After receiving feedback from user, the bot then proceeds to send the entire conversation to Microsoft's engineering for inspection and provide further details on how to go about fixing the reported issue to client.

3.3 Process Flow



4. Technical Specifications

System Architecture



For the system architecture, our bot will be hosted in Microsoft Azure Bot Services. A Microsoft Intune user will be able to communicate using the Microsoft Intune Company Portal app. When the user sends a message to the bot, the message will be sent to the API.ai model to figure out the intent of the user's message. Once an intent has been classified for the message, then the model will parse out keywords in the message to figure out proper articles to send to the users using the GitHub search API. The article URL's will be returned to the bot, where the bot will be able to provide the article links to the Intune user to help figure out their issue. The SQL database hosted in Microsoft Azure will keep track of which articles are returned from the certain keywords of a user's message. The bot will prompt the user to see if the article they viewed was helpful or not and will keep track of that in the database in order to present the help articles in the order of being the most relevant related to the user's issue.

There are two other cases that could occur in the Company Portal user and bot conversation. If the user sends a greeting message, such as "Hello, I'm having a problem.", the bot will begin to prompt the user to figure out the specifics of the problem so it can attempt to figure out an intent.

The last case would be when none of the provided articles helped the user figure out a solution to their problem. In this case, the bot will create a transcript of the whole conversation and send a bug report to Microsoft Intune engineers so they can begin to debug the issue.

Mobile Applications

Our Android and iOS applications will allow Microsoft Intune users to have a conversation with the bot. The application will be able to communicate with the bot via Direct Line API (https://docs.botframework.com/en-us/restapi/directline/). This API allows the bot to execute HTTP GET and POST requests. The application will receive and send data as a JSON object.

Microsoft Intune users will be able to send text messages and also attach files to their messages. The file attachments (such as images) will be beneficial to the Microsoft Intune engineers as it will make it easier to understand what type of bug is occurring.

API

Our bot will be using three API's. The first one will be the GitHub Search API. This will retrieve Microsoft Intune documentation based upon search terms that is received from a user. Once retrieved, the bot will then be able to link users to appropriate articles.

The second API used is the Direct Line API. This API will connect the bot to the mobile application. Once the bot is connected, it will be able to receive messages from Intune users and send messages back to them, to help them figure out any potential issues they are having.

The last API we are using is our natural language processing model called API.ai. When a user sends a message in the mobile application, our bot will send the message to the

API.ai model which will figure out the intent of the message. If the message has been classified as a general question (the user is trying to figure out how to do something on Intune), then our bot will proceed to use the GitHub Search API to gather articles to send back to the user. If the user's message has not been classified as a general question, then our model will generate an appropriate message to send back to the user.

Microsoft Azure

The bot will be hosted in Microsoft Azure using Azure Bot Services. The Direct Line API allows our Android application to communicate with a bot in Azure Bot Services which will allow the application to carry out a real time conversation with the bot.

Also, the bot will be using a Azure SQL Database. The database will be used to be able to make the bot smarter. The bot will keep track of which articles are clicked on by the user in the mobile application. Once a user views an article, the bot will prompt the user to see if the article was helpful or not. This type of information will help the bot learn which articles are helpful and which ones are not.

Also, the SQL database will allow Microsoft database administrators to enter in any possible new bugs that have just been discovered in the Company Portal app. If an Intune user sends a message that resulted in zero help articles being returned, then the bot will search the SQL database to see if the user's message relates to any new bugs that have been discovered. If that is the case, the bot will let the user know that their issue is currently being resolved by Microsoft engineers and would prompt the user to send a bug report to the Microsoft Intune team.

Software Technologies

1. Microsoft Bot Framework (C#)
2. Api.ai
3. Android (Java)
4. Direct Line API (Java)
5. Microsoft Azure
6. GitHub Search API
7. iOS (Swift)

Development Environments

The bot will be created in Visual Studio using the Microsoft Bot Framework, written in C#. The APi.ai model will be used to create intents for the bot to recognize and then test the bot using the Microsoft Bot Framework Emulator. Once the bot is successfully tested on the Emulator, the bot will be moved to Microsoft Azure to be able to be used with the mobile applications.

The Android application will be created in Android Studio. In order to connect the Android app with the bot, the Direct Line API will allow users to have conversations with the bot in the application. The iOS application will be developed in Xcode and will also be using the Direct Line API to allow conversations.

Test Plan

The bot will be tested locally using the Microsoft Bot Emulator, in which our bot will be tested on how it responds to certain messages. The Android application will be tested by running the app in Android Studio, and the iOS application will be tested locally on Xcode. Once a prototype is developed for the bot, the Microsoft Android Company Portal team will provide real life scenario testing. The testing team will ask the bot frequently asked questions and report test bugs and then the team will receive and analyze that data to see if there are ways to improve the user experience.

5. Risk Analysis

Microsoft Bot Framework

Difficulty: Easy
Description: The framework that the team will be using to develop a natural language processing bot. None of the team members have worked with the bot framework before. This bot framework is essential for the project.
Mitigation: Review the documentation of the Microsoft Bot Framework, as well as going over various tutorials of how to use the framework.

REST API with Microsoft Azure Bot Services

Difficulty: Moderate
Description: Android mobile application will use REST APIs to send messages to the bot from the Intune Company Portal user and vice versa. The format of the messages will be a JSON object and will need to be formatted to fit into the Android application chat interface. The bot should be able to handle multiple types of data and information.
Mitigation: Review Azure Bot Services documentation and go over tutorials. Experiment with the Azure IDE available through the Azure website.

API.ai

Difficulty: Moderate
Description: Advanced natural language processing model, which has the ability to determine the intent(s) of a conversation. API.ai will parse out certain keywords in a message that will make searching for appropriate articles more efficient.
Mitigation: Go through tutorials and documentation. Create automatic models to optimize the fluidity of the conversation between the user and bot.

Android Development

Difficulty: Moderate/Hard
Description: The team must develop an Android application that allows a Microsoft Intune user to have a conversation with the developed bot. Since the team has little experience with Android this will be a challenge.
Mitigation: Reviewing documentation and going over tutorials.

iOS Development

Difficulty: Moderate/Hard
Description: Along with the Android application, the team has to develop an iOS application. With almost no experience in iOS development, this will be difficult to mitigate.
Mitigation: Reviewing documentation and going over tutorials.

6. Schedule

Stages

- **Week 1: 01/16 - 01/22**
    1. Review documentation and tutorials
    2. Get familiar with all software we are going to use
    3. Team Assignment
- **Week 2: 01/23 - 01/29**
    1. Set up meeting schedules
    2. Build screen mock up
    3. Process flow
    4. Project Design Document
- **Week 3: 01/30 - 02/05**
    1. Build Basic UI
    2. Research various searching methods
    3. Bot transcript for conversation uploading to a text file
    4. Let bot framework interact with LUIS
- **Week 4: 02/06 - 02/12**
    1. API to link back to the framework
    2. Implement a basic search engine
    3. Conservation storage and what data do we store
    4. Build up LUIS models
- **Week 5: 02/13 - 02/19**
    1. Chat interface setting on Android
    2. Come up with a score threshold for the search engine
    3. Make alpha presentation PowerPoint
    4. Practice alpha presentation
- **Week 6: 02/20 - 02/26**
    1. Pop open Chrome browser setting
    2. Get article bank up and running
    3. Figure out how to get attachment into the email
    4. Figure out intent and entities in LUIS model
- **Week 7: 02/27 - 03/05**
    1. Design Bot Icon

2. Compare and store results from previous requests
3. Link intentions from LUIS as independent functions
4. JSON Parsing (make this on C# and upload to LUIS)

- **Week 8: 03/06 - 03/12**
  1. Store the conversation on the device temporarily for retrieval
  2. Compare and store results from previous requests
  3. Send bot information to Android using API
  4. Discuss what type of intents
- **Week 9: 03/13 - 03/19**
  1. Instructional tutorial for taking and storing screenshots
  2. Make Bot smarter based on search
  3. JSON "cheat sheet" based on intentions from LUIS
- **Week 10: 03/20 - 03/26**
  1. Research on annotations on images
  2. Make bot smarter based on search
  3. Get familiar with Azure
- **Week 11: 03/27 - 04/02**
  1. Refresh button/reset the connection
  2. Make Beta presentation PowerPoint
  3. Practice for the Beta presentation
- **Week 12: 04/03 - 04/09**
  1. Refresh button/reset the connection
  2. Make bot smarter based on search
  3. Put bot in the cloud
- **Week 13: 04/10 - 04/16**
  1. Practice for status report
  2. Start working on the project video
  3. Make bot smarter based on search
  4. Build IOS UI
  5. Chat interface setting on IOS
- **Week 14: 04/17 - 04/23**
  1. Store the conversation on the device temporarily for retrieval on iOS
  2. Working on project video
  3. Instructional tutorial for taking and storing screenshots on iOS
- **Week 15: 04/24 - 04/30**

1. Working on project video
2. Refresh button/reset the connection on iOS
3. Design day

- **Week 16: 05/01 - 05/07**
  1. Finish project video
  2. Complete the application on Android and iOS