Michigan State University

Team MSUFCU

Digital Banking with Chatbots

Project Plan

Spring 2017

MSUFCU Contacts:

Samantha Amburgey

April Clobes

Ben Maxim

Michigan State University Capstone Members:

Joshua Benner

Gustavo Fernandes

Syed Naqvi

Cori Tymoszek

Chuanyun Xiao

Contents

1 Executive Summary

Founded in 1937, Michigan State University Federal Credit Union is based in East Lansing, MI. MSUFCU serves members of the Michigan State University and Oakland University communities, providing a wide range of financial services including checking and savings accounts, credit cards, personal and business loans, and financial planning. With $3.7 billion in assets, 17 branch locations, and more than 246,000 members, MSUFCU is the largest university-based credit union in the world.

In 2017, MSUFCU celebrates its 80th anniversary. A key component of a successful, long-standing business is quality customer service. Currently, MSUFCU allows members to contact a service representative by phone or through live chat on the MSUFCU website. This chat feature provides members with easily accessible assistance over 60,000 times each year.

Another crucial factor in the Credit Union's success is a competitive edge. To maintain that advantage, MSUFCU has decided to introduce a digital banking chatbot. This chatbot allows members to ask questions and complete common tasks at any time of day and with no wait time. It also reduces the number of chat requests that require attention from a human representative. This means that members with straightforward questions get an instant answer, and representatives are more readily available to help when a member has a complex request.

Finally, MSUFCU members can now access this enhanced support experience through several new platforms. The chatbot is available through Facebook Messenger on the Facebook website and in the Messenger mobile app. For members with connect home devices like Google Home and Amazon Alexa, the chatbot can be accessed by voice control. Finally, the chatbot is integrated with the familiar live chat service on the MSUFCU website. With this expanded accessibility, members can get help with banking questions at their convenience – anytime, anywhere.

2 Functional Specifications

2.1 Overview

This project seeks to demonstrate that members can receive the support they need from an automated chatbot, without losing the conversational fluidity of a true live chat. Because this conversational nature is an integral feature of the project, the chatbot is implemented on platforms that have the greatest level of convenience for users. Specifically, these platforms are the existing MSUFCU web chat service, Facebook, Amazon Alexa, and Google Assistant (via a Google Home device or an Android smart phone).

Each of these platforms offer the same core chat functionality. Users can perform tasks such as checking an account's balance, transferring funds between accounts, or setting up automatic loan payments. Each of the available functions can be requested in a conversational way. For example, a user might start a conversation with, "Hi, how do I set up automatic payments?" Then, the chatbot would ask for key pieces of information, such as the source and destination of the funds and the payment amount. When all necessary information has been provided, the chatbot connects to the member's bank account and performs the requested action.

Some additional functionality is based on the specific platform. Amazon Alexa and Google Assistant are fundamentally built around voice control technology, so members can interact with the chatbot hands-free on these devices. On the web application, a member can adjust account settings. For example, a user might not want their Alexa device to read account balances aloud. This could be disabled through the web application.

To process conversational requests, the chatbot utilizes a technology called natural language processing (NLP). The chatbot uses NLP software developed by Google and Amazon to understand a member's speech and determine the steps the chatbot should take to fulfill the request.

3 Design Specifications

3.1 Overview

Since the chatbot will be available on multiple platforms, the user interface will vary from one device to another. However, the content of the chatbot's responses and the logic that it follows will remain consistent across all devices.

For each platform, the member needs to complete some authentication upon initial use in order to enable account-sensitive functions. For Google Assistant and Alexa, account linking is available to remember a user's credentials. This allows the user to log in only once initially, and then access the chatbot conveniently with a 4-digit PIN code. Account linking is not available on Facebook Messenger and the web application, so the user will instead provide their account number and the answers to some security questions.

If a user does not wish to authenticate, some basic functionality is still available on each platform. For example, a member might ask for the Credit Union's routing number or branch hours without authentication.

3.2 Facebook Messenger

Since the chatbot integrates with Facebook's existing Messenger platform, the user interface is identical to any Facebook Messenger chat. The member navigates to the MSUFCU Facebook Page and clicks a button to begin messaging customer support. Instead of being routed to a human representative initially, the member's requests are sent beyond Facebook to the NLP service, which interprets and acts on the request. After determining the correct action, the chatbot's response is routed back through Facebook to the Messenger conversation window. An example conversation through Facebook Messenger is illustrated in Figure 1.

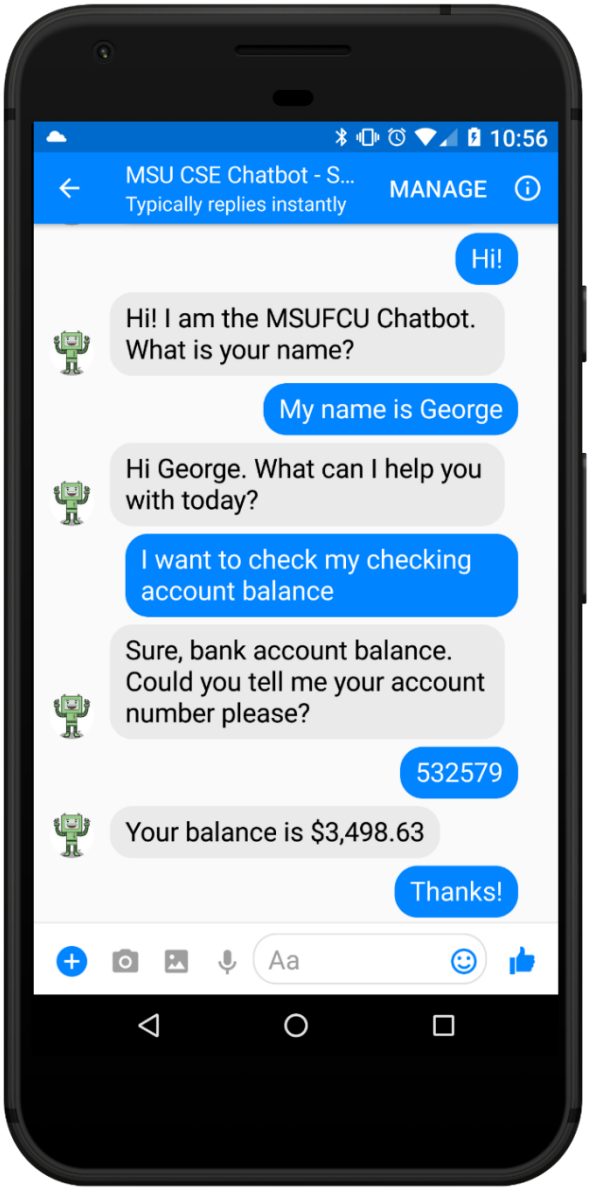


FIGURE 1: REQUESTING ACCOUNT BALANCE ON FACEBOOK MESSENGER

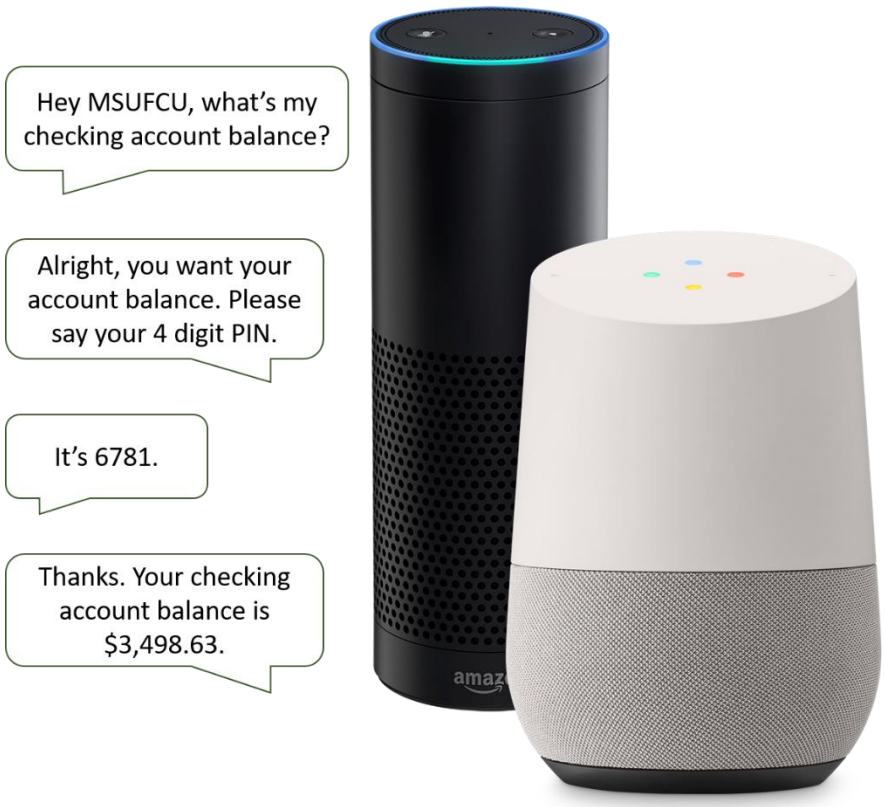


FIGURE 2: REQUESTING ACCOUNT BALANCE ON GOOGLE HOME OR AMAZON ALEXA

3.3 Amazon Alexa

With the Alexa app, users can control the flow of the conversation using predefined voice commands and intents. In order for Alexa owners to take advantage of the Alexa app, the user must first download the MSUFCU Alexa chatbot from the Alexa Skills Kit. Once downloaded, the skill is saved on the device for future uses. To activate the MSUFCU Alexa skill the user can initialize the conversation by asking Alexa to open the MSUFCU Alexa skill. Alexa will then initialize the skill and start actively listening for the voice commands. At this point, the MSUFCU Alexa app can perform 2 tasks. The user can ask for basic information like branch hours or locations. These are simple requests that do not require a layer of security as the response is general and not specific to a user. If the user would like to perform transactions specific to the user's account, the user must first login to the application. In order to login, the user would say "Log me in." Alexa will respond with a prompt for a four-digit security PIN. This PIN would authenticate the account number and password that were provided during the user's initial account linking setup. Once authenticated, the user can perform the same account specific chatbot functions that can be performed on any of the other platforms that are provided. At any point if the user would like to end the conversation with the chatbot, the user can say "log me out". At that point, the conversation and the skill will end. An example conversation through Alexa is illustrated in Figure 2.

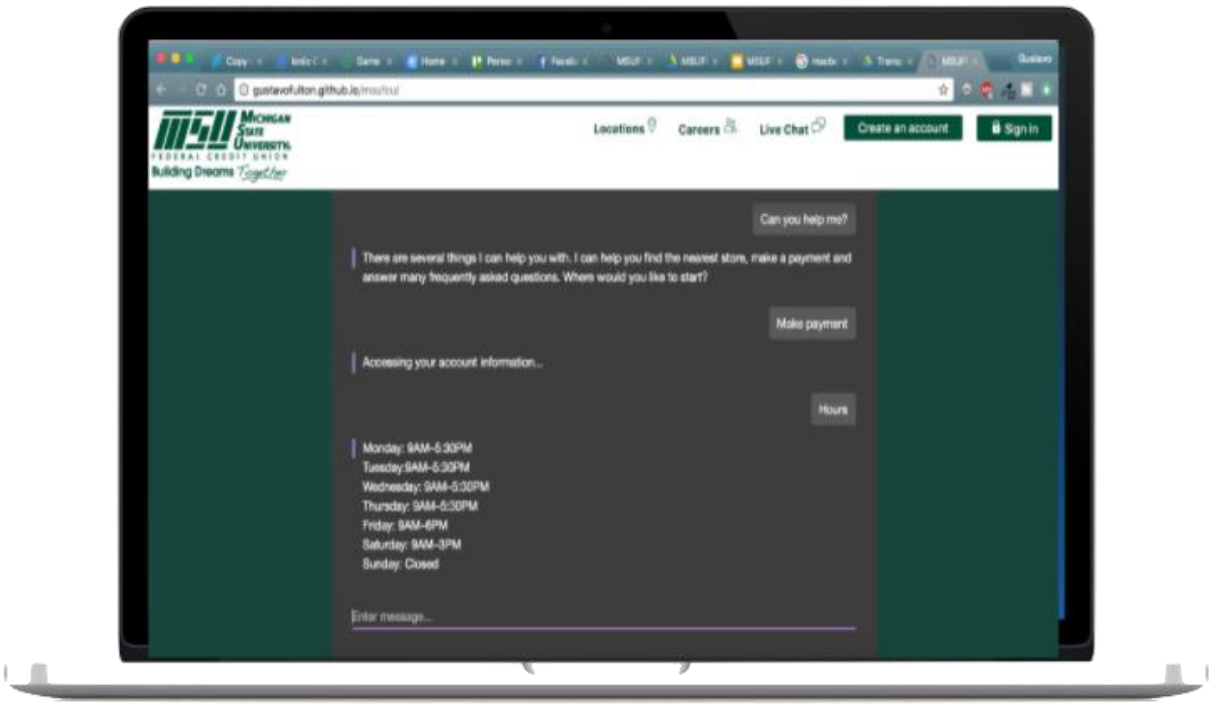
3.4 Google Assistant

Members can access the chatbot through the Google Assistant on any modern Android phone or Google Home device. Like Amazon Alexa, the member would specify to the Assistant that their request is for MSUFCU. For example, a member might say “Hey Google, ask MSUFCU what my checking account balance is.” From this point, the process is identical to the other platforms. The chatbot interprets the requests using NLP, acts on the request, and then replies to the user. An example conversation through the Google Assistant is illustrated in Figure 2.

While Google Assistant’s focal point is its easy voice control, it does allow users to type requests as well. If a member did not want to speak a request to the Assistant, they could type it into the Assistant’s chat box instead, and the chatbot would respond in the same way.

3.5 Web Application

The web application functions similarly to Facebook Messenger with regard to authentication. Account linking is not available in the web application, so the member would provide an account number and the answers to several security questions before being allowed to access sensitive account information.



The appearance of the web application is shown in Figure 3.

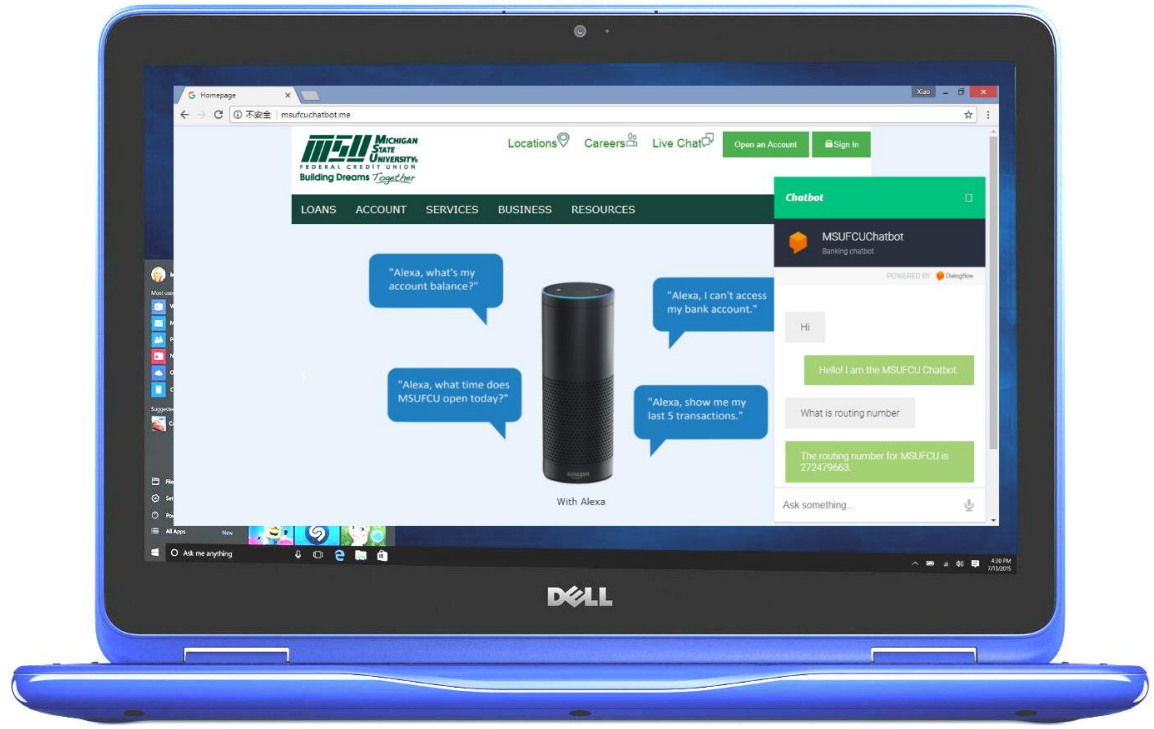


FIGURE 3: WEB APPLICATION WITH POP-UP CHAT WINDOW

4 Technical Specifications

4.1 System Architecture Overview

Besides Amazon Alexa, all other platforms are designed to use the same natural language processing backend, Dialogflow. Even though they are different platforms and the input formats vary, Dialogflow can interface with each of them. This means that the bot's responses will be standardized regardless of platform. The only significant variation exists in the Alexa application, since Amazon's products strictly use their own Natural Language Processing, Amazon's Lex and does not offer integration with Dialogflow, even though their functionality is very similar.

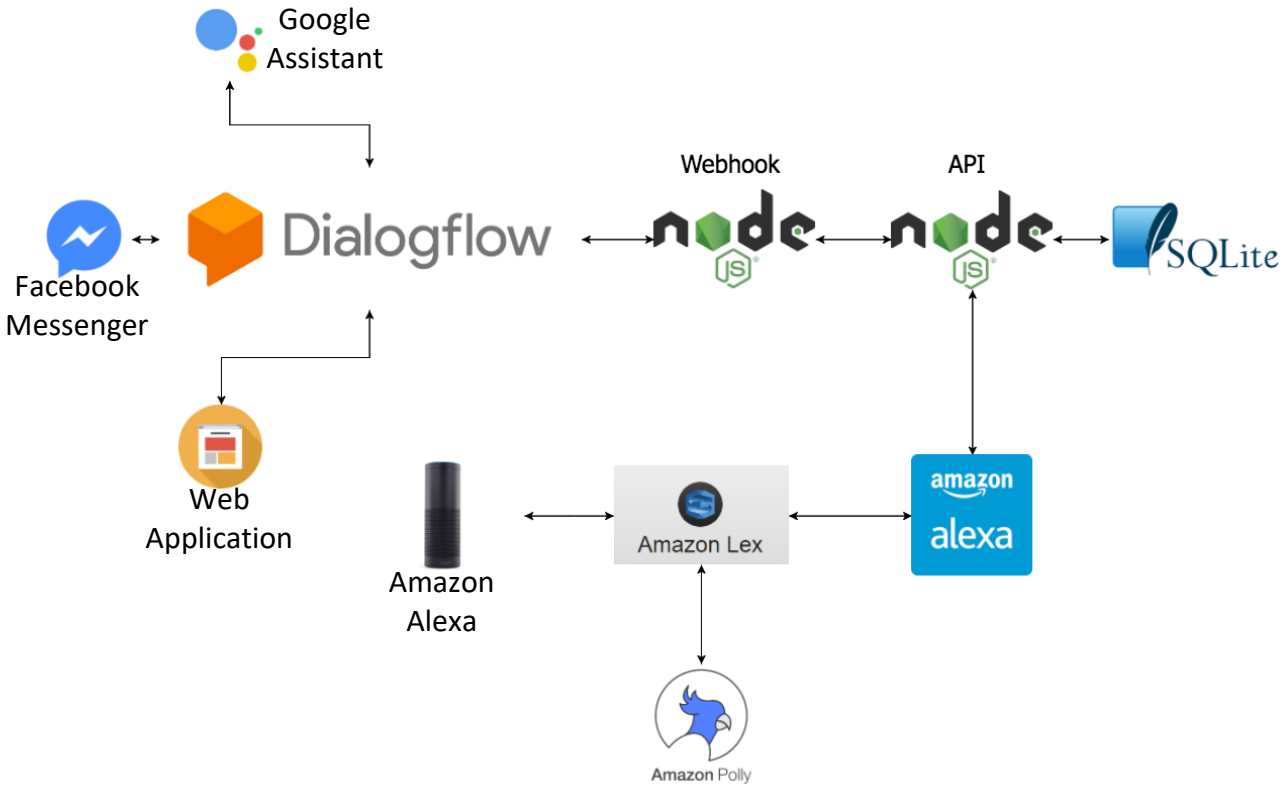


FIGURE 4: SYSTEM ARCHITECTURE DIAGRAM

In general, the chatting process begins with the user initiating a conversation by sending the first message. It can be either a greeting message or a request. In both cases, the user's initial message is sent to the Dialogflow agent, which parses the information using keywords from the user's request to formulate a response. These responses are based on topics that Dialogflow has been trained on, which will be explained in detail in the following section. For some user queries, Dialogflow can respond immediately using only its learned responses. For example, Dialogflow may ask for more information from the user, such as an account number or security questions, to make sure the user is authenticated. Other times, Dialogflow may already have the information it needs from the user. After all the information is gathered from the user, it sends a request to the webhook to check specifically what request is being made and where to

forward the API request to. After detecting the intent, it sends a POST request to the related path in Node.js API, which is where the algorithm logic takes place.

Subsequently, the API will parse all parameters sent from the webhook and if necessary, query the MSUFCU database to perform some operations and return the final information to the customer, like their checking account balance or if the transfer went through. Independently from the API's response, a reply will always be redirected back to the user, either confirming their request or stating that there was a problem with the request. Here again, the response is platform agnostic, meaning that the response is standard for all different platforms.

These connections between the different technologies in our chatbot are illustrated in Figure 4.

4.2 Server

For our server, we are using Digital Ocean and running Ubuntu with Apache as the web server. We are using Node.js version 8.6 with Express as the middleware to serve up the API. For our database, we are using SQLite which is accessed through the SQLite3 npm package. Our Alexa and Api.ai apps are hosted on AWS Lambda and Heroku respectively as they offer easier setup with their respective platforms.

4.3 Platform-Specific Considerations

4.3.1 Web

The major difference regarding API.AI integration from the web app is that the user's initial message is in a different format. MSUFCU's current chat feature requires that the customer fill out a short form to begin a conversation. This includes the customer's full name, an optional account number, and a description of their issue. This means that some additional work will need to be done to handle this information, but the benefit is that the customer will need to answer fewer questions later on.

A second consideration is that the web app requires an iframe to be created from scratch. For other platforms, such as Facebook, the chatbot would appear as a standard Facebook message. For the web app on the MSUFCU website, both the front end of the iframe and the backend in Node.js will be built.

4.3.2 Facebook

MSUFCU currently offers live chat through Facebook where the customer speaks to a live representative. When the chatbot is added, users will be able to access it through the same "Message" button on the MSUFCU Facebook page. There is no established way to implement a chatbot alongside the live chat support. Therefore, the chatbot will completely replace the live chat on the existing Facebook account. When a chat needs to be escalated, all support agents will receive an alert via email with the conversation history and link to the customer's Facebook page. The agent would then message the user directly from a second MSUFCU Facebook page designated for customer service in order to help with the escalated request.

4.3.4 Google Assistant

The chatbot will be available on Google Home and Android phones via Google Assistant. This platform is unique in that it can be accessed using voice commands. This means that before

customers' data is sent to API.AI for parsing, it first needs to be converted to text using Google's speech recognition technology. Fortunately, this does not need to be implemented manually since API.AI is fully integrated with Google Action.

4.3.6 Alexa

With the Amazon Alexa, the chatbot application will go through Amazon's Lex as opposed to Dialogflow making it different from all other platforms. With each request that a user asks for, the chatbot is looking for a specific intent from the user in order to determine the action the skill would perform in order to handle that request. With each intent, there are specific slots or keywords that are required in order to fulfill the task the user would like. Slots help define the intent the user is trying to make. Actions are performed by the skill as a response to the intent. If the user asks for their bank balance, the chatbot may require the account number as a slot. If the user does not provide the required information and simply asks for their bank balance, the chatbot will prompt the user for it. Once the required slots are appropriately filled, the AWS lambda function which defines the action the skill would take, will then fulfill the request by going to a predefined route in the API which will then query the SQLite Database to request required information and provide the retrieved information back to the user in a conversational manner.

4.4 System Components

4.4.1 Hardware Platforms

- Mobile Phones (Facebook Messenger, Google Assistant)
- Desktop (Facebook and web app)
- Amazon Echo
- Google Home

4.4.2 Software Platforms and Development Tools

- Natural Language Processing (NLP) Platforms (Dialogflow, Amazon Lex)
- WebStorm/PhpStorm
- Heroku Webhook
- DigitalOcean Server

4.5 Database

The Alexa Chatbot, Web App, APIs need database to store some basic information, account balance, permission, etc...

- ❑ Profile: For each user, there is a table to record the profile information about user's phone number, address, email address, and other basic information. Users can edit profile at website.
- ❑ Permission: User can decide their personal permission, such as permissions of transfer money, voice control through Alexa and so on.
- ❑ Fund and loan: Record user' account balance and type of account. If user have loans from bank, it will record loan account, interest rate, due date and some detail of autopay (if user enabled autopay).

- ❑ Transaction: Record the transactions' details. Include the history of transaction, status, fee and card number.
- ❑ Location: Create a set of databases to store MSUFCU's ATM and branch location and business hours. Search for the nearest location for the users.

5 Testing, Documentation and Risks

5.1 Testing

Testing is crucial to our application as any system interacting with banking must be consistent and secure. Due to the structure of our project (See System Architecture), we have testing in multiple places. To make sure our project is consistent and stable we have three layers of testing: Unit Testing, Integration Testing, and User Testing. All of our code testing is using the JavaScript Mocha and Chai testing frameworks. We are extensively testing our API as it is the most important as it powers our Alexa, Dialogflow, and Web App. For Unit Testing, we are testing all small library functions that are used across our application such as capitalizing a person's name or sending an email. We also have Integration Tests which expect a request and check that the action performed and response are correct. Finally, we are performing User Testing to ensure that our speech models work and flow well with real users and a host of issues such as accents. In addition to testing on the API, we are also using Mocha and Chai to test the Alexa Skill and Dialogflow webhook.

- Test logic flow of login to see if chatbot only allows financial services after user is logged in.
- Test database to chatbot through API connection to make sure that only requested data is correctly pulled from schema and displayed on screen
- Test chatbot to database through API connection to make sure that data is updated correctly in the database schema
- Test conversation portion of chatbot by making small talk to ensure chatbot responds with context
- Test voice commands on the Amazon Alexa to ensure correct NLP processing is performed and the correct intents are used for each action
- Ensure that Twilio API is working correctly by sending multiple messages from various phones to ensure chatbot can still respond to each connection

5.2 Project Management and Documentation

5.2.1 Project Management

We are using a Trello board to track the status of all features and bugs with states such as "To-do", "In Development", "Ready for testing", "In testing", "Unit Tested", "Done". Each is labeled with its respective platform, who's working on it, and when it is due. We are also using a Google Drive to store all our presentations and other assets and documents. Our timeline is being managed through an Excel Spreadsheet and we are using Slack as a communication tool to interact between team members as well as ask any questions with our client team members.

5.3.1 Documentation, Code Reviews, and Branching

For code documentation, we are using a private GitHub repository which is broken into separate folders for each platform and app. We are using separate branches for each platform, and team members will create a new branch for each new feature or bug they're working on. To keep code quality high, we are performing code reviews and require at least one approved

review before code is available to be merged. We also have a master and dev branch and both require the code owner to merge all pull requests. Every feature requires a pull request into dev and must not have any merge conflicts with the dev branch. Due to all the code being in Node.js, we are using JSDoc as a documentation engine to automatically generate HTML pages based on our file and function headers. This documentation is uploaded to our website.

6 Risk Analysis

Risk 1: Authenticating multiple users through Amazon Alexa

Difficulty: Moderate

Importance: Moderate

Description: Amazon's Alexa cannot securely manage two user profiles who are using the same device making the MSUFCU Alexa App difficult to share.

Mitigation: Incorporate a texting feature in the MSUFCU Alexa App upon login that will send a randomly generated code through SMS to the user's phone in order to authenticate the user whilst also allowing share-ability.

Data Interception through transit

Difficulty: Moderate

Importance: High

Description: The API that obtains data from the database may not be secure as information travels to and from the chatbot meaning that the data can easily be stolen.

Mitigation: Encrypting the channel that the API uses to connect the database schema to the chatbot would protect the security of the data.

~~Implementing chatbot with iMessage~~

~~Difficulty: Hard~~

~~Importance: High~~

~~Description: The documentation regarding implementing chatbots in iMessage is limited, making it difficult to understand and grasp the requirements of implementing the chatbot in iMessage.~~

~~Mitigation: Discuss with the client to understand the requirements and needs of the iMessage chat bot for implementation purposes. Understand the structure of iMessage to see how to create a prototype.~~

Client's requirement for chatbot integration

Difficulty: Moderate

Importance: High

Description: The client requires the chatbot to be integrated with iMessage, SMS, Mobile App, Web App and Amazon Alexa. As a team, we have to decide upon the best NLP API to create the chatbot that has the ability to be integrated with all or most of the platforms.

Mitigation: Decided upon Google's API.ai as the best fit for the chatbot as it has the most integrations for the platforms. Furthermore, API.ai also incorporates contextual conversation which is a requirement from the client.

7 Schedule

Week 2 (09/11 – 09/17)

- Due: Status presentation (09/12)
- Design wireframes and screen mockups for web app and mobile app
- Finish Project Plan document and presentations
- Develop sample apps for Amazon Alexa, Android app, web app

Week 3 (09/18 – 09/24)

- Due: Project Plan document and presentation (09/18)
- Design config file to get demo web app running
- Setup permissions and SSL certificates
- Develop demo app on android for testing of chatbot
- Create MSUFCU Chatbot in Amazon Alexa using Alexa Skills Kit
- Create an API that will be connected to database schema

Week 4 (09/25 – 10/01)

- Integrate database schema with chatbot which stores sample data with the developed chatbot
- Begin testing chatbot on all platforms to ensure sound integration with web app and Amazon Alexa as well as integration with the database schema
- Begin developing chatbot with specific functionality commands
- Begin preparing for Alpha Presentation

Week 5 (10/02 - 10/08)

- Setup iframe in web app for location of chatbot
- Setup JavaScript calls and create a user feedback page
- Add more contextual conversation commands in chat bot
- Set up Facebook account and page

Week 6 (10/9 - 10/15)

- Connect Google Action to Facebook page
- Implement pass tokens and formulate and send responses
- Prepare for Alpha Presentation

Week 7 (10/16 - 10/22)

- Due: Alpha Presentation (10/16)
- Develop more features in Amazon Alexa with voice commands by using Amazon’s Lambda

Week 8 (10/23 - 10/29)

- Implement final commands to Chatbot and finalize core functionality
- Ensure all platforms are connected and running smoothly by running tests

- Ensure API is securely connected to database and can handle multiple requests

Week 9 (10/30 - 11/1)

- Implement the live chat services
- Obtain the MSUFCU business hours, locations of ATM
- Test live chat and chatbot

Week 10 (11/6 - 11/12)

- Implement the transfer funds and payments
- Check Amazon Alexa and Google Action voice controls work well
- Prepare the Beta Presentation

Week 11 (11/13 - 11/19)

- Due: Beta Presentations (11/13)
- Integrate with Website, Mobile app and Chatbot (Amazon's Alexa)
- Checking the live chat works well
- Testing and fix problems

Week 12 (11/20 - 11/26)

- Due: Team Status Reports (11/22)
- Check bugs and fix the problems

Week 13 (11/27 - 12/3)

- Final test, bug fixing and check
- Works on the project documentation
- Works on the project video

Week 14 (12/4 - 12/10)

- Due: All project (12/06)
- Due: Design Day Setup (12/07)
- Due: Design Day (12/08)

Week 15 (12/11)

- Due: Project Videos (12/11)