Michigan State University

Team Amazon

Asa: Amazon Shopping Assistant

Project Plan

Fall 2016

Amazon Staff:

Derek Gebhard

Michigan State University Capstone Members:

Evan Moran

Samuel Chung

Yiming Li

Renee Dennis

Aaron Beckett

1 Table of Contents

2 Executive Summary

Amazon is one of the largest, most well-known electronic commerce and cloud computing companies in the world. It was founded in 1994 and is currently headquartered in Seattle, WA. Over the years, Amazon has consistently provided its customers with a streamlined interface for purchasing a wide range of products from a wide range of vendors through the web.

Amazon's success comes from constantly engineering and developing products with the customer and user experience in mind. Amazon effectively brought shops to the people by changing the shopping experience from something you had to go do to something you could do wherever you had internet access. As people spent more and more time on the internet, this strategy proved very effective. Recently, messaging platforms have surpassed traditional social media sites in popularity, and once again Amazon seeks to bring the shopping experience to the space where users already hang out by introducing Asa, the Amazon Shopping Assistant.

Asa is a chatbot which users can interact with on various messaging platforms. Asa is currently available on Facebook Messenger and provides customers with a new way to interact with Amazon.com. Users can prompt Asa at any time with a single message to find a specific product or to provide a product recommendation based on their past purchases. Asa occasionally sends users tailored suggestions for products they may be interested in. Much like a real assistant, users can ask Asa to remind them to do something, such as buying a gift for their mother or purchasing textbooks before the start of the semester. Asa is initially available for Facebook Messenger with expectations to expand to other messaging platforms in the near future.

3 Functional Specifications

The primary goal of this project is to bring an Amazon shopping experience to messenger platforms. Amazon recognizes the value of interacting with users in whichever space they spend the most time and that space has increasingly become messaging applications. To fill this need, Amazon developed the idea of an Amazon Shopping Assistant named Asa. Asa is a chatbot that users interact with on messaging platforms. She adds a level of convenience to the shopping experience by interacting with users through natural language and eliminating the need to leave a messaging app in order to search for products on Amazon's website.

Shopping from messaging platforms is simply a conversation between you and Asa. There are two ways in which conversations can begin. Firstly, a user can initiate conversation by sending a message to Asa directly. For example, if a user would like to find a specific book by its title, they would say, “Can you find The Glass Castle?" and Asa would promptly respond with the top listings on Amazon.

As users purchase items via Asa, she gradually builds a model of each user. Asa uses this model to perform product recommendations. Instead of searching for a specific item, users can ask Asa to recommend an item and she will use the user’s generated model to suggest items she thinks they will like. A good example of this would be asking for a book recommendation. Users can request a recommendation by saying, “Can you recommend a good book?” or by selecting the “Personal Filter” option during the search process.

Besides purchasing items, users can also ask Asa to remind them to do something. For example, if a user needs to remember to purchase a gift for their mother’s birthday, they can ask Asa to remind them on a specified date by saying, “Remind me to get a present for my mom on 05/01/16.” Asa will send that reminder on 05/01/16 and users can purchase the gift directly from Asa if desired.

The second way conversations start is if Asa initiates them by sending the user item recommendations based on their user profile. For example, if a user purchases a lot of video games through Asa, Asa will use their generated user profile to send one or more items they may be interested in based on their video game purchases.

Each time Asa provides a list of items to the user, they can further narrow down their search through various search filter options. If a user says, “Search for Harry Potter,” the user may be interested in one of many categories (books, movies, apparel). Asa lets the user filter their search by such categories and further limit their search up to three filter levels deep.

Once the user finds an item they want, they can choose to purchase this item by selecting "Checkout," which sends them to a virtual cart they can purchase through Amazon. If that item has variations, such as the size or color of a T-shirt, Asa prompts the user to choose between the available options before redirecting them to the cart. In the future, Asa will be able to guide users through the checkout routine but this is outside the scope of the current project.

Asa will first be available on Facebook Messenger; however, the core processing units are built to allow for easy extension to various messaging platforms such as Slack and SMS/text messaging.

4 Design Specifications

4.1 Overview

The interface for Asa is entirely dependent on which messaging platform is being used. At first, Asa is only available on Facebook Messenger to anyone with a Facebook account. To access Asa, any Facebook user can locate Asa's Facebook page named, "Asa – Your Shopping Assistant" by searching its name on Facebook. They then can message Asa directly by selecting "Message." The next several pages detail the different user interfaces one may encounter while conversing with Asa from Facebook Messenger.

4.2 User Interfaces and Use Cases

Figure 1 is the initial screen a user sees when they create a conversation thread with Asa. There is a page icon, page title, short description, and a “Get Started” button. When a user selects the “Get Started” option, Asa sends a help message that elaborates on the possible queries a user could send and the user can now message Asa. Figure 2 illustrates this and a simple small-talk session with Asa, complete with a message from the user that Asa does not understand. This occurs if the natural language processing platform’s confidence level for a message’s intent is below a defined threshold.

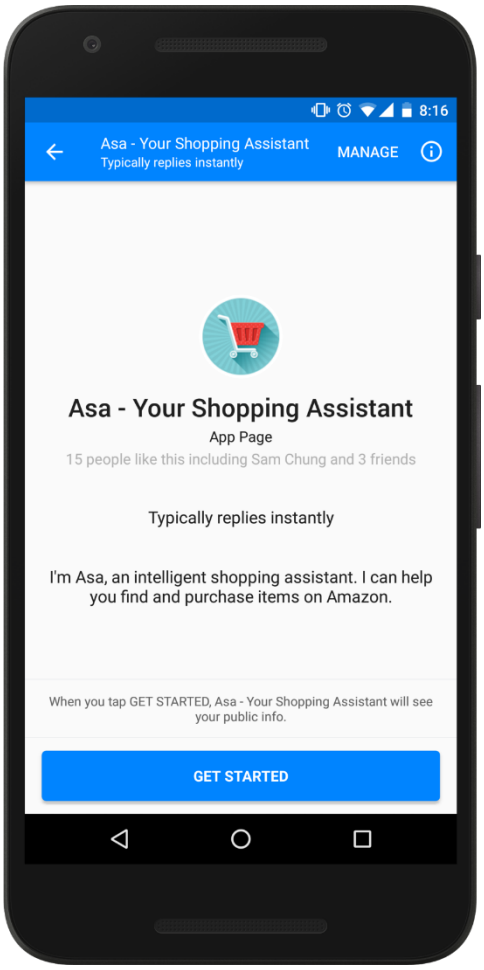

Figure 1: Starting page upon initiation of conversation with Asa

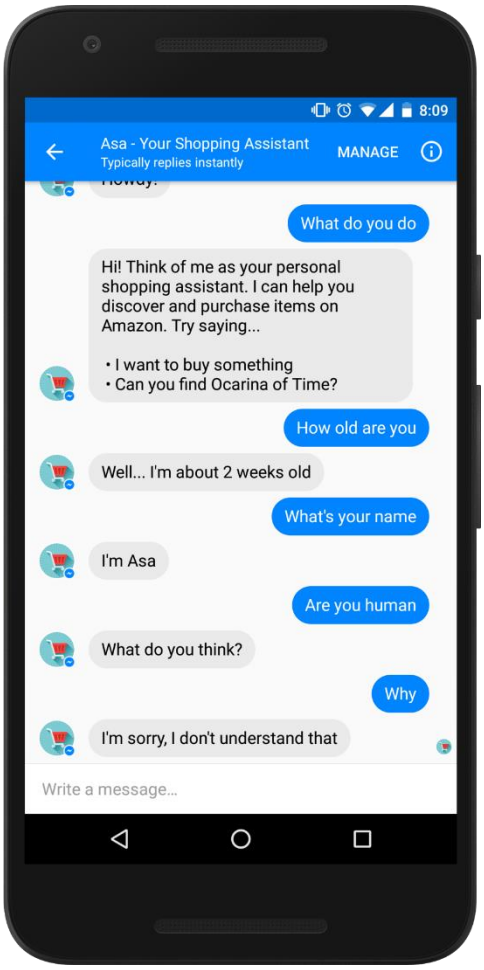

Figure 2: Example of small talk and an unclear message intent

Figures 3 and 4 show what a user can expect if they query Asa for a specific item (such as a book title, video game title, or a piece of clothing). Asa returns a list of items that best matches the user's request. Each response is complete with an image of the item, the item's name and the item's price.

Figure 3 exemplifies a single item lookup where no additional options need to be specified. Users are given the option “Checkout” which, when selected, redirects users to a cart on Amazon that contains the item they chose. Figure 4 is an example of an item that has additional options associated with it (such as size, color, or quantity). Choosing “Select Options” starts a dialogue with Asa to determine the additional specifications before adding an item to the cart.

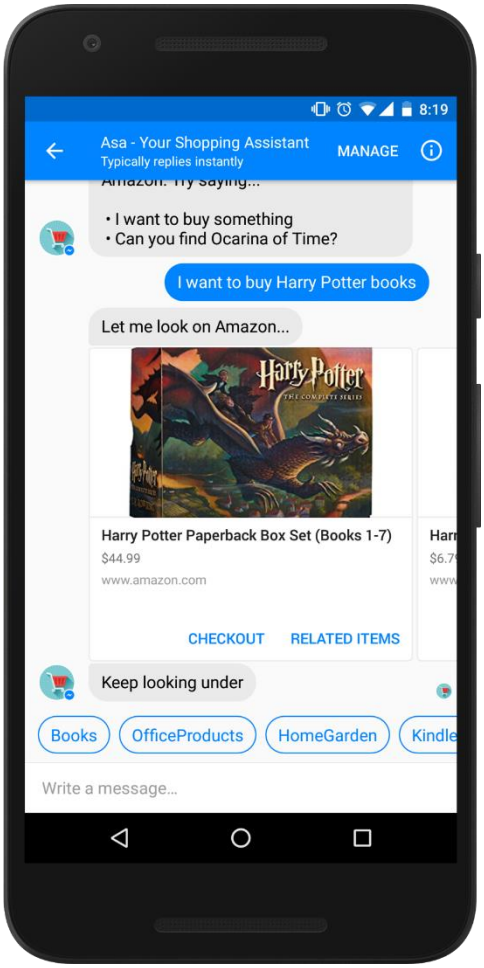

Figure 3: Example of a user requesting a specific item that does not require additional options before adding to a cart

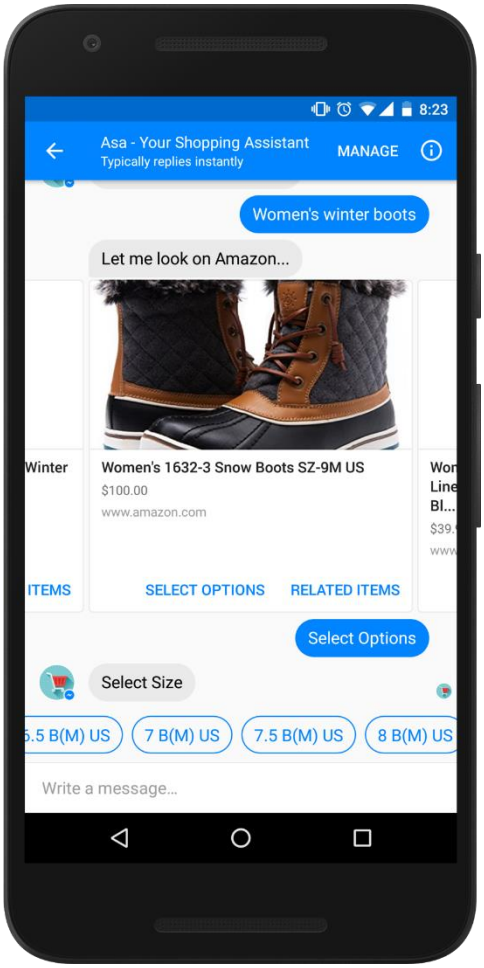

Figure 4: Example of a user requesting a specific item that needs additional options before adding to a cart

If a user would like to delve into different items based on their original search, they have a couple of options: selecting “Related Items” on an item they’re interested in and using the search filters available after every search. Figures 5 and 6 illustrate these scenarios.

In Figure 5, users are presented a collection of new items relevant to the item were “Related Items” was selected. Figure 6 shows the process of delving deeper into the query by specifying different search filters, such as a broad product category or department, a price range, or a more specific filter such as the genre of a book. Users also have the option of selecting a “Personal Filter”, which filters based on the user’s profile and aims to find items tailored to the user’s interests.

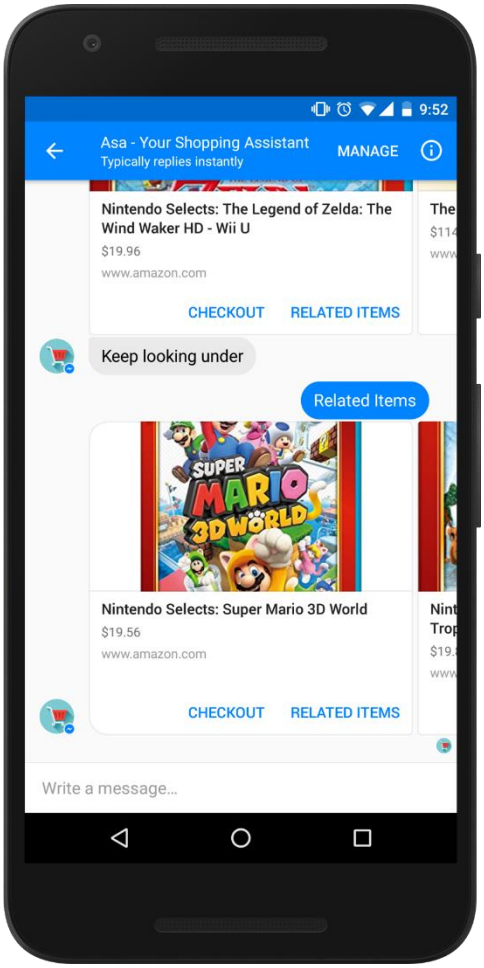

Figure 5: Example of the “Related Items” feature for an item

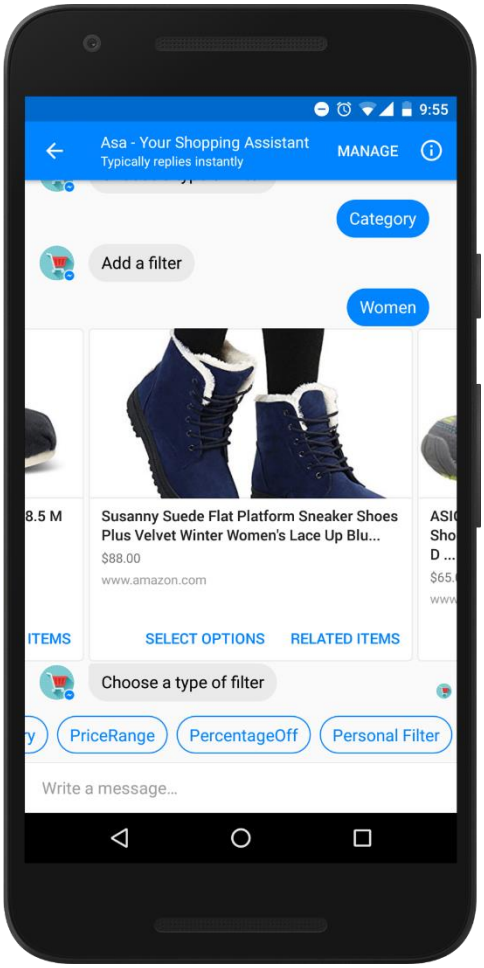

Figure 6: Example of a user using search filters to narrow down their product search

Figure 7 shows what a user sees if they ask Asa for an item recommendation. This interface looks very similar to Figures 3 and 4, but differs in what items are being returned. Asa uses what she knows about a user (from a user profile developed over time using the user's purchases through Asa) to find items tailored to the user's interests. In Figure 7, as the user has purchased Harry Potter related items in the past, several Harry Potter books showed up.

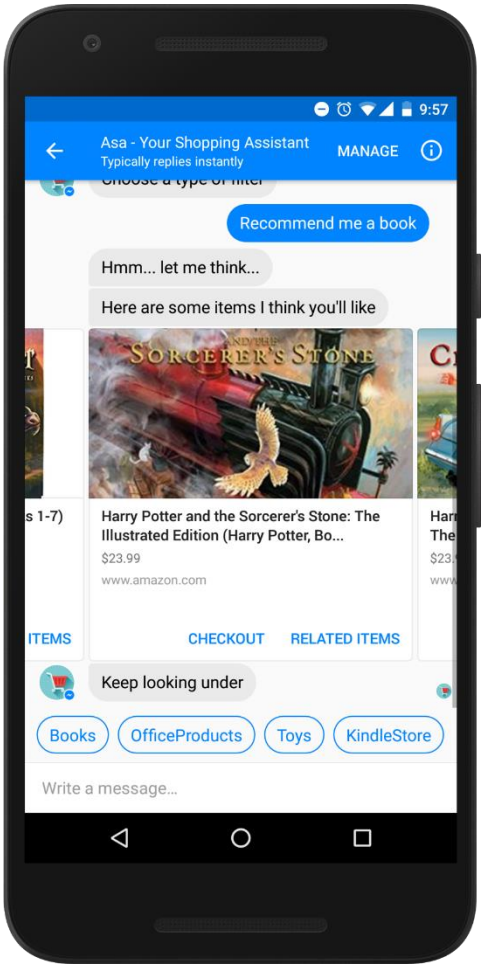

Figure 7: Example of a user asking for an item recommendation

Having a user contact Asa is not the only way a conversation can begin. Asa also re-engages users by periodically sending tailored item suggestions, as shown in Figure 8. She, again, uses the user's profile based on their past purchases with her to create a collection of items specific to that user's interest. The user can scroll through the items and purchase if desired.

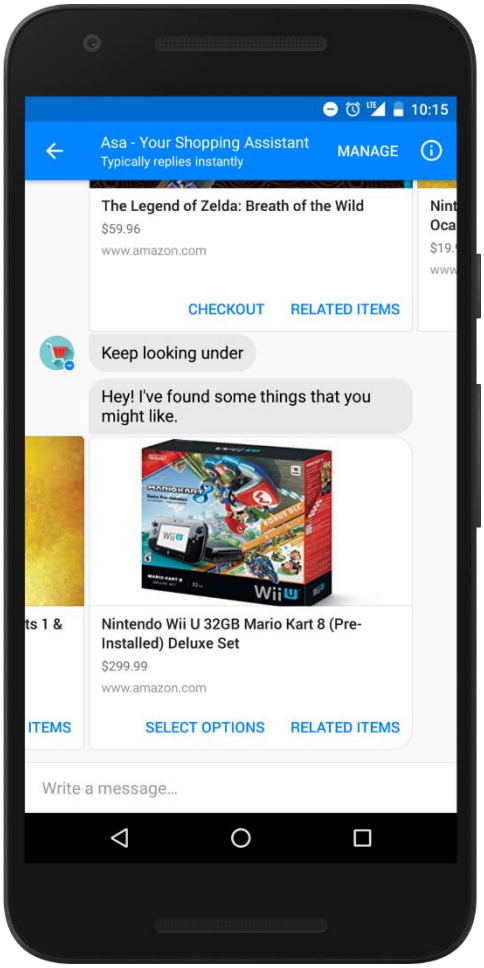

Figure 8: Example of a bot-initiated suggestion

Besides purchasing items, users can also ask Asa to remind them to do something. For example, they can ask her to remind them to buy textbooks a month before their semester starts or to find their mom a gift a couple of weeks before her birthday. Figure 9 is an example of a user requesting a reminder.

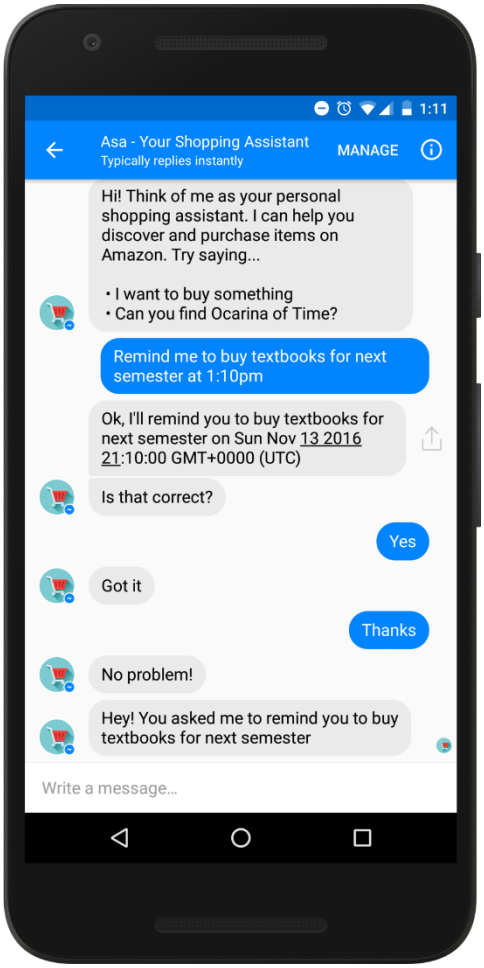

Figure 9: Example of a user requesting a reminder

4.3 Conversational Interface

Alongside the visual interface, there is also hidden aspect of the user interface embedded into the conversations themselves. A great deal of this project's user experience depends on the flow of the conversation, and maintaining a natural flow of conversation is a challenging task. Asa aims to converse with a user in the most cohesive, intuitive and human-like way in order to offer the best user experience possible.

5 Technical Specifications

5.1 Serverless Architecture

Asa's architecture is based on modular serverless microservices deployed to AWS Lambda. AWS Lambda abstracts the need to provision and manage servers, allowing us to focus on our business logic and scale with high availability without worrying about infrastructure.

Endpoints deployed to AWS Lambda will be exposed through AWS API Gateway, which provides security controls and monitoring features.

5.2 Development Tools

The runtime environment for each Lambda function is Node.js v4.3.2. Team members are able to use their favorite JavaScript dev environment as long as they can run the command line tools for testing and deployment.

Our project is hosted on GitHub. We use a workflow based on the Gitflow model for managing and versioning our code base. Automatic builds and deployments to production are kicked off with new pushes to the master branch using Travis CI.

We are using Serverless, an open source tool, to build, deploy, and manage our AWS Lambda projects. We set up Serverless to deploy to different environments (staging, production) based on environment variables in Travis CI. Travis CI is also set up to automate unit tests upon building and to lint to enforce project conventions and code quality.

Besides using Serverless to deploy Asa to development and production stages, the team developed a local test bench for running the Asa endpoints during active development. This increases the speed of development and decreases the cost of maintaining separate environments in the cloud for each developer.

5.3 Test Plan

There are two ways this project is tested: unit tests and end-user testing. Unit tests are implemented for each AWS Lambda function. This is accomplished using Mocha, a JavaScript testing framework, accompanied by Chai, an assertion framework for JavaScript.

End-user testing is done not only by the team members and the Amazon client, but by outside parties as well. The conversations each user has with Asa are logged by Facebook and the NLP client used by Asa to process messages. These conversations are used to both train the NLP app and discover needed improvements to the conversational interface.

5.4 System Architecture

Figure 10 is an illustration of the system architecture. The Asa system architecture was designed with three key considerations in mind:

1. Adding support for a new messaging platform should be as easy as possible
2. Changing the specific NLP platform used to parse messages should be simple
3. Upgrading the User Profiler should not require structural changes to the system

To achieve these goals, the Conversation Manager communicates with any external resources, such as the NLP Platform and User Profiler, through predefined interfaces. The components themselves can then handle specific API details and run custom business logic while the Conversation Manager simply needs to talk to a consistent client interface.

For example, each messaging platform has its own platform client. That client is responsible for changing events sent by the platform into a uniform JSON format to be consumed by the Conversation Manager. In order to send messages back to a user, the platform client supports a set of functions which package and send certain message types to a specific user on that platform. These message types include text, image slideshows, and quick reply buttons. In this way, the Conversation Manager can expect consistent message formats and send messages through a standard set of functions. Adding support for a platform is as easy as adding another platform client.

This modular architecture also increases the ability to thoroughly test edge cases for isolated functions. This is especially important when using AWS Lambda since running tests on integrated components is both complex and costly so increasing the amount of tests for isolated functions as opposed to integrated components decreases the time it takes to run tests while increasing the reliability and scalability of the test suite.

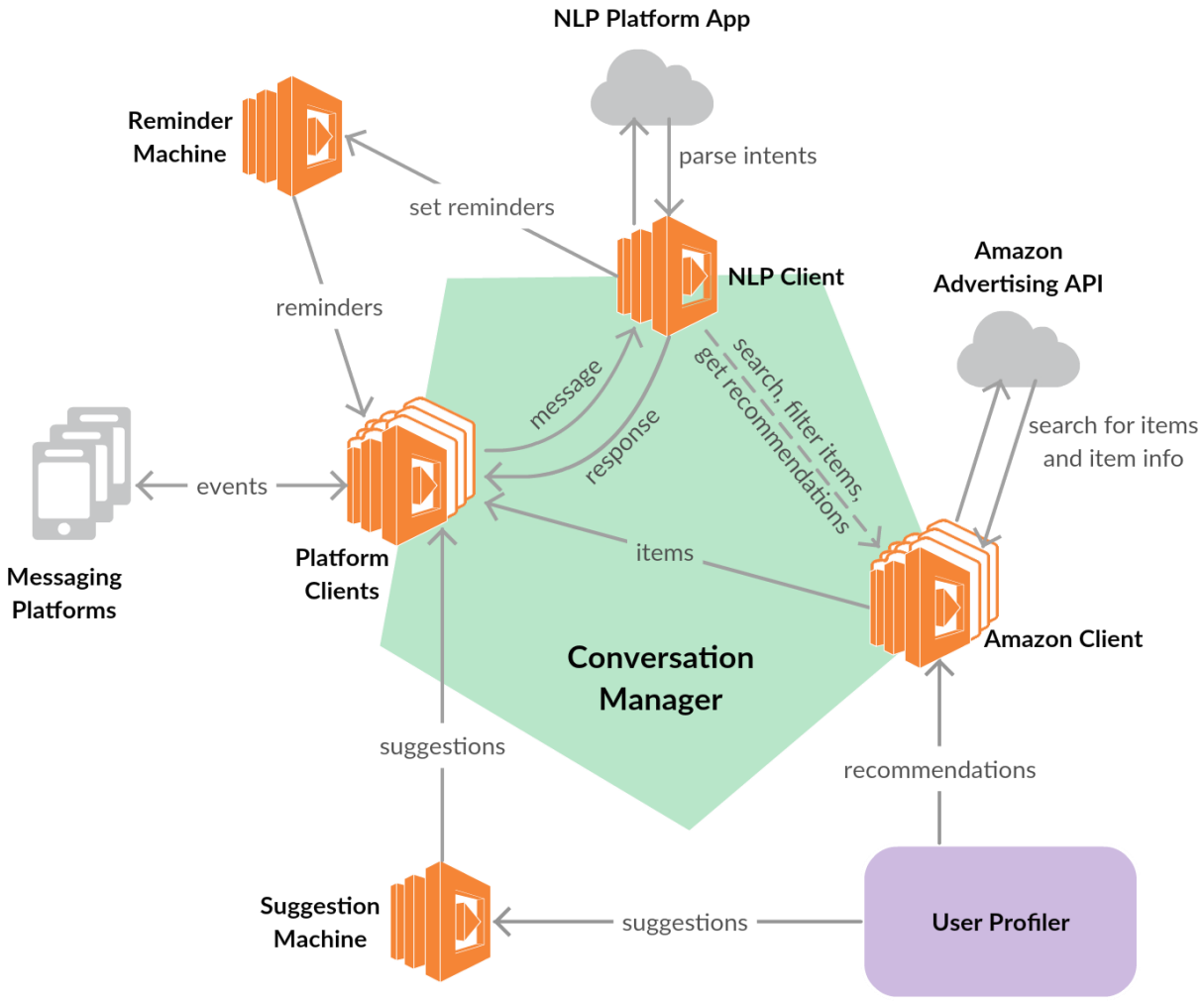

Figure 10: System architecture diagram

5.5 Facebook Messenger API

The Facebook Messenger API allows for an easy way to send messages to people through Facebook messenger, as well as subscribe to events from Facebook (such as a user sending the bot a message). Each message from a user gets sent to the Facebook Messenger platform endpoint deployed as a Lambda function and available as an HTTP endpoint through API Gateway. First, the endpoint uses the Facebook Messenger client to convert the message JSON into the standard format before sending that message to the Conversation Manager for processing. Eventually the other part of the Facebook Messenger client, the message sender, translates a return message back into a platform specific event and sends that message to the user using the Messenger API. Messages vary based on what the NLP platform decides should be sent to the user. Responses range from simple text messages to images and buttons.

Messages sent through the messenger API can follow a couple different templates that allow for images and buttons to be added to the response. When sending item information, Asa primarily uses the generic template, which has an image, title, sub-title, description, and also buttons that the user can click. The messenger API also has the ability to specify quick responses that allow users to send predefined messages back to Asa with the click of a button. This is useful when a user is picking variations on items, because Asa can direct the user to provide valid responses.

5.6 NLP Platform

In order to parse user intent from natural language messages, this project uses an already existing NLP platform called Wit.ai which utilizes machine learning to aid in training an app for natural language processing. The general conversation flow is a back and forth conversation between the user and the NLP platform until the conversation context maintained by the NLP platform contains the correct information needed to fire off one of Asa's functions such as searching for an item or asking for a recommendation.

Asa is developed to be agnostic of the natural language processing platform by expecting calls to the NLP client to return the next response to send or action to take.

5.7 Amazon Client and Amazon Advertising API

The Amazon client consists of individual functions that interface with the Amazon Advertising API. The Advertising API provides access to Amazon's product selection. Through the API, clients can make search requests to get different details on products including: item attributes, images, price, offer, and even product reviews. At its core, Asa provides a natural language interface to the Advertising API, allowing users to accomplish the procedures of product search, selection and check out through conversation.

The NLP platform interfaces with the product advertising API through the Conversation Manager to search for exactly what the user is looking for. The output of the NLP platform, such as the keywords to search for as well as the categories to filter the search, aids the product advertising API in retrieving the items a user is looking for.

The API also sends back variations of the item if they exist. Variations for a t-shirt might include multiple sizes (S, M, L, XL, etc.) and colors (Green, White, etc.). This allows Asa to prompt the user as needed in order to pinpoint the item they want.

5.8 The User Profiler

The User Profiler has two main jobs:

1. Provide a custom item recommendation based on a query from the user
2. Suggest an item for a particular user when asked

There are many ways to accomplish these two tasks and the primary goal of this project is to provide an architecture that could support a much more complex solution for the User Profiler than can be reasonably developed in 15 weeks. The Conversation Manager passes in all important user information such as item traffic to ensure any future solutions will be able to use such information when building models for recommendation and suggestion.

With that being said, this project does implement a solution for the two tasks mentioned above. When suggesting items to a user, the implemented User Profiler uses the Similarity Lookup function in the Amazon Advertising API. This is the function responsible for displaying the items at the bottom of a product page on Amazon that other users purchased. Since the User Profiler tracks purchase events for each user, it can pass that information into the Similarity Lookup function to find items others users with the same purchase history purchased.

Recommending items to a user within a category they provide to Asa is more difficult, especially since this project does not have the wealth of data needed to build a model with machine learning. Our solution uses a TF-IDF strategy to match a user profile with the most similar items returned from a large search query. Essentially what this does is provide a way for each user to filter search results with a custom filter built for them from their past purchases.

5.9 The Suggestion Machine

One of the key features of Asa is the ability to suggest items or gift ideas to the user based on learned preferences or events like birthdays. The Suggestion Machine is responsible for making suggestions to users by asking the User Profiler for items to suggest on a regular interval or at important times like birthdays.

For this project, a timed lambda function will run daily in order to send suggestions to users on a monthly basis.

5.10 The Reminder Machine

Similar to the Suggestion Machine, the Reminder Machine runs on a schedule and reads entries from a database in order to send bot-initiated messages to a user. In this case, the message is a reminder the user has set through conversation such as “buy a gift for my mom” and an associated time they want to be reminded. The Reminder Machine is responsible for sending this reminder message at the specified time.

5.11 Database

The NLP client, User Profiler, Suggestion Machine, and Reminder Machine need persistent storage in order to store conversation contexts, user profiles, suggestion times, and reminders respectively. AWS has several database services and we chose to use DynamoDB for all persisted storage. We chose the NoSQL document based DynamoDB over a more restrictive relational database due to the dynamic schema needed to represent conversation contexts and user profiles.

6 Risk Analysis

Throughout the development of this project, several risks will need to be addressed and mitigated accordingly. These risks and their corresponding mitigation plans are discussed below in the order of most urgent to least urgent.

Amazon Web Services (Lambda, API Gateway) and Bot Platforms

Difficulty: Easy

Description: Most of the developers on this project have not previously used Amazon Web Services or developed a chatbot

Mitigation: Research AWS and create prototype chatbots for Facebook, Slack, and Twilio deployed on Amazon Web Services

Extensive Use of Natural Language Processing (NLP)

Difficulty: Medium

Description: Retrieving the correct intent from natural language messages which are infinitely diverse is a challenging but vital task when building chatbots

Mitigation: Research and learn how to use NLP platforms like Wit.ai and Api.ai

Retrieving Required Additional Information for Items

Difficulty: Medium

Description: When a user wants to purchase an item that has variations such as size, color or quantity, we must devise a strategy to retrieve this information

Mitigation: Learn how to find all variations of an item using the Advertising API then utilize the "quick replies" feature in Facebook Messenger to prompt the user to choose from the available options

Learning About a User

Difficulty: Hard

Description: Without access to Amazon customer data, we lack an easy way to build a customer profile for accurate product recommendations

Mitigation: Depend on Amazon's Advertising API initially and develop a modular architecture that will allow for future enhancements to be easily "plugged in." Try to devise a way to create recommendations based on a user's past purchases and even look for publicly available datasets that could be used to build customer models

Making Suggestions

Difficulty: Medium

Description: We need a way to know when to send suggestions and how to choose the items being suggested

Mitigation: Send suggestions on a simple timer and use the Similarity Lookup function in the Amazon Advertising API

7 Schedule

- **Product Stages:**
  - **P0:** Search for specific item, specify variations, purchase item
  - **P1:** Filter searches by category through conversation
  - **P2:** Ask for recommendations from Asa
  - **P3:** Bot-initiated product suggestions
  - **P4:** (Stretch goal) Set and send purchase reminders

- **Week 1: 8/31 – 9/2**
  - Team Assignment
  - Initial Team Meeting
  - Workstation Setup & Software Installation
  - Created rudimentary Slack, Facebook, and SMS chat bots

- **Week 2: 9/5 – 9/9**
  - Initial Conference Call with Amazon Client
  - Familiarization with Natural Language Processing services
  - Used NLP Services to create applications that recognize search requests
  - Familiarization with Amazon Advertising API
  - Created simple Facebook Messenger chat bot
  - Started developing project plan

- **Week 3: 9/12 – 9/16**
  - 9/14 – Status Report Presentation
  - Design system architecture
  - Complete Project Plan
  - Integrate NLP services with Facebook Messenger Bot
  - Present Functional Prototype to Amazon Client

- **Week 4: 9/19 – 9/23**
  - Add ability to collect specific item variations to NLP app
  - Setup project structure and implement prototype with correct architecture

- **Week 5: 9/26 – 9/30**
  - Explore Facebook API and possible implementations of the User Profiler
  - Integrate product variation selection in chatbot
  - Complete P0

- **Week 6: 10/3 – 10/7**
    - 10/3 – Project Plan Presentation
    - Plan out prototypes for recommending items and making suggestions
    - Research how to filter search results through the Product API
    - Research how to run lambda functions daily
    - Start working on Alpha presentation

- **Week 7: 10/10 – 10/14**
    - Create prototype for recommending items based on a query
    - Create prototype for item suggestions
    - Finish Alpha presentation and practice presenting

- **Week 8: 10/17 – 10/21**
    - Implement search filtering through conversation
    - Begin initial end-user testing
    - Create User Profiler from prototype scripts for recommendations and suggestions

- **Week 9: 10/24 – 10/28**
    - 10/24 – Alpha Presentation
    - Implement recommendations
    - Complete P1

- **Week 10: 10/31 – 11/4**
    - Implementing bot initiated suggestions
    - Start working on Beta Presentation
    - Complete P2

- **Week 11: 11/7 – 11/11**
    - Train Wit app to collect information needed to set purchase reminders
    - Implement purchase reminders
    - Finish Beta presentation and practice presenting
    - Complete P3

- **Week 12: 11/14 – 11/18**
    - Bug fixes and product refinement
    - Start working on project video and design day presentation
    - Complete P4

- **Week 13: 11/21 – 11/25**
    - 11/21 – Beta Presentation
    - Polish current product
    - Film project video

- **Week 14: 11/28 – 12/2**
    - Visit Amazon Detroit and present product
    - Finish project video and keep practicing design day presentation
    - Extend simple functionalities of Asa to another platform (if time allows)
    - Finalize product for delivery

- **Week 15: 12/5 – 12/9**
    - 12/5 – Project Videos
    - 12/9 – Design Day