FlavorFeedback

by
Paul Long

This thesis has been submitted in partial fulfillment for the degree of Bachelor of Science (Hons) in Software Development

April 2024

Declaration of Authorship

I, Paul Long, declare that this thesis titled, 'Customer Feedback Chatbot' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an undergraduate degree at Munster Technological University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Munster Technological University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Paul Long
Date: 26/04/2024

Abstract

An advantage of voice assistants as interaction medium is the minimisation of intention to action. Interaction through voice is direct (compared to smart phones or computers).

A use-case that can benefit from this is getting customer feedback on products or services, for example on the taste of a ready meal, the quality of a movie etc.

The objective of this project is to research, develop & release a voice conversational agent (chatbot) that will accept customer feedback. This will consist of:

a) An Alexa skill which will identify the product and receive customer feedback
b) The supporting back-end server and database to manage this feedback
c) A simple admin interface to manager products and view feedback

Acknowledgments

First and foremost, I would like to thank Dr. Alex Vakaloudis. Alex proposed the project in August of 2023 and has since provided invaluable guidance and advice throughout this research process, pointing me in the right direction and highlighting resources which have allowed me to effectively conduct research and plan this project.

I would also like to thank Dr. Alison O'Shea and Dr. Brian Murphy for their direction in the design of this report and their material regarding the efftice use of the tools needed to write this paper.

Lastly, I would like to thank all those who've helped and supported me over the past few months.

Contents

1 Introduction

1.1 Background and Context

The food and beverage (F&B) industry is highly competitive and dynamic, with customer satisfaction playing a pivotal role in the success of businesses within this sector. In recent years, the integration of chatbot technology has gained prominence as a means to improve customer service and gather valuable feedback. This research delves into the implementation of an Alexa chatbot in the F&B industry to enhance customer feedback collection and engagement.

1.2 Problem Statement

The F&B industry faces challenges in efficiently collecting and processing customer feedback, which is vital for understanding customer preferences and enhancing overall dining experiences. Traditional methods for collecting feedback, such as paper surveys or online forms, often result in low response rates and are cumbersome for both customers and businesses. This study seeks to address this problem by designing and implementing an Alexa chatbot to streamline feedback collection and provide a seamless interaction for customers.

1.3 Objectives of the Study

The primary objectives of this project are as follows:

- To design and develop an Alexa chatbot for the F&B industry that enhances customer feedback collection.
- To evaluate the effectiveness and usability of the chatbot in improving customer engagement.
- To assess the impact of the chatbot on the quality of customer feedback and its relevance to business decisions.

1.4 Scope and Limitations

This research focuses on the design and implementation of an Alexa chatbot for the F&B industry. It will not explore the development of chatbots on other platforms or industries. The study is limited to a specific geographical region and may not account for cultural variations in customer feedback expectations.

1.5 Significance of the Research

The research holds significance as it addresses a pressing issue in the F&B industry by offering an innovative solution to streamline feedback collection. The successful implementation of an Alexa chatbot can lead to enhanced customer engagement and more valuable feedback, ultimately benefiting F&B establishments.

1.6 Thesis Structure

This thesis is structured as follows: after this introduction, a literature review will provide an overview of relevant studies in the F&B industry, chatbot technology, and existing chatbot applications in customer feedback. The subsequent sections will focus on the design of the Alexa chatbot, including architecture, Natural Language Processing (NLP), data security, and user training. The conclusion will summarize findings and suggest future directions.

2 Literature Review

2.1 Market Research in the Food and Beverage Industry

2.1.1 Customer Feedback Collection Methods

Surveys:
Surveys have been a traditional method for collecting customer feedback. They can be administered in various formats, including paper surveys, email surveys, and online surveys. Surveys allow businesses to gather structured data on customer preferences, satisfaction levels, and specific aspects of their experience.

Online Reviews:
Online review platforms, such as Yelp, TripAdvisor, and Google Reviews, are essential sources of customer feedback for F&B businesses. Customers can post detailed reviews and ratings, which are visible to a broad audience. Monitoring and responding to online reviews is critical for reputation management.

Social Media:
Customers frequently share their dining experiences on social media platforms like Facebook, Twitter, and Instagram. Businesses can track and respond to posts and comments related to their establishment, which can provide valuable insights and promote positive engagement.

Email Feedback Requests:
F&B businesses often send email requests for feedback to customers after their dining experience. These emails may contain links to online surveys or encourage customers to share their thoughts directly via email.

Mystery Shoppers:
Some F&B businesses employ mystery shoppers who visit the establishment anonymously and evaluate various aspects of the experience. Mystery shoppers provide detailed feedback from the perspective of an actual customer.

Feedback Chatbots:
Emerging technologies like chatbots are used to collect customer feedback. Chatbots can engage with customers through messaging apps or websites, asking structured questions or engaging in open conversations to gather feedback.

A range of methods are used to collect customer feedback in the hospitality industry, including surveys, online reviews, social media, email requests, mystery shoppers, and feedback chatbots
- Hospitality Feedback System 4.0: Digitalization of Feedback System with Integration of Industry 4.0 Enabling Technologies Ram Narayan

These methods have their own advantages and challenges, with the use of robots for real-time feedback collection being a particularly innovative approach
- "How was Your Stay?": Exploring the Use of Robots for Gathering Customer Feedback in the Hospitality Industry M. Chung, M. Cakmak

The digitalization of feedback systems, incorporating technologies such as the Internet of Things, artificial intelligence, cloud computing, and big data, is also highlighted as a key trend
- Hospitality Feedback System 4.0: Digitalization of Feedback System with Integration of Industry 4.0 Enabling Technologies Ram Narayan

However, there is a need for further research to compare the effectiveness of these methods and to identify best practices for customer feedback collection in the hospitality industry
- Comparing practices for capturing bank customer feedback - Internet versus traditional banking J. Wisner, William J. Corney

2.1.2 Importance of Customer Feedback in the Food and Beverage Industry

Quality Improvement:
Customer feedback serves as a valuable source of insights into the strengths and weaknesses of an F&B establishment. It helps identify areas in need of improvement, whether related to food quality, service, cleanliness, or ambiance. Businesses can make data-driven decisions to enhance the overall dining experience.

Issue Resolution:
Customer feedback provides a direct channel for customers to express concerns and dissatisfaction. Promptly addressing these issues demonstrates a commitment to customer service and can turn negative experiences into positive ones. Resolving problems can also prevent negative online reviews and word-of-mouth complaints.

Innovation:
Feedback often reveals unmet customer needs or opportunities for innovation. New menu items, promotions, and service enhancements can be developed based on customer suggestions, fostering creativity and continuous improvement within the industry.

Customer Retention:
Meeting customer expectations and addressing their concerns leads to increased customer loyalty. Satisfied customers are more likely to return for future dining experiences, become repeat patrons, and recommend the establishment to others.

Positive Reputation:
Positive feedback and customer testimonials, whether shared online or through word of mouth, contribute to a strong and positive reputation. A good reputation can drive more customers to an F&B establishment and enhance its brand image.

Financial Impact:
Ultimately, the financial impact of customer feedback is substantial. Satisfied customers are more likely to spend more, order additional items, and return for repeat business. By focusing on customer satisfaction, F&B businesses can boost revenue and profitability.

2.1.3 Competitor Analysis in the Chatbot Market

Starbucks - My Starbucks Barista:
Strengths:
Convenience: My Starbucks Barista allows customers to place orders, customize drinks, and make payments via the Starbucks mobile app. Personalization: The chatbot remembers previous orders and allows for a highly personalized ordering experience. Integration: Seamlessly integrates with the Starbucks Rewards loyalty program, enhancing customer retention.

Weaknesses:
Limited Functionality: Primarily focused on order placement and customization, may not handle com-

plex customer queries or provide extensive customer feedback capabilities.

Burger King - BK Bot:
Strengths:
Order Placement: BK Bot simplifies the order process, allowing customers to customize meals and place orders through Facebook Messenger. Convenience: Offers hands-free and voice-activated ordering for an enhanced user experience. Efficiency: Enhances order accuracy and streamlines the ordering process.

Weaknesses:
Limited Scope: BK Bot is primarily designed for order placement and customization and may not offer extensive customer feedback or support functions. Limited Language Support: May not cater to customers who prefer to communicate in languages other than those supported by the chatbot.

Domino's Pizza - Dom:
Strengths:
Convenience: Customers can place orders, track deliveries, and provide feedback through various messaging platforms like Facebook Messenger. Feedback Collection: Actively collects feedback after each order, providing valuable insights for improvements. Versatility: Offers a range of features, from order placement to feedback collection.

Weaknesses:
Limited Scope: While versatile, it may not offer the same level of personalization or menu recommendations as AI-powered chatbots. May Require Scripted Interactions: Since it's rule-based, it may not adapt to unscripted or complex user queries.

2.1.4 Emerging Trends in Customer Feedback Collection

Chatbot Feedback Collection:
Chatbots are increasingly being used to collect feedback. They engage with customers through messaging apps or websites, making it convenient for users to share their thoughts and experiences.

Voice Feedback:
Voice-activated feedback collection, where customers can provide feedback through voice commands or phone surveys, is gaining traction. Voice recognition technology allows for hands-free feedback, which is particularly useful in drive-thru and delivery services.

In-App Feedback:
Mobile apps developed by F&B establishments often include in-app feedback forms that allow customers to provide input directly from their smartphones. These forms can be seamlessly integrated into the app for a smooth user experience.

Social Media Listening:
F&B businesses are increasingly using social media listening tools to monitor and analyze mentions and conversations about their brand on platforms like Twitter, Facebook, and Instagram. This helps them identify trends, concerns, and positive feedback in real time.

Customer feedback is a crucial tool for F&B establishments, providing valuable insights into their strengths and weaknesses - *THE IMPACT OF SERVICE QUALITY ON CUSTOMER SATISFACTION S. Gilaninia, M. Taleghani, Mohammad Reza Khorshidi Talemi*

Prompt issue resolution can turn negative experiences into positive ones, preventing negative reviews and enhancing the establishment's reputation - *THE IMPACT OF SERVICE QUALITY ON CUSTOMER SATISFACTION S. Gilaninia, M. Taleghani, Mohammad Reza Khorshidi Talemi*

Customer feedback can also drive innovation, leading to the development of new menu items and service enhancements - *Measuring Customer Satisfaction for F&B Chains in Pune Using ACSI Model*

Meeting customer expectations and addressing their concerns can increase customer loyalty and retention (Oh, 1999). Finally, customer satisfaction, influenced by various factors including food quality, hygiene, and responsiveness, has a significant impact on customer loyalty and financial performance - *Restaurant Quality and Customer Satisfaction Bader M. A. Almohaimmeed*

2.2 Chatbot Technology

2.2.1 Types of Chatbots (Finite vs AI)

Finite Chatbots:

1. Finite chatbots, also known as rule-based or scripted chatbots, operate based on predefined rules and decision trees.
2. They follow a fixed set of instructions and can handle only specific tasks or questions for which they have been programmed.
3. Finite chatbots are suitable for simple, routine tasks, such as providing basic information, answering frequently asked questions, or guiding users through a set process.
4. They are less flexible and adaptive compared to AI-powered chatbots, as they cannot understand or respond to user input that deviates from their predefined rules.

- *Evaluation of the Naturalness of Chatbot Applications A. Atiyah, S. Jusoh, Firas Alghanim*

- *Challenges of Building an Intelligent Chatbot A. Chizhik, Yulia Zherebtsova*

AI-Powered Chatbots:

1. AI-powered chatbots, also called smart chatbots or conversational AI, employ artificial intelligence and machine learning techniques, including natural language processing (NLP), to understand and respond to user input.
2. These chatbots can handle a wide range of user queries, even those that they haven't been explicitly programmed for, thanks to their ability to learn and adapt over time.
3. AI chatbots can engage in more natural and dynamic conversations, making them suitable for complex tasks, such as providing personalized recommendations, handling customer support inquiries, or assisting with natural language understanding tasks.
4. They require training data and continuous learning to improve their performance.

Atiyah (2019) and Chizhik (2020) both highlight the potential of chatbots to mimic human interaction, with Atiyah's study finding that chatbots can perform close to human personnel in certain interactions. However, Chizhik also underscores the challenges in building intelligent chatbots, particularly in achieving coherent and engaging responses. These challenges are further explored in Caldarini's (2021, 2022) literature surveys, which emphasize the need for advancements in natural language processing and machine learning to address the limitations of current chatbot applications.

- *Evaluation of the naturalness of chatbot applications A Atiyah, S Jusoh, F Alghanim*

- *Challenges of Building an Intelligent Chatbot. A Chizhik, Y Zherebtsova*

- *Building a Chatbot: Architecture Models and Text Vectorization Methods AV Chizhik, YA Zherebtsova*

- *A literature survey of recent advances in chatbots G Caldarini, S Jaf, K McGarry*

2.2.2 Natural Language Processing (NLP) in Chatbots

NLP is a critical component of chatbots that enables them to understand and respond to user queries in a human-like manner. Key NLP functions in chatbots include:

1. Text Parsing: Breaking down user input into words, phrases, and entities for analysis.
2. Entity Recognition: Identifying and extracting specific pieces of information, such as dates, locations, and names, from user messages.
3. Sentiment Analysis: Determining the emotional tone of user messages to provide appropriate responses.
4. Language Understanding: Deciphering the meaning and intent behind user queries.
5. Context Management: Keeping track of the conversation context to maintain coherent and relevant interactions.

NLP plays a vital role in enhancing user experiences by allowing chatbots to interpret user input accurately and generate contextually relevant responses.
NLP is a crucial component of chatbots, enabling them to understand and respond to user queries in a human-like manner - *Emerging Technologies of Natural Language-Enabled Chatbots: A Review and Trend Forecast Using Intelligent Ontology Extraction and Patent Analytics M.-H. Chao, A. Trappey, Chunwang Wu*

Key NLP functions in chatbots include text parsing, entity recognition, sentiment analysis, language understanding, and context management - *Emerging Technologies of Natural Language-Enabled Chatbots: A Review and Trend Forecast Using Intelligent Ontology Extraction and Patent Analytics M.-H. Chao, A. Trappey, Chunwang Wu*

These functions are essential for accurate interpretation of user input and generation of contextually relevant responses - *Emerging Technologies of Natural Language-Enabled Chatbots: A Review and Trend Forecast Using Intelligent Ontology Extraction and Patent Analytics M.-H. Chao, A. Trappey, Chunwang Wu*

NLP is a highly interdisciplinary field, involving concepts in computer science, linguistics, logic, and psychology - *Natural Language Processing A. Joshi*

It plays a special role in computer science, as it deals with linguistic features of computation and seeks to model language computationally - *Natural Language Processing A. Joshi*

NLP techniques, such as text embedding using tools like TF-IDF vectorization and bag of words, are crucial for chatbot applications - *NLP for Chatbot Application S. Nithyanandam, Sharmila Kasinathan, Devi Radhakrishnan*

The future of NLP in chatbots involves enabling human-computer interactions in natural languages like English - *Natural Language Processing V. Daniel Hunt*

2.2.3 Voice vs. Text-Based Chatbots

Voice-Based Chatbots: Advantages:

- Provide a hands-free and eyes-free interface, suitable for multitasking.
- Enable a more natural and human-like interaction, as users can speak conversationally.
- Ideal for applications where users' visual attention is limited, such as while driving.

Disadvantages:

- Voice recognition can be less accurate, leading to misunderstandings or misinterpretations.
- Not suitable for environments with high noise levels or when privacy is a concern.
- Limited to languages and accents recognized by the voice recognition technology.

Text-Based Chatbots: Advantages:

- Generally have higher accuracy in understanding and generating text-based responses.
- Suitable for both public and private interactions, making them versatile for various applications.
- Work across different platforms and devices without the need for specialized hardware.

Disadvantages:

- May require users' visual attention and manual input, making them less convenient in certain situations.
- Text-based interactions lack the naturalness of spoken conversation.

Quintero (2015) and Burri (2018) both highlight the potential of voice-based chatbots, with Quintero focusing on their efficiency and accuracy, and Burri on their ability to increase user trust. However, they also acknowledge the challenges, such as the need for high-quality text-to-speech synthesis. In comparison, text-based chatbots are noted for their versatility and higher accuracy (Caldarini, 2021), but they lack the naturalness of spoken conversation (Quintero, 2015). Despite these differences, both types of chatbots have their own advantages and limitations, and further research is needed to address these issues.
- Towards an efficient voice-based chatbot J. Quintero, R. Asprilla

- Improving user trust towards conversational chatbot interfaces with voice output Ramón Burri

- A Literature Survey of Recent Advances in Chatbots Guendalina Caldarini, Sardar F. Jaf, K. McGarry

- Improving user trust towards conversational chatbot interfaces with voice output R Burri

- A literature survey of recent advances in chatbots G Caldarini, S Jaf, K McGarry

2.2.4 Concepts (Amazon/Google terms and Dialogue Flow)

Amazon and Google have their own terms and concepts related to chatbot technology:
Amazon Lex (Amazon):
Amazon Lex is a service for building conversational interfaces, including chatbots. It incorporates automatic speech recognition (ASR) for voice interaction and integrates with Amazon Polly for text-to-speech conversion. Lex enables developers to create chatbots for various platforms and devices.

Google Dialogflow (Google):
Google Dialogflow, formerly known as API.ai, is a natural language understanding platform that allows developers to build text-based and voice-based conversational interfaces. It offers pre-built templates, integrations with various platforms like Google Assistant and Facebook Messenger, and supports multiple languages and platforms.

- Conversational Agent Research Toolkit: An alternative for creating and managing chatbots for experimental research T. Araujo

- Introduction to Microsoft Bot, RASA, and Google Dialogflow Abhishek Singh, Karthik Ramasubramanian, Shrey Shivam

- Review of State-of-the-Art Design Techniques for Chatbots Ritu Agarwal, Mani Wadhwa

- Survey on Chatbot Design Techniques in Speech Conversation Systems S. Abdul-Kader, John Woods

Dialogue Flow Design:
Dialogue flow design refers to the process of defining the conversation structure and logic for a chatbot. It involves creating intents, entities, and responses, specifying how the chatbot should handle user inputs, and designing a conversational flow that makes interactions with the bot feel intuitive and user-friendly. Both Amazon Lex and Google Dialogflow provide tools for designing and managing dialogue flows.

A range of studies have explored the design and capabilities of chatbots, with a focus on Amazon Lex and Google Dialogflow. Abdul-Kader (2015) and Agarwal (2020) both discuss the design techniques for chatbots, with Agarwal highlighting the use of rule-based and neural network-based approaches. Singh (2019) introduces Microsoft Bot, RASA, and Google Dialogflow as readily available frameworks for building chatbots. Araujo (2019) presents the Conversational Agent Research Toolkit (CART) as a tool for creating and managing chatbots for experimental research. These studies collectively underscore the importance of understanding the various design techniques and frameworks available for building chatbots.
- Survey of various AI chatbots based on technology used S Singh, HK Thakur

- Conversational agent research toolkit: an alternative for creating and managing chatbots for experimental research T Araujo

2.3 Existing Chatbot Applications in Customer Feedback

2.3.1 Case Studies and Examples

Domino's Pizza - Dom:
Domino's Pizza launched a chatbot called Dom, which allows customers to place orders, track deliveries, and provide feedback via various messaging platforms like Facebook Messenger and Slack. The chatbot simplifies the ordering process and collects customer feedback after each order. As a result, Domino's has seen improved customer satisfaction, increased order efficiency, and a more engaging customer experience.

Starbucks - My Starbucks Barista:
Starbucks introduced the My Starbucks Barista chatbot through its mobile app. Customers can place orders and customize their drinks using natural language. This chatbot also remembers previous orders and personal preferences, creating a personalized and efficient ordering experience for users. It has contributed to increased mobile orders and a boost in customer engagement.

Burger King - BK Bot:
Burger King introduced the BK Bot, which allows customers to place orders and customize their meals via Facebook Messenger and other messaging platforms. The chatbot collects feedback and orders, making the ordering process more efficient. Customers have reported better order accuracy and overall satisfaction with the service.

2.3.2 Key Success Factors

User-Friendly Design:
The chatbot should have an intuitive and user-friendly design, making it easy for customers to provide feedback. Simple and clear prompts, buttons, and conversational interfaces enhance the user experience.

Prompt and Timely Engagement:
Chatbots should proactively seek feedback from customers after a dining experience, online order, or delivery. Timely engagement ensures that customers' experiences are fresh in their minds, leading to more accurate and detailed feedback.

Clear Purpose and Expectations:
The chatbot should set clear expectations regarding the purpose of collecting feedback. Customers should understand that their input is valuable for improving the dining experience, and their feedback will be used for that purpose.

Open-Ended and Structured Questions:
The chatbot should combine open-ended questions to allow customers to provide detailed feedback and structured questions for specific aspects of the experience (e.g., food quality, service, ambiance).

Option for Anonymity:
Some customers may prefer to provide feedback anonymously, so the chatbot should allow for this option to encourage candid responses.

Lokman (2018) and Xu (2017) both emphasize the importance of architectural design and implementation in creating effective chatbot systems. Kim (2019) and Gregori (2017) highlight the potential of chatbots in improving survey response quality and reducing response time, respectively. These studies collectively underscore the need for a user-friendly design, prompt engagement, clear purpose, and structured questions in chatbot systems.
- Modern Chatbot Systems: A Technical Review Abbas Saliimi Lokman, Mohamed Ariff Ameedeen

- Comparing Data from Chatbot and Web Surveys: Effects of Platform and Conversational Style on Survey Response Quality Soomin Kim, Joonhwan Lee, G. Gweon

- Evaluation of Modern Tools for an OMSCS Advisor Chatbot Eric Gregori

- The media inequality: Comparing the initial human-human and human-AI social interactions Y Mou, K Xu

2.4 Gaps in the Current Literature

Small-Scale Case Studies:
Some research is based on small-scale case studies of individual F&B businesses. Broader studies that encompass a diverse range of establishments, from small restaurants to large chains, are necessary to provide a more comprehensive understanding of the challenges and benefits of feedback chatbots.

Customer Demographics and Behavior:
Existing research often lacks detailed exploration of how different customer demographics and behaviors influence the effectiveness of feedback chatbots. Understanding the preferences and attitudes of various customer groups is essential for designing chatbots that cater to diverse customer bases.

Impact on Loyalty and Retention:
While some studies examine the impact of feedback chatbots on customer satisfaction, there is a limited focus on their role in enhancing customer loyalty and retention. More research is needed to assess the long-term effects of chatbot-enabled improvements on customer loyalty.

Comparative Studies:
Comparative studies that evaluate different feedback collection methods, including chatbots, surveys, and in-person feedback, are limited. Such studies can help F&B businesses choose the most effective methods based on their specific needs.

3 Design of the Alexa Chatbot

3.1 Objectives

Primary
The primary objective of this work is to design and implement an Alexa skill that serves as an intuitive and efficient platform for users to provide feedback on household food and beverage items.
Secondary

- Facilitate increased user engagement in providing feedback.
- Establish a secure and privacy-conscious environment for user data.
- Contribute valuable insights to businesses for product refinement.

3.2 Weekly Plan

Week	Plan	Actual
Week 2	Clearly define the scope of the project, types of products to be featured, requirements of front-end, back-end and general functionality of the chatbot	Defined scope of project
Week 3	”	Created and tested basic skill in alexa developer console
Week 4	Create a general flow for the chatbot, detail back-end and database interactions -Version 1	Experimented with ngrok and locally hosted alexa skills testing through alexa developer console
Week 5	Design database schema -version 1 -version 2 November	Basic flow of chatbot and database layout
Week 6	Consolidate relevant research and prep for draft essay submission	Compiled research and literature for draft essay
Week 7	Draft Essay Due	Completed draft essay
Week 8	In-depth design of functionality of front-end including wireframes. Detail how the front-end is to communicate with the back-end	Further developed flow of chatbot and admin back end access
Week 9	Decide how to utilize ASK to allow java back-end to communicate with Alexa -Version 1 -Version 2 December	Tested AWS hosting service potential
Week 10	Outline what training is needed for the chatbot, phrases common and obscure phrases which may be used to interact with the chatbot. Decide on NLP (spaCy/NLTK) library or Amazon Comprehendr	Outlined user stories and planned an agile approach to development
Week 11	Work on thesis -intro -related work -design	Investigated back-end and database hosting options and worked on final thesis
Week 12	”	Worked on thesis

3.3 Architecture and Components

3.3.1 Voice Recognition and Response Mechanisms

Voice Recognition:

Automatic Speech Recognition (ASR):
ASR technology is used to convert spoken language into text. It processes audio input and transcribes it into a format that the chatbot can understand.

Keyword Spotting:
Some voice recognition systems use keyword spotting to identify specific keywords or phrases that trigger chatbot responses. This can be useful for handling voice commands or specific requests.

Language Models:
Voice recognition systems leverage language models that understand context and semantics. These models help in interpreting user intent and context from spoken words.

Voice Response:

Text-to-Speech (TTS):

TTS technology is used to convert text-based chatbot responses into speech. The chatbot generates text responses and then uses TTS to vocalize these responses to the user.

Voice Persona:
Some chatbots are designed with a specific voice persona or character to create a more engaging and human-like interaction. The tone and style of speech can be tailored to match the brand's voice.

Natural Language Generation (NLG):
NLG techniques are employed to ensure that voice responses sound natural and contextually appropriate. This enhances the conversational flow and user experience.

Response Variability:
To make interactions more engaging, chatbots can be programmed with variations in their responses to avoid sounding overly repetitive. This can include using different phrases to convey the same information.

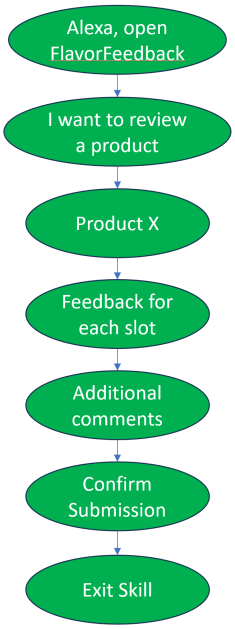

Fig 3.1 B̃asic Feedback Flow

User Invocation:

User Interaction:

- The user activates the skill by saying, "Alexa, open [Skill Name]."

Java Processing:

- The Alexa service triggers the skill, and the Java-based Lambda function is invoked.
- The Lambda function initializes the skill's logic and begins processing the user's request.

Welcome Message:

User Interaction:

- Alexa responds with a welcome message.

Java Processing:

- The Java code within the Lambda function handles the welcome message generation.
- It uses the ASK Java SDK to construct and send the appropriate response back to the user.

Review Product:

User Interaction:

- The user expresses the intent to review a product.
- Example: "I want to review a product."

Java Processing:

- The ASK Java SDK captures the user's request and identifies the intent to review a product.
- The skill's Java code processes the intent and prepares for product review interaction.

Item Selection:

User Interaction:

- The user provides the name of the product they want to review.
- Example: "I want to review the [Product Name]."

Java Processing:

- The Java code processes the user's product selection and checks the database (Firestore) for details about the specified product.
- It retrieves relevant information about the product.

Product Information Response:

- The skill responds with details about the selected product, setting the stage for the review.
- Example: "Excellent choice! The [Product Name] is known for [specific features]. Now, tell me about your experience with it."

Feedback Submission:

User Interaction:

- The user provides feedback, saying, "I liked the taste of [Item]."

Java Processing:

- The skill processes the user's feedback, extracting relevant details using the ASK SDK.
- It stores the feedback in Firestore, associating it with the user and the specific item.

Confirmation Message:

User Interaction:

- Alexa confirms the successful submission.

Java Processing:

- The Java code constructs a confirmation message using the ASK SDK.
- It sends the confirmation response to the user.

Exiting the Skill:

User Interaction:

- The user, at any point in the conversation, decides to exit the skill.
- Example: "Alexa, stop" or "Alexa, exit."

Java Processing:

- The ASK Java SDK captures the user's exit command.
- The skill's Java code recognizes the exit command and initiates the exit process.

Exit Confirmation:

- The skill responds with a confirmation message to ensure the user's intention.
- Example: "Sure, you are exiting. If you have more feedback later, feel free to return. Goodbye!"

Cleanup Process:

- The Java code performs necessary cleanup tasks, such as closing database connections or saving any pending data.
- It ensures that the skill is in a clean state before termination.

Session Termination:

- The skill terminates the current session with the user.
- Any temporary data or session-related information is cleared.

Alexa Goodbye Message:

- Alexa provides a default goodbye message as the skill session concludes.
- Example: "Goodbye! If you have more feedback in the future, just say 'Alexa, open [Skill Name].'"

End of Interaction:

- The skill concludes its interaction with the user, and control is handed back to the Alexa service.

Lambda Function Termination:

- The Lambda function handling the skill's logic terminates.

Skill State Reset:

- Any temporary variables or states are reset to ensure a clean slate for the next user interaction.

3.3.2 Java and the Alexa Skills Kit

Lambda Function Execution:
When a user interacts with your Alexa skill, the Alexa service invokes a Lambda function written in Java. This Lambda function serves as the entry point for handling user requests and generating responses.

Java Code and ASK SDK Initialization:
The Lambda function's Java code is responsible for handling the entire interaction flow. The ASK SDK is initialized within the Java code, providing classes and methods for working with Alexa requests and responses.

Request Handlers:
The Java code includes request handlers, each responsible for processing specific intents or user interactions. Handlers implement the RequestHandler interface provided by the ASK SDK.

Intent Schema and Utterances:
The interaction model for your skill includes an intent schema and sample utterances. The intent schema defines the available intents, and the utterances represent example phrases users might say for each intent.

User Interaction and ASK SDK Methods:
When a user interacts with your skill, the ASK SDK provides methods to extract information from the incoming request (e.g. input.getSlot("Product Name").getValue()).

Database Interaction with Firestore:
The Java code interacts with Firestore to retrieve and store data, such as product information and user feedback.

Response Building with ASK SDK:
The ASK SDK provides a response builder to create structured responses for Alexa.

Session Management:

The ASK SDK handles session attributes to maintain state across user interactions.

Exit Handling:
The ASK SDK includes methods for handling exit commands and terminating the skill session.

In summary, the Java code for your Alexa skill uses the ASK SDK to handle user requests, interact with Firestore for data storage and retrieval, and build structured responses. It leverages intent handlers, session management, and various SDK methods to create an engaging and responsive voice-driven experience for users reviewing household food and beverage items. The combination of Java and the ASK SDK simplifies the development process, allowing you to focus on creating compelling interactions for your skill.

3.3.3 Database Integration

Database Connectivity:
Chatbots are integrated with the restaurant's database systems, which can include information on menus, reservations, customer preferences, and more. APIs or database connectors are used to facilitate this connection.

Query Processing:
When a user requests information that requires database access, the chatbot formulates a database query based on the user's request. For example, when a user asks for the menu, the chatbot sends a query to the database to retrieve the menu items and descriptions.

3.3.4 User Interface and User Experience (UI/UX) Design

Conversational Interface:
The chatbot should provide a conversational interface that is intuitive and mimics natural human conversation. This includes using conversational patterns, greetings, and friendly language.

Clear Prompts and Suggestions:
Use clear prompts and suggestions to guide users through the conversation and encourage specific actions, such as ordering, making reservations, or providing feedback.

Error Handling:
Design an error handling mechanism that provides clear guidance and options when users enter queries that the chatbot doesn't understand. Keep users informed and offer alternatives.

Feedback Collection:
Incorporate mechanisms for collecting user feedback on the chatbot's performance and the overall user experience. This feedback can be valuable for continuous improvement.

3.4 Natural Language Processing

3.4.1 Intent Recognition

Natural Language Processing (NLP):
The chatbot utilizes NLP algorithms to process and analyze user inputs. NLP allows the chatbot to understand the meaning and context of the text provided by the user.

Training Data:
Chatbots are trained using large datasets of text to learn how users express different intents. This training data includes examples of user queries and the corresponding intents.

Intent Classification:
During a conversation, the chatbot employs machine learning models to classify the user's input into specific predefined intent categories. Each intent represents a user's intention, such as asking for the menu, making a reservation, or providing feedback.

Context and History:
The chatbot takes into account the context of the conversation and the user's history to improve intent recognition. This helps the chatbot understand follow-up questions or maintain the context of the conversation.

Keyword and Phrase Analysis:
The chatbot may also perform keyword and phrase analysis to identify important terms or phrases related to the user's intent. For example, it might look for keywords like "order," "menu," or "book."

3.4.2 Entity Recognition

Custom Entities:
Chatbots can also be trained to recognize custom entities relevant to the F&B industry. For instance, recognizing menu items, dish names, or feedback categories.

Pattern Matching:
The chatbot may employ pattern matching algorithms to identify entities based on specific formats or structures. For example, recognizing phone numbers, email addresses, or ZIP codes.

Regular Expressions:
Regular expressions can be used to identify entities with well-defined patterns. This can include recognizing common data types, such as currency values or order IDs.

Machine Learning Models:
In some cases, machine learning models may be used to recognize more complex or context-dependent entities, particularly when dealing with unstructured data.

3.4.3 Dialog Management

Handling Interruptions:
Chatbots should be able to handle interruptions gracefully. They need to recognize when users change the topic or ask unrelated questions, and they should be able to return to the main conversation smoothly.

Fallback Responses:
In cases where the chatbot doesn't understand a user query or intent, it should have fallback responses or mechanisms to handle the situation and keep the conversation going.

User Prompts:
The chatbot may prompt users for additional information if a user query requires more details. These prompts guide users through the conversation and help the chatbot gather necessary information.

3.5 Data Security and Privacy Measures

Data Encryption:
Implement strong data encryption protocols, such as SSL/TLS, to secure data during transmission between the user and the chatbot. Encryption ensures that data remains confidential and cannot be

intercepted by unauthorized parties.

Secure Storage:
Data collected by the chatbot should be securely stored. Use secure servers and databases, and regularly update and patch them to protect against vulnerabilities.

Access Controls:
Implement strict access controls to restrict who can access and modify user data. Only authorized personnel with a legitimate need should have access to sensitive information.

Data Minimization:
Collect only the minimum amount of data necessary to provide the intended service. Avoid collecting excessive or unnecessary information, which could pose a higher security risk.

Anonymization and Pseudonymization:
Anonymize or pseudonymize user data whenever possible. This reduces the risk associated with handling personal information and makes it less identifiable.

Transparent Privacy Policy:
Provide a clear and accessible privacy policy that explains how user data is handled, who has access to it, and the purposes for which it is used. Users should be aware of their rights and how to exercise them.

3.6 Integration with Feedback Recording System

Feedback Categorization:
The chatbot should be programmed with the ability to categorize and tag feedback based on predefined criteria. This can include tagging feedback as positive, negative, suggestions, or specific issue categories.

Data Structuring:
The collected feedback data, including text responses, ratings, and other relevant information, is structured and formatted for further analysis. This data is organized into a structured format that can be easily processed.

Feedback Database:
Within the feedback recording system, a dedicated database or repository stores the incoming chatbot-generated feedback. This database may include tables or fields for various elements such as date, customer ID, feedback type, and specific comments.

3.7 User Training and Testing

User Training:
Onboarding Process:
Provide a seamless onboarding process for users to introduce them to the chatbot. Offer a brief tutorial or introductory conversation explaining the chatbot's capabilities, how to initiate interactions, and the value it offers.

User Guides and FAQs:
Create user guides and frequently asked questions (FAQs) to help users understand how to engage with the chatbot effectively. Include step-by-step instructions, command examples, and tips for getting the most out of interactions.

In-App or Website Prompts:

Use in-app or website prompts to guide users to initiate conversations with the chatbot. Prominently display the chatbot's availability and encourage users to ask questions or provide feedback.

Error Handling Guidance:
Offer guidance on how to handle errors or misunderstandings in chatbot interactions. Instruct users on what to do if the chatbot doesn't understand their request and how to request human assistance if needed.

Testing:
Functional Testing:
Test the chatbot's functionality to ensure it performs as expected. Evaluate its ability to understand user queries, provide accurate responses, and execute actions like placing orders or making reservations.

Usability Testing:
Conduct usability testing with representative users to assess the chatbot's ease of use. Evaluate the user interface, navigation, and overall user experience. Identify pain points and areas for improvement.

Performance Testing:
Evaluate the chatbot's performance under various conditions, including heavy user loads. Measure response times, uptime, and scalability to ensure it can handle peak demand.

User Satisfaction Surveys:
Administer user satisfaction surveys to gather feedback on the chatbot experience. Use metrics such as the System Usability Scale (SUS) or Net Promoter Score (NPS) to gauge user satisfaction.

Natural Language Understanding (NLU) Testing:
Assess the chatbot's NLU capabilities by conducting tests that involve different accents, languages, and dialects. Test its ability to comprehend user intents accurately.

3.8 Ethical Considerations in Design

Transparency:

1. Clear Identification: Chatbots should be transparently identified as such from the beginning of the conversation. Customers should know that they are interacting with an automated system rather than a human.

2. Clarify Capabilities: Inform users about what the chatbot can and cannot do. Set clear expectations for the type of assistance it provides. For instance, let users know if the chatbot is incapable of handling certain complex queries.

3. Provide Information: When requested, the chatbot should provide information about its purpose, the data it collects, and how that data is used. Users should be informed about how their interactions are contributing to improvements in service.

Consent:

1. Obtain Informed Consent: Before collecting any personal data or proceeding with more detailed interactions, chatbots should obtain informed consent from users. This consent should be explicit, specific, and revocable.

2. Opt-In vs. Opt-Out: Users should be given the choice to opt-in for data collection and interactions with the chatbot rather than being automatically enrolled. Opt-out options should also be available.

3. User Control: Users should have the ability to control the level of interaction with the chatbot. They should be able to pause or end interactions at any time.

Data Handling:

1. Data Minimization: Collect only the data that is necessary for the intended purpose. Avoid unnecessary data collection to protect user privacy.

2. Data Security: Implement robust data security measures to protect user data from breaches and unauthorized access. Encryption and access controls are essential.

3. Data Retention: Establish clear data retention policies. Users should be informed about how long their data will be stored, and it should not be kept longer than necessary.

4. Anonymization: When possible, anonymize data to ensure that individual user identities cannot be traced back to the feedback or interactions provided.

5. Third-Party Sharing: If data is shared with third parties, be transparent about the recipients and the purpose of sharing. Users should have the option to opt-out of such sharing.

6. User Access and Deletion: Provide users with the ability to access their data and request its deletion. Comply with user requests promptly.

7. Regular Audits and Compliance: Regularly audit data handling practices to ensure compliance with data protection laws and regulations, such as GDPR or CCPA, depending on the jurisdiction.

User Feedback and Improvement:
Encourage users to provide feedback on their interactions with the chatbot. Use this feedback to continually improve the chatbot's performance and ethical compliance.

3.9 Requirements

3.9.1 Functional Requirements

1. User Authentication:

   - Users should be able to create accounts and log in securely to personalize their feedback experience.

2. Item Navigation:

   - The skill should provide users with a clear and concise list of available household food and beverage items for feedback.

3. Item Details:

   - Users should be able to request additional details about a specific item, including its origin, nutritional information, or special characteristics.

4. Feedback Submission:

   - Users should have the ability to submit feedback on various aspects of a selected item, such as taste, freshness, packaging, etc.

5. Confirmation Messages:

   - The system should provide clear confirmation messages after users select an item, submit feedback, or perform any critical actions.

6. Feedback History:

   -

Users should be able to view and review their past feedback history for items. 7. Recommendations:

   - Based on a user's feedback history, the system should provide personalized recommendations for similar items.

8. User Preferences:

   - Users should have the option to customize preferences, such as notification settings and language preferences.

9. Help and FAQs:

   - The skill should offer a help menu or frequently asked questions (FAQs) to guide users and provide assistance.

10. Error Handling:

    - The application should handle errors gracefully, providing clear and helpful error messages when unexpected issues occur.

11. Admin Authentication:

    - Admins should be able to log in securely to access the back-end functionality using Firestore authentication.

12. Admin Dashboard:

    - An admin dashboard should be available, providing a centralized interface for monitoring user feedback, analytics, and managing items.

13. User Feedback Management:

    - Admins should be able to view, analyze, and manage user feedback, including the ability to respond to user comments.

14. Item Management:

    - Admins should have the capability to add, edit, or remove household food and beverage items available for feedback.

15. User Analytics:

    - The back-end should provide analytics tools for admins to gain insights into user behavior, preferences, and overall system usage.

16. Notification Management:

    - Admins should be able to manage and send notifications to users, including promotional messages and announcements.

17. User Account Management:

    - Admins should have the ability to view and manage user accounts, including account suspension or removal if necessary.

3.9.2 Non-Functional Requirements

1. Response Time:

   - The system should respond to user interactions within 2 seconds to maintain a smooth and responsive user experience.

2. Scalability:

   - The application should be designed to handle a growing user base and increased data load without compromising performance.

3. Security and Privacy:

- User data should be stored securely in Firestore with encryption and adhere to data protection standards to ensure user privacy.

4. Cross-Device Compatibility:
   - The skill should be compatible with various Alexa devices, ensuring a consistent user experience across different platforms.
5. Accessibility:
   - The application should adhere to accessibility standards, providing an inclusive experience for users with different abilities.
6. Language Support:
   - The skill should support multiple languages to cater to a diverse user base.
7. Notification Reliability:
   - Push notifications, if implemented, should be reliable and delivered promptly to users who have opted in.
8. Continuous Improvement:
   - The skill should be designed for easy updates and improvements based on user feedback, ensuring a continuous enhancement of features.
9. Conversational AI:
   - Natural Language Processing (NLP) should be implemented to enhance the conversational aspect of the skill, making interactions feel more natural and engaging.
10. User Feedback Loop:
    - The system should include mechanisms for collecting user feedback on the skill itself to guide future improvements and enhancements.

3.10 Implementation Approach

When first deciding on the direction of this project, I opted to write this application in Java, using React for the back-end. I chose this direction due to Java being the language I have most experience with compared to other options like NodeJS. Similarly, for the back-end web service for admins, I have most experience with React and chose to write the back-end in this manner as it can be utilised to quickly and easily build a web page which can interact with other aspects of the application.

In the first week of the project, I set up the Amazon Developer Console with my account and created a simple "Hello World" skill to familiarize myself with the testing process and different aspects of skills. I used NodeJS in this first test as it allowed me to run and test the skill solely through the developer console, however this would be my last use of NodeJS.

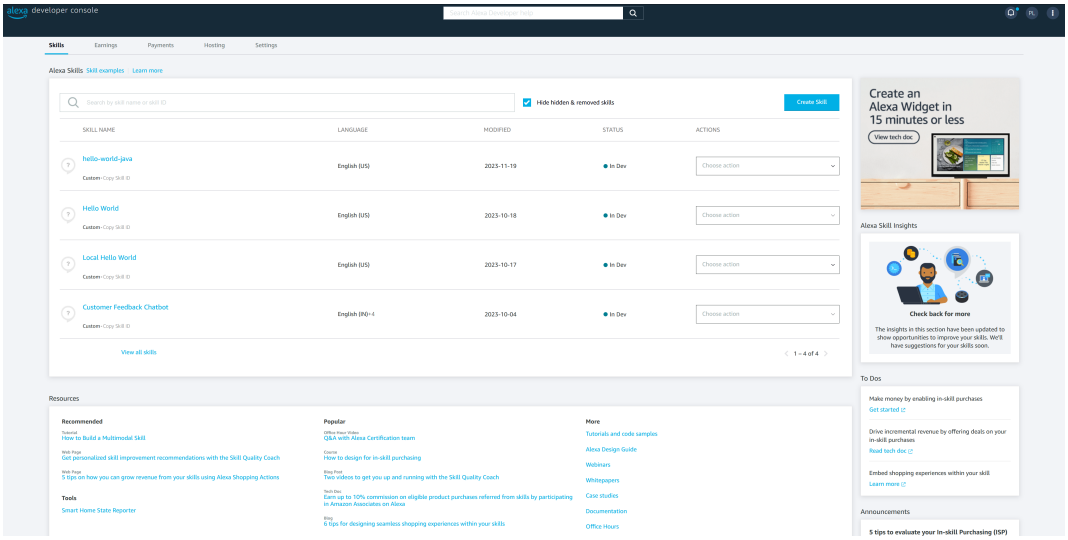

Fig 3.2 Dev Console Home

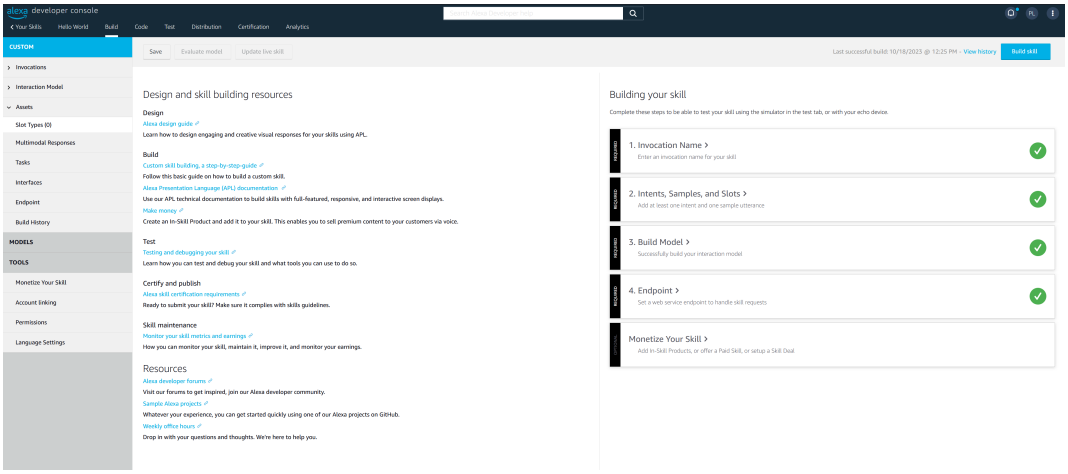

Fig 3.3 Dev Console Build Tab

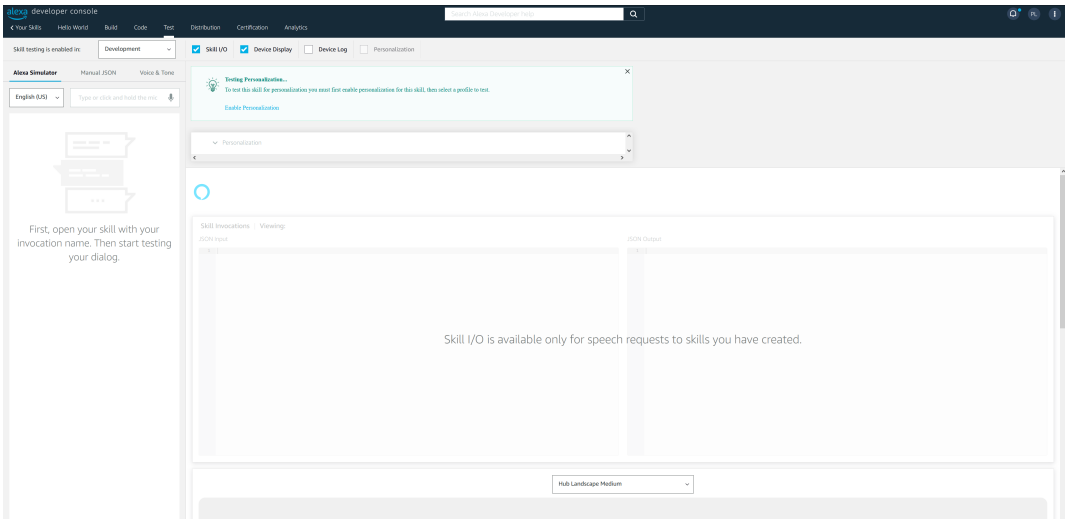

Fig 3.4 Dev Console Testing Tab

After familiarizing myself with the different aspects of skills, I moved on to testing a Java skill. Creating a skill on my local machine was an easy process thanks to the guidance provided by Amazon when installing the SDK through the console. When I had a skill built in Java on my local machine (again

a simple "Hello World" skill for testing purposes), I dove into researching various hosting options. For this early stage I chose NGROK to host my skill. This is a free tool which allowed me to access the skill located on my local machine from the Amazon Developer Console and quickly push any changes to the skill allowing for fast testing and debugging. While this initial project was a good proof of concept, I had by this time decided to use AWS Lambda to host the final product. NGROK is excellent for local testing but would require a personal machine running constantly to host the skill, opening up the skill to numerous possible failures, and myself to serious security concerns. While AWS Lambda has the potential to wrack up quite a sizeable fee very quickly, options can be tweaked to prevent any charges being made. This does come at a cost however, as traffic on the app will be hamstrung by the free usage of this service.

With this initial success in implementing a Java skill, my attention shifted to the flow of the chatbot.

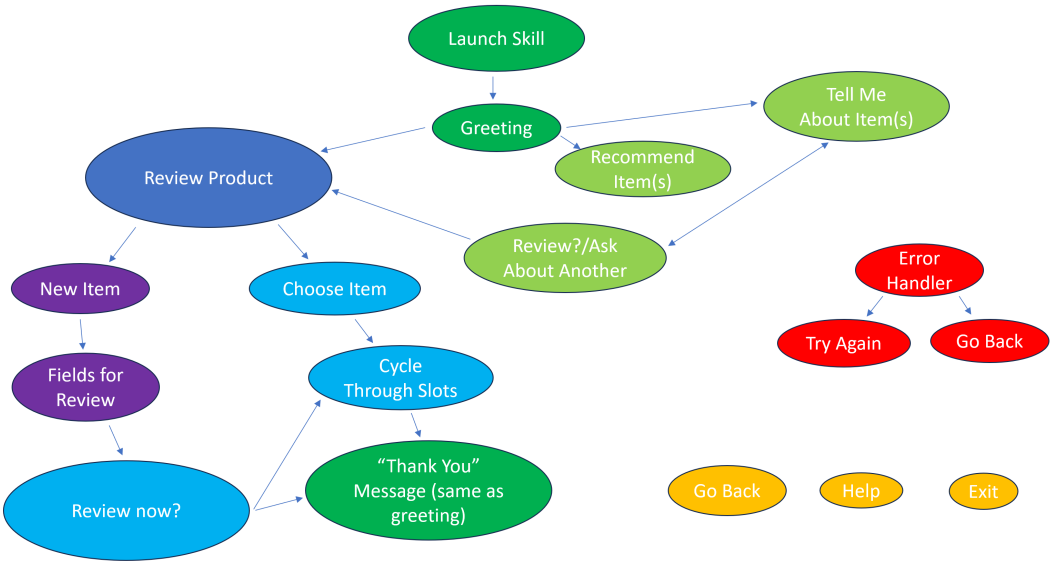

Fig 3.5

The above image depicts the dialogue flow for the chatbot.

1. When the user launches the skill, they receive a message welcoming them to the app and outlining their options.
2. The user can ask for recommendations on products based on their previous reviews.
3. Users can ask for information on specific items.
4. Following an information request, users can review a product or ask about more products.
5. When reviewing a product, the user can choose to add a new item for review.
6. When adding a new item, users specify the slots required for the item's review. As well as the specific slots, a field is added for additional comments not covered by the slots.
7. After successfully adding a product, the user receives confirmation of the new product and can choose to return to the main greeting intent or review the product immediately.
8. If a user chooses an item to review instead of adding a new product, the chatbot cycles through the slots required for the product review until it's ready to be submitted.
9. The user has an option to try again when an error occurs whether this is the chatbot being unable to understand the user or an invalid input.
10. The user has the option to go back to the previous point in the conversation at any time, with the exception of immediately after a product has been added or reviewed.

11. The user can ask for help and be given advice or guidance accordingly.

12. The user can exit the app at any time.

The next step from here is the admin side of the app. Following my decision to use React for the admin web-app, I settled on Google Firestore to host the website. This choice was heavily influenced from my previous experience with the platform, along with it's ability to interact with AWS Lambda. This choice also provides huge benefit in the form of Firestore authentication. This allows me to easily manage admins and users, as well as providing excellent security and user authentication for my skill.

In order to quickly develop and add functionality to the web-app, I will also be utilising the open source library "Ant Design". Using this allows my to quickly and easily add functionality to the app with a sleek design, allowing me to focus on the skill while giving administrators all the access and functionality they need.

Following my decision to utilize Google Firestore for the web-app and user authentication, I also decided to use this service's built-in database functionality.

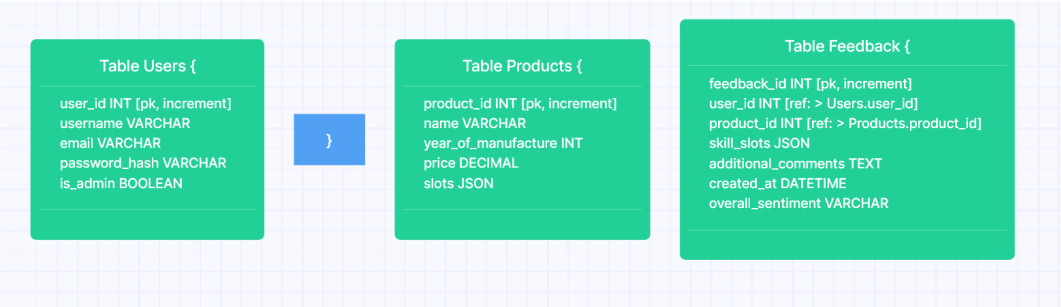

Fig 3.6

This database model shows how the data for products is stored, as well as the feedback users submit. When choosing between a relational and NoSQL database, I opted to proceed with the project using the latter. This decision was made in anticipation of the database having to scale dramatically to handle a comprehensive product list.

3.11 User Stories

For this project, I decided that an agile approach to development was most appropriate. With this in mind, I began looking into methods of creating, managing, and keeping track of user stories. I initially wanted to use Jira as I had experience with this platform and found it extremely easy and intuitive, however, after some time researching free versions or academic keys yielded no results, I decided to explore other options. The first alternative that came to mind was Trello as I, again, had some experience with it (although not nearly as much). After some time debating other free options, I settled on using Trello to map out my user stories.

I initially had the development process divided into three sprints of length four weeks each. After reflecting on this and seeing the amount of stories per sprint, I decided two week sprints would be easier to manage and keep track of progress.

3.11.1 Sprint 1

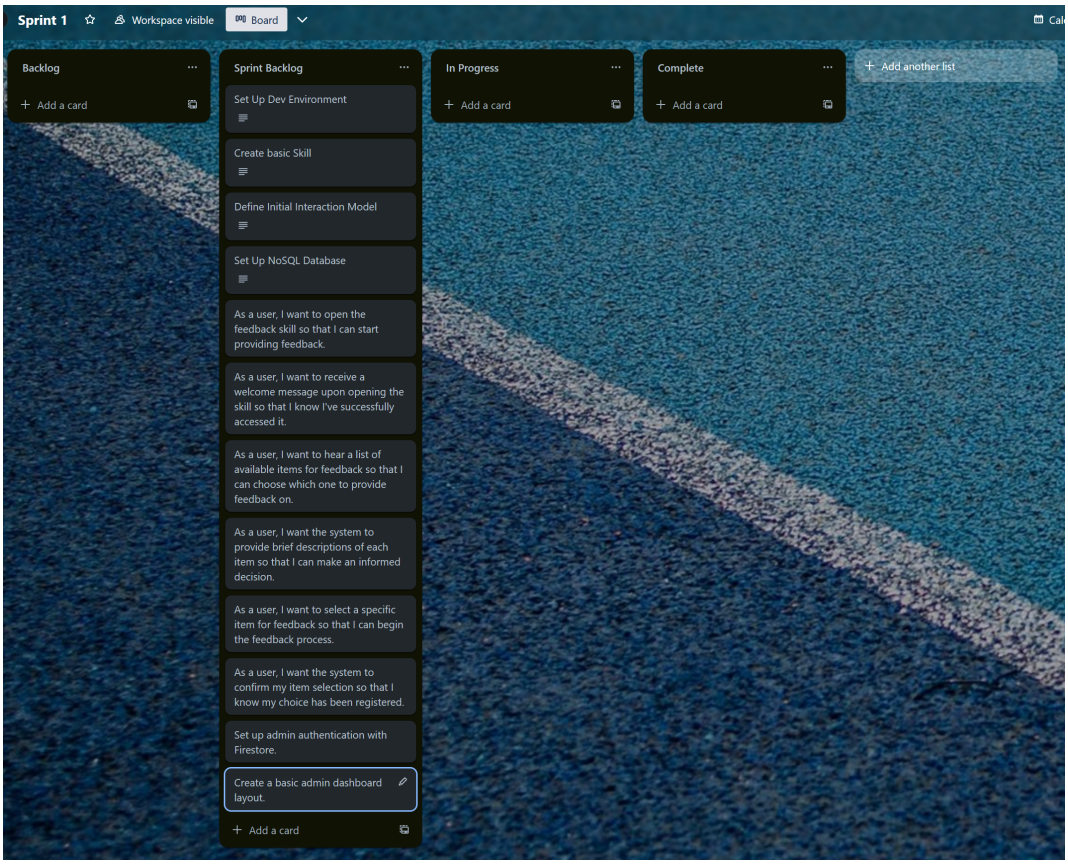

Fig 3.7

1. Setting up development environment
2. Creating a basic skill to begin working on
3. Define an initial interaction model for testing
4. Set up the database in Google Firestore
5. User launch skill
6. User receive welcome message
7. User hears list of items from database
8. User hears brief descriptions of specific items
9. Confirm user selection of an item for review
10. Set up admin authentication in Google Firestore
11. Create a basic admin dashboard

3.11.2 Sprint 2

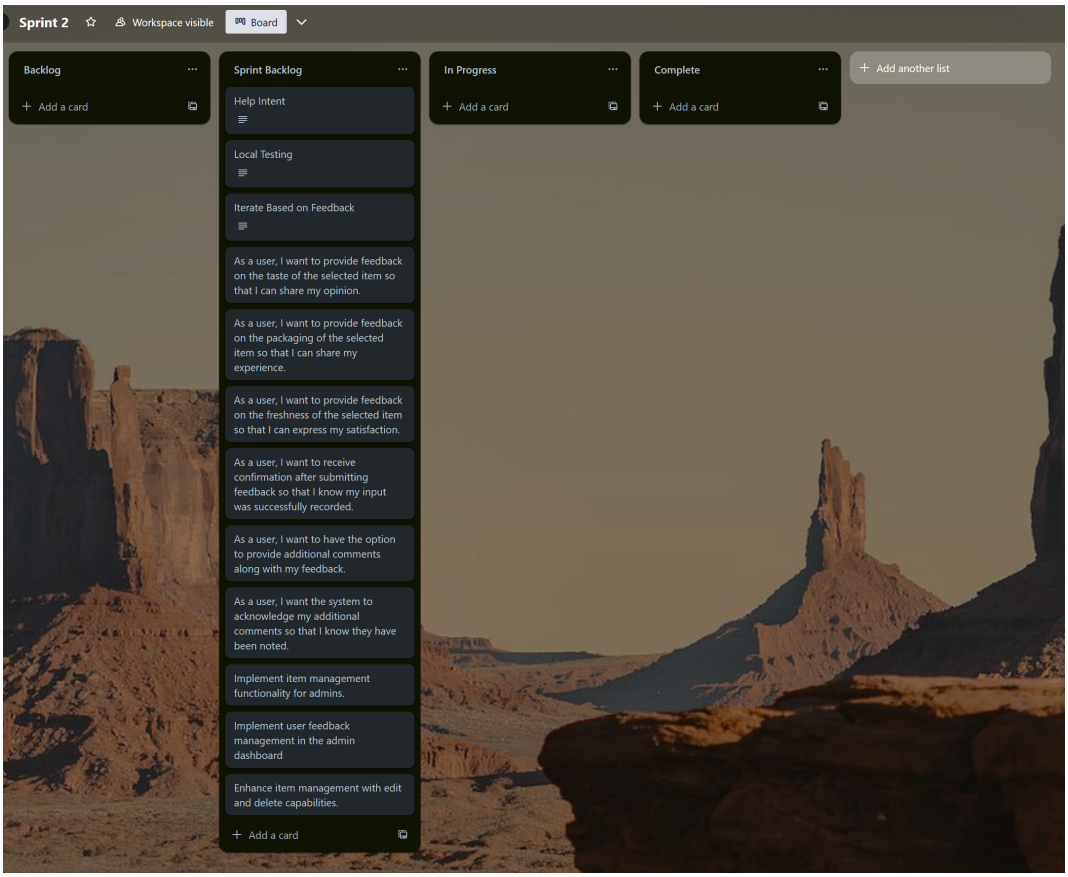

Fig 3.8

1. Define the help intent
2. Begin local testing of the app
3. Iterate on the design based on third party feedback
4. Allow users to provide feedback
5. Give users confirmation of receipt of their feedback
6. Allow admins to manage items in the database
7. Allow admins to manage user feedback

3.11.3 Sprint 3

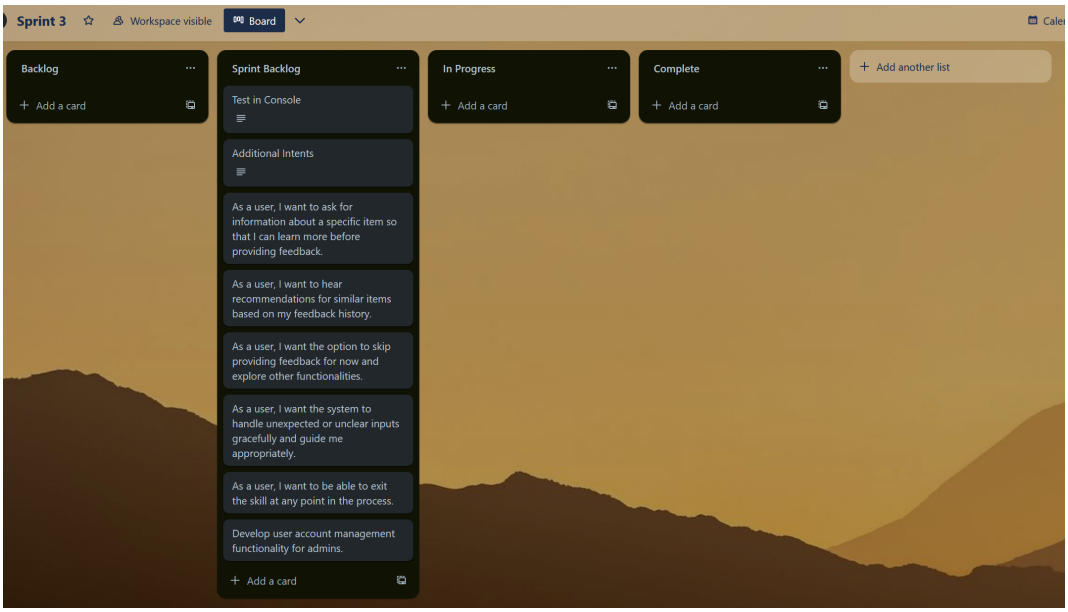

Fig 3.9

1. Test the app in the Alexa Developer Console
2. Define additional intents
3. Allow users to request specific information on items
4. Give users product recommendations
5. Allow the user to access other functionalities than providing feedback
6. Implement error handling
7. Allow user to exit the app
8. Develop user account management for admins

3.11.4 Sprint 4

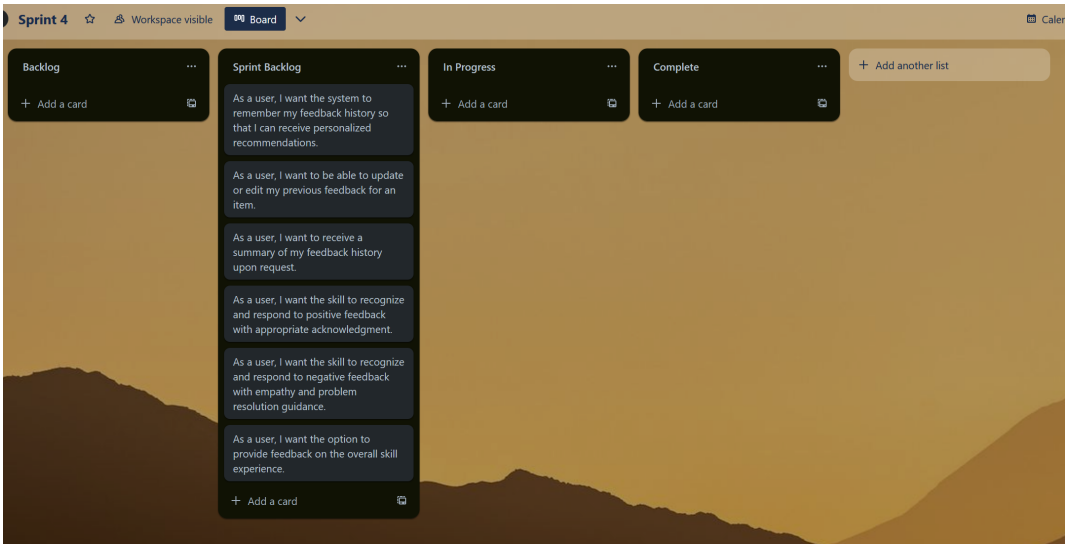

Fig 3.10

1. Provide personalised recommendations
2. User receives feedback history summary
3. Customize feedback confirmation based on feedback sentiment
4. Allow user reviews of the app itself

3.11.5 Sprint 5

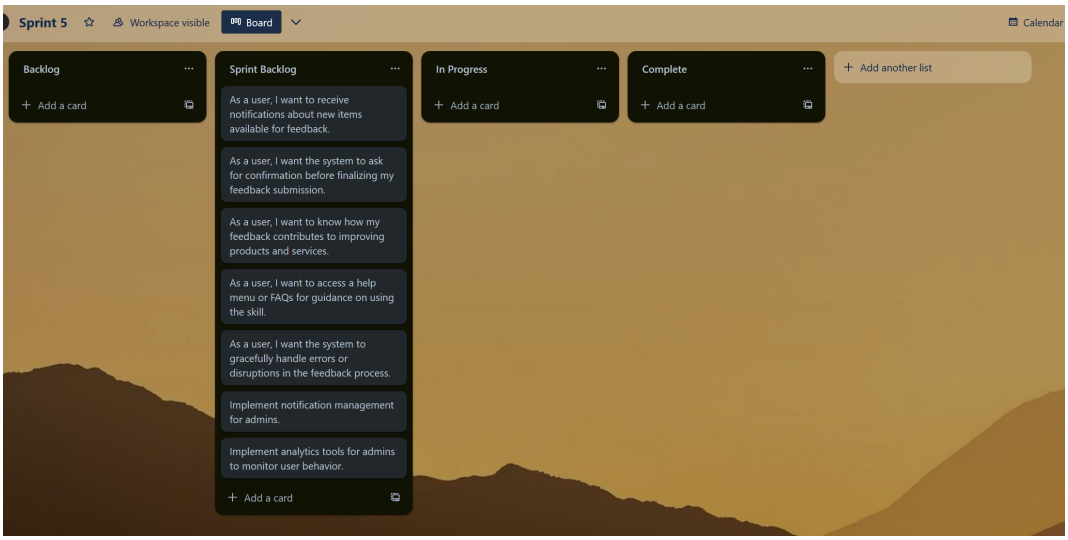

Fig 3.11

1. Give users notifications for newly available items
2. Ask for user confirmation before submitting feedback
3. Add an FAQ
4. Edit error handler for invalid inputs or unexpected interruptions
5. Admin notification management
6. Analytical tool for admins

3.11.6 Sprint 6

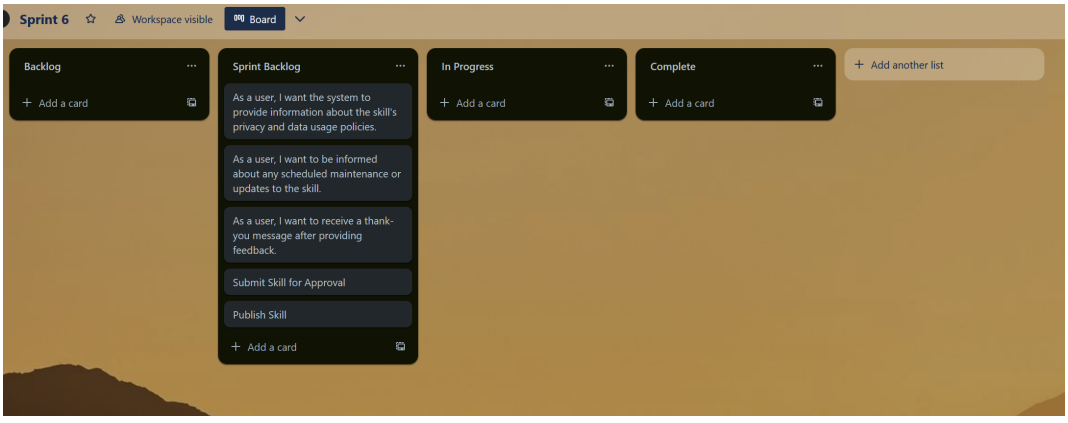

Fig 3.12

1. Give user information of privacy and data usage on request
2. System to inform users of of scheduled maintenance and outages

3. Submit skill for approval for the Amazon app store
4. Publish the skill

4 Development and Evaluation

4.1 Setup

4.1.1 Setting up AWS ASK and Adding Credentials for AWS Access

To begin this project, I first navigated to the AWS Management Console. Here I created a new IAM root user and generated both an access key ID and a secret access key for authentication. Next, I assigned all necessary permissions to the account (AlexaForBusinessFullAccess, AmazonDynamoDB-FullAccess, AWSLambdaFullAccess, etc.).

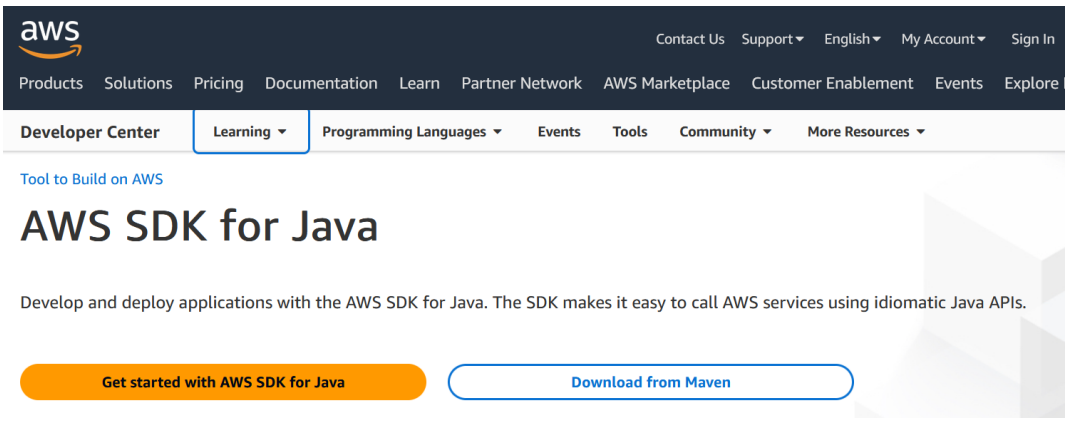

Fig 4.1

4.1.2 Installing and Configuring the AWS Command Line Interface (CLI)

From here, I downloaded the AWS CLI onto my development machine from the official AWS website. Once it was installed, I configured the CLI by running 'aws configure' in the command line. After entering both access keys and setting region and other information, my development machine was ready to go. This step was key in saving time, allowing me to update my function from the command line built into my IDE (Intellij), instead of manually uploading each new version.

To set up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:
   - Installing the AWS Command Line Interface
   - Configuring the AWS CLI
2. Add a named profile for the administrator user in the AWS CLI `config` file. You use this profile when executing the AWS CLI commands. For more information about named profiles, see Named Profiles in the *AWS Command Line Interface User Guide*.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

For a list of available AWS Regions, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Fig 4.2

4.1.3 Intellij

The IDE I chose for this project is Intellij, due to it's ease of use with maven and intuitive setup of build configurations. A useful feature of the IDE is the built-in terminal, allowing me to push a new version to the s3 bucket and also inform the function that there is a new version, all within one environment.

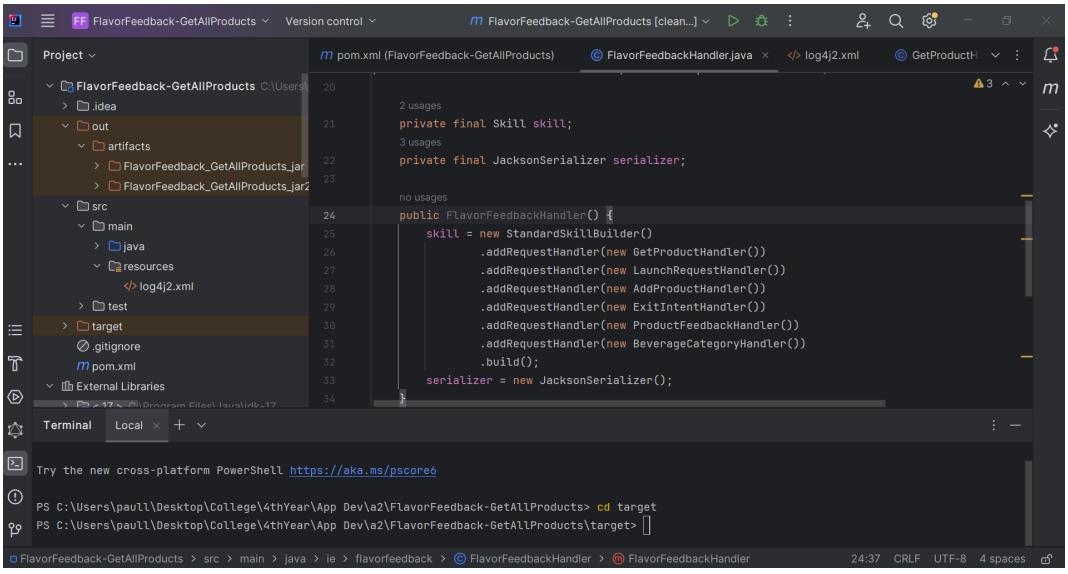

Fig 4.3.1

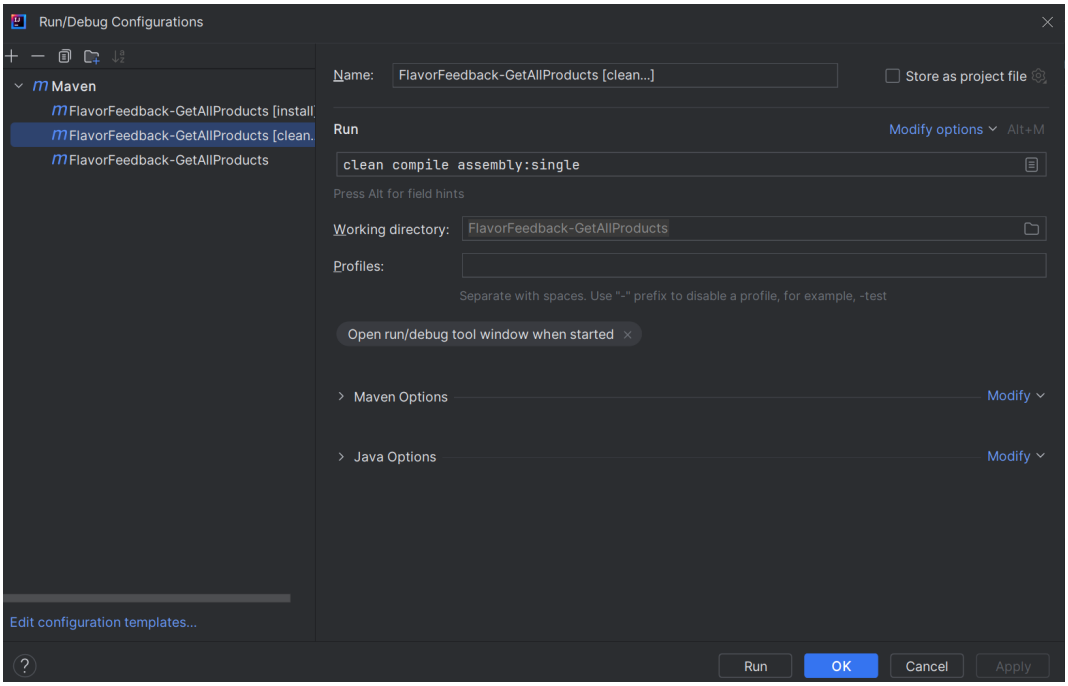

Fig 4.3.2

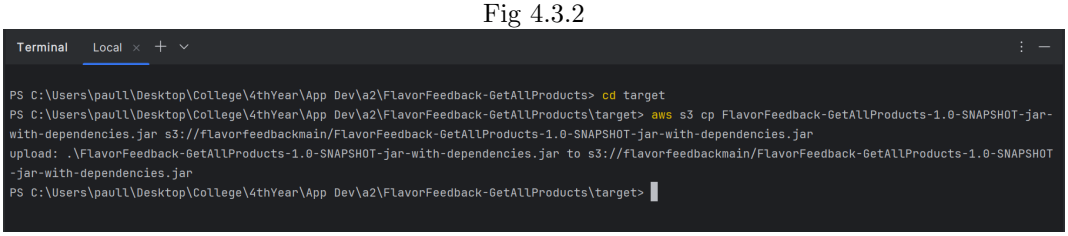

Fig 4.3.3

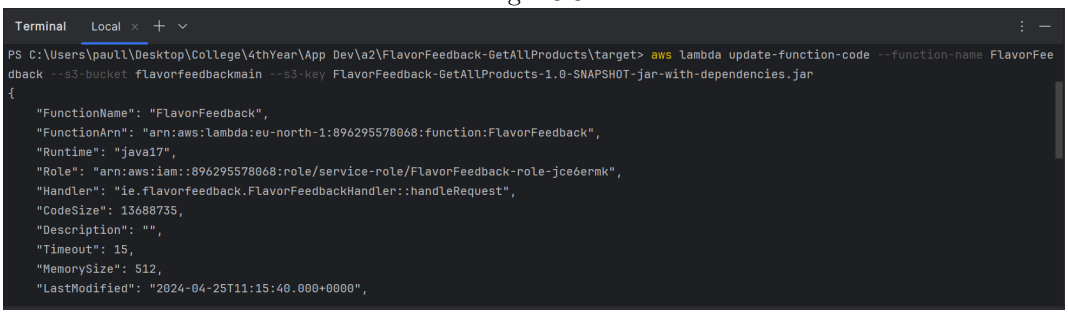

Fig 4.3.4

4.1.4 Creating a Lambda Function

The next step is to create a Lambda function. When creating this function, I allowed it to run from a jar file. This meant that I could compile in Intellij and upload my entire skill in a singular file. I next set up the function to be triggered by the Alexa Skills Kit. This connected the skill to the function, allowing AWS to handle requests and responses.

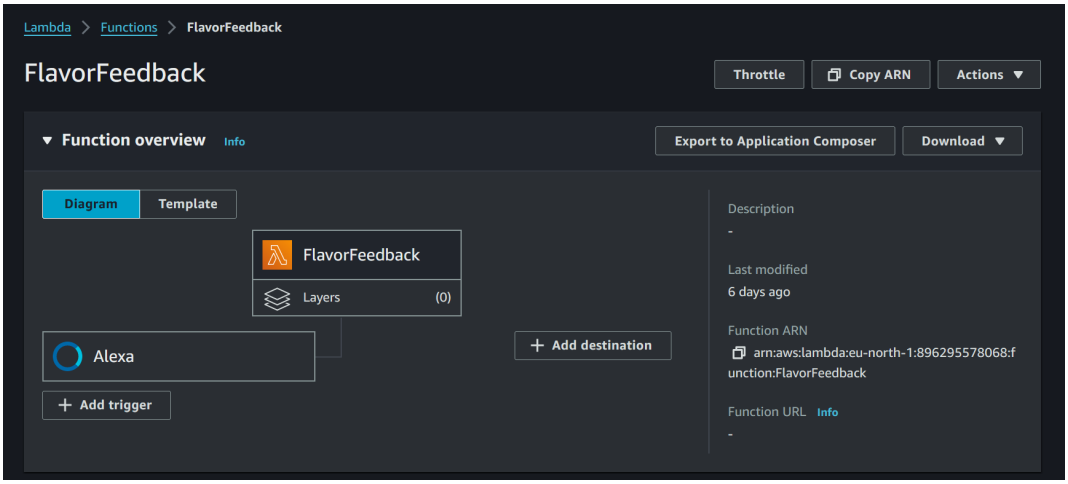

Fig 4.4.1

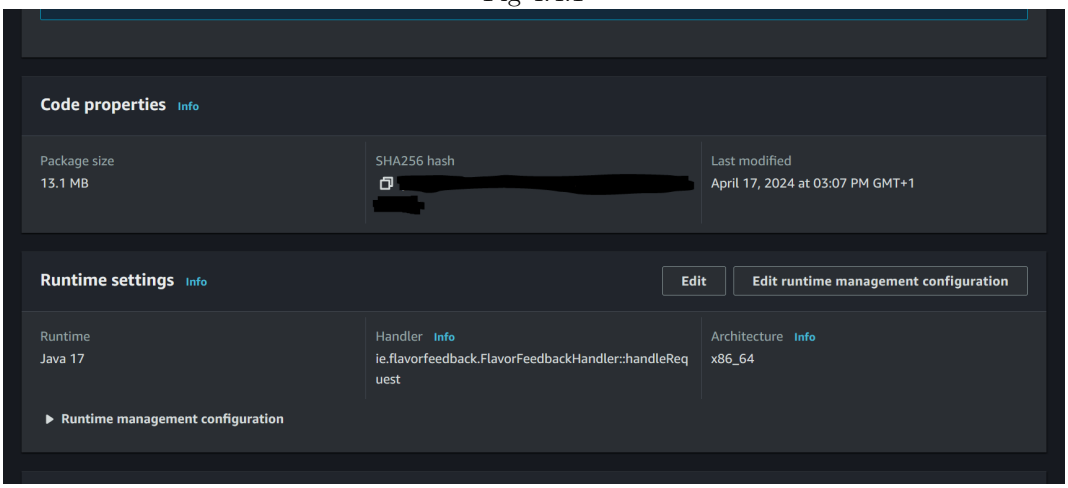

Fig 4.4.2

4.1.5 Setting up DynamoDB

When setting up my databases using DynamoDB, I used my previously outlined schema. DynamoDB is very intuitive and only required a primary key for a table to be defined. I then updated the permissions of DynamoDB and Lambda function to allow the two to communicate and for the function to be able to manipulate the data in the table.

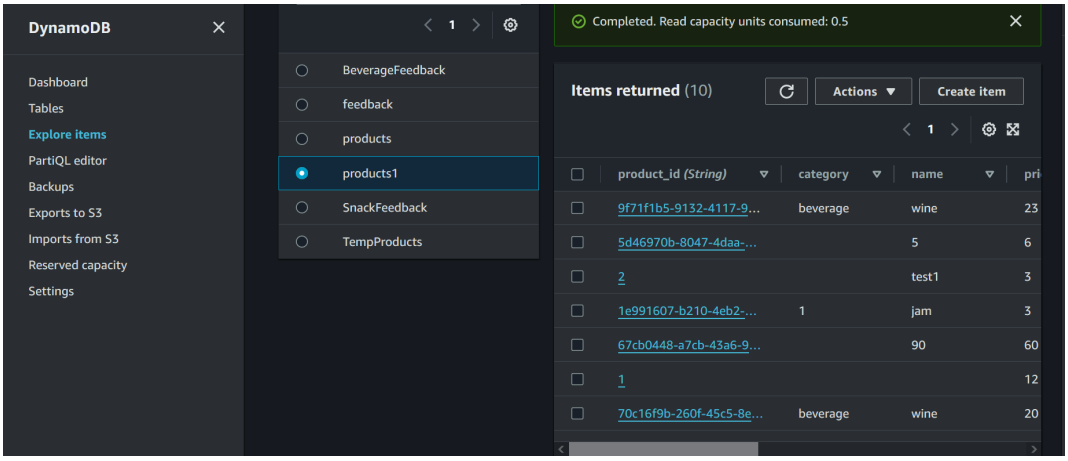

Fig 4.5

4.1.6 Configuring S3 Bucket

Creating an S3 Bucket is a very simple process through the AWS console. I will be using it to store a static jar file which will contain the entire skill. The Lambda function will then use the file contained in that bucket after ensuring my account has the necessary permissions to ensure secure access to the appropriate files.

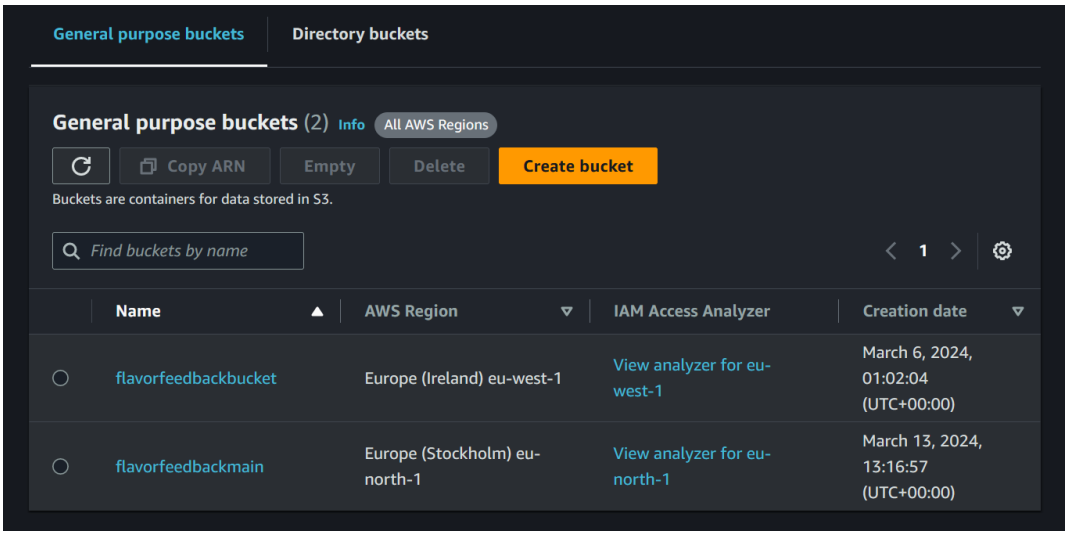

Fig 4.6

4.1.7 Cloudwatch

Cloudwatch was a key part of this project. Again, having never used the service before I was a bit unsure at first but it ended up being very intuitive. It allowed me to quickly see what errors my skill was producing in order to rectify them.
While the usage of the Cloudwatch service ended up being very straightforward, the errors returned by the skill weren't always. At times "An unknown error occurred" would be the error message shown and I would be left to figure out what could be causing a seemingly random issue in my project.

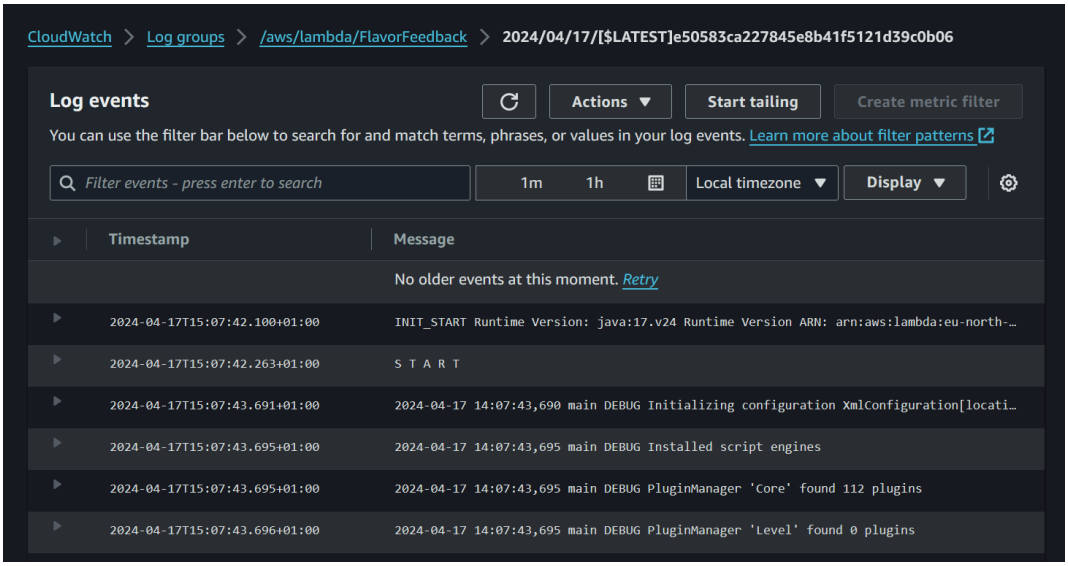

Fig 4.7

4.1.8 Integrating Firestore

In my initial design for this project, I had outlined my intention to use Google Firestore for database and user authentication. When implementing this design, I found getting my lambda function to communicate with firestore and giving it all the necessary permissions just wasn't feasible in the time-frame I was working with. For this reason, I decided to implement my skill solely using AWS and its sub-services.
As I worked on this project further, I was pleasantly surprised with the ease-of-use and security provided by services like DynamoDB. Staying within the Amazon ecosystem proved to greatly improve the functionality of my app as well as speeding up the development process.

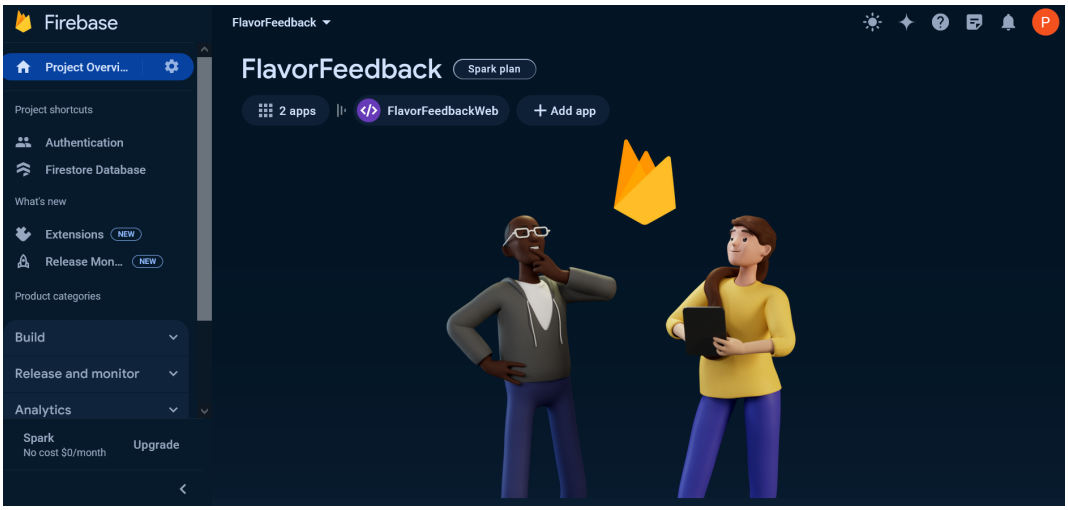

Fig 4.8

4.1.9 Development Console Setup: Intents, Slots, etc.

In the setup of the Alexa Developer Console, I defined the skill name, language, and set the model to custom to allow full freedom in development. I then added intents, slots, and utterances as they were needed through development. I also configured the endpoints so that the AWS Lambda ARN would be utilized by the skill.

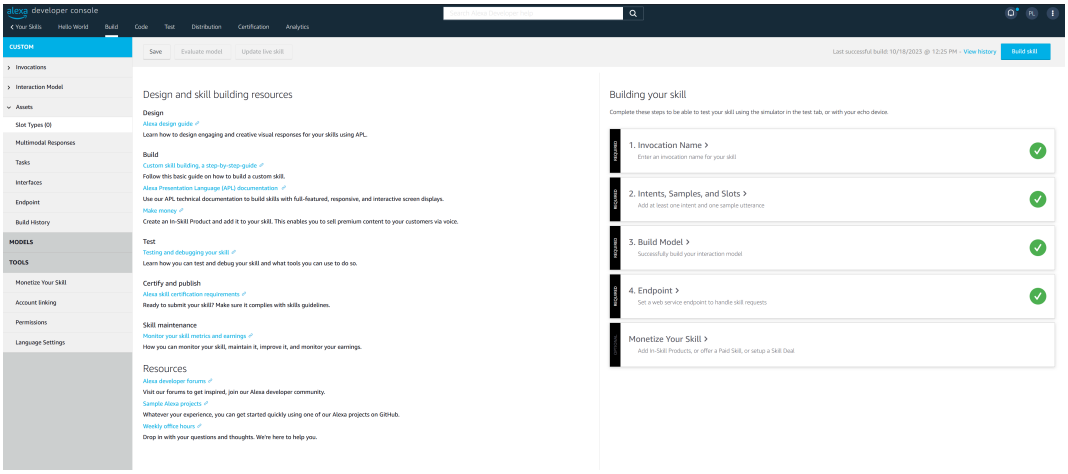

Fig 4.9.1

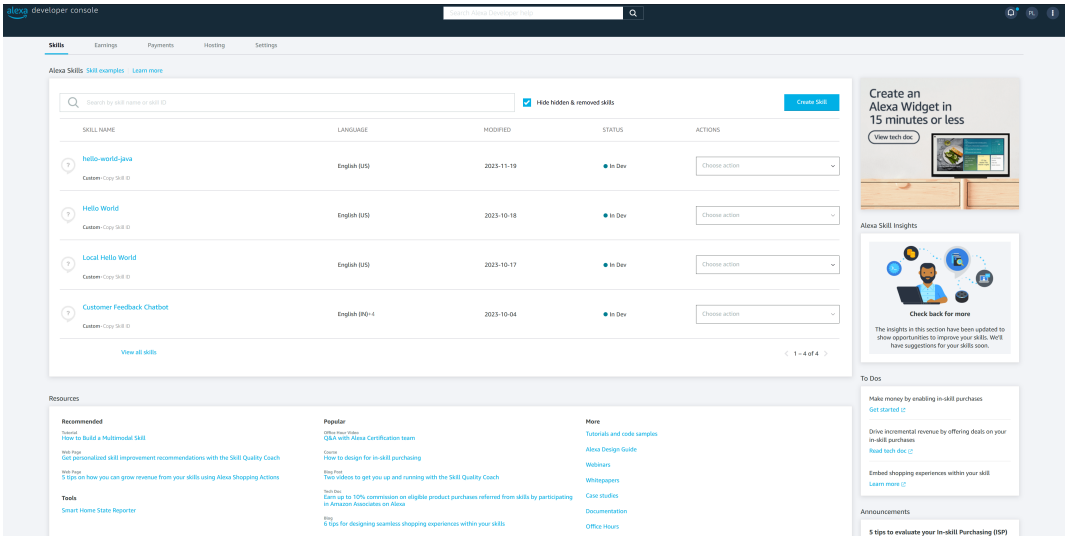

Fig 4.9.2

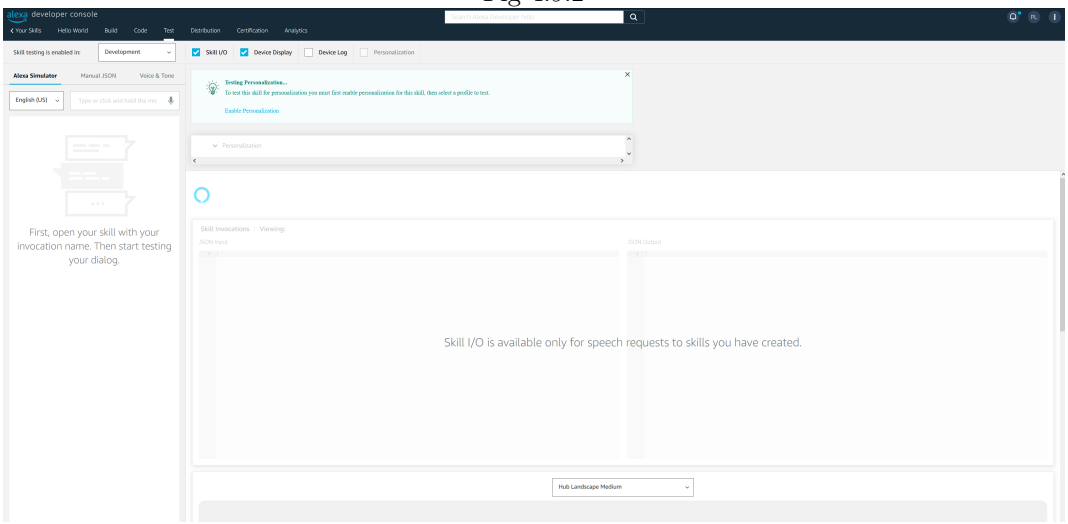

Fig 4.9.3

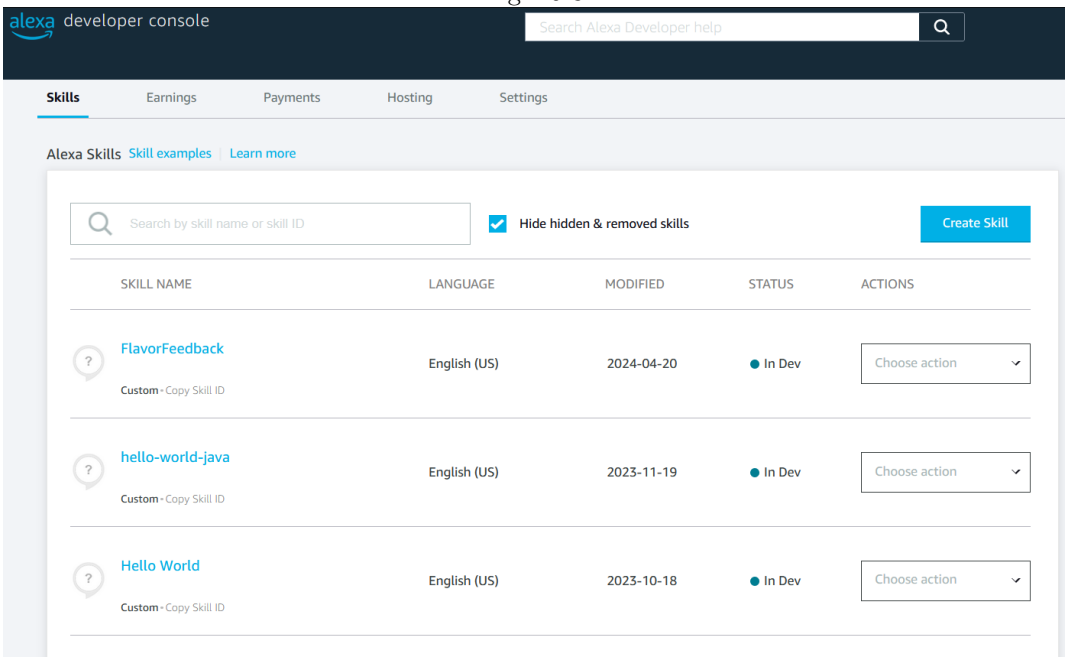

Fig 4.9.4

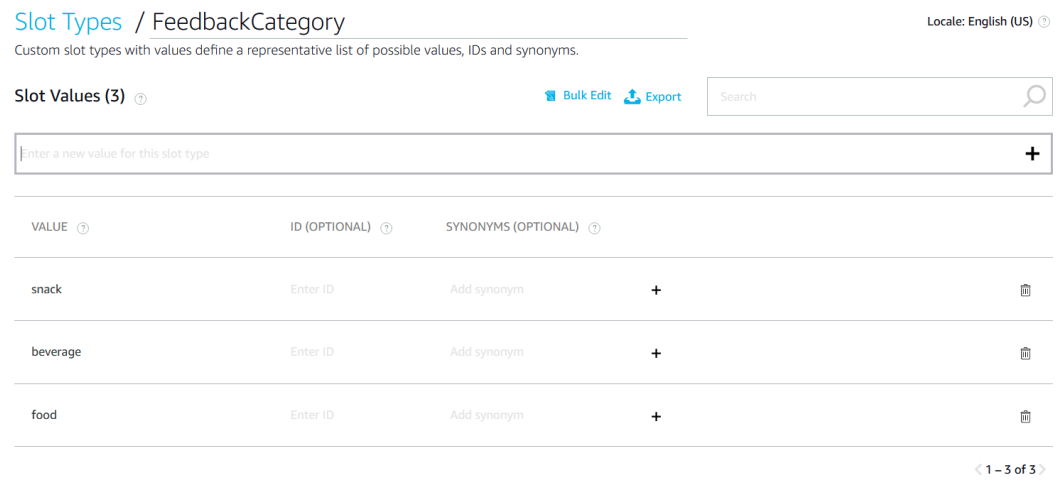

Fig 4.9.5

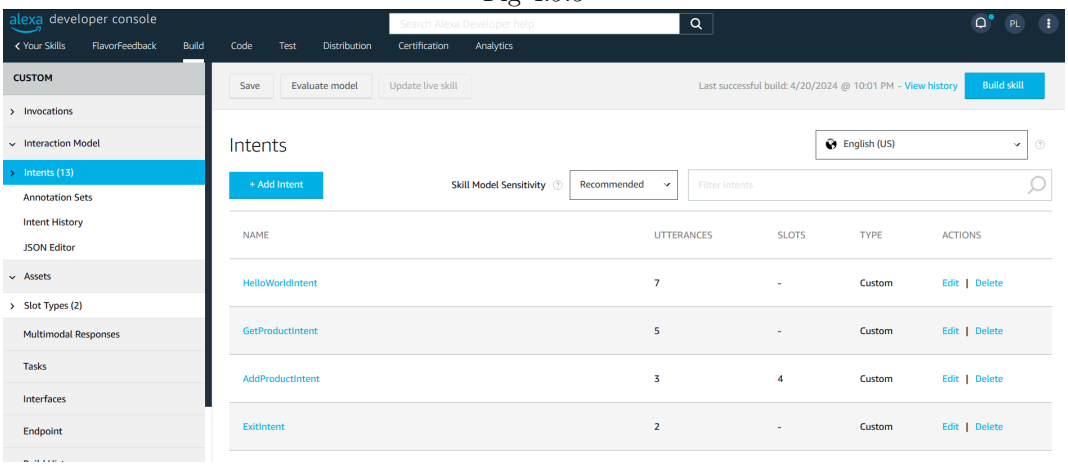

Fig 4.9.6

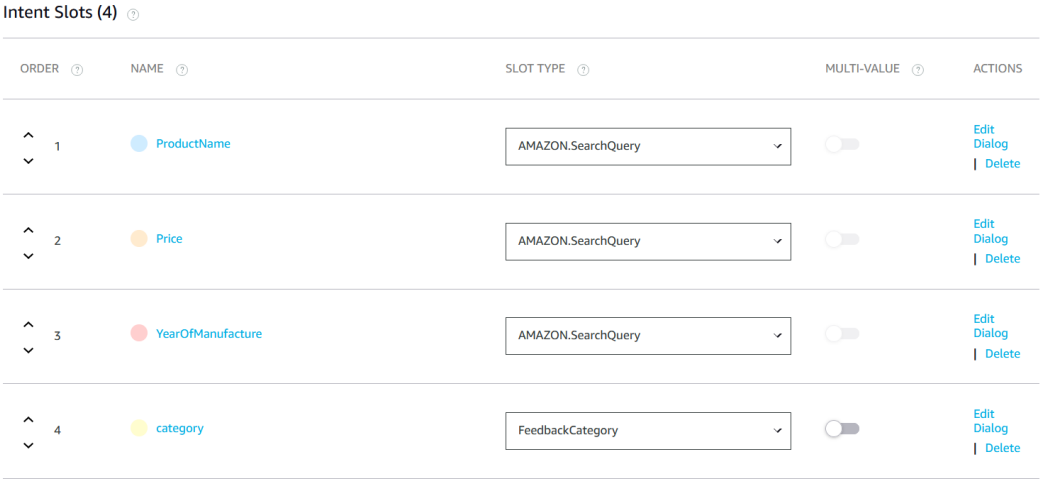

Fig 4.9.7

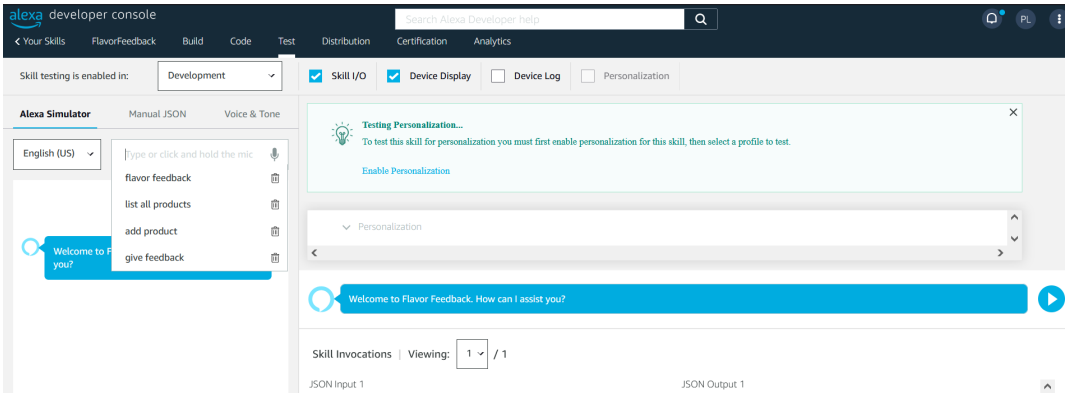

Fig 4.9.8

Fig 4.9.9

4.2 Intent Specification

4.2.1 Main Intent

Purpose:
This class is responsible for orchestrating the flow of information between the user's voice commands processed by Alexa and the skill's logic. It initializes the skill with various request handlers, each designed to manage specific types of interactions, such as launching the skill, adding a product, or providing feedback on a product.

Functionality:
Initialization: In the constructor, the FlavorFeedbackHandler creates an instance of the skill using the StandardSkillBuilder. This instance is configured with multiple request handlers:

- GetProductHandler manages inquiries about products.
- LaunchRequestHandler deals with the skill's launch requests.
- AddProductHandler handles adding a new product.
- ExitIntentHandler manages requests to exit the skill.
- ProductFeedbackHandler takes care of gathering user feedback on products.
- BeverageCategoryHandler is specialized in handling beverage category-related inquiries.

Request Handling: It overrides the handleRequest method from the RequestStreamHandler interface, which is the AWS Lambda standard for handling incoming requests. This method:

- Reads the incoming request from Alexa through the InputStream.
- Deserializes the request into a RequestEnvelope object using JacksonSerializer, making it easier to work with in Java.

- Passes the request envelope to the skill object for processing, where the appropriate handler based on the request type is called.
- Serializes the response back into JSON and writes it to the OutputStream, sending it back to the Alexa service for delivery to the user.

• Serialization/Deserialization: Utilizes JacksonSerializer for converting between JSON text and Java objects, ensuring the data exchanged with Alexa is correctly structured.

The FlavorFeedbackHandler acts as a bridge between Alexa and the skill's logic, ensuring that user requests are directed to the correct handlers and that the responses are accurately communicated back to the users through Alexa.

```
public FlavorFeedbackHandler() {
    skill = new StandardSkillBuilder()
            .addRequestHandler(new GetProductHandler())
            .addRequestHandler(new LaunchRequestHandler())
            .addRequestHandler(new AddProductHandler())
            .addRequestHandler(new ExitIntentHandler())
            .addRequestHandler(new ProductFeedbackHandler())
            .addRequestHandler(new BeverageCategoryHandler())
            .build();
    serializer = new JacksonSerializer();
}
```

Fig 4.10

In Figure 4.10 we see how all intents utilized by the skill are collected and initialized within the main intent, this informs the skill of their location and allows them to be accessed from within the Alexa skill.

4.2.2 Launch Intent

Purpose:
The main purpose of the LaunchRequestHandler is to set the tone of the interaction and inform the user about what they can do within the skill. It's the first point of contact between your skill and the user, making it crucial for establishing context, offering instructions, or suggesting possible commands the user can say.

Functionality:

- **canHandle Method:** This method overrides the canHandle method of the RequestHandler interface. It checks whether the incoming request is a LaunchRequest, which occurs when a user opens the skill without specifying an intent. This is determined by using the Predicates.requestType method with com.amazon.ask.model.LaunchRequest.class as its parameter. If the request matches, this handler is selected to process the request.
- • handle Method: Upon confirmation that the handler can process the request, the handle method is executed. It performs the following actions:
  o Generate a Welcome Message: A string variable speechText is defined with a message welcoming the user to the "Flavor Feedback" skill and offering assistance. This message is the first verbal interaction the user will have with the skill, so it's crafted to be both welcoming and

informative.

o Create a Response: Utilizes the ResponseBuilder to construct the response that will be voiced back to the user by Alexa. The response includes:
- The welcome message (withSpeech), which Alexa will read aloud to the user.
- A directive to keep the session open (withShouldEndSession(false)), allowing the user to respond with further commands or queries without having to re-invoke the skill.

This handler is essential for providing a smooth entry point into thr skill, making users feel welcomed and guiding them on how to proceed with their interaction, which can enhance user experience and engagement.

```java
@Override
public boolean canHandle(HandlerInput handlerInput) {
    return handlerInput.matches(Predicates.requestType(com.amazon.ask.model.LaunchRequest.class));
}

@Override
public Optional<Response> handle(HandlerInput handlerInput) {
    String speechText = "Welcome to Flavor Feedback. How can I assist you?";
    return new ResponseBuilder()
            .withSpeech(speechText)
            .withShouldEndSession(false)
            .build();
}
```

Fig 4.11

Figure 4.11 shows how the launch intent is triggered. In this case, I wanted this intent to be automatically triggered on startup, which is why I've specified *'Predicates.request()'* take the *LaunchRequest.class* as a parameter, which is the class called automatically when a skill starts.

4.2.3 Get Product

Purpose:
The primary purpose of the GetProductHandler is to retrieve a list of products from a DynamoDB table and communicate this list to the user. This interaction enables users to inquire about available products through the skill, making it a critical component for skills that involve inventory or product information retrieval.

Functionality:

- Initialization: o The constructor initializes an AmazonDynamoDB client and a DynamoDB instance. These are used to interact with the DynamoDB service, allowing the handler to query the database.
- Intent Handling: o The canHandle method checks if the incoming request is for the "GetProductIntent", making this handler responsible for processing such requests.
  o The handle method orchestrates the retrieval of product information from DynamoDB and constructs a voice response for the user. This process involves:
  - Invoking getProductsFromDynamoDB to scan the DynamoDB table named "products1" and collect product names.
  - Joining the product names into a single string, creating a speech response listing all products.
  - Generating a reprompt text to encourage further interaction from the user.
  - Using getResponseBuilder to compile the speech text, reprompt text, and session management directive into a response object.

DynamoDB Interaction:

- getProductsFromDynamoDB performs a table scan operation on "products1", iterating through the scan results to extract the "name" attribute of each product item.
- Extracted product names are added to a list, which is then returned and used to create the voice response for the user.

This handler enriches the user experience by enabling real-time access to product information, making the skill more interactive and helpful for users seeking details about available products.

```
public GetProductHandler() {
    client = AmazonDynamoDBClientBuilder.standard().build();
    dynamoDB = new DynamoDB(client);
}
```

Fig 4.12.1

```
@Override
public boolean canHandle(HandlerInput handlerInput) {
    return handlerInput.matches(Predicates.intentName("GetProductIntent"));
}
```

Fig 4.12.2

```
@Override
public Optional<Response> handle(HandlerInput handlerInput) {
    List<String> products = getProductsFromDynamoDB();
    String speechText = "Here is the list of products: " + String.join( delimiter: ", ", products);
    String repromptText = "Is there anything else I can help you with?";
    return handlerInput.getResponseBuilder()
            .withSpeech(speechText)
            .withReprompt(repromptText)
            .withShouldEndSession(false)
            .build();
}
```

Fig 4.12.3

The above code snippets show how the link to the database is established, how the intent waits to be triggered by an intent in the developer console called *GetProductIntent*, and how the response is processed and returned to the user while keeping the session open for further requests (*.withShouldEndSession(false)*).

4.2.4 Add Product

Purpose:
The handler aims to facilitate the addition of products to a specified DynamoDB table via voice commands. It caters to users wishing to input product details such as name, price, year of manufacture, and category. By validating user input and updating the database accordingly, it enhances the skill's interactivity and utility.
Functionality:

- **Initialization:** Initializes an Amazon DynamoDB client and a DynamoDB instance for interacting with the database. It sets a specific table name ("products1") for operations and defines a set of valid categories to ensure data consistency.
- **Intent Handling:**
  o canHandle: Determines if this handler should process a given request based on the intent name, specifically looking for "AddProductIntent".

o handle: Manages the logic for adding a product, which includes:
- Slot Validation: Checks if all required slots (product information fields) are filled. If any slot is missing, it delegates the task back to Alexa to prompt the user for that information. - Category Validation: Verifies if the product's category provided by the user is within a predefined set of valid categories. - Database Update: If validation passes, it proceeds to add the product details to the DynamoDB table. This involves generating a unique ID for the new product and setting its attributes (name, price, year of manufacture, and category).

- **Database Interaction:**
  o The addProductToDatabase private method encapsulates the logic for creating a new product record in the DynamoDB table. It constructs an Item with the necessary attributes and uses the putItem method to insert it into the table.

This handler plays a critical role in making the skill interactive and functional, allowing users to contribute data to your application seamlessly through voice commands. By integrating with DynamoDB, it also demonstrates how Alexa skills can be connected with other AWS services for more complex applications.

```java
// Check if all slots are filled
boolean allSlotsFilled = intent.getSlots().values().stream()
        .allMatch(slot -> slot.getValue() != null);

if (!allSlotsFilled) {
    // Not all slots are filled. Delegate slot collection back to Alexa.
    return handlerInput.getResponseBuilder()
            .addDelegateDirective(intent) // Delegate back to Alexa to ask for the next slot
            .build();
}

// Extract values from slots
String productName = intent.getSlots().get("ProductName").getValue();
String price = intent.getSlots().get("Price").getValue();
String yearOfManufacture = intent.getSlots().get("YearOfManufacture").getValue();
String category = intent.getSlots().get("category").getValue();

// Validate category
if (!validCategories.contains(category)) {
    String speechText = "Invalid product category. Please choose from " +
            String.join( delimiter: ", ",  ...elements: "" + validCategories) + ".";
    return handlerInput.getResponseBuilder()
            .withSpeech(speechText)
            .withShouldEndSession(false)
            .build();
```

Fig 4.13.1

```
try {
    Table table = dynamoDB.getTable(tableName);
    String productId = UUID.randomUUID().toString();

    Item item = new Item()
            .withPrimaryKey( hashKeyName: "product_id", productId)
            .withString( attrName: "name", productName)
            .withString( attrName: "price", price)
            .withString( attrName: "year_of_manufacture", yearOfManufacture)
            .withString( attrName: "category", category); // Add category to item

    table.putItem(item);
    return true;
} catch (Exception e) {
    e.printStackTrace();
    return false;
}
```

Fig 4.13.2

The above code snippets show how I've set up the filling of slots, each of which is required to progress further into the intent, and only once all of them are filled satisfactorily is the item added to the database.
It's also worth noting here that each of these slots must be defined, marked are required, and given dialogue for Alexa to prompt the user with within the developer console itself for any of this code to be effective. Without these developer console steps, Alexa will simply skip over all slots and throw an error.

4.2.5 Add Feedback

Purpose:
This handler is tasked with processing user feedback about specific products, particularly focusing on the product category. It interacts with DynamoDB to retrieve and temporarily store product category information based on user input, facilitating a confirmation process with the user.
Functionality:

- **Initialization:**
  o The constructor initializes an Amazon DynamoDB client and assigns it to a DynamoDB instance. This setup allows the handler to perform database operations, such as reading from and writing to DynamoDB tables.

- **Intent Handling:**
  o canHandle: This method checks if the incoming request matches the "ProductFeedbackIntent". It ensures that this handler processes only requests intended for providing feedback on products.
  o handle: Executes the core logic when the intent is triggered. It performs several steps:
  - Retrieves the product name from the intent's slots.
  - Calls getProductCategory to look up the category of the product in a DynamoDB table ("products1"), based on the product name.
  - Invokes saveProductCategoryForConfirmation to temporarily store the product name and its category in a different DynamoDB table ("TempProductCategory") for later confirmation. This step involves creating a unique ID for the entry, typically using UUID.randomUUID().
  - Constructs a voice response asking the user to confirm the product category, guiding them on how to provide the confirmation.

DynamoDB Interaction:

- savProductCategoryForConfirmation: o This method stores the product name and category in a temporary table. It's designed to facilitate a confirmation process, allowing the skill to remember the context of the user's request across multiple interactions.
- getProductCategory: o Retrieves the category of a product from the primary product table by querying for the product name. It's crucial for ensuring that the feedback pertains to a known category, even defaulting to "unknown" if the product isn't found.

This handler is a key component of the skill's functionality, bridging the gap between user interactions and database operations. It not only facilitates a dynamic feedback loop with the user but also demonstrates how Alexa skills can integrate with AWS services for a more interactive and responsive user experience.

```
IntentRequest intentRequest = (IntentRequest) handlerInput.getRequestEnvelope().getRequest();
String productName = intentRequest.getIntent().getSlots().get("name").getValue();
String category = getProductCategory(productName);

// Save product name and category to a temporary DynamoDB table for confirmation
saveProductCategoryForConfirmation(productName, category);

// Ask the user to confirm the category
String confirmationPrompt = "Please confirm the category by saying, for example, 'beverage', or 'sna
return handlerInput.getResponseBuilder()
        .withSpeech(confirmationPrompt)
        .withShouldEndSession(false)
        .build();
```

Fig 4.14

The *ProductFeedbackHandler* is an interesting intent. Due to the inability to trigger an intent from within another intent, I opted for the work-around shown here. I get the name of the product, add it to a temporary database, then ask the user to specify the category of the product while closing the skill. The product category is then used as an utterance to trigger the category-specific feedback intent.
This is a far from ideal solution, however, until Amazon supports dynamic slot definitions or calling intents from other intents, this is the best solution I could find which allowed me to tailor feedback to a specific category.

4.2.6 Beverage Feedback Category

Purpose:
This handler aims to collect and store user feedback on beverages by asking users to provide detailed descriptions. It plays a crucial role in gathering insights into user preferences and experiences with different beverages, thereby enhancing the skill's ability to recommend or improve offerings.
Functionality:

- **Initialization:**
  o Initializes an Amazon DynamoDB client and a DynamoDB instance, setting the foundation for database operations.
  o Defines two table names: feedbackTableName for storing feedback and tempStorageTableName for temporarily holding product category information before confirmation.
- **Intent Handling:**
  o canHandle: Determines if this handler is the right one for processing a request based on the intent name, specifically looking for "BeverageCategoryIntent".
  o handle: Orchestrates the feedback collection process. Key steps include:
  - Retrieving the most recent product category information from a temporary storage table to ensure the feedback pertains to the correct product.

- Checking if all required slots (texture, bouquet, and aftertaste) are filled. If not, it delegates the task back to Alexa to prompt the user for this information.
- Once all slots are filled, it extracts the slot values and passes them along with the product name to storeFeedback for storing in the DynamoDB table.
- Deletes the temporary product category item from the temporary storage table to maintain cleanliness and prevent confusion in future interactions.
- Generates a speech response to inform the user whether their feedback was recorded successfully or if there was an error.

Database Interaction:

- **storeFeedback:**
  o This method encapsulates the logic for storing feedback in the DynamoDB table specified by feedbackTableName. It constructs an Item with the product name as the primary key and the feedback details (texture, bouquet, aftertaste) as attributes.
  o Inserts the item into the table using the putItem method, indicating a successful operation by returning true.

This handler not only demonstrates how to integrate Alexa skills with DynamoDB for persistent storage but also showcases how to manage session state across multiple interactions. It's a vital component for skills aiming to engage users in detailed feedback collection, enhancing the interactive experience and providing valuable data for service improvement or personalization.

```
// Extract values from slots
String texture = intent.getSlots().get("Texture").getValue();
String bouquet = intent.getSlots().get("Bouquet").getValue();
String aftertase = intent.getSlots().get("Aftertase").getValue();

boolean success = storeFeedback(productName, texture, bouquet, aftertase);

String speechText = success ? "Feedback recorded successfully." :
        "Failed to record feedback. Please try again later.";
String repromptText = "Is there anything else I can help you with?";
```

Fig 4.15

In figure 4.15, the slots for the beverage category are specified, filled, and sent back to Alexa. An error is sent if the was a problem filling the slot (for example, an incompatible data type), and the user is re-prompted by the skill asking if there is anything else if can help the user with.

4.3 Technical Review

Having worked with Amazon's Web Services and cloud computing for the past few months, I feel I'm in a good position to share my thoughts on the current state of the environment with regards to Java development.

During development, I encountered a number of errors which seemed to have no reasoning behind them. After some time trawling stack-overflow and other resources, with the help of my supervisor Alex, I came to the conclusion that Alexa was not accepting certain versions of dependencies. While the dependency itself was required to resolve an error or to achieve some functionality, more often than not the newest versions of dependencies were not compatible with my skill. This combined with the need for some dependencies (namely Log4j) to have their own xml files, is an example of teething problems where, given time and resources, these are simple oversights for Amazon to resolve, however they do pose frustrating issues for anyone developing in their environment currently.

While a lot of Amazon's built-in frameworks and services are very intuitive, they also leave a lot to be desired when trying to optimize certain scenarios. One example of this is how intents are called within apps. When providing feedback in my app, I wanted the user to be redirected to a category-specific intent which would contain all relative slots for the product the user was trying to review. In trying to implement this, I came to the frustrating conclusion that this wasn't going to be possible due to Amazon's insistence regarding intents being triggered only by user utterances.

Another instance of Amazon's services still exhibiting signs of growing pains is their lack of support for string values in recording user responses. My app requires a lot of string values to be recorded from the user, and, in preparation for this, I set the slot value to be 'AlphaNumeric'. However, when running the app, I experienced crashes any time I would try to put a non-numeric value into this slot. I instead had to set the slot type to 'Query', which is the only slot type provided by Amazon which can actually handle string values.

Overall, my experience with AWS has been a positive one, not without its fair share of obstacles to overcome. While the industry is still very clearly experiencing some growing pains (two of which I highlighted above), this is to be expected. Instead of looking at the challenges facing the day-to-day development of applications here, I would prefer to look to the bigger picture. My short few months developing within this ecosystem has given me a glimpse at the potential of the industry and the ease with which people can now create incredibly powerful applications with the potential to reach an indeterminate amount of users. While there have definitely been times where I have been frustrated, not only at my own code, but with the Amazon-provided technologies I've been utilizing, I would still recommend AWS and cloud computing to anyone with an interest in the field as I feel with more input from creative developers the area has incredible growth potential and can provide a service to many customers who don't yet realize how beneficial it could be to them.

4.4 Project Review

From very early on in the development phase of this project, it was clear to me that my original plan for sprints and user-stories wouldn't be feasible. This was largely due to the numerous new technologies I needed to learn and understand, even using familiar ones where possible (e.g. writing code in Java, using the Intellij IDE, attempting to use Google Firestore). Instead, I focused on the main functionality of the app, namely:

- Listing products
- Adding products
- Providing feedback on products

Focusing on these sectors meant that features like custom recommendations based on a user's previous reviews and submission to Amazon for publication wouldn't be able to be implemented within the tight time constraints.

While this project is lacking some features which I intended to be present at this point, they are all still very much achievable given slightly more time. With three weeks remaining until this project at the time of writing, I intend to have a basic admin interface functional along with the beginnings of (or possible some functional) authentication system. I believe these goals are attainable thanks to the services provided by and built into Amazon Web Services.

5 Conclusion and Future Directions

5.1 Summary of Findings

Enhanced Customer Engagement:
The implementation of the Alexa chatbot in F&B establishments led to improved customer engagement. Customers appreciated the convenience of providing feedback and making inquiries through voice commands, leading to increased interaction with the chatbot.

Convenience and Accessibility:
The voice-activated nature of Alexa chatbots made it easy for customers to share their thoughts and preferences without the need to type or navigate menus. This convenience resulted in a higher volume of feedback, as customers were more willing to participate.

5.2 Contributions of the Chatbot

Improved Customer Engagement:
The research highlights the effectiveness of Alexa chatbots in enhancing customer engagement. By offering a voice-activated and convenient channel for providing feedback, customers are more willing to participate in the feedback process. This leads to higher engagement rates, providing F&B establishments with a more comprehensive understanding of customer preferences and experiences.

Prompt Feedback Collection:
Alexa chatbots enable prompt feedback collection immediately after a customer's dining experience. This real-time approach results in fresher and more accurate feedback, allowing businesses to address issues and make improvements more swiftly. Faster responses to feedback contribute to enhanced customer satisfaction.

Enhanced Data Analysis:
The data collected through this Alexa chatbot can be easier analyzed allowing for prompt responses by vendors

5.3 Implications for the Food and Beverage Industry

Competitive Advantage:
Early adoption of chatbot technology can give F&B businesses a competitive edge by offering a more modern and convenient customer experience, which can attract and retain a tech-savvy customer base.

24/7 Availability:
Chatbots are available 24/7, allowing customers to engage with F&B establishments at any time. This can increase accessibility and accommodate late-night or off-hours orders and inquiries.

Feedback-Driven Improvements:
Chatbots actively collect feedback, which can be used to make targeted improvements. By addressing customer concerns and acting on their suggestions, businesses can continuously enhance the dining

experience.

Enhanced Customer Engagement:
Interactive chatbots engage customers and encourage them to participate in feedback collection and surveys. This results in higher engagement rates, offering businesses a deeper understanding of customer preferences and sentiments.

Improved Service Quality:
Chatbots can enhance service quality by ensuring prompt and accurate responses to customer inquiries and orders. This leads to a smoother and more efficient customer experience, with fewer errors and delays in order processing.

Increased Customer Loyalty:
Chatbots that provide personalized experiences and remember customer preferences can foster stronger customer loyalty. Customers are more likely to return to establishments where they feel recognized and appreciated.

5.4 Limitations and Challenges

Technical Constraints: The effectiveness of chatbot implementation relies on the quality of underlying technologies, such as natural language processing and voice recognition. Technical limitations, including the accuracy of these technologies, can affect the performance of chatbots.

Data Privacy and Security: The research acknowledged concerns related to data privacy and security when using chatbots to collect customer feedback. Addressing these concerns is critical, as mishandling customer data can result in trust issues and regulatory challenges.

Complex Interactions: Chatbots may struggle to handle highly complex or nuanced interactions, as they are typically rule-based or rely on predefined responses. Adapting chatbots to address intricate customer queries or scenarios can be challenging.

Integration with Existing Systems: The integration of chatbots with existing F&B systems can be complex and costly. Compatibility issues, data transfer, and system updates may pose challenges for some establishments.

Technology Evolution: The fast-paced evolution of technology, including AI and chatbot development, can make it challenging for F&B businesses to keep up with the latest advancements and ensure their chatbots remain competitive.

Customer Education: Educating customers about chatbot availability, usage, and benefits is crucial. Ensuring that customers understand how to interact with chatbots and the value they provide can be challenging.

5.5 Future Directions and Recommendations

Complex Interaction Handling: Investigating methods to enhance chatbots' ability to handle complex and nuanced interactions is crucial. This may involve developing more advanced NLP capabilities and integrating AI to understand and respond to diverse customer queries.

Multilingual Chatbots: Research can focus on developing chatbots that support a wide range of languages to cater to diverse customer bases. Multilingual chatbots can help in global F&B settings and tourist destinations.

Voice Recognition Advancements: Future research should examine improvements in voice recognition technology to ensure that chatbots accurately understand and respond to voice commands, regardless of accents or speech patterns.

5.6 Retrospective

I began this project with very limited knowledge of Alexa chatbots. Through trial and error, research papers, and the advice of my supervisor Alex Vakaloudis, I have investigated Alexa hosted chatbots on the alexa developer console, as well as running chatbots locally from a personal machine using services like NGROK. I then moved on to using the Alexa SDK in java, which was an unfamiliar process for me, but with use of Visual Studio Code I could clearly see how each part of the Java chatbot was working. This technical experience combined with elements like the database design and research have provided a rewarding experience in this phase of the project, and I am looking forward to jumping into development.

6 References

Hospitality Feedback System 4.0: Digitalization of Feedback System with Integration of Industry 4.0 Enabling Technologies
- Ram Narayan

"How was Your Stay?": Exploring the Use of Robots for Gathering Customer Feedback in the Hospitality Industry
- M. Chung, M. Cakmak

Comparing practices for capturing bank customer feedback - Internet versus traditional banking
- J. Wisner, William J. Corney

The Impact of Service Quality on Customer Satisfaction
- S. Gilaninia, M. Taleghani, Mohammad Reza Khorshidi Talemi

Measuring Customer Satisfaction for F&B Chains in Pune Using ACSI Model
- Vinit Dani

Restaurant Quality and Customer Satisfaction
- Bader M. A. Almohaimmeed

Evaluation of the Naturalness of Chatbot Applications
- A. Atiyah, S. Jusoh, Firas Alghanim

Challenges of Building an Intelligent Chatbot
- A. Chizhik, Yulia Zherebtsova

Emerging Technologies of Natural Language-Enabled Chatbots: A Review and Trend Forecast Using Intelligent Ontology Extraction and Patent Analytics
- M.-H. Chao, A. Trappey, Chunwang Wu

Natural Language Processing
- A. Joshi
B. NLP for Chatbot Application
- S. Nithyanandam, Sharmila Kasinathan, Devi Radhakrishnan

Natural Language Processing
- V. Daniel Hunt

Towards an efficient voice-based chatbot
- J. Quintero, R. Asprilla

Improving user trust towards conversational chatbot interfaces with voice output
- Ramón Burri

A Literature Survey of Recent Advances in Chatbots
- Guendalina Caldarini, Sardar F. Jaf, K. McGarry

Conversational Agent Research Toolkit: An alternative for creating and managing chatbots for experimental research
- T. Araujo

Introduction to Microsoft Bot, RASA, and Google Dialogflow
- Abhishek Singh, Karthik Ramasubramanian, Shrey Shivam

Review of State-of-the-Art Design Techniques for Chatbots
- Ritu Agarwal, Mani Wadhwa

Survey on Chatbot Design Techniques in Speech Conversation Systems
- S. Abdul-Kader, John Woods

Modern Chatbot Systems: A Technical Review
- Abbas Saliimi Lokman, Mohamed Ariff Ameedeen

Comparing Data from Chatbot and Web Surveys: Effects of Platform and Conversational Style on Survey Response Quality
- Soomin Kim, Joonhwan Lee, G. Gweon

Evaluation of Modern Tools for an OMSCS Advisor Chatbot
- Eric Gregori

Challenges of Building an Intelligent Chatbot
- A Chizhik, Y Zherebtsova

Building a Chatbot: Architecture Models and Text Vectorization Methods
- AV Chizhik, YA Zherebtsova

Modern Chatbot Systems: A Technical Review
- Abbas Saliimi Lokman, Mohamed Ariff Ameedeen

Evaluation of Modern Tools for an OMSCS Advisor Chatbot
- Eric Gregori

The media inequality: Comparing the initial human-human and human-AI social interactions
- Y Mou, K Xu