Implementing a Low-Code AI-Based Chatbot on Azure combined with Boomi Integration

Redeat Haile
Haaga-Helia University of Applied Sciences
Bachelor's Thesis
2022
Bachelor of Business Information Technology

Abstract

Author(s)
Redeat Haile

Degree
Bachelor of Business Information Technology

Report/thesis title
Implementing a Low-Code AI-Based Chatbot on Azure combined with Boomi Integration

Number of pages and appendix pages
59 + 2

The research discusses the background, importance and different types of a chatbot. It also covers a low-code chatbot implementation's theoretical and empirical parts. Low-code platforms are becoming increasingly popular for building applications, including chatbots. For example, using a low-code platform, developers can quickly and easily create chatbots without writing a large amount of complex code.

Different platforms are available for building a low-code chatbot. The author shows how easily a low-code chatbot can be developed using Microsft Azure Bot Framework Composer and Boomi Integration. The two low-code platforms are a perfect combination for chatbot development. Microsoft Bot Framework Composer creates the UI for the chatbot, and Boomi Integration fetches the data from a database.

In the research implementation, Boomi integration uses free data available in a cloud database and creates an API. On the other hand, Bot Framework Composer uses the API to access the data and provide the information a user needs. An interaction between a bank and its customer is used as a case for the implementation.

The implementation doesn't include data creation in a database; instead, it uses available data in the cloud database. Also, the chatbot developed is not published on any website or messaging platform.

The final result of the implementation is tested in Microsoft Bot Framework Emulator and works perfectly. The whole chatbot development process shows how easy and flexible it’s to use a low-code chatbot building method. Therefore, this method can be beneficial for creating chatbots focused on specific tasks, such as answering frequently asked questions or customer support.

Keywords
Artificial Intelligence (AI), Application Interface (API), Natural Language Processing (NLP)

Table of contents

Table of Figures

1 Introduction

Chatbots are advanced computer programs that imitate human conversations in their standard form (Bala & al., 2017, 4). Chatbots are utilized to process input provided by the user and generate an outcome. Generally, a chat about accepting Natural Language text as the input and the result ought to be the most significant outcome of the client's input sentence. (Kumar & Ali 2020, 11.) Therefore, Chatbots could also be characterized as online human-computer conversational systems with Natural Language. A chatbot comprises, thus, an automatic dialogue framework that can handle hundreds of possible users simultaneously.

Chatbots are presently used in multiple areas and applications, such as education, medical care, banking, e-commerce, entertainment, etc. Accordingly, chatbots can offer clients help in various fields and entertainment. Those offers could be ideal for small talk-designed chatbots, like Jessie Humani and Mitsuku, which typically provide users with a feeling of social connection. (Okuda & Shoda 2018, 5.) Chatbots seem more attractive to users than a site's traditional FAQ (Frequently Asked Questions) webpage. Simultaneously, a chatbot could help numerous users. Developing a chatbot is subsequently more profitable and affordable than customer support services. As well as supporting and providing help to consumers, chatbots could be utilized for entertainment purposes by the end-users. Chatbot programs have become extremely common in the past few years, with the modification and the rise of computational capacity and sharing of open-source systems and innovations.

New advancements in AI (Artificial Intelligence) and NLP (Natural Language Processing) methods have helped make implementing chatbots simpler, more adaptable to application and sustainability, and progressively able to imitate human speech. Nonetheless, human-chatbot interactions could be more precise. A few areas for improvement include understanding a conversation logically and emotionally and gender discrimination. Furthermore, a chatbot is less capable of understanding the conversational setting and emotional and verbal signals or signs than human beings, which impacts their capability to discuss in a seriously engaging and friendly way.

Implementing an Artificial Intelligence chatbot could assist organizations in interacting with consumers who have questions about a website or require information regarding the products or services they offer. Most consumers have the same questions regarding the infor-

mation they need while purchasing a particular product or service. Using an Artificial Intelligence chat makes it easier to provide the information required as quickly as possible and simultaneously precisely. (Buntak & al., 2017, 406.)

With the assistance of cloud computing, it is feasible to gather users' and organizations' questions and answers in one place. Cloud computing lets users and organizations ask questions and find solutions immediately from where they are utilizing their personal computers, tablets, or smartphones. Currently, most organizations are experiencing issues responding to client questions and requests. The task is difficult in any way. Moreover, it requires a considerable workforce.

That is why organizations provide an AI chatbot which prompts questions to communicate with users and gives back a response with the required info or provide an AI chatbot to answer Frequently Asked Questions web page to reduce the workload. Due to technological advancements, organizations adopt a better technique to handle these problems by replacing humans with robots to answer clients' questions.

Deploying an Artificial Intelligence chatbot will assist any organization in interacting with consumers quickly and provide the required information regarding a specific product or service. Organizations can utilize Artificial Intelligence chatbots on their website, email, and messaging platform. There are several benefits an organization could get from implementing an Artificial Intelligence chatbot, such as providing customer service 24/7, increasing customer satisfaction, decreasing the workforce, etc.

1.1 Objective

The project aims to create a low-code Artificial Intelligence chatbot by combining Microsoft Azure Bot Framework Composer and Boomi API Management. Microsoft Azure Bot Framework Composer will be utilized to develop the chatbot, and we will use Boomi API Management for machine learning and data mining.

The data for the chatbot will be fetched from a cloud SQL database using Boomi. Microsoft Azure Bot Framework Composer will request the data from the cloud SQL database using the API created by Boomi. Therefore, Boomi API Management will be essential in transferring the data from the database to the Azure chatbot.

1.2 Limitation

One of the project limitations was a lack of available data on the cloud for the chatbot. Most of the data found were accessible only with a unique API key. Unfortunately, using the API key didn't let the researcher practice the API-creating process in different applications. Platform KNIME was the first option to try creating RESTful API, but the server was too big for a personal computer to download and accessing the server from outside was not an option.

1.3 Scope

The implementation will cover a basic chatbot implementation, but due to a lack of time and resources, the researcher could not incorporate the following:

- The data on the database will not be created from scratch.
- The published chatbot will not be used on any messaging platform or website.

2 Literature review

2.1 The rationale for the research

The researcher had always been interested in working in the Artificial Intelligence area and discovered that Artificial Intelligence chatbots are a hot commodity presently comprising a valuable area of study for Artificial Intelligence. Other than that, to begin a profession as a Quality Assurance Manager would be the most effective way to use this research opportunity on probably the most recent advancements that already exist in the market and have enhanced abilities. Researching Artificial Intelligence chatbots will cover a few aspects: Artificial Intelligence chatbot advancement procedure, technologies, constraints, conceivable outcomes, and various kinds of advancement strategies. To acquire a deep understanding of these aspects, the researcher decided to go on and study the Artificial Intelligence chatbot.

2.2 Research Problems

Within this study, the researcher faced the following issues while creating and implementing an Artificial Intelligence chatbot:

- In some cases, chatbots have similar answers to different questions, and solutions to a few are not answered according to the client's expectations.
- If hackers gain access to the chatbot, they act like a customer organization to start discussions with the internal business faculty.
- Chatbots behaves like a mechanical robot. It is programmed by computer programmers and can only answer questions if the discussion flows in an anticipated manner.

Natural Languages are continuously changing and very inconsistent or unsteady. Recent Natural Language processing codes might try to catalogue the various principles of Natural Languages. However, Natural Languages can be replaced by new principles or punctuations, which later become another language.

2.3 Key Terminologies

An Artificial Intelligence chatbot is a framework that can comprehend and answer verbal inputs to a relatively broader client level. Presently chatbot is quite possibly the most accessible way of connecting without wasting too much time going through the web pages of

a complex site. A few organizations have even begun utilizing robots to provide customer service to their consumers. For any chatbot, the most significant terminologies are:

- **Utterance** – It refers to the whole message that a client says.
- **Intent** – It implies the client's intention.
- **Conversational User Interface** – UI (User Interface) permits a client to interact with the chatbot. Conversational user interfaces include buttons, text boxes, links, and graphic components.
- **ASR (Automatic Speech Recognition)** - is a computer technology that recognizes and processes human voices.
- **NLU (Natural Language Understanding)** – Manages machine reading understanding and acts as a section of natural-language processing in AI.
- **NLP (Natural Language Processing)** UI (User Interface) permits a client to interact with the chatbot. Conversational user interfaces include buttons, text boxes, links, and graphic components.
- **TTS (Text-To-Speech)** – This assistive program reads digital text aloud.
- **API (Application Programming Interface)** – UI (User Interface) permits a client to interact with the chatbot. Conversational user interfaces include buttons, text boxes, links, and graphic components.

2.4 Hypothetical frameworks

To create an Artificial Intelligence chatbot, the researcher went through a few studies regarding a simple chat application web-based and Artificial Intelligence and faced discovered the following hypothetical frameworks:

- **ASR (Automatic Speech Recognition)** is a technology that converts words spoken by the user into text. With the help of Automatic Speech Recognition, voice technology can distinguish sounds and identify them as messages. Automatic Speech Recognition is the foundation of the whole voice experience, enabling computer systems to eventually comprehend a standard form of communication, speech. (Viraktamath & al., 2016, 49.)
- **NLU (Natural Language Understanding)** – Understanding the structure and significance of human language through computers by permitting clients to interact with computers using regular sentences. Natural Language Understanding is AI that utilizes programs to interpret unstructured information. It could also process

part of a text, convert it into a language only a computer understands, and develop a language people could comprehend. (Loper & Bird 25 March 2022.)

- **Text-to-Speech** – It is an assistive computer program that reads digital texts aloud. It is usually referred to as "read aloud' technology. Text-to-Speech could take words on a PC or other advanced gadget and transform them into sound by clicking a button. It is beneficial for youngsters who find it difficult to read. However, it could also assist children with proofreading and writing. (Choi & Nam 2019, 1694.)
- **Client-server model** – This is a distributed app system that splits tasks or amounts of work among the suppliers of an asset or service, referred to as servers and individuals who request these services, known as clients. Usually, servers and clients communicate using a computer network on different hardware. However, the server and the client might be on the same computer system. The server host runs at least one server program that helps share its resources with clients. On the other hand, the client cannot share their resources; however, it can request service or data from the server. Hence, clients start communicating with the servers, anticipating requests. (Choi & Nam 2019, 1694.)
- **JavaScript** is a programming language that enables an individual to execute complicated features on a website. JavaScript is concerned with showing content updates, animated two and three-dimensional illustrations, etc. (Kim & al., 2018, 24).
- **Python** – This is a computer programming language shining for Artificial Intelligence, data science, automating systems, and developing Application Programming Interfaces. (Kim & al., 2018, 24.)
- **Google Assistant API** - This is an Application Programming Interface of Artificial Intelligence assistance created by Google that permits a computer to register or update a gadget manually by utilizing a JSON document and the rest of the API. It offers a method for defining and writing the model of a device or gadget. Developers working at Google provide Application Programming Interfaces. These tools can be used to develop software and technical resources that enable organizations to communicate with Google Services and other services. Google's Assistant SDK (Software Development Kit) service exposes the user to a low-level Application Programming Interface. Exposing users to a low-level API allows them to alter directly or control the sound bites of an assistant request, and respond to languages such as C++, Go, Node.js, and Java on each platform that supports **gRPC**. (Kim & al., 2018, 24.)

2.5 Background information on Chatbots

Even though the mission of having the option to make something that could communicate and comprehend with its maker has profound roots in ancient history, Alan Turing is believed to be the first individual who conceptualized the concept of a chatbot back in 1950. When he formulated the question that "Cab machines think?" Turing's explanation regarding the behaviour of an intelligent machine prompts the generally interpreted idea of a chatbot (Turing 1950, 433-460). Chatbots have advanced with the progressive increase in computational abilities and advancement in NLP devices and procedures. The first chatbot was implemented in 1966, depended substantially on language and pattern-matching standards and was developed with the help of ELIZA. First, it would communicate with the client using a program that matches keywords. Then, it looks for a suitable conversion regulation to formulate the input again and produce a result, for example, answering the user's question. ELIZA was a robust framework that facilitated further exploration within the field. However, ELIZA's extent of information was restricted since it relied upon identifying minimal context, and typically, rules for matching patterns are not adaptable to be effectively executed within new domains.

A substantial advancement in chatbots during the 1980s is the utilization of AI ALICE (Artificial Intelligent Internet Computer Entity) depending on the AIML (Artificial Intelligence Mark-up Language), which is also an extension of XML (Extensible Mark-up Language). It was created mainly to dialogue pattern knowledge that can be included in the Artificial Intelligent Internet Computer Entity computer program to increase its information base. Artificial Intelligence Mark-up Language data objects are made up of topics and classes. Classes are the fundamental unit of information, including a regulation to match the user's input to the output produced by the chatbot. The user's input is defined by the rule patterns, while the work produced by the chatbot is represented by the rule format, Artificial Intelligent Internet Computer Entity information base. The new data objects in the Artificial Intelligence Mark-up Language significantly improved past pattern matching frameworks because the information base was easy to expand. Moreover, **ChatScript**, the replacement or substitute for Artificial Intelligence Mark-up Language, was also the base innovation behind other successful chatbots. (AbuShawar & Atwell 2015, 4)

The fundamental concept behind this creative innovation was to match the users' input to a topic, and every case has an explicit rule related to it to produce the result. ***ChatScript*** introduced a new era for chatbots' innovation development. It shifted the focus to semantic examination and understanding. (Shum & al. 2018, 10-26.) The primary restriction of depending on regulations and pattern matching in chatbots is that they rely on the domain

that makes them unadoptable. Chatbots count on rules or guidelines written manually for explicit spaces. With the advancements in Artificial Intelligence methods and NLP tools along with the accessibility of computational power, new structures and programs were developed to execute intelligent chatbots without depending on regulations and pattern-matching methods and facilitated the commercial usage of chatbots. The utilization of Artificial Intelligence programs in chatbots has been researched, and new structures of chatbots have evolved.

The use of chatbots has increased with the development of Deep Learning programs. One of the new and the most intriguing app is the advancement of intelligent personal assistants, like Google Assistant, Siri, Alexa, Cortana, and Watson. A personal assistant chatbot or a conversational agent can communicate with the client through voice, generally incorporated in mobile phones, smartwatches, home screens, loudspeakers, and cars. For instance, intelligent personal gadgets get activated when the client speaks a word or sentence by listening. Through NLU, the personal assistant could understand the command and answer the client's request, typically by providing information. For example, Siri, how is the climate today in New York? In New York, the weather is sunny, and it is 70°F, or by finishing different tasks. For example, "OK, Alexa, play my evening music playlist." However, comprehending human language has shown to be very difficult due to local, regional, tonal and individual variations in how a user speaks.

Every intelligent personal assistant presents similar critical elements regarding technologies utilized, UI, and functionalities. A few chatbots have a more advanced personality than others, and the most evolved ones are for entertainment purposes and not simply to help with everyday tasks. These chatbots are alluded to as social, such as Microsoft's **XiaoIce**. **XiaoIce** is intended to be long-term assistance to the client and to accomplish high client engagement, have been created to have a unique personality, which is IQ (Intelligent Quotient) and an EQ (Emotional Quotient). Information and memory models, picture and understanding of the English language, generation and forecast are examples of Intelligent Quotient abilities. These are crucial elements of the advancement of dialogue capabilities. These capabilities are needed in social chatbots to satisfy or fulfil clients' explicit wants and help them. Core Chat is the most significant and sophisticated capacity, which could engage in long and open discussions with clients. Compassion and social abilities are two essential parts of the Emotional Quotient. The conversational motor of **XiaoIce** utilizes a dialogue administrator to monitor the condition of the discussion and

chooses either the Core Chat (the open-domain Generative element) or the dialogue ability to produce an output or response. Hence, the model includes both, Generative abilities and retrieval of information. (Zhou & al. 2020, 53–93.)

3 Related information

3.1 Different kinds of Chatbots

Chatbots could be categorized by utilizing various parameters, such as the knowledge domain, the provided services, the goals, the response generation technique, the processing of input, the build method and the human aid. The category that depends on the knowledge domain considers the information a chatbot could access or how much data it is trained. For example, open-domain chatbots can discuss venereal topics and respond suitably; on the other hand, closed-domain chatbots are emphasized a specific knowledge domain and could fail to answer different questions. (Champaneria & Nimavat 2017, 1020.)

The category that depends on the provided services considers the chatbot's sentimental area to the client and how many intimidating interactions happen. It is also based on the chatbot's task. An interpersonal chat lies within the area of communication and offers different services, for example, booking seats at a restaurant or on an aeroplane and FAQ bots. There are no companions of the client; instead, they get information and give it to the client. They could have individuality, be friendly and possibly recollect information about the client; however, they are not required to do so. An intrapersonal chatbot exists inside the private domain of the client, for example, char applications such as WhatsApp and Messager. They are allies to the client and comprehend the client as human beings do. An inter-agent chatbot turns out to be ubiquitous, while each chatbot would need some inter-chat communication possibilities. The requirement procedures for inter-chat, not communication, have already arisen, for example, Alexa-Cortana. (Champaneria & Nimavat 2017, 1020.)

The category that depends on the goals considers the fundamental goal a chatbot intends to accomplish. For example, an informative chatbot is created to provide the client with information stored in advance or accessible from fixed sources, such as a FAQ chatbot. A chat-based/conversational chatbot talks to the client, similar to another human, and their objective is to answer accurately to the sentence they have been given. Therefore, a task-based chatbot performs a particular task, like booking a flight seat or helping someone. These chatbots are intelligent in requesting information and comprehending the user's input, such as a FAQ chatbot. (Kucherbaev & al., 2018, 38.)

The category depends on the processing input provided by the user, and the response generation technique considers the technique for processing inputs and creating responses. Three models are utilized to generate suitable responses: rule- and retrieval-based, and generative. (Kucherbaev & al., 2018, 38.) A rule-based model chatbot is the kind of architecture with which most of the first chatbots were created, like various online chatbots. They select the framework response based upon the fixed pre-established set of rules, based on recognizing the linguistic form of the text of the input without producing any new text responses. The information utilized in the chatbot is hand-coded by humans and is organized and introduced with conversational patterns. A more extensive rule database permits the chatbot to respond to more input from the user side.

Nonetheless, this kind of model could be more robust to grammatical errors in the input provided by the user. Most current study on rule-based chatbots researches response choice for single-turn discussion, which considers the last message. Within a more human-like chatbot, multi-turn response choice considers past parts of the discussion to choose a response appropriate to the entire discussion setting. Somewhat different to a rule-based model, a retrieval-based model gives more adaptability as it queries and assesses available resources utilizing Application Program Interfaces. (Kucherbaev & al., 2018, 38.) A retrieval-based chatbot obtains a few response candidates from an index before it refers to the matching method for the response choice.

The generative model produces answers preferably compared to the other three models, especially if it's based on existing and past client messages. Moreover, these chatbots are human-like and utilize Artificial Intelligence algorithms and Deep Learning methods. Nonetheless, there are problems in creating and training them.

3.2 JavaScript and JavaScript structure

JavaScript is the most utilized computer language by web developers. This particular computer language has become extremely popular due to its overpowering use across various web pages and modern Internet browsers. (Abrahamsson & Graziotin 2013, 335.) Some of the crucial features of the JavaScript language is that recent Internet browsers include a built-in interpreter for JavaScript codes that could interpret and execute the programming language. Specifically, JavaScript permits complicated programs directly access DOM (Document Object Model) objects and web browser events. The DOM is a substantial hierarchical object with numerous components that form a tree. In the event of web browsers, it is feasible to discover the components within the Internet browser itself

and on the accessed web page (Mariano 2017, 25), and JavaScript controls the Document Object Model. Frontend web development, or client-side development, is designing and developing the UI (User Interface) and its interactions. (Souders 2018, 38.) A frontend web developer is an individual who has the responsibility of developing the UI. They are accountable for the app functionality, and UX (User Experience) is a major problem for these specialists.

The fundamental set of web development languages consists of HTML (Hyper Text Mark-up Language) to establish the content of the web page, CSS (Cascading Style Sheets) to specify the presentation of the website and lastly, JavaScript to establish the actions of the website. (Myers & Oney 2009, 106.) Internet browsers transform the web pages encoded in Hyper Text Mark-up Language and Cascading Style Sheets into a document that the user can easily understand. Furthermore, JavaScript code is used in various Internet browsers and several machines. These environmental aspects could result in problems when evaluating enhanced JavaScript performance. (Christodoulou & al., 2012, 514.)

JavaScript frameworks have recently become fundamental tools for developing web applications using Agile. They also serve as a structure for creating a web application with single pages, allowing computer programmers to think less about the code structure and maintenance. They could also focus on developing complicated elements and rich interfaces with the help of these frameworks. The benefits of utilizing JavaScript structures are their effectiveness level, security, cost-effectiveness, or cheaper. Some of the most important JavaScript frameworks are AngularJS, Aurelia and Ember. (Mariano 2017, 24.)

3.2.1 AngularJS

According to the meaning of JavaScript that affirms that Angular JS1 is not just a library but also a structure that helps computer programmers with the problems associated with the development of SPA's (Single-Page Applications). In simple terms. AngularJS permits a programmer to design a Hyper Text Mark-up Language web page with an exceptional Mark-up that synchronizes with JavaScript. This division of concerns separates the application logic from the application's views. In addition, several frameworks that are accessible within the market are developed and maintained by the open-source community. (Mehta & al., 2015, 19.)

Nonetheless, Angular is created and maintained by engineers currently working at Google. In 2010, Google created and launched AngularJS. It is not the initial try for Google to launch a framework tool created using JavaScript.

The increase of **HyperText** Mark-up Language 5, Cascading Style Sheets 3 and JavaScript is a trio for front and back-end developers developing web pages. Web developers tend to abandon a project after understanding that web pages cannot be created using only Java. The fundamental benefits of Angular JS are the mark-up in Document Object Model, data as POJO (Plain Old JavaScript Objects) and dependency injections for modules. (Mehta & al., 2015, 19.)

Templates in a few JavaScript structures are executed like:

- Template with mark-up -> template engine of the framework -> Hyper Text Mark-up Language -> Hyper Text Mark-up Language.

AngularJS utilizes the following method:

- **HyperText** Mark-up Language with Angular mark-up -> Document Object Model -> template engine of Angular.

Angular tends to skip the template pattern by incorporating mark-up directly to Hyper Text Mark-up Language and assesses the mark-up solely after Hyper Text Mark-up Language has been loaded into the Document Object Model. The primary benefit of this method is the integration with applications already available in the market since assessment begins just after the web page has loaded completely. Dirty checking, otherwise called digest cycle, is the procedure that keeps the information and view that are synchronized. The structure constantly checks every one of the values to look for modifications to automatically update the model. AngularJS also develops a watchlist, and it notes' down the list utilized to look for any modifications or changes within the model.

3.2.2 Ember

Ember2 is one of the oldest JavaScript Austen utilized by developers. The history began in 2007 when Ember was an essential component of the **SproutCore** MVC (Model-View-Control) system. Towards the end of 2011, Ember2 was renamed Ember.js to avoid confusion between the app structure and the Sprout Core 1.0 widget library. Ember is an open-source JavaScript structure based on the MVC (Model-View-Control) patterns. Like Angular and Aurelia, it permits computer programmers to develop Single-Page Applications with their method. Compared with Aurelia and Angular JS, Ember targets ambitious web programs with many elements that feature adaptability. Ember was intended for designing a particular website page with numerous Ajax requests and UI changes. (Christodoulou & al., 2012, 514.)

Hence, if there is a requirement for CRUD (Create, Read, Update and Delete) on the web page, it is crucial to improve its performance further, avoiding loading the web page again with every action. The Model-View-Control patterns used by Ember make the entire procedure much simpler. Ember depends mainly on convention instead of the configuration or structure paradigm. In most cases, the structure would automatically produce the nodules required for the web app to work accurately. The computer's memory loads these kinds of modules without specifically having to launch any class.

- **Handlebars 3** – Ember utilizes Handlebars3 to create templates by default. Handlebars is a type of JavaScript template framework to create semantic layouts. It aims to separate the view from the logic of the business.
- **Models4** – It is used to control the data of the application. They are entirely independent of the UI; however, they are expected. After refreshing, this particular model informs or alerts the observer, which interprets this into the User Interface.
- **Controllers5** – They represent a model within a template and store properties that could not be saved on the main server. Simply put, it alters the data within a model to change the view. (Abrahamsson & Graziotin 2013, 335.)

3.2.3 Aurelia

Aurelia was first launched in 2015. It is characterized as a platform which can be used to develop Single-Page Applications based upon open-source technologies typically used on the web. A collection of new JavaScript modules offered by Aurelia transforms it into a feature-oriented collection of modules. These modules include injection, dependency, creating templates, binding, etc. However, the most substantial standout from this structure is how it performs the information-binding process. By default, Aurelia utilizes a one-directional information stream by pushing the information from the model into the view through the Document Object Model-batching tool. In simple terms, the changes impacting the Document Object Model would be stored within a queue to be collectively executed afterwards. Moreover, the syntax is very easy and user-friendly. (Reynolds 01 April 2022.)

3.3 Python Language

Python is a famous computer language. It was released in 1991 and was developed by Guido van Rossum. It is utilized on a server to develop web programs, perform complicated mathematics, handle Big Data and scripting systems and connect to different database frameworks to read and make changes to files. Some of the applications of Python are:

- **It is simple to learn** – Python does not have many keywords. It's an easy framework and a clearly established syntax, allowing programmers to learn the computer language quickly.
- **It is easy to read** – Python codes can be defined clearly and visible to human eyes.
- **It is simple to maintain** – The source code of Python is relatively easier to maintain compared to other computer languages.
- **It has a wide standard library** – Python's library is highly portable and compatible with cross platforms, such as Windows, Macintosh, and UNIX.
- **It has an interactive mode** – Python supports an interactive mode, which permits computer programs to debug snippets of computer codes and interactive testing.
- **It is portable** – Python can run on various hardware platforms and has a similar interface on every different platform'.
- **It is extendible** – The programmer could add different low-level modules to the interpreter of Python. These modules also allow computer programmers to add to or custom-make their tools to be more effective.
- **Databases** – Python also provides an interface for each significant commercial database.
- **Graphical User Interface programming** – Python supports graphical user interface programs. Those interface programs could be developed and ported to numerous framework calls, libraries and Windows frameworks, like Macintosh, Windows MFC (Microsoft Foundation Class Library) and Unix's X Window System.
- **It is scalable** – It provides an excellent structure and support for applications than shell scripting.

Some of the most crucial characteristics of Python language are:

- It supports operational and organized programming techniques and OOP (Object Oriented Programming).
- It could be utilized as a scripting language or compiled to byte code to develop massive computer programs.
- It could easily integrate with different computer languages, such as JavaScript, C, COM, CORBA, C++, and ActiveX.
- It also gives relatively high-level dynamic data types and supports dynamic kind checking.

Python has tools for nearly every element of scientific computing. For example, the Bank of America utilizes Python to process its financial data, and social media platforms, such

as Facebook, use the Python library called "Pandas" for analysing data. Even though there are numerous libraries accessible for analysing data in Python, some of the most common libraries are:

- **NumPy** – NumPy is crucial for scientific computing using Python. It supports huge, multidimensional matrices and consists of a collection of high-level mathematical operations used in these matrices.
- **SciPy** – This only works with NumPy matrices and provides effective methods for integrating and optimizing.
- **Pandas** – This is also created on top of NumPy and provides data systems and processes for altering tables that contain numerical data and time series.
- **Matplotlib** – A two-dimensional library that produces data visualizations in the form of a histogram, scatter plot and bar chart with only a few coding lines. (Srinath 2017, s. 2396.)

3.4 R Programming

R is a type of computer program and a software environment that can be used for statistically analysing data, graphically representing the results obtained after analysing the data and reporting. R was developed by Robert Gentleman and Ross Ihaka from the University of Auckland, New Zealand, and is presently being improved by the R Development Core Team. It is freely accessible under the GNU GPL (General Public License) and a precompiled binary version. It is offered for different OS (Operating Systems), such as Windows, Mac and Linux. Some of the most important features of R programming are:

- It is an easy and efficient computer language that has been well-developed.
- It is software that is typically used for analysing data.
- R consists of various operators for various calculations on matrices, vectors and lists.
- It has a compatible and incorporated set of tools that can be utilized for analysing data.
- It provides exceptionally extensible graphical methods.
- It enables users to perform several calculations simultaneously by using vectors.

3.5 Microsoft Azure

Microsoft Azure, previously called Windows Azure, is Microsoft's public cloud computing platform. It offers a broad scope of cloud services, comprising analytics, networking, storage and computing. A user can select from these services to create or run apps already

existing within the public cloud. The main aim of the Azure platform is to assist organizations in managing different challenges and fulfilling their organizational objectives. It provides tools that support every industry, including finance, online businesses and several Fortune 500 organizations and is compatible with open-source technologies. In addition, the platform offers users the adaptability to utilize the technologies and tools they prefer. Moreover, Microsoft Azure provides four various types of cloud computing: IaaS (Infrastructure as a Service), PaaS (Platform as a Service), SaaS (Software as a Service) and serverless. Since Microsoft Azure comprises several service offerings, its use case is very different. (Johnston & al., 2013.)

One of the most famous uses for Azure is running containers or virtual machines on the cloud. These computing resources could host infrastructure elements, for example, DNS (Domain Name System) servers, services that run on Windows servers, like IIS (Internet Information Services) or third-party apps. Microsoft also supports using third-party OS (Operating-System), like Linux. Microsoft Azure is also generally utilized as a platform that can be used to host databases on the cloud. Moreover, this platform is often used to store backups and discovery recovery. Several companies utilize Azure storage as an archive to fulfil their long-term data retention needs. (Peter & Grance 2011.) Microsoft categories Azure cloud service into approximately various categories, comprising:

- **Compute** – This category consists of services that allow users to manage and utilize VMs (Virtual Machines), batch jobs, and containers and enable users to access applications remotely. Compute resources developed inside Azure could be configured using private and public IP (Internet Protocol) addresses based on whether the resource should be open or available to the rest of the world or not.
- **Mobile** – This class comprises products that assist software engineers in creating cloud programs or apps for smartphones. It also provides alert services, supports different back-end tasks, provides tools for developing APIs (Application Program Interfaces) and connects geospatial settings with data. (Bocchi & Mellia 2014.)
- **Web** – This group of services supports web programs' creation and utilization. They also provide features for reporting, notifying, managing Application Program Interface, delivering content and searching.
- **Storage** – This category includes services that offer accessible cloud storage for unstructured and structured data. It also supports Big Data projects and archival and permanent storage.
- **Analytics** – This category comprises services that offer distributed storage and analytics and components for cloud analytics, Big Data analytics, ML (Machine

Learning), IoT (Internet of Things), BI (Business Intelligence) and data warehousing, lakes and streams. (Insik & al., 2012, 527.)

- **Networking** – This class consists of networks that are created on Virtual Machines, gateways and IP addresses that are dedicated to a particular network and services for managing network traffic and diagnostics, DNS (Domain Name System) and protecting the network against DDoS (Distributed Denial of Service) attacks.
- **CDN and media** - These CDN (Content Delivery Network) services consist of encrypting, indexing, media playback, streaming as required and digital license protection. (Steven & al., 2013.)
- **Integration** – This class consists of services for backing up servers, recovering sites and linking public and private clouds.
- **Identity** – These services ensure just approved clients can access services offered by Azure and assist with protecting encrypted keys and other essential data on the cloud. Services comprise supporting Azure Active Directory and MFA (Multifactor Authentication).
- **IoT** – This category consists of services that assist users with capturing, monitoring and analysing Internet of Things data acquired from sensors and different gadgets. Services comprise supporting coding and implementation, monitoring, analytics and notifications.
- **DevOps** – This category offers collaboration and project tools, like Azure DevOps, once known as Visual Studio Team Services, that work with DevOps programming development procedures. It also provides components for program diagnostics, DevOps integration tools and test labs for building tests and conducting experiments.
- **Development** – These services assist programmers with sharing code, testing programs and tracking possible problems. Azure supports several computer programming languages that are used to create applications, such as Python, Node.js, JavaScript and . NET. This within this class also comprises supper for SDKs (Software Development Kits), Blockchain and Azure DevOps. (Insik & al., 2012, 527.)
- **Security** – This category offers products that recognize and respond to various cloud security threats and manage encrypted keys and essential resources.
- **Machine Learning and Artificial Intelligence** – This is a broad scope of services that a programmer could utilize to integrate AI (Artificial Intelligence), ML (Machine Learning) and analytical computing abilities into programs and datasets.

- **Containers** – This group provides services that assist a company with creating, registering, organizing and managing large volumes of containers within the Azure cloud, utilizing popular platforms like Kubernetes and Docker.
- **Databases** – This class consists of DBaaS (Database as a Service) for SQL (Structured Query Language) and NoSQL and different database examples, like Azure Cosmos DB and Azure Database for PostgreSQL. It also comprises Azure SQL Data Warehouse support, storing, integrating, and migrating hybrid databases. Azure SQL is a relational database that offers all the functionality of SQL without the requirement of using a SQL server. (Peter & Grance 2011.)
- **Migration** – This tool assists in estimating the cost of migrating workload and migrating the workload from a local data centre to the Azure cloud.
- **Governance and management** – This group of services offer various planning, computing, backup, managing and recovery tools that could assist a cloud manager in managing the deployment of Azure.
- **Mixed reality** – This category of services is intended to assist programmers with making content for the Windows mixed reality setting.
- **Blockchain** - The Azure Blockchain service permits the user's to join the blockchain association or make their own.

3.6 Low-Code AI Chatbots

Chatbots so far have been developed by coding in different programming languages combined with web API. The well-known programming languages for deployment are Node.js, PHP, Python and Java. Recently the trends are changing from coding to no-code or low-code methods deploying. But, No-coding options are not flexible like low-code development. (Jarocinski 25 August 2020.)

Low-code platforms are becoming increasingly popular for building applications, including chatbots. By using a low-code platform, developers can quickly and easily create chatbots without needing to write a large amount of complex code. This type of chatbot creation method can be especially useful for creating simple chatbots focused on specific tasks, such as answering frequently asked questions or providing customer support.

Another advantage of using a low-code platform for chatbot development is the ability to easily integrate the chatbot with other systems and services. For example, a chatbot created with a low-code platform can easily connect to a customer relationship management (CRM) system to access customer data or to a messaging platform to handle incoming

user messages. This can greatly simplify creating and deploying a chatbot, making it possible to get it up and running quickly and with minimal effort. (Karthik 21 February 2022.)

3.6.1 Microsoft Azure Bot Service

The main goal of having Artificial Intelligence (AI) in place of actual human interaction is to improve user engagement and experience much more efficiently, especially answering routinary or frequently asked questions. In today's technology, various well-known platforms offer chatbot UI. The most known companies, such as Microsoft Azure, AWS, and HubSpot, offers chatbot to different industries. Research by Maly (21 January 2022) in the Customer Service industry shows that "chatbots can increase engagement by up to 90% and sales by 67%. In 2020, 57% of businesses said conversational bots deliver substantial ROI for minimal effort." The results show that having a chatbot in place greatly benefits industries such as fintech, healthcare, retail/eCommerce, education, banking, and travel. Using chatbots saves time and can help users focus more on conversations that matter most for specific inquiries.

One of the most well-known conversational AI platforms is Microsoft Azure Bot Service. With its enterprise-grade chatbots, it can intelligently build a bot that can be easily deployed to multiple channels.

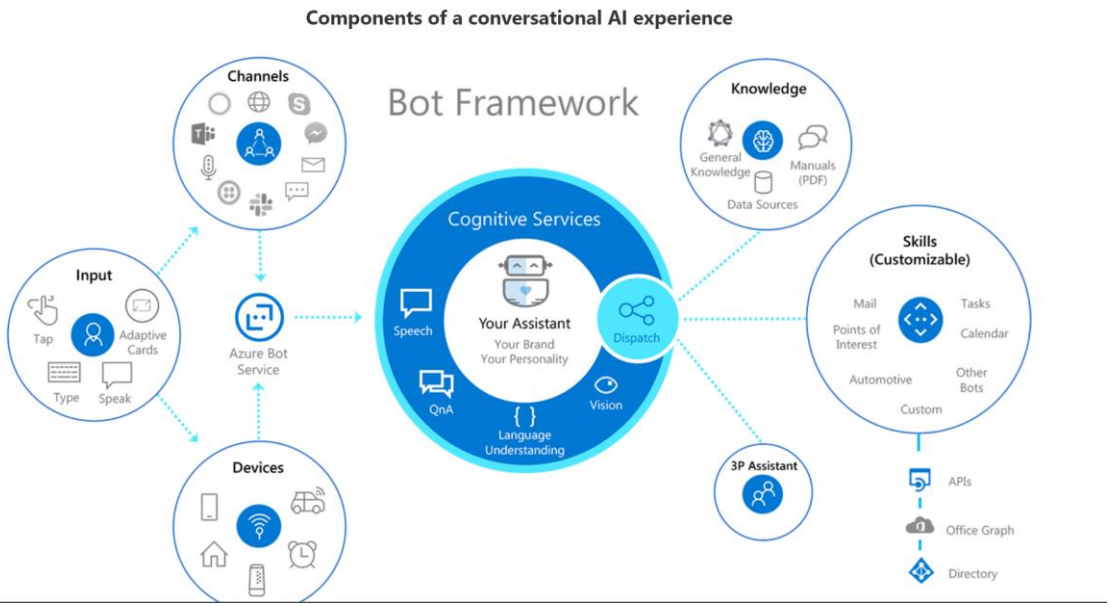


Figure 1 Components of a conversational AI experience (Microsoft 2019)

Azure Bot Service has three options to create a chatbot; we can build a chatbot using No-Code, Low-Code or Coding options.

As shown in figure 2, the complete process of building a chatbot needs channels where a user inputs a text, speech, or image. This will directly go to the Bot application, which processes the user input and gives back the necessary information. A Bot application is

where a Bot UI is developed.

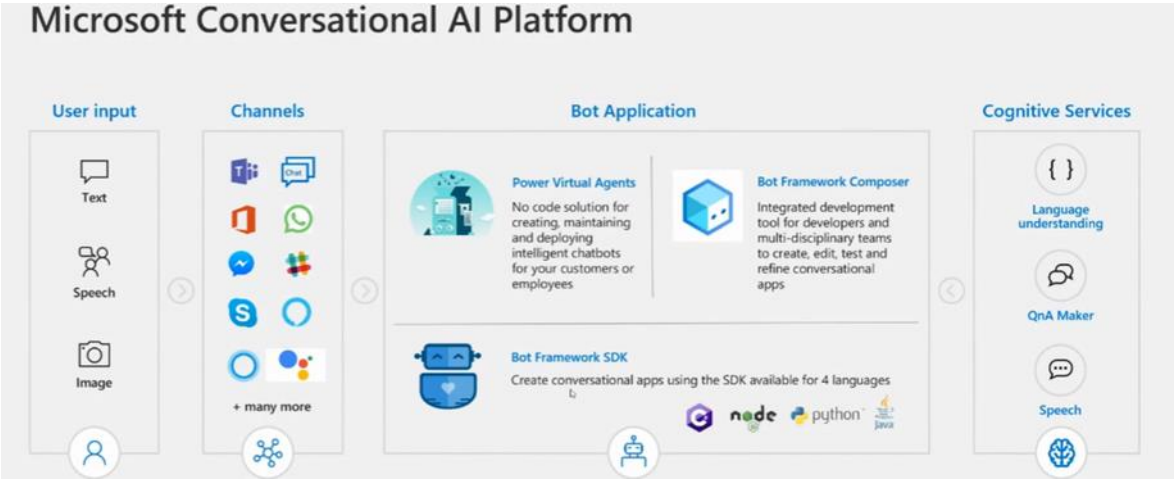


Figure 2 Improve Customer Engagement and Productivity with Conversational AI ((Microsoft 2022)

Fig 2 shows the three Bot building methods/Bot Applications:

- Microsoft Azure Power Virtual Agent is a No-code option,
- Microsoft Azure Bot Framework Composer is a Low-code option, and
- Microsoft Azure Bot Framework SDK is a coding option which uses different programming languages like C#, Java, JavaScript, or Python.

3.6.2 AWS Chatbot

Amazon Web Services (AWS) Chatbot enables DevOps and software developers to use messaging program chat rooms to monitor and respond to operational events in their AWS Cloud.

AWS Chatbot utilizes natural language processing for its conversational AI. It also includes an Interactive Voice Response (IVR) and automated customer voice authentication for a more seamless transition to what the end user seeks. Its machine learning function also stores valuable insights from its users, which easily integrates and connects to multiple supported channels such as Slack and Amazon Chime Webhook. (Tiwari 8 June 2022.)

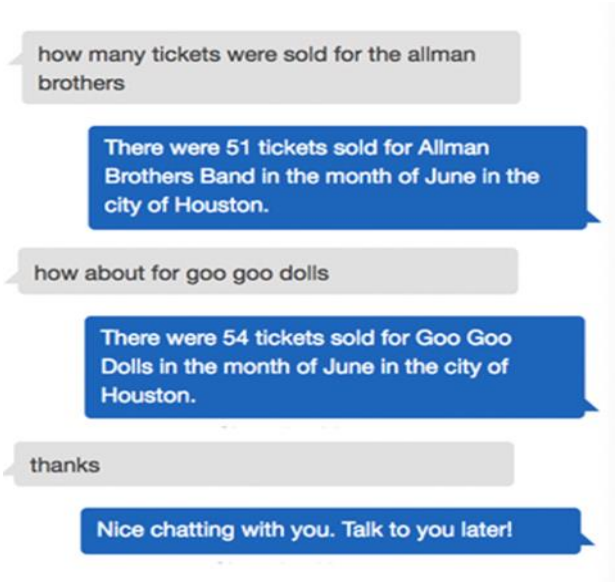


Figure 3 Sample Interaction with BIBOT (AWS)

3.6.3 HubSpot

HubSpot offers a free chatbot builder software as it can easily create messenger bots that help users in qualifying leads, booking meetings, providing answers to common customer support questions, and more. Furthermore, users can create and customize chatbots without coding skills due to its canned templates and intuitive chatbot builder, making it more user-friendly to non-developers. In addition, it can easily integrate and connect to platforms such as Facebook Messenger and other social media platforms.

“Research and development in chatbots grew after Facebook opened their Messenger platform to make it easy for anyone to build a chatbot on top of their Wit.ai Bot Engine. This effectively made it easy to quickly and easily deploy a chatbot that interacts with customers” (Jarocinski 25 August 2020).

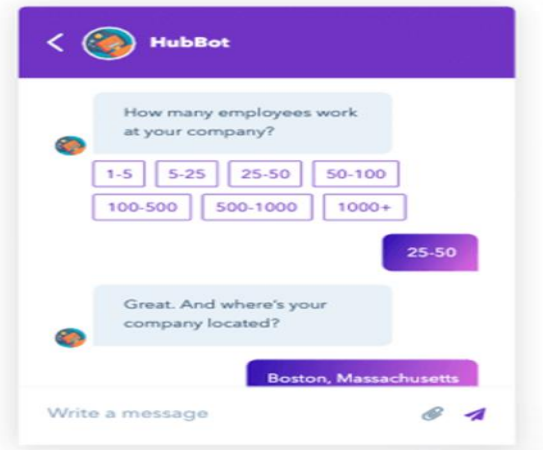


Figure 4 Shared Inbox for Customer Conversations (HubSpot 2022)

3.7 Data Integration

Data mining is the method of findings patterns and relationships in massive datasets. This advanced type of data analysis draws from Artificial Intelligence, statistics and Machine Learning to assist a user with locating significant data. The individuals who are experts at data mining could give their company insights regarding client needs and ideas on reducing costs and increasing profits. The current demand for effective techniques for data mining is predominant across different industries. Data mining is typically utilized in fields such as medical care and professions that use geographic mining, for example, spacecraft design and Global Positioning System-powered tools used for navigation, such as Google Maps. (Lausch & al., 2015, 7.)

3.7.1 KNIME

KNIME is an important data integration and prescient analysis platform. Specifically, the program assists in reporting the complicated steps typically executed in pre-processing, statistical modelling and analysis and proactive analysis. Another advantage is its open-source collaborative system, where developers are free to create new tools and algorithms and information manipulation or representation or illustration techniques. (Lausch & al., 2015, 7.) KNIME was developed in 2004 and led by a group of Silicon Valley engineers studying at the University of Konstanz, Germany, and Michael Berthold was the project head. The program discovered heavy usage in cheminformatics due to the degree of automation needed to investigate atomic structures. Currently, the tool detects or observes a minimum of 50% of the clients coming from relatively different areas of Business, Social Science and Medicine. For example, economists utilize KNIME within CRM (Customer Relationship Management) and private banks, such as the Grant Casino and Zurich, located in Lucerne.

Some of the accessible KNIME nodes cover an enormous scope of functionalities, and nodes could be classified as follows:

- **Input/Output** – Data which is retrieved from different documents or databases can be exported in several different formats.
- **Data manipulation** - Pre-procedures, the input data through filtering, sampling, normalization, binning, accumulation, partitioning and different operations, for example, replacing the missing data.
- **Views** – Represent information and results using a few interactive views, enabling interactive information analysis.

- **Mining** – It utilizes new data mining techniques. For example, assembling, creating a decision tree, induction of rules, association rules, neural networks, and helping vector machines give some examples, permitting pattern recognition, a deduction in dimensionality and reconstruction of feature tasks. Each region of ML has been used for numerous applications in the past ten years of geoscience.

A few of the most broadly utilized computer languages, such as some of the languages and tools stored inside the KNIME library as snippet nodes, are MATLAB, R Language, JavaScript, Octave and Python. Apart from the proprietary KNIME node, configurable or customizable notes offer practically unlimited adaptability. These nodes could be incorporated into hybrid coding solutions, using various armoury of tools that are accessible in every computer language or platform. Different nodes are characterized as wrappers, which incorporate or combine third-party functionalities. Specifically, KNIME incorporates the functionality of a few open-source projects that cover each significant area of data analysis, such as Weka and **JFreeChart** can be used to graphically represent the results (Lausch & al., 2015, 7). The huge range of ML and DM (Data Mining) tools in KNIME makes the platform's relevancy exceptionally broad. KNIME is currently utilized in several fields, for example, Environmental Science, Production, Medical Care, Consumer Intelligence, Retailing and Finance.

There are numerous benefits to consider when utilizing KNIME for data mining and analytics, especially for these types of projects are:

- KNIME will integrate well with the Structured Query Language databases.
- Its instructive drop and drag function enable clients to automate several database operations, like writing, reading and combining.
- KNIME is especially useful for repetitive tasks since it could automate the procedure, significantly decreasing the time the user would have to manually finish these tasks.
- It can easily connect with BI (Business Intelligence) tools typically used to share files.
- Even people who do not have enough knowledge or training about how to code in different computer languages can utilize KNIME.
- KNIME could connect to several computer languages, like R Language, Python and JS.
- It is free and open source. (Khan 6 April 2021.)

Even though the KNIME Analytics Platform increases the level of productivity of people or teams of data engineers and researchers, the KNIME server enables organizations to develop different programs related to data science. It assists companies with deploying their programs as RESTful service endpoints, analytics programs and visualizations. KNIME server also helps to provide planning and distributed execution abilities. These consist of features, for example, deploying data science work processes in analytical services and programs, providing a platform for teams to collaborate and automation administration. With the KNIME server, companies could control who can access the data and work processes, making it simple to act under different data protection policies. They could also create workflows that can be easily edited and traced that could be used again across different teams, assisting the team with breaking through information silos. KNIME servers could be deployed on-premises, such as AWS and various cloud platforms. Despite where the organization host it, the KNIME server can run the KNIME analytic platform and each fundamental extension, integration and work process utilizing connectors as required. (Falgout 9 January 2020.)

3.7.2 Boomi

Boomi offers traditional integration platforms. It is an enterprise integration platform as a service (iPaaS). When it comes to developing and maintaining business processes, Dell Boomi Atmosphere provides a user-friendly User Interface (UI), all in a single place. (Chaubey 15 November 2021.)

The focus of Boomi is integration, and with the growth of cloud-based services, it has become more popular with users as an iPaaS solution. For those users who confine their data on-premise, Boomi provides a way to integrate their on-premise data with the cloud and vice-versa. In addition, Boomi can connect with the following: (Chaubey 15 November 2021.)

- Cloud to Cloud
- SaaS to SaaS (software-as-a-service)
- Cloud to On-Premise System
- On-premise to On-premise
- B2B integration

In a way, Boomi gives users' data a voice to be heard on different platforms. Let's dig deep into Boomi and look at the most important features for data Integration.

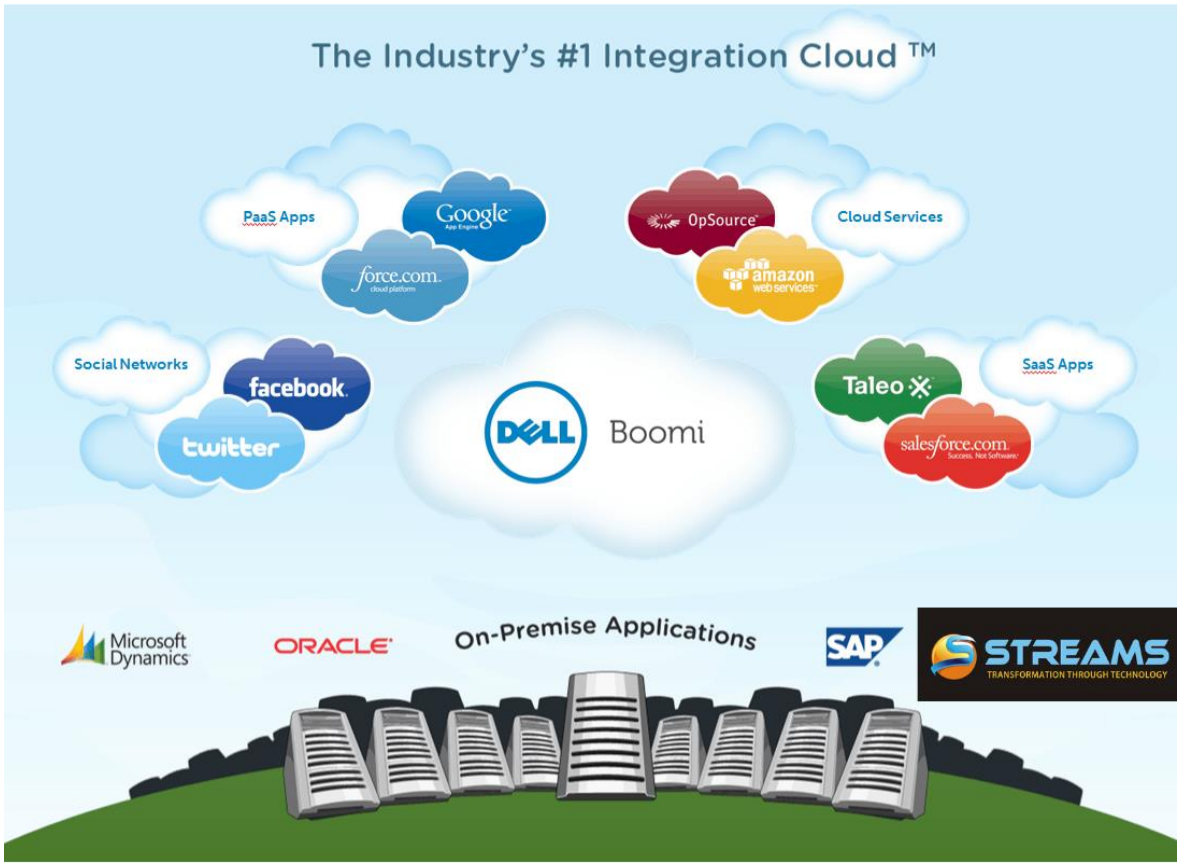


Figure 5 Streams Solution (Boomi 2022a)

- **AtomSphere** - is a single platform for Boomi, which in addition to integration services, houses the whole services provided by Boomi, which includes Master Data Hub, API Management, Flow, Data Catalog and Preparation Environment. All of these mentioned are handful for users to switch between services easier and integrations faster. (Rebernik October 2012.)
- **Environmen**t - In **AtmoSphere**, the environment is a workspace a user creates for testing and production purposes. It houses atoms and makes managing deployments more applicable, and it is also used to distinguish between test and production, classify on-premises resources and manage client-specific projects. "Environments help to ease the management of larger implementation projects that require multiple application setups, on-premises resources, and a distributed architecture."
- **Atoms** - are lightweight, dynamic runtime engines which contain all the components needed to create our integration processes, such as connectors, maps, logic and error handling. They are small, highly scalable, autonomous, flexible and powerful. Based on the integration demand, atoms can be installed locally or on the cloud.

- **Process** - is the central component within Boomi Integration; it is a graphical representation of the path that a document takes from the point at which it is received or retrieved to the point at which it is sent to one or more destinations. Processes are easier to understand at a basic level; they take data in, modify it and provide an output; that's what they do. However, the input requires a source and a way to connect to that source. The modification requires logic, making sense of the data and shaping the data in a certain way. Finally, the output also needs to know the destination and how to connect to it.
- **Connectors** - are an integral part of the integration and bring data from the source and send it to a destination. There are two types of connectors, inbound connectors and outbound connectors. The inbound connector brings data into the process while the outbound sends data out to the destination. The connector specifies where and how the data is imported by using connections. The connection is where we determine the data source or the destination. For example, suppose we want to connect to a database. In that case, we specify the host, username, password(if available), port number, database name, and other configurations required for the connection. The other part of the connectors is the operation; here, we specify what actions our connector will perform, whether to get or send information. It is also possible to riddle the data using parameters if the operation supports it. Hundreds of connectors are readily available with pre-configured settings in the Atmosphere platform, making it convenient for users.
- **Map -** helps transform the data into the desired structure and provides readily available functions to make it easier. Still, if those aren't enough, writing a script in Java(Groovy) or JavaScript can get the desired outcomes. For example, if we want XML data to be converted into JSON, we provide the input with an XML profile and the output to JSON. (Boomi 2022b.)

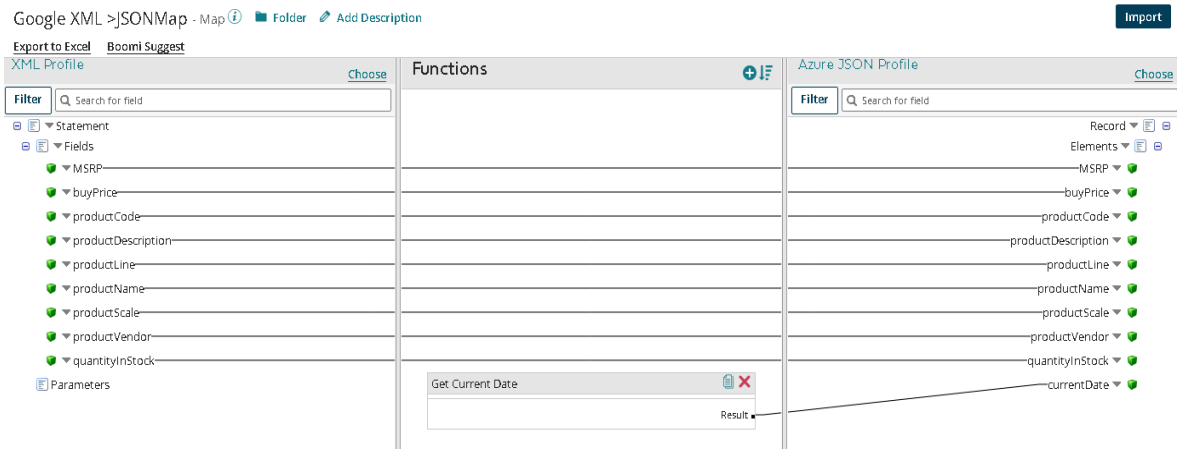


Figure 6 Map Function Components (Boomi 2022c)

As shown in figure 6, **currentDate** is not mapped from the source file but is needed in the JSON file, so by creating a function, we can map it to the output of the "Get Current Date" function.

- **Profile -** Profiles can be manually built into the profile page or automatically imported from the source data. If additional properties are needed, it is always possible to add them manually. Profiles are what our data look like and in what data format we want them to be. It gives the users total control of the input and output of the data type, shape, size and more.

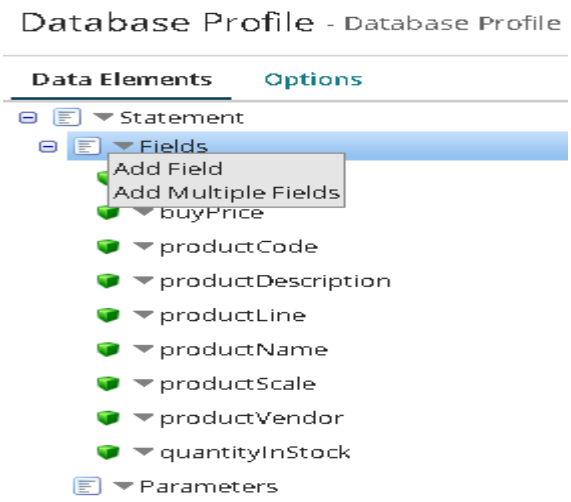


Figure 7 Database profile for a MySQL product database table.

4 Chatbot design

4.1 Development environment

The researcher will develop the chatbot for the thesis showcase in the Microsoft Conversational AI platform, which offers both AI and Deep Learning methods. As discussed in part 3.6.1, the Microsoft Conversational AI platform provides No-Code, Low-Code And Coding options to create a bot application. From the available three chatbot-creating methods, Bot Framework Compose, a low-code building option, is chosen for the development.

Bot Framework Composer is chosen because It's a Low Cod bot application. Anyone without programming knowledge could create a chatbot using the Bot Framework Composer. At the same time, Bot Framework Composer offers a code for developers to manipulate and improve the capability as needed.

4.2 Database

A database is where we keep data in a structured way. It's set up for easy access, management and updating. The set of data in a database can be customer data, sales transactions, financials, and product information. A database can improve business processes, keep track of customers, secure personal health information and store personal data.

There are different types of databases available to use. According to MD, the following ten are the best to use in 2021, MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB, Redis, Elasticsearch, Cassandra, MariaDB, and IBM Db2.

The researcher chose MySQL database, which is a Rational database. The reason behind choosing SQL database is because it's open source and safe. In addition, MySQL offers structured data, which is what the bot implemented needs. The chatbot needs a database where the data is stored and fetched. It uses the stored data to provide the necessary information for the user's needs, and MySQL is a good fit.

4.3 Data analysis and Machine learning

The researcher chose Boomi, a low-code Integration Platform, for data transferring. As it's explained in part 3.7.2, Boomi mainly focuses on integration. Boomi Offers different connections; the connection we need for our chatbot is from the cloud to the Azure chatbot.

Different types of formats are used to store and transport data. Depending on their capabilities, they are utilized on all digital platforms. But because of their structure, it is impossible to use them on any platform as needed. So, for example, let's say we are storing data in our database; it would be hard to utilize the data from it on our website. So instead, the data has to be converted into an XML or JSON to be read in our Frontend. The data for the Implemented chatbot is stored in a cloud database, so we need an Integration platform to fetch it and convert it to JSON. This is where Boomi comes in and changes the cloud data to JSON.

The researcher chose Boomi for the same reason as the Bot Framework composer. It's a low-code option, which makes it easy to develop, but at the same time, it offers a chance to code and improves the features if needed.

5 Implementation

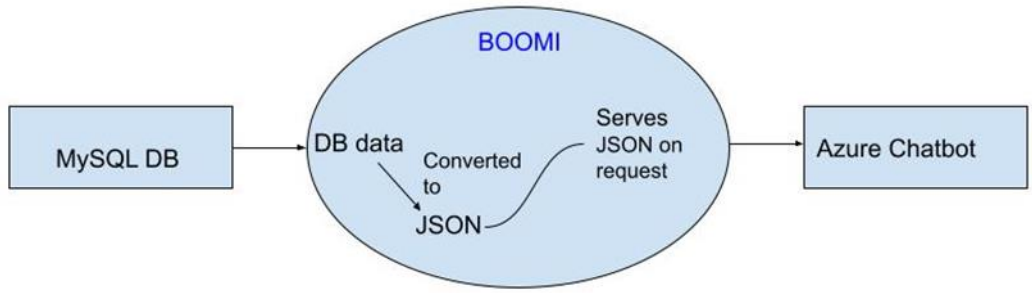
This section discusses the empirical part of creating a Low-code AI chatbot. The researcher chose to implement the chatbot on the Microsoft Azure Bot Framework Composer. The chatbot is created for a Banking Information search showcase.

The chatbot data is fetched from an external database source. In our case, it's stored in a MySQL database. To access the data Boomi Integration is practised. This part will also discuss Boomi Integration, which is creates an API for the chatbot.

5.1 Preparing the data on Boomi

As mentioned above, Boomi provides applications and platforms to communicate and integrate easily without spending too much time on technical coding. So, the writer chose Boomi to integrate the data from the database and utilize it in my azure chatbot.

The process looks more like this.



The data from the database is a list of users' **accountID**, **accountType**, and balances, which the user will use in the chatbot to ask questions about their account details.
But our chatbot needs to request some API and get the data back in JSON.
And that's what we'll be doing with Boomi.

First, we'll set up the test and production environments, then atoms, on which we'll test our workflow and deploy the server.

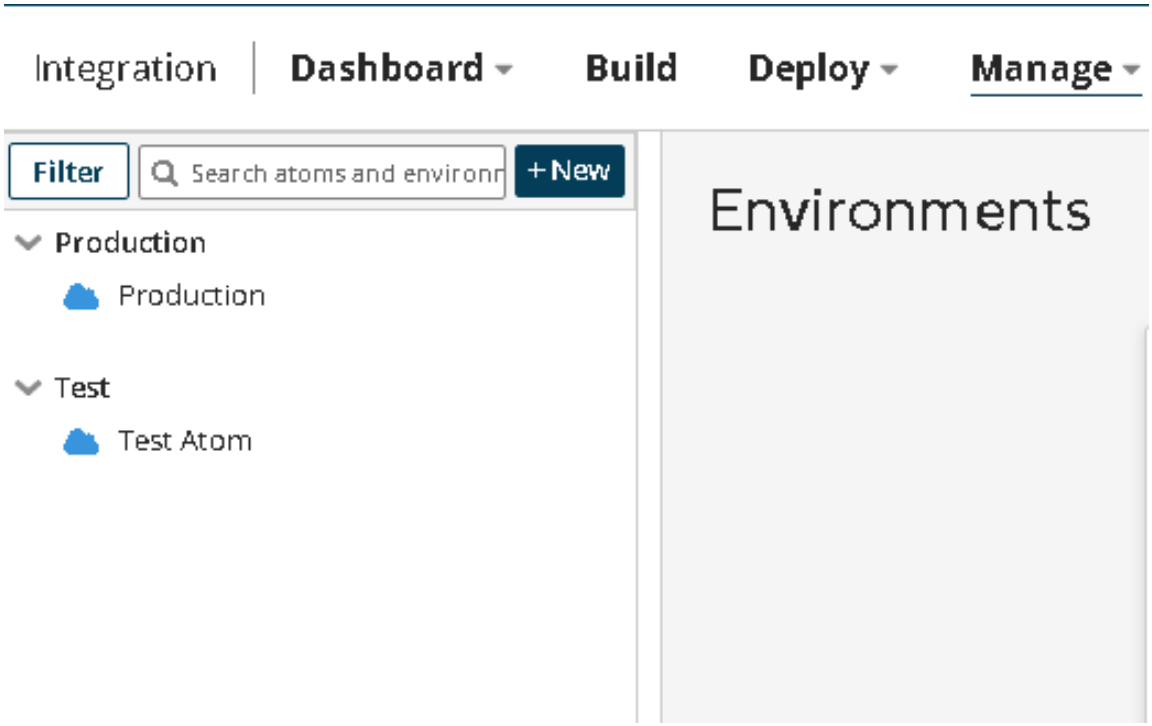


Figure 8 Creating environments and atoms.

After creating our environment, as shown in Fig 8, we can start creating the processes to get the data from the database.

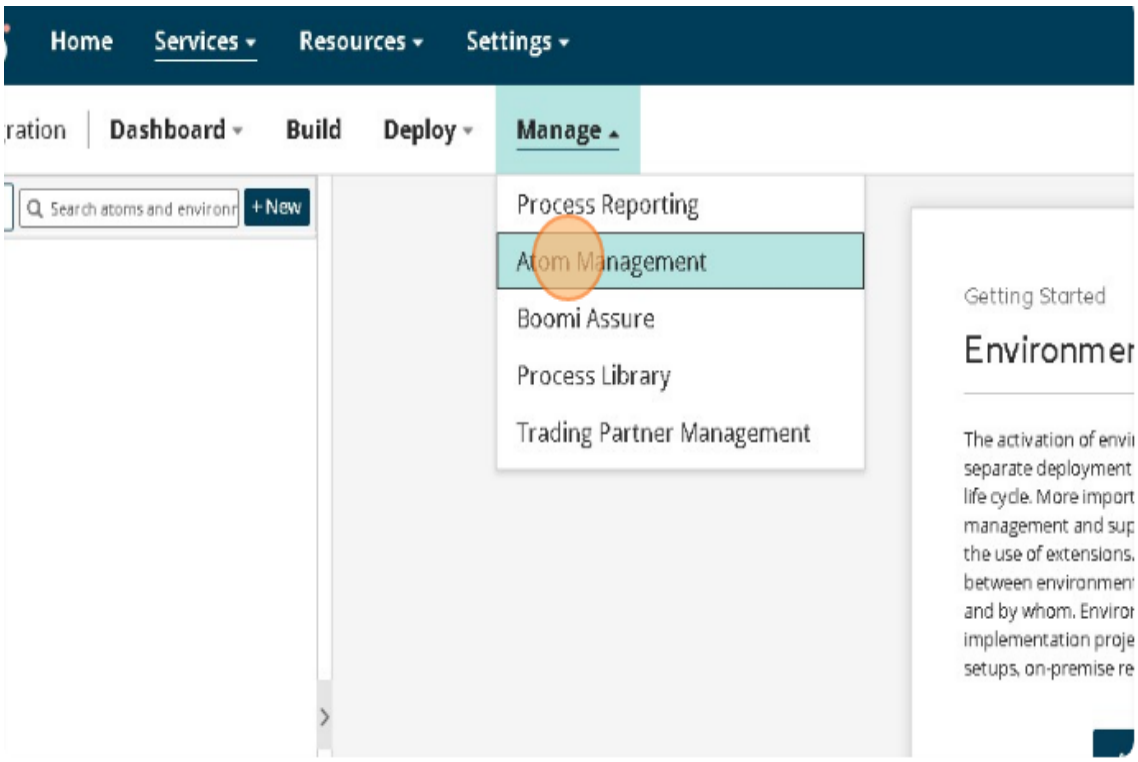


Figure 9 Atom management page

Figure 9 shows that the Atom management page is selected from Manage.

Figure 10 Boomi process to convert DB data to JSON.

The above Figure (Fig 10) shows the process that converts the data received from the database to JSON. In addition, the operation specifies the return data for a certain ID.

The start shape is a Database connector, which holds the connection, operation, and parameters to fetch the data. It is a common connector that is readily available in the Boomi platform.

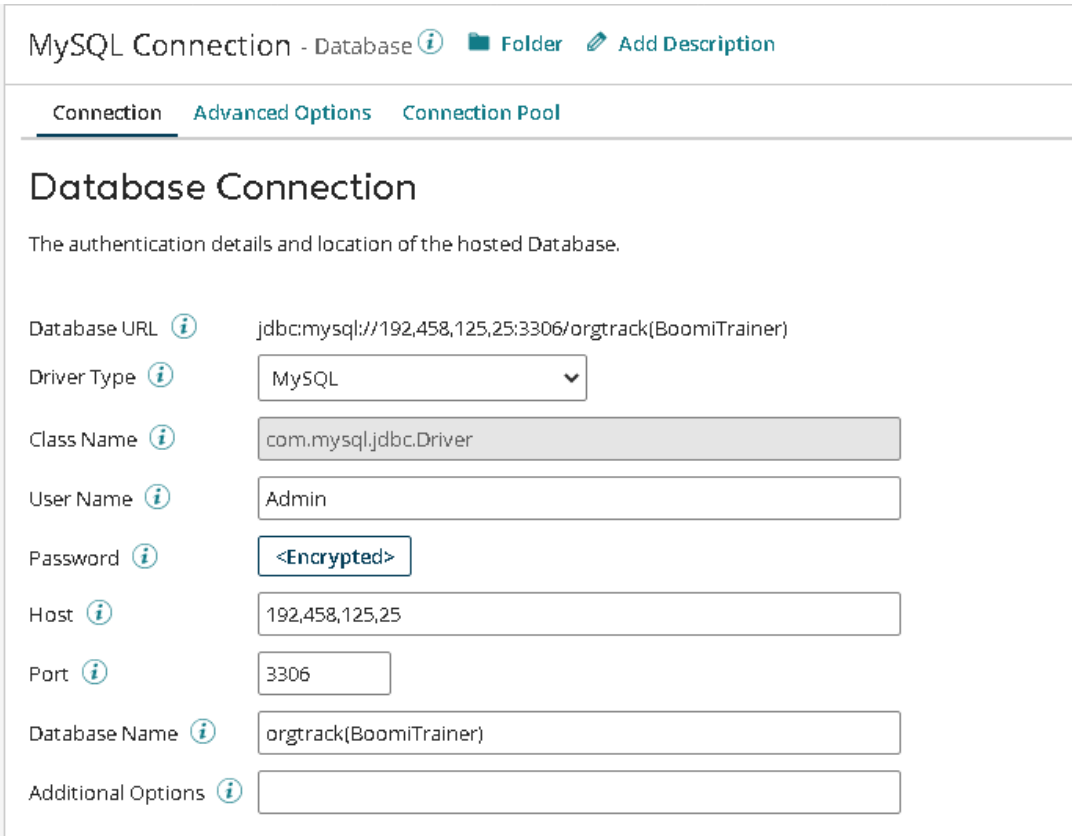


Figure 11 MySQL DB connection

As shown in figure 11, the connection keeps track of the connection settings and where we are connecting, but we will need the operation to define whether to get or send information and how to batch-commit database inserts.

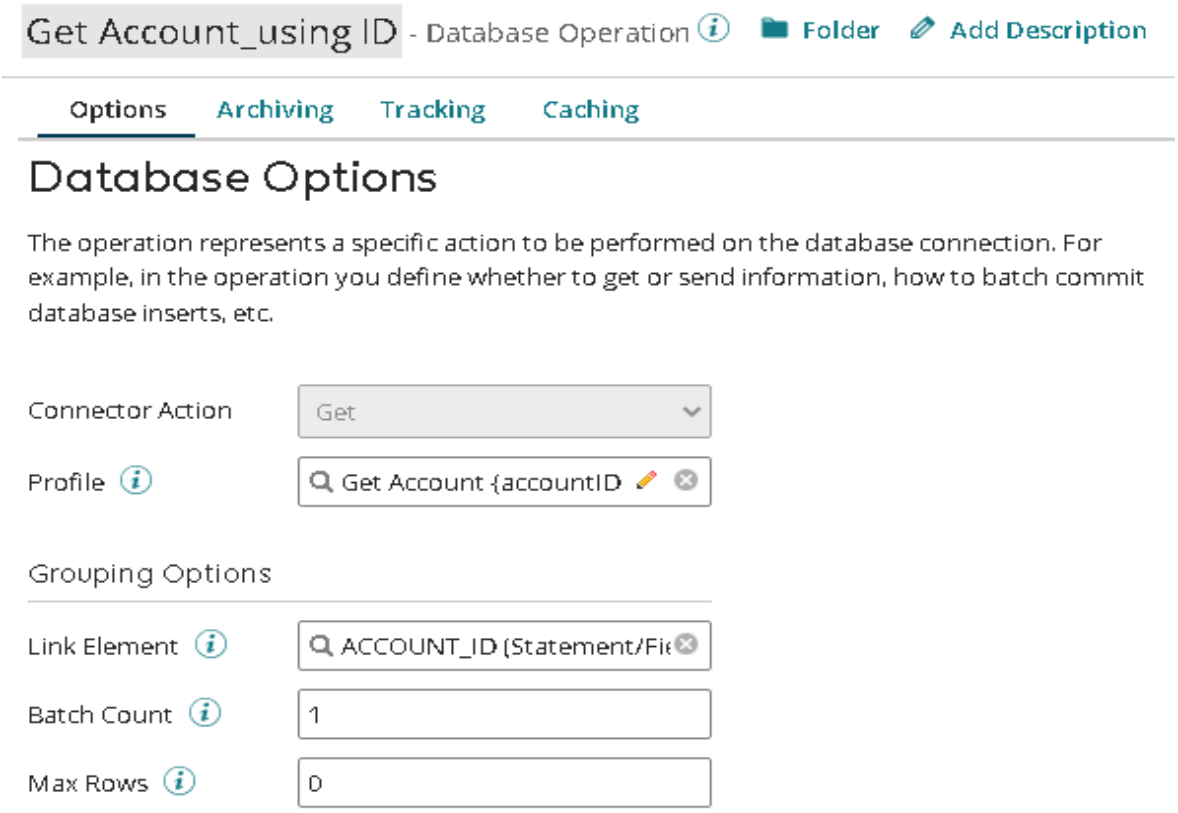


Figure 12 Connector operation

In the above figure (Fig 12), we are defining the action we have achieved by using this connector which is to "Get" the data from the database. With that, it is also possible to easily modify and limit the amount of data received. This result can be achieved by writing a database query, which we can do when importing the profile. The profile is the layout of data the process expects from the database; this can be created manually or by importing and automatically building the profile based on the table from the database.

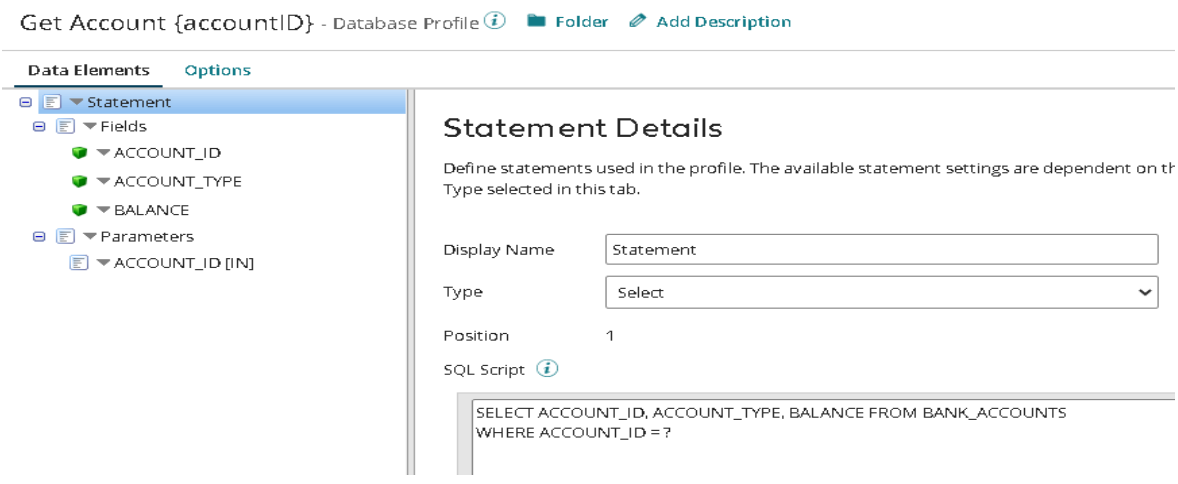


Figure 13 Building the database profile

Now we have the data from the database, as shown in Fig 13, and it is possible to convert it to JSON using the map shape, which will use lines to map an individual column to the desired JSON value. Although it is possible to create more complex JSON objects using this, we'll not need nested values for now.

See the image below for the mapping. On the right, we have the JSON profile we'll be mapping it to, and on the left, the Database profile we built based on the data coming from the database.

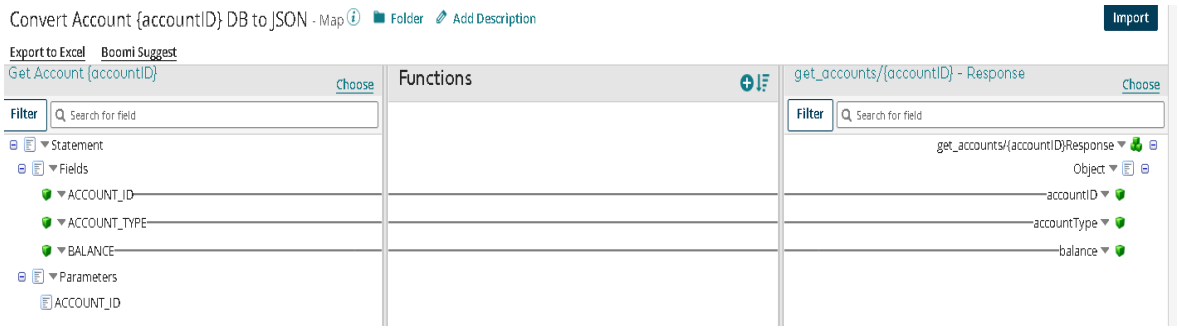


Figure 14 Mapping from DB profile to JSON profile

Once we finish the mapping, as shown in Fig 14, the output will be JSON data, this data will be used in our chatbot, but our chatbot doesn't have a source to get this data, so we have to build a new process that can interact with our chatbot using predefined endpoints.

The first step would be to create a connector. This connector only listens for certain requests that meet the necessary authentications requirement. Upon a successful request, it will execute the above process as a subprocess and returns a JSON document as a response.

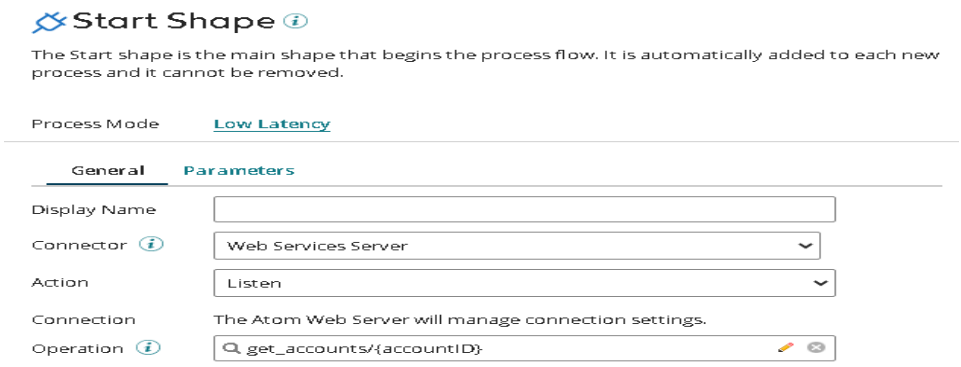


Figure 15 Configuration for the server connector

As mentioned in previous sections, a connector is essential to getting data into the integration process. In this case, the connector is a "web service server" that listens to requests. In operation, we would define a specific action requirement, the expected type of request sent from the client application and the type of the expected response. For example, look at Fig 15 and Fig 16.

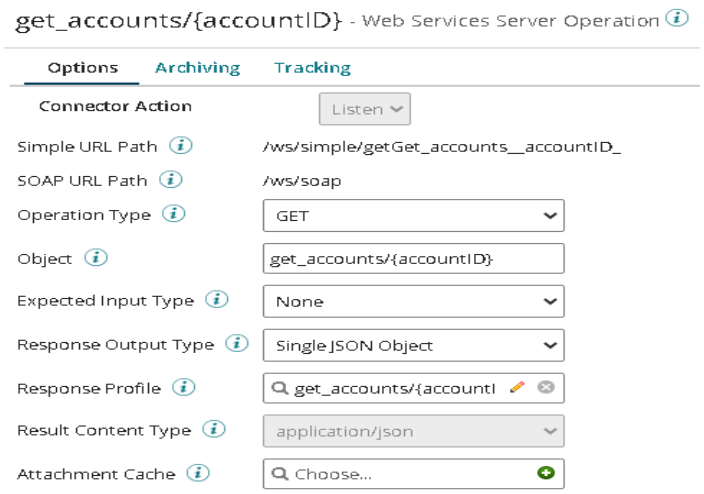


Figure 16 Connector action

Once we're done setting up our operation, we will need to create a service to define the resource URL. This process can be done by creating a new "API service" component. This component will specify the metadata to be used when the APIs are displayed to the API end users, such as the published API title, version number and description. In addition, it specifies the base URL for the API.

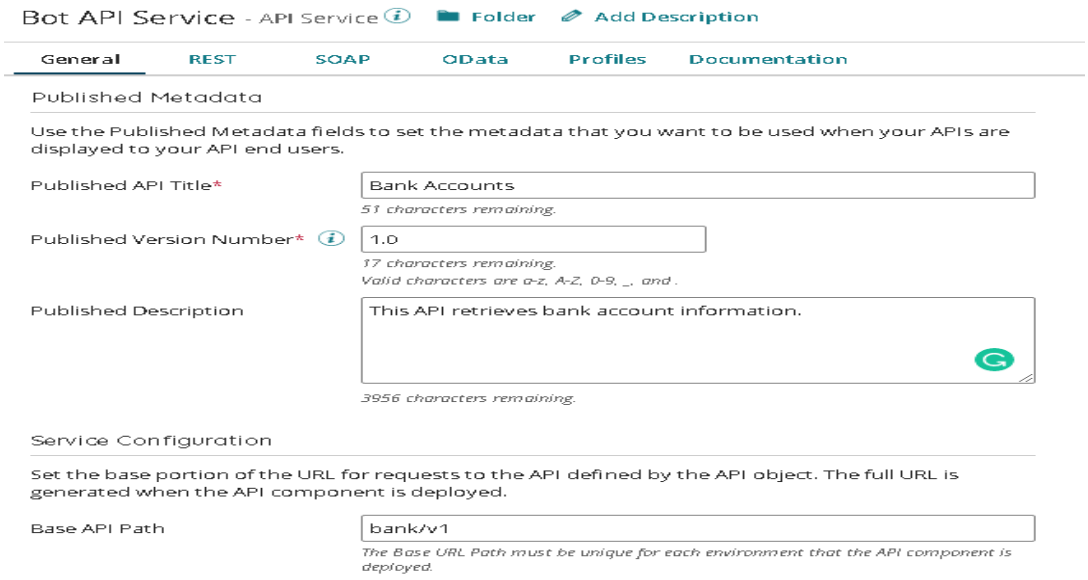


Figure 17 API service to create base API path

After configuring the general settings for the API, we click "REST" to specify that we are serving our data over a REST API. And what resource paths will activate our processes and the actions required for that, such as get, post, delete, patch or put. The action is shown in Fig 18.

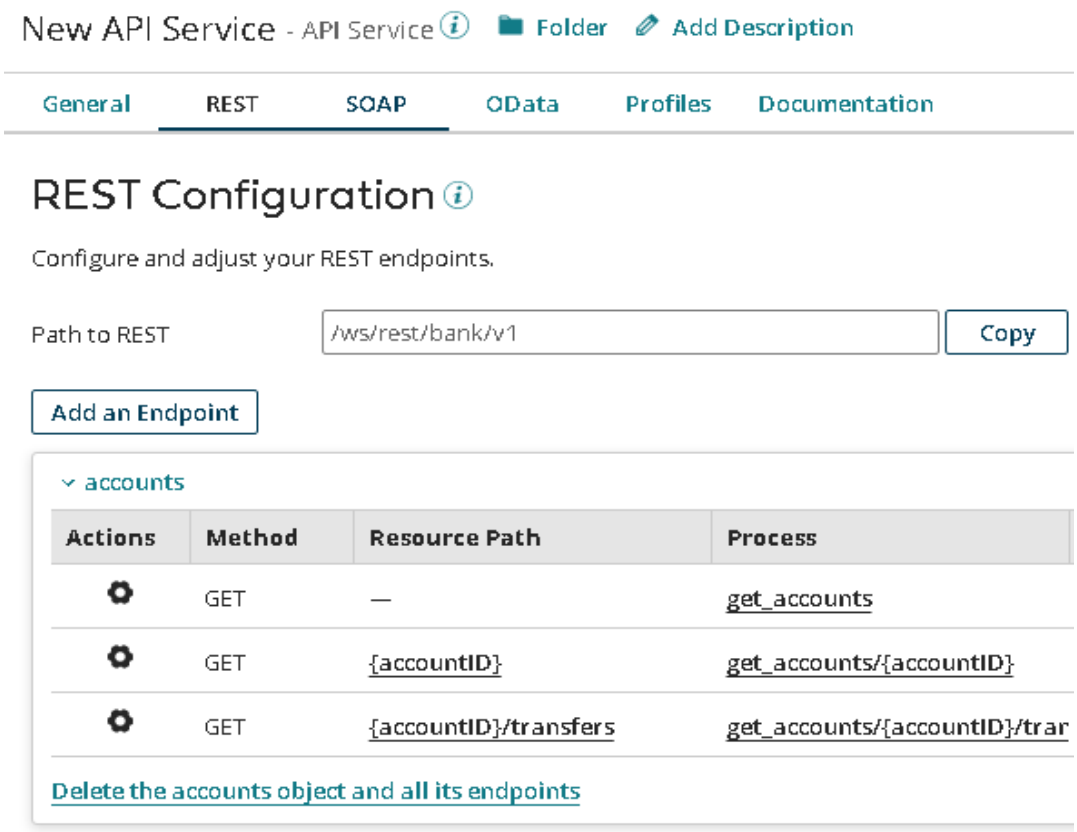


Figure 18 Setting up endpoints

Once the setup is finished with our connector, we'll need to get the data from somewhere. So we'll use our first process and attach it to our "web services server" the connector will also pass the **accountID**, which was used in the request from our Azure chatbot, as a value to execute the process. The output of the above process, as shown in Fig 20, will be a single JSON object that will be returned in the "Return Documents" shape.

Figure 19 Final look at our server process

The output of the above process, shown in figure 19, will be a single JSON object; it will be returned in the "Return Documents" shape, and it is triggered by the start shape, which listens to an API request with an account ID, which will be made by Azure chatbot.

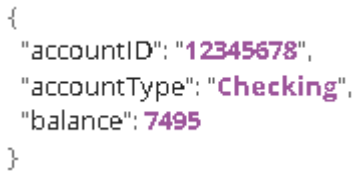


Figure 20 Final output of our process

Figure 20 shows the API response. It gives **accountID**, **accountType** and balance.

5.2 Building the Chatbot on Microsoft Azure Bot Framework Composer

As explained above in the implementation, once an account is created, select a subscription on Microsoft Azure and download Microsoft Azure Bot Framework Composer. Next, we will open the Bot Framework Composer and sign in with our Microsoft Azure account. The Bot Framework Composer application canvas has different elements available to help us build the AI chatbot easily.

There are four main parts to the Bot Framework Composer.

- **Navigation Pane**: Is used to navigate the composer's option and feature
- **Bot Explorer**: displays the components to Implement the project. This part includes Dialogs and Triggers.
- **Authoring Canvas**: This is where we can find all the trigger actions. All logic appears here.
- **Properties pane**: All properties are handled here. Different actions like Prompt for text and sending an HTTP request are done in this part.

5.3 Creating the Bot welcome page

The first step in building the chatbot is to select create from the Navigation Pane. We then create an empty chatbot on the Bot Explorer and rename it to what we want our chatbot name should be. In our case, it's “**BankAccountBot”**. The "**BankAccountBo**t" will then be created and stored on the local computer. Then a Dialog is created for the bot and given a name according to our needs. In our case, it's named the same as the bot name. See the figures below.

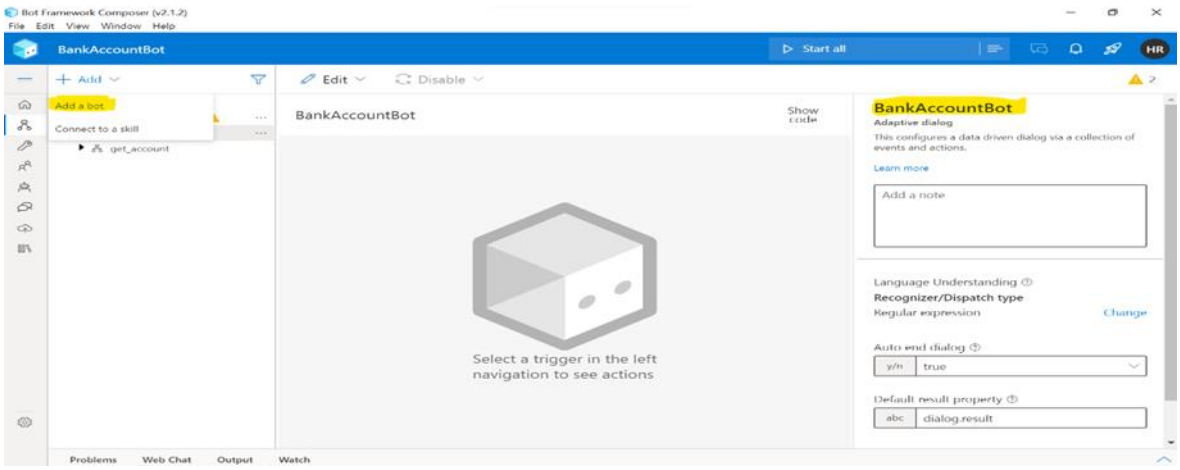


Figure 21 Bot "**BankAcoountBot**"

Now that the bot is created and stored locally on our computer. Let's add other important parts to it. On the Bot Explorer, a Dialog named "**BankAccountBot**" should be available; from that, the new Trigger will be initiated. This will create a Trigger on the Bot Explorer. This is a welcome page the user lands on when the chatbot is started. The name of the Trigger can be edited on the Properties Pane. We named the Trigger "**WelcomeToYour-BankInfo**". This is what a developer sees on the application users don't see any of the Dialogs or Triggers titles.

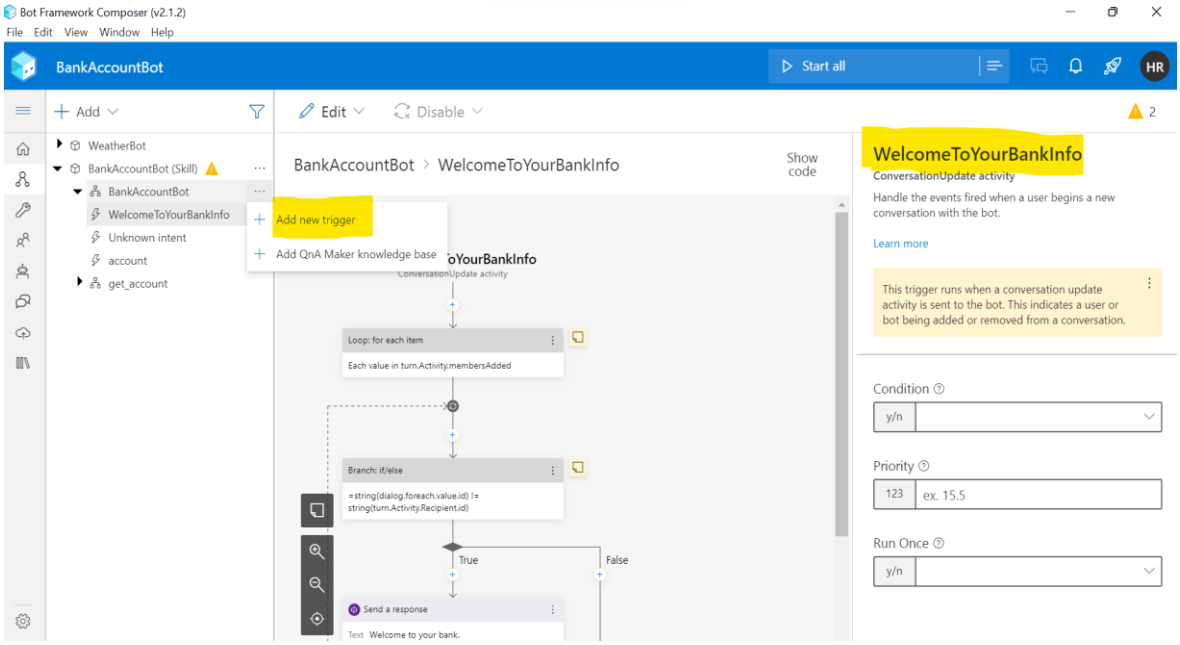


Figure 22 A Trigger "**WelcomeToYourBankInfo"** added to the Dialog.

The above figure (Fig 22) shows that a Trigger is added and named "**WelcomeToYour-BankInfo**". The Trigger is open on the Authoring Canvas and lets us add some features.

Users see that on the first landing page when they communicate with the bot. On the Authoring Canvas, we then select send a response action. The send-a-response action is now opened in Properties Pane.

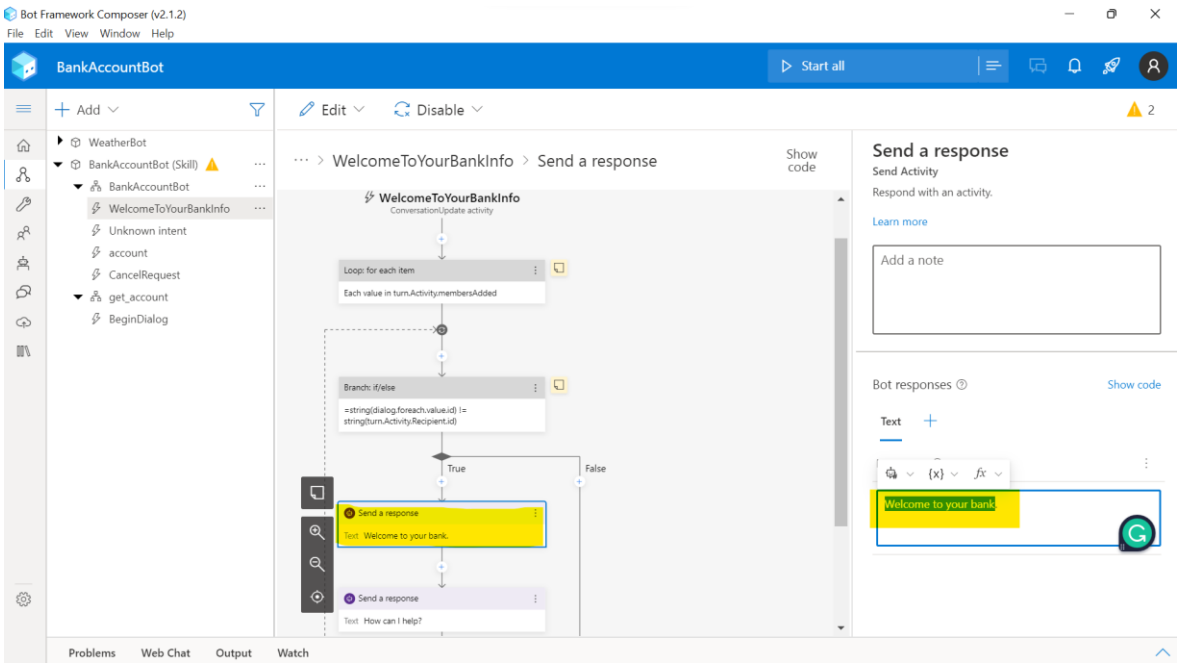


Figure 23 Welcome Text is added to Send a response action.

We can see from the above figure that the send a response action is open on the properties pane, and under the Bot Response, we added a welcoming message "Welcome to your bank". Then we added another send-a-response action by selecting the + button on the authoring canvas under the previous send-a-response action. Finally, we set the text on the properties pane to "How can I help?". So now, the chatbot welcomed the user and asked a question.

5.4 Adding a Dialog to get the bank account info

Now that our bot engages with the user, we can expand its ability by adding Dialogs. The Dialogs need a unique word that triggers the Dialog when the user mentions it. In this case, the triggering word is "account ". So, considering what has been built until now, the user will answer the question "How can I Help?" with any sentence, but it will always include the word "account" because the bot is created to give account info. So, whenever the "account" word is used in the chatbot, it triggers the Dialog, and a response will then be sent back to the chatbot.

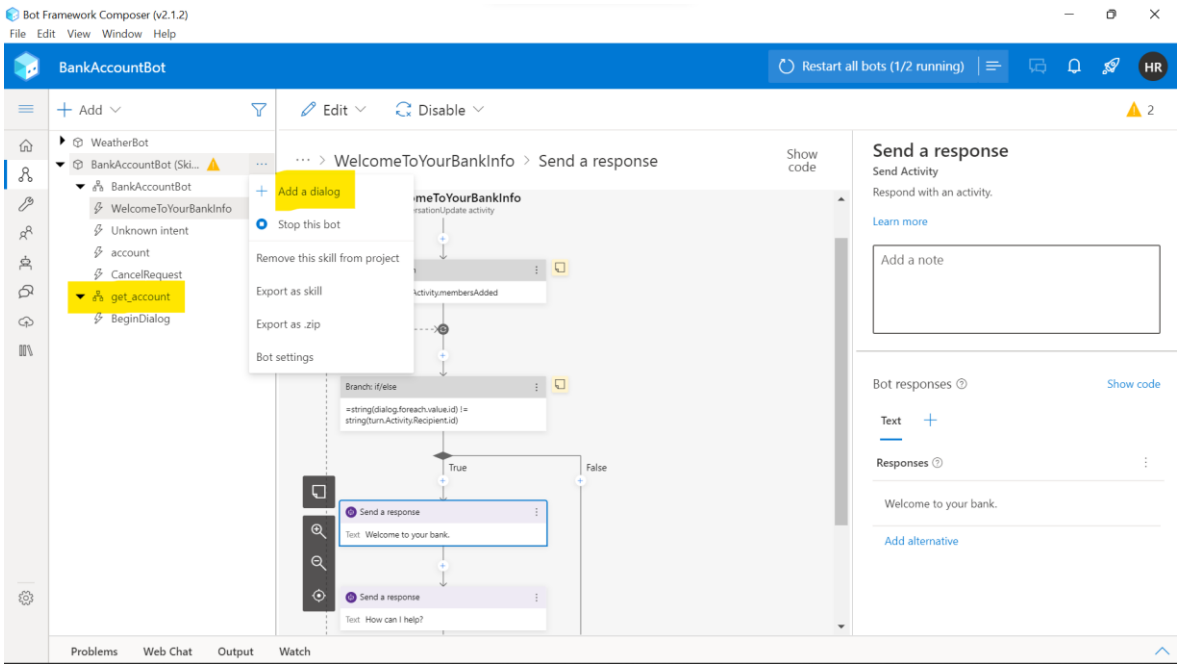


Figure 24 A new Dialog **get_account** is added.

The above figure (Fig 24) shows that the **get_account** Dialog is created from Bot Explorer. On the Properties Pane, the description of the Dialog is kept the same as the Dialog name **get_account**. This is a Dialog triggered whenever the "account" word is found in the user input.

Next, the **BiginDialag** Trigger is selected on the Bot Explorer, and on the Authoring Canva, the send response and prompt action are filled in with the Bot response text. For send response, the text is "Let's check your account", and for the prompt for text, the question "what's your account ID" is initiated.

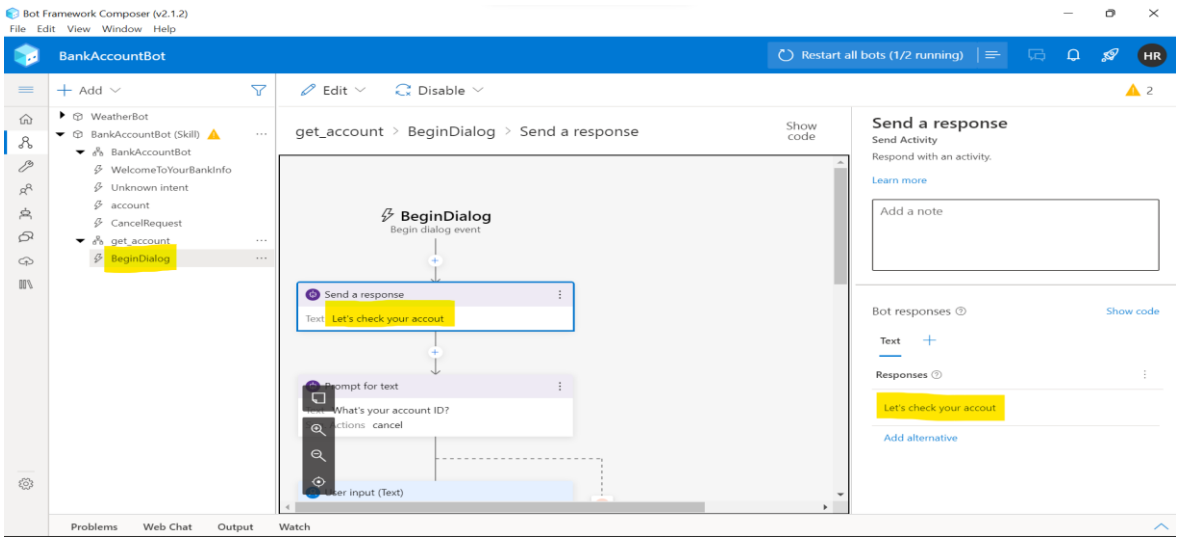


Figure 25 **BeginDialog** triggered, and Bot response text is added.

The above figure (Fig 25) shows that **BeginDialogs** send a response, and the prompt for text actions is edited in a properties pane.

Assume the user has entered a text input in the chatbot with the word "account" in it. The bot transaction would look something like this.

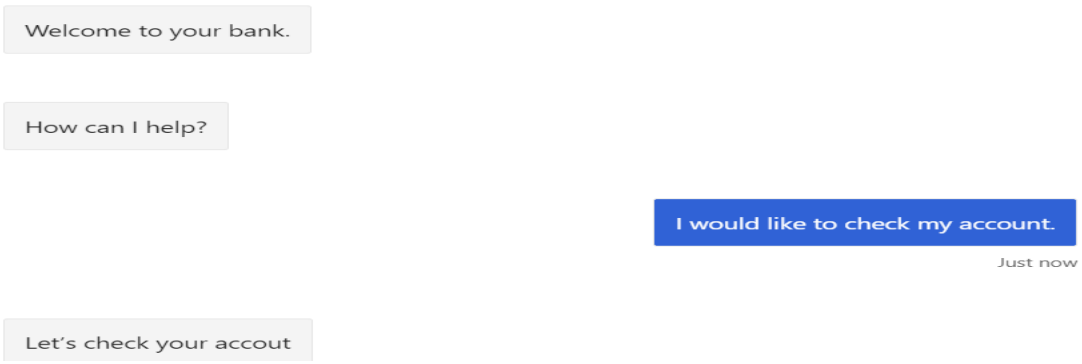


Figure 26 The "**BankAccountBot**" chatbot and user transaction.

After the above transaction, the user will be asked to enter the unique Bank account ID to retrieve the requested information. The user then provides the personal account ID. This input goes directly to the user input section on the Author Canva. To evaluate the user input, we select the user input tab and set the property field to "**user.accountID**" because the user is providing an account ID. On the Output format section, method **"=trim(this.value)**" is added to remove any white spaces from the string. Adding the method is vital in case there are spaces between an input.

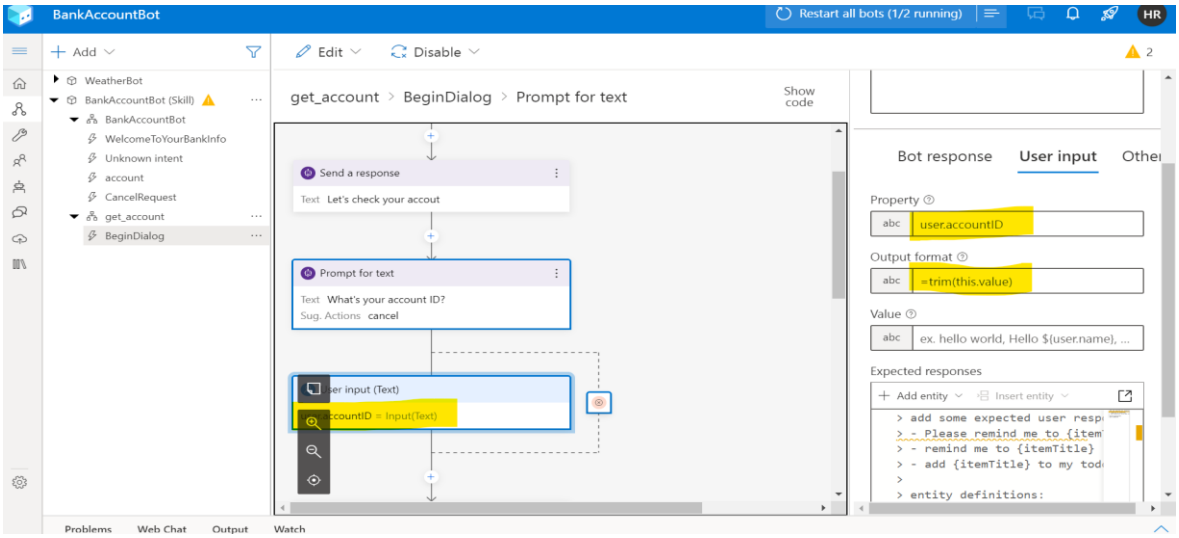


Figure 27 User Input Property and Output Format is set.

Figure 27 shows that the user input is validated as a “**user.accountID”**. Note that the Properties Pane offers Bot response, User inputs and Other when the User input is selected in the Authoring Canvas. From there, the User input Property and Output values are edited accordingly. The “**user.accountID”** validate the user input, and method " =trim(this. value)" removes unnecessary white-spaces.

Next “Other” tab is selected. Under Other, there are Recogniser, Validation, and Prompt configuration section. In our case, we set an Unrecognized Prompt text under Recogniser to "**Sorry, I don't understand '${this.value}'. Please specify the eight-digit account ID in the format 12345678**.".

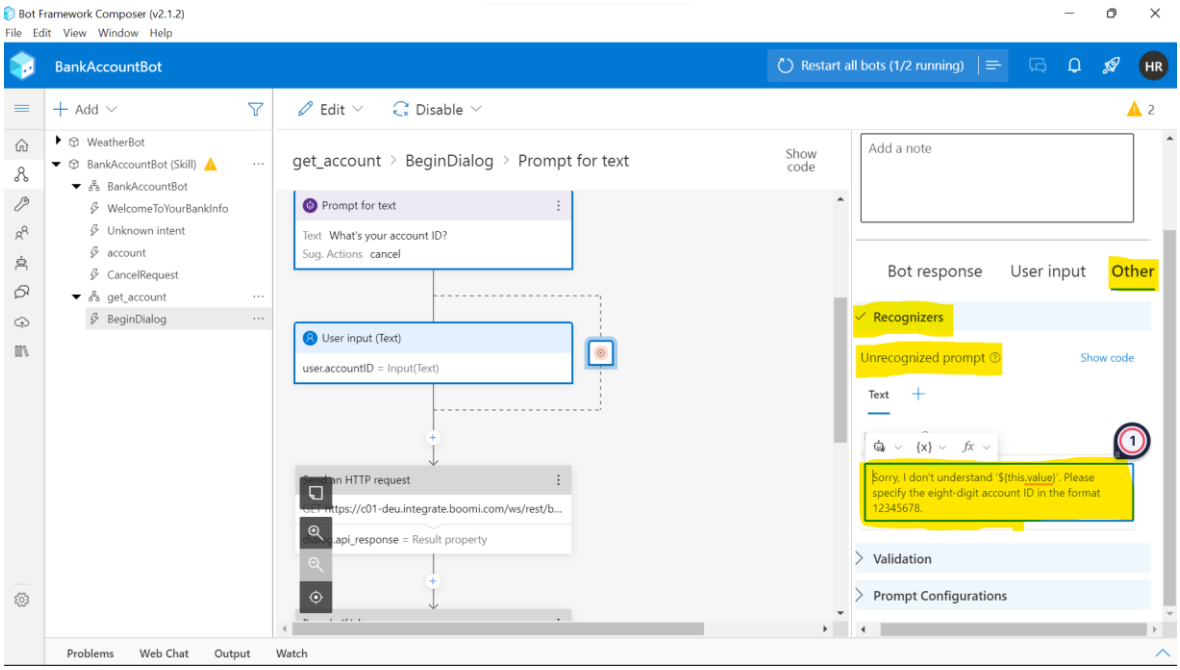


Figure 28 Recognizers' Unrecognized prompt text is set.

This response set on Fig 28 under Unrecognized prompt text will be shown to the user in case unrecognized input is entered. The **"$(this.value)"** is the value the user puts, and the code validates it and gives a response to the user.

Now let's validate the user input. Validation is very important. Setting the input to a fixed digit number is useful, especially when we know the exact digits. Validate the input and showing the right response gives the user information about what is missing from the input.

The bot created accepts only an eight-digit number, which is an account ID. In return, it provides information about an account ID, type and balance. In addition, the bot is now set to respond with an error message in case a wrong input is entered.

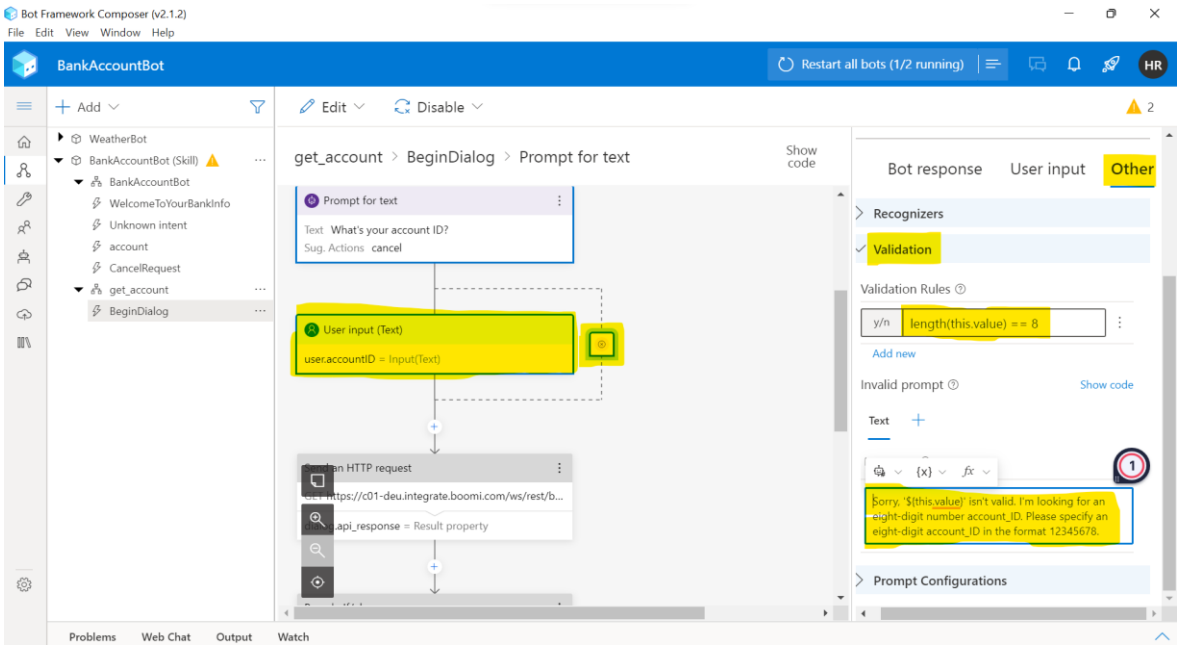


Figure 29 User input validation

In Fig 29, Validation Rules is set to an eight-digit number, and Invalid prompt text is added. Validation Rule set to **"length(this.value) == 8"** limits the users' input to eight digits. What if the input is different? Let's add this text, "**Sorry, '${this.value}' isn't valid. I'm looking for an eight-digit number accountID. Please specify an eight-digit account_ID in the format 12345678.**" to the Invalid prompt section. Now, if the input is other than the eight-digit number registered in the database user will see the text specified.

The Prompt Configuration section lets us add an interruption to the interaction. This option is handy in case the user wants to cancel the interaction. Allows interruption section is set with the" true" value to activate the interruption.

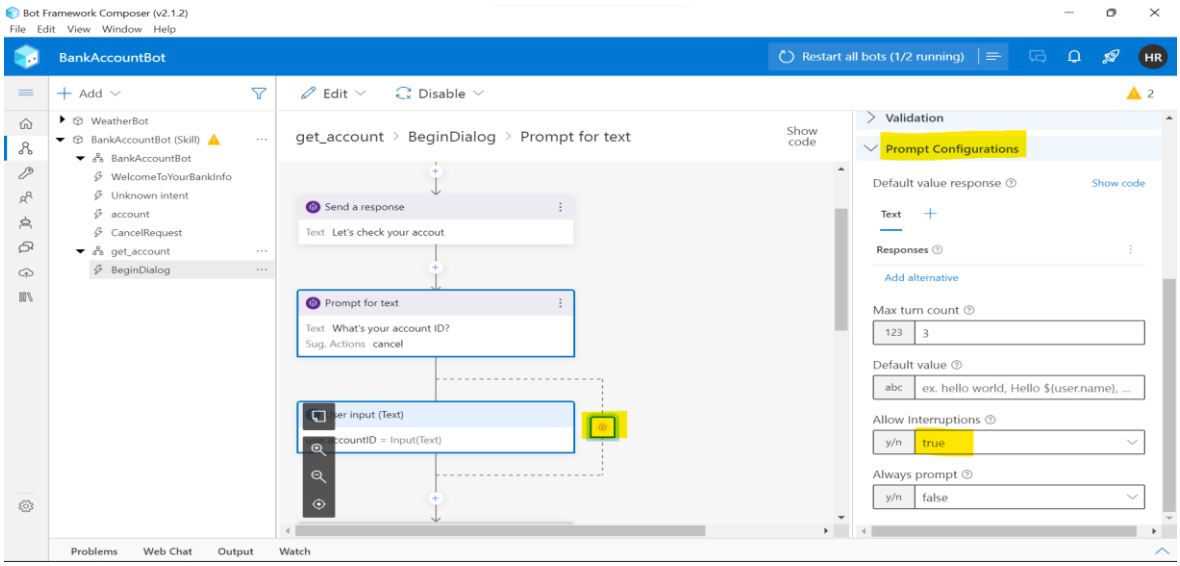


Figure 30 Prompt configuration response section, allow Interruptions set to true.

Figure 30 shows that under Allow Interruption, a "true" option is selected for the Default value response. Prompt Configuration also lets us select actions for Always prompt and Max turn count. The Always prompt value is kept to a "false" because collecting information is not needed when
"Properties" are empty.

5.5 Make an HTTP Request

Now that the basic UI of the chatbot is Implemented, let's expand its capability by adding different actions to the Dialog we created. The "**get_account**" dialogue has a **BeginDialoge** created under it for the bot interaction. In addition, **BeginDialogue** has a triggered word setup. In our case, the Dialog added is triggered when the user mentions" account". This Trigger tells the bot to look for the word " account" in any incoming message. Whenever a user types "account", there will be a prompt asking for their account ID, and that value is saved in “**user.accounID”**. An error message is shown if the user enters a value of fewer than eight characters.

Assume the user entered an eight-digit account ID into the chatbot box. The basic chatbot accepts this value but cannot respond with any information because no dataset is connected to the chatbot. Bot Framework Composer allows us to access a dataset from external resources by sending an HTTP request.

The HTTP methods in API development enable us to GET, POST, PATCH, PUT and DELETE data. The five HTTP methods are the most common regarding data retrieving or

sending from and to a server. To access the data from the database, we need an API. The API has to be understood by Microsoft Azure Bot for us to be able to provide the information the user needs.

As shown in part 5.1, under implementation, Boomi API Management created the API that the Bot Framework Composer understands. Boomi accessed the Cloud database and created an API for our chat. The API output is in an URL form. The Bot Framework Composer uses this URL to get the data and send information to our users. Let's see how it is implemented in Bot Framework Composer.

On the Bot Explorer, the Dialog **BeginDialog** is selected, and on the Authoring Canvas, select the plus button under user input; this will pop up different actions to choose from.

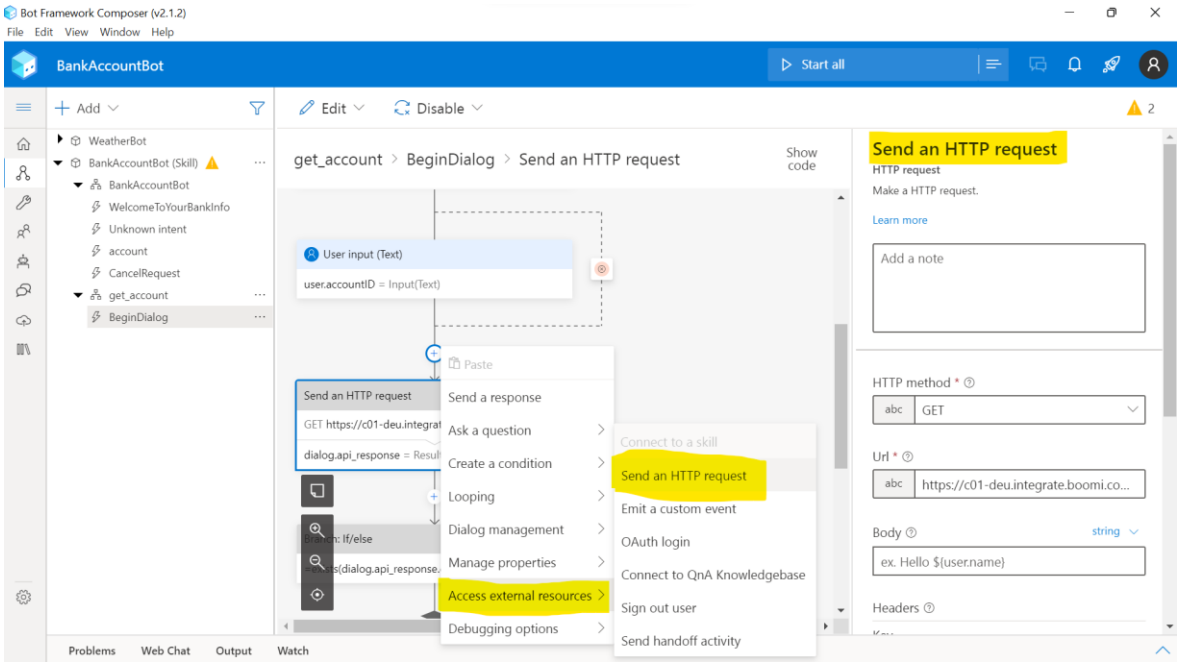


Figure 31 Send an HTTP request to an external resource.

The above figure (Fig 31) shows that the popup lists the "Access external resources" and "Send an HTTP request" selected. This process will create the Send an HTTP request box on the Authoring canvas, and the actions are now defined in the Properties pane.

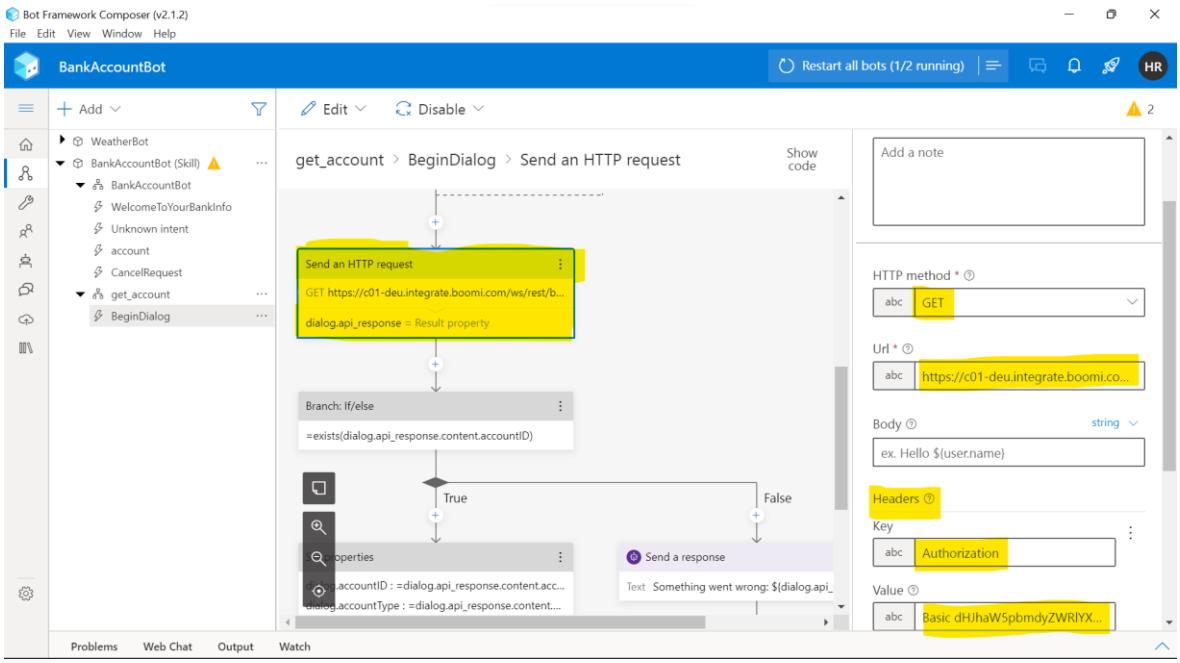


Figure 32 Retrieving the data using a GET HTTP method through URL.

Figure 32 shows that in the Authoring Canvas, Send an HTTP request is selected, and the GET method is chosen inside the HTTP method. The GET method will fetch the data from the SQL database using the API created by Boomi. The API created on Boomi API management is an URL. Therefore, this API is added in the URL box under the HTTP method.

Under HTTP methods:

· The URL is set to "https://c01-deu.integrate.boomi.com/ws/rest/bank/v1/accounts/"

Next, under Headers:

· Key is set to "Authorization".

· value is set to" Basic ------PASSWORD

The above inputs are generated On Boomi API Management. The process can be seen in part 5.1 under implementation.

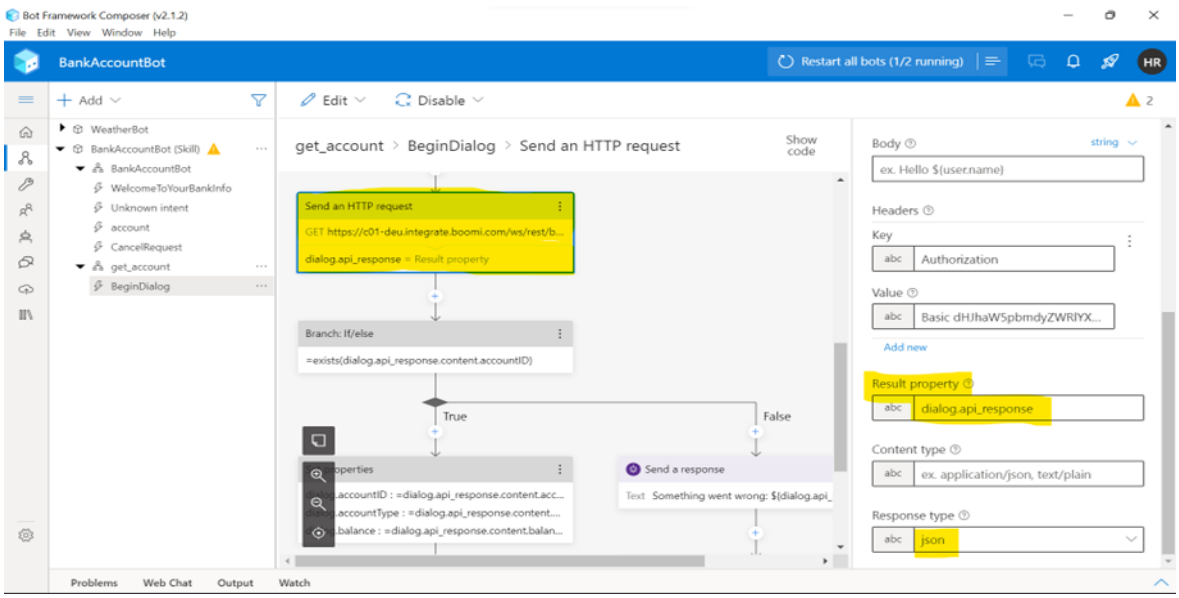


Figure 33 Result property and Response type set.

In Figure 33, the Send an HTTP request is selected, and the Result property is set to "**dailoge_api_response**" this enables the response to create the dialogue from the API added. Since Boomi converted the data from the database to JSON to be able to read by Azure Bot Framework Composer, the Response type" JSON" is selected under the Result property.

After the above setups, the chatbot should now be able to retrieve the data from the database and give the necessary information whenever the user sends a message with the correct account ID.

Now that the cloud SQL data is changed to JSON. The API transfers the data to Azure Bot Framework Composer to give the users the information they need when they put in the correct account ID. The next step is to add logic to the bot. First, the if/else branch is added in the Author Canva under the send HTTP methods by clicking the + button.

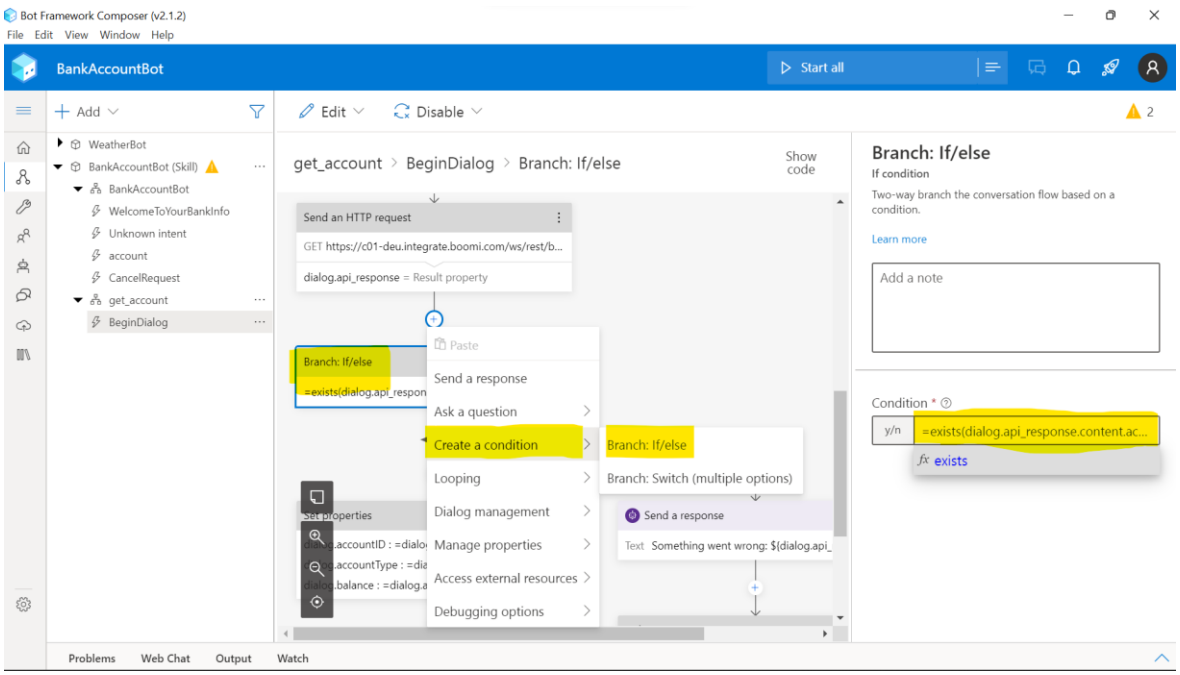


Figure 34 Branch if/else added and condition set.

Fig 34 shows the new Brach if/else created. Depending on the user input, this will take the response in two different directions.

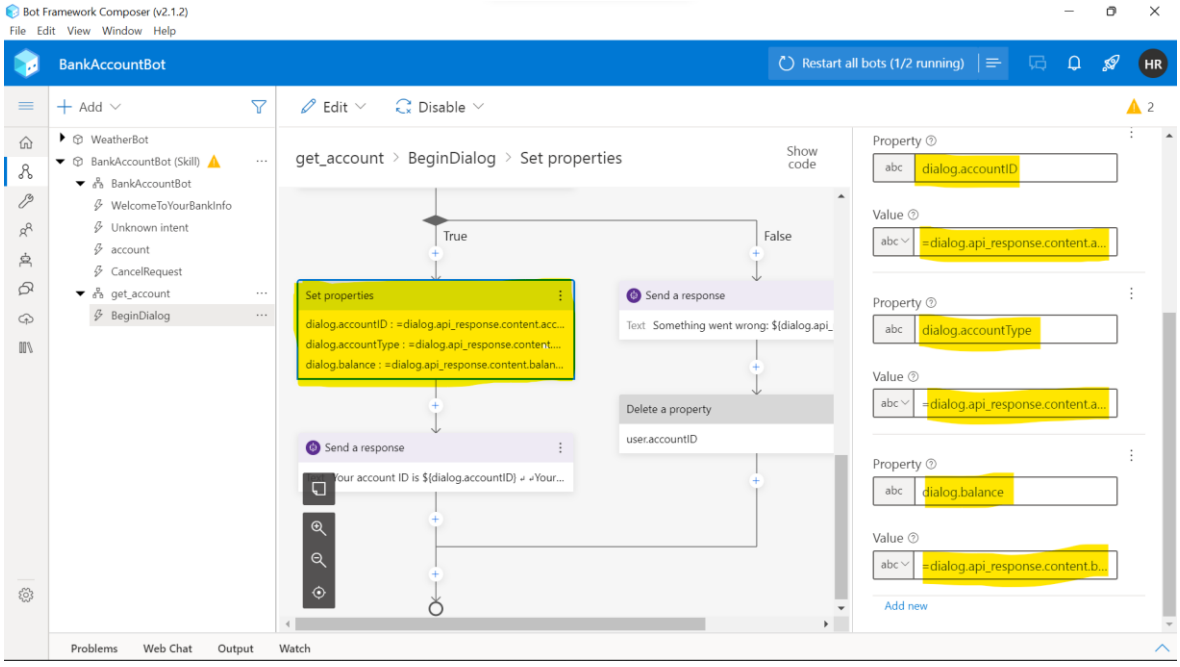


Figure 35 Property and Value added to the if/True Dialog.

As shown in Fig 35, if the user entered the correct value, the logic is gone be "True". Therefore, if the **statusCode** is 200, the return is the account Info. If the **statusCode** is other than that, an error message will be returned.

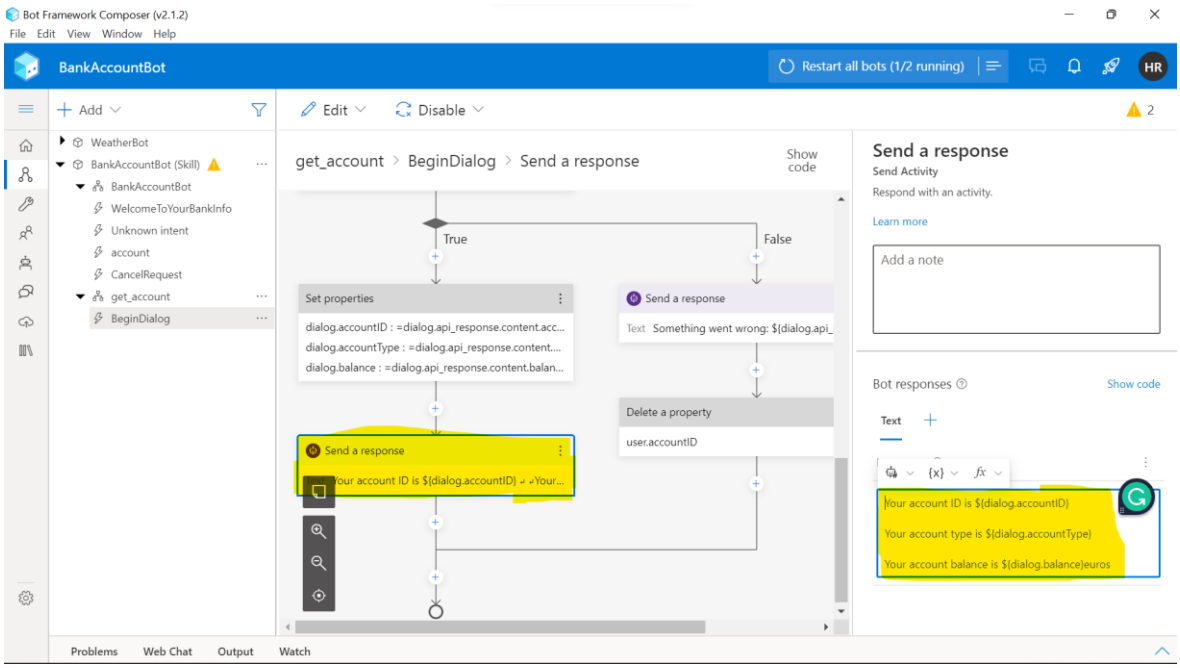


Figure 36 Bot response set for true properties.

If the user enters a correct value and the **statusCode** is 200, the account info displayed in Figure 36 will be delivered to the user.

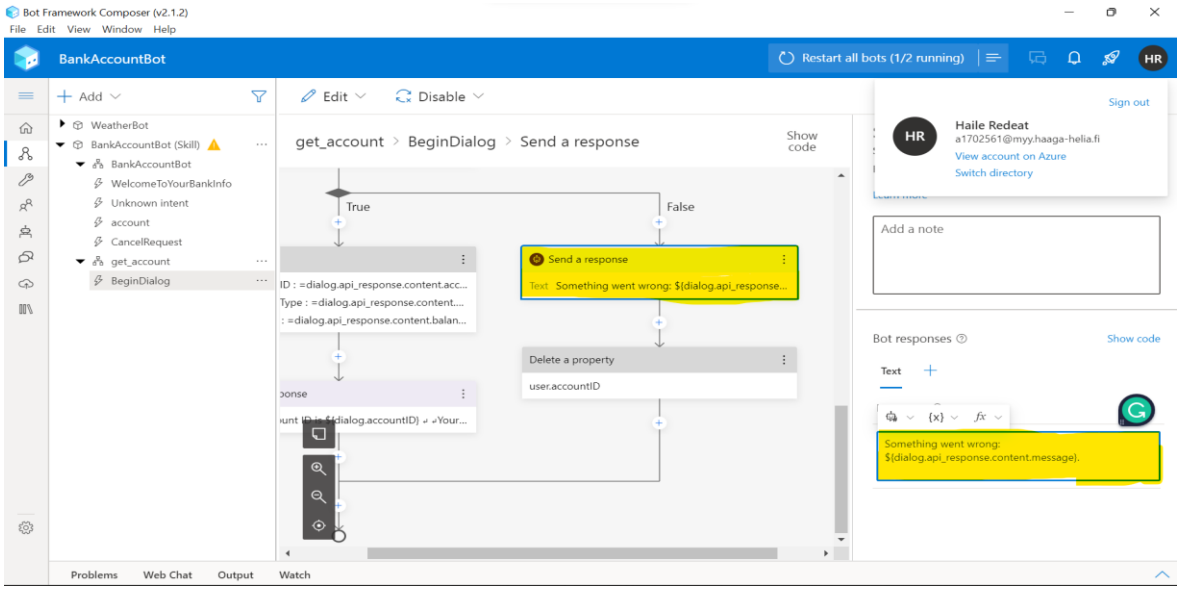


Figure 37 Bot response set for false properties

Else the user entered an incorrect value, and the Bot response seen in Figure 27 will be delivered to the user. Note that there is a "Delete property" in the Authoring Canva under send a response. The "Delete a property" delete the input after any wrong value is entered.

6 Testing and Publishing

The bot is tested using the Bot Framework Emulator. First, the Bot Framework Emulator is downloaded, and then the Emulator needs to be run through the Bot Framework Composer. As soon as we lunch it from the composer, we should see the chatbot's welcome page.

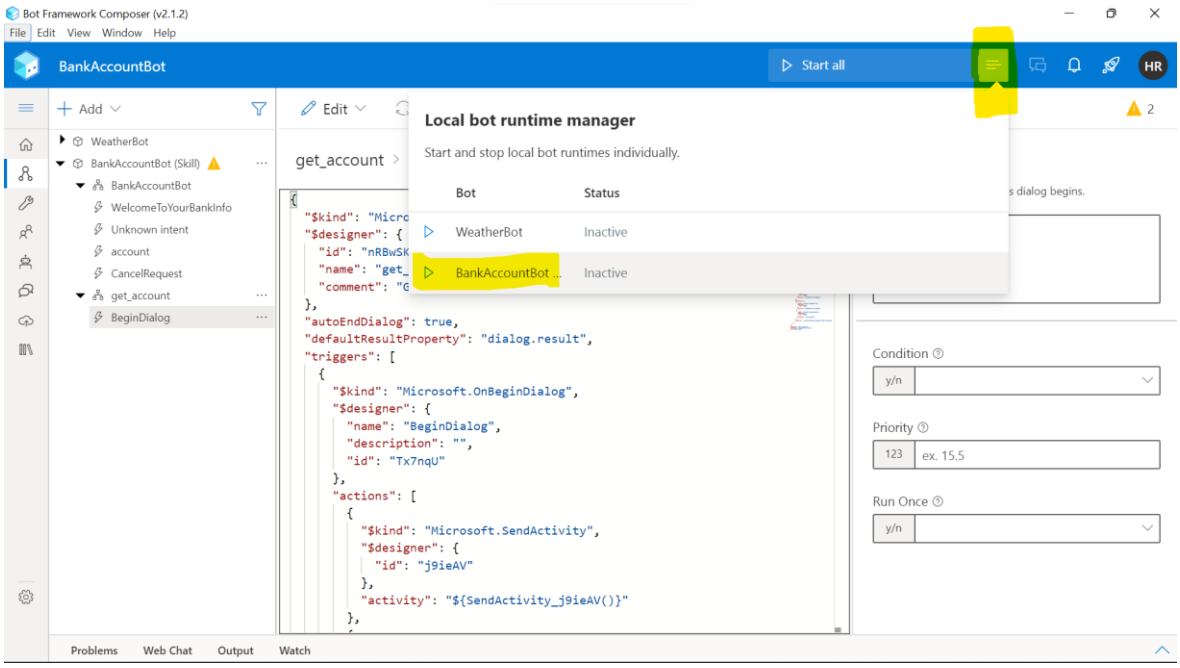


Figure 38 Launching "**BankAccountBot"** from Bot Framework Composer.

In figure 38, **"BankAccountBot"** is selected from the dropdown button.

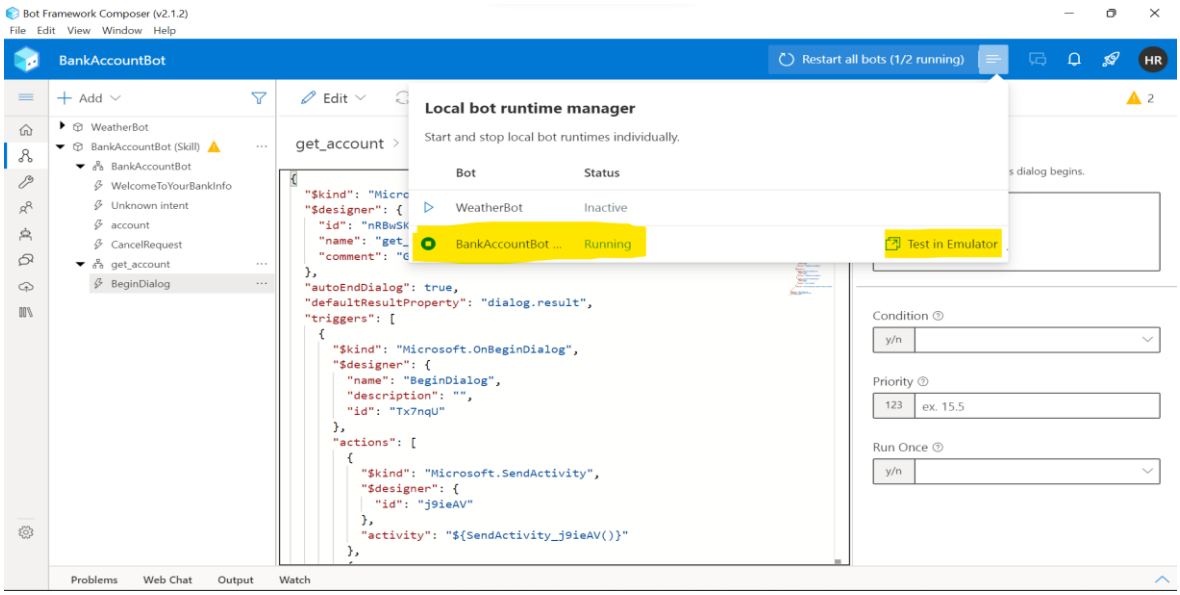


Figure 39 "**BankAccountBot"** is running and ready to be tested in Emulator.

In the above figure 39, it's shown that the bot is started and it's running. Now that it's running, we click on the Test in Emulator; as soon as we click that, it will take us to the Bot Framework Emulator.

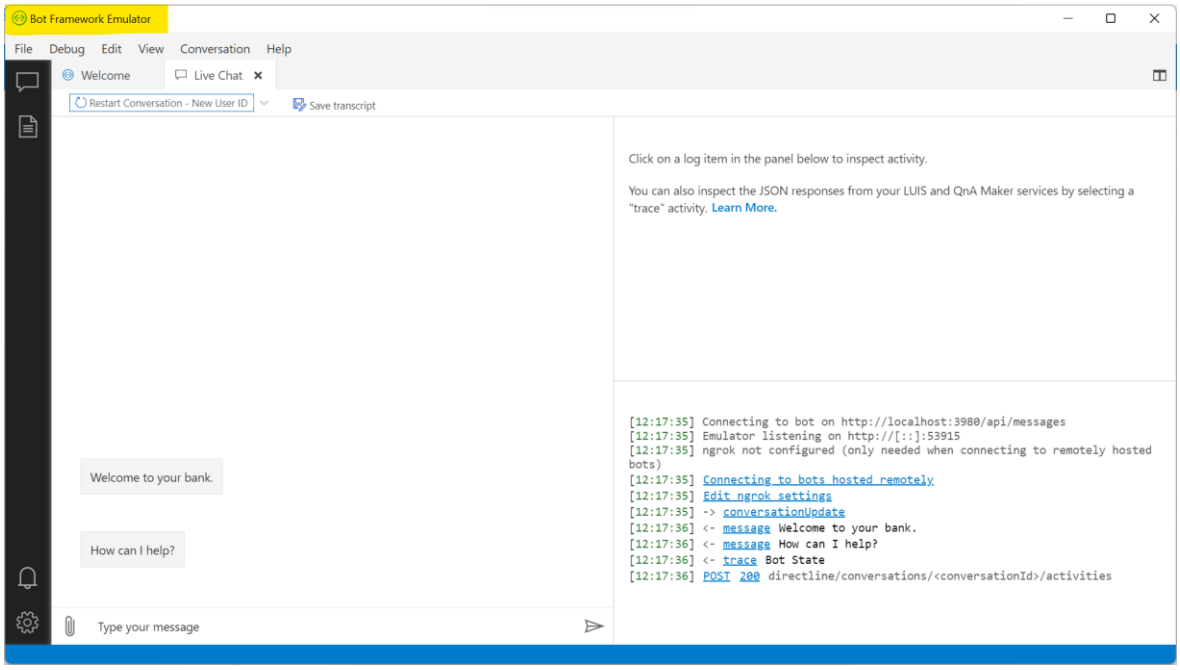


Figure 40 “**BankAccountBot”** Is now running In Bot Framework Emulator.

Now that the bot is running in Bot Framework Emulator. We should be able to see the welcoming message followed by a prompt question. Look at the result in Fig 40.

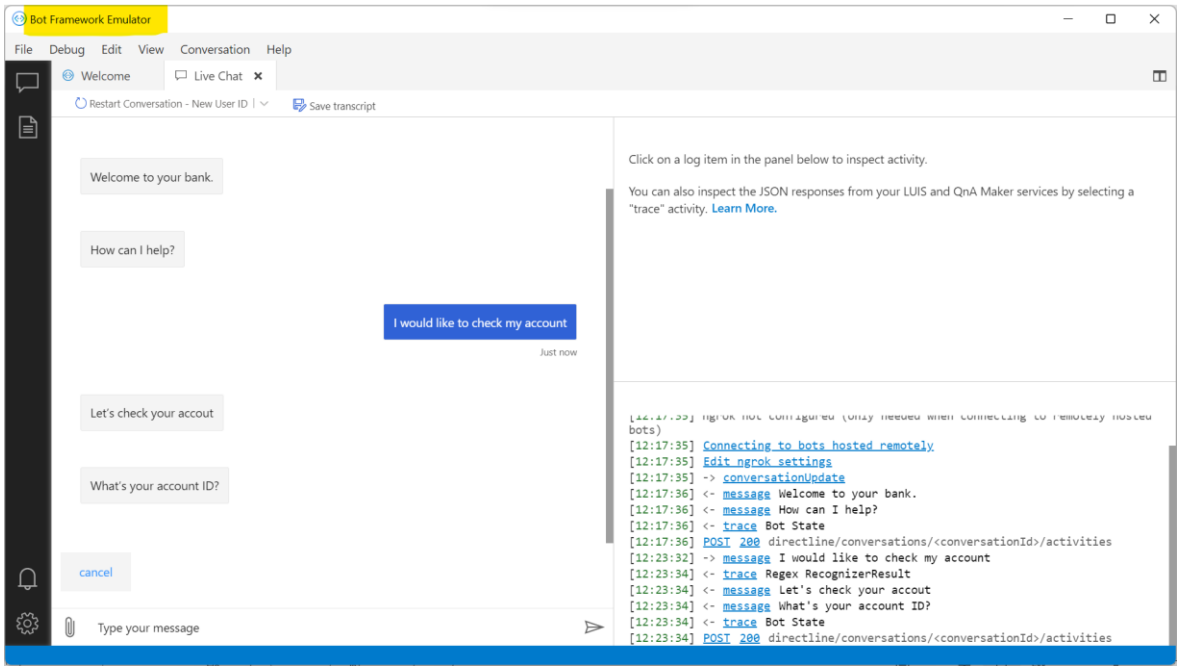


Figure 41 **“BankAcountBot”** and user interaction.

Fig 41 shows that the user inputs a text asking for account info. Then the Bot gives information on what actions it will take and sends another question asking the user to enter the unique account ID. Note that there is a "cancel" under the bot question. If users don't want to continue with the interaction, they can cancel it.

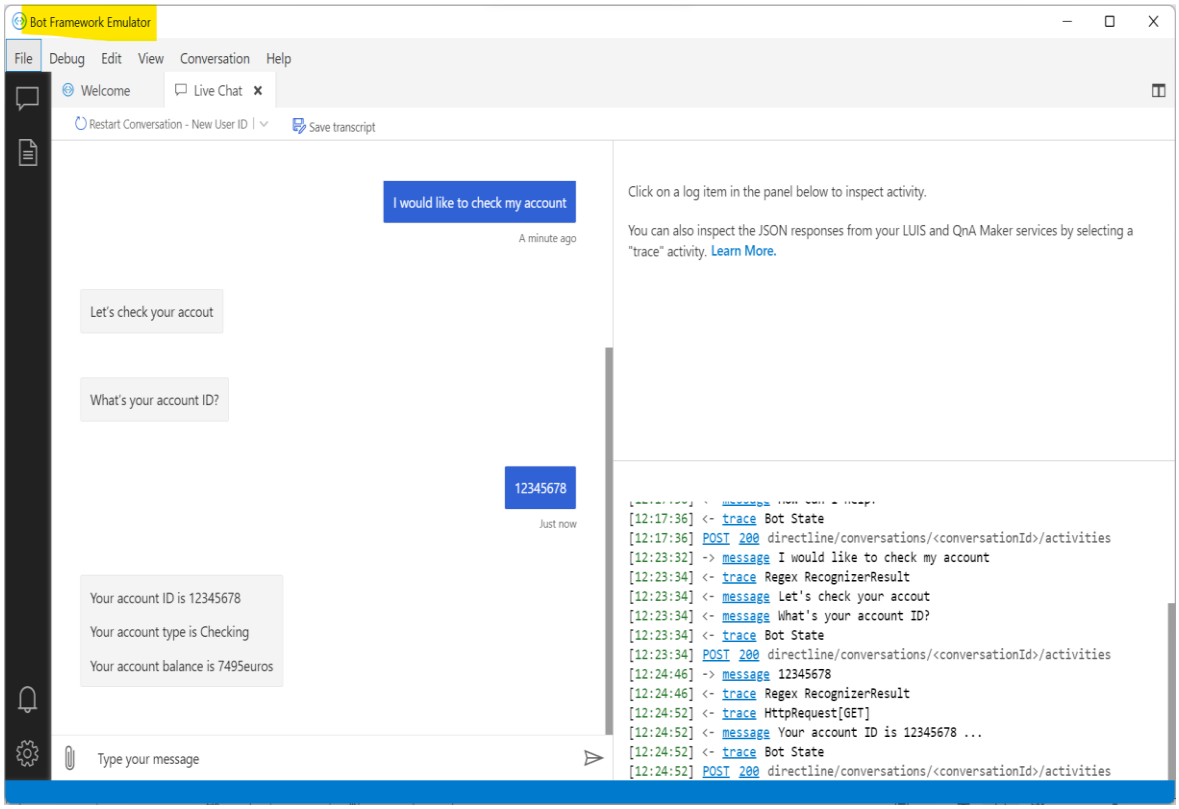


Figure 42 “**BankAccountBot”** shows the required info to the user.

In the above Figure (Fig 42), it's displayed that the user entered the correct unique account ID, and the account ID, account type and balance are shown. This is the info fetched from a database.

Users have unique account IDs in the database. The Bot only responds to those unique account IDs; otherwise, the error message displays.

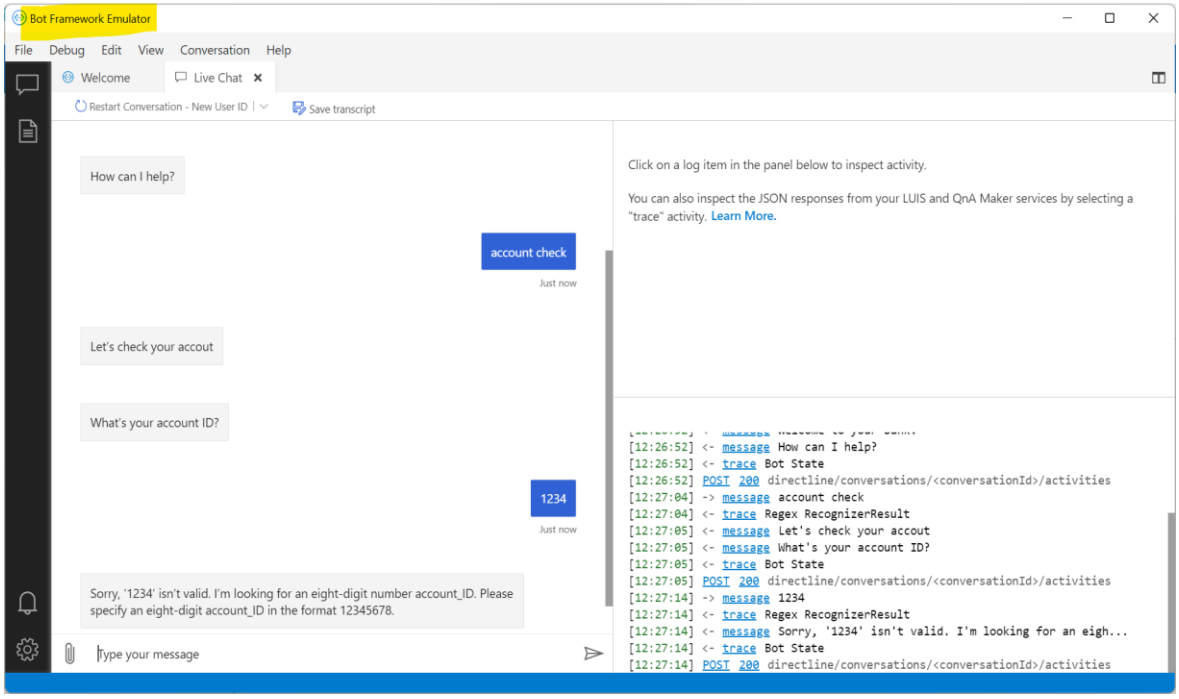


Figure 43 "**BankAcoountBot**" sends an error message.

Figure 43 shows that the user input digits are not recognized by the bot not, so the user is now seeing a message asking to put in the correct number of digits.

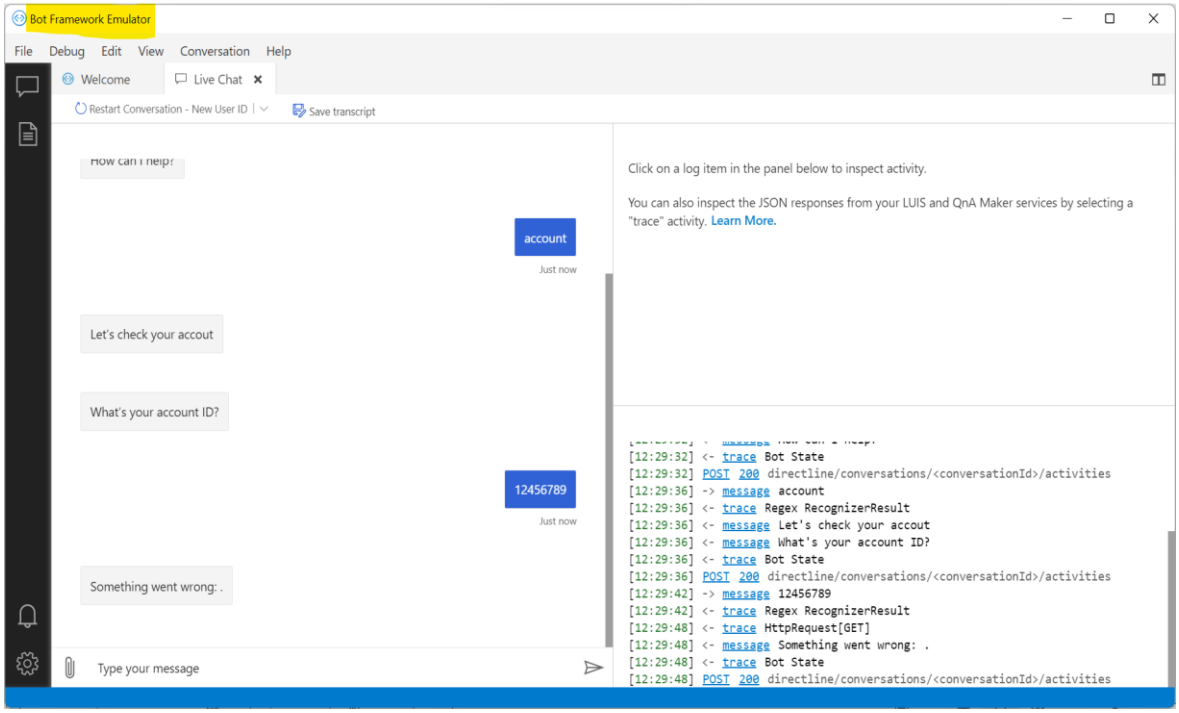


Figure 44 "**BankAcoountBot**" couldn't find the account ID in the database.

Figure 44 demonstrates the user entered eight-digit character, but the Bot could not give back the information needed. The reason for that is there was no data related to that numbers. So, the info was not retrieved even if the character digit was correct.

The bot is now tested and working. The next step is to publish the bot to any messaging platform or website. The azure platform provides a resource to deploy a bot created on

Azure to any platform or website. The process starts with the Bot Framework Composer and takes us to Bot Framework Emulator.

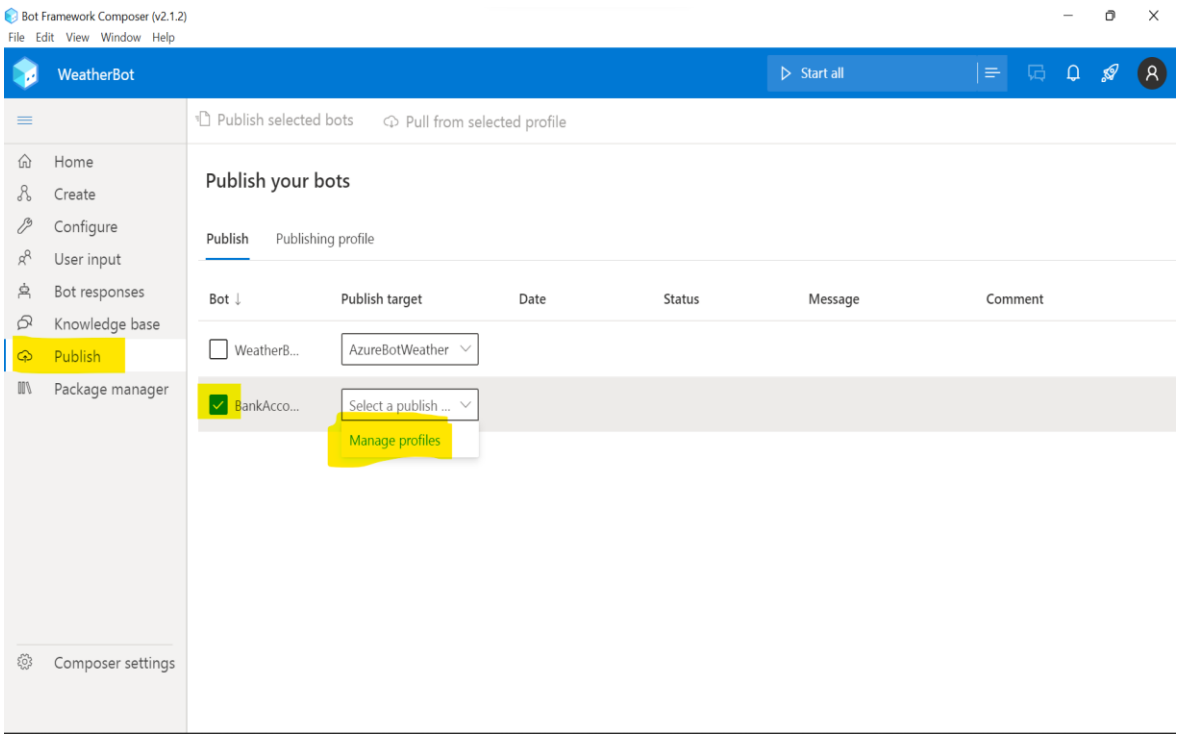


Figure 45 Publishing the "**BankAcoountBot**" bot.

As shown in Fig 45, we navigate to Publish the Navigation Pane and select the bot created, "**BankAccountBot**", then from the dropdown button; manage profiles are selected. This will create a publishing profile. Check Fig 46 for the result.

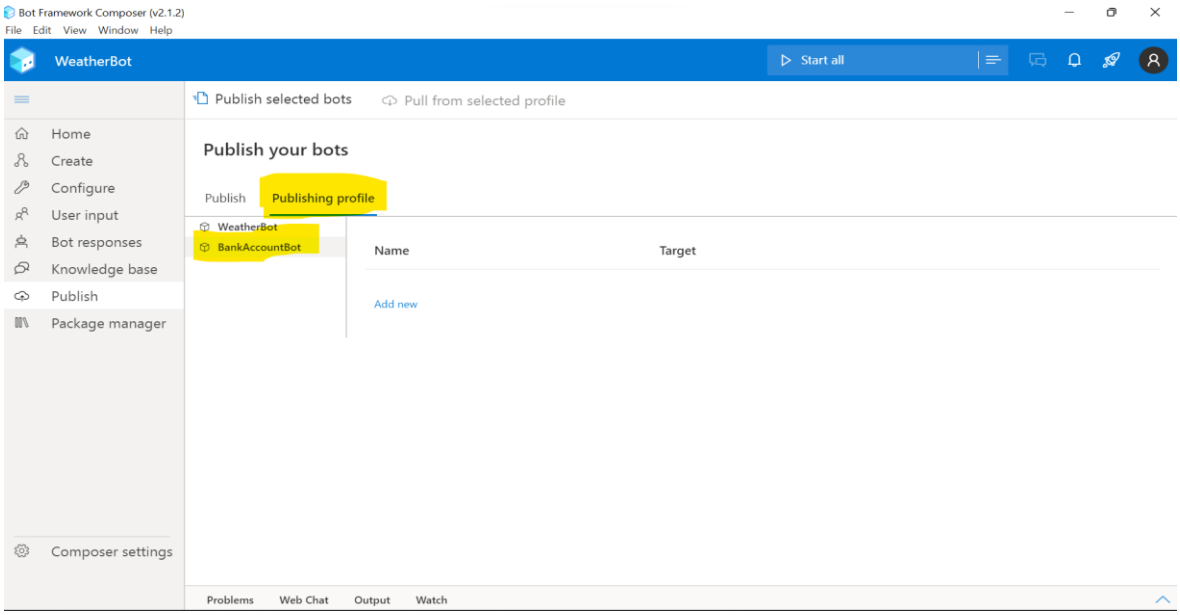


Figure 46 Publishing Profile

Figure 46 shows that the Publishing profile is selected. The next step is to click Add new on the Authoring Canva, which will take us to the Publishing Account creation page.

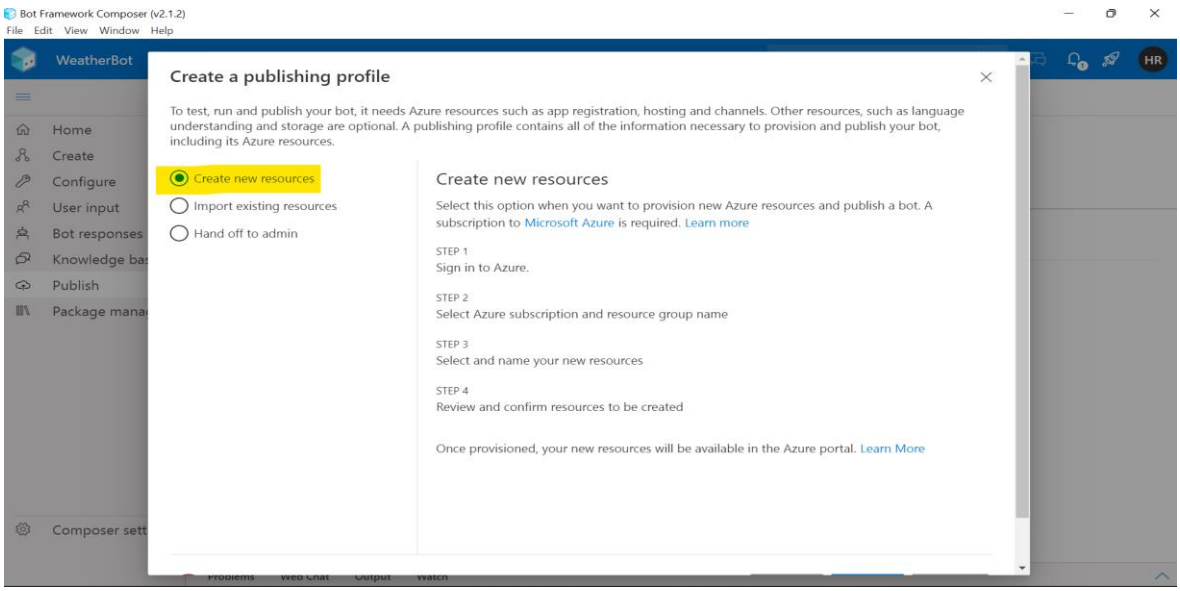


Figure 47 Resource creating.

The display in Fig 47 shows us to publish the bot, we need to create a resource for the subscription we have. Creating the resource is not free. After the resource is created, it will take us to the previous page. In Figure 48, a different resource profile is used to show the last step of publishing.

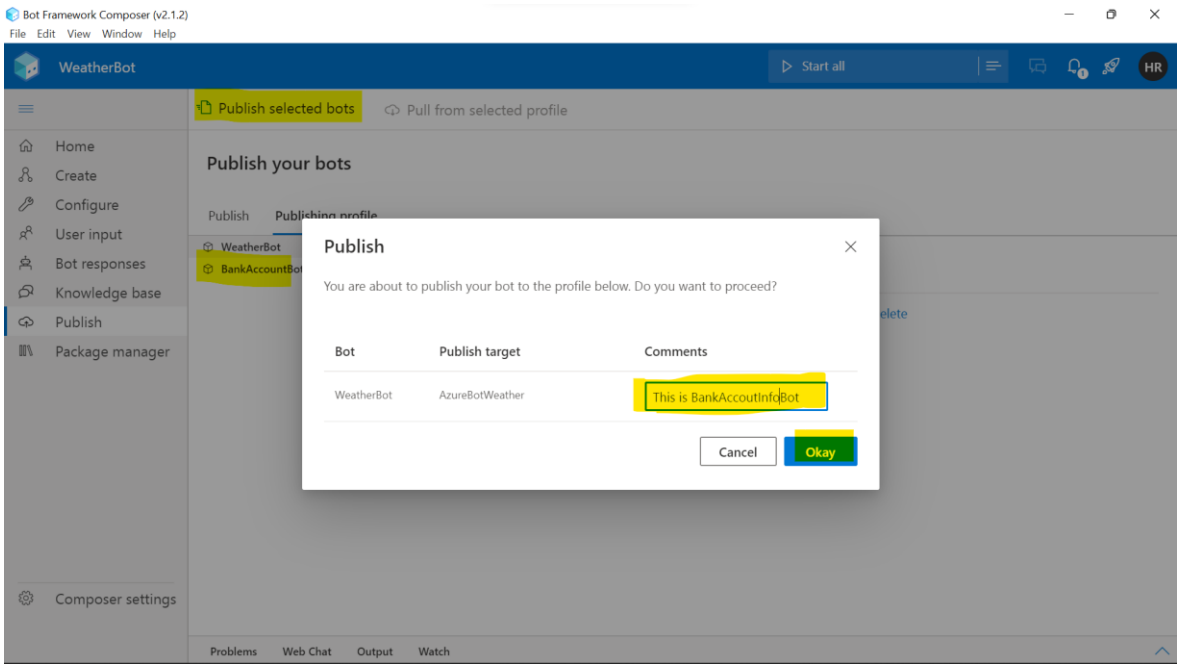


Figure 48 Publishing "**BankAcoountBot**"

Figure 48 shows the final stage of the publishing process. After selecting the “Publish selected bots”, another page will pop up with our bot name, target resource to be published and a comment box. After adding the comment and the “okay” selected, our Bot is published on the Azure portal. From the portal, we can connect it to any messaging platform or website.

Note that the resource profile name "WeatherBot" on Fig 25 differs from the one created in this project. It's just used to show the last step of the publishing process.

7 Discussion

The Bot created is functional. The data used was available on a clod SQL database. Both platforms, Microsoft Azure, and Boomi Integration, are available for free. The research is done following Haaga-Helia guidelines.

The Bot developed used simple data as a showcase for the implementation. The data used is available in a cloud database. In the future, the author can use an Excel sheet to create data and store it in a database, then create an API using that data. The API then be used to fetch data to the Bot. Also, the Bot hasn't been published to any platform because the resource hasn't been bought, so in the future, we can add different cognitive services like pictures and voices and publish it.

The researcher has gained vast knowledge of thesis writing and product implementation. In addition, different platforms, and tools like Boomi Integration, KNIME, Microsoft Azure Bot Framework Composer, Bot Framework Emulator, and Azure Portal have been practised.

8 Conclusion

Based on the finding and practice, the researcher concludes that a low-code AI chatbot can easily be deployed and gives an organization many advantages. The research shows that Microsoft Azure Bot Framework Composer is the best option for creating and deploying a low-code AI chatbot on any platform or website. The most significant advantage of using a Microsoft Azure Bot Framework Composer for creating a chatbot is its easiness and flexibility. Anyone with no programming background can build a chatbot in the Microsoft Azure Bot Framework Composer. It's a Streat forward step. For anyone who doesn't understand what this means, think of it like you are building a website using a drag-and-drop option in any website-building platform.

Microsoft Azure Bot Framework Composer has so much flexibility Compared to the other options like Microsoft Powe Virtual agent (a no-code) and Bot Framework SDK, which needs programming skills. Its flexibility empowers developers with an alternative to open the code and add extra functions whenever necessary. So, companies or individuals could quickly deploy and use the chatbot without prior programming skills. And later, if needed, a developer can add any features available in any other implementing methods.

Using Microsoft Azure bot Framework Composer for chatbot building is much simpler, cheaper, and more versatile, considering the options it offers compared to AWS and HubSpot. The only downside is the cost of using the service for individuals and companies. But it's more than reasonable considering the benefit of having a chatbot, whether the aim is customer satisfaction or reduced workload.

In conclusion, using a low-code platform for chatbot development offers many benefits, including the ability to create chatbots quickly and easily without needing to write a lot of code and the ability to integrate the chatbot with other systems and services easily. Therefore, as low-code platforms continue to grow, we will see more and more chatbots being developed using this approach.

References

Abrahamsson, P., & Graziotin, D., 2013. Making Sense Out of a Jungle of JavaScript Frameworks. Product-Focused Software Process Improvement, s. 335. Accessed: 10 January 2022.

AbuShawar, B., & Atwell, E., 2015. ALICE Chatbot: Trials and Outputs. Computacion. Y Sist, 19, s. 4. Accessed: 20 December 2021.

AWS 2018. Building a conversational business intelligence bot with Amazon Lex. Sample Interaction with BIBOT. URL: https://aws.amazon.com/blogs/machine-learning/building-a-conversational-business-intelligence-bot-with-amazon-lex/. Accessed: 14 September 2022.

Bala, K., Kumar, M., Hulawale, S., & Pandita, S., 2017. Chat-Bot For College Management System Using A. I. Int. Res. J. Eng. Technol. (IRJET), 4, s. 1. Accessed 15 December 2021.

Bocchi, E., & Mellia, M., 2014. Cloud Storage Service Benchmarking: Methodologies and Experimentations. IEEE 3rd International Conference on Cloud Networking. Accessed: 15 March 2022.

Boomi 2022a. Atmosphere Documentation. Environment management. URL: https://help.boomi.com/en-US/bundle/integration/page/c-atm-Environment_management.html. Accessed: 25 October 2022.

Boomi 2022c. Atmosphere Documentation. Map Function Components. URL: https://help.boomi.com/en-US/bundle/integration/page/r-atm-Map_Function_components.html: Accessed: 25 October 2022.

Boomi 2022b. Atmosphere Documentation. Profile components. URL: https://help.boomi.com/en-US/bundle/integration/page/c-atm-Profile_components.html. Accessed: 25 October 2022.

Buntak, K., Kovacic, M., & Mutavdzija, M., 2017. Application of Artificial Intelligence in the Business. International Journal for Quality Research, 15(2), s. 406. Accessed: 28 March2022.

Chaubey, M. 15 November 2021. Why do you choose Boomi as your iPaaS Solution? STREMS TRANSFORMATION THROGH TECHNOLOGY. URL: https://www.streamsolutions.com/why-do-you-choose-boomi-as-your-ipaas-solution/. Accessed: 25 October 2022.

Choi, S.W., & Nam, J.H., 2019. The use of AI chatbot as an assistant tool for SW education. Journal of the Korea Institute of Information and Communication Engineering, 23(12), s. 1694. Accessed: 28 February 2022.

Christodoulou, S., Gizas, A., & Papatheodorou, T., 2012. Comparative Evaluation of JavaScript Frameworks. In Proceedings of the 21st International Conference on World Wide Web, s. 514. Accessed: 7 March 2022.

Falgout, J. 9 January 2020. KNIME and AWS Machine Learning Service Integration. Open for Innovation KNIME. URL: https://www.knime.com/blog/knime-on-aws. Accessed 25 March 2022.

HubSpot 2022. Shared Inbox for Customer Conversations. URL: https://blog.hubspot.com/service/customer-service-chatbots. Accessed: 28 Oct 2022.

Insik, K., Jung, J., Todd, F., Tristan, H., & Dennis, P., 2012. Cloud computing for comparative genomics with windows azure platform. Evolutionary bioinformatics online, 8(1), s. 527. Accessed: 20 February 2022.

James, A. 26 Feb 2022. Allen 1995: Natural Language Understanding. AuthorZilla. [Allen 1995: Chapter 1 - Introduction / 1]. Accessed 16 March 2022.

Jarocinski. K. 25 August 2020. Low Code Chatbots – The Fastest Way to Implement Chatbots. Netguru. URL: https://www.netguru.com/blog/low-code-chatbots-the-fastest-way-to-implement-chatbots. Accessed: Oct 29 202.

Johnston, S.J., O'Brien, N.S., Lewis, Hart, E.E., White, A., & Cox S.J. 2013. Clouds in Space: Scientific Computing using Azure. Journal of Cloud Computing. Accessed: 1 March 2022.

Karthik, G. 21 February 2022. What is a Low-Code Conversational AI Bot Builder Platform For Business. Gnani.ai. URL: https://www.gnani.ai/resources/blogs/low-code-conversational-ai-bot-builder-platform-for-business/. Accessed: 12 September 2022.

Khan, W. 6 April 2021. The KNIME Server REST API. Open for Innovation KNIME. URL: https://www.knime.com/blog/the-knime-server-rest-api. Accessed: 25 December 2021.

Kim, S.G., Shin, M.C., & Kang, J.Y., 2018. Chatbot technology introduction and case analysis. KICS Information& Communication Magazine, 35(2), s. 24. Accessed: 1 May 2022.

Kumar, R., & Ali, M.M., 2020. A Review on Chatbot Design and Implementation Techniques. Int. J. Eng. Technol, 7, s.11. Accessed: 25 December 2021.

Kim, S.G., Shin, M.C., & Kang, J.Y., 2018. Chatbot technology introduction and case analysis. KICS Information& Communication Magazine, 35(2), s. 24. Accessed: 1 May 2022.

Lausch, A., Tischendorf, L., & Schmidt, A. 2015. Data mining and linked open data—New perspectives for data analysis in environmental research. Ecol. Model, 295, s. 7. Accessed: 8 March 2022.

Loper, E., & Bird, S., 25 March 2002. NLTK: The natural language toolkit. URL: https://www.nltk.org/. Accessed 16 march 2022

Maly, J. 21 January 2022. Chatbots are becoming more efficient, the results couldn’t be better. Digital Signage Today. URL: https://www.digitalsignagetoday.com/blogs/chatbots-are-becoming-more-efficient-the-results-couldnt-be-better/. Accessed: 20 February 2022.

Mariano, C.L., 2017. Benchmarking JavaScript Frameworks. Master’s dissertation. Dublin Institute of Technology, Dublin, Ireland. URL: https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1100&context=scschcomdis. Accessed: 11 March 2022.

Mehta, D., Mangal, P., & Jain, N., 2015. AngularJS: A modern MVC framework in JavaScript. Journal of Global Research in Computer Science, 5(12), s. 19. Accessed: 28 February 2022.

Microsoft 2019. Components of a conversational AI experience. URL: https://dev.botframework.com/. Accessed 15 October 2022.

Microsoft 2022. Improve Customer Engagement and Productivity with Conversational AI. URL: https://learn.microsoft.com/en-us/events/all-around-azure-all-around-azure-a-developers-guide-to-ai/improve-customer-engagement-and-productivity-with-conversational-ai. Accessed 15 October 2022.

Myers, B. A, & Oney, S., 2009. Fire Crystal: Understanding interactive behaviours in dynamic web pages. In 2009 IEEE Symposium on Visual Languages and Human-Centric Computing, s. 106. Accessed: 8 February 2022.

Okuda, T., & Shoda, S., 2018. AI-based Chatbot Service for Financial Industry. *FUJITSU* Sci. Tech. J, 54, s. 5. Accessed: 10 January 2022.

Peter, M., & Grance, T., 2011. The NIST Definition of Cloud Computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. Accessed: 10 January 2022.

Rebernik, D. October 2012. Definition Dell Boomi. TechTarget. URL: https://www.techtarget.com/searchcloudcomputing/definition/Dell-Boomi#:~:text=Dell%20Boomi%20AtomSphere%20is%20an,cloud%20and%20on%2Dpremises%20applications. Accessed: 20 October 2022

Reynolds, B. 01 April 2020. Angular vs, Aurelia: Who wins? Baytech consulting. URL: https://www.baytechconsulting.com/blog/angular-vs-aurelia. Accessed: 01 April 2022.

Shum, H.Y., He, X.D., & Li, D., 2018. From Eliza to XiaoIce: Challenges and opportunities with social chatbots. Front. Inf. Technol. Electron. Eng, 19, s. 10-26. Accessed: 15 January 2022.

Souders, S., 2008. High-performance Web Sites. Communication of ACM, 51(12), s. 38.
Srinath, K. R, 2017. Python – The Fastest Growing Programming Language. In International Research Journal of Engineering and Technology, s. 2396. Accessed: 17 March 2022.

Tiwari, P. 8 June 2022. AWS Chatbot. CODING NINJAS. URL: https://www.codingninjas.com/codestudio/library/aws-chatbot. Accessed: 8 June 2022.

Turing, A.M., 1950. Computing Machinery and Intelligence. Mind, LIX, s. 433-460. Accessed: 20 January 2022.

Viraktamath, S.V., Shet, C.P., & Nayak, P.R., 2016. Artificial Intelligence and its Application in Speech Recognition. Bonfring International Journal of Research in Communication Engineering, 6, s. 49. Accessed: 21 February 2022.

Zhou, L., Gao, J. Li, D., & Shum, H.Y., 2020. The Design and Implementation of XiaoIce an Empathic Social Chatbot. MIT Press Direct. 46 (1), s. 53–93. Accessed 16 January 2022.

Appendices

Appendix 1. Microsoft Azure Platform

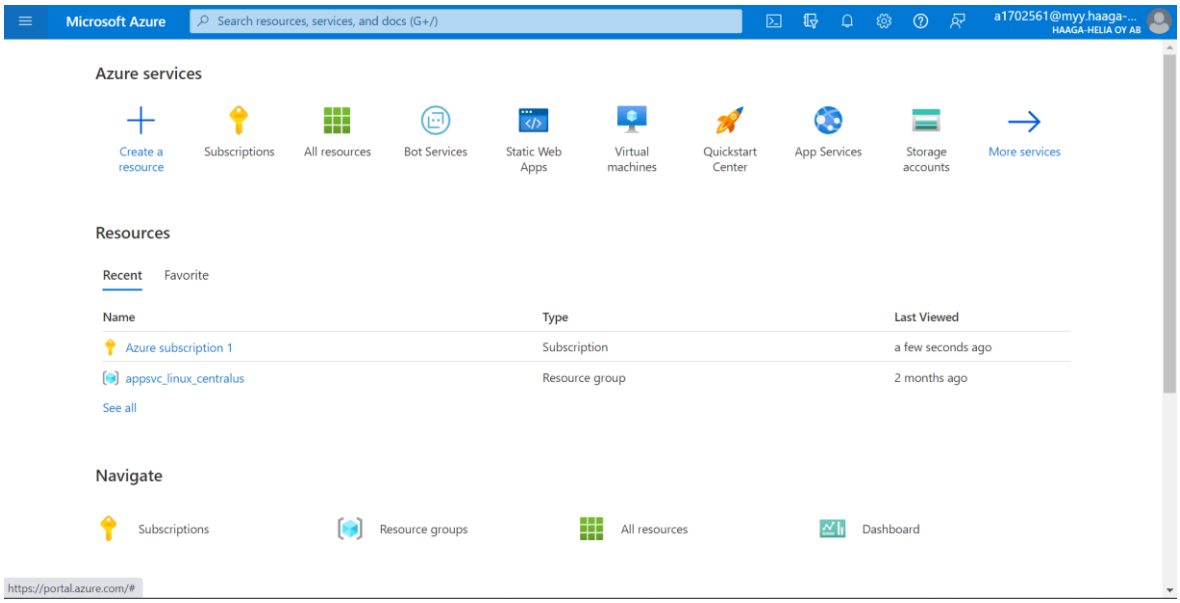


Figure 49 Azure Platform Home Page

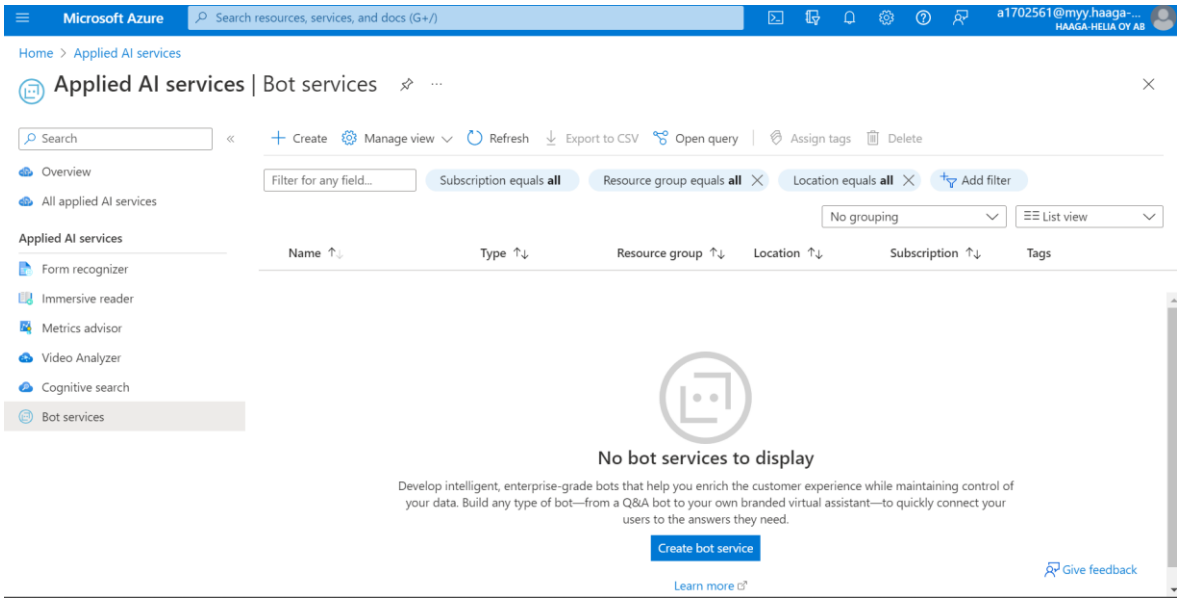


Figure 50 Azure Platform Bot Service

Appendix 2. Boomi Platform

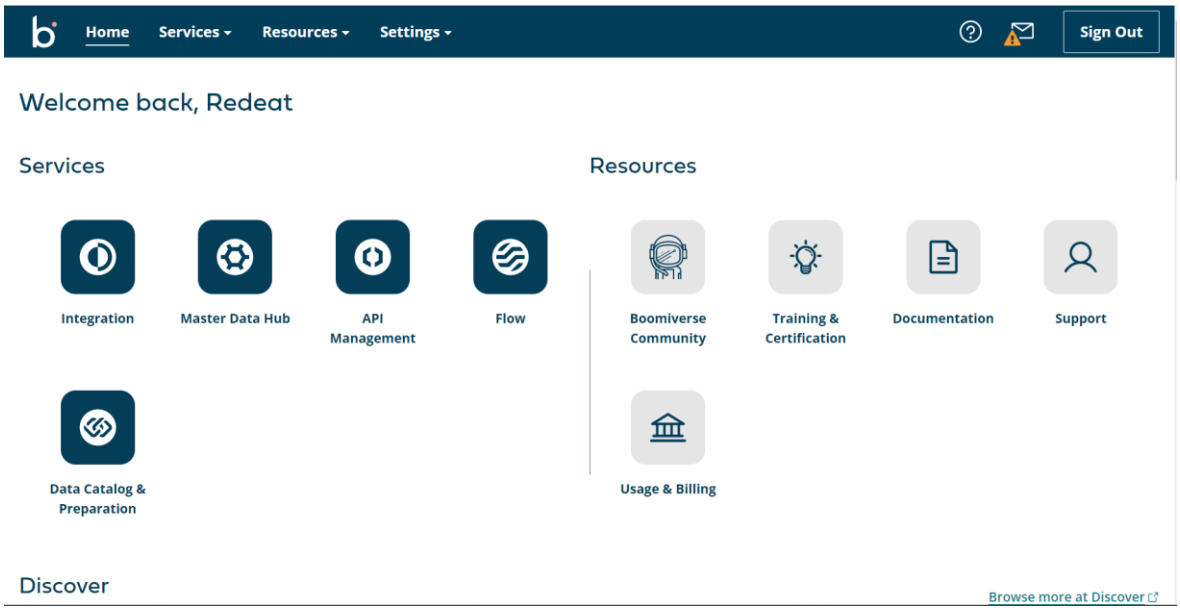


Figure 51 Boomi Platform Home page

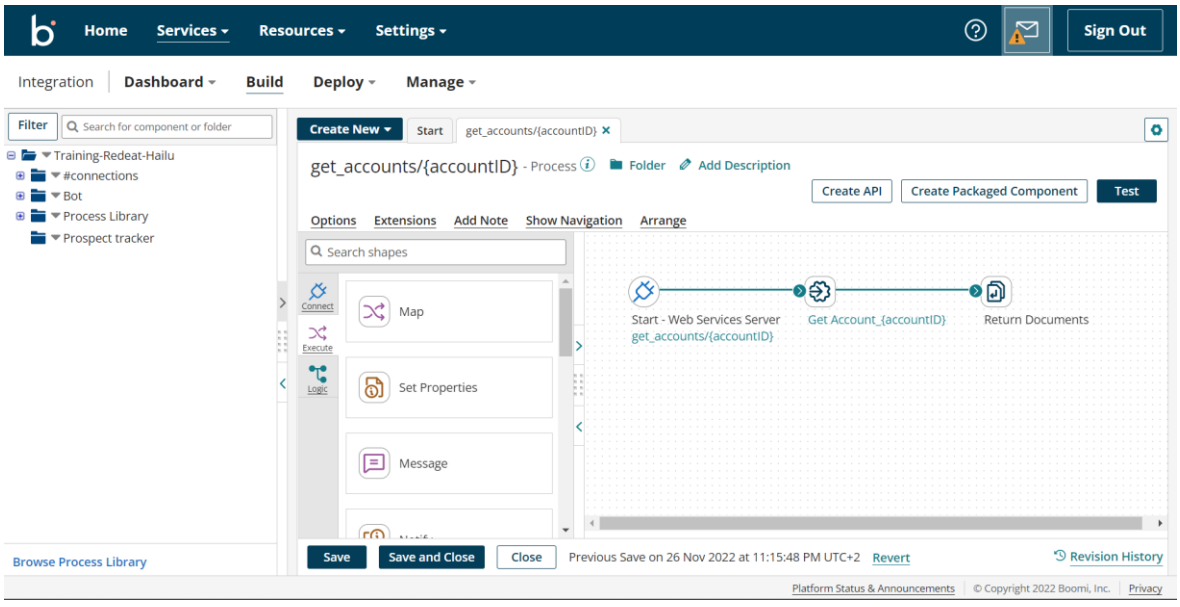


Figure 52 Boomi Platform Services