

Table of Contents

Table of Contents.....	1
Overview	3
Project Summary.....	3
File Structure Breakdown	3
Frontend.....	4
Backend	5
Documentation	6
Testing	7
Miscellaneous	7
Prerequisites	9
Presentational Version of the Chatbot	9
Setting up Watson AI Instances	10
Setup Watson Discovery Instance.....	10
Setup Watson Assistant Instance	15
Add Discovery and Assistant API Keys	19
Setup Node.js Server.....	22
How to Start and Stop the Node.js Server	23
Using the Chatbot.....	24
How to Ask a Query	25
How to Rate a Query.....	29
How to Access Links	30
How to Load More Answers.....	31
How to go to Dashboard (Admin)	32
Manage Data Sources	33
Adding Documents	33
Discovery UI.....	33
Uploading Documents.....	33
Retraining the Model.....	35
Node.js Server.....	37
Uploading a Single File Document	37
Uploading a Multiple File Document	37
Uploading an Updated Glossary	38

Deleting Documents.....	39
Node.js Server.....	39
Managing Document Ratings	40
Discovery UI.....	40
Troubleshooting and Maintenance	42
Troubleshooting	42
Tests	42
IBM Cloud Documentation	44
Contact Details.....	45
FAQs	45
Maintenance.....	46
Premium Watson Service Plans	47
NPM Packages	47
GitHub Repository Access Policy.....	47
Keep Local Glossary Up-to-date.....	48
User Feedback Effects on the AI Model	48
Future Changes	49

Overview

Project Summary

The project undertaken is a self-learning chatbot that answers questions about IBM Cloud for Financial Services (C4FS). It uses Watson Discovery, an IBM Cloud service that processes, enriches, and selects data based on an input query. IBM's documents are uploaded to Discovery, allowing the chatbot to send questions (queries) and receive the most relevant answers. The main process is simple, the chatbot accepts a question from the user, queries Watson Discovery, and displays the answer in a chat format. The user can also click a "*Load More*" button which displays more answers. The chatbot's self-learning aspect is seen in the ratings, where users can rate an answer positively or negatively (with a thumbs up/down), allowing Watson Discovery to learn what a 'good' answer is. In case of any internet issues, the chatbot will inform the user if it cannot connect to IBM Discovery, due to the user's connection or server issues on Discovery's side. The chatbot also features an auto-update function that constantly renews the data being used to train Discovery. This is done by periodically checking the IBM docs, if a change is detected then the docs are parsed and uploaded to discovery. This enables the chatbot to always be updated on IBM C4FS without maintenance needed on IBM's side.

File Structure Breakdown

The delivered project has several components, but it is mainly split into three areas: frontend, backend, and documentation. The frontend is responsible for the display of the chatbot and handling user interaction. The backend is responsible for communicating with Watson Discovery and keeping track of ratings. The documentation contains explanations of all functions in each file, to assist with the project's use, as well as its maintenance. There are also unit tests included to ensure the project has been set up correctly.

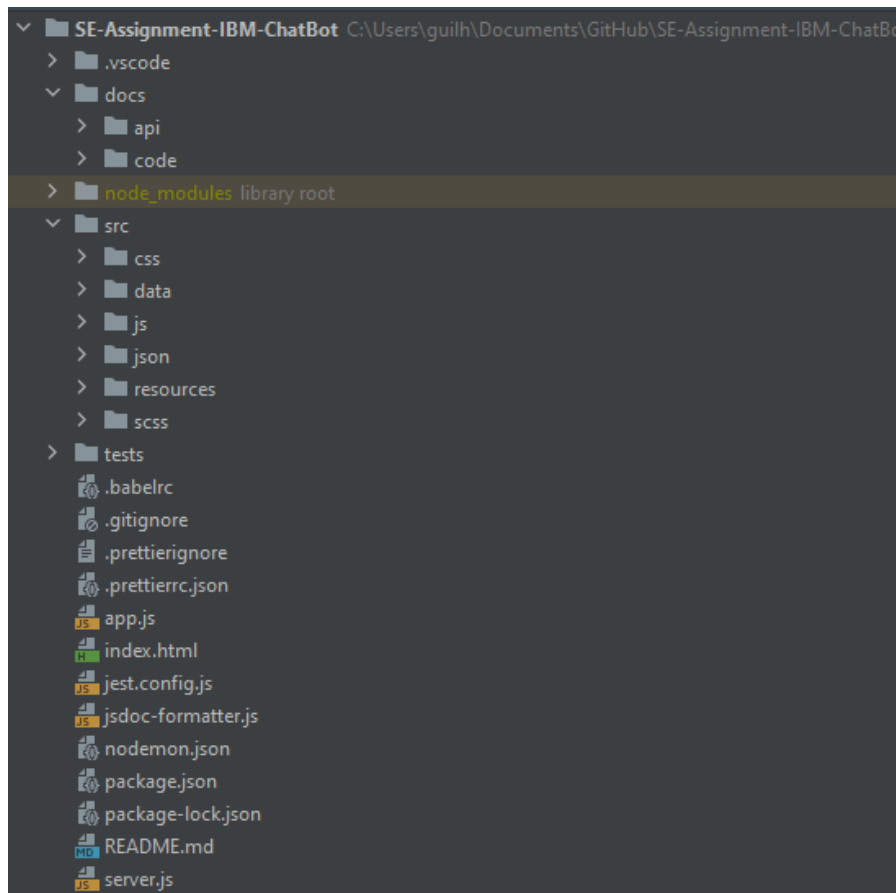


Figure 1: Project File Structure

Frontend

The frontend of the project is written in vanilla HTML, SCSS and JS. It is mainly spearheaded by the “**index.html**” file. This is the page that is rendered when the chatbot is loaded and has a script that loads the messages into the chat window. That script is “**/src/js/script.js**”, the JavaScript file in the project that does most of the work, connecting frontend to backend. “**script.js**” communicates with Watson Discovery through the “**app.js**” file (backend), enabling it to send the user’s queries and display the answers through use of DOM (Document Object Model). The front end also uses other files in the “**/src**” folder.

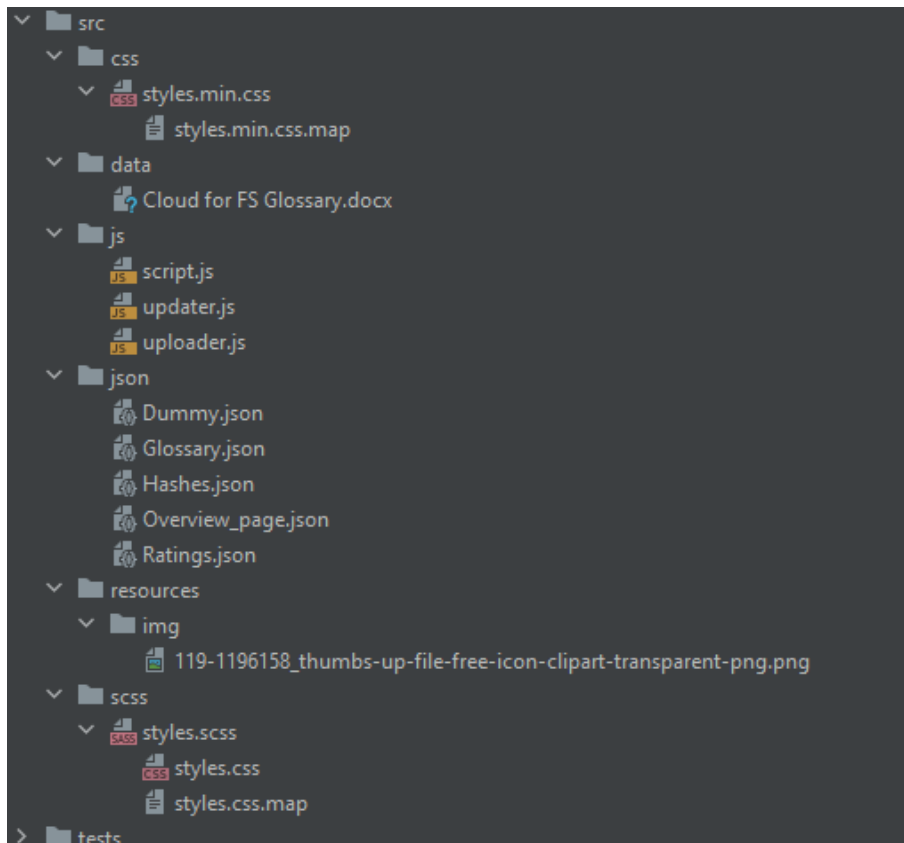


Figure 2: src Folder File Structure

The “/src/css” and “/src/scss” folders contain stylesheets that are used in the visual design of the chatbot. The .scss allows code to be written that is compatible with all browsers when formatting of the website. This file is then compiled into a .css using SASS, which is linked to the “**index.html**” file. The .css files should not be edited because they are recompiled from the .scss files on server launch, any changes should be done to the .scss files directly. The .map files contain version data and other minor information. The “/src/data” folder contains documents to be parsed and uploaded to Watson Discovery, included is IBM’s FAQ regarding their C4FS service. The “/src/resources” folder is meant to include any miscellaneous frontend resources that need to be kept as files, an icon of a thumb is used here for the rating icons.

Backend

The back end of the project mainly focuses on three things, querying Watson Discovery, handling ratings, and keeping the database up to date. The majority of it happens through the “**app.js**” file (running through “**server.js**”), which takes inputs from the frontend and deals with them accordingly. Queries received from “/src/js/script.js” are submitted to Watson Assistant and the answers returned.

When answer ratings are sent, appropriate scores are calculated and stored (in the “/src/json/Ratings.json” file), these scores are submitted to discovery, enabling self-learning. The “server.js” also calls the `AutoUpdater()` function in “/src/js/updater.js”. This function checks the IBM documentation repository every 24 hours, if it detects any changes (different GitHub hash to the one in “/src/json/Hashes.json”), it parses that information and segments it into several question-and-answer blocks which are uploaded to Watson Discovery using “/src/js/uploader.js”.

Documentation

The project files include documentation in the “/docs” folder. This folder contains the documentation for the most relevant files and the resources needed to render the documentation. The “/docs/api” folder contains the documentation for the API and the “/docs/code” folder contains the documentation and source code for the major files. Within each folder, the “/docs/.../fonts”, “/docs/.../scripts”, and “/docs/.../styles” folders contain fonts, JavaScript files, and style sheets used in the documentation. The .html files are the actual documentation, a mixture of source code and function descriptions.

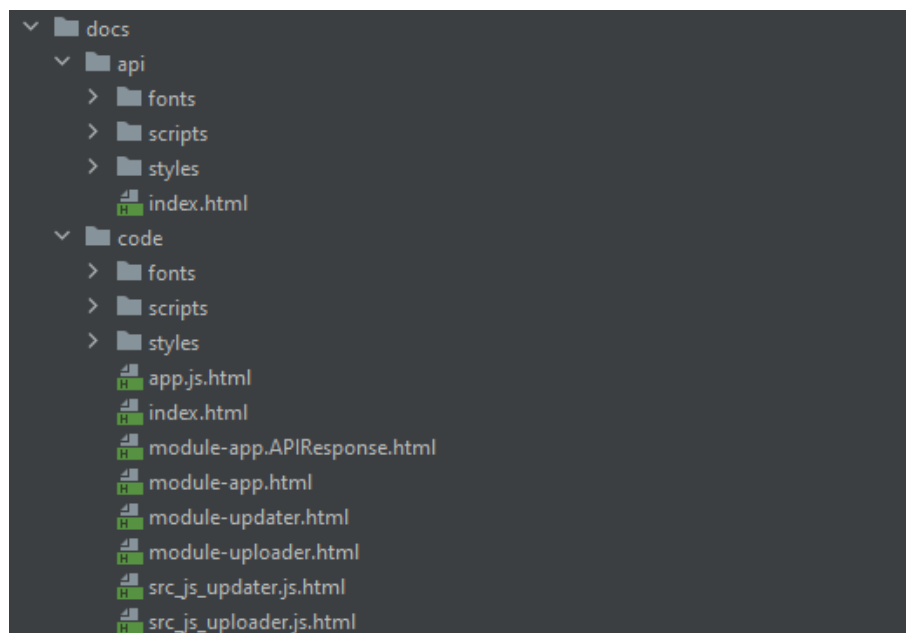


Figure 3: out Folder File Structure

Testing

The testing framework used is Jest along with Babel and SuperTest, due to its wide use capabilities and being easy to maintain/add to. The “/tests” folder includes “/tests/unit.test.js” a batch of unit tests that can be run using the command “npm test”. Tests that are created further down the line should be added to this folder and functionality could be implemented to run all tests in the folder at once to improve efficiency. The file “/src/json/Dummy.json” contains a sample question used in testing.



Figure 4: tests Folder File Structure

Miscellaneous

There are other files that don't exactly fit into any of these categories, they are described below. The code formatter Prettier was also used to ensure a consistent code style, all files with “prettier” in the name are used by it.

File	Description
/src/json/Glossary.json	Glossary file, auto generated from IBM's C4FS FAQ, includes pairs of terms and definitions. Since discovery cannot be trained on single terms, this JSON is kept locally.
/src/json/Overview_page.json	Auto-generated JSON file keeping track of how the Overview Page from GitHub has been split up and uploaded to discovery. Used to avoid duplicate file uploads
.babelrc	Config file for Babel
.gitignore	Standard 'git ignore' file
.prettierignore	Same as .gitignore but for the Prettier code formatter
.prettierrc.json	Config file for the Prettier code formatter
jest.config.js	Config file for Jest

nodemon.json	Config file for nodemon
package.json	Standard package file (includes dependencies and npm commands)
package-lock.json	Standard package-lock file (includes dependencies' version)
README.md	Standard README file, contains no relevant information

Prerequisites

This section will go through how to set up the chatbot to run on your own. It will assume you have pulled/downloaded the code from the IBM chatbot GitHub repository into your local system (<https://github.com/ArijusLengvenis/SE-Assignment-IBM-ChatBot>).

Note: When accessing the C4FS and FAQ documents, one must be logged in to GitHub (<https://github.com>) and ask for an invite to the repository from Arijus Lengvenis (arijus.lengvenis@durham.ac.uk), ensuring to provide your GitHub username in your request.

For deployment of the Chatbot as intended, including the use of the original Watson Assistant and Watson Discovery instances, users should continue and observe the guidance listed in this section. If a user wants to set up the pre-made presentational version of the chatbot, they may ignore the sections titled “*Setup Watson Discovery Instance*” and “*Setup Watson Assistant Instance*”, and after completing the following section below can carry on and advance to “*Setup Node.js Server*” without creating a new IBM Cloud account.

Presentational Version of the Chatbot

For access to the Watson Discovery and Watson Assistant instances created by Group 18 of Durham and used to form the presentational chatbot AI, contact James Jushchuk (jushchuk@ibm.com) and ask for permission to access the account. Once permission is granted, users should enter the IBM Cloud main page and access the “*Services and Software*” section from the home page.

Resource summary

[View all](#)


3

Resources

[Services and software](#)

✓ 3

Once opened, the user can view the condition of each Watson technology instance. Clicking on them will allow users the option to launch the Watson Assistant / Watson Discovery tools in order to access the chatbot AI.

^ Services and software (3)						
 Watson Assistant-9i	arijus-lengvenis	London	Watson Assistant	 Active	—	
 Watson Discovery-nc	arijus-lengvenis	London	Watson Discovery	 Active	—	

Setting up Watson AI Instances

This set of instructions will guide users who wish to create their own instances of Watson Discovery and Assistant on their personal IBM Cloud accounts. Users who are satisfied with using the presentational chatbot may advance to “*Setup Node.js Server*” without doing the steps in this section.

Note: For a group of users only one set of instances needs to be created. Other users can be given access using the access control options in the IBM Cloud account profile. Detail on how to do this can be found in the IBM documentation (<https://cloud.ibm.com/docs/assistant?topic=assistant-access-control>).

Setup Watson Discovery Instance

To create a Watson Discovery instance, users must follow the instructions up to Step 1 on the “*Getting Started with Watson Discovery*” page from the IBM Cloud Watson Discovery documentation (<https://cloud.ibm.com/docs/discovery-data?topic=discovery-data-getting-started>). Ensure the plan selected for the instance is at least a Plus plan. The cost of the Plus plan is £500/month, but it is only necessary to upload the AI model which will be detailed shortly. Afterwards the plan can be switched to the Lite version, which is free, but has a very limited number of documents that it can store and is limited on the amount of API calls that can be done monthly.

Similar to creating a Watson Discovery instance, to create an instance of Watson Assistant, users must follow the instructions up to Step 1 on the “*Getting started with Watson Assistant*” page from the IBM Cloud Watson Assistant documentation (<https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started>). Ensure the plan selected for the instance is at least a Plus plan. The cost of the Plus plan is £500/month and this is a mandatory cost that cannot be avoided.

Once both instances are created, users should do the following to set up their Watson Discovery instance:

STEP 1: Launch the Watson Discovery tool and create a new project.

The screenshot shows the 'Project name' field with the text 'My Chatbot'. Below it, the 'Project type' section displays three options: 'Document Retrieval' (selected with a blue border and a checkmark), 'Conversational Search', and 'Content Mining' (marked 'Enterprise'). Each option has a corresponding icon and a brief description. At the bottom right, there is a blue 'Next' button with a right arrow.

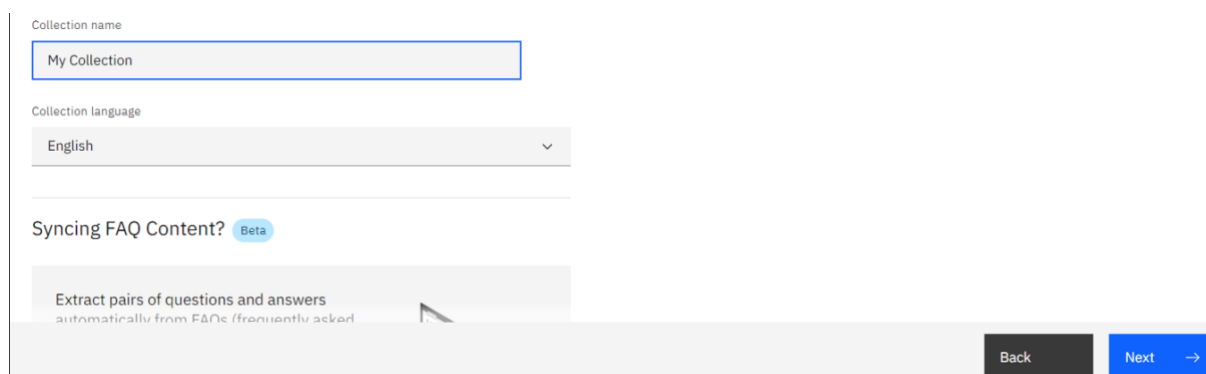
Users should name their project something convenient that they can refer to later on. They should select the project type “*Document Retrieval*” and click “*Next*”. Then, users should be able to select the location for their data.

STEP 2: Select “*Upload data*” and then click “*Next*”.

The screenshot shows the 'OK, where is your data?' screen with several data source options: Salesforce, SharePoint Online, SharePoint On Prem, Box, Web crawl, IBM Cloud Object Storage, and Upload data (selected with a blue border). Below these options, there is a link 'Reuse data from an existing collection' and a question 'Don't see the data source you need?'. At the bottom right, there is a blue 'Next' button with a right arrow.

After completing that step, users should be given the option to name their data collection.

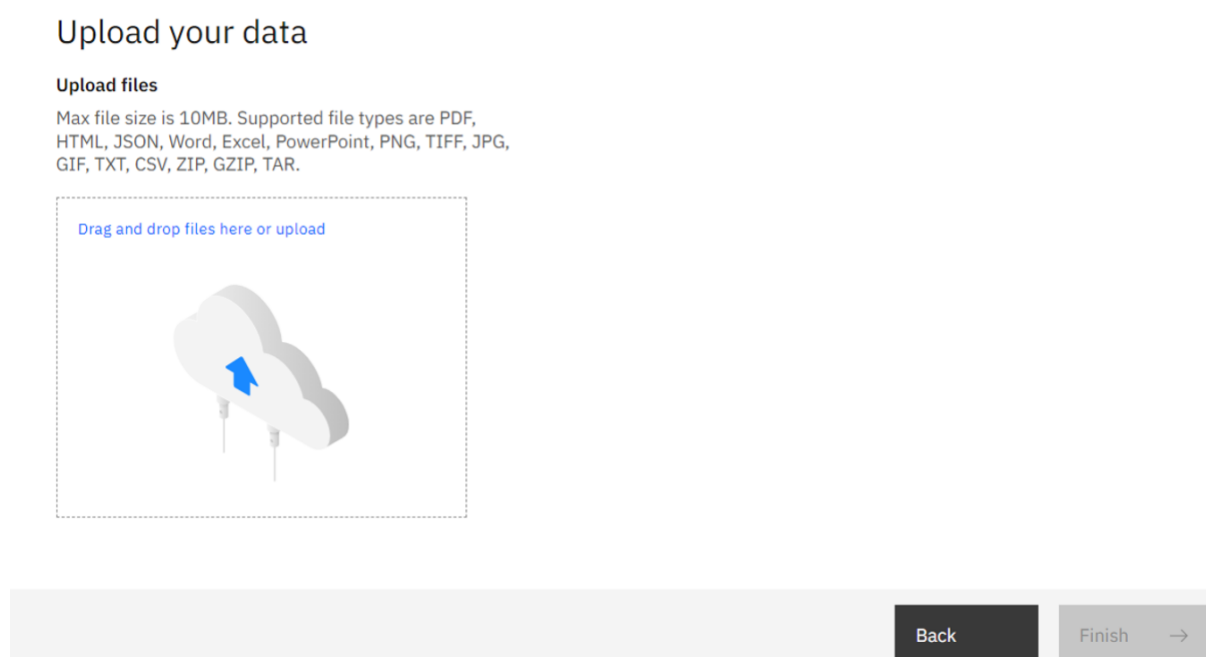
STEP 3: Users should name their collection and click “*Next*”.



The screenshot shows a form for creating a new collection. At the top, there is a text input field labeled 'Collection name' with the placeholder text 'My Collection'. Below this is a dropdown menu labeled 'Collection language' with 'English' selected. Further down, there is a section titled 'Syncing FAQ Content?' with a 'Beta' badge. Below this is a grey box containing the text 'Extract pairs of questions and answers automatically from FAQs (frequently asked questions)'. At the bottom right of the form are two buttons: 'Back' and 'Next' with a right-pointing arrow.

Users will be taken to a screen where they can upload their data files.

STEP 4: Upload the C4FS (<https://ibm.box.com/s/dalvq9j9x2wavegnaa6zoezr4mc0blwt>) and FAQ (<https://ibm.box.com/s/2xlqe4bh4s8l94g4joumqslslzw1v9odp>) documents, and click “*Finish*” to begin processing your documents.



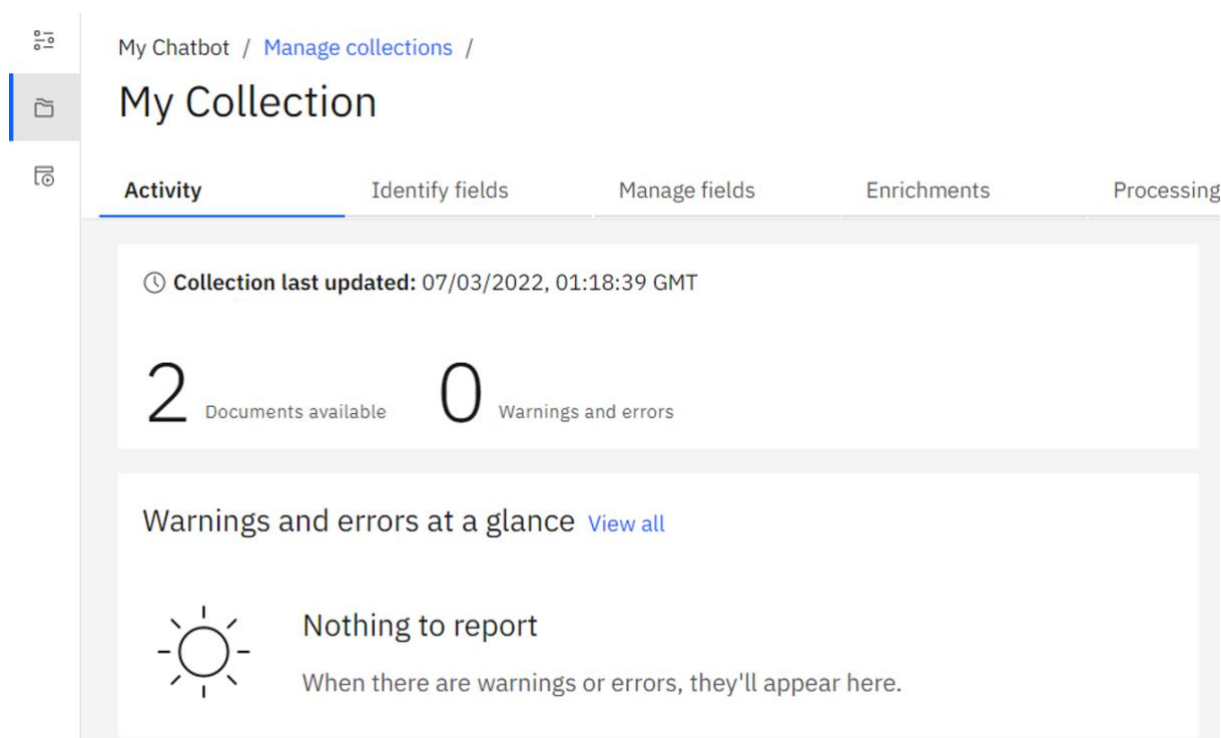
The screenshot shows the 'Upload your data' screen. At the top, the heading 'Upload your data' is followed by the sub-heading 'Upload files'. Below this, a message states: 'Max file size is 10MB. Supported file types are PDF, HTML, JSON, Word, Excel, PowerPoint, PNG, TIFF, JPG, GIF, TXT, CSV, ZIP, GZIP, TAR.' In the center is a large dashed box containing the text 'Drag and drop files here or upload' and an illustration of a cloud with a blue arrow pointing upwards into it. At the bottom right of the screen are two buttons: 'Back' and 'Finish' with a right-pointing arrow.

Users may notice that these documents do not include any glossary terms - those keywords are included in the Cloud for FS Glossary document, which stores a set of term and definition pairs and sends them to Watson Discovery when the server initialises. For more details on how to upload and update the

glossary, please refer to the section discussing “*Uploading an Updated Glossary*”. For details on maintaining this glossary of terms, please refer to “*Keep Local Glossary Up-to-date*”.

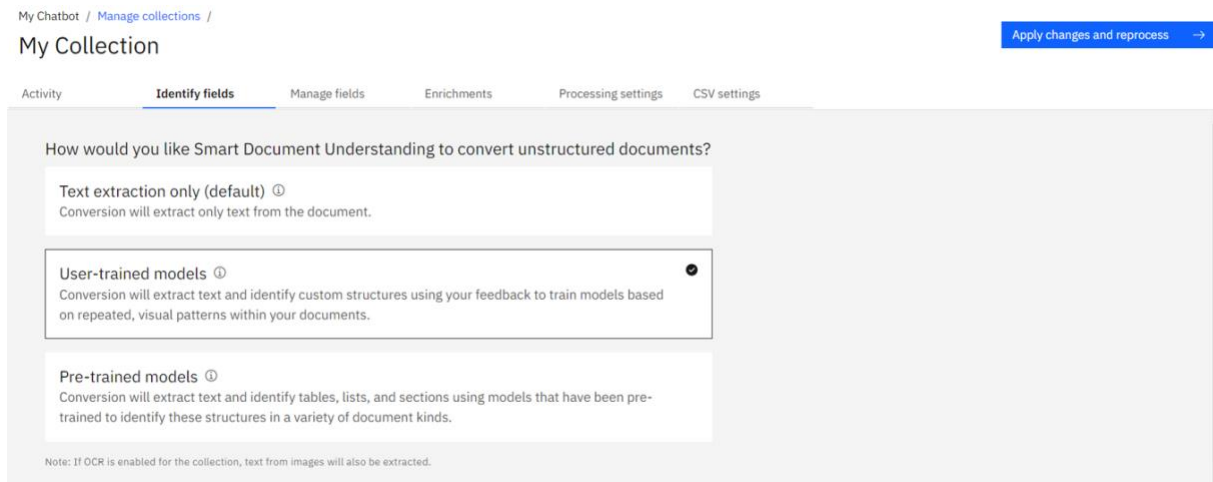
Note: When accessing the C4FS and FAQ documents, one must be logged in to IBM Box (<https://ibm.ent.box.com>) and ask for an invite to the box repository from Mark O’ Kane (mark.okane@ie.ibm.com).

STEP 5: Once the processing has been completed, navigate to “*Manage Collections*”, and select the new collection.



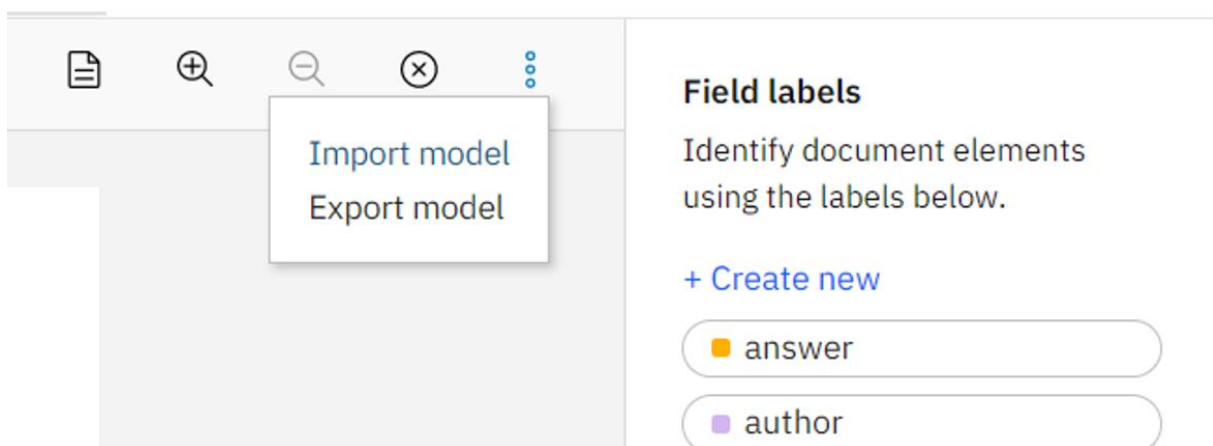
The screenshot shows the 'My Collection' management interface. The breadcrumb navigation at the top reads 'My Chatbot / Manage collections /'. The main heading is 'My Collection'. Below this is a horizontal tab bar with five tabs: 'Activity' (selected), 'Identify fields', 'Manage fields', 'Enrichments', and 'Processing'. The 'Activity' tab displays a clock icon and the text 'Collection last updated: 07/03/2022, 01:18:39 GMT'. Below this, there are two large numbers: '2' for 'Documents available' and '0' for 'Warnings and errors'. A section titled 'Warnings and errors at a glance' includes a 'View all' link. At the bottom, there is a sun icon and the text 'Nothing to report' followed by 'When there are warnings or errors, they'll appear here.'

STEP 6: Navigate to the “*Identify fields*” subsection and select “*User-trained models*”, then “*Apply changes and reprocess*”.



STEP 7: Now on the page to edit the Watson Discovery training, either implement the default chatbot model (<https://ibm.box.com/s/nr4g2qqr61m7pzx5zfjcdcdir6rk71rjr>) by using the “*Import model*” option. For more information on how to effectively retrain the chatbot, please refer to the “*Retraining the model*” section.

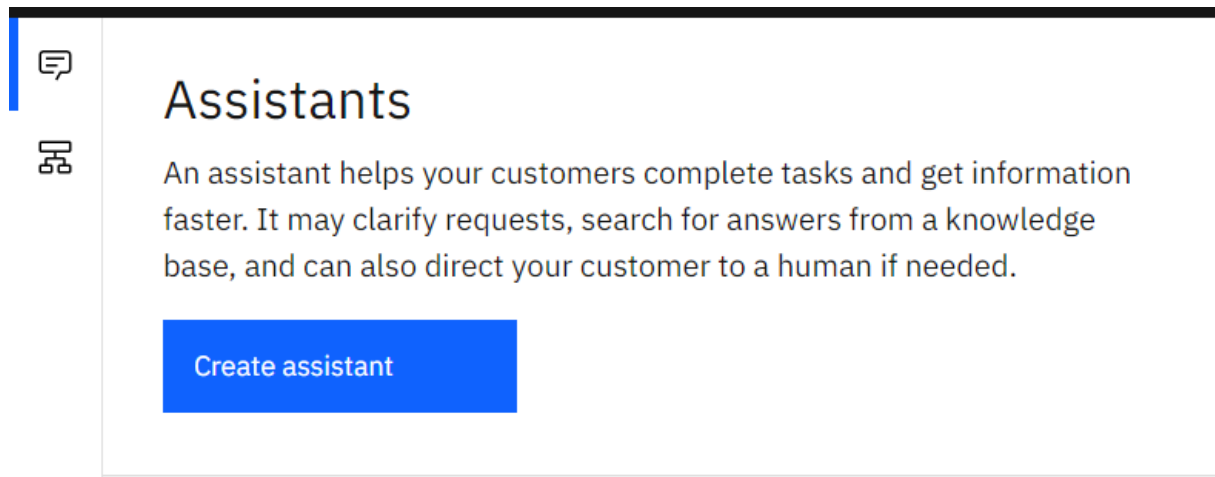
Note: When accessing the default chatbot model, one must be logged in to IBM Box (<https://ibm.ent.box.com>) and ask for an invite to the box repository from Mark O’Kane (mark.okane@ie.ibm.com).



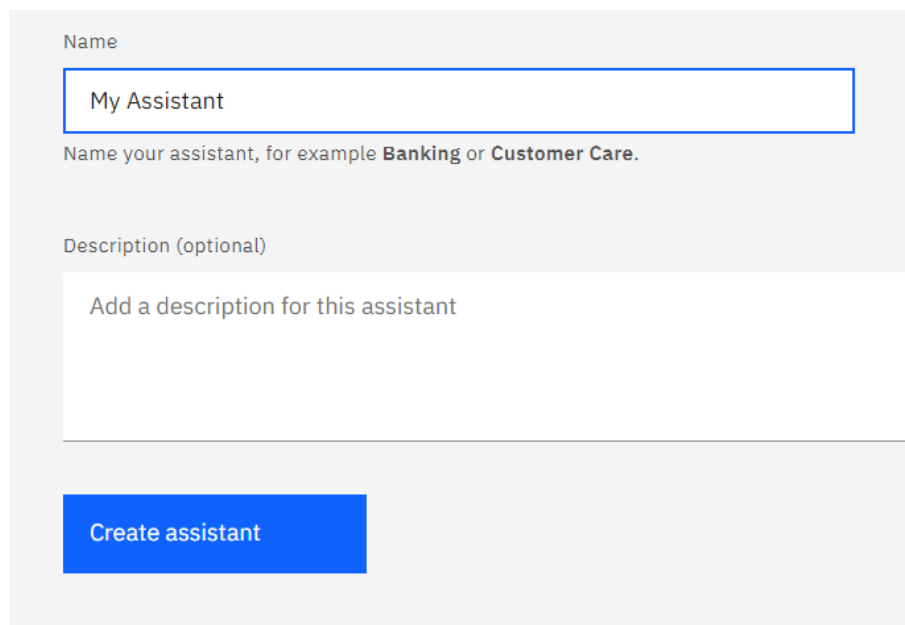
Setup Watson Assistant Instance

After following these steps, users end up with a working Watson Discovery instance. The next steps will refer to correctly setting up the Watson Assistant instance.

STEP 1: Launch the Watson Assistant tool from the IBM Cloud dashboard and create a new assistant.



STEP 2: Name the assistant as per convenience and click the “*Create assistant*” button.

A screenshot of the Watson Assistant creation form. It has a light gray background. At the top is a label 'Name' above a text input field containing 'My Assistant'. Below the input field is a hint text: 'Name your assistant, for example **Banking** or **Customer Care**.' Below this is a label 'Description (optional)' above a larger text input field containing the placeholder text 'Add a description for this assistant'. At the bottom of the form is a blue button labeled 'Create assistant'.

On creation, the user will be taken to a display giving them options to customise their assistant.

STEP 3: Advance by adding a search skill from the options available.

Search

Turn any content into answers

- Create Q&A experiences in minutes
- Sync with websites and data sources for always up-to-date answers
- Handle even complex questions with inclusive, contextual responses

[Watch a brief demonstration](#)

Add search skill

STEP 4: Create a new skill and name it as per preference, then continue.

Add existing skill

Create skill

Name

My skill

Name your skill; for example, Account application or Personal banking.

Description (optional)

Add a description for this skill

Continue

STEP 5: Select the Watson Discovery instance and the relevant project in the search skill, then click “Next” to carry on.

← Back

Search Skill/My skill

Cancel

Next

Choose a Discovery instance to connect to ⓘ

Watson Discovery-nc

To create a new Discovery instance, visit the [discovery catalog](#)

Choose which project you want to use ⓘ

Create a new project +

Project name	Collection name	
My Chatbot	My Collection	

After this, the user must select the relevant labels from a dropdown menu such that “*Title*” is the set of questions, and “*Body*” is the set of answers (normally, the corresponding default labels from the exported model are “*questions*” and “*answers*”).

STEP 6: Finalise the question and answer tags, then click the “*Create*” button to form the Search Skill.

← Back
Search Skill/My skill

Learn how to troubleshoot and improve the answers generated by your Search integration. [Learn more](#)

Discovery instance: Watson Discovery-nc Project: My Chatbot [↗](#)

Configure result content

Map your data schema from Discovery to the title, body, and URL fields below then check the preview to review your configuration.

Title

question | Example: compliance with IBM Cloud for Financial Services™

Body

answer | Example: Accelerate innovation through security and

Once this search skill is created, users can create a dialog skill to form more natural dialog options within the chatbot. Examples of results caused from such dialog skills can be seen in the “*Using the Chatbot*” section, referring to the third kind of response from the chatbot.

To begin making a dialog skill, do the following:

STEP 1: Click on the “*Add an actions or dialog skill*” button

Actions or Dialog

Build conversations

Understand and address questions or requests that your customers typically ask.

- **Actions** lets you have an assistant ready to chat in less time, with less effort. Compose step-by-step flows for any range of simple or complex conversations.
- **Dialog** offers a set of full-feature editors that you use to define both your training data and the conversation, with greater control over the logic flow.

[Learn more](#)

[Add an actions or dialog skill](#)

Search

STEP 2: Click on “*Create skill*” header, fill in fields as preferred, and ensure the “*Dialog*” option is selected

Add Actions or Dialog skill

Add an existing skill or use the sample skill.

[Add existing skill](#) **Create skill** [Use sample skill](#) [Upload skill](#)

Name

My Dialog Skill

Name your skill; for example, Account application or Personal banking.

Description (optional)

Add a description for this skill

Language ⓘ

English (US) ▼

Skill type

☐ Action

☒ Dialog

Create skill

STEP 3: Clicking the blue “*Create skill*” button will create the dialog skill, which users can view and edit at convenience

Dialog

My Dialog Skill

LANGUAGE

English (US)

TRAINED DATA

0 Intents | 0 Entities | 2 Dialog nodes

VERSION

draft

DESCRIPTION:

VERSION CREATED:

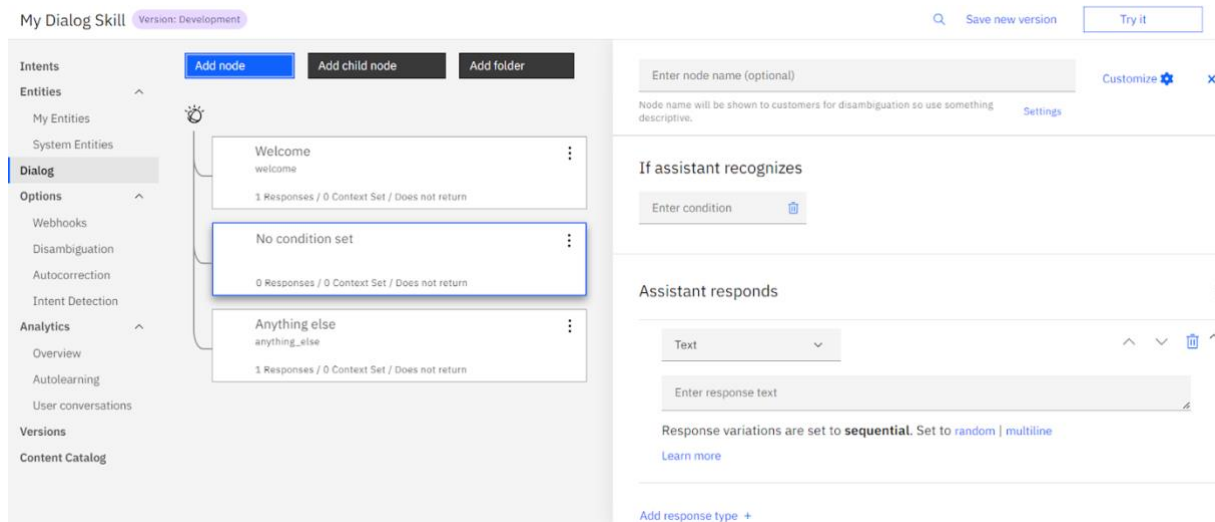
10 Mar 2022 15:59

LINKED ASSISTANTS (1):

My Assistant

STEP 4: Clicking the formed dialog skill will allow users to view the dashboard used to personalise the dialog skill as they see fit. For further detail on how to construct the ideal dialog skill, please refer

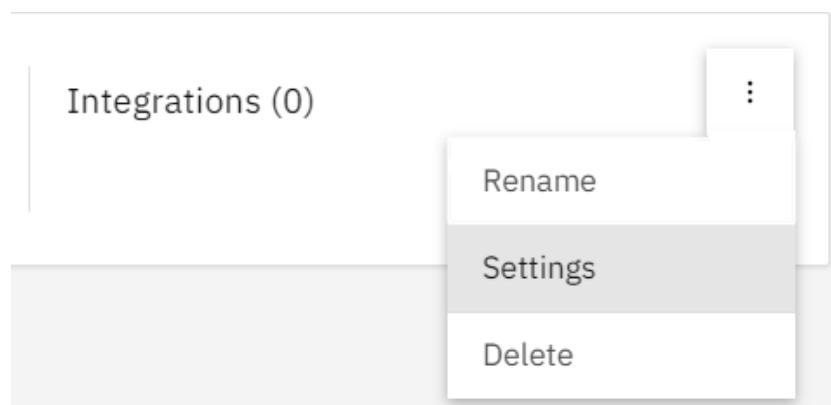
to the “*Creating a dialog*” section of the IBM Watson Assistant documentation (<https://cloud.ibm.com/docs/assistant?topic=assistant-dialog-overview>).



Now, the Watson Assistant has also finished its basic setup and is ready to be modified along with the Watson Discovery project.

Add Discovery and Assistant API Keys

In order to connect user made instances to the server, certain ID variables must be edited to meet the requirements for a connection. Two files must be edited in total: “**app.js**” and “**uploader.js**,” using any code/text editor on the user’s machine. The variables required for change to connect to Watson Assistant include the **apikey** value, the **serviceUrl** value, and the **AssistantID** value. To find what the values for these variables should be, the user must navigate to the settings of their Watson Assistant through IBM Cloud.



After completing that, users will be facing all the API details they require to note down.



The screenshot shows the 'API details' section of the IBM Cloud dashboard. It is divided into two sub-sections: 'Assistant details' and 'Service credentials'. In the 'Assistant details' section, the 'Assistant name' field is empty. The 'Assistant ID' field contains the value '7b730136-c29d-4c4c-8ce7-e64bd8c20fb5'. The 'Assistant URL' field contains the value 'https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/79dfff8b-2db7-4e46-'. In the 'Service credentials' section, the 'Credentials name' field contains the value 'Auto-generated service credentials'. The 'API key' field contains the value 'z359XU0gNHHMa1mm6A233NRUpu0Sr8DR26D04omZ3HH1'. Each field has a copy icon to its right.

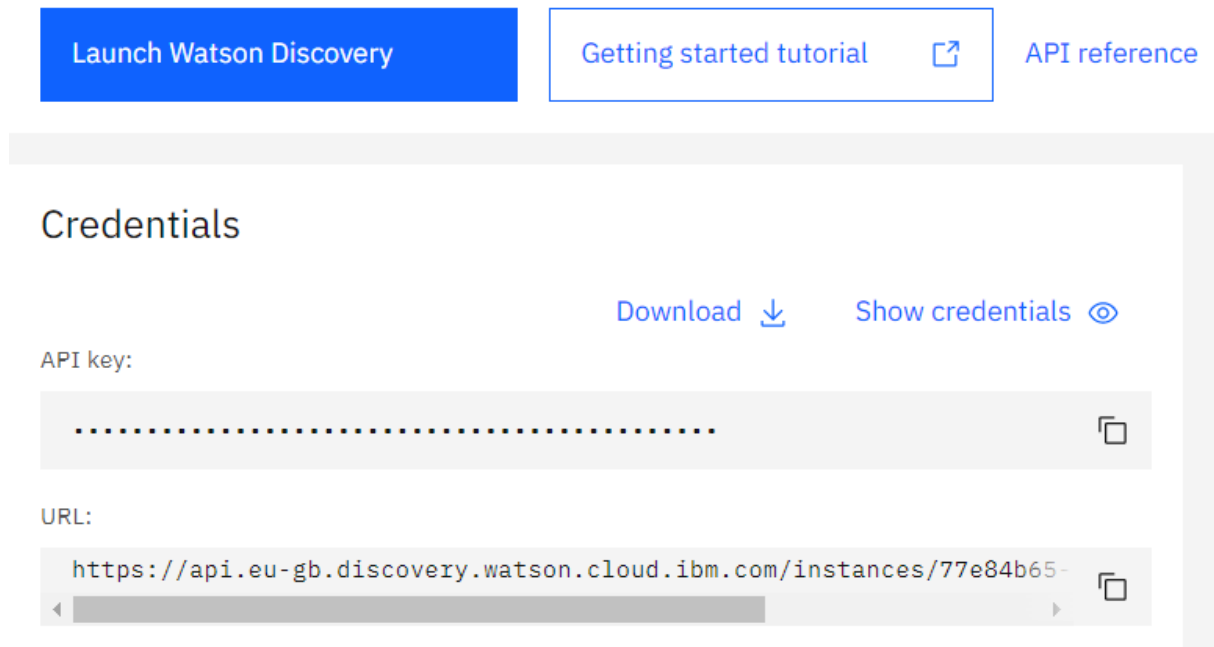
API details	
Assistant details	
Assistant name:	
Assistant ID:	7b730136-c29d-4c4c-8ce7-e64bd8c20fb5
Assistant URL:	https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/79dfff8b-2db7-4e46-
Service credentials	
Credentials name:	Auto-generated service credentials
API key:	z359XU0gNHHMa1mm6A233NRUpu0Sr8DR26D04omZ3HH1

The user should open “**app.js**” from the files on GitHub and replace the **apikey** variable in the text on line 34 with the new Watson Assistant API key. Then do the same for the **serviceUrl** variable on line 37, replacing them with the Assistant URL found in the same place from the IBM Cloud dashboard. As for the **AssistantID** value, the user should replace the ID on line 30 with their new Watson Assistant ID.

To connect to Watson Discovery, the variables required include the **apikey** value, the **serviceUrl** value, the **DiscoveryProjectID** value, and the **DiscoveryCollectionID** value. To find these, the user must check three separate places on IBM Cloud.

Firstly, before launching their instance, the user should note down the **apikey** value and the **serviceUrl** value below the “*Launch Watson Discovery*” button.

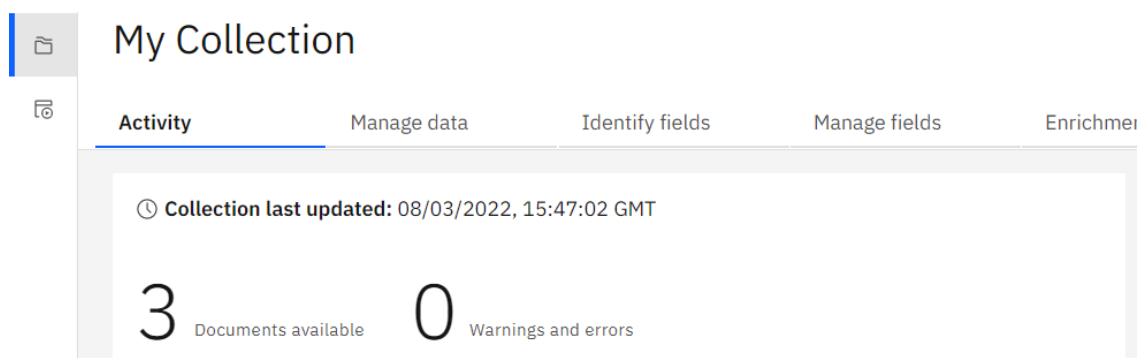
Start by launching the tool



Secondly, the user should launch their Watson Discovery tool and view the “*Integrate and deploy*” section. From there, the `DiscoveryProjectID` value is available.



Finally, the user should navigate to the “*Manage collections*” section for their data collection.



From here, observe the URL of the webpage, and scroll to the end of the line, which should say “*activity*”. Copy the sequence of numbers and letters between the keywords “*collections*” and “*activity*” to have successfully copied the `DiscoveryCollectionID` value.



`/collections/83726344-5e20-0c2f-0000-017f5ab396dd/activity`

The user should open “**app.js**” from the files on GitHub and replace the `apikey` variable in the text on line 46 with the new Watson Discovery API key, then do the same for the `serviceUrl` variable on line 49, replacing them with the URL found in the same place from the IBM Cloud dashboard. As for the `DiscoveryProjectID` and `DiscoveryCollectionID` values, users should replace these ID on lines 41 and 42 respectively with their new Watson Discovery Project and Collection IDs. The user should also open Users should open “**src/js/uploader.js**” from the files on GitHub and change the same `DiscoveryProjectID` and `DiscoveryCollectionID` values on lines 9 and 10 respectively with the same new Watson Discovery Project and Collection IDs. The `apikey` and `serviceUrl` variables on lines 14 and 17 should also be changed with their respective new values in this file.

Once all api keys and IDs have been replaced, users can move on to testing and starting the actual server.

Setup Node.js Server

Users must open a Terminal (macOS) or Command Prompt/PowerShell (Windows) and type the following command in to navigate to the relevant directory:

```
cd <file path to files from chatbot GitHub repository>
```

After that, users must run the following to begin downloading the necessary packages:

```
npm install
```

After that command has finished running, users must run the following to resolve any potential package warnings:

```
npm audit fix
```

The above command should ensure there are no faults with the package installation. Afterward, users should run the following command to test the system functions:

```
npm test
```

The terminal will run a series of Jest tests, each coming with a symbol of either a red cross, or a green tick. If all tests appear with a green tick, and there are no red crosses present in the Terminal output, then the chatbot set up has succeeded. Lastly, if a user edits the code, they can format it via Prettier by using the following command:

```
npm run format
```

Users can now progress with starting the server.

How to Start and Stop the Node.js Server

On the Terminal (macOS) or Command Prompt/PowerShell (Windows), remaining in the same directory, run the following to start the server:

```
npm start
```

The server will then be running on <http://localhost:8081>.

To close this server, the Terminal (macOS) or Command Prompt/PowerShell (Windows) must be closed using the command **Ctrl+C**, so users must ensure to close the window hosting the Terminal (macOS) or Command Prompt/PowerShell (Windows) when they wish to close the server.

Using the Chatbot

The chatbot is available for use on both mobile devices and desktop devices. Both are functionally identical, and the following segment of the manual will detail how to use this chatbot service and its functionality while referring to screenshots taken on the Desktop chatbot.

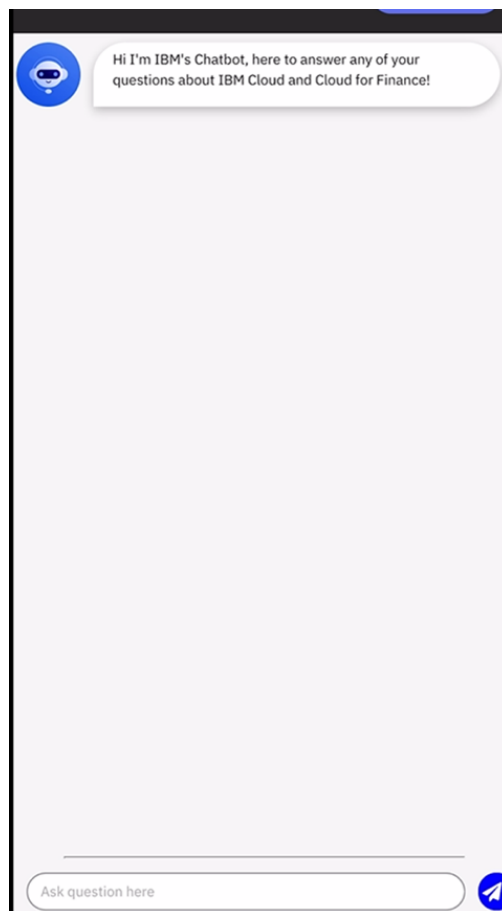
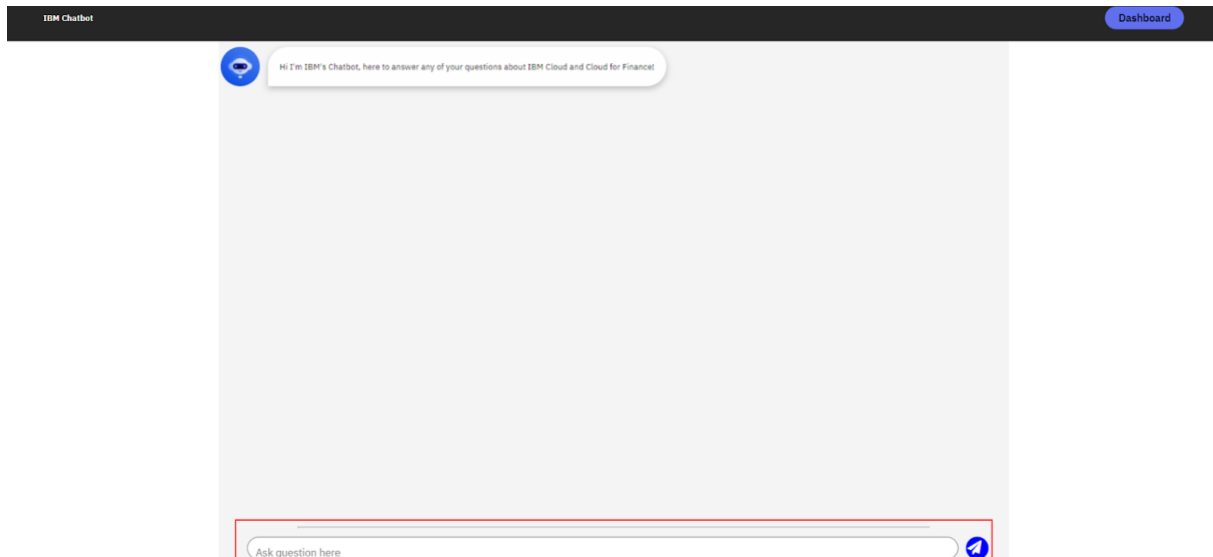


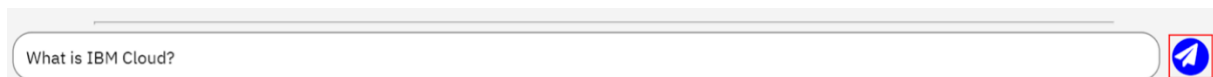
Figure 5: Chatbot on a Mobile Device

How to Ask a Query

The chatbot webpage has an input bar available for users to input their queries:

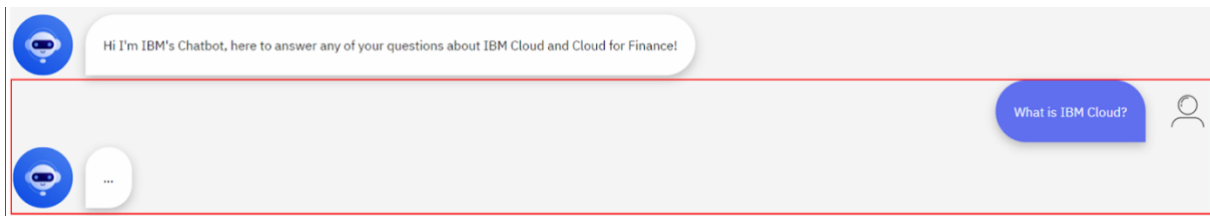


After typing a query out so it displays in the input field, users can submit it by pressing enter or clicking the circular blue-arrow button on the right of the input bar.



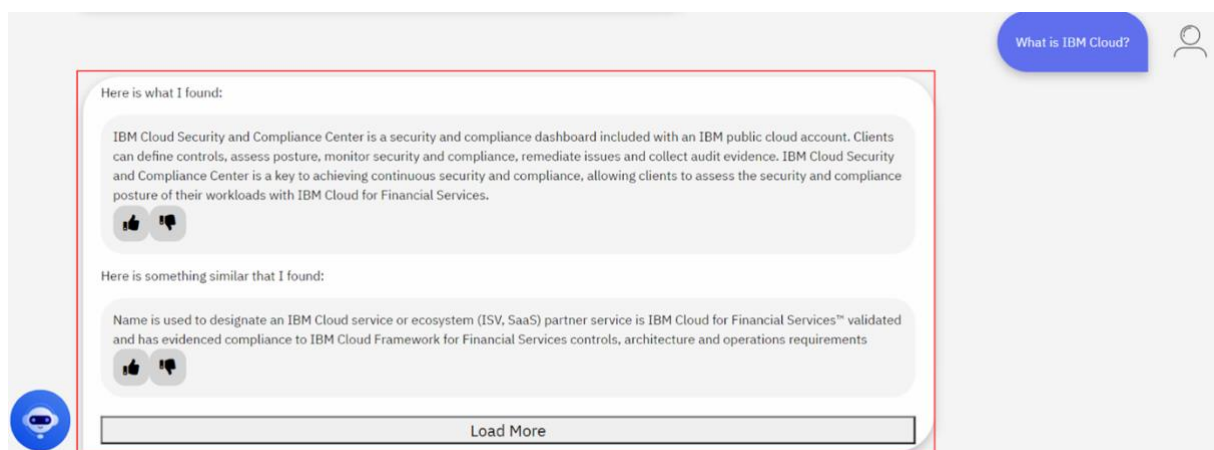
After a successful submission, the query will be passed to the backend server, and it will load the input into Watson Discovery and parse through the possible relevant excerpts from its database. Afterwards it will return a list of paragraphs that answer the query the best. The server then orders the answers from most confident to least deemed by Watson Assistant and keeps the top 5 (if they exist) answers as the others are deemed to be not relevant enough. Finally, the sorted answer array of objects with its metadata (query, document ID and a confidence metric) is passed back to the frontend, where the answer text will be displayed in the designated chat box above the input bar.

While this is processing, the user inputted query will appear on the right side of the screen and a “thinking” speech bubble will emerge from the chatbot, representing how it is searching for an answer to your query. The user input will disappear from the input field; however, users can scroll back along the chat to refer to former questions and answers anytime during the session.



There are three kinds of results depending on the kind of input:

1. A successful response with at least one answer, an option to rate the answers, and possibly a “Load More” option if more than two answers are returned and processed by Watson technologies – this comes from relevant-to-purpose inputs that Discovery has been trained to handle and a question intent analysis process that was conducted by Watson Assistant, such that it can provide a clear set of answers in return:

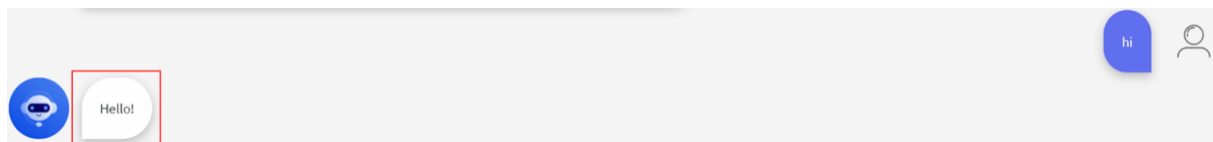


More detail on what can occur after a successful response can be found in the “*How to Rate a query*”, “*How to Access Links*”, and “*How to Load More Answers*” sections.

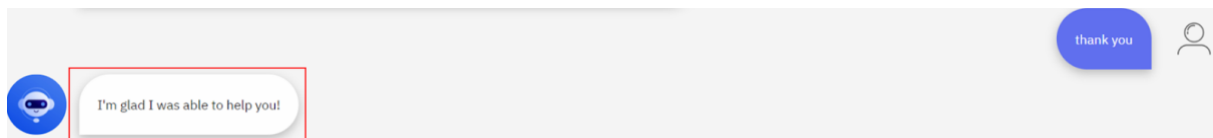
2. An unsuccessful response stating how Watson Discovery was unable to find an answer – this comes from inputs that Discovery has not been trained to handle, usually referencing topics outside of its database:



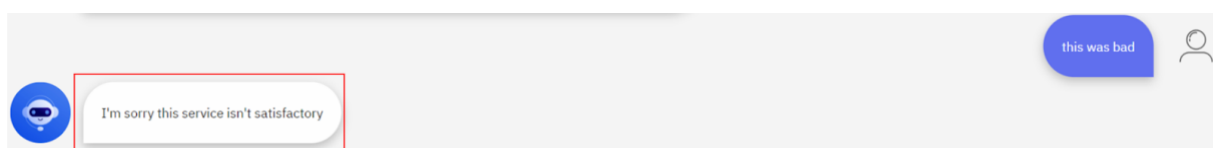
3. A unique response given by Watson Assistant depends on certain natural inputs:
 - a. One for common greetings:



- b. One for positive reactions:

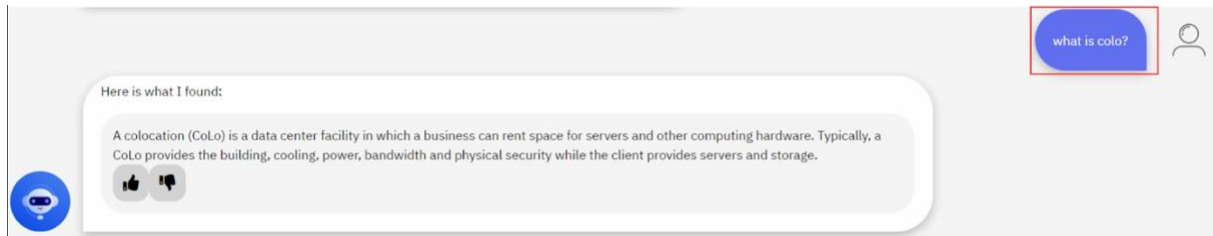


- c. One for negative reactions:

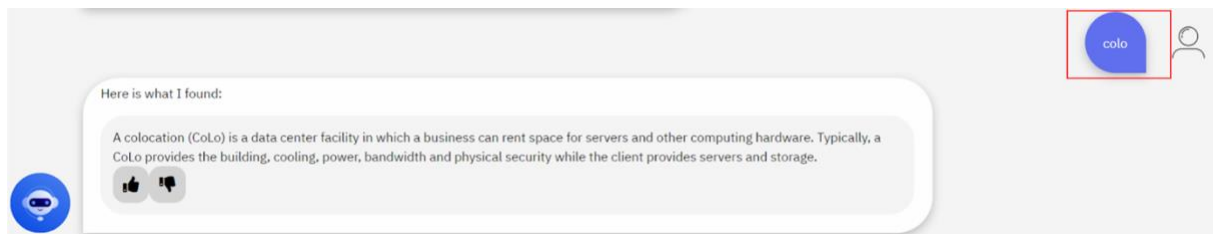


More general-purpose queries like the ones listed above are also available.

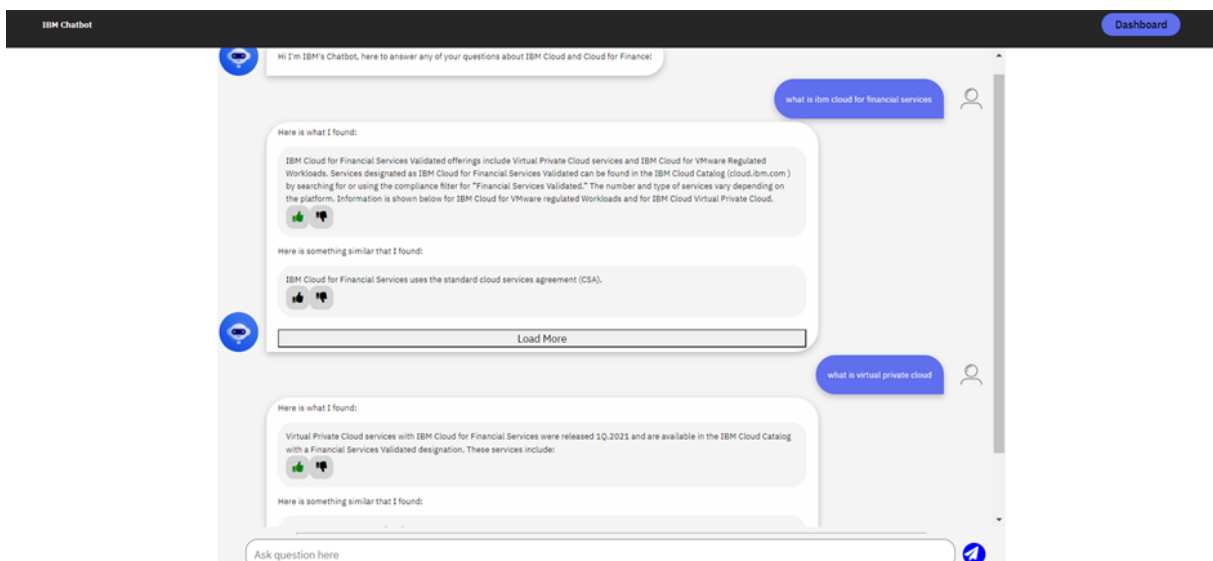
Another way to achieve successful responses is using keywords relating to a stored glossary of terms in the backend. For example, a user could format a simple question asking for the definition of a keyword.



Alternatively, users can also input the keyword or term itself to achieve the same definition as a response.

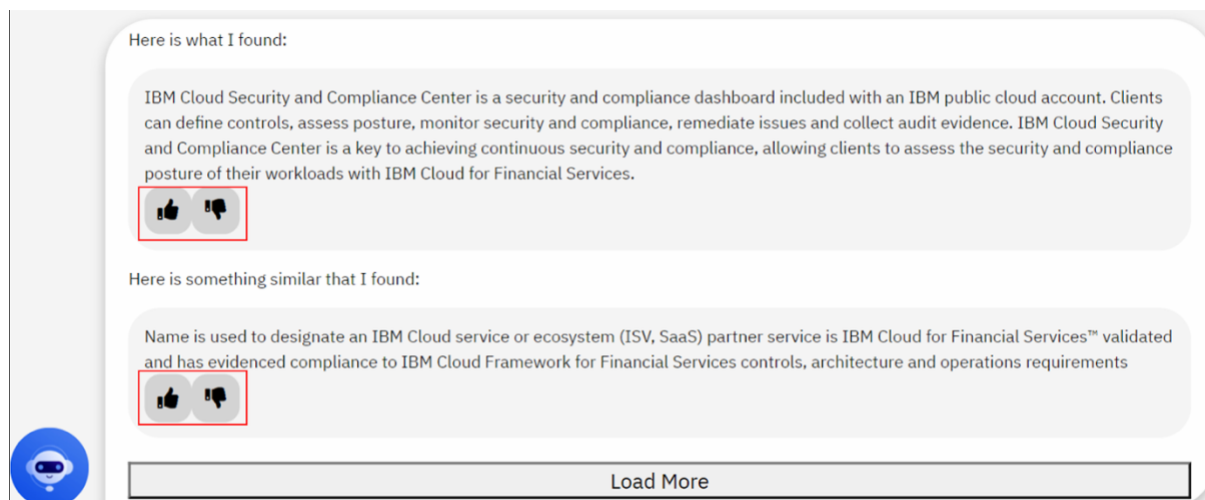


Users can ask as many questions as they would like in this format, continuing long flowing conversations to gather the answers they seek quickly and efficiently.



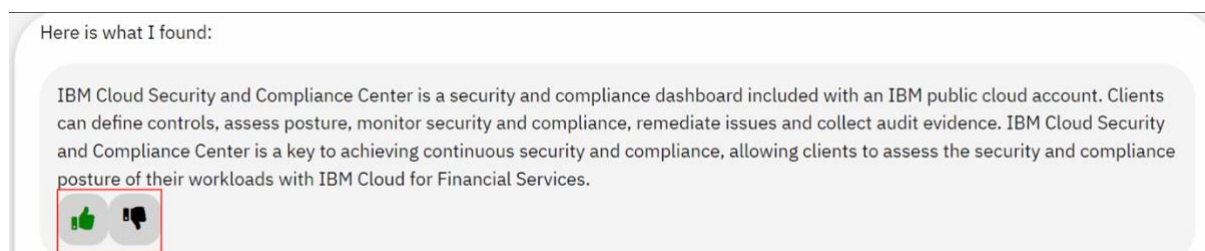
How to Rate a Query

Successful responses provide one or more answers from the chatbot, alongside two clickable icons for every answer.

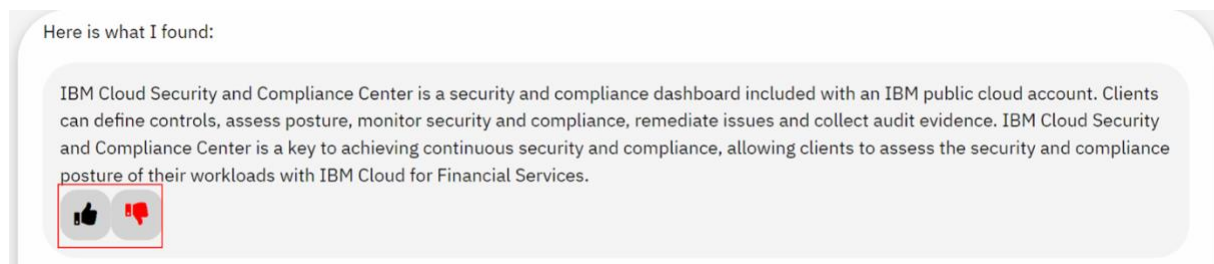


These icons can be used to rate the response as “*relevant*” or “*not relevant*” within Watson Discovery, so that the AI can be trained by users over time on which answers are relevant to the queries and which are not. After each click, the relevant information is passed to the backend server with the associated query and answer data, which is then saved in the “`/src/json/Ratings.json`” file for data aggregation purposes (saving the average rating as well as the number of ratings accumulated of a specific question-answer pair) as well as sent to Discovery to provide feedback for the model.

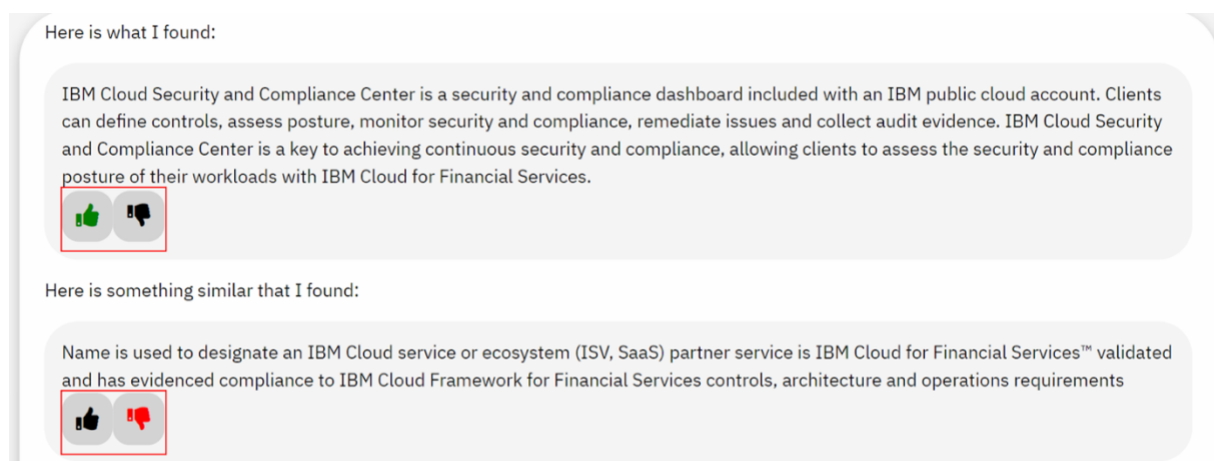
1. Relative rating – users can rate the answer to a query positively by clicking the “*thumbs up*” icon (the left icon), where it will become a green colour when clicked.



2. Not relevant rating – users can rate the answer to a query negatively by clicking the “thumbs down” icon (the right icon), where it will become a red colour when clicked.

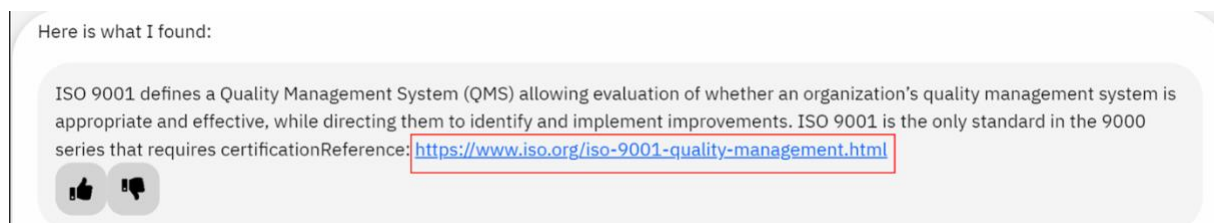


If users change their mind and wish to change their rating (assuming they haven't left the session), they can simply click the rating icon they wish to change to and the previous rating will be overwritten, or they can click the icon they have already selected to remove their rating. Furthermore, it should be noted that multiple ratings for multiple answers can be sent at one time.



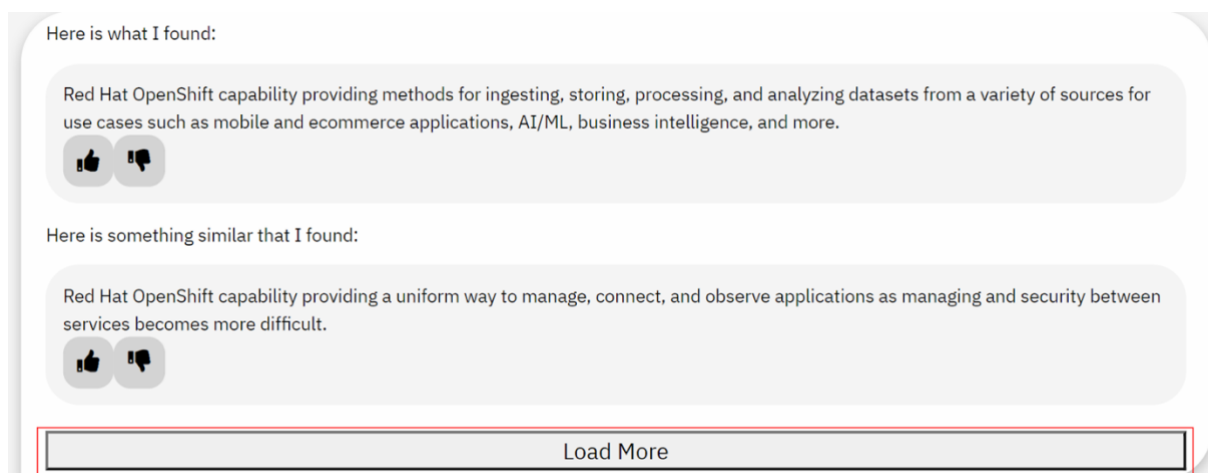
How to Access Links

Some successful responses return a blue highlighted URL, which users can click on to take them to the referenced web page.

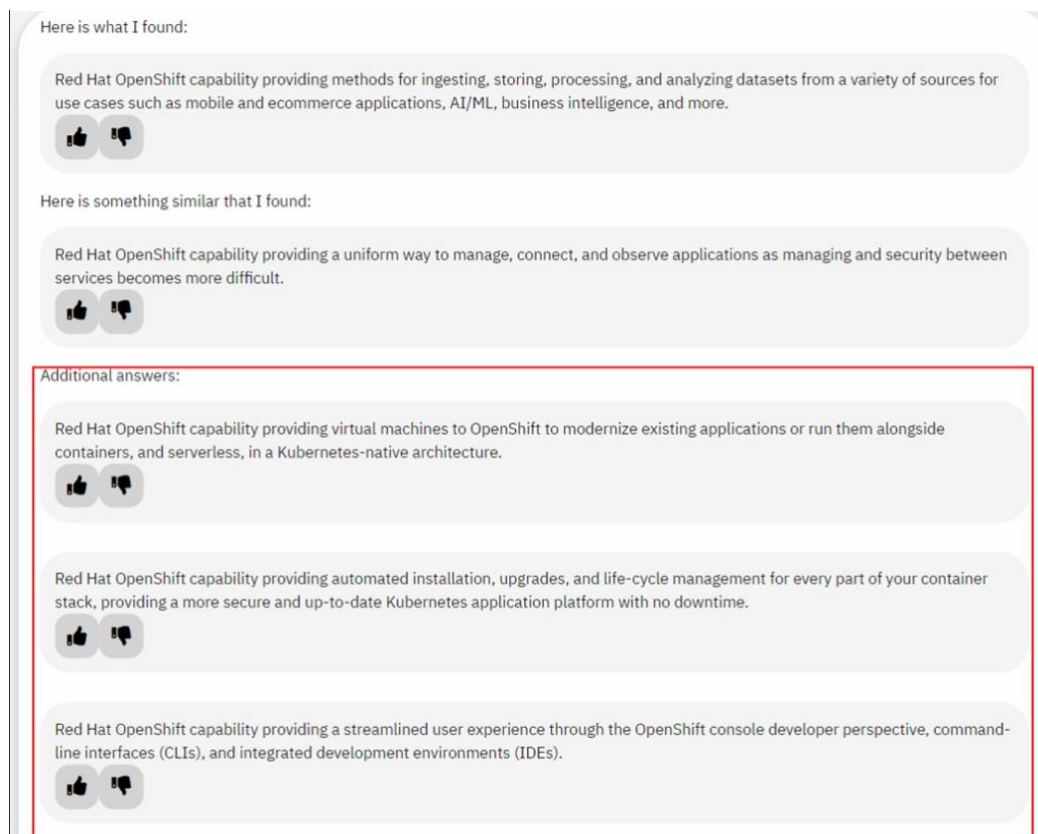


How to Load More Answers

Some Chatbot responses include a load more button at the bottom of the “*speech bubble*” to prompt users to load additional answers the Chatbot may think are relevant.

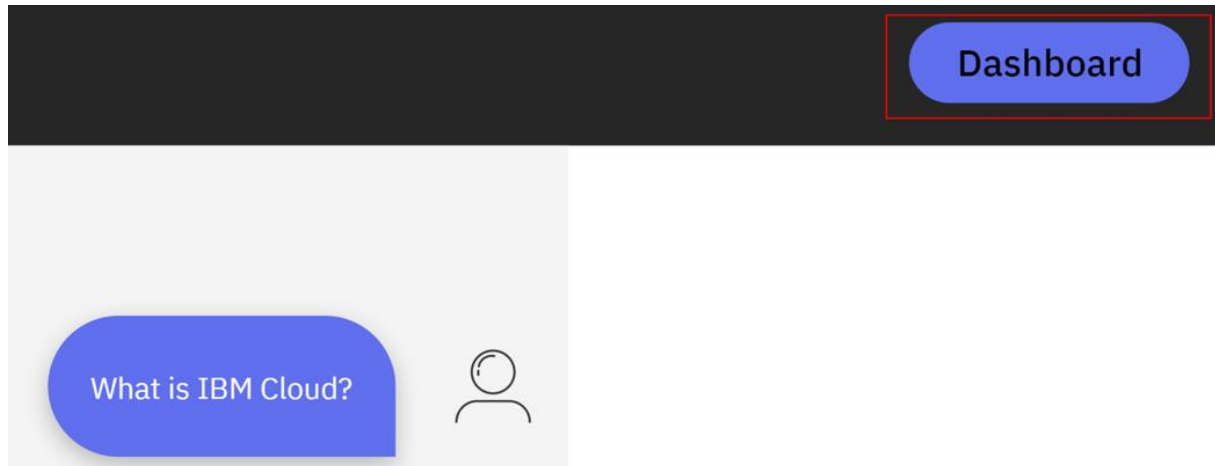


Clicking this button will display up to three more answers to show in addition to the default two answers displayed ordinarily.

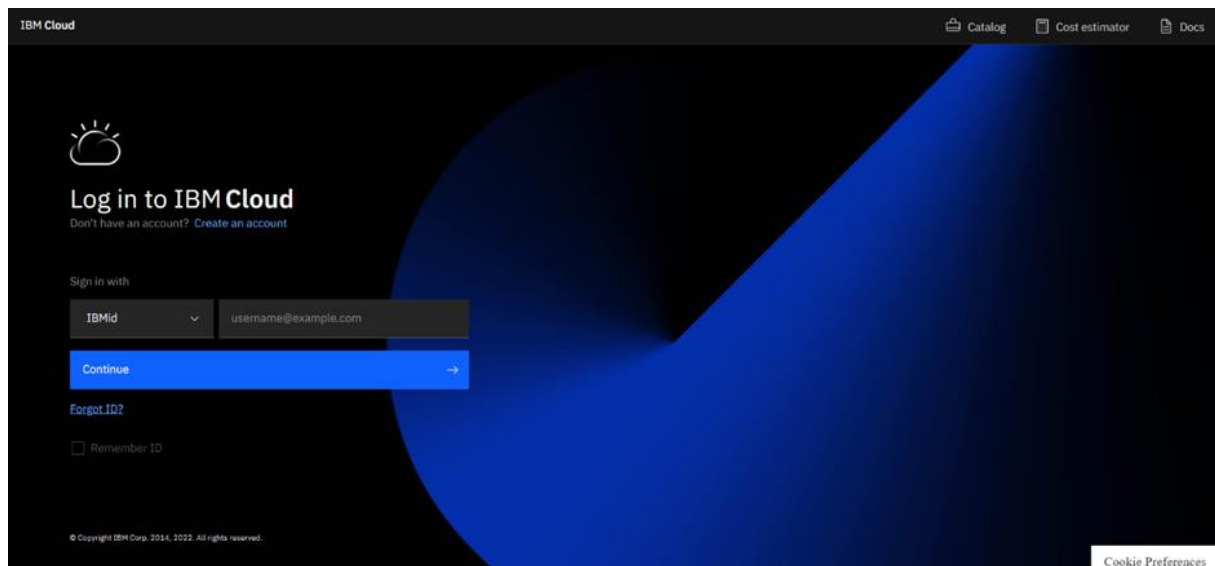


How to go to Dashboard (Admin)

An admin level user can click the button on the right side of the header/navbar at the top to access the IBM Watson Dashboard.



After clicking this, users will be taken to the standard IBM Cloud login page (<https://cloud.ibm.com/login>), where they can log in and use IBM Cloud technologies. This includes the same IBM Watson technologies that supported the development of this chatbot, thus giving users access to the AI model, the training database and user ratings.



Manage Data Sources

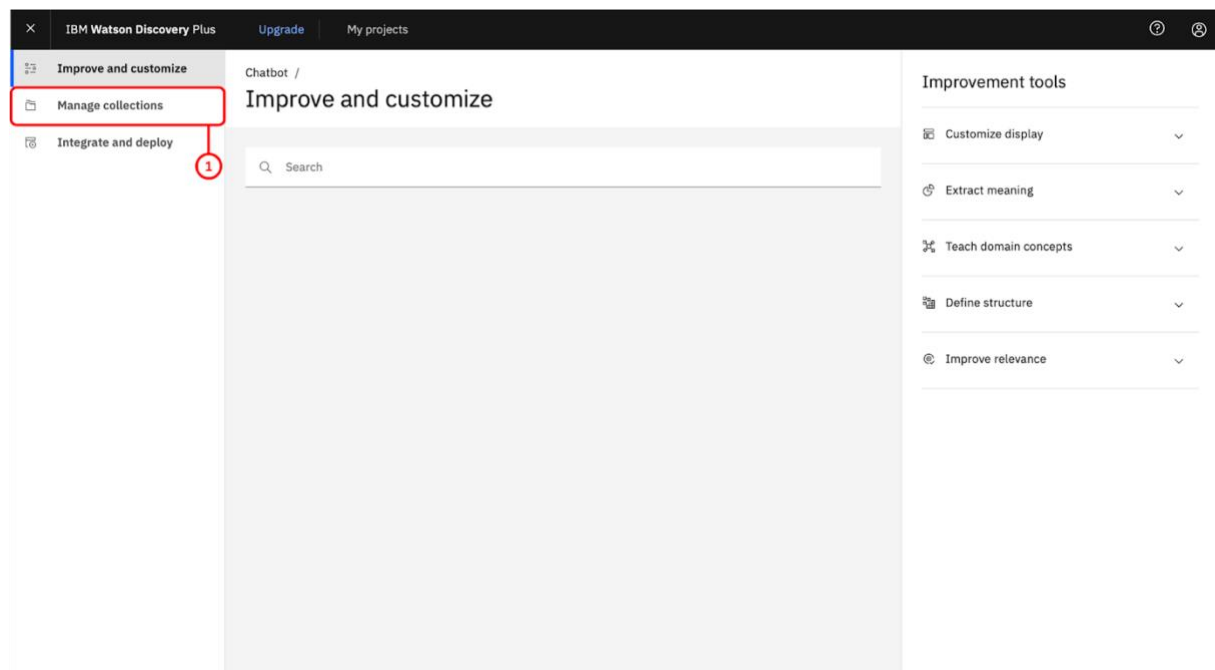
In the future, it is likely that data in the existing database becomes outdated and therefore needs to be deleted from the chatbot, or new data sources become available for the chatbot to train on. The following section focuses on how to add and delete data sources to the Discovery service.

Adding Documents

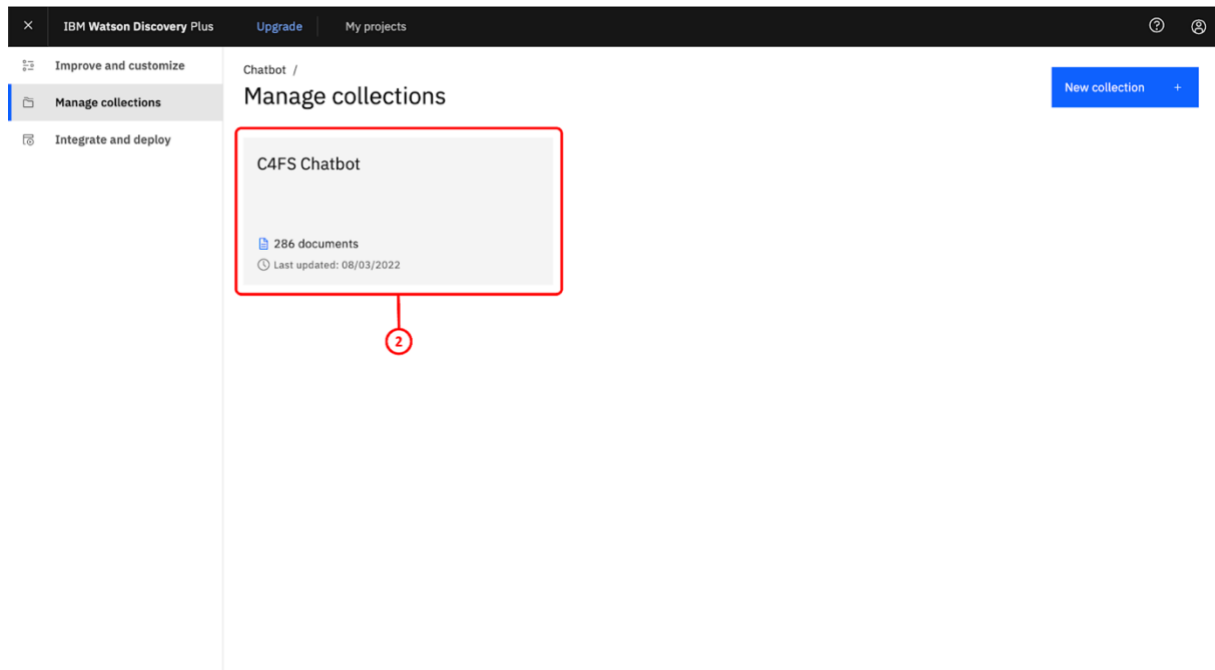
Discovery UI

Uploading Documents

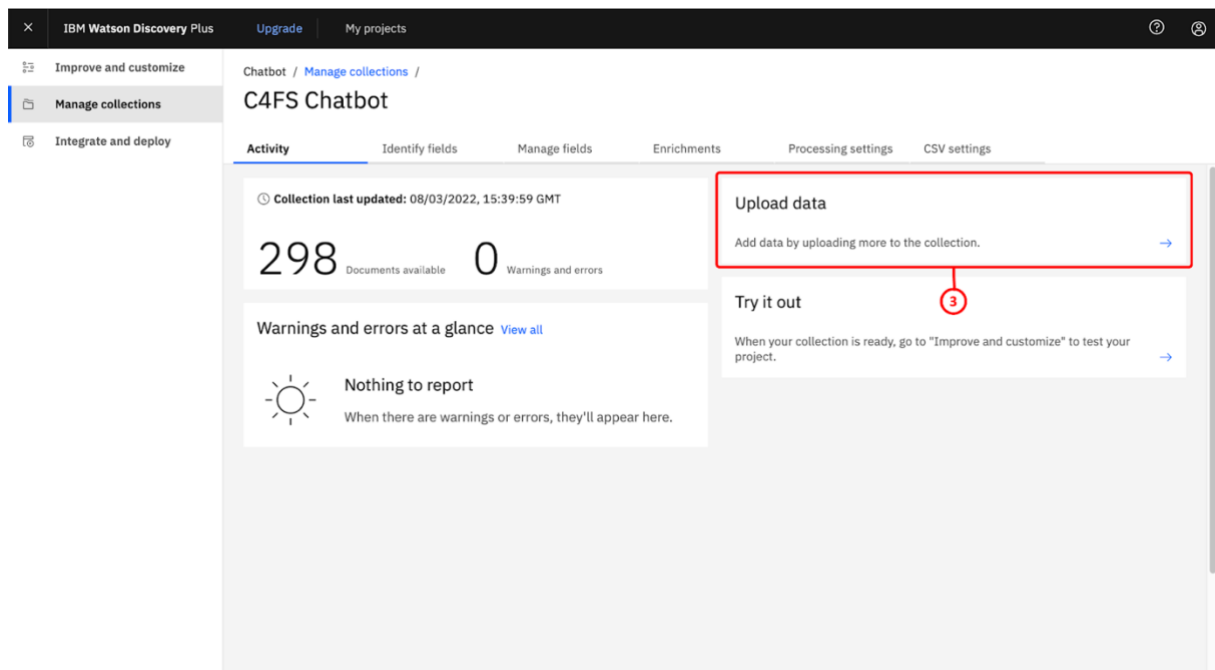
STEP 1: Press the “*Manage collections*” button in the left sidebar



STEP 2: Select the collection containing the data sources for the chatbot



STEP 3: Select the “*Upload data*” button and select the file to upload in the file picker that appears



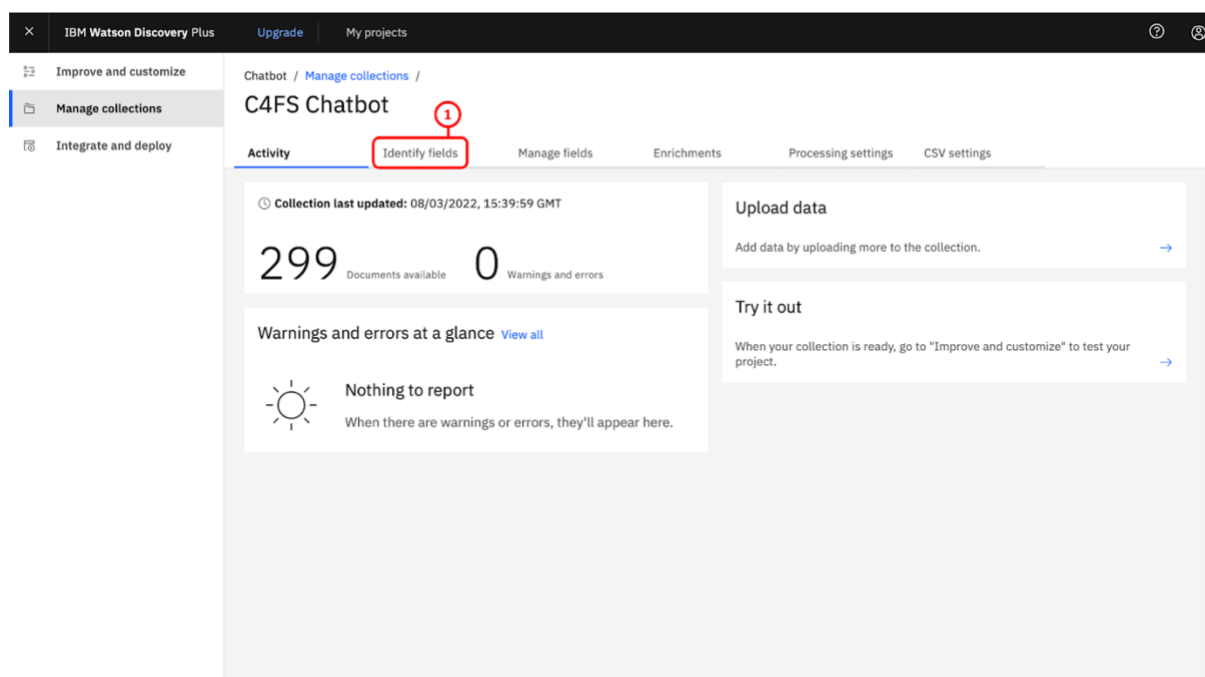
After the user has uploaded their document, the file picker dialog should disappear, and they should be automatically returned to the Discovery UI Homepage. Discovery will be uploading your document

in the background and as soon as it has finished the upload, a popup alert will appear to notify this. Once the document has finished uploading, it will immediately be accessible from the chatbot.

Retraining the Model

If the document that was added has a different formatting to any previous document uploaded, the AI model that labels and extracts data from the documents may need to be retrained. It should be noted that it is only possible to retrain the model on certain document types that vary based on what plan the Discovery instance is on. Lite plans support model training on PDF, Word, PowerPoint & Excel files, and Advanced plans extend this by also supporting the image formats PNG, TIFF & JPG (see <https://cloud.ibm.com/docs/discovery?topic=discovery-sdu>).

STEP 1: Press the “*Identify fields*” tab button



The user will now see the model training screen where passages of text in the uploaded document can be manually labelled to aid the Discovery AI to understand the rest of the document. On the right side of the screen there is a list of all the possible field labels that can be used to highlight the document. For the purposes of the chatbot AI, only the “*question*” and the “*answer*” labels are relevant as these are the only labels that the chatbot is set up to read.

All text that should be possible to return through the chatbot should be labelled as an “*answer*” whilst the “*question*” label should be reserved for headers that help split passages of text into sections with a common topic. All other text that should be ignored by the chatbot can be labelled with any other field label, however the recommended label to use is “*text*”.

STEP 2: Label the document page

To label the document, the user should first select the field label in the pane on the right. In the centre of the screen there are two pages, the left page is the actual document page whilst the right page shows the lines of text detected by Discovery, colour-coded based on what label is assigned to them. Once the user has selected the new field label, they should either select a line of text or drag over multiple lines on the page on the right to change the labelling – this should also change the colour of the line accordingly.

STEP 3: After labelling the page, press the “*Submit page*” button. To continue labelling more pages, return to Step 2, otherwise continue to Step 4. It should be noted that it is not necessary to label every page in the document as Discovery will learn from all the assigned labels to predict how to label every future page in the document.

STEP 4: Once finished labelling, press “*Apply changes to collection*”

The screenshot displays the IBM Watson Discovery Plus interface. On the left, a sidebar contains navigation options: 'Improve and customize', 'Manage collections' (selected), and 'Integrate and deploy'. The main header shows 'Chatbot / Manage collections / C4FS Chatbot'. Below the header, a tabbed interface includes 'Activity', 'Identify fields' (active), 'Manage fields', 'Enrichments', 'Processing settings', and 'CSV settings'. The central workspace shows a document page 'FAQs_doc.pdf' with a '2/2' indicator and a '4 / 68' page indicator. The document content is split into two panes: the left pane shows the original document text, and the right pane shows the text with colored highlights corresponding to field labels. A red box labeled '2' highlights a section of text in the right pane. A red box labeled '3' highlights the 'Submit page' button at the bottom right of the workspace. A red box labeled '4' highlights the 'Apply changes and reprocess' button in the top right corner. On the right side, a 'Field labels' panel lists various labels: 'answer', 'author', 'footer', 'header', 'question', 'subtitle', 'table_of_contents', 'text', 'title', 'table', and 'image'. A red box labeled '1' highlights the 'text' label in this panel. A red box labeled '5' highlights the 'Submit page' button in the bottom right corner of the interface.

Node.js Server

Whilst the server does not provide an API endpoint to upload a document from a client-side program, it contains functions that allow a document to be uploaded to Discovery. This means that in the future, an API endpoint could easily be developed to expose this function. There are two main functions that allow adding new documents to Discovery: `uploadDocument` and `uploadSplitDocument`.

Uploading a Single File Document

The `uploadDocument` function in “`/src/js/uploader.js`” enables you to upload a single document to Discovery. The extract from the code documentation corresponding to the function can be seen below:

```
(async) uploadDocument(documentId, name, buffer, mime) → {Promise.<boolean>}
```

Upload a file or document to Discovery

Parameters:

Name	Type	Description
documentId	string	The id of the document
name	string	The name of the document
buffer	NodeJS.ReadableStream Buffer	The content of the document
mime	string	The content type of the uploaded document

Source: [src/js/uploader.js, line 94](#)

Returns:

Document successfully uploaded

Type

Promise.<boolean>

Uploading a Multiple File Document

The `uploadSplitDocument` function in “`/src/js/uploader.js`” enables you to upload a JSON file containing many individual documents. This is useful in cases where you have a JSON file where the keys are the questions, and the values are the answers. This function handles splitting the parent JSON file into smaller documents and managing the upload and deletion of these smaller files as the parent JSON file is updated. The extract from the code documentation corresponding to the function can be seen below:

(*async*) uploadSplitDocument(baseDocumentId, baseDocumentName, data)

Upload a split-up file or document to Discovery

Parameters:

Name	Type	Description
baseDocumentId	string	The id of the document
baseDocumentName	string	The name of the document
data	object	The data to upload in JSON form. Each key-value pair will be uploaded as a separate document.

Source: [src/js/uploader.js, line 27](#)

Uploading an Updated Glossary

Currently the main data source for the chatbot is a word document that contains FAQs on IBM Cloud for Financial Services as well as a glossary section defining key terms. Whilst the FAQ section of this document is uploaded directly to Discovery UI using the steps in the “*Setup Watson Discovery Instance*” section, the glossary must be uploaded in a special way through the Node.js server. These steps are described below:

STEP 1: Split the parent FAQ & Glossary document so that you are left with a word document containing only the glossary section. This should be renamed to “*Cloud for FS Glossary.docx*”.

IBM Cloud for Financial Services™		April 6, 2021
Appendix A: Acronyms and Definitions		
1H	First Half (of the year noted)	
2H	Second Half (of the year noted)	
Accelerator	A tool, function, framework, defined process, industry / best practices that allow something of benefit to be delivered faster. Example: IBM Cloud for Financial Services™ with automated configurations allows the environment to be ready to accept data in two weeks which saves time and cost on the front-end and delivers business benefits sooner.	
Access Group	A group of users and service IDs can be organized so that the same access can be assigned to all members within the group by using one or more policies. With access groups, clients can streamline the access assignment process so that they can manage a smaller number of policies and reduce the number of policies in an account, which in turn increases performance. After groups are set up, they can start assigning policies by selecting an access group as the subject of the policy. For more information, see “Setting up access groups” Reference: https://cloud.ibm.com/docs/account?topic=account-groups	
Access policies	Access policies are how users, service IDs, and access groups in the account are given permission to access and take actions on account resources. Policies include a subject, target, and role.	

STEP 2: Replace the existing “**/src/data/Cloud for FS Glossary.docx**” document with the updated version

STEP 3: Restart the server by following the instructions in the “*How to Start and Stop the Node.js Server*” section. The glossary is automatically uploaded to Watson Discovery whenever the server is started.

Note: If the format of the glossary section in the parent file is updated at all, the `uploadGlossary` function found in “**app.js**” might need to be updated. The extract from the code documentation corresponding to the `uploadGlossary` function can be seen below:

`(async) uploadGlossary(filepath)`
Uploads the glossary to Discovery

Parameters:

Name	Type	Description
<code>filepath</code>	string	File path pointing to the DOCX file containing the glossary

Source: [app.js, line 701](#)

Deleting Documents

Node.js Server

The `deleteDocument` function in `uploader.js` enables you to delete a specific document from the Discovery data store. You simply need to pass the document id of the document to delete into the function and the document will be instantaneously deleted. This action cannot be undone, and the function doesn’t ask for any extra confirmation before deleting a document, so caution must be taken whenever calling the `deleteDocument` function. The extract from the code documentation corresponding to the function can be seen below:

`(async) deleteDocument(documentId)`
Delete a document from Discovery

Parameters:

Name	Type	Description
<code>documentId</code>	string	The id of the document to delete

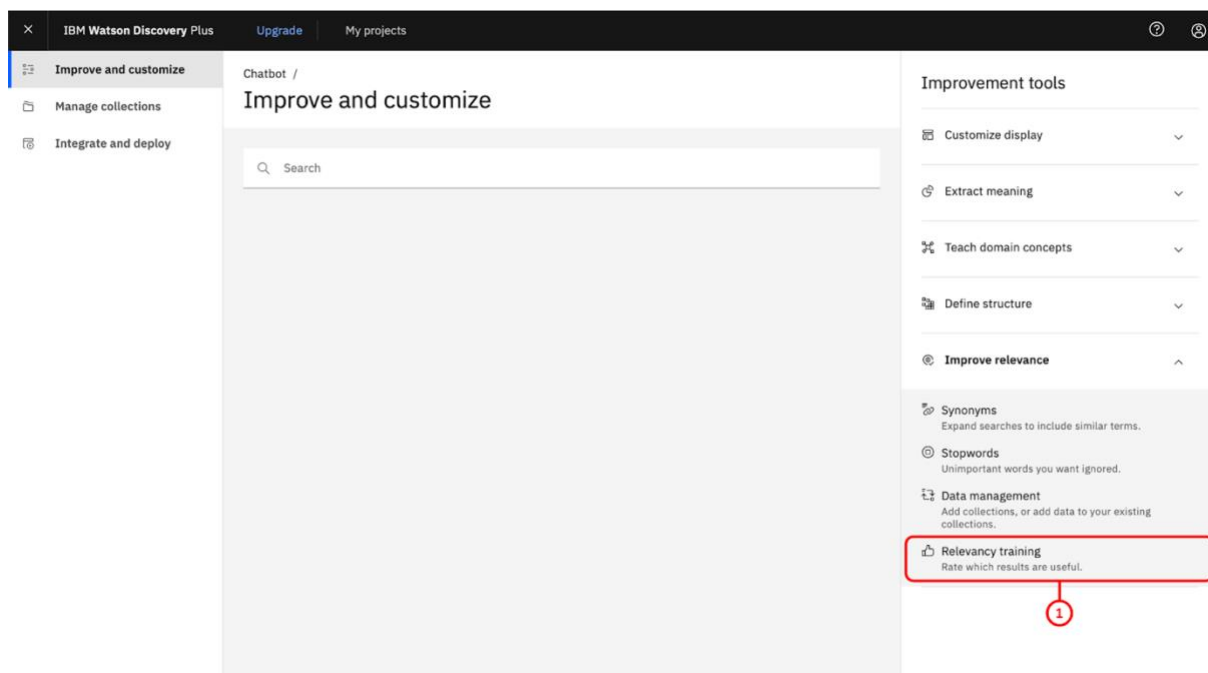
Source: [src/js/uploader.js, line 115](#)

Managing Document Ratings

Discovery UI

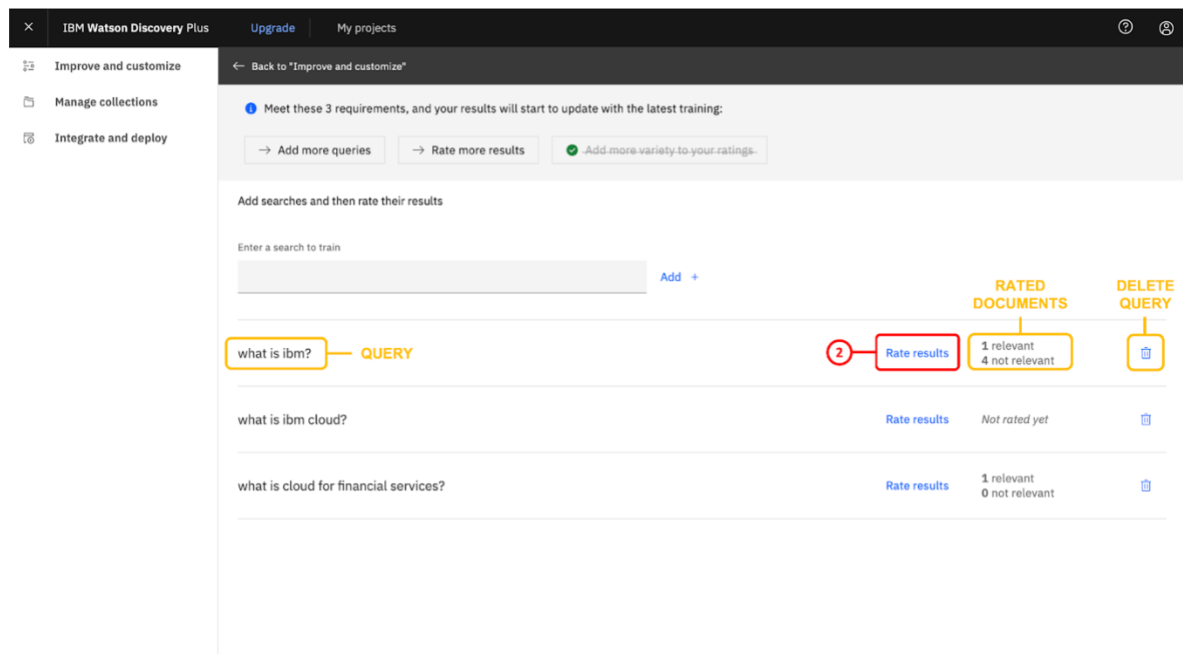
To access the screen to manage the document ratings, the user must first navigate to the Discovery UI Homepage to their Discovery instance. The steps to get from there to the pages where you can manage all the document ratings can be found below:

STEP 1: Select the “*Relevancy Training*” option in the right sidebar

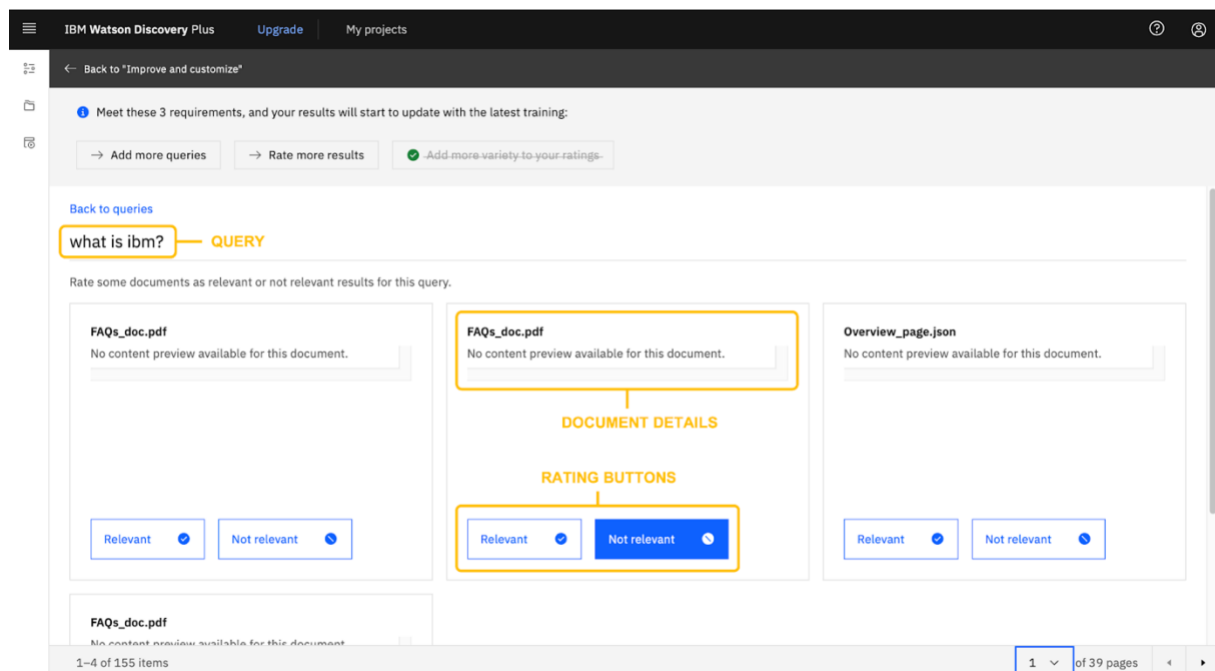


The user will now have reached the Collection Training Overview page, where they can see a list of all the queries that have ratings associated with it. Each list item shows the plaintext query as well as how many documents have been labelled as relevant and not relevant for that query. This view also allows you to delete a query, which removes all training associated with that query and should therefore only be used as a last resort method to fix the results obtained from the query. To manage the individual document ratings given for each query, carry on to Step 2.

STEP 2: Select the “Rate results” button for a query whose ratings needs exploring



The page the user now sees is the Query Training Overview page and it displays all the documents that have been given a relevancy rating for the selected query. The highlighted rating button displays whether the document was deemed relevant or not relevant as a result of all the training that has occurred. These buttons are clickable which means that the user can manually change the relevancy rating of each individual document if they believed it was falsely assigned.



Troubleshooting and Maintenance

Troubleshooting

This section details the first steps that can be taken when trying to troubleshoot or debug code during development or integration. This includes predefined tests, useful documents and frequent problems that can be encountered and some popular solutions.

Tests

Table 1: Unit Tests

No.	Item
1	Check server connection to IBM Watson.
2	Sending a message to IBM Watson.
3	Check server connection to GitHub.
4	Downloading data from GitHub.
5	Uploading documents to IBM Watson Discovery.
6	Sending a rating to a query to Watson Discovery.

Table 2: System Tests

No.	Item	Nature of Requirement
1	Chatbot start-up procedure.	Functional
2	Chatbot answering a query.	Functional
3	Chatbot providing other relevant answers.	Functional
4	User provides feedback for relevancy of query.	Functional
5	Chatbot offline functionality.	Functional / Non-functional

Table 3: User Acceptance Tests

No.	Item	Nature of Requirement
1	Relevancy of answers provided.	Functional / Non-functional
2	Chatbot answering latency.	Non-functional
3	Webapp UI is compatible and supported on modern browsers and Desktop and Mobile platforms.	Non-functional
4	Design is minimalistic and the purpose is clear.	Non-functional

There is a test plan report available which covers most major and minor aspects of the web app, down to minute details. The report details 3 types of tests: unit, system, and user acceptance. The unit tests cover integral API endpoints and atomic functions, without which the Chatbot would not be able to function. These tests have been pre-defined using Jest, SuperTest and Babel, and created in the file “/tests/unit.test.js”. The user simply needs to run the test script using “npm test” and check if every

test passes. The tests were designed to be comprehensive in their specific sections and thus if the tests pass successfully – in most likelihood, those sections are working as intended.

System and user acceptance tests have to be performed manually, but they are straightforward and easy enough to execute. The system tests assess the integration of various components and their functionality, including the connections between the frontend and the backend of the webapp. They were designed to completely cover all of the functionality of the webapp including various edge cases, thus if they pass successfully – the functional part of the webapp is working as intended. Lastly, the user acceptance tests focus on the UX of the webapp and thus evaluate performance metrics such as responsiveness, relevancy of answers and latency. Because of the subjective nature of this test category, the evaluation of whether the test has passed or failed depends on the level of standard that the user wishes to achieve.

For more information about test procedures, instructions, equivalence classes and more can be found on the test plan report (<https://app.box.com/s/kxb6mh2cmszus86t7xlnxd3lzviquh6hy>).

Note: When accessing the test plan report one must be logged in to IBM Box (<https://ibm.ent.box.com>) and ask for an invite to the box repository from Mark O’Kane (mark.okane@ie.ibm.com).

IBM Cloud Documentation

The core aspect of the webapp, the AI of the chatbot, is not stored locally on the server. Instead, it is located offsite at IBM Cloud services. The main services that are used are IBM Watson Discovery and IBM Watson Assistant. The documentations regarding the maintenance of these platforms are on the IBM Cloud documentation website for Discovery (<https://cloud.ibm.com/apidocs/discovery-data?code=node>) and Assistant (<https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started>). The main features that are accessible through these platforms are the database and training process of the underlying AI. If there is a need to alter the training data, reformat the structure of the model, or solve any particular issue related to the IBM API, the steps should be explained in great detail on the aforementioned website. The documentation is robust and provisioned by IBM, therefore it should be clear enough to understand and cover mostly anything that a user would need in the future.

Contact Details

Name of individual	Contact information	Reason to contact
Mark O’Kane	mark.okane@ie.ibm.com	Ask for invite to IBM box.com
James Jushchuk	jushchuk@ibm.com	Ask for invite to the demo Chatbot account
Arijus Lengvenis	arijus.lengvenis@durham.ac.uk	Invitation to the GitHub repository and additional questions

FAQs

Where and how do I access the website?

After following all of the steps set out in the prerequisites, the website can be reached on any browser by going to <http://localhost:8081/>. If the chatbot needs to be accessed from a secondary device, such as a mobile phone, make sure the secondary device is connected to the same Wi-Fi as the device running the server and enter `http://{IP_ADDRESS}:8081` in a browser (replacing {IP_ADDRESS} with the Wi-Fi’s IP Address).

When I access the website, nothing is loaded or the chatbot is displaying an error message. What should I do?

Firstly, check the internet connection of the computer. Make sure that the server is running and that there are no errors. Then open the inspector view in the browser and check for any errors there.

When I ask questions, the chatbot answers with an analogue of “I don’t know the answer”. What should I do?

Firstly, make sure that there is no error neither in the server console nor the browser inspect view. Secondly, make sure that the query that is sent is well defined and relates to IBM Cloud for Financial services. A safe question to ask is “*What is Cloud for Financial Services?*”. If after asking this question the chatbot responds with “*I don’t know*”, then check if the prerequisites have been completed or go to the IBM Cloud dashboard and check if the AI model is in good condition and has all of the relevant training data.

There is an error relating to GitHub that is appearing in the server console. What should I do about this?

The error might indicate that there is a problem with the auto-updater. There is a chance that Git has updated their terms of service which would disallow remote access to public GitHub repositories from Node.js servers. If this is the case, please read GitHub repository access policies (<https://docs.github.com/en/github/site-policy/github-terms-of-service>).

Are there people whom I could contact that would be able to invite me to private repositories or accounts?

See the “*Contact Details*” section in the user manual.

The server has reported an error with a certain API code. Where could I find out what it means?

A detailed description of every API endpoint and their responses exists in the API Documentation which is covered in the “*Documentation*” section. Furthermore, every API endpoint is covered by the unit tests (discussed in the subsection of “*Troubleshooting*” labelled “*Tests*”), therefore the test plan report (<https://app.box.com/s/kxb6mh2cmszus86t7xlnxd3lzvigh6hy>) might also help narrow down what could have gone wrong.

Note: When accessing the test plan report, one must be logged in to IBM Box (<https://ibm.ent.box.com>) and ask for an invite to the box repository from Mark O’Kane (mark.okane@ie.ibm.com).

Maintenance

This section includes the preventative checklist of things which need to be maintained throughout the product lifecycle so that no bugs or security vulnerabilities occur. This also includes the addition of features and scalability improvements by purchasing better service plans for Watson technologies, updating the local glossary items the intended way and keeping an eye on user feedback induced changes to the AI model.

Premium Watson Service Plans

The minimum versions of each service required for the web app to work are the Plus versions of Assistant and Discovery. These versions are necessary for API integration purposes as well as AI model importation respectively. The user should note that the services are not free and require a monthly payment of \$500 each for both Assistant and Discovery. In theory, if the user imported the model into Discovery and then converted the Discovery instance into the Lite version, then the user would only have to pay for Watson Assistant, though doing so does limit the number of API calls that can be done each month as well as the number of documents available.

In the case that the user wants to expand the capabilities and feature-set of either service, they also have a choice of purchasing the Enterprise versions of the Watson services. For more details on the price and the enhancements provided, please check the service plans for Watson Discovery (<https://www.ibm.com/cloud/watson-discovery/pricing/website>) and Watson Assistant (<https://www.ibm.com/products/watson-assistant/pricing>).

NPM Packages

The npm packages provided in the “**package.json**” may need upgrading after new versions of these node modules are released. The extensive list of the packages used can be found in the “**package.json**” along with their current version numbers. Newer versions of the modules might fix security vulnerabilities and bugs that were present, therefore it is integral that they are always kept up-to-date. To update all available node packages to the latest version automatically run “`npm install`”. This should be done at least monthly.

GitHub Repository Access Policy

Currently the auto-updater system uses git to directly interface with repositories on GitHub, to retrieve relevant information. It is possible that in the future, GitHub might prevent software to directly query and/or download files from other public repositories, which is where some of the data that trains the database is located at. Therefore, if a policy change were to occur, the automatic updating procedure using the implemented GitHub system would break and would need to be reimplemented in another way. Thus, it is imperative to keep this in mind, and if this system is malfunctioning at any point please

consult the GitHub terms of services forum found here: <https://docs.github.com/en/github/site-policy/github-terms-of-service>.

Keep Local Glossary Up-to-date

The glossary of the terms used for Clouds for Financial services is currently stored locally on the server in “**/src/data/Cloud for FS Glossary.docx**”. This document is then compiled during server start-up and converted into “**/src/json/Glossary.json**” and then uploaded to Watson Discovery. This has to be done, because the training process of Watson Discovery is very intricate, and the format of the question-and-answer pairs provided must be handled in a specific format. The AI can sometimes get confused about where the answer (the definition of the term) for a particular question (the term itself) resides in the document if the formatting is done in an unusual way (for instance - split into two columns dedicated for terms and definitions respectively). Therefore, if the user wishes to update the glossary or add additional definitions to it – to ensure that they are processed as intended, they should update the “**/src/data/Cloud for FS Glossary.docx**” file, extending the currently implemented format (question and answer pairs) and then after launching the server for the first time, the data will be sent to Discovery and immediately updated to the new version. **Beware – once the server starts, it will overwrite any previously existing glossary in the Discovery database and the process is irreversible!** The “*Uploading an Updated Glossary*” subsection in the “*Manage Data Sources*” section contains more information on this process.

User Feedback Effects on the AI Model

After integration, the users of this chatbot will be actively giving feedback to it whether the answers that it provided were relevant to the queries asked. Over time this will change the AI model to favour more upvoted questions. Currently these changes only start taking effect after 1000 user ratings (positive or negative) of a specific answer provided by the chatbot. This is to ensure that the effects of outliers and user error are dampened and only the answers that were agreed by the majority are kept. Despite this, there is still a possibility that some answers to some questions might get enough negative feedback to start affecting the AI model in a negative way. Therefore, it is important to keep this in mind and spend some time inspecting the progress of self-learning that is shown in the Watson

Discovery instance UI and changing it if necessary. For more information on this please refer to the “*Managing Document Ratings*” section.

Future Changes

- Some future changes that are not currently implemented are the integration of the IBM navbar. The current implementation is a placeholder with a button that acts as a link to the login page of IBM Cloud. This serves as a gateway for the admin to quickly access the Watson Discovery and Assistant instances from the chatbot itself for more sophisticated control over the underlying AI of the chatbot. In the future, there could be a more convenient form of integration, perhaps the Watson technology instance UI could be a side menu with the most useful features quickly accessible if the user is logged in as an admin.
- Another future feature could be the implementation of a live assistant button/option if the user cannot get a satisfying enough answer from the chatbot. There would be a button which would initiate the already pre-existing live correspondent chat window, where the user would be able to talk to a person. Though the design of the UI should promote the use of the chatbot first, and only after the user gets an unfavourable response – then it would recommend connecting to the live assistant.
- Currently the system is designed to work independently from the IBM server which runs the whole IBM ecosystem. Therefore, the auto-updater, which downloads the document templates from the IBM C4FS GitHub repository, does not have access to the object variable that is used to substitute the template fields with their corresponding values. So once the user (presumably an employee at IBM) is implementing the Chatbot server into the parent server, they should open the JavaScript file “`/src/js/updater.js`”, delete the currently existing dictionary variable called `replacementDictionary` on line 175 and replace the look-up variable (called `replacementDictionary`) on line 229 with the server one with any additional changes needed for compatibility.
- Whilst the document rating system that enables the chatbot AI to improve over time is designed to work in the background, it may sometimes be useful as an admin to manage all the ratings that have been given. Currently this is only possible through the IBM Watson Discovery Web

UI, however future implementation of such a system in the Node.js server is possible by making use of the Discovery API.

- The system could also have a local cache of the most frequently asked questions saved on the server itself. This would provide the ability for the webapp to have basic functionality even when there has been a connection loss when connecting to Watson Discovery. Furthermore, there would also be a latency improvement since this would bypass the need to query Discovery every time. This cache would also save a timestamp of when an answer was cached and then an arbitrary time limit could be set from which if a query that exists in the cache were to be called it would instead update it from Discovery to avoid serving outdated information. Finally, this would be feasible to do for a few popular questions since the data needed (a paragraph at most) is not very memory intensive after it has been processed by the chatbot AI.
- There could also be additional measures taken to persuade the user into providing a rating instead of leaving the chatbot immediately. This could be done by modifying the CSS and UI of the rating options to be more visible and enticing. There could also be some chatbot feedback provided back (for example, a thank you message) for giving a rating which would reward the user for their efforts.
- Another feature could be to provide the user with a “*Give Feedback*” button such that the user, if they wanted to, would be able to quickly report a bug or a very unexpected answer to the admin. This functionality would save the current chat data along with an additional user comment which would describe the problem itself and then send it to the admin, who would then be able to review it and implement fixes if necessary. This feature would notify the user that the conversation would be saved in order for the admin to be able to assess the problem.
- Finally, some additional time could be spent inspecting various security vulnerabilities along with additional spam protection to prevent a massive amount of API calls to be made to Discovery and the Node.js server.