MIAMI

CSC 431

ResBot

System Architecture Specification (SAS)

Team 2

Justin Prince	Requirements Engineer
Kevin Wright	System Architect
Vraj Patel	Scrum Master

Version History

Version	Date	Author(s)	Change Comments
1	4/5/2024	Kevin Wright, Justin Prince, Vraj Patel	First Draft

Table of Contents

Table of Figures

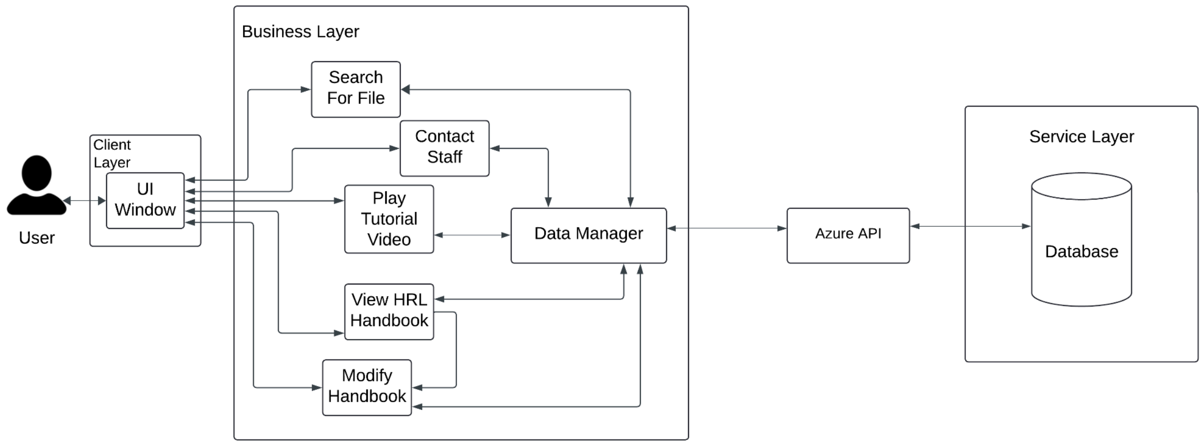
1. System Analysis

1.1 System Overview

This document describes the architecture design and specification of the ResBot Artificial Intelligence Chatbot (intended for the use of and integration with the existing University of Miami Housing and Residential Life channel on Microsoft Teams). The goal of the ResBot is to allow users - mainly student staff personnel - to have an interactive guide as they complete front desk operations. As users will need access to a digitized version of the handbook along with tutorials, and restricted staff forms, the system must be capable of interfacing with external components at the user's behest. This will be achieved through workflows designed in Microsoft Power Automate.

This system uses a three-tier architecture, which consists of a Client Layer, an Application Layer, and a Service Layer. The client server is the Microsoft Teams client which can be hosted on the web browser application, the Teams mobile application, or the Teams desktop application. The business layer consists of the entirety of the application logic, with mission-critical features such as: search for file, contact staff, play tutorial video, view HRL Handbook, and modify handbook. The service layer employs Microsoft Azure Database for MySQL in order to read and store data, with automated protocols to verify which user accounts have the authorization to access and modify database elements. The ResBot database will store user information as well as HRL file information, both of which can be accessed and retrieved by ResBot and be embedded into JSON formatted messages on the user-facing side.

1.2 System Diagram



1.3 Actor Identification

There are three types of human actors:

- HRL Student User: Can view/access any files in the database, but are restricted from making edits.
- Authorized HRL Administrator: They can view/access any files in the database, while also being able to make edits to master files such as the employee handbook.
- Developer Administrator: Can make changes to the system software, but cannot view/access material in the database or make edits.

1.4 Design Rationale

1.4.1 Architectural Style

We plan to utilize the three-tier architecture to implement ResBot. The 3 tiers will be as follows:

- UI Layer: This layer uses Microsoft Power Automate to create UI elements such as Interactive Tiles and customize the look and feel of the chat box.
- Application Logic Layer: This layer includes the logic for determining which files and/or what information to return to the user.
- Service Layer: This layer uses Azure Database for MySQL to host a scalable MySQL database to store and retrieve necessary files and data.

1.4.2 Design Pattern(s)

We predict that the Adaptor and Decorator design patterns will be featured in our project. The Adaptor design pattern is used to allow otherwise incompatible interfaces to communicate without having their source codes modified. This is relevant to the project as the goal is to allow various disparate areas of the Housing and Residential Life (HRL) information space - the HRL Teams channel, the online staff registry, the staff form hub and the HRL Youtube channel - to be compiled into a single application space. The ResBot will be able to interact with all of the aforementioned components.

As for the Decorator design pattern, it seeks to add functionality to an existing component without modifying its source code. This suits the project as the capabilities of the ResBot will greatly increase the functionalities of the HRL Microsoft Teams channel it will be hosted in without directly changing the backend of the channel itself.

1.4.3 Framework

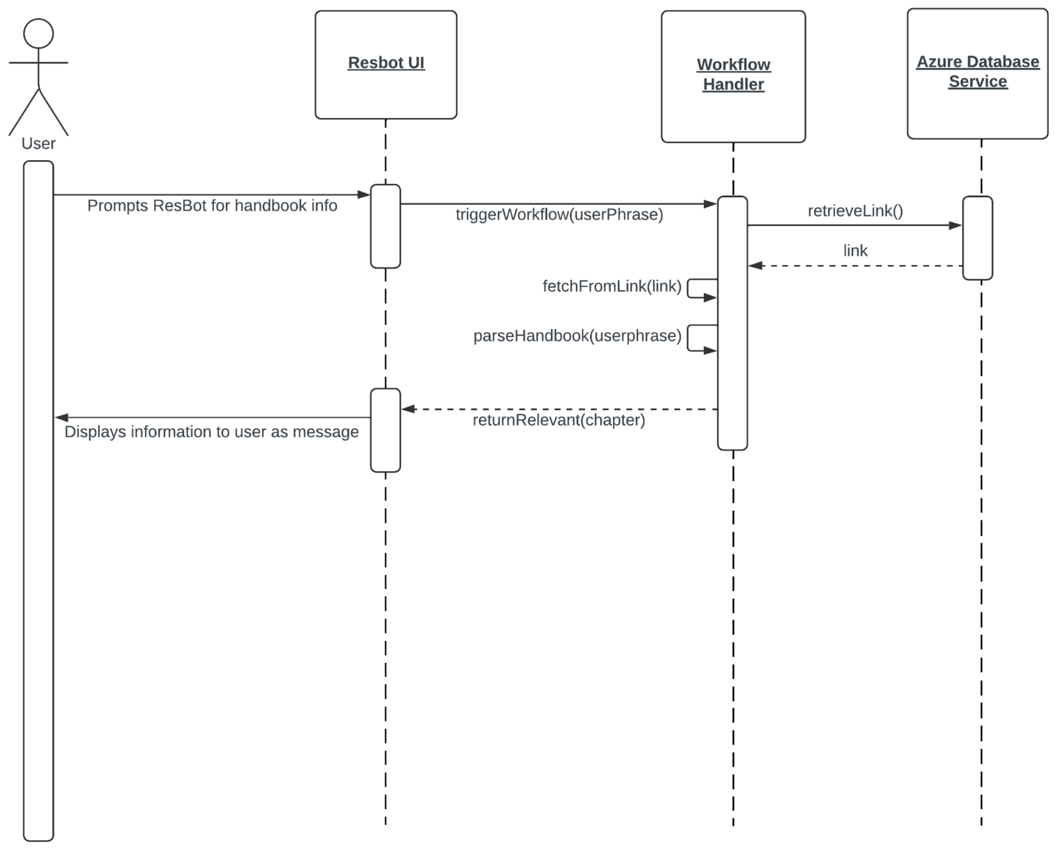
Resbot will be built primarily on Microsoft Power Virtual Agent (which is now a product within Microsoft’s Copilot Studio) in order to develop the chatbot. Through Power Virtual Agent, we will be able to format the UI of the chatbot. For the backend, we will be using JSON to format the responses returned by ResBot. This portion of the backend, which will not be managed by Microsoft Copilot Studio, will be written in Javascript. The inclusion of JSON formatting allows for the use of adaptive cards, a feature within

Microsoft Virtual Agent that makes it possible to display multimedia or customizable messages.

The Microsoft Teams application itself was built using the Electron framework and Node.js. The database will run through Azure Database Services, which is powered by Microsoft and will allow for smooth integration with Microsoft Teams and Microsoft Copilot Studio.

2. Functional Design

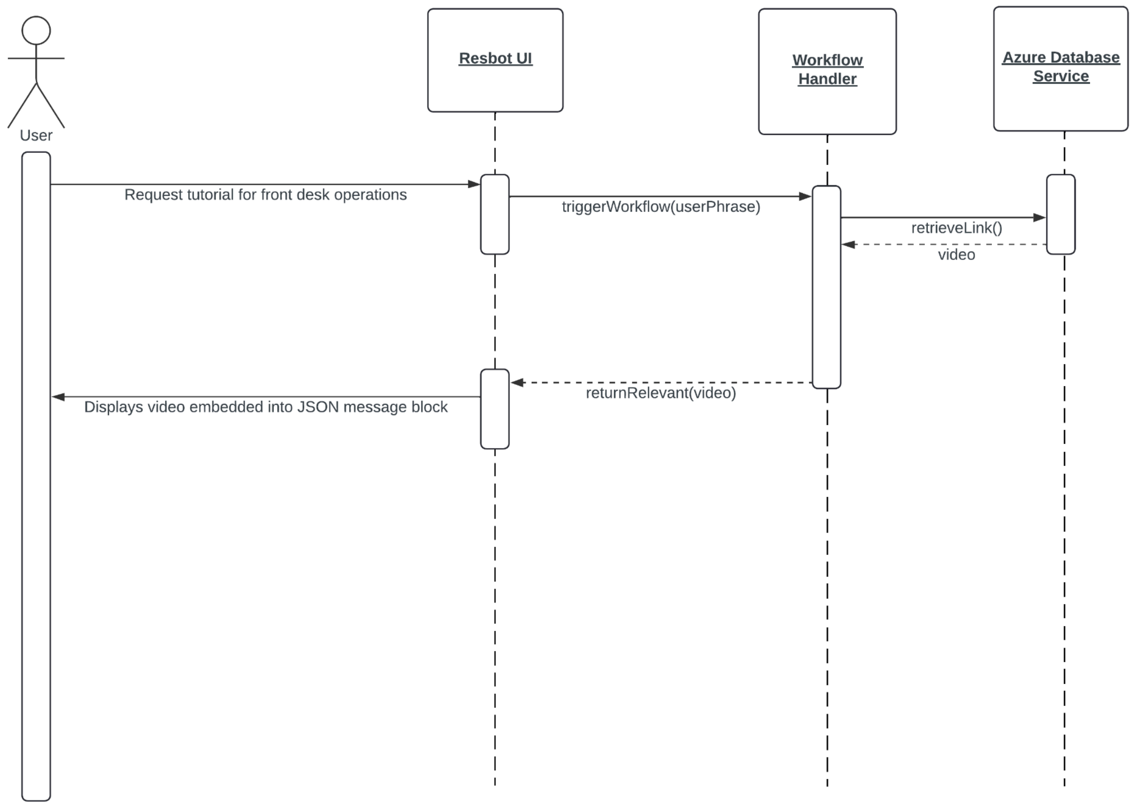
2.1 Fetch Handbook


User
Resbot UI
Workflow Handler
Azure Database Service
Prompts ResBot for handbook info
triggerWorkflow(userPhrase)
retrieveLink()
link
fetchFromLink(link)
parseHandbook(userphrase)
returnRelevant(chapter)
Displays information to user as message

- The user will ask ResBot for a specific piece of information that can be found from the handbook.
- After the user gives ResBot a prompt, in this case asking for relevant information from the handbook, the Workflow Handler (i.e. the backend) will be triggered, and keyword search through pre-fabricated responses to determine the next course of action.
- After it has been determined that the user would like to view the handbook, Power Automate will interface with the Azure Database Service by calling retrieveLink() in order to retrieve the internet link that hosts the handbook file.
- After retrieving the handbook file, Power Automate will call fetchFromLink(), with the link as the input, to access the contents of the file.
- Power Automate will then call parseHandbook() taking userPhrase as the input to search for the section relevant to the user's query.

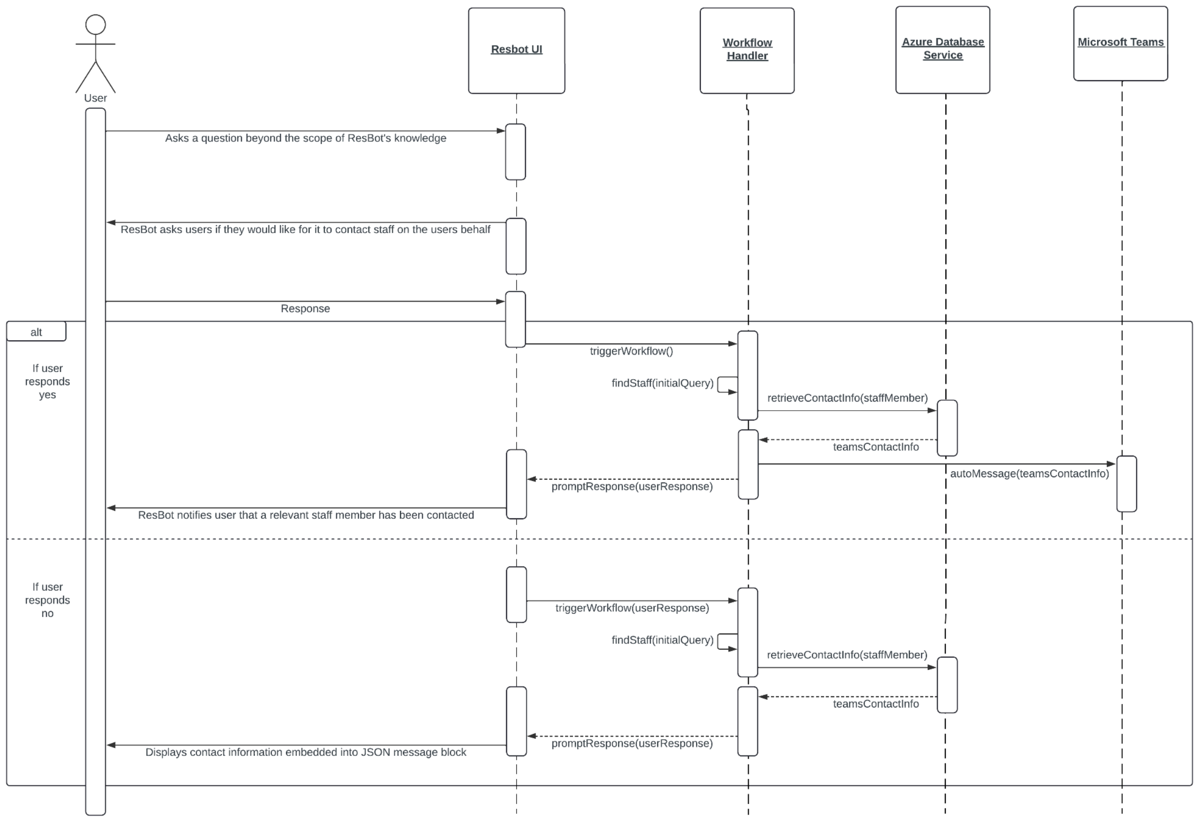
- Power Automate will then call the returnRelevant() Method, sending in as a parameter the relevant Chapter pertaining to the users query to the ResBot UI.
- ResBot UI will then embed the relevant information as a chat message in response to the user's query.

2.2 Play Tutorial Video


User
Resbot UI
Workflow Handler
Azure Database Service
Request tutorial for front desk operations
triggerWorkflow(userPhrase)
retrieveLink()
video
returnRelevant(video)
Displays video embedded into JSON message block

- The user will ask how to perform a certain front desk task, which is explained in the handbook and may have a corresponding tutorial video.
- The ResBot UI will trigger the workflow, taking the user's phrase and keywords as input, to allow the Workflow Handler to search for a relevant video.
- Workflow Handler will then call retrieveLink() to obtain the appropriate video link for the user.
- The Azure Database will return the link to the video file.
- Power Automate will receive the video file from the database and return it to the ResBot UI for it to be displayed.
- The ResBot UI will finally receive the video file and display the contents to the user based on their query.

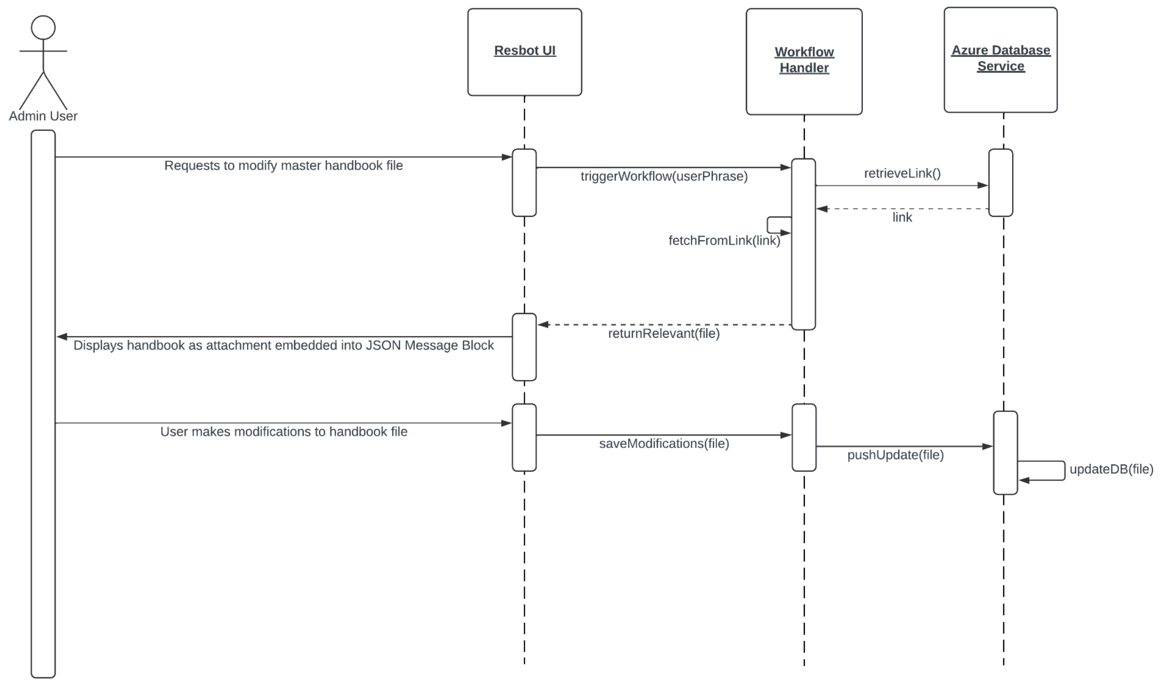
2.3 Contact Staff


User
Resbot UI
Workflow Handler
Azure Database Service
Microsoft Teams
Asks a question beyond the scope of ResBot's knowledge
ResBot asks users if they would like for it to contact staff on the users behalf
Response
alt
If user responds yes
triggerWorkflow()
findStaff(initialQuery)
retrieveContactInfo(staffMember)
teamsContactInfo
autoMessage(teamsContactInfo)
promptResponse(userResponse)
ResBot notifies user that a relevant staff member has been contacted
If user responds no
triggerWorkflow(userResponse)
findStaff(initialQuery)
retrieveContactInfo(staffMember)
teamsContactInfo
promptResponse(userResponse)
Displays contact information embedded into JSON message block

- The user will ask how to perform a certain front desk task, where the user's query does not contain any trigger phrases/keywords. This means that ResBot does not search through the database for the information, as it recognizes that there is no pre-existing information related to the query.
- The ResBot UI will display a message confirming that the user would like to contact a staff member for assistance. The message contains a "Yes" and a "No" button for the user to select.
- If the user selects "Yes," the ResBot UI will call the triggerWorkflow() function to let the Workflow Handler begin searching.
- The Workflow Handler calls the findStaff() function on itself with the initial trigger phrases from the query as input.
- Once the Workflow Handler finds the appropriate staff member, it calls the retrieveContactInfo function, with the staff member's name as input, to obtain the contact information from the database.
- The database will return the contact information to the Workflow Handler.

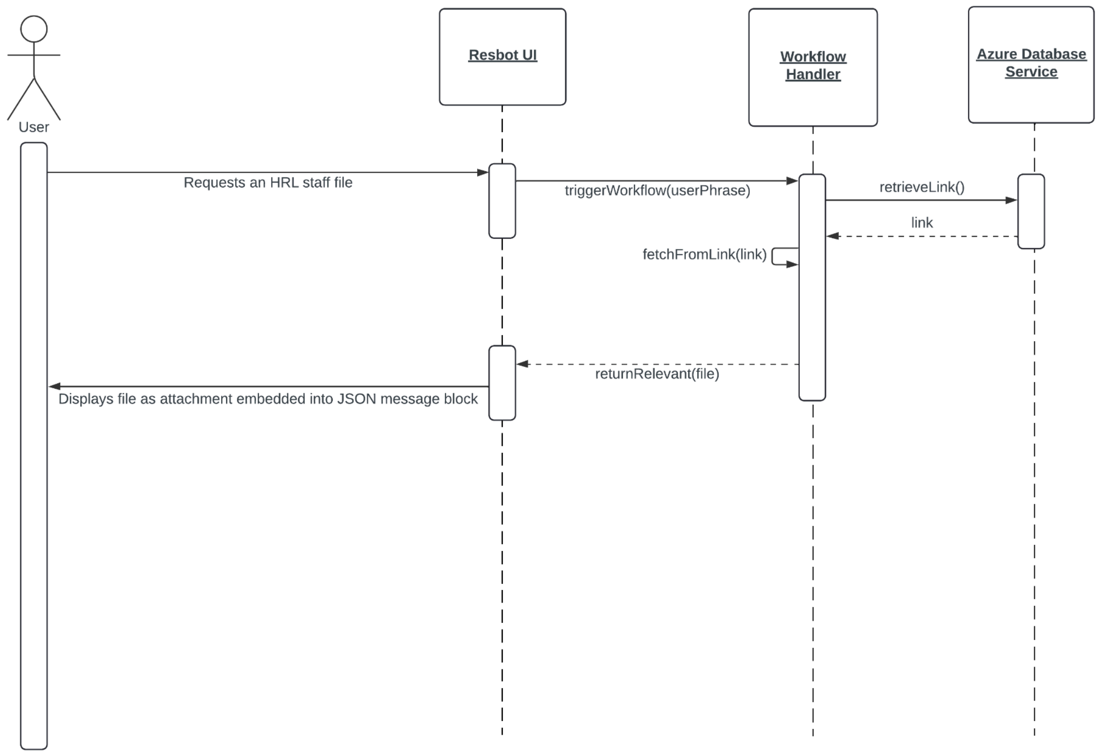
- The Workflow Handler will then call the autoMessage() function, with the staff member’s Microsoft Teams contact information as input, to send a message to the member letting them know that the user is requesting assistance.
- The Workflow Handler will send a confirmation message through the ResBot UI to the user, letting them know that the staff member was contacted.
- The ResBot UI will display the message confirming that the staff member has been contacted for assistance.
- If the user selects “No,” the ResBot UI will call the triggerWorkflow() function to let the Workflow Handler begin searching.
- The Workflow Handler calls the findStaff() function on itself with the initial trigger phrases from the query as input.
- Once the Workflow Handler finds the appropriate staff member, it calls the retrieveContactInfo function, with the staff member’s name as input, to obtain the contact information from the database.
- The database will return the contact information to the Workflow Handler.
- The Workflow Handler will not send a message to the staff member, but rather just prepare the message to return to the user
- The Workflow Handler will send a confirmation message through the ResBot UI to the user, providing the staff member’s contact information.
- The ResBot UI will display the message containing the staff member’s contact information, in the format of a JSON message block.

2.4 Modify Handbook


Admin User
Resbot UI
Workflow Handler
Azure Database Service
Requests to modify master handbook file
triggerWorkflow(userPhrase)
retrieveLink()
link
fetchFromLink(link)
returnRelevant(file)
Displays handbook as attachment embedded into JSON Message Block
User makes modifications to handbook file
saveModifications(file)
pushUpdate(file)
updateDB(file)

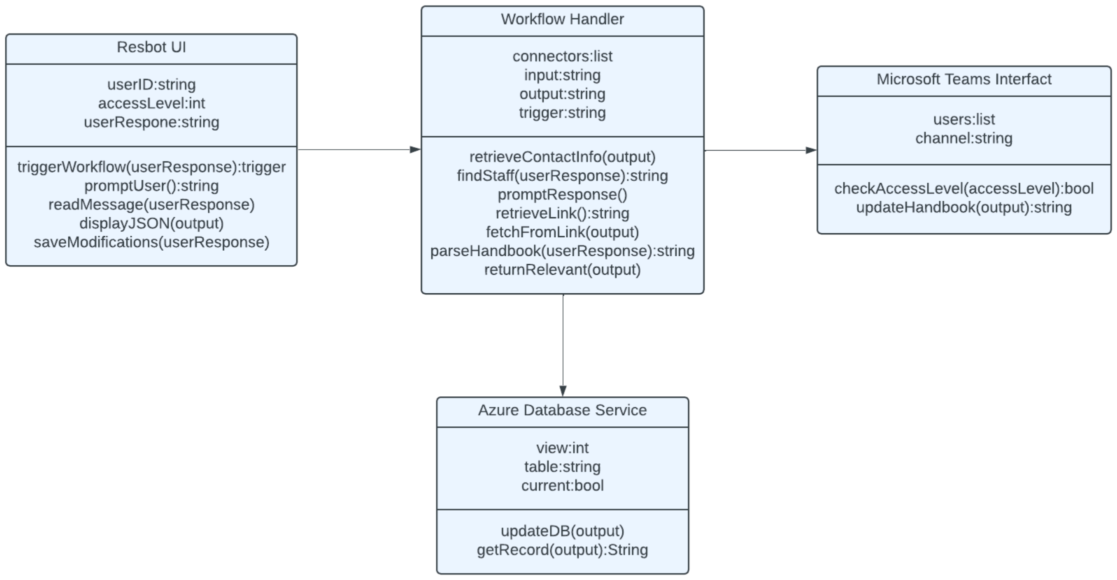
- The user with administrative access (already logged in to their admin account) tells ResBot via text input that they would like to edit the master handbook file.
- ResBot UI will call triggerWorkflow(), with the user's trigger phrase as input, to call on the Workflow Handler.
- The Workflow Handler will then call retrieveLink() to obtain the link to the handbook from the Azure database.
- The database will return the appropriate link to the Workflow Handler to be processed.
- The Workflow Handler will call fetchFromLink(), with the link to the handbook file as input, to fetch the contents of the file.
- The Workflow Handler will return the relevant file to the ResBot UI in order for the UI to display the file.
- The ResBot UI will display the master handbook file as an embedded attachment within a JSON message block.
- The user will then make modifications to the master handbook file, which will open in Edit mode due to their administrative access.
- The ResBot UI will save the changes made to the handbook file and send them to the Workflow Handler.
- The Workflow Handler will push the modified handbook file to the Azure database to replace the previous version of the file.
- The database will update its master handbook file.

2.5 Search for File


User
Resbot UI
Workflow Handler
Azure Database Service
Requests an HRL staff file
triggerWorkflow(userPhrase)
retrieveLink()
link
fetchFromLink(link)
returnRelevant(file)
Displays file as attachment embedded into JSON message block

- The user will ask ResBot for a specific file that can be found within the database, by giving the ResBot UI a prompt.
- The Resbot UI will call triggerWorkflow(), with the keyword trigger phrase as the input, prompting the Workflow Handler to keyword search.
- The Workflow Handler will call retrieveLink() to obtain the proper link to the appropriate file from the Azure database.
- The Azure database will return the link to the Workflow Handler.
- The Workflow Handler will call fetchFromLink(), with the link to the file as input, to process the contents of the link in order to access them and send them back to the ResBot UI.
- The ResBot UI will display the file as an attachment within the JSON message field.

3. Structural Design


Resbot UI
userID:string
accessLevel:int
userRespone:string
triggerWorkflow(userResponse):trigger
promptUser():string
readMessage(userResponse)
displayJSON(output)
saveModifications(userResponse)
Workflow Handler
connectors:list
input:string
output:string
trigger:string
retrieveContactInfo(output)
findStaff(userResponse):string
promptResponse()
retrieveLink():string
fetchFromLink(output)
parseHandbook(userResponse):string
returnRelevant(output)
Microsoft Teams Interfact
users:list
channel:string
checkAccessLevel(accessLevel):bool
updateHandbook(output):string
Azure Database Service
view:int
table:string
current:bool
updateDB(output)
getRecord(output):String