August, 2020

Engineering Team: David Gelbtuch
Eitan Jeselsohn
Eli Perl
Elie Shapiro

Project Manager: Alex Porcelain

Machine-automated Academic Communication Service (MACS)

Yeshiva University Tech Integration Project, Summer 2020

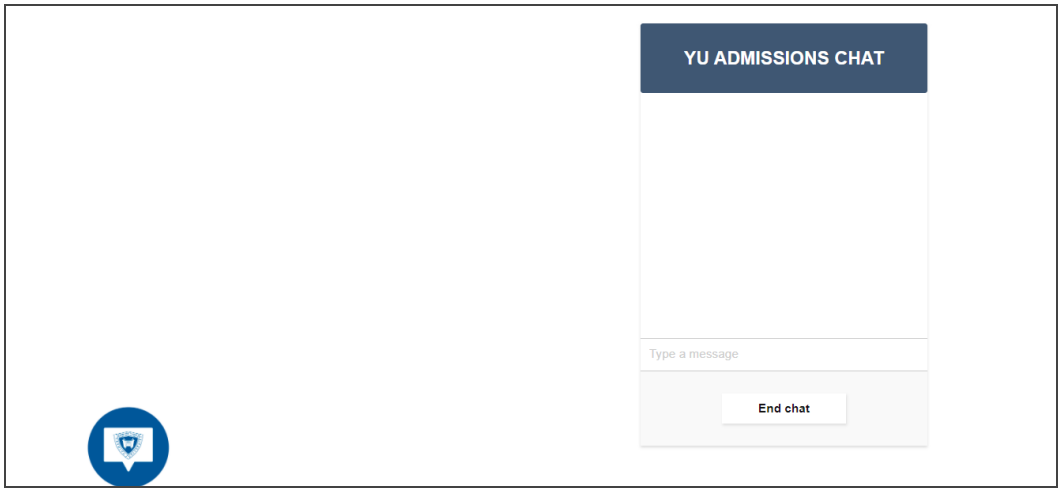
Abstract

The Machine-automated Academic Communication Service (MACS, aka "Max") aims to serve the Yeshiva University Office of Admissions (chat "agents") and prospective YU students (chat "users") by providing an easy and helpful chat infrastructure for interactive query and correspondence. MACS incorporates both an intelligent bot to perform user intake, answer basic questions, and route users to the appropriate human agents, as well a chat platform to facilitate live communication between users and agents.

Product Usage

User Experience

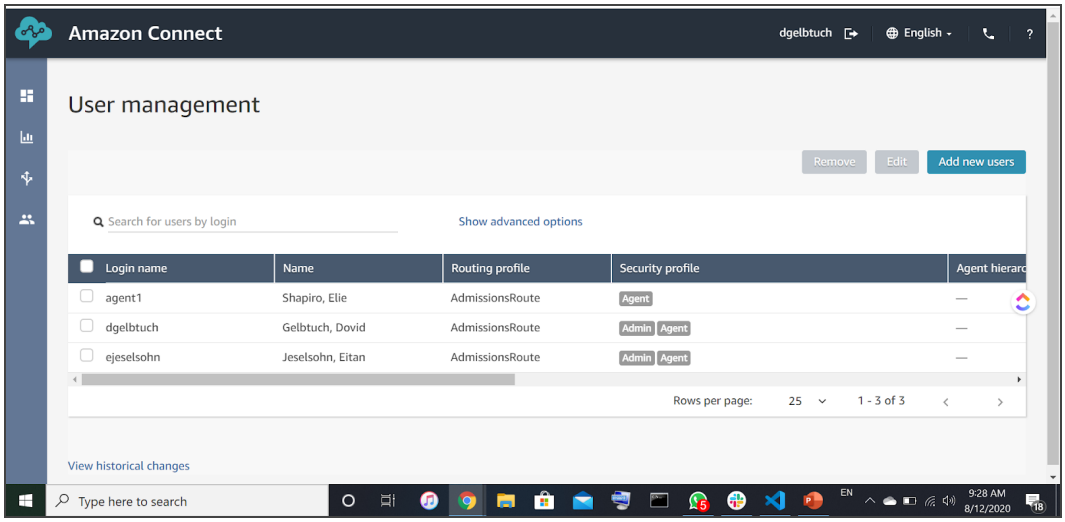
The user initiates a MACS-chat by clicking the widget on the Admission Office's website. A chat window will open, at which point the user begins to interact with the bot. The bot prompts the user for their name and email address, after which the user can ask the bot their question. The bot quickly checks if that question is covered by the FAQ's it's been programmed with; if so, it provides the user with the answer that best matches the question. If the user is unsatisfied with the answer, or if the bot cannot provide a response, the user is transferred to a queue waiting to continue the live conversation with an agent from the Admissions Office. In the event that no agents are immediately available, the user is notified of their place in the queue. If the bot's answer is satisfactory, and the user doesn't have any more questions, the bot will terminate the chat.



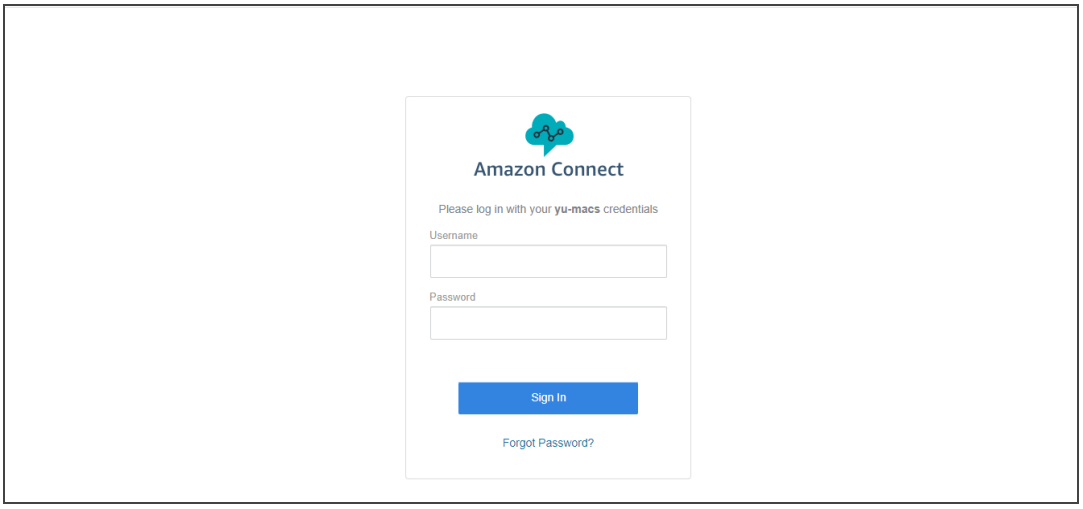
User Chat Widget on YU Website

Agent Experience

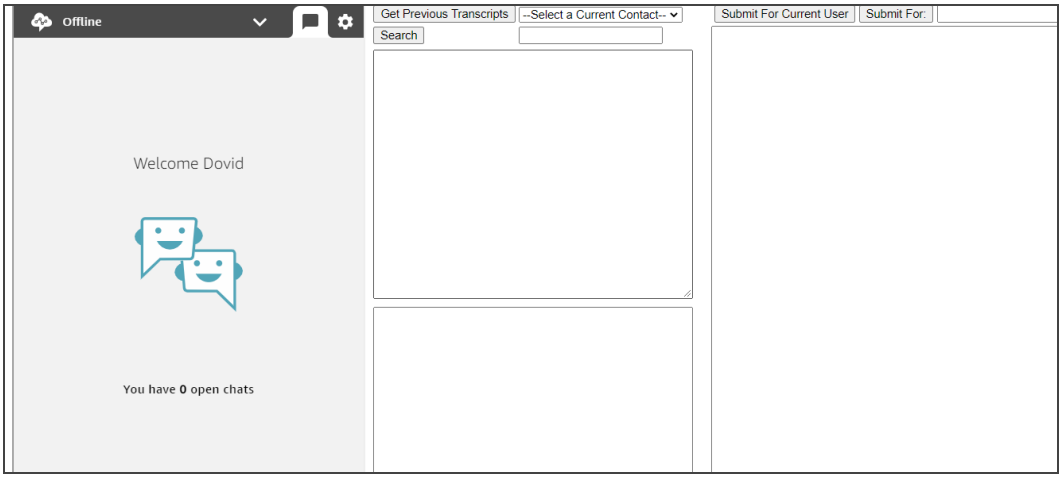
Each Admissions Office agent is provided by an administrator with a set of credentials to log into the live-chat portal. Here, after changing their status to available, they can initiate chats with users waiting in the queue. An agent can be actively involved with 3 chats at one time. In addition to chatting with users, an agent can enter notes relating to the chat in the dedicated sidebar, submitting those notes with the relevant user's email address to ensure it's correctly filed. Furthermore, the central panel of the portal is dedicated to pulling up past chat transcripts and agent notes filed under a particular user. These records can be accessed for any user currently involved in a chat or by searching for a user's email address.



Agent management page on the chat's administrative console



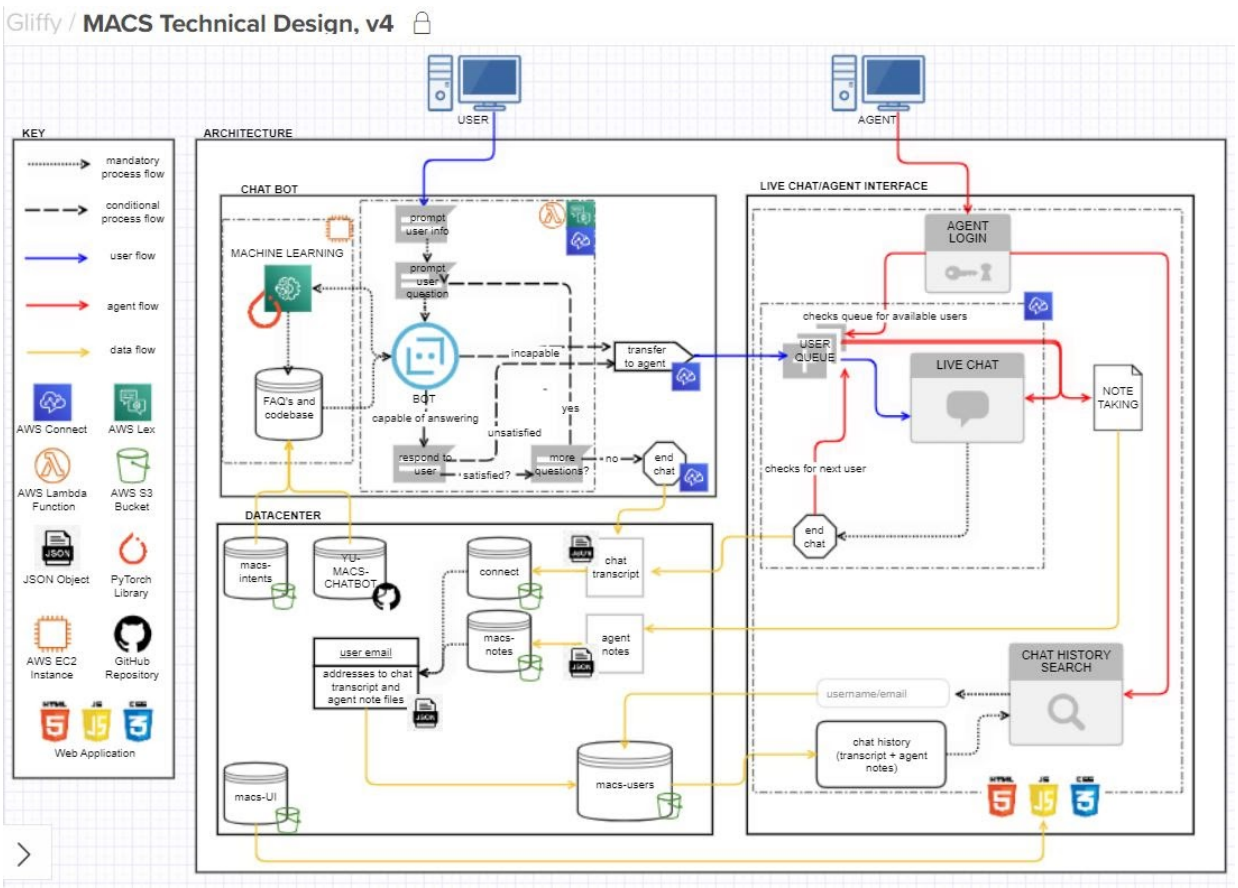
Agent login page



Agent portal with (left-to-right) chat window, noted chat transcript search window, and note-taking window

---

Technical Specs and Components



AWS Connect

- Connect Instance: yu-macs

  Access URL: https://yu-macs.awsapps.com/connect/login

  ARN: arn:aws:connect:us-east-1:244314150687:instance/49d6dd70-be38-4c69-a04c-9067a84a1daf

  Description: Amazon Connect instance containing administrative controls for queues and agent credentials, as well as the contact flow dictating the chat behavior.

- Connect Contact Flow: macs-cf

  ARN: <see above arn for yu-macs>/contact-flow/2fc43b04-9d7b-4be9-b6cf-b96e179e55d3

  Description: This contact flow maps the following behavior after a user initiates the chat:

  1. The bot introduces itself to the user.
  2. A Lambda function (refresh-repo) is called which performs a "git pull" command on the chat-bot's dedicated EC2 instance (i-0891c36d110b1c4aa), ensuring that the bot is equipped with the most recent code for the forthcoming conversation.
  3. The bot prompts the user for their name and email using the AWS Lex integration (MACS_Lex), and awaits user confirmation before proceeding.
  4. The bot prompts the user for their question using the AWS Lex integration (MACS_Lex), and returns a response to the user.
  5. If the bot can provide an answer, and the answer is satisfactory to the user, the bot asks the user if they have any further questions.
     a. If they do, the flow repeats step 4; if not, the chat is terminated.

6. If the bot cannot provide an answer to the user, or if the provided answer is unsatisfactory, the bot asks the user if they would like to be connected to a live agent.
    a. If not, the chat is terminated.
7. The bot transfers the user to the queue waiting to speak to a representative.
    a. As soon as an agent is available, they'll be connected with the user.
    b. If the user has to wait in the queue, they'll be notified of their place in line.
    c. If there are no agents available, the bot will inform the user and terminate the chat.

AWS *Lex*

- Lex Bot Instance: MACS_Lex

  ARN: arn:aws:lex:us-east-1:2443-1415-0687:bot:MACS_Lex

  Description: This Lex bot is integrated into the Connect contact flow to handle interaction between users and the intelligent bot. The bot is configured with 4 different "intents," each designed to translate user input into a specific response.

  1. UserIntake: Calls Lambda Function (saveUserIntake) to collect and store user's name and email address for later reference.
  2. ConvoFallback: Default intent, triggered irrespective of input content. Calls Lambda function (lex-predict) to analyze the input as a user question, and return the appropriate answer if possible.
  3. Yes/No: Used by Connect for user confirmations.
  4. ExitChat: Intent triggered by keywords such as "quit" or "exit" which indicate that the user would like to terminate the chat.

AWS API *Gateway*

- API: StartChatContact

  ARN: arn:aws:execute-api:us-east-1:244314150687:v5w0y6re08/*/POST/

  Description: This API serves as an endpoint for the user to interface to invoke the appropriate Lambda function (startChatContactAPI-StartChatLambda) when a chat is initiated.

AWS *Lambda*

- Lambda Function: startChatContactAPI-StartChatLambda

  ARN: arn:aws:lambda:us-east-1:244314150687:function:startChatContactAPI-StartChatLambda-1SV3MP71M02A6

  Description: Invoked by the user interface through an API Gateway (StartChatContact) when a chat is initiated, this CloudFront-generated Lambda function assists in setting up the chat environment.
- Lambda Function: startChatContactAPI-CustomResourceHelper

  ARN: arn:aws:lambda:us-east-1:244314150687:function:startChatContactAPI-CustomResourceHelper-5G9G1A5RX2JW

  Description: Invoked by the user interface when a chat is initiated, this CloudFront-generated Lambda function assists in setting up the chat environment.
- Lambda Function: refresh-repo

  ARN: arn:aws:lambda:us-east-1:244314150687:function:refresh-repo

  Description: Invoked by the Connect contact flow (macs-cf) when initiates a user chat, this Lambda function executes a shell script on the chat-bot's dedicated EC2 instance (i-0891c36d110b1c4aa) which navigates to the GitHub Repository containing the FAQ data and the machine-learning codebase (YU-MACS-CHATBOT) and performs a "git pull" command to ensure everything is up to date before proceeding with the ensuing conversation between user and bot.
- Lambda Function: saveUserIntake

  ARN: arn:aws:lambda:us-east-1:244314150687:function:saveUserIntake

Description: Invoked by the UserIntake intent of the Lex bot (MACS_Lex), this Lambda function stores the user's name and email address as Lex session attributes for later reference by Connect.

- Lambda Function: lex-predict

  ARN: arn:aws:lambda:us-east-1:244314150687:function:lex-predict

  Description: Invoked by the ConvoFallback intent of the Lex bot (MACS_Lex), this Lambda function takes the input and passes it to the chat-bot's dedicated EC2 instance (i-0891c36d110b1c4aa), where it executes the chat script. The script navigates to the GitHub Repository containing the FAQ data and the machine-learning codebase (YU-MACS-CHATBOT), activates the virtual environment which enables reference to the Pytorch machine-learning library, and runs the code which analyzes the user's input. If it can predict a response with 75% confidence or higher, it returns that answer to the Lambda, which in turn provides it to the bot and sets the question status to "Answered." Otherwise, it replies that it doesn't know, and the question status is set to "Unanswered."

- Lambda Function: macs-storage

  ARN: arn:aws:lambda:us-east-1:244314150687:function:macs-storage

  Description: Triggered when a new chat transcript is added to the S3 bucket for chat-transcript storage (connect-5f000137a7a1), this Lambda function parses the user's email from the transcript, and then appends the S3-address of the transcript to the user's log file (referenced by email) in the user data S3 bucket (macs-users). Chat transcripts are automatically saved to the S3 bucket when a chat is terminated, and are sorted by date.

- Lambda Function: macs-notes-storage

  ARN: arn:aws:lambda:us-east-1:244314150687:function:macs-notes-storage

  Description: Triggered when a new agent note is added to the S3 bucket for note storage (macs-notes), this Lambda function parses the user's email from the note file, and then appends the S3-address of the note to the user's log

file (referenced by email) in the user data S3 bucket (macs-users). Agent notes are saved to the S3 bucket when submitted by an agent from the chat portal, and are sorted by date.

- Lambda Function: update-intents

  ARN: arn:aws:lambda:us-east-1:244314150687:function:update-intents

  Description: Triggered when the file housing the FAQ data is updated in the S3 bucket (macs-intents), this Lambda function updates that information on the chat-bot's dedicated EC2 instance (i-0891c36d110b1c4aa) and executes the training script. The script navigates to the GitHub Repository containing the FAQ data and the machine-learning codebase (YU-MACS-CHATBOT), activates the virtual environment which enables reference to the Pytorch machine-learning library, and runs the code which retrains the machine-learning model on the new data to ensure that future chat-bot predictions are accurate.

AWS S3

- S3 Bucket: macs-users

  ARN: arn:aws:s3:::macs-users

  Description: This S3 bucket stores a log file for each user who initiates a chat. The file is titled by the user's email address, and contains the S3 addresses of all of the chat transcripts and agent notes associated with that user.

- S3 Bucket: connect-5f000137a7a1

  ARN: arn:aws:s3:::connect-5f000137a7a1

  Description: This S3 bucket stores chat transcripts saved by Connect. Chat transcripts are organized by date. When a new transcript is added, it triggers a Lambda function (macs-storage) which appends the S3 address of the new transcript to the user's log file.

- S3 Bucket: macs-notes

  ARN: arn:aws:s3:::macs-notes

Description: This S3 bucket stores agent notes submitted on the chat portal. Notes are organized by date. When a new note is added, it triggers a Lambda function (macs-notes-storage) which appends the S3 address of the new note to the user's log file.

- S3 Bucket: macs-intents

  ARN: arn:aws:s3:::macs-intents

  Description: This S3 bucket stores the FAQ data for the chat-bot to predict accurate responses to user questions. If the file is changed, it triggers a Lambda function which retrains the machine-learning model on the new data.

- S3 Bucket: macs-ui

  ARN: arn:aws:s3:::macs-ui

  Description: This S3 bucket stores the codebase for the agent chat portal UI. The contents include a README detailing the specifics of the codebase.

AWS EC2

- EC2 Instance: i-0891c36d110b1c4aa

  ARN: arn:aws:ec2:us-east-1:2443-1415-0687:instance/i-0891c36d110b1c4aa

  Description: This EC2 instance operates as a remote server, storing the machine-learning library and chat-bot codebase, and executing the training and prediction routines when invoked by the corresponding Lambda functions (update-intents, lex-predict).

GitHub Repository

- Remote Repository: YU-MACS-CHATBOT

  URL: https://github.com/jeselsohn470/YU-MACS-CHATBOT

  Description: This remote repository stores chat-bot codebase in the cloud. The contents include a README detailing the specifics of the codebase.

Contact Us

Please feel free to follow up with us at the following email addresses:

David Gelbtuch: dgelbtu2@mail.yu.edu

Eitan Jeselsohn: jeselsohn470@gmail.com

Eli Perl: elimelekhperl@gmail.com

Elie Shapiro: ashapir5@mail.yu.edu

Alex Porcelain: alexporcelain@gmail.com