Pool and Save

Indraneel Ray, Hemanth Teja, Durga Prasad, Vivek Kumar
Cloud Computing and Applications
New York University

I . ABSTRACT

Transportation in cities has been a major problem due to traffic and pollution. We aim to reduce these problems through a web based cab-pooling application service that is completely hosted on Amazon's cloud computing platform - Amazon Web Services (AWS). This application aims to present a solution to the problems mentioned above, using high-end cloud technologies. The application also aims at increasing security using facial recognition and multi-factor pin authentication. The application helps users share their rides with other users who happen to have a similar route or ETA.

II. INTRODUCTION

Pool and Save is a web hosted application, where users can log/sign in as a Driver or Rider. The application picks up the current location of both the rider as well as the driver using the GPS system. However, the destination is inputted by them manually. Once this data is fetched, the application maps the matching and relevant users with the drivers that travel along similar ETA. For Drivers, the application shows the nearby available Riders and vice versa. Therefore, two people can share the same car and reduce their cost of transportation and it also has environmental advantages.

III. IMPLEMENTATION

The user can log in or sign-up into the application through the Sign-in or Sign-Up page. They are required to verify their email-id's during the signup process. Driver's also have a similar flow for Sign-up and Sign-in.

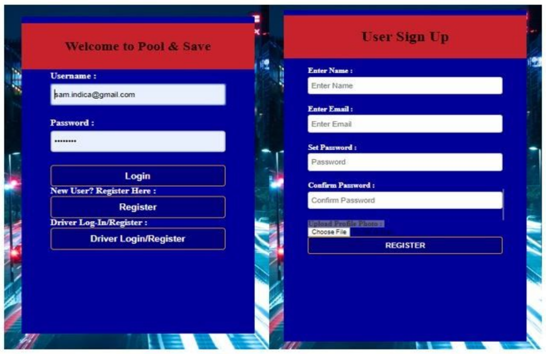


Fig. 1- Sample Sign-Up/Login Pages

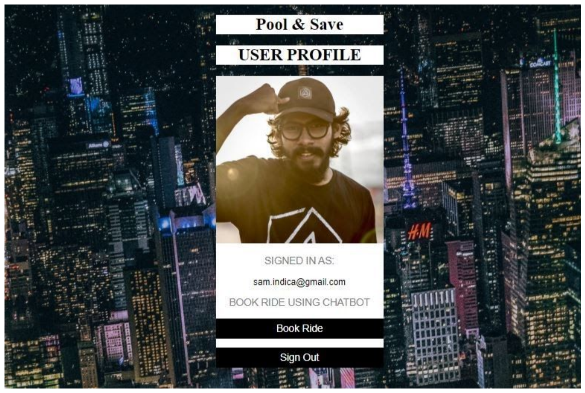


Fig. 2 - User Profile

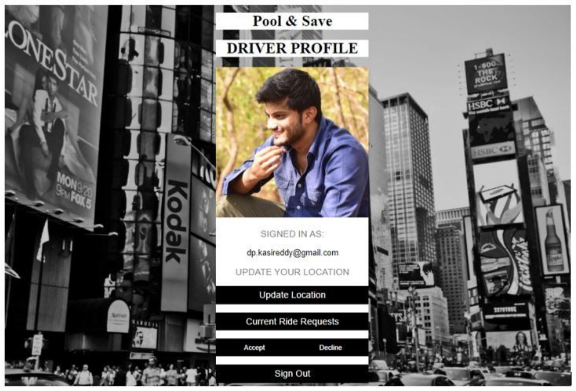


Fig. 3- Driver Profile

The application uses Google Maps to pinpoint the source and destination of both the Rider and Driver on the map. When a Driver clicks on the Search Ride button on the website, all the possible relevant nearby riders are displayed on the map. These users are are depicted by red icons on the map.

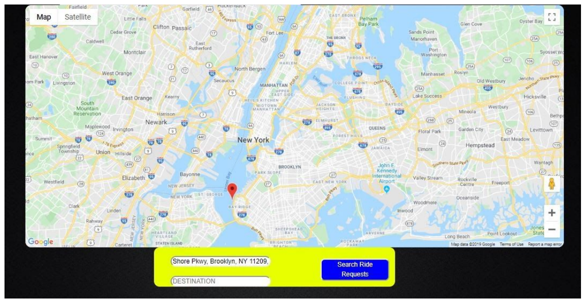
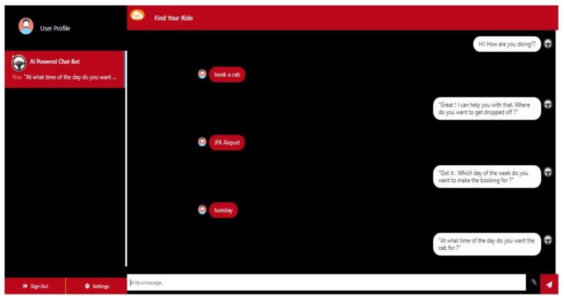


Fig. 4- Driver Options in Map

The riders are redirected to a chatbot where they can communicate with an NLP-powered chatbot. Users can give details about source, destination, time of ride, the number of people who will accompany that rider.



You: "At what time of the day do you want ...

Hi! How are you doing??

book a cab

"Great ! I can help you with that. Where do you want to get dropped off ?"

JFK Airport

"Got it . Which day of the week do you want to make the booking for ?"

tuesday

"At what time of the day do you want the cab for ?"

Fig. 5- Communication with chatbot

One of the best parts of this web application is the security it provides. Each user will have to upload a photo when he signs up for the first time. And whenever he books a ride, he will undergo a facial recognition check by an algorithm that checks if it is the same user who booked the ride. The driver does not get a request from these users unless and until their facial identity has been verified.

The driver chooses one path among all of his available options. The driver then picks up the users from their starting locations and the ride commences. When the driver reaches the destination, he is marked idle and can now search for new requests and a similar mapping will take place.

IV. ARCHITECTURE

This whole web application is engineered using the following Amazon Web Services- AWS Lex, S3, Lambda, Rekognition, DynamoDB, Cognito, CloudWatch, API Gateway, IAM, SNS, SES and SQS.

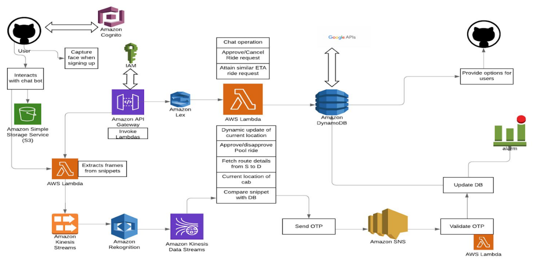

Amazon Cognito
User
Capture face when signing up
Interacts with chat bot
IAM
Amazon Simple Storage Service (S3)
Amazon API Gateway
Invoke Lambdas
Amazon Lex
Chat operation
Approve/Cancel Ride request
Attain similar ETA ride request
AWS Lambda
Google APIs
Amazon DynamoDB
Provide options for users
alarm
Extracts frames from snippets
AWS Lambda
Dynamic update of current location
Approve/disapprove Pool ride
Fetch route details from S to D
Current location of cab
Compare snippet with DB
Update DB
Amazon Kinesis Streams
Amazon Rekognition
Amazon Kinesis Data Streams
Send OTP
Amazon SNS
Validate OTP
AWS Lambda

Fig. 6- Architecture

V. AWS SERVICES USED

AWS Cognito: Login and Signup of the user and driver will be authenticated by Cognito. Userpools are hosted on Cognito, one for drivers and one for users.

AWS API Gateway: Multiple API Gateways are setup to ensure cohesive integration of all the AWS microservices used. API Gateways that are setup include - gateway to post images from the front end to the S3 buckets of the driver and users respectively, and gateways that helps in communication between DynamoDB instances, SQS queue and the respective lambda functions.

AWS S3: The Front-end for this web application is hosted on S3. The pictures of users and drivers are securely stored in S3 buckets. The front-end for Lex-chatbot is also hosted in S3.

AWS Lambda: Various lambda functions help in the communication between various applications. The lambda functions provides functionalities such as - attaining the nearby passenger, dynamically update the driver's location, provide lex chat communication, post driver's current location, post the rider's details and facial recognition for added security. .

AWS Lex: A Lex chatbot is hosted to satisfy the user requirements for travel. It queries the information about destination, date and time of travel and number of people accompanying the rider.

AWS Rekognition: Rekognition is used to compare the image of the user that was uploaded during sign up and current live image that is extracted by the camera through front-end.

AWS CloudWatch: It provides visualized logs, helps set alarms and timeouts depending on the requirements.

AWS DynamoDB: A table Rider is created to store all the details about rider's current location, date and time of travel. A table Driver is created to store data about driver's details such as driver location. Another table Rides is created to store the details about each user and driver's past rides.

AWS IAM: Helps provide managed access to AWS services and resources securely by giving respective permissions using roles and policies.

AWS SES: This service is used to send confirmation emails to users and drivers when they Sign up for the first time in our application.

AWS SQS: Queues are setup to store the chat details with lex, current wait list of available riders and drivers.

VI. LAMBDA FUNCTIONS

Attain Nearby Passengers: This function shows the list of ride requests to the driver and he/she can accept/decline a ride.

Attain Dynamic Driver Location: If the user ride request is accepted, this lambda function allows him/her to live track the location of the driver.

Lex Chat Operation: This piece of code takes the input from the user such as pickup location, date and time of the ride, drop location and the number of passengers that will be travelling and puts in the SQS of ride request..

Post Driver Destination Details: This function takes the driver's current location, and the final destination of the ride and puts this information in the queue to create a route for the driver to follow.

Post Rider Details: This lambda function shows the details of previous rides of a user which includes the source location, time of travel and destination location.

Facial Recognition: Before the start of the ride the passenger picture is matched with the profile picture of the user account using Rekognition. If it is matched, the driver gets a verification message.

VII. CONCLUSION

This application is scalable for multiple users and drivers to enjoy the service. In an AI-powered environment where the cab is driverless and everything is automated, this application would fit right in because of the facial recognition we run. The cab would only allow user to hop-in if verified by Rekognition ensuring the safety of all passengers.

VIII. FUTURE WORK

We plan to implement a pairing algorithm to match a ride request to a driver based on distance.
Currently, a driver can choose riders manually. Our future implementation will allow the driver to accept multiple ride requests based on the remaining number of seats in the vehicle.