Columbia Forum

Penghe Zhang(pz2244), Ruochen You(ry2349), Hang Yin(hy2568)

Abstract - **Columbia University does not have its own forum for the time being. An exclusive forum for Columbia will provide a platform and subsequently promote information sharing. Thus, our team came up with the idea of building our own forum. In our work, we deployed a single-page application (SPA) by React and AWS services, meanwhile, we also implemented news feed section through big data technology tech stack and job hunting section with real-time crawler.**

Keywords - **AWS, Serverless, Crawler, Big data, Cloud Computing, Microservices, SPA**

1. INTRODUCTION

This is the final project for Cloud Computing and Big Data. We have built a forum especially for all people related with Columbia University, like ungrads, masters, phds, teachers, staffs and also alumnis. We built this platform which promotes efficient communication, share all fun things happening in the campus, and get to know the world's news and job opportunities. By using the Columbia University Campus Forum, students can discuss academic questions and arrange after-school activities easily.

Columbia University students always need a dedicated forum where they can exchange valuable information and conduct social activities. In the fierce study competition environment, the Columbia Forum provides students with a pure land of self-reliance. Before entering a career full of uncertainty, the Columbia Forum provides students with an efficient and transparent job search experience.

In this forum, we would like to provide three basic sections including *Job Hunting*, *Daily Life*, and *News Feed*. In the Job Hunting section, we would like to provide the latest job hiring events posted on those job websites. In the Daily Life section, we set up a forum for everyone say anything on the website, you can send a post, reply to somebody, and you will get to new friends. In addition, the news feed part will also provide the newest hot topics for the users, which will save a lot time for them on browsing different sites.

Fig. 1 Online forum [1]

Our application is a single-page web application, and we implemented it with React and Redux. Also, for the back-end part, we set it a serverless back-end. It makes full use of Amazon Web Services (AWS).

Section 2 will introduce the technologies we used in our application. Section 3 will focus on what the architecture is like and how we implemented our application. Section 4 will show the demo of our application and all the instructions of the web. Section 5 will tell about our summary and feeling of what we learned in this course.

2. RELATED TECHNOLOGIES

After we planned to develop the Columbia forum, we reviewed all the materials covered in the course Cloud Computing and all of technologies introduced by Prof. Sambit. Our group decided to include as many technologies introduced in class as possible. In this section, we will introduce all technologies we applied for our application from front-end to back-end.

2.1 Front-end Technologies

JavaScript - JavaScript (JS) is a lightweight interpreted programming language with first-class functions. It has good compatible with the front-end.

React[2] - React is a JavaScript library for building user interfaces. React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Redux[3] - Redux is a predictable state container for JavaScript apps. It is predictable and centralized. It will help you write applications that behave consistently.

2.2 Back-end Amazon Web Services[4]

API Gateway - Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.

Amazon Amplify - AWS Amplify makes it easy to create, configure, and implement scalable mobile and web apps powered by AWS. Amplify seamlessly provisions and manages your mobile backend and provides a simple framework to easily integrate your backend with your iOS, Android, Web, and React Native frontends.

Cognito - Amazon Cognito lets you add user sign-up/sign-in and access control to your web and mobile apps quickly and easily.

Lambda Function - AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration.

S3 - Amazon S3 is object storage built to store and retrieve any amount of data from anywhere web sites and mobile apps, corporate applications, and data from IoT sensors or devices. It has high durability, and stores data for millions of applications used by market leaders in every industry.

DynamoDB - Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed cloud database and supports both document and key-value store models.

EC2 - Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction.

Lex - Amazon Lex is a service for building conversational interfaces into any application using voice and text.

2.3 Back-end Other Technologies

Spark[5] - Apache Spark is a unified analytics engine for large-scale data processing. It achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.

Docker[6] - Docker containers wrap up software and its dependencies into a standardized unit for software development that includes everything it needs to run: code, runtime, system tools and libraries. This guarantees that your application will always run the same and makes collaboration as simple as sharing a container image.

Kafka[7] - Kafka is used for building real-time data pipelines and streaming apps. It is horizontally scalable, fault-tolerant, wicked fast

3. ARCHITECTURE

3.1 Basic Structure

In this section, we are going to tell more about our application's basic structure, the basic structure will be shown in *Fig. 2*.

From the basic structure, you can see what our web application's structure looks like.

The user will access the Columbia Forum which held by AWS S3, since our forum is a single page web application, we also do some modifications on AWS Cloud Front to handle with the error page routing to deal with the SPA redirect problem.

After the user accesses to the front-end, the user will be routed via API Gateway into three different sections, the News Feed, the forum and the Job Hunting part. The whole application will be authenticated by AWS Cognito and AWS Amplify, we will focus all these parts in a deeper in the following parts.

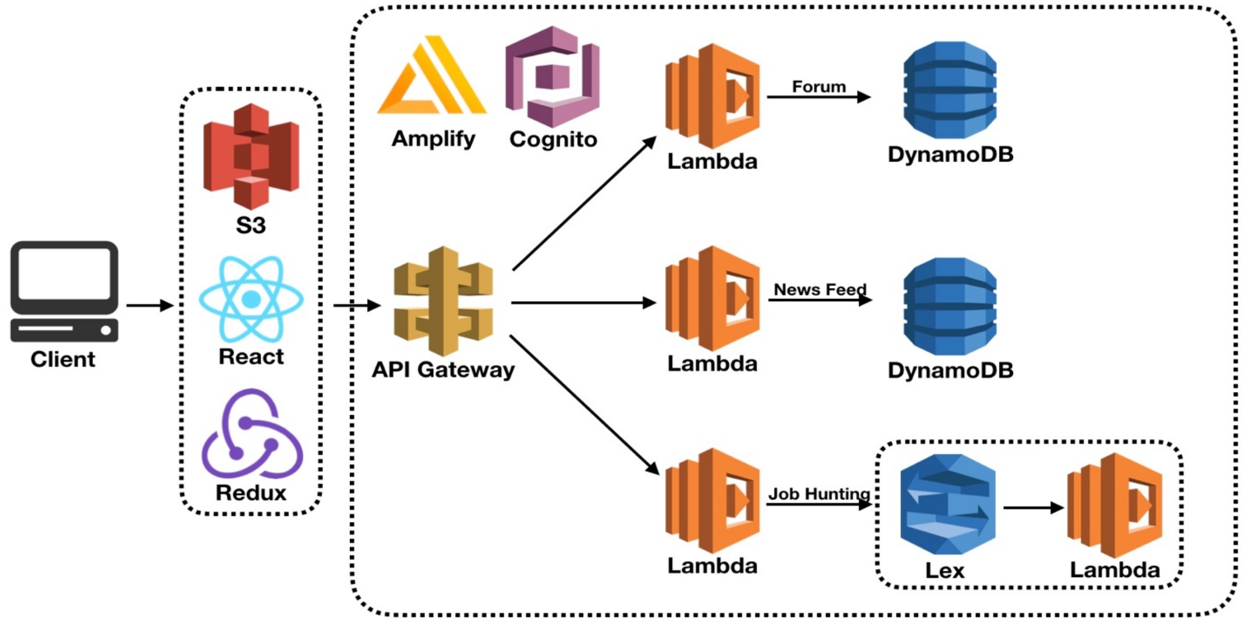


Fig 2. Basic Structure

3.2 Authentication

After the user access the login/sign up page, the front-end will connect with the Amplify for authentication. There are mainly two parts of this part:

If the user is a new user, then the front-end will send all the user's request to the Amplify, Amplify will call the Cognito. Then the Cognito will save new user's information into the User's Pool. Meanwhile, it will send an email to the user for authentication. After the user verified the email address, the Cognito will activate the trigger, and the trigger will record the user's information into a DynamoDB for further use. The user will become a verified user after this whole process.

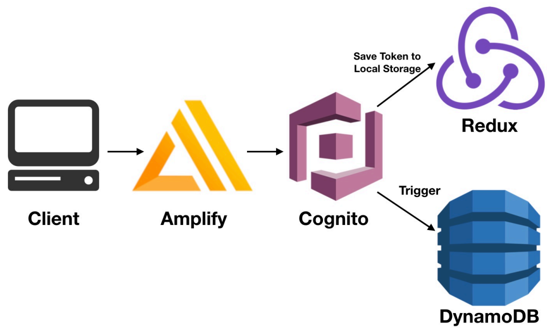
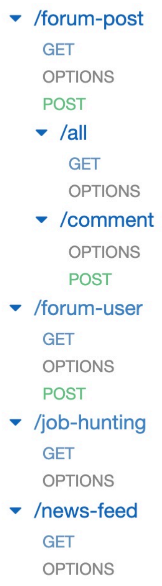


Fig 3. Authentication Structure

If the user is a verified user, the Cognito will send a token from its Identity pool which will allow the user to access all the other services provided by the Columbia Forum. And after the front-end get the token, it will save the user's token into local storage via Redux, which will provide a much faster way on verifying current user.

3.3 API Gateway

The API Gateway is really a clear one, you can see there are mainly four resources: forum-user, forum-post, news-feed, job-hunting, which are just as their names mean, three for three sections, and one for user update their profile.



```
▾ /forum-post
    GET
    OPTIONS
    POST
  ▾ /all
      GET
      OPTIONS
  ▾ /comment
      OPTIONS
      POST
▾ /forum-user
    GET
    OPTIONS
    POST
▾ /job-hunting
    GET
    OPTIONS
▾ /news-feed
    GET
    OPTIONS
```

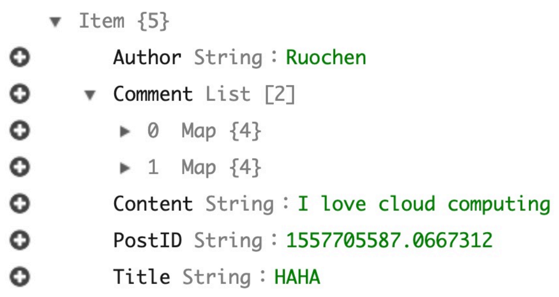
Fig 4. API Gateway

For the forum-post part, it has a GET method, it will help the user get a specific post with the PostID, the post method will let the user send their post to the forum; it also has two child resources, one is for the user to get all the posts, one is for the user to post their replies to a post.
For the forum-user part, the GET method is for getting user information and the POST method is for the user update his or her profile.

For the job-hunting and news-feed parts, they are just getting the information from the back-end and return them to the clients.

3.4 Daily Life (Forum) Part

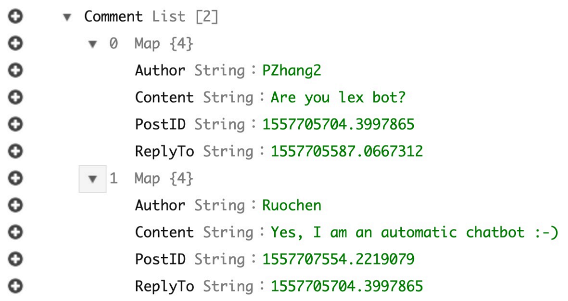
The main structure of this part is relatively simple, all the interaction will happen in Lambda and DynamoDB. And the connection between them is based on boto3 library. The table will be shown in the below figure.



```
▼ Item {5}
    Author String : Ruochen
  ▼ Comment List [2]
      ▶ 0  Map {4}
      ▶ 1  Map {4}
    Content String : I love cloud computing
    PostID String : 1557705587.0667312
    Title String : HAHA
```

Fig 5. Post Structure

From the figure, we can see the primary key of the item is the PostID, which will identify this specific post, and the Author, Content, Title are also some key information of this post. Additionally, there is an attribute called Comment which means the comments of this post, it is a list, and the detailed structure of this part will be shown as below:



```
▼ Comment List [2]
  ▼ 0  Map {4}
      Author String : PZhang2
      Content String : Are you lex bot?
      PostID String : 1557705704.3997865
      ReplyTo String : 1557705587.0667312
  ▼ 1  Map {4}
      Author String : Ruochen
      Content String : Yes, I am an automatic chatbot :-)
      PostID String : 1557707554.2219079
      ReplyTo String : 1557705704.3997865
```

Fig 6. Comment Structure

Take a look at single comment, it contains Author, Content, PostID, ReplyTo attributes, the ReplyTo attribute means the comment or the post the user is replying to, by using this attribute, we can get a clear connection of the relations between all these stuffs.

And all this part is simply implemented in a single lambda function, by detecting different parameters in the route, the lambda will get to know what the client is going to do, and then save the corresponding things in the DynamoDB or read the items in DynamoDB.

3.5 Job Hunting Part

The structure is shown in Fig, you can see from the fig that this part is basically based on Lex, which will provide interactions with users. The user will ask certain query to the Lex, and the Lex will collect all the key information of user's query. After getting the key information, a real-time crawler will be activated, and the crawler will crawl all the job information from Indeed.com by keywords. Then it will save the information it crawled to DynamoDB. And the user will read from DynamoDB to get all the data to the user. But a key factor is that the crawler is a real time one, so once one user search on this crawler, the whole DynamoDB will be refreshed to make sure all the data in this section is something up-to-date.

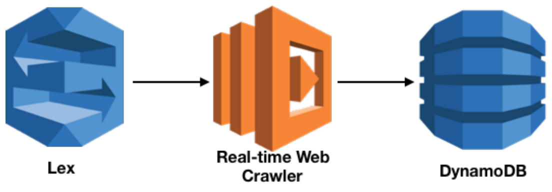


Fig 7. Job Hunting Structure

3.6 News Feed Part

The news feed part is mainly based on Big Data technologies, the architecture will be shown in the following figure.

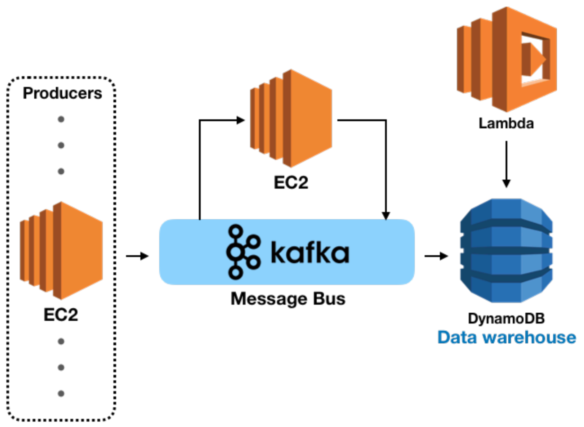
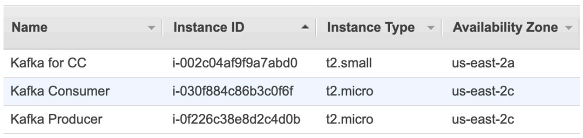


Fig 8. News Feed Structure

Our group came up with this architecture from the Facebook paper "*Realtime Data Processing at Facebook*" [8]. From the architecture, we can see the basic part is a message broker. It is built on an EC2 and we set up a zookeeper and a Kafka as the message bus. On the left side of the fig, we can see there are several producers are writing to the Kafka with a same topic.

And here we want to discuss about why we choose Kafka as the broker: On one hand, it is open source and an Apache application, which means it has strong compatibility with other Apache things, like Spark. On the other hand, traditional message brokers provide publish options like publish-subscribe and point-to-point mode, while Kafka provide us topic mode. Also, traditional message brokers mainly support non-persistent messaging, it means no message-replay support, which is also not what we want. Thus, we decide to use Kafka as our broker.



Name	Instance ID	Instance Type	Availability Zone
Kafka for CC	i-002c04af9f9a7abd0	t2.small	us-east-2a
Kafka Consumer	i-030f884c86b3c0f6f	t2.micro	us-east-2c
Kafka Producer	i-0f226c38e8d2c4d0b	t2.micro	us-east-2c

Fig 9. EC2 instances

In this part, we use on EC2 running two producer processes to emulate this part. All these EC2 will scrape raw news (title, author, time, content) and write them to a same topic (suppose it is news). The news came from six resources: CNN, NYtimes, Google News, Medium, USAtoday and the Guardian.

Above the message bus, you can see another EC2 is reading data from Kafka, and write it back to the Kafka. This consumer will do the job of analyzing key words and summary of the news. Getting these parts is based on a powerful python NLP library nltk[10], it will help us getting all these keywords. After getting the keywords, this consumer will do another job on pySpark, which will help us do a word count action to get the hottest news topic in this time period. After all these works done, it will write back to Kafka with another topic, and all the information will save to data warehouse via Kafka, we use DynamoDB as the data warehouse this time. And the users can read from DynamoDB get the current hottest topics right now.

Take a further look into the EC2 (the producer and the consumer). We set up a docker to make sure our application can be transferrable. Inside the docker, we installed a Jupyter Notebook and pySpark with JVM 1.8, also kafka-python to link the consumer/producer with the Kafka. [9]

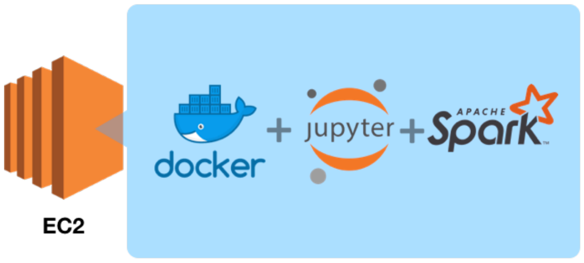


Fig 10. Inside of EC2

Since all the data will be finally saved in the data warehouse, what the user needs to do is just a GET method can he or she see the results of this part.

3.7 Front-end Part

The front-end as said before, is totally build with React 4.0 with React-router. It is a single-page application, it loads a single HTML page and dynamically update that page as the user interacts with the app. Thus, it has a really fast responsive and also it has a good compatible with the local storage and Redux.

On the UI part, we used Bootstrap (to specify, React-Bootstrap) as the main theme of our application, thus it has a really good-looking theme for the users.

4. IMPLEMENTATION

The Columbia Forum website follows the general logistic when you first use a website. In the subsections, we list all of the functions of our web application and we introduce how to use this website.

4.1 Sign Up

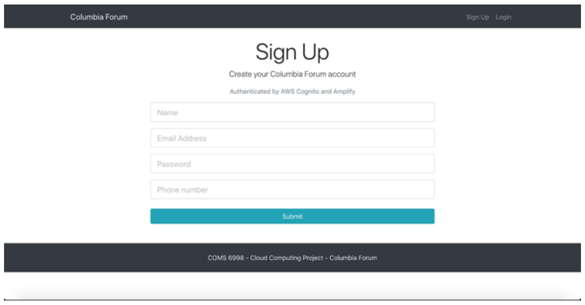

Columbia Forum
Sign Up
Login
Sign Up
Create your Columbia Forum account
Authenticated by AWS Cognito and Amplify
Name
Email Address
Password
Phone number
Submit
COMS 6998 - Cloud Computing Project - Columbia Forum

Fig 11. Sign up page

After opening our website, this is the sign up page for you. If you don't have a user account, you will need to register an account first. You should input your name, email, password and phone number. The phone number should be in the format of +1xxxxxxxxxx. After you finish the sign up process, you will be automatically directed to the Log In page.

4.2 Log In

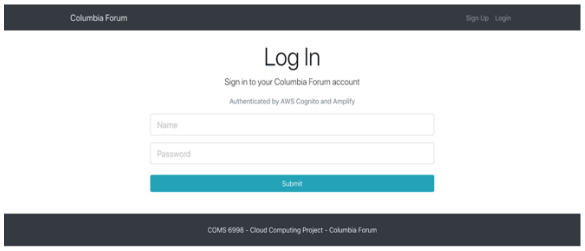

Columbia Forum
Sign Up
Login
Log In
Sign in to your Columbia Forum account
Authenticated by AWS Cognito and Amplify
Name
Password
Submit
COMS 6998 - Cloud Computing Project - Columbia Forum

Fig 12. Log in page

This is the log in page, here you should input your name and password you have used for the sign in part. After you logged in, you will be directed to the dashboard.

4.3 Dashboard

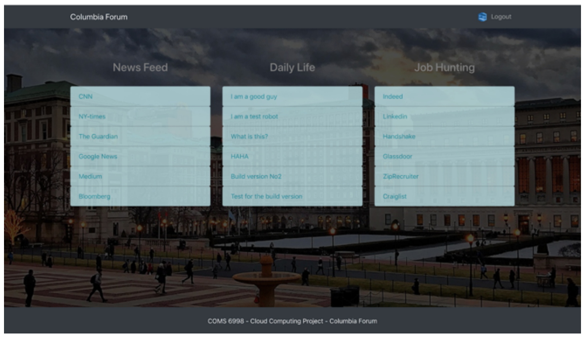


Fig 13. Dashboard page

At the dashboard, there are three sections on this forum, and you can go to the specific part by clicking each title. The left part is the News Feed part where you can get various kinds of real-time news. The middle part is a forum where registered students are able to post and reply to other posts. The right part is the job hunting, students can input job title and location they want to work for, and then they can get job lists from Indeed website based on their requests. At the top columns, there is Logout button where you can exit your current account and your profile picture. If you click on the profile picture, it will direct you to your own profile.

4.4 Profile Page

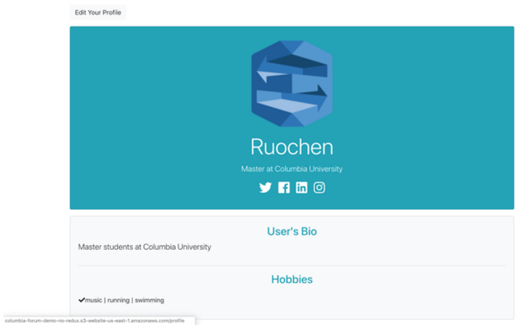
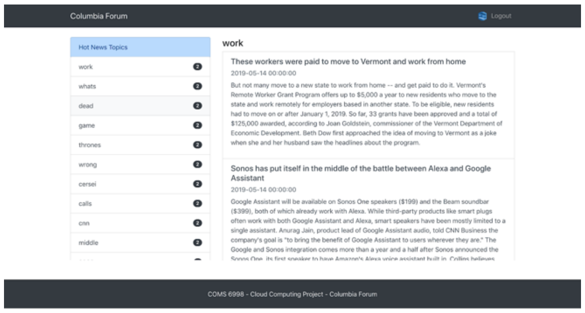

Edit Your Profile
Ruochen
Master at Columbia University
User's Bio
Master students at Columbia University
Hobbies
✓music | running | swimming
columbia-forum-demo-no-redux.s3-website-us-east-1.amazonaws.com/profile

Fig 14. Profile page

In your own profile page, you can edit your own profile including adding your education information, showing the current friends you have on the forum and your hobbies and bios. Also, you can change your user images and add your social links

4.5 News Feed


Columbia Forum
Logout

Hot News Topics

work	2
whats	2
dead	2
game	2
thrones	2
wrong	2
cersei	2
calls	2
cnn	2
middle	2

work

These workers were paid to move to Vermont and work from home
2019-05-14 00:00:00
But not many move to a new state to work from home -- and get paid to do it. Vermont's Remote Worker Grant Program offers up to $5,000 a year to new residents who move to the state and work remotely for employers based in another state. To be eligible, new residents had to move on or after January 1, 2019. So far, 33 grants have been approved and a total of $125,000 awarded, according to Joan Goldstein, commissioner of the Vermont Department of Economic Development. Beth Dow first approached the idea of moving to Vermont as a joke when she and her husband saw the headlines about the program.

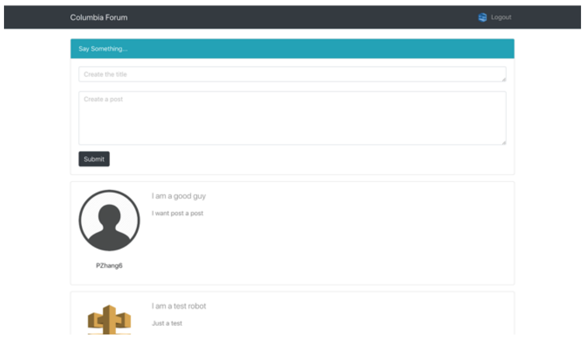
Sonos has put itself in the middle of the battle between Alexa and Google Assistant
2019-05-14 00:00:00
Google Assistant will be available on Sonos One speakers ($199) and the Beam soundbar ($399), both of which already work with Alexa. While third-party products like smart plugs often work with both Google Assistant and Alexa, smart speakers have been mostly limited to a single assistant. Anurag Jain, product lead of Google Assistant audio, told CNN Business the company's goal is "to bring the benefit of Google Assistant to users wherever they are." The Google and Sonos integration comes more than a year and a half after Sonos announced the Sonos One, its first speaker to have Amazon's Alexa voice assistant built in. Collins believes

COMS 6998 - Cloud Computing Project - Columbia Forum

Fig 15. News Feed page

In the News Feed section, you can see a lot of real-time news from various kinds of websites. The left part is the hot topics about those news, and right part is the news with details. By clicking the title of these news, you will be directed to the original page of this news where you can see more detail information about it.

4.6 Daily Life

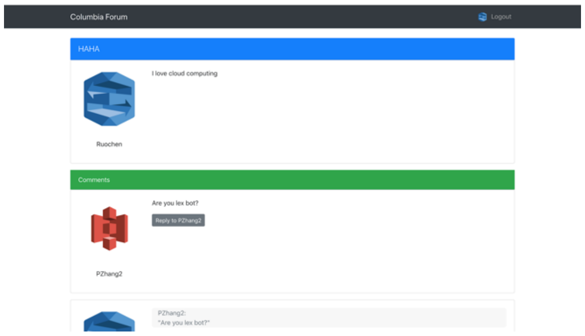

Columbia Forum
Logout

Say Something...
Create the title
Create a post
Submit

I am a good guy
I want post a post
PZhang6

I am a test robot
Just a test

Fig 16. Daily Life page 1


Columbia Forum
Logout

HAHA
I love cloud computing
Ruochen

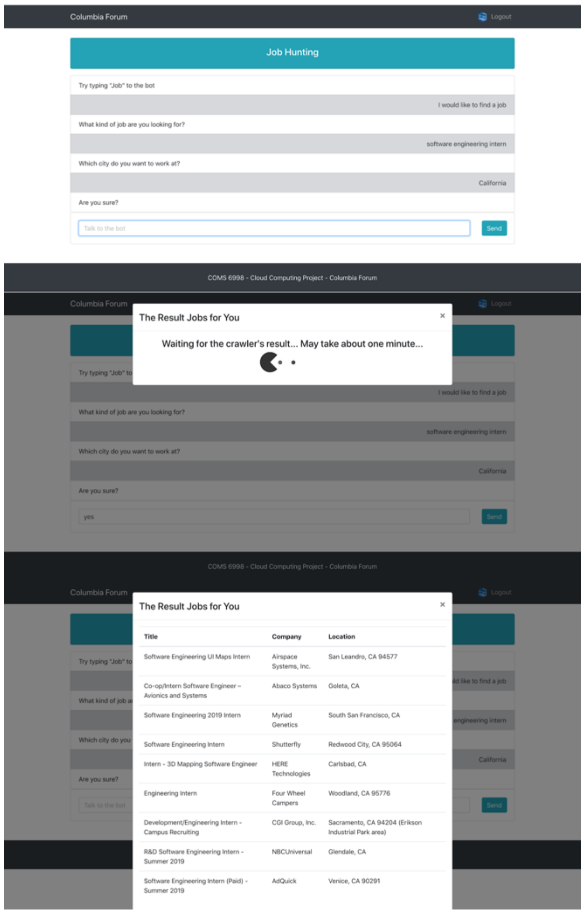
Comments
Are you lex bot?
Reply to PZhang2
PZhang2

PZhang2:
"Are you lex bot?"

Fig 17. Daily Life page 2

By clicking "Daily Life" in the dashboard, you will be directed to the Daily Life section where you can see posts and comments from other students, make comments and make a post on your own.

4.7 Job Hunting


Columbia Forum
Logout

Job Hunting

Try typing "Job" to the bot
I would like to find a job
What kind of job are you looking for?
software engineering intern
Which city do you want to work at?
California
Are you sure?
Talk to the bot
Send

COMS 6998 - Cloud Computing Project - Columbia Forum

The Result Jobs for You

Waiting for the crawler's result... May take about one minute...

Are you sure?
yes
Send

COMS 6998 - Cloud Computing Project - Columbia Forum

The Result Jobs for You

Title	Company	Location
Software Engineering UI Maps Intern	Airspace Systems, Inc.	San Leandro, CA 94577
Co-op/Intern Software Engineer – Avionics and Systems	Abaco Systems	Goleta, CA
Software Engineering 2019 Intern	Myriad Genetics	South San Francisco, CA
Software Engineering Intern	Shutterfly	Redwood City, CA 95064
Intern - 3D Mapping Software Engineer	HERE Technologies	Carlsbad, CA
Engineering Intern	Four Wheel Campers	Woodland, CA 95776
Development/Engineering Intern - Campus Recruiting	CGI Group, Inc.	Sacramento, CA 94204 (Erikson Industrial Park area)
R&D Software Engineering Intern - Summer 2019	NBCUniversal	Glendale, CA
Software Engineering Intern (Paid) - Summer 2019	AdQuick	Venice, CA 90291

Fig 18. Job Hunting Page

In the Job Hunting section, you can input the job title and location you want for working. And then you can get job lists from Indeed website containing job links, job descriptions and other information.

5. CONCLUSION

The whole project let us learned a lot on Cloud Computing and micro services. With the techniques we learned from the class, all of our team members think these techniques are a must for every developer. By using Amazon Web Services, we can easily build up serverless application, which is really friendly for those start-ups. On the other hand, besides learned a lot techniques, we also know that team work is really important for a not to group projects. Reasonable division of labor and cooperation is the guarantee of efficiency.

To talk about future works, we want to focus more on the interaction with users in our forum. We may set up the Job Hunting Part and News Feed part also as some where people can share things with others. Similarly, not only these three parts, we may set a clearer classification on different parts of the forum. Meanwhile, a chat room should also be a good choice to implement. Furthermore, we also want to build mobile applications for users.

Finally, thanks for Prof. Sambit's Cloud Computing and Big Data course make us know something state of the art in the industry. And also, thanks for TAs (Robby, Aditya, Pankil)' help on assignments and project.

6. REFERENCES

1. https://thesocialmediamonthly.com/the-importance-of-online-forums/
2. https://reactjs.org/
3. https://redux.js.org/
4. https://docs.aws.amazon.com/
5. Zaharia, Matei, et al. "Spark: Cluster computing with working sets." HotCloud 10.10-10 (2010): 95.
6. https://docs.docker.com/
7. Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." Proceedings of the NetDB. 2011
8. Chen, Guoqiang Jerry, et al. "Realtime data processing at Facebook." Proceedings of the 2016 International Conference on Management of Data. ACM, 2016.
9. https://hub.docker.com/r/jupyter/pyspark-notebook
10. https://www.nltk.org/

7. APPENDIX

Demo link:
http://columbia-forum-demo-no-redux.s3-website.us-east-1.amazonaws.com/

Github repo link:
https://github.com/zhangph1220/Columbia-Forum

Youtube video link:
https://youtu.be/IbzOZIZyzRM

Slides link:
https://drive.google.com/open?id=18HRcw2wvYLU4WmILiLT3D-MBb31iRf-0