Newsec
Recommend Interesting News for You in a Second

Weicheng Zhao
MSCS
wz2578@columbia.edu

Hu Zheng
MSCS
hz2709@columbia.edu

Mingjun Wang
MSCS
mw3542@columbia.edu

Di Wu
MSCS
dw3013@columbia.edu

***Abstract*—To meet the need for people nowadays to quickly browse news on their mobile devices, we developed an AWS base news reading and recommending application – Newsec. Multiple Amazon services are applied in every stage of the application, including user management, API management, data processing and storage, and model training, etc. Presently, 31,102 records of news from thousands of sources has been crawled by our Newsec periodically. These records are saved, processed and categorized together with DynamoDB, OpenSearch and Amazon Comprehend. User interaction interface of the application is designed in Typescript with React framework. To be mentioned, our application recommends news to users according to interest analysis by user preference and events with Amazon Personalize.**

***Index Terms*—news abstract, news recommendation, AWS development**

I. Introduction

Due to the fast living pace nowadays, people tend to access information from the Internet with their mobile devices during small and fragmented moments. In this case, reading news articles that are detailed and several pages long may be tiring and stressful, especially when reading news that the person is not so interested in. They can scarcely keep focused on tedious long news articles. Neither they can comprehend the ideas merely with the news titles.

As a result, there is a need for an AWS service making it easier and more efficient for people to understand ideas of news through title, summary and important details retrieved from articles. To meet the need, we come up with our Newsec – an application not only help people to save their time but also keep themselves up-to-date with the important and personalized news around the world. Our Newsec application automatically and periodically crawls global and domestic news data with title, summary and detailed information retrieved from news media APIs. In this way, our users can keep up with the latest information around the world. Then the crawled news records are categorized by Amazon Comprehend and processed by Amazon Lambda functions defined by us before being stored into the database. For users, they can either browse certain categories they are interested in, or choose to view recommended news and topics according to their preferences. To collect personal preferences, users submit ”Like” or ”Dislike” attitudes toward news by clicking buttons when browsing within our application. There are choices to view related news or topics to your preferences. Then the preference information will be put into training models of Amazon Personalize for recommendation. Our web page also has the original source link for each piece of news. Users can follow the link when they would like to read that news in detail.

II. Related Work

There are many related works on Github. Many AWS based projects works on news publishing, reading and summarizing. For example, Awswhatsnew[1] by user adilosa published AWS News to Twitter by reading the official RSS feed with AWS Lambda. It simply get news data and post them to Twitter with its API. JustComment[2] is another AWS application launched in this year. It enables writing newsletters in Markdown and send to many recipients from your machine using AWS SES. However, the news information is sent to users by email rather than a user-interaction interface. It neither send specific news that users are interested in.

When it comes to the media format of news, there is another official AWS example Amazon Transcribe News Media Analysis[3]. It allows creating transcriptions of live streaming video using AWS Transcribe. It includes a Web UI where users can submit URLs of videos to obtain an ECS task per URL running in Fargate to begin the transcription. Text can be shown to users in real time with the video by clicking on the link provided by the UI. This solution is quite special because it works on news of different media type to traditional text contents.

Works of news recommendation on Github scarcely cooperate with a cloud plateform like AWS. They are more focused on algorithms for news, like clustering and user-based collaborative filtering. There is an example of AWS news recommendation by user XinxinTang[4]. It continuously monitored news source based on News APIs and check repeatness based on encoded title-Digest. News records are fetched non-repeatedly with Kafka into database. Then the data is put into offline training with Tensorflow and served online before being displayed on the front end.

Our work do further on the AWS news recommendation job, and the whole application is serverlessly deployed on cloud

[1]https://github.com/adilosa/awswhatsnew
[2]https://github.com/JustComments/newsletter-cli
[3]https://github.com/aws-samples/amazon-transcribe-news-media-analysis
[4]https://github.com/XinxinTang/News_Recommendation_System-AWS

platform. News crawling are triggered periodically by Amazon EventBridge. We applied Amazon Comprehend which not only works on news of text content but also can analyze information in the media type of video. Meanwhile, the recommendation part also works with AWS Personalize service for recommendation based on keywords or news content. News information and user interaction are displayed on our self-designed front end pages.

III. Architecture

A. Application Architecture

Our application is entirely designed and deployed serverlessly on AWS platform. Main features of project architecture drawn by Lucidchart[5] is shown in Fig. 1. We apply CloudFront to speed up distribution of static and dynamic web content, like our .html, .css, .js, and image files, to users. For user authentication and authorization, we apply Amazon Cognito for sign up and login management. Users have to login to save their preferences by clicking like or dislike buttons when browsing the news. Front-end pages are stored in an S3 bucket B2. We designed our APIs as RESTful API managed by API Gateway. There are three Lambda functions for the interaction between users and news:

1) Get User Event obtains all events related to a user and a piece of news while checking repetition for news shown to a user.
2) Write User Event saves user interaction of pressing LIKE/DISLIKE buttons. News information with key words, catogories, and content is passed in to this Lambda function. Output stream of this function sent user events into Event table of DynamoDB and model of Amazon Personalize.
3) Get News Recommendation returns latest news with recommended keywords by Amazon ElasticSearch.

For the new processing part, there's a Lambda function Crawl News from New API. This API is triggered periodically by EventBridge to detect entities and invoke classification with Amazon Comprehend, and save news into News table and part of Comprehend job information into Jobs table in DynamoDB. Classification results will be stored in another S3 bucket B1. Finally, the Lambda function Index Classified News save another part of Comprehend job information into Jobs table in DynamoDB. News category information will also be sent to News table by the function.

B. User Interaction Architecture

For the front end design of our application, we have only one main page for users to conduct their activities on. User case flowchart is shown in Fig. 2. At the very beginning where there is no user login, the relative routing path should be "/category". Users can browse news by clicking NEXT button or TELL_ME_MORE button on the page. But no preference data will be saved. Neither news on the page is recommended according to user data.

[5]https://lucid.app

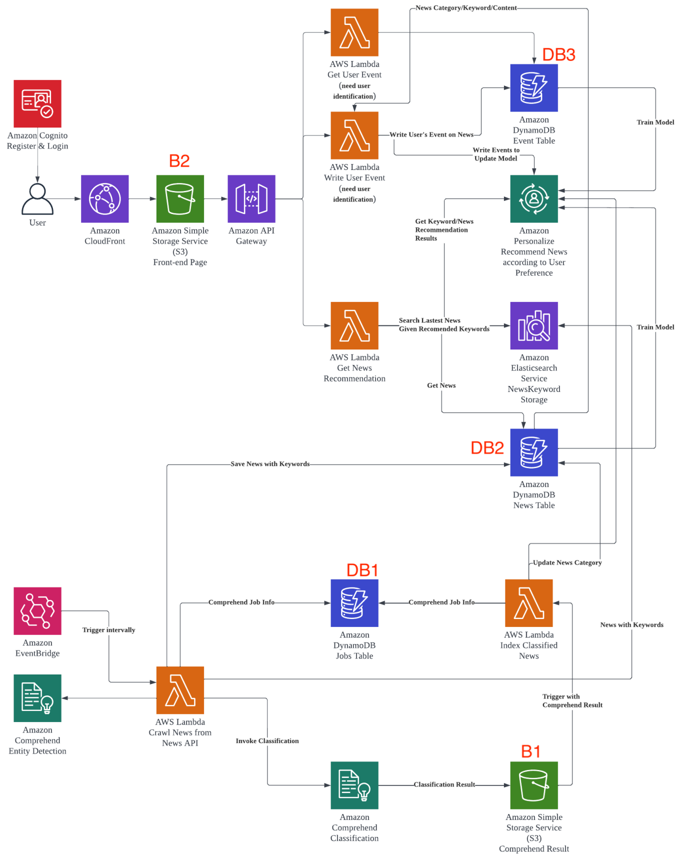

Amazon Cognito Register & Login
User
Amazon CloudFront
B2
Amazon Simple Storage Service (S3) Front-end Page
Amazon API Gateway
AWS Lambda Get User Event (need user identification)
AWS Lambda Write User Event (need user identification)
AWS Lambda Get News Recommendation
News Category/Keyword/Content
DB3
Amazon DynamoDB Event Table
Write User's Event on News
Write Events to Update Model
Train Model
Amazon Personalize Recommend News according to User Preference
Get Keyword/News Recommendation Results
Search Lastest News Given Recomended Keywords
Amazon Elasticsearch Service NewsKeyword Storage
Train Model
Get News
DB2
Amazon DynamoDB News Table
Save News with Keywords
Update News Category
DB1
Comprehend Job Info
Amazon DynamoDB Jobs Table
Comprehend Job Info
AWS Lambda Index Classified News
News with Keywords
Amazon EventBridge
Trigger intervally
AWS Lambda Crawl News from News API
Amazon Comprehend Entity Detection
Trigger with Comprehend Result
Invoke Classification
B1
Amazon Comprehend Classification
Classification Result
Amazon Simple Storage Service (S3) Comprehend Result

Fig. 1: Architecture of Newsec

If a user wants to save his/her preference by clicking LIKE or DISLIKE button, the user have to sign up or login. Both actions require an email address and a password. A user has to confirm the password by entering it twice when signing up. After successfully signing up or login, the user should be on the recommendation page, whose relative routing path is "/recommended". News that is recommended base on contents agreeing with personal interests of the user will be shown on this page.

The user can also browse news by keywords recommended by interest on "/focused" and news under different category on "/category/specific_category_name". Here, "specific_category_name" refers to a certain category in the categories bar that is chosen by the user. Finally, there is also an option to sign out. Once the SIGN_OUT is clicked, the user will not be in the login status. And the page will go back to "/category".

IV. Screenshots

Figure. 2 shows the homepage of Newsec when we don't log in. Figure. 8 shows the homepage of Newsec when logged in. The news title, abstract and news image is shown on the page, where users can get an overview. Keywords are arranged under the title. Users can viewing more news by clicking the "NEXT" button, or jumping to external websites

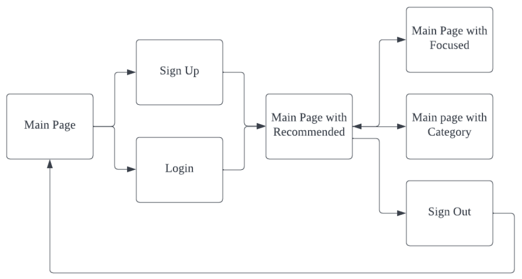
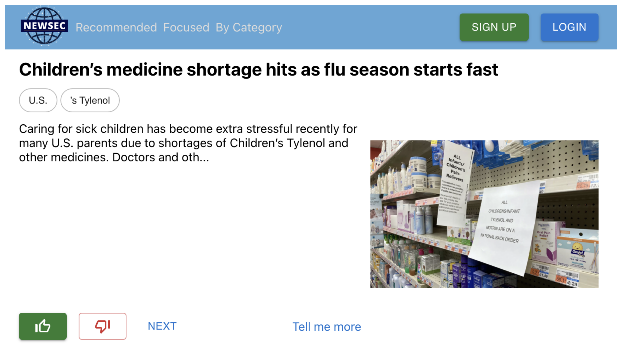


Fig. 2: User Case Flowchart

that containing detailed news articles by clicking the ”Tell me more” button.

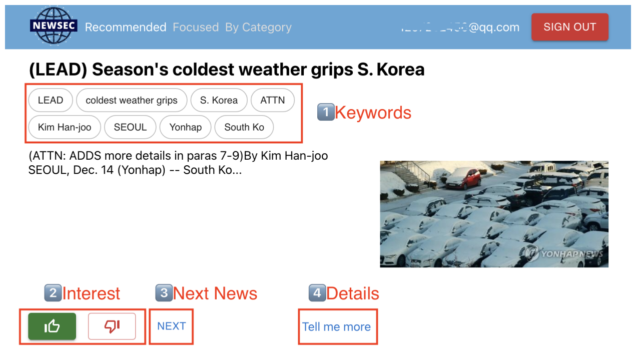
The ”By Category” button in the navigation bar provides users with ”viewing news by category” function, where a drop-down menu will show all available news categories, as shown in Figure. 5. If users choose one option, e.g. ”Tech”, Newsec will only deliver Technical news to them.



Children's medicine shortage hits as flu season starts fast

Caring for sick children has become extra stressful recently for many U.S. parents due to shortages of Children's Tylenol and other medicines. Doctors and oth...

Fig. 3: Homepage (Guests Page)



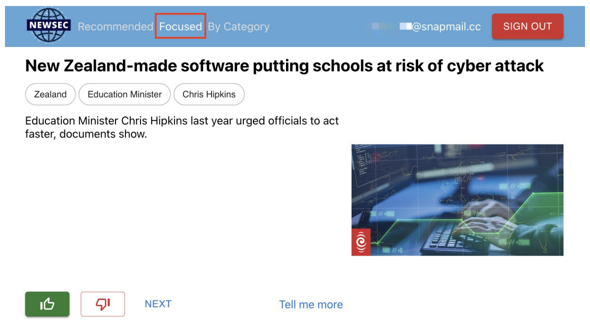
(LEAD) Season's coldest weather grips S. Korea

(ATTN: ADDS more details in paras 7-9)By Kim Han-joo SEOUL, Dec. 14 (Yonhap) -- South Ko...

Fig. 4: ”Recommended” Homepage(Logged In)

”Interested / Uninterested” buttons are located on the bottom left side of the homepage. However, Login box(Figure. 8) will pop up if we clicked them under Guest status. We can switch between sign up(Figure. 7) and login depending on if we are new users or not.

Once we logged in, we will see news recommending page Figure. 4 by default. And ”Interested / Uninterested” buttons



New Zealand-made software putting schools at risk of cyber attack

Education Minister Chris Hipkins last year urged officials to act faster, documents show.

Fig. 5: ”Focused” Homepage(Logged In)

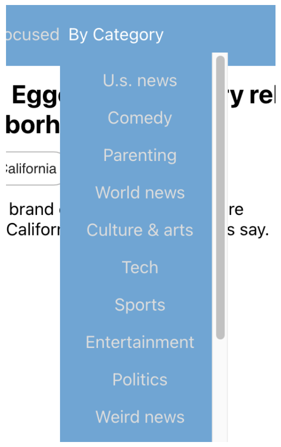


Fig. 6: Browse News by Category

will take effect. Users are only allowed to push either ”Interested” (thumbs up button) or ”Not Interested” (thumbs down button). Both button will turn grey when users press either of them (Figure. 6) Newsec will learn users preference according to all user activities, and trying to deliver more news that users are interested in.

Newsec designs two kinds of news recommending: ”Recommended” button will render news whose content is corresponding to users' preference, Figure. 4; ”Focused” button will render news with keywords that users liked, Figure. 5.

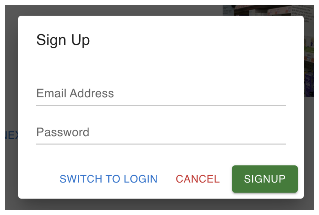


Fig. 7: Sign Up Interface

V. Implementation

We build Newsec using AWS service. We implement the compute service of Newsec in Python 3.9. We use React as the UI framework. S3, DynamoDB, and OpenSearch are our

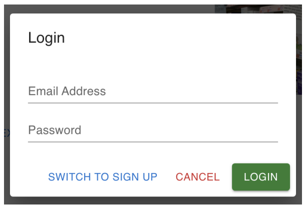
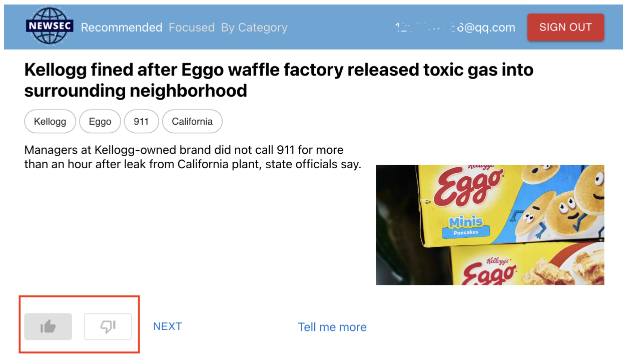


Fig. 8: Login Interface



Kellogg fined after Eggo waffle factory released toxic gas into surrounding neighborhood

Managers at Kellogg-owned brand did not call 911 for more than an hour after leak from California plant, state officials say.

Fig. 9: Liked or Disliked News

data storage. And our application has two main components: News Crawler and News Recommendation.

TABLE I: Jobs Table Schema (DB1)

Attribute	Data Type	Key
job_name	String	partition key
uuids	array[String]	

TABLE II: News Table Schema (DB2)

Attribute	Data Type	Key
uuid	String	partition key
category	String	partition key
content	String	
image_url	String	
keywords	array[String]	
published_at	Integer	Sort key
title	String	
url	String	

A. News Crawler

News Crawler periodically retrieves global news from thousands of sources, and save news content and its corresponding classification and keywords into DynamoDB and OpenSearch, which will be used as data sources by News Recommendation. Approximately over 1 million articles will be retrieved every week.

We get news from https://api.thenewsapi.com/v1/news/top every 30 minutes. This periodical event is triggered by EventBridge scheduler.

TABLE III: Event Table Schema (DB3)

Attribute	Data Type	Key
uuid	String	partition key
event	String	
news_id	String	
timestamp	String	
user_id	String	

Once a piece of news is retrieved, we process them by Amazon Comprehend[6]. To extract the news keywords, we use real-time entity recognition API ”detect_entities”, and save the extracted keywords in OpenSearch and DynamoDB news table (DB2), whose schema is shown in Table. II; to classify the news categorization, we launch a classifier(Comprehend job) that trained by a custom classifier using News Category Dataset [1], which have 210k news with 42 categories.

Since we are making asynchronous classification, we output Comprehend results in an S3 bucket (B1) and store comprehend job information in DynamoDB jobs table (DB1). Its schema is shown in Table. I. We set up a PUT event trigger on the S3 bucket (B1), such that whenever comprehend job output new classification results in the B1, it triggers another Lambda Function. The lambda function will index the results and update the news categories in DynamoDB news table (DB2) and Amazon Personalize[7], which we will introduce more details in News Recommendation section.

B. News Recommendation

Based on the resource we collected from news crawler, we construct a news browsing platform and present personalized experience to users by developing a recommender system that predicts individual preferences.

The front-end code of Newsec is saved in a public S3 bucket (B2), and distributed by CloudFront for faster delivery.

Using Amazon Personalize service, we implement two kinds of recommendation: keywords recommendation and news recommendation. Each recommender system has custom training dataset groups. We collected data from the beta-testing stage. For each dataset, we generate 1000 interaction events for initial training. Table. V demonstrates the dataset schema. EVENT_TYPE will be selected from one of the following values: LIKE/DISLIKE/DETAIL/VIEW. Keywords-Recommend collects all the interaction between users and keywords, so the field ITEM_ID being filled by keyword instead. News-recommend collects all interaction between users and news, therefore field ITEM_ID is news id. And we build an item dataset with schema as Table. **??** to store the news content and news category. This item dataset will also participate in recommending training, providing multiple attributes to item and contributing to more accurate recommendation.

To create a solution, we use USER_PERSONALIZATION as our recipe type. For solution configuration, we are setting ”event type” to be ”LIKE” when news recommendation (i.e.

[6]https://docs.aws.amazon.com/comprehend/index.html
[7]https://docs.aws.amazon.com/personalize/index.html

specify ”LIKE” event to use in training). However, we set ”event type” to be ”DETAIL” when keywords recommendation because we intend to dig out the keywords that users mostly interested and even trying to read more information.

In order to make real-time recommendations, we deploy a solution by creating a Personalize campaign. While calling campaign from lambda function ”get-news-list”, we append a filter that excludes all disliked news if using keyword recommendation; or we use a filter that exclude all viewed news if using news recommendation.

TABLE IV: Personalize User-item Interaction Dataset Schema

Attribute	**Data Type**
USER_ID	String
ITEM_ID	String
TIMESTAMP	String
EVENT_TYPE	String

TABLE V: Personalize Item Dataset Schema

Attribute	**Data Type**
ITEM_ID	String
CREATION_TIMESTAMP	Long
CATEGORY	String
CONTENT	String

When a new interact event comes, lambda function ”write-user-event” will save events both in keywords recommendation and news recommendation. Recommender systems will update the solution based on newly added events. In order to retrieve user events, we implement a DynamoDB (DB3) to store historical user events. Its schema is shown in Table. III

Since we are mainly providing two kinds of news recommendations, we add different fields in in client query that is sent to lambda function ”get-news-list” to figure out which recommendation are requested. If a query have ”list” entry, it means the user didn’t request recommendation. If a query have ”recommend_by_keyword” entry, it means the user request keyword recommendation. If a query have ”recommend” entry, it means the user request news recommendation. This feature is for test. So it doesn’t appear in our main project structure.

C. User Identity

We use Amazon Cognito for user authentication and authorization. Users have to sign up or login before clicking LIKE/DISLIKE buttons to save their preference. Once a user has successfully login, events of that user sent to back-end will be with a USER_ID in the request body. We create a Cognito user pool to acommondate all registered users. We implement the sign up and login function with APIs provided by amazon-cognito-identity-js[8] package. Once the user press buttons on our front-page to sign up or login, the asynchronous functions will be called. Usually, there is a default confirmation of account by email or phone number, which improves the security of account and can be used for account recovering. Users have to enter the confirmation code when signing up an account. To make our account managing process more convenient, we used another Lambda function to ”Pre sign-up”. The function hard code the status of accounts and emails to be confirmed.

[8]https://github.com/aws-amplify/amplify-js

VI. Code Structure

We set up 7 lambda functions throughout Newsec. Two of them are in the news crawler component: ”crawl-news”, ”index-news”; the remaining four are in new recommendation part: ”get-event”, ”get-news-list”, ”write-user-event”, ”user-pool-auto-verify” and ”get-news”. Due to limited space of article, we only describe general workflow for each function by pseudo-code.

The logic of ”crawl-news” is shown as following:

```
# retrieve news from news API
news = http.request_encode_url("GET", TOP_NEWS_API,
    require_fields)

# save all raw news title and content into S3 for
    backup.
string_to_upload = news_title + news_content
s3_put_response = s3_client.put_object(
            Body=string_to_upload,
            Bucket=INPUT_S3_BUCKET,
            Key="comprehend-news-x"
        )

# Execute news classification job, save
    classification result in S3 bucket.
classification_response = comprehend_client.
    start_document_classification_job(
        InputDataConfig=INPUT_S3_BUCKET,
        OutputDataConfig=B1
    )

# execute news entity detection
entities_response = comprehend_client.
    batch_detect_entities(TextList=string_to_upload)

# save news and corresponding keywords into
    OpenSearch and DynamoDB
r = http.request('PUT',
                 opensearch_url+'/'+news_uuid,
                 headers=headers,
                 body=newsId+keywords
                )

news_table = dynamodb.Table(NEWS_TABLE)
news_table.put_item(Item=newsId+keywords)
```

Any new record in the S3 bucket (B1) will trigger ”index-news”, which will analyze the new record, and update the DynamoDB news table (DB2) as well as news item properties in Amazon Personalize.

```
# Retrieve job data source (news uuids) from
    dynamodb
news_uuids = jobs_table.get_item(job_name)

# Update news category
category_comprehend_results = s3.Object(bucket,
    object_key)
response = news_table.update_item(Key=news_uuid,
                Update='SET category =
    category_comprehend_result',
            )
```

To verify the user identity with Cognito, we use ”user-pool-auto-verify” as a trigger, which is invoked when a user submits their information to sign up, allowing system to perform custom validation to accept the sign up request.

User activities like viewing news or clicking LIKE/DISLIKE/DETAIL button on the web page will generate user events. Front-end sends those events along with ”POST /event” request, which will trigger ”write-user-event”:

```
# generating uuID with user_id, news_id and event_id
uuID = str(event['user_id'] + "-" + event['news_id']
     + "-" + event['event'])
# If events already existed in DynamoDB, we just
    update the timestamp of that event.
if search_event_db(uuID) == True:
    dynamodb.update_item(
        TableName='Event',
        Key={'uuid':uuID},
        UpdateExpression='SET #t = latest_timestamp'
    )
# insert event into dynamoDB event table.
insert_dynamo_table(event, uuID)
# retrieve news keywords and send them as interact
    events into news and keywords personalize
    datasets group.
news_keywords = search_os(event['news_id'])
import_keywords_personalize_interact(event,
    session_id, news_keywords, TIMESTAMP)
import_news_personalize_interact(event, session_id,
    TIMESTAMP)
```

”get-event” provides an endpoint to get all events related to a user and a piece of news. This function can be used to avoid showing same news repeatedly to users.

```
item_response = dynamo_event_table.get_item(Key=uuID
    )
item = item_response['Item']
returned_events.append(item)
```

”get-news-list” is the core function that deliver general or personalized news feed.

```
# If "list" entry in the request, then search and
    return the news without recommending.
if 'list' in request_body:
    # If there are specific "Category" entry in the
    request, then search all the news in DB with
    same category
    # But if no category specified,
    if request_category:
        query_paginator = dynamodb_client.
    get_paginator('query')
        news = query_paginator.paginate(
            TableName=NEWS_TABLE,
            IndexName=NEWS_TABLE_GSI,
            categoryValue=request_category
        ).build_full_result()
    else:
        # retrieve news ID by search generally in
    opensearch, then get news content from DynamoDB
        news_uuids = http.request(
                'POST',
                os_search_url,
                headers=headers,
                body=general_query)
        news = dynamodb_client.batch_get_item(
                RequestItems={'uuids': news_uuids}
            )
# If request have recommend_by_keyword entry, we
    should return personalized news results to users
elif 'recommend_by_keyword' in body:
    # get keyword recommendation from Personalize
    keyword_recommendations = personalize_client.
    get_recommendations(
            recommender =
    keyword_recommendation_CAMPAIGN
            userId=request_user_id
        )
    # search news ID with corresponding keywords in
    opensearch, then get news content from DynamoDB
    news_uuids = http.request(
            'POST',
            os_search_url,
            headers=headers,
            body=keyword_recommend_query)
    news = dynamodb_client.batch_get_item(
            RequestItems={'uuids': news_uuids}
        )
elif "recommend" in body:
    # get news recommendation from Personalize
    keyword_recommendations = personalize_client.
    get_recommendations(
            recommender =
    news_recommendation_CAMPAIGN
            userId = request_user_id
        )
    # search news ID with corresponding keywords in
    opensearch, then get news content from DynamoDB
    news_uuids = http.request(
            'POST',
            os_search_url,
            headers=headers,
            body=keyword_recommend_query)
    news = dynamodb_client.batch_get_item(
            RequestItems={'uuids': news_uuids}
        )
return news
```

”get-news” will be triggered when calling ”GET /news/{newsId}”, which will return complete news information with the corresponding newsId.

```
news_obj = dynamo_news_table.get_item(Key=uuID)
news = news_obj['Item']
return news
```

VII. Links

Newsec website : https://news.asachiri.com/
Opening with mobile phone will provide better user experience.

Presentation slides:
https://docs.google.com/presentation/d/1MXdkDKSmn9PQJ0yIaXFa42YAml_p2RPppQhq5YAm6YE/edit?usp=sharing

Demo Video:
https://youtu.be/Vj5EEG-rqmc

VIII. Conclusion

We developed Newsec, which is a news reading and recommending application. We build the efficient and robust system architecture by implementing 12 kinds of AWS services. Currently, more than 30,000 records of news in total from worldwide news websites has been crawled and saved in our database. They will also be updated in a weekly fashion. For the recommender system, we designed ”KEYWORDS” recommendation and ”NEWS” recommendation that focus on learning users’ preference based on their interests in ”news content" or "news keyword". Both of them shows good accuracy and efficiency. Newsec has been deployed in production stage, and presented outstanding capability of delivering up-to-date and personalized national and international news to users.

References

[1] News Category Dataset, Kaggle, Sept 2022. [Online]. Available: https://www.kaggle.com/datasets/rmisra/news-category-dataset.

[2] Amazon Cognito. Available: https://docs.aws.amazon.com/cognito/index.html

[3] awswhatsnew, adilosa, 2019 Aug 4. Availbale: https://github.com/adilosa/awswhatsnew

[4] JustComments, OrKoN, 2021 Mar 17. Available: https://github.com/JustComments/newsletter-cli

[5] amazon-transcribe-news-media-analysis, amazon-samples, svozza, 2022 Jun 23. Available: https://github.com/aws-samples/amazon-transcribe-news-media-analysis

[6] News_Recommendation_System-AWS, XinxinTang, 2018 Jun 27. Available: https://github.com/XinxinTang/News_Recommendation_System-AWS

[7] amplify-js, aws-amplify, tannerabread, 2022 Dec 17. Available: https://github.com/aws-amplify/amplify-js

[8] Amazon Comprehend Documentation. Available: https://docs.aws.amazon.com/comprehend/index.html

[9] Amazon Personalize Documentation. Available: https://docs.aws.amazon.com/personalize/index.html