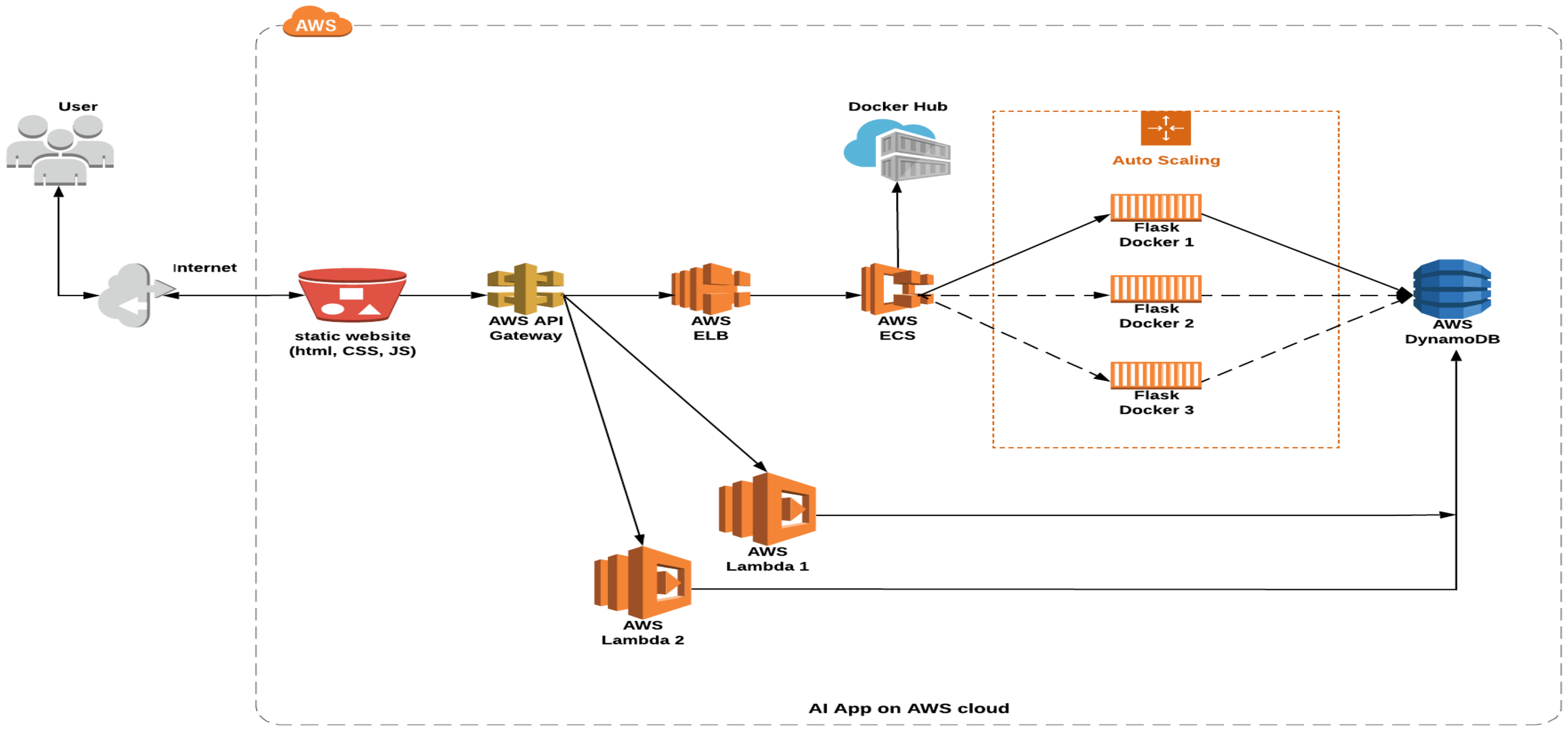
Bertelsmann Challenge Cloud Scholarship Project

Deploy An AI Sentiment Prediction App to AWS Cloud

Collaborated by
Adrik Sondarjee, Audrey S Tan & Christopher Rauh

21st February 2020

Infrastructure of the AI Sentiment Prediction App on AWS



GitHub Repository and Project Website URL

Project information repo: ai-projpage

Code and artifact repo I: ai-frontend

Code and artifact repo II: ai-automation

Code and artifact repo III: ai-backend

.

Website: AI Sentiment Prediction APP on AWS

About the Project, Goals and Team

About the Project: transform the original infrastructure of an AI sentiment prediction app (trained on the RNN model) to an AWS cloud deployable infrastructure.

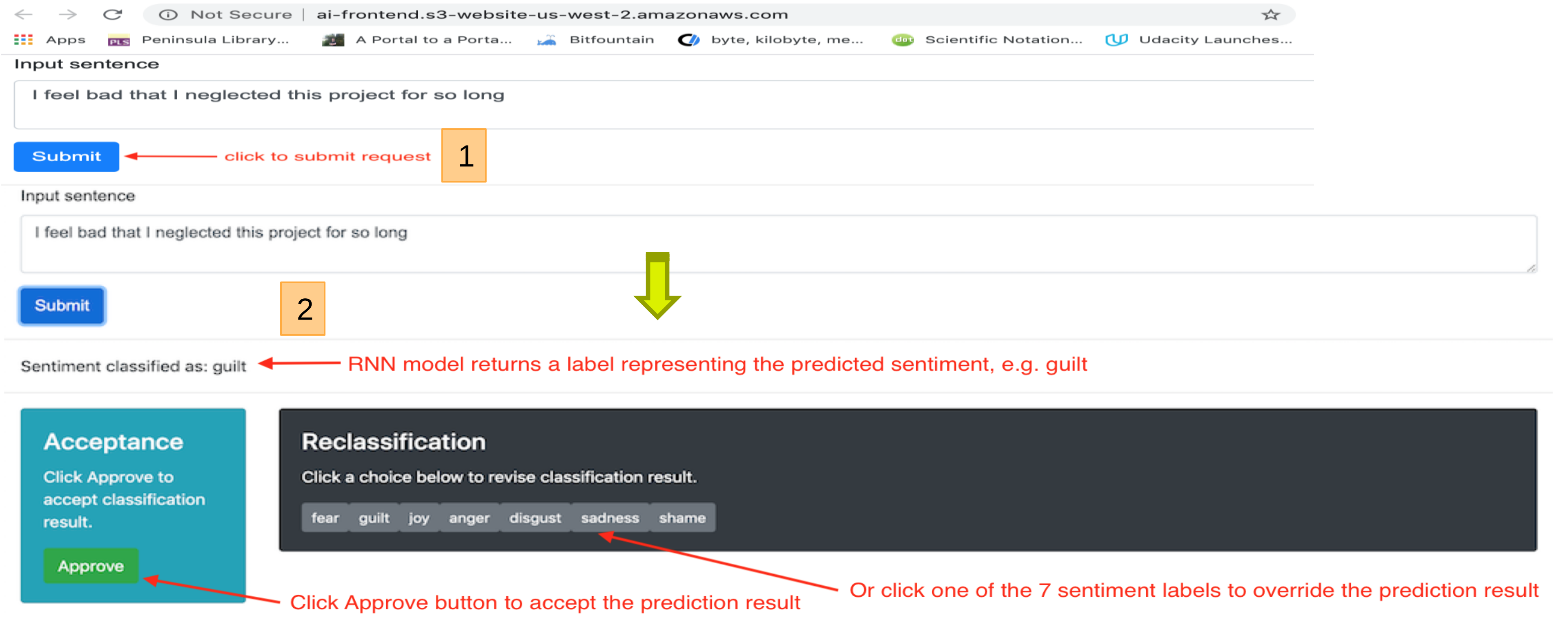
Our goals: Implements various AWS stack concepts covered in Bertelsmann Challenge Cloud Phase I from Lesson 12 thru 23, plus additional advanced concepts like Serverless Framework, CI/CD, Docker, DynamoDB and Microservices.

Our team: an international team with 3 members from Bertelsmann Challenge Cloud Phase I:

Adrik Sondarjee (France) - DevOps Engineer
Audrey S Tan (Australia) - Data Analyst
Christopher Rauh (Germany) - Software Engineer

Website: AI Sentiment Prediction APP on AWS

AI Sentiment Prediction App Use Case Demo – Scenario I



Not Secure | ai-frontend.s3-website-us-west-2.amazonaws.com

Apps Peninsula Library... A Portal to a Porta... Bitfountain byte, kilobyte, me... Scientific Notation... Udacity Launches...

Input sentence

I feel bad that I neglected this project for so long

Submit

click to submit request

1

Input sentence

I feel bad that I neglected this project for so long

Submit

2

Sentiment classified as: guilt

RNN model returns a label representing the predicted sentiment, e.g. guilt

Acceptance

Click Approve to accept classification result.

Approve

Reclassification

Click a choice below to revise classification result.

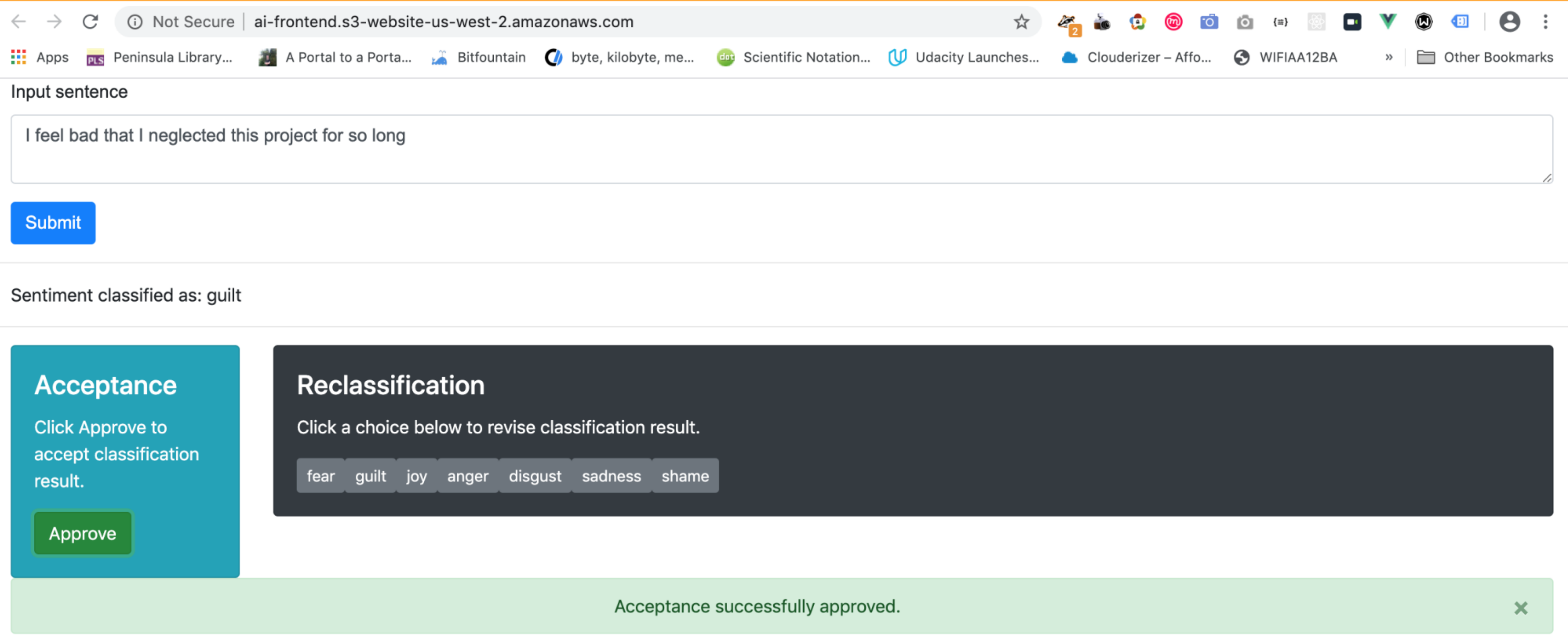
fear | guilt | joy | anger | disgust | sadness | shame

Click Approve button to accept the prediction result

Or click one of the 7 sentiment labels to override the prediction result

1. User enters a text in the web UI and click Submit to get a sentiment prediction result
2. The model returns a label (e.g. guilt) representing the predicted sentiment, which user can approve or revise

AI Sentiment Prediction App Use Case Demo – Scenario I (cont'd)



Input sentence

I feel bad that I neglected this project for so long

Submit

Sentiment classified as: guilt

Acceptance

Click Approve to accept classification result.

Approve

Reclassification

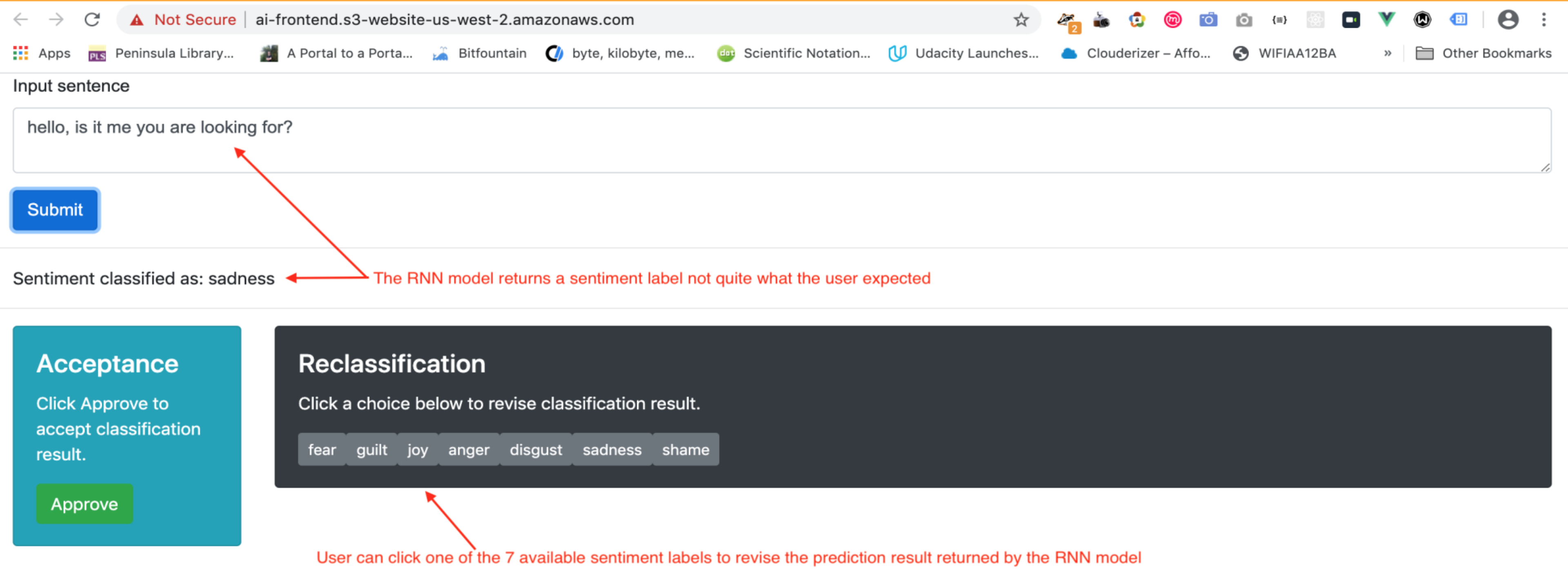
Click a choice below to revise classification result.

fear guilt joy anger disgust sadness shame

Acceptance successfully approved.

User clicks Approve button to accept the prediction result. The result is recorded to the DynamoDB

AI Sentiment Prediction App Use Case Demo – Scenario II



Not Secure | ai-frontend.s3-website-us-west-2.amazonaws.com

Apps | Peninsula Library... | A Portal to a Porta... | Bitfountain | byte, kilobyte, me... | Scientific Notation... | Udacity Launches... | Clouderizer – Affo... | WIFIAA12BA | » | Other Bookmarks

Input sentence

hello, is it me you are looking for?

Submit

Sentiment classified as: sadness

The RNN model returns a sentiment label not quite what the user expected

Acceptance

Click Approve to accept classification result.

Approve

Reclassification

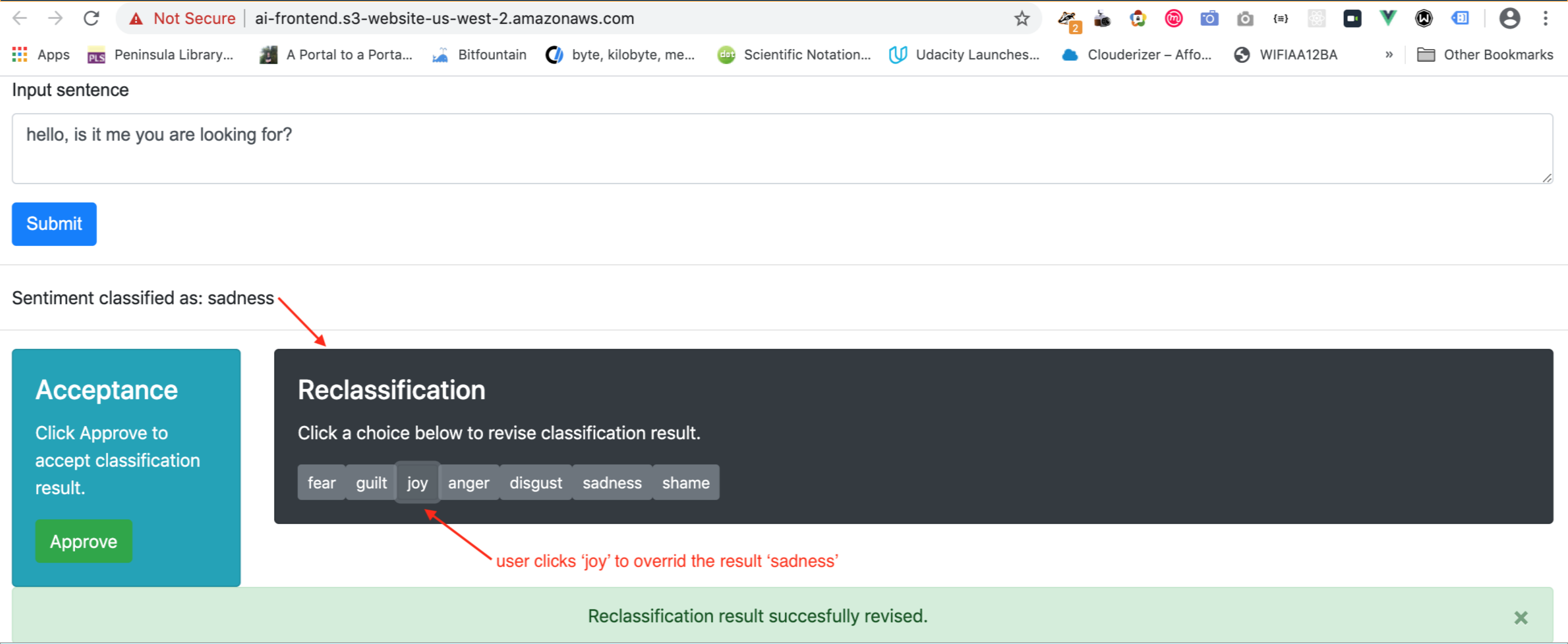
Click a choice below to revise classification result.

fear | guilt | joy | anger | disgust | sadness | shame

User can click one of the 7 available sentiment labels to revise the prediction result returned by the RNN model

The RNN model prediction result returned does not quite met the user's expectation. The user can click 1 of the 7 available label to override the returned result.

AI Sentiment Prediction App Use Case Demo – Scenario 2 (cont'd)



Not Secure | ai-frontend.s3-website-us-west-2.amazonaws.com

Apps | Peninsula Library... | A Portal to a Porta... | Bitfountain | byte, kilobyte, me... | Scientific Notation... | Udacity Launches... | Clouderizer – Affo... | WIFIAA12BA | » | Other Bookmarks

Input sentence

hello, is it me you are looking for?

Submit

Sentiment classified as: sadness

Acceptance

Click Approve to accept classification result.

Approve

Reclassification

Click a choice below to revise classification result.

fear | guilt | joy | anger | disgust | sadness | shame

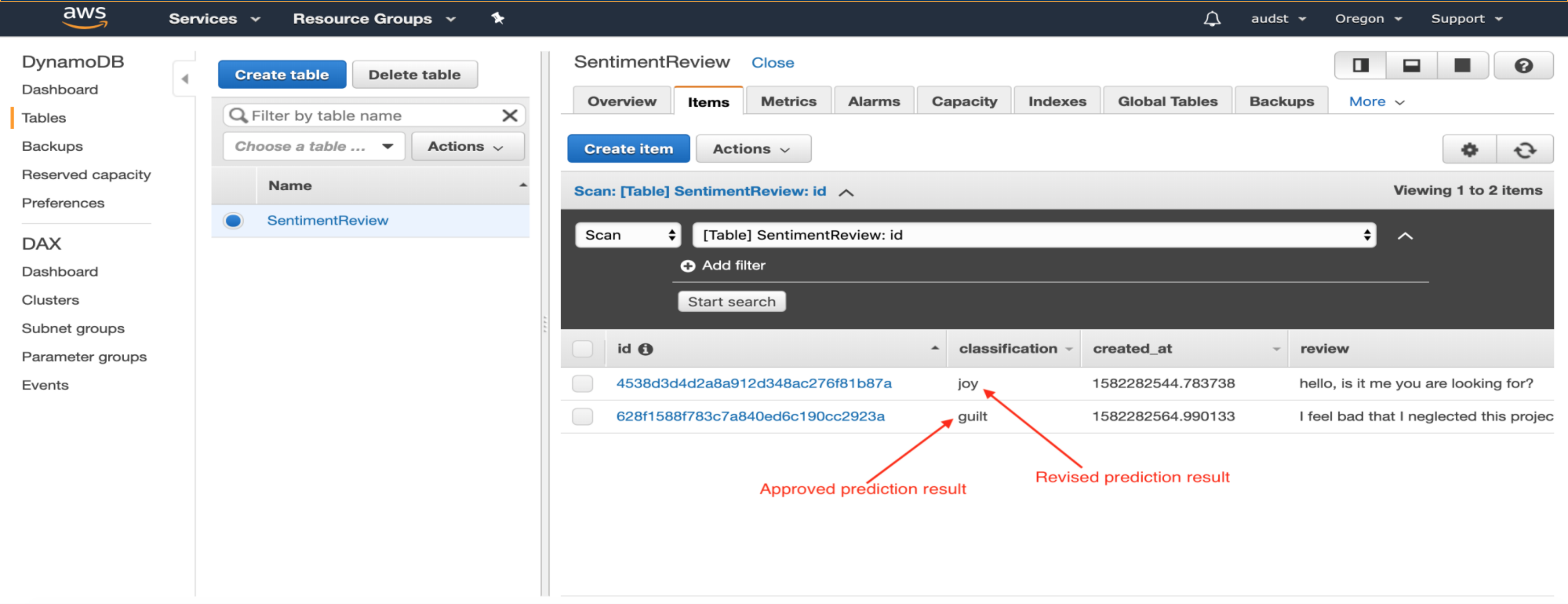
user clicks 'joy' to overrid the result 'sadness'

Reclassification result succesfully revised.

User clicks 'joy' label to override the returned result 'sadness'. The revised result is recorded to the DynamoDB.

.

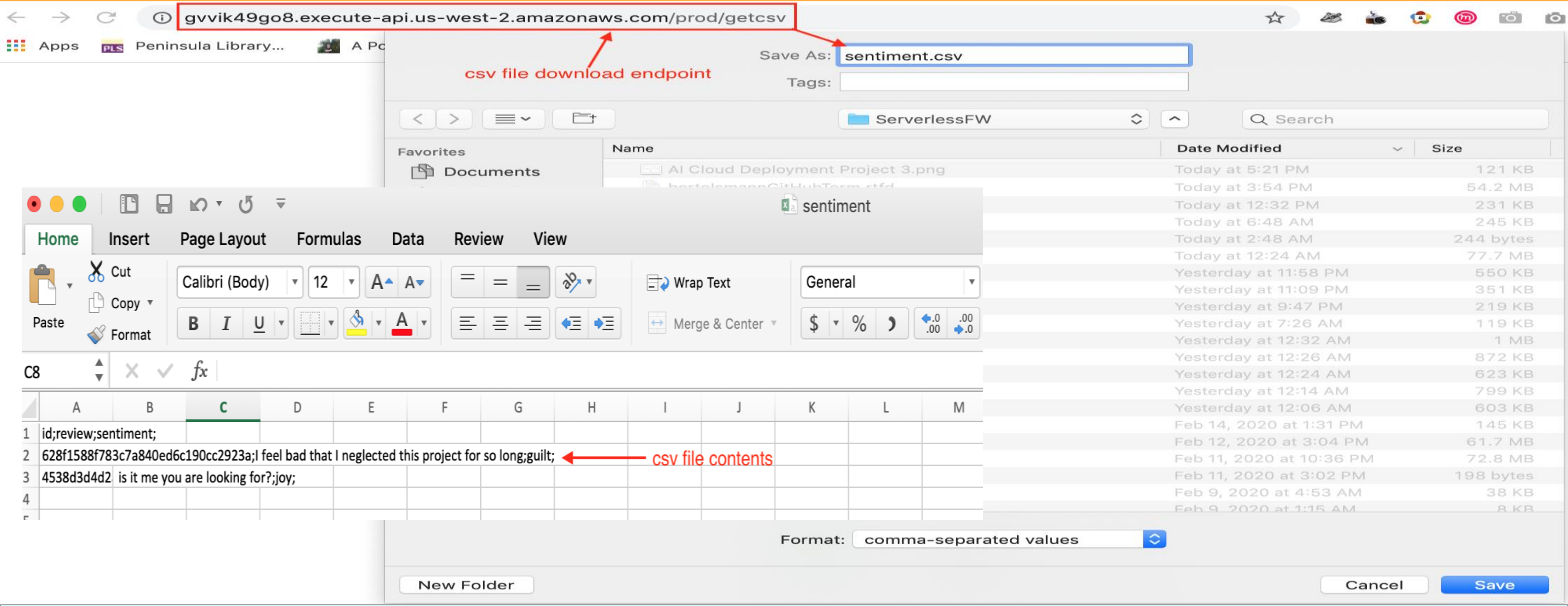
AI Sentiment Prediction App Use Case Demo – Scenario III



id	classification	created_at	review
4538d3d4d2a8a912d348ac276f81b87a	joy	1582282544.783738	hello, is it me you are looking for?
628f1588f783c7a840ed6c190cc2923a	guilt	1582282564.990133	I feel bad that I neglected this projec

Approved and revised prediction results are stored in the DynamoDB. The data can be exported to a csv file from the Web UI as a new dataset for retraining the RNN Sentiment Prediction model.

AI Sentiment Prediction App Use Case Demo – Scenario III (cont'd)



gvvik49go8.execute-api.us-west-2.amazonaws.com/prod/getcsv

csv file download endpoint

Save As: sentiment.csv

Tags:

ServerlessFW

Name	Date Modified	Size
AI Cloud Deployment Project 3.png	Today at 5:21 PM	121 KB
	Today at 3:54 PM	54.2 MB
	Today at 12:32 PM	231 KB
	Today at 6:48 AM	245 KB
	Today at 2:48 AM	244 bytes
	Today at 12:24 AM	77.7 MB
	Yesterday at 11:58 PM	550 KB
	Yesterday at 11:09 PM	351 KB
	Yesterday at 9:47 PM	219 KB
	Yesterday at 7:26 AM	119 KB
	Yesterday at 12:32 AM	1 MB
	Yesterday at 12:26 AM	872 KB
	Yesterday at 12:24 AM	623 KB
	Yesterday at 12:14 AM	799 KB
	Yesterday at 12:06 AM	603 KB
	Feb 14, 2020 at 1:31 PM	145 KB
	Feb 12, 2020 at 3:04 PM	61.7 MB
	Feb 11, 2020 at 10:36 PM	72.8 MB
	Feb 11, 2020 at 3:02 PM	198 bytes
	Feb 9, 2020 at 4:53 AM	38 KB
	Feb 9, 2020 at 1:15 AM	8 KB

Format: comma-separated values

sentiment

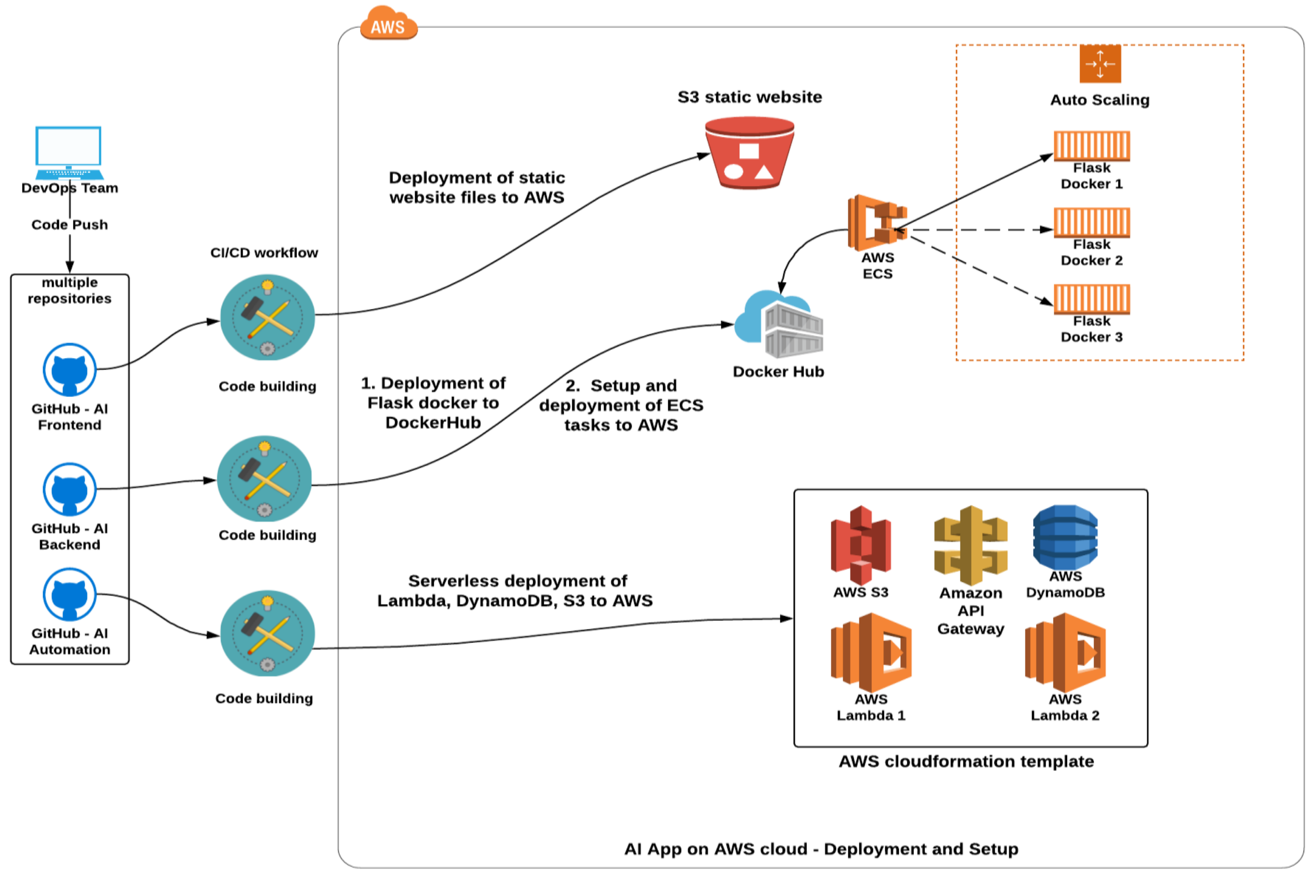
C8

```
1  id;review;sentiment;
2  628f1588f783c7a840ed6c190cc2923a;I feel bad that I neglected this project for so long;guilt;
3  4538d3d4d2  is it me you are looking for?;joy;
```

csv file contents

User accesses the csv file download endpoint to download prediction results stored in the DynamoDB. This csv file can then be used as a new dataset for retraining the RNN Sentiment Prediction model.

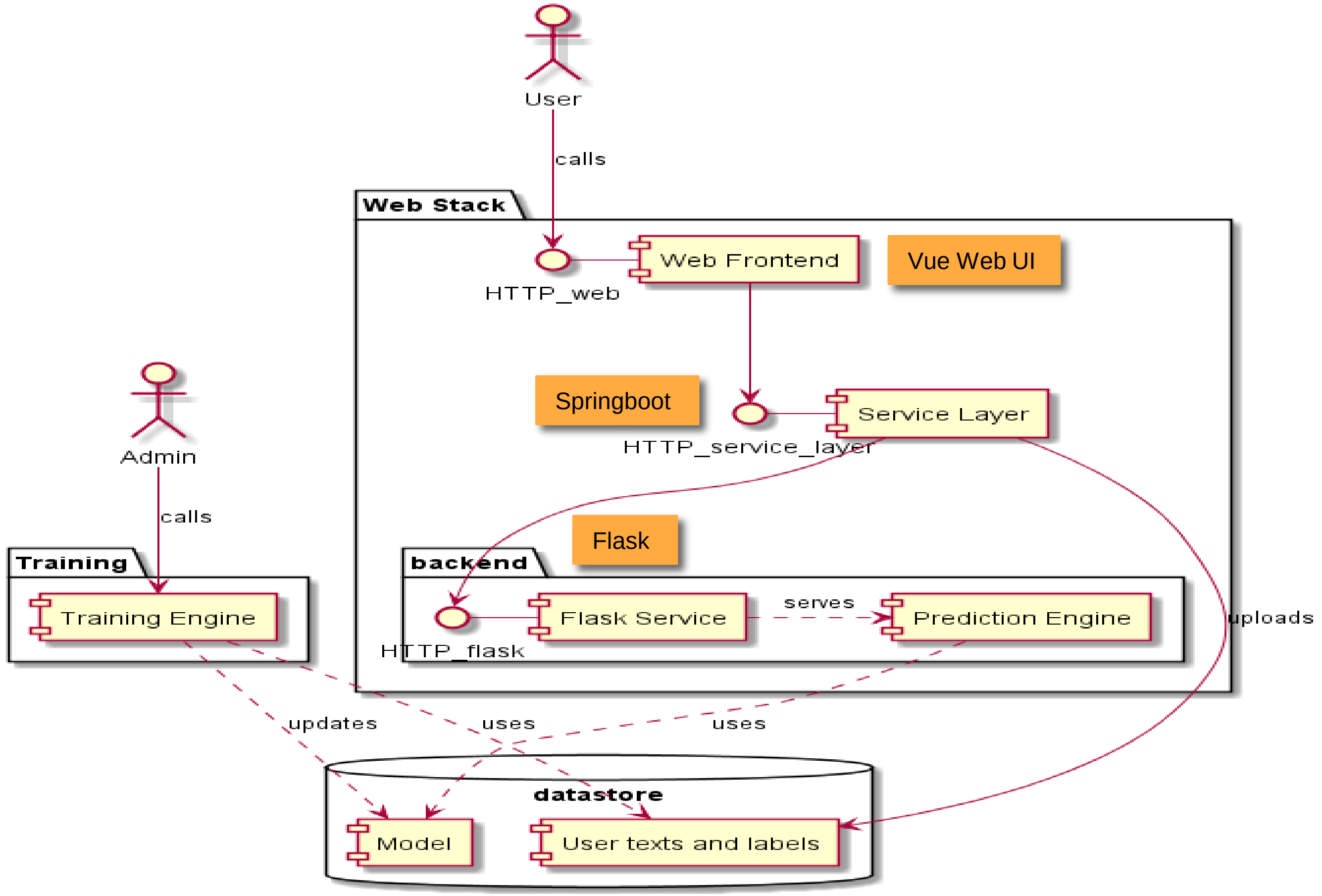
Project Artifact Repositories



The project has 3 code and artifact repositories.

- ai-frontend
  - contains the project website static files index.html and app.js.
  - on updates to the files, they are copied to the S3 bucket hosting the project website on AWS
- ai-backend
  - contains the code files for building a Flask docker image
  - on updates to the files, the docker is rebuilt and ECR task triggered to start container operation on AWS
- ai-automation
  - contains the Serverless Framework configuration and Lambda function code files
  - on updates to the files, Serverless Framework is started to deploy a cloudformation template to AWS

Original Infrastructure – Logical View

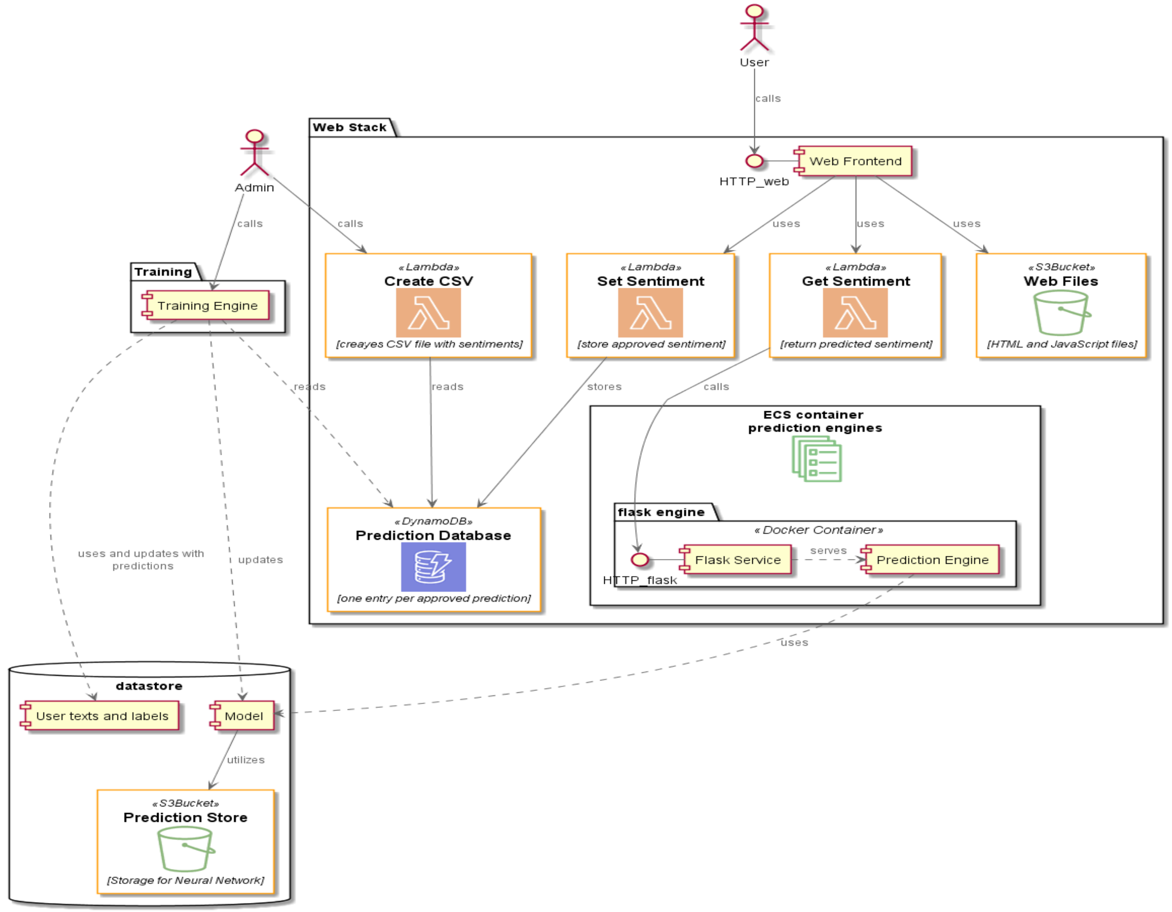


The original architecture of the AI app consists of

- A Flask app backend hosting a RNN sentiment prediction model
- A website with a Vue Web UI powered by Springboot Framework
- A datastore built on MySQL

It is styled to operate in Microservices fashion. This makes the infrastructure and its underlying components easily transformable to AWS cloud deployable infrastructure.

Cloud Infrastructure – Logical View



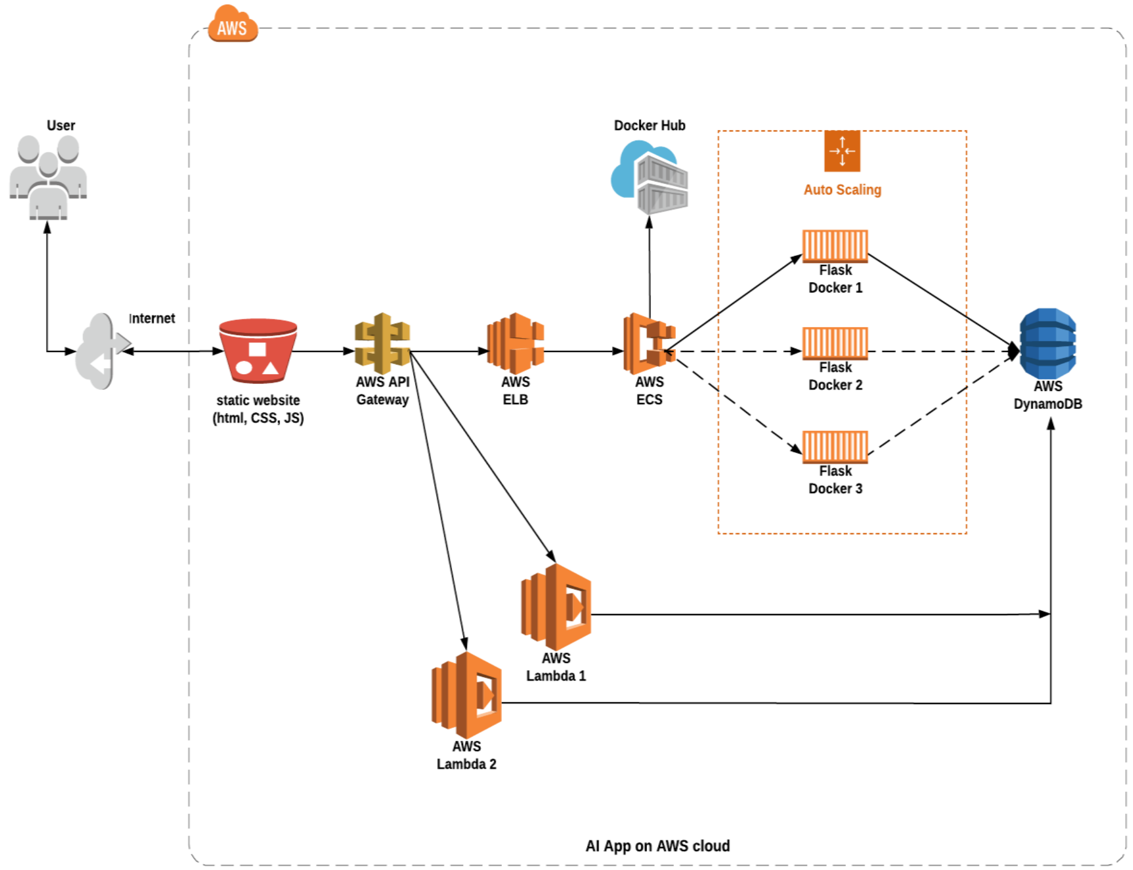
The re-architecting effort resulted in a streamlined infrastructure as below:

- the Flask backend now in a docker container utilizes AWS ECS
- the website is now hosted on S3 bucket powered by AWS Lambda functions :
- the MySQL datastore now replaced by a light weight noSQL DynamoDB

The new AWS cloud infrastructure comes with these benefits:

- costly specialist support effort in Springboot, MySQL, Infrastructure resource deployment & provisioning no longer needed
- built-in auto failover and user demand driven infrastructure scaling features
- predictable operation performance with minimum effort and improved overall user experience

Cloud infrastructure – Physical Implementation



The infrastructure incorporated various AWS resources & DevOps functionality (denoted by *).

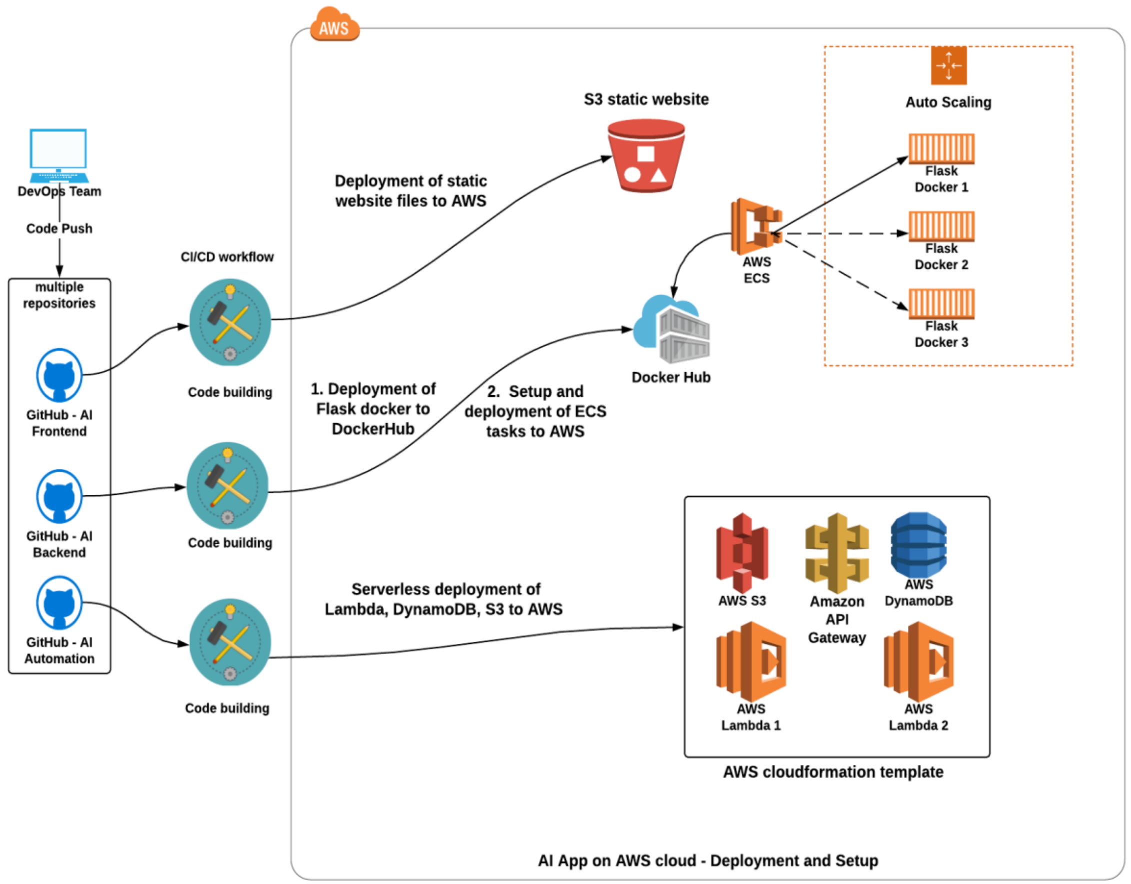
Cloud Lesson Concepts Implemented

- GitHub Repos (**ai-backend**, **ai-automation** & **ai-frontend**): Lesson 1-12
- AWS S3 Static Website: Lesson 14, 23
- AWS Lambda function: Lesson 13
- AWS Elastic Load Balancer: Lesson 16, 20
- AWS Auto Scaling Group: Lesson 20
- AWS Cloudformation: Lesson 19
- AWS IAM: Lesson 15

Advanced Concepts Implemented

- GitHub CI/CD workflow pipelines*
- AWS API Gateway
- AWS Elastic Load Balancer
- AWS ECS (elastic container service)
- DockerHub (container image registry)
- Flask Docker (min 1, max 3)
- AWS DynamoDB
- AWS Serverless Framework*
- Microservices*

Cloud Infrastructure Deployment Workflow



1. DevOps team merges feature branches to master and pushes to 1 of the 3 remote masters

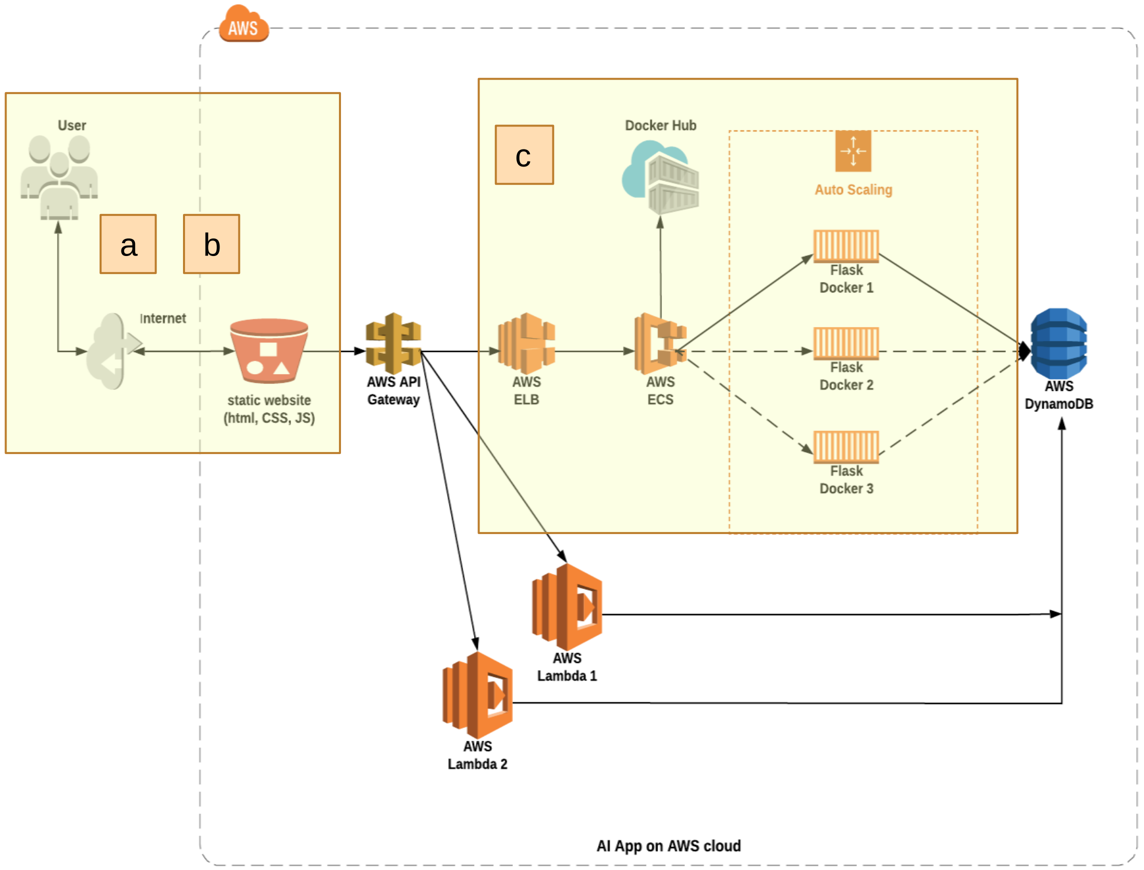
2. Code build – 3 build paths:

a. If the push is onto ai-frontend repo, CI/CD Action **Upload Website** automatically runs to upload updated static files (index.html, app.js) to AWS S3 website

b. if the push is onto ai-backend repo, CI/CD Action **Deploy to Amazon ECS** automatically runs to build a new Flask container to push to DockerHub, then deploys a new ECS task definition to start container operation on AWS

c. If the push is onto ai-automation repo, CI/CD Action **Serverless deployment** automatically runs Serverless Framework to deploy an AWS cloudformation template that keeps the Lambda functions, their triggering events and required resources up to date on AWS

Cloud Infrastructure Operation Workflow



RNN Sentiment Prediction App Operation

a. User submits a sentiment prediction request thru website UI and receives a result
   I. User approves the prediction result, the approved result is written to the DynamoDB
   II. User revises the prediction result, the revised result is written to the DynamoDB
b. User downloads prediction results stored in the DynamoDB as a CSV file for use as a new dataset for retraining of the RNN model
c. Depending on website traffic, AWS ECS and Auto Scaling group orchestrate to scale up to 3 Flask container instances to optimize workload distribution and app response time

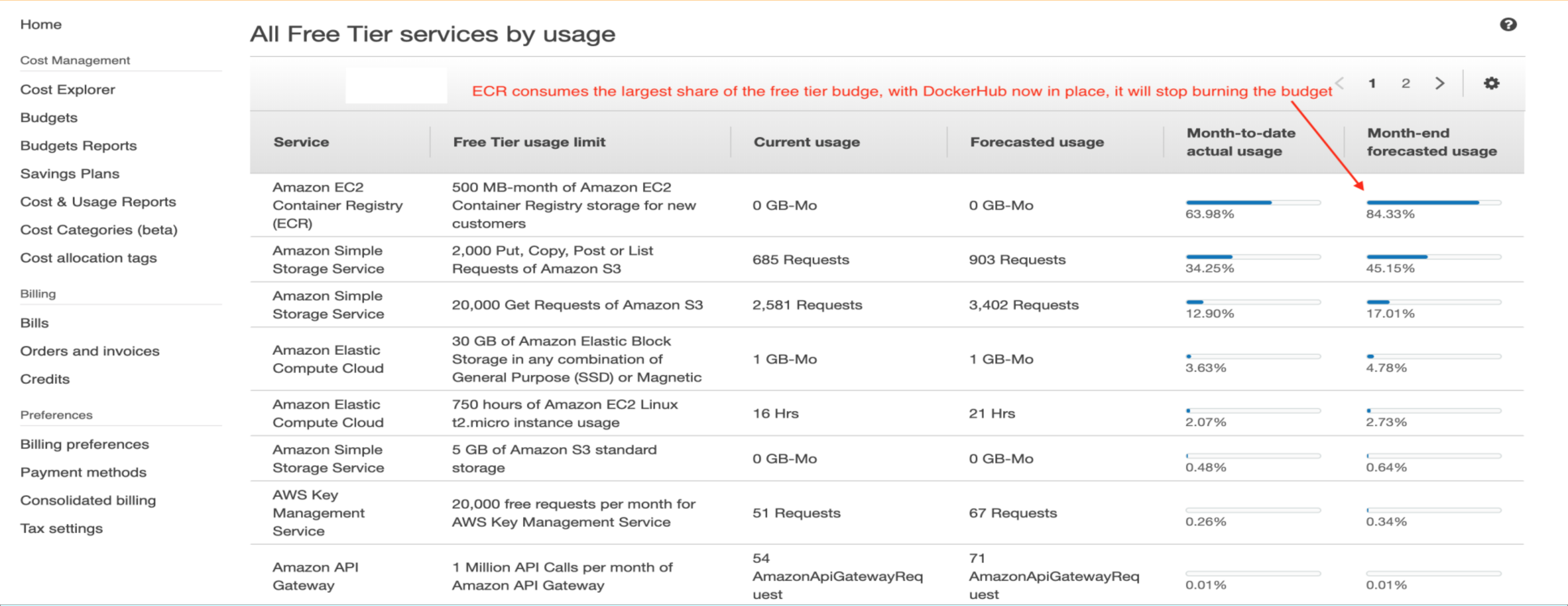
Implementation Impediments and Resolutions

This is a POC (proof of concept) project the project team put together to implement and practice basic cloud DevOps concepts from this phase I Challenge and experiment advanced concepts nominated by team members.

The project was 100% unfunded and utilized our AWS free-tier account to conduct the POC. Below is the impediments we experienced during the implementation and how we resolved them:

	Impediment	**Resolution**
1	AWS Route 53 is not part of the free tier, can't use it to get a custom domain name for the project website.	Used S3 generated endpoint as the website URL to avoid incur extra charges.
2	AWS EKS (Elastic Kubernetes Service) is the ideal tool for implementing microservices, failover and user demand driven dynamic infrastructure scaling concepts, but it is not part of the free tier. AWS Fargate can be used to overcome operation limitations of Auto scaling group + ECS, but it is not part of the free tier.	Used auto scaling group with the free ECS to meet implementation goals. Set default scaling factor to one t2.micro EC2 instance to eliminate extra charges. For the POC, the limitations were not a concern, Fargate could be omitted.
3	The Flask docker image builds will quickly exceed the 500MB-month of storage for ECR on free tier.	To save cost, push Flask docker images onto DockerHub in lieu of AWS ECR.
4	Keeping RNN Sentiment Prediction App running on AWS cloud for live website demo may incur extra charges quickly.	With the 3 preceding cost saving measures and cost budget monitoring, the **All Free Tier services by usage** on the Cost Management Console should stay within budget.

Free Tier Budget Analysis and Usage Forecast



All Free Tier services by usage

ECR consumes the largest share of the free tier budge, with DockerHub now in place, it will stop burning the budget

Service	Free Tier usage limit	Current usage	Forecasted usage	Month-to-date actual usage	Month-end forecasted usage
Amazon EC2 Container Registry (ECR)	500 MB-month of Amazon EC2 Container Registry storage for new customers	0 GB-Mo	0 GB-Mo	63.98%	84.33%
Amazon Simple Storage Service	2,000 Put, Copy, Post or List Requests of Amazon S3	685 Requests	903 Requests	34.25%	45.15%
Amazon Simple Storage Service	20,000 Get Requests of Amazon S3	2,581 Requests	3,402 Requests	12.90%	17.01%
Amazon Elastic Compute Cloud	30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic	1 GB-Mo	1 GB-Mo	3.63%	4.78%
Amazon Elastic Compute Cloud	750 hours of Amazon EC2 Linux t2.micro instance usage	16 Hrs	21 Hrs	2.07%	2.73%
Amazon Simple Storage Service	5 GB of Amazon S3 standard storage	0 GB-Mo	0 GB-Mo	0.48%	0.64%
AWS Key Management Service	20,000 free requests per month for AWS Key Management Service	51 Requests	67 Requests	0.26%	0.34%
Amazon API Gateway	1 Million API Calls per month of Amazon API Gateway	54 AmazonApiGatewayRequest	71 AmazonApiGatewayRequest	0.01%	0.01%

The **All Free Tier services by usage** report on the Cost Management Console shows ECR is the largest consumer of the free tier budget. With DockerHub in its place, ECR will stop burning the budget. The AI Sentiment Prediction App on AWS cloud can now stay on for live demo purpose