DALHOUSIE UNIVERSITY

CSCI 5409 - Cloud Computing

Group 25 – CloudUp
Final Report

Group Members

Hardee Rameshchandra Garala	B00869195
Kavan Patel	B00869224
Sarthak Patel	B00912612

Faculty - Prof. Robert Hawkey

Table of Contents

Project Overview:

As part of the cloud project, we decided to create a game "Word On Cloud". It is a word game where the player needs to guess a word based on color-coded hints. An individual can play it alone while in isolation or just for enjoyment. A professor can give test questions on this game and make the quiz entertaining. It can be played by small kids to whom this can help increase vocabulary, general knowledge, of any field they would like to explore. A host of the party can make the question of the quiz fun, personal and leading to laughter. The main aim of our game is fun with knowledge and entertainment.

Our application requires registered users to log in to the game, they need to verify their email in order to be able to log in. On login, the user will get two options. One option is to create a word and the other is to play.

When a user selects the option to play, they will be given a grid to enter a random word which can be guessed from the number of letters that word has. On guessing the word, the user will be provided with a hint on to how much extent the word is correct. If a letter of the word guessed is present in the actual word and is also present at the right place the letter will be highlighted in green, if the letter guessed is present in the actual word but is not at the right position then it will be highlighted in the gray and if the letter guessed is not present in the actual word, then it will be highlighted red. For example, the actual word is "cloud", and someone guesses the word as "drone". Here we can see that "o" is at the correct position so it will be highlighted green, "d" is present in the actual word but not at the correct position so it will be gray, "r", "n" and "e" are not present in actual word so they will be highlighted in red color. The player will also be able to check the leaderboard which will show the number of attempts, the time it took for the correct attempt, the score they achieved through a logic based on the total time taken for guessing the correct word, and the number of attempts taken to guess the correct word. The leaderboard also shows where they stand as compared to other people who played this word. In this format, the same word will be displayed to all people at a certain time to make the leaderboard unbiased. This word will change at an interval of 24 hours. Now let's say the user forgot the password! Yes, they can update it. We will send a change password email to the user.

When the user selects the option to create a game. They can decide if they want to provide the hint in form of a question or just a word. Once they submit the hint and the answer, a link will be generated which can be shared with the group of people who wants to play. When this shared link is visited, the player does not require to register themselves. They can play directly, and this link will remain valid only for a specific time. In this format, we will not display the leaderboard.

In building this game we have used Amazon Web Services like Elastic Beanstalk, Lambda for computing, DynamoDB for storage, VPC for Network, and others. The game will be available as software to play and therefore, the delivery model is SaaS (Software as a Service).

GIT Repository Link: **https://git.cs.dal.ca/courses/2022-summer/csci4145-5409/group-25**

Application Link: **http://awseb-e-h-awsebloa-kuuivj5bb0ic-672384310.us-east-1.elb.amazonaws.com/login**

Final Architecture:

Our application is a web application, the frontend for which is built in react js and the backend is built using nodejs and expressjs frameworks. The frontend is hosted on the Elastic Beanstalk. The AWS Cognito is used for user management. Elastic Beanstalk will automatically scale the instance if more userbase will come. And for the backend, we are using the lambda function to build the APIs. All game data is stored in the DynamoDB database. The dictionary of the words is stored in the s3 storage. To fetch the word at a specific time we have set a rule in Eventbridge which calls the lambda function which will then fetch the random word from the dictionary and store that word in the DynamoDB. All of these AWS services will be configured with AWS Virtual Private Cloud (VPC).

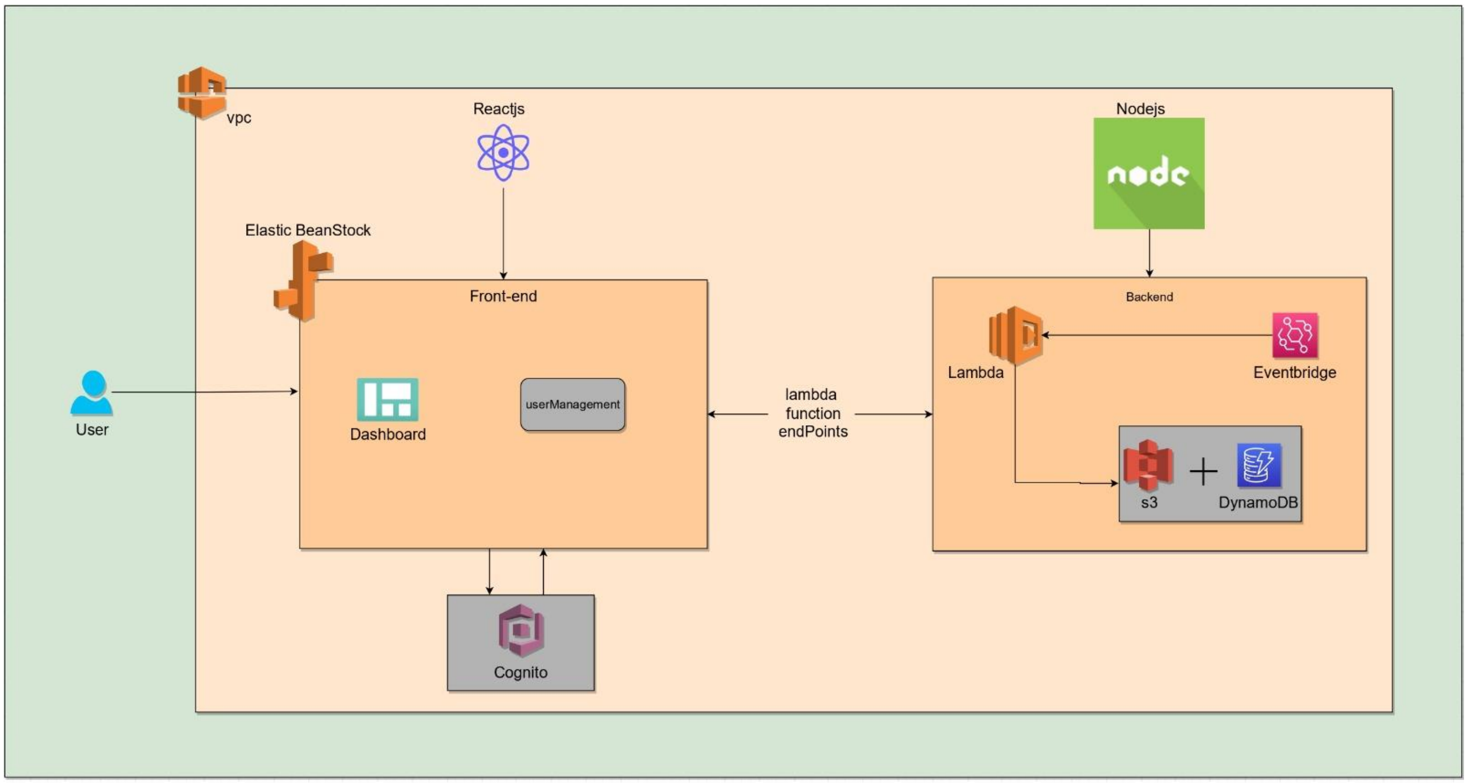


Figure 1: The architecture of the application

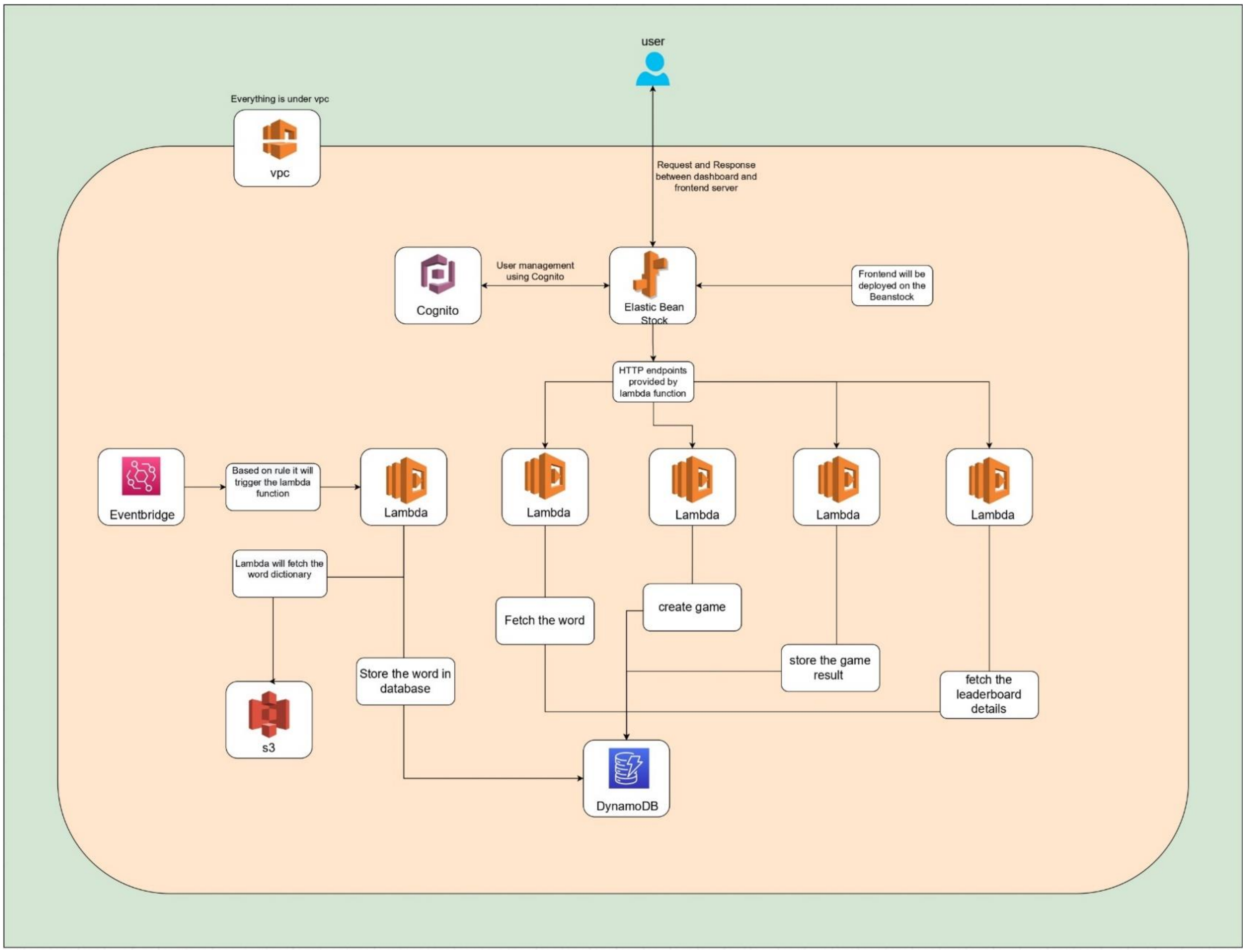

user
Everything is under vpc
vpc
Request and Response between dashboard and frontend server
Cognito
User management using Cognito
Elastic Bean Stock
Frontend will be deployed on the Beanstock
HTTP endpoints provided by lambda function
Eventbridge
Based on rule it will trigger the lambda function
Lambda
Lambda
Lambda
Lambda
Lambda
Lambda will fetch the word dictionary
Fetch the word
create game
store the game result
fetch the leaderboard details
Store the word in database
s3
DynamoDB

Figure 2: Web Application's dataflow diagram

How do all of the cloud mechanisms fit together to deliver your application?

The front end of our web application is hosted on Elastic Beanstalk which interacts with other services such as AWS Cognito, AWS S3, AWS DynamoDB, and AWS EventBridge with help of Lambda Functions. All of these services are configured within AWS VPC, allowing us to create our own private network area. To be more explicit, by utilizing AWS VPC, we can eliminate the direct connection between our frontend and backend services. As a result, AWS VPC adds security by concealing AWS Elastic Beanstalk services within the network. All access requests are sent through AWS Elastic Beanstalk to AWS Cognito, which authenticates users using legitimate credentials such as email address and password.

Users are sent to the application's dashboard page after successfully logging in, where they may decide whether to play the game or make a new one. If a user chooses to participate in the game, an API request is performed via a Lambda function that is triggered by an AWS EventBridge. The Lambda function retrieves a random word from an AWS S3 bucket that has a dictionary of words and then sends the result back to the front end. Using the HTTP endpoints made available by the second Lambda function, DynamoDB will get information about whether the user correctly identified the word within the allotted chances or not. The user's username and the likelihood that they correctly or incorrectly guess the word as returned from the front end will be fetched via a lambda function and stored in a table in DynamoDB. If the user wins the game, DynamoDB will keep track of the time and odds at which they received the correct answer, as well as whether they won or lost.

If the user wants to create a game in this case the front-end application takes the hint and word from the user as data input and will then call the HTTP request provided by the third Lambda Function. The username, hint, and created word sent from the front end will be retrieved by this Lambda function, which will then store the information in a database in DynamoDB. In addition, when a user hits the submit button to create a new game, the Lambda function will build a new link that will be returned as a response to the front end. The user may then send this link to others and play a game to try and guess the word that the user just created. A new HTTP request will also be made when a user clicks on the Leaderboard page, triggering a fourth Lambda function to retrieve the usernames of the top 5 users as well as each user's score from a DynamoDB table. This response will then be sent to the front end, which will retrieve the data and display it on the web application.

Where is data stored?

Figure 2 represents the application's data flow, which gives a detailed explanation of user requests and data through several cloud services [1] and receives generated responses from the system. All the data of the game is stored in s3 storage and DynamoDB. The dictionary of the words is stored in s3. From this dictionary, periodically a word is fetched and stored in DynamoDB to get on the frontend in play game functionality. Here we are fetching a word periodically rather than every time a user chooses to play the game and store it to DynamoDB to avoid calling s3 multiple times and make it cost-efficient. Moreover, when the user guesses a word the number of attempts it took for the user to guess the word correctly and the time taken to guess the word correctly is stored in DynamoDB. With this time and number of attempts, we are also calculating the score which is stored in DynamoDB. We also store the username along with whether the user is able to correctly identify the word within the allocated six chances. While for the create game functionality we store the username the hint/question they have created and the answer to that question in DynamoDB. Users’ sign-up details are stored and managed by Cognito.

What programming languages did you use (and why) and what parts of your application required code?

- Front-end: React JS
- Back-end: NodeJS

React is used in front-end development. We chose these Frameworks since they support all of the user's capabilities. ReactJS allows for major data changes, which result in automated updates to chosen areas of user interfaces. Because of this progressive feature, you do not need to do any additional functions to update your user interface. Furthermore, ReactJS provides reusable components that developers utilize to design a new application [2]. Reusability is similar to a developer's remedy. Another reason for using the React framework is the expertise of our team. Last but not least, React allows developers to reuse components created for another application with the same functionality. As a result, the development work is reduced while assuring perfect performance.

NodeJS is used for the backend development of the application. We used NodeJS as it is lightweight and has an event-driven approach which is helpful to build the real-time application. It also provides lots of built-in libraries which are very helpful for our application. The whole backend of the application is implemented in an AWS lambda function and rather than using an API gateway we used AWS lambda's inbuild functionality called function URL which provides a public URL which is accessible outside of the AWS network. The user stats and game data are stored in DynamoDB using the lambda function. All the lambda functions are written in NodeJS.

How is your system deployed to the cloud?

Using CloudFormation, we deployed the application's front-end to AWS Elastic Beanstalk. AWS Elastic Beanstalk's primary goal is to deploy apps to the cloud. This is what is used to install and operate this application. We placed our lambda codes in the lambda folder, one file named words.txt containing a dictionary of 5 letter words, and one folder containing the zip of our source code into the AWS S3 bucket and attached that bucket access link to the CloudFormation deployment file. When we execute the YAML file, the AWS stack will accept the code from the S3 Bucket and conduct all of the necessary steps to properly deploy tasks, eventually returning the deployment URL. We have configured the CloudFormation file to create all of the lambda functions, the EventBridge rule using cron jobs to trigger the lambda function every 12 hours, the Cognito user pool for authentication and authorization, the Dynamo DB tables required to store user details, and the Virtual Private Network to contain all of these services via private subnet and Elastic Beanstalk via public subnet. CloudFormation is used to configure all of these services.

Proposed Architecture VS Final Architecture:

If your final architecture differs from your original project proposal, explain why that is. What did you learn that made you change technologies or approaches? If your final architecture is the same as your original project proposal, reflect now on whether there is something you could have implemented with what you know now that would have resulted in a better architecture or a more cost-efficient product?

Our final architecture is the same as the architecture in our first project proposal. We may have incorporated a few things to improve architecture to make it more cost-effective. For example, we may leverage the DynamoDB table's auto-scaling and on-demand functionality. Enabling these capabilities allows us to pay-per-request for reading and writing requests, allowing us to easily balance costs and performance. Furthermore, the SQS queue may be used as an alternative for workloads that need a lot of writing [3]. When lambda is linked directly to a DynamoDB table, it is unknown how much capacity will be required on the DynamoDB side. Setting a throttling limit for this second Lambda function simplifies capacity allocation on the database side, allowing us to employ the considerably cheaper Provisioned capacity mode.

When we utilize the JavaScript SDK in a lambda function, the complete SDK is imported, although our function may only use one or two AWS services. So, we may have reduced memory use and setup time by requesting just the services needed in our lambda function. We should have modified the needed line to specify the service and then sent any configuration parameters to the service constructor [4]. In this way, we'll be able to make Lambda more memory efficient, lowering total costs

Future possibilities of Application

How would your application evolve if you were to continue development? What features might you add next and which cloud mechanisms would you use to implement those features?

We were able to achieve all the initially proposed features. However, due to time restrictions, we were not able to implement the new ideas and updates to features that we got while developing the current application. We still are planning to update the features and make the application better. The first idea we thought of was in the play game feature, to provide the player with an option to select from a category of options they would like to get the word from. Moreover, give them an option to select difficulty level so that it provides flexibility for kids and adults to play as per difficulty and domain. We will use s3 storage to store the dictionary of different category words.

Moreover, in create game feature, currently, we do not have a leaderboard as the player is not required to register. But there is a possibility that we can collect a player's name, not necessarily their real name when they click to play the game created by a verified user. Later we can use this name to display the score and ranks of all the people who played through that link. For this we will use lambda function and will get the user time, number of attempts details of the user from the dynamoDB for the logic to calculate score and return to the frontend on calling.

Other update we have thought of is providing the average score of all the games the player has played till now in their profile along with giving them how many times they have attempted correct one in one chance, how many times in 2 chances and so on till six attempts. For this also we are planning to use lambda and DynamoDB. Lastly, we have thought of making the application mobile-friendly.

Data Security

How does your application architecture keep data secure at all layers? If it does not, if there are vulnerabilities, please explain where your data is vulnerable and how you could address these vulnerabilities with further work.

Security to data at all layers is important in current times to protect it from getting unauthorized access and to mitigate risks to data. If the user data remains secure it helps hold the trust of the user. Hence to protect the information of the player we have used specific services that provide security. The services we used in our application architecture to make our application secure are as follows:

1. Amazon Virtual Private Cloud: VPC is a logically isolated area in AWS Cloud [1] and hence the access to data within it is restricted. This provides a layer of security to the application. As all of our data is present in the Virtual Private Network, unauthorized users or other cloud users will not be allowed access to this data, thus providing security to the data.
2. Amazon Cognito: We have added AWS Cognito for sign-up and sign-in user management. A user who wants to sign in will require to provide their email address which will be verified before they can log in thus adding security by eliminating the chances of people using incorrect email IDs or someone else's email IDs. Once a user is verified then only they are given access to all the functionalities of the application. The user is also supposed to add a password with specific conditions like a small letter, a capital letter, a special character, and a number required thus making it difficult to guess for other people and not able to guess easily through brute force.

Which security mechanisms are used to achieve the data security described in the previous question? List them, and explain any choices you made for each mechanism (technology you used, algorithm, cloud provider service, etc.)

The security mechanism we are used for achieving data security are,

1. AWS VPC: VPC will restrict the direct access to the cloud services we used from the internet. Only services in which we set up the public subnet can access from the

unauthenticated user or public. And the services which don't need public access and used internally have a private subnet assigned to them.

2. AWS Cognito: AWS Cognito provides a user pool where all our users' data is stored like email, password, username etc. It will store the password in an encrypted format, and it is not visible to anyone. User is only able to access the application APIs only if they are successfully authenticated using Cognito. Basically, Cognito works as a middleware for the authentication of the User. We can set a configuration on how user authentication will work while creating the user pool.

Costs

What would your organization have to purchase to reproduce your architecture in a private cloud while providing relatively the same level of availability as your cloud implementation? Try to give a rough estimate of what it would cost, don't worry if you are far off. These systems are complicated, and you don't know all the exact equipment and software you would need to purchase. Just explore and try your best to figure out the combination of software and hardware you would need to buy to reproduce your app on-premises.

When implementing all cloud services entirely inside of a private cloud, there are a number of factors and elements to take into account. For the database services and API services, we must first choose subnets and security groups. Also, we need good hardware where we can host our application frontend and backend. We also required good hardware to host our database. Furthermore, we need an extra layer of security which required more resources, which would increase expenses, as well as the possibility of other extraneous charges for designing and developing the application. In addition to that, once the application is deployed there will be maintenance charges. Therefore, when the user base grows, it is important to include maintenance costs while undertaking a critical review. We were able to replicate the architecture in the private cloud while maintaining the same level of availability as our present cloud deployment by considering all these factors.

Rough Estimation:

The list of hardware requires to reproduce the application on-premises are,

1. Computer devices and other accessories.
2. Internet connection
3. Storage devices for database
4. Setup firewall

This will cost around 3000$ - 4000$

The list of software required is,

1. Operating system license
2. Anti-virus software

3. Data encryption tools
4. Security tools

It will cost around 2000$ - 2500$

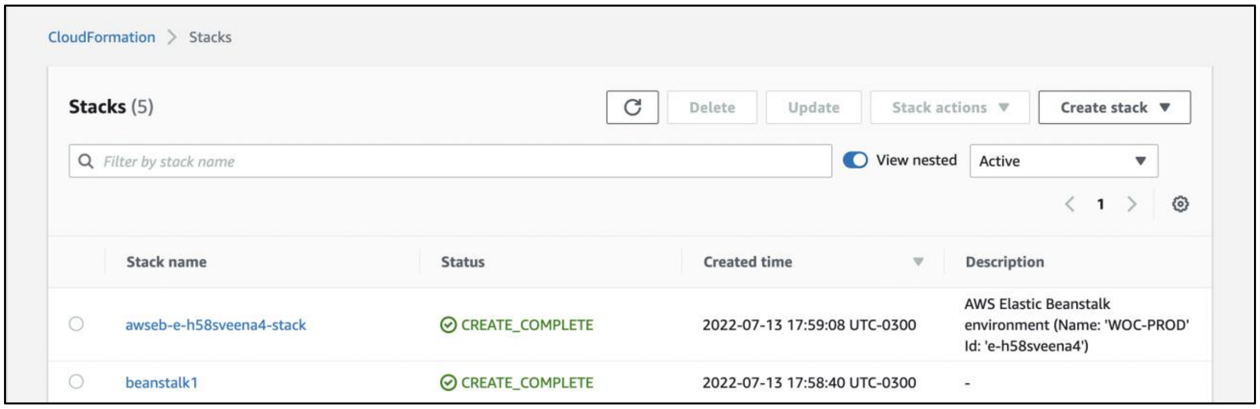
Additionally, the development and management of all these on-premises tools would require a team with significant experience. It will thus cost between 10,000$ to 12,000$ or more.

Which cloud mechanism would be most important for you to add monitoring to in order to make sure costs do not escalate out of budget unexpectedly? In other words, which of your cloud mechanisms has the most potential to cost the most money?

According to us, Elastic Beanstalk will be the most significant cloud technique for monitoring to ensure that expenditures do not unexpectedly escalate out of budget. Elastic Beanstalk handles deployment elements such as capacity provisioning, load balancing, auto-scaling, and application health monitoring automatically. The pricing and flexibility of Elastic Beanstalk are excellent. Because the platform itself is free, there is no additional payment on top of the AWS services we use. We can also simply match our server demands to our service load since we can choose our instance size as we used the t2 medium in our project and quickly add extra front-end servers to the load balancer. We can save money on operating expenditures by using a shared load balancer instead of a separate load balancer for each environment.

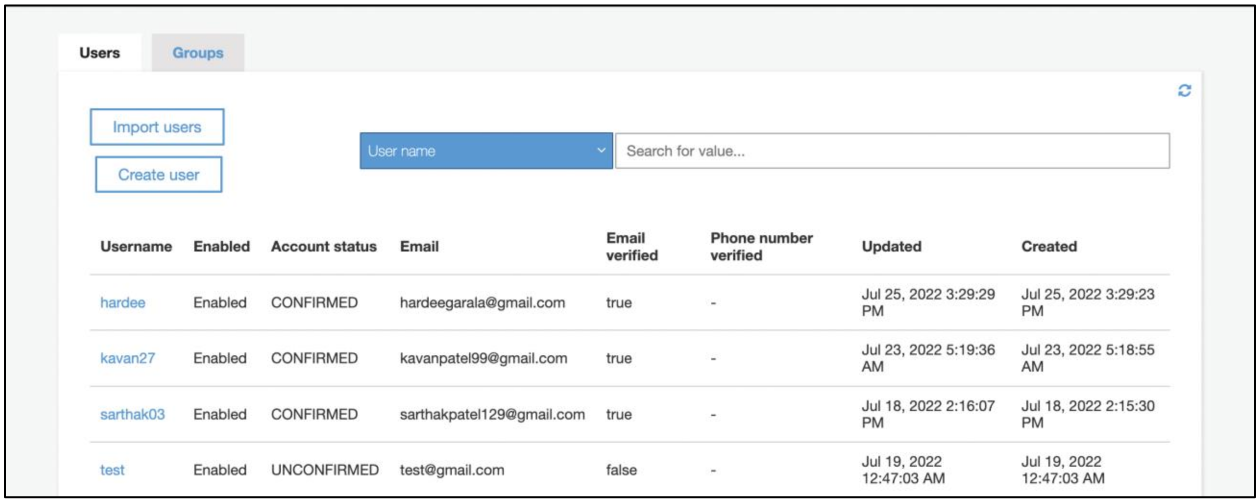
Elastic Beanstalk also includes built-in auto-scaling technology, which we never utilized but would be a great way to save money for larger applications by simply bringing up more servers when needed. Furthermore, Elastic Beanstalk's health monitoring functionality includes features that monitor statistics about our application and generate warnings when thresholds are surpassed.

Services Screenshots



Stack name	Status	Created time	Description
awseb-e-h58sveena4-stack	CREATE_COMPLETE	2022-07-13 17:59:08 UTC-0300	AWS Elastic Beanstalk environment (Name: 'WOC-PROD' Id: 'e-h58sveena4')
beanstalk1	CREATE_COMPLETE	2022-07-13 17:58:40 UTC-0300	-

Figure 3: Cloud formation stacks for backend and elastic beanstalk


Users Groups

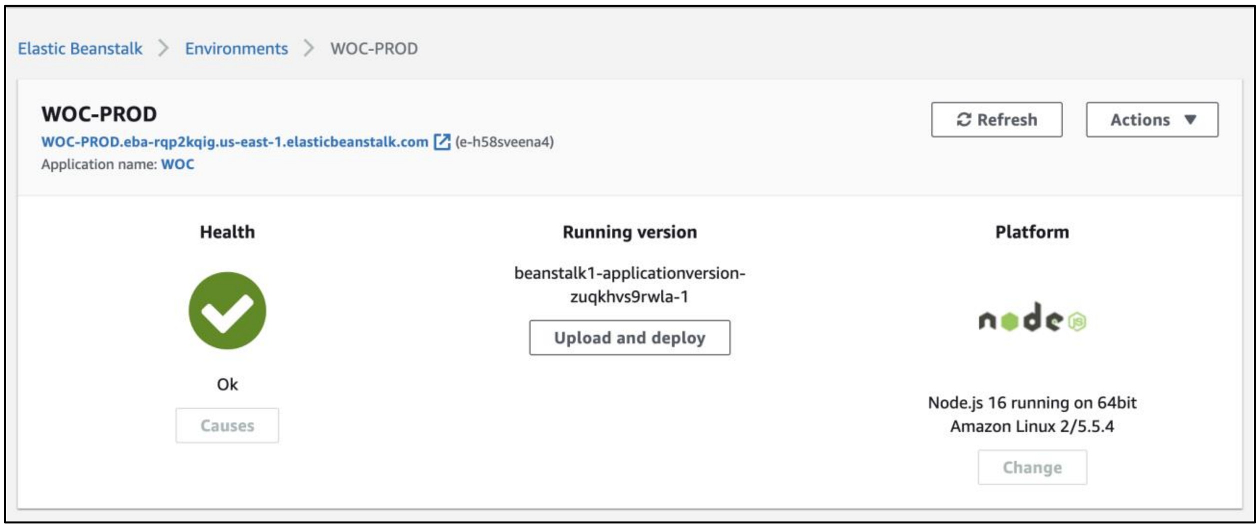
Import users

Create user

User name | Search for value...

Username	Enabled	Account status	Email	Email verified	Phone number verified	Updated	Created
hardee	Enabled	CONFIRMED	hardeegarala@gmail.com	true	-	Jul 25, 2022 3:29:29 PM	Jul 25, 2022 3:29:23 PM
kavan27	Enabled	CONFIRMED	kavanpatel99@gmail.com	true	-	Jul 23, 2022 5:19:36 AM	Jul 23, 2022 5:18:55 AM
sarthak03	Enabled	CONFIRMED	sarthakpatel129@gmail.com	true	-	Jul 18, 2022 2:16:07 PM	Jul 18, 2022 2:15:30 PM
test	Enabled	UNCONFIRMED	test@gmail.com	false	-	Jul 19, 2022 12:47:03 AM	Jul 19, 2022 12:47:03 AM

Figure 4: Cognito user pool


Elastic Beanstalk > Environments > WOC-PROD

WOC-PROD

WOC-PROD.eba-rqp2kqig.us-east-1.elasticbeanstalk.com (e-h58sveena4)

Application name: WOC

Refresh | Actions

Health

Ok

Causes

Running version

beanstalk1-applicationversion-zuqkhvs9rwla-1

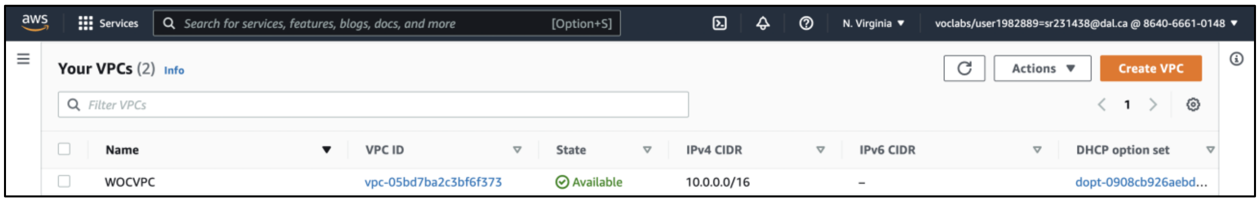
Upload and deploy

Platform

Node.js 16 running on 64bit Amazon Linux 2/5.5.4

Change

Figure 5: Frontend deployment on ElasticBeanstalk


aws
Services
Search for services, features, blogs, docs, and more [Option+S]
N. Virginia
voclabs/user1982889=sr231438@dal.ca @ 8640-6661-0148

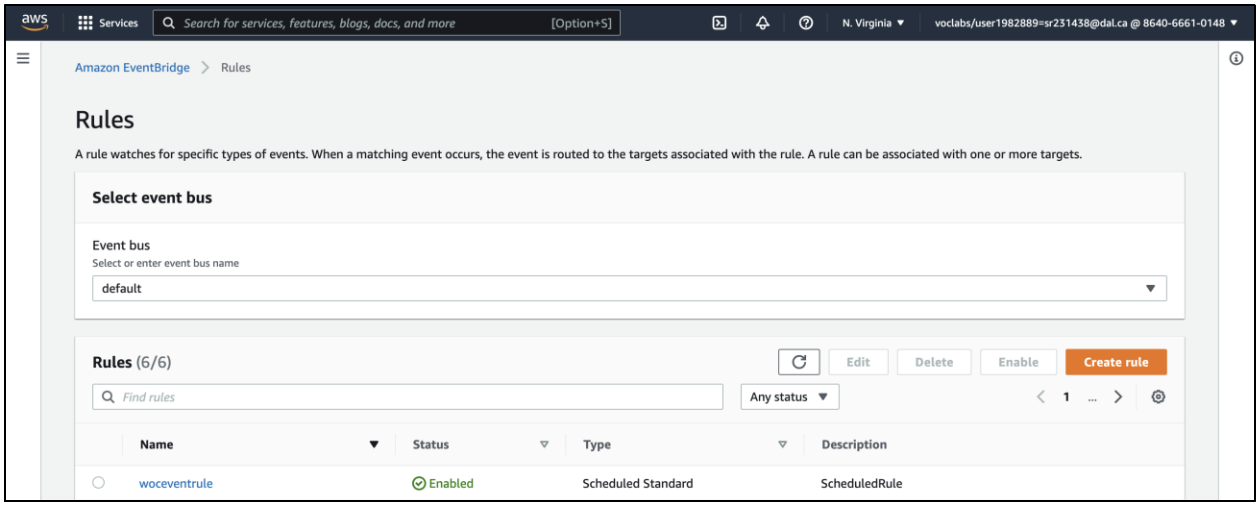
Your VPCs (2) Info

Actions | Create VPC

Filter VPCs

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set
WOCVPC	vpc-05bd7ba2c3bf6f373	Available	10.0.0.0/16	–	dopt-0908cb926aebd...

Figure 6: Virtual Private Cloud


Amazon EventBridge > Rules

Rules

A rule watches for specific types of events. When a matching event occurs, the event is routed to the targets associated with the rule. A rule can be associated with one or more targets.

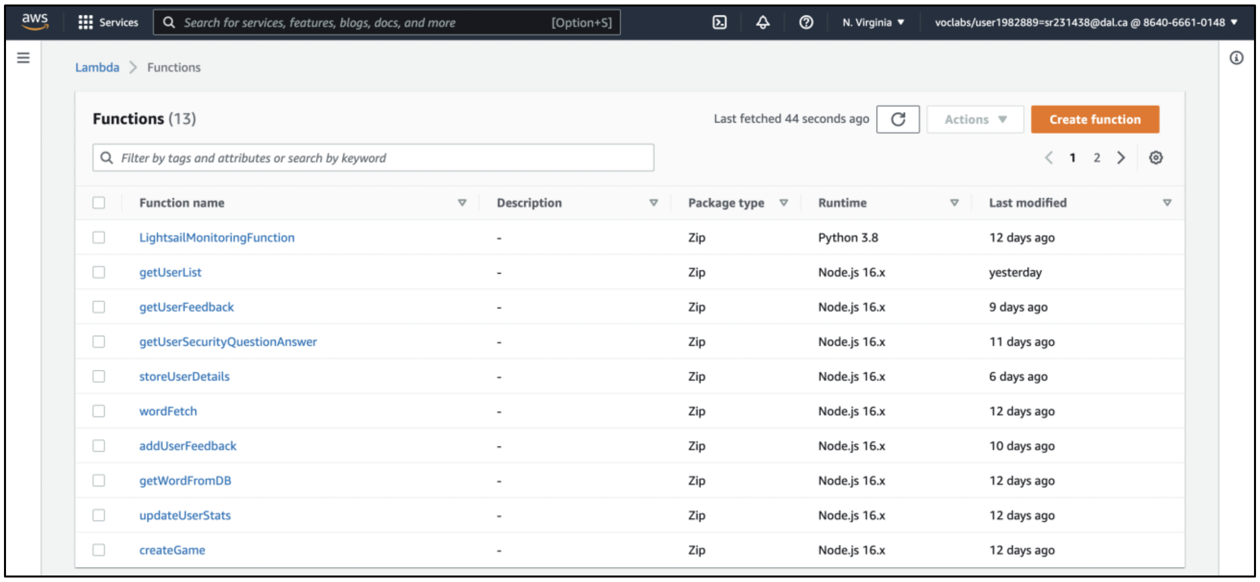
Select event bus

Event bus
Select or enter event bus name
default

Rules (6/6)

Name	Status	Type	Description
woceventrule	Enabled	Scheduled Standard	ScheduledRule

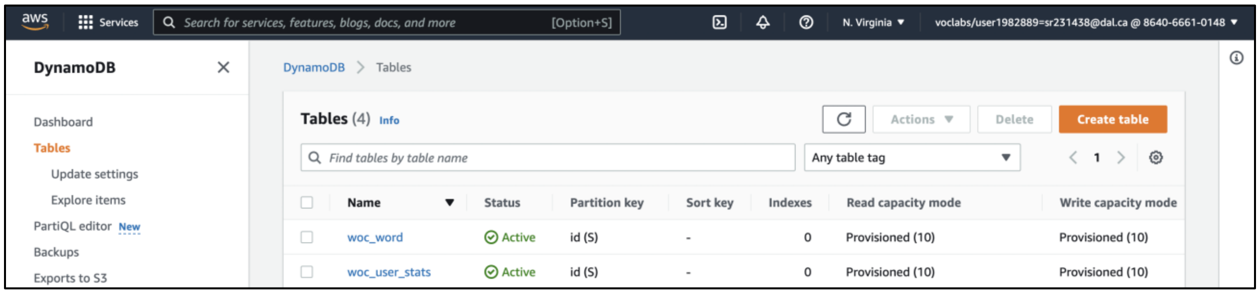
Figure 7: Eventbridge rule


Lambda > Functions

Functions (13) Last fetched 44 seconds ago

Function name	Description	Package type	Runtime	Last modified
LightsailMonitoringFunction	-	Zip	Python 3.8	12 days ago
getUserList	-	Zip	Node.js 16.x	yesterday
getUserFeedback	-	Zip	Node.js 16.x	9 days ago
getUserSecurityQuestionAnswer	-	Zip	Node.js 16.x	11 days ago
storeUserDetails	-	Zip	Node.js 16.x	6 days ago
wordFetch	-	Zip	Node.js 16.x	12 days ago
addUserFeedback	-	Zip	Node.js 16.x	10 days ago
getWordFromDB	-	Zip	Node.js 16.x	12 days ago
updateUserStats	-	Zip	Node.js 16.x	12 days ago
createGame	-	Zip	Node.js 16.x	12 days ago

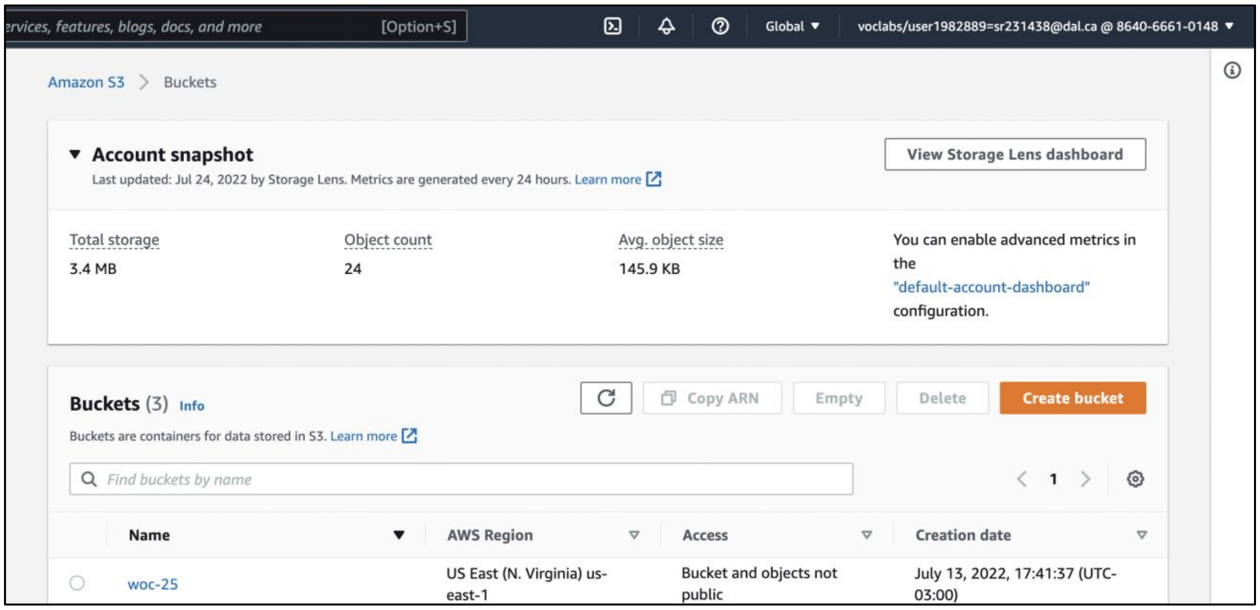
Figure 8: Lambda Functions


DynamoDB > Tables

Tables (4)

Name	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode
woc_word	Active	id (S)	-	0	Provisioned (10)	Provisioned (10)
woc_user_stats	Active	id (S)	-	0	Provisioned (10)	Provisioned (10)

Figure 9: DynamoDB tables


ervices, features, blogs, docs, and more [Option+S]
Global
voclabs/user1982889=sr231438@dal.ca @ 8640-6661-0148
Amazon S3 > Buckets
View Storage Lens dashboard
Copy ARN
Empty
Delete
Create bucket
Find buckets by name

Account snapshot

Last updated: Jul 24, 2022 by Storage Lens. Metrics are generated every 24 hours. Learn more

Total storage	Object count	Avg. object size	
3.4 MB	24	145.9 KB	You can enable advanced metrics in the "default-account-dashboard" configuration.

Buckets (3) Info

Buckets are containers for data stored in S3. Learn more

Name	AWS Region	Access	Creation date
woc-25	US East (N. Virginia) us-east-1	Bucket and objects not public	July 13, 2022, 17:41:37 (UTC-03:00)

Figure 10: s3 bucket for storing dictionary

Application Screenshots

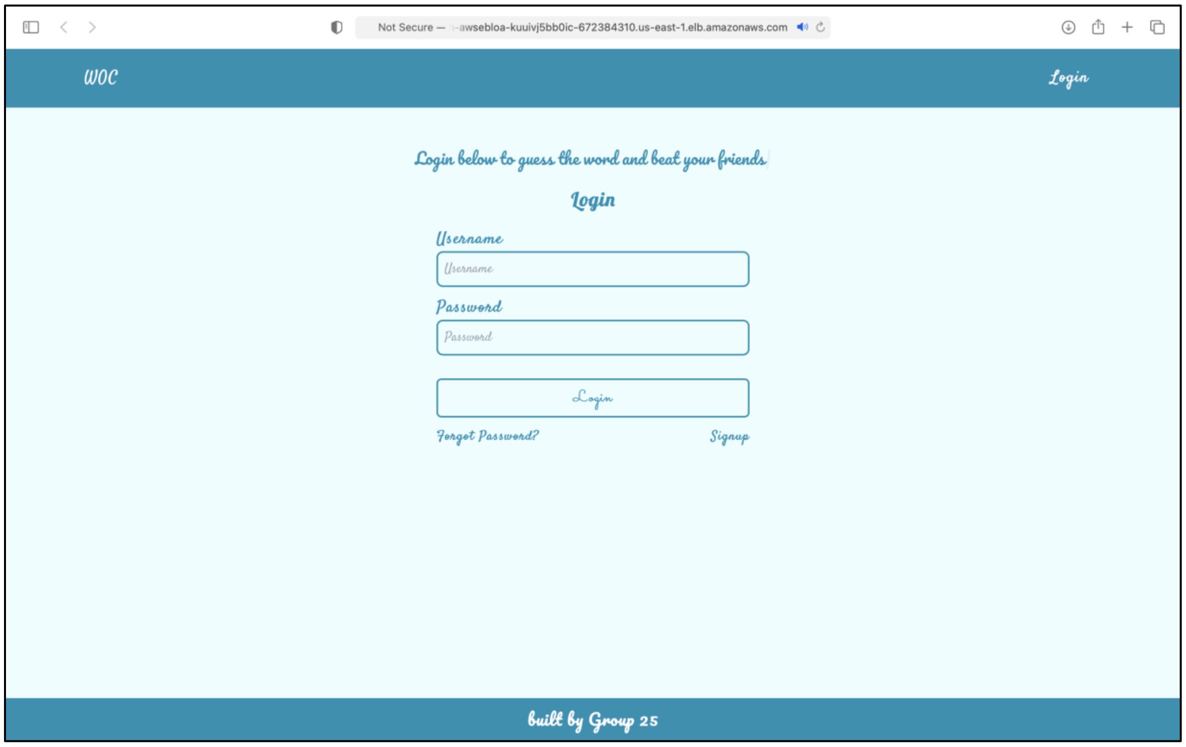

Not Secure — awsebloa-kuuivj5bb0ic-672384310.us-east-1.elb.amazonaws.com
WOC
Login
Login below to guess the word and beat your friends!
Login
Username
Username
Password
Password
Login
Forgot Password?
Signup
built by Group 25

Figure 11: Login Page

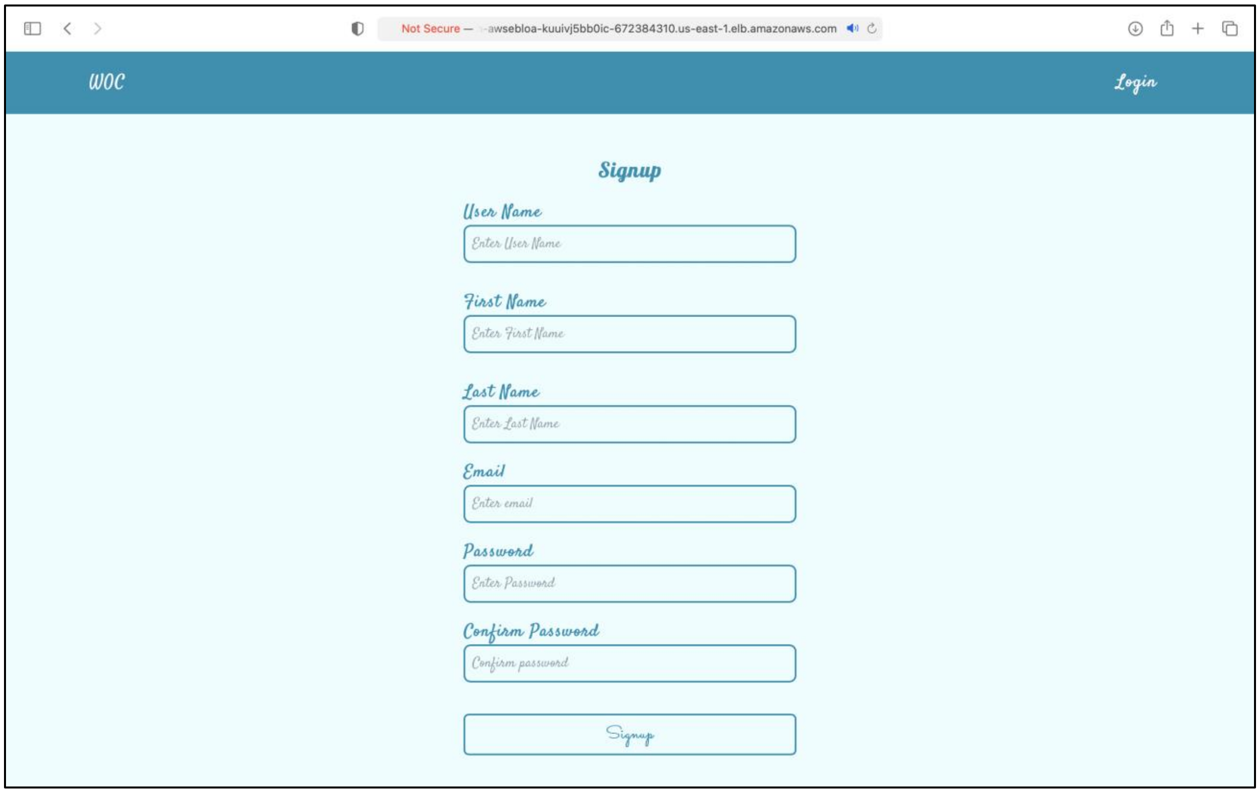


Figure 12: sign up page

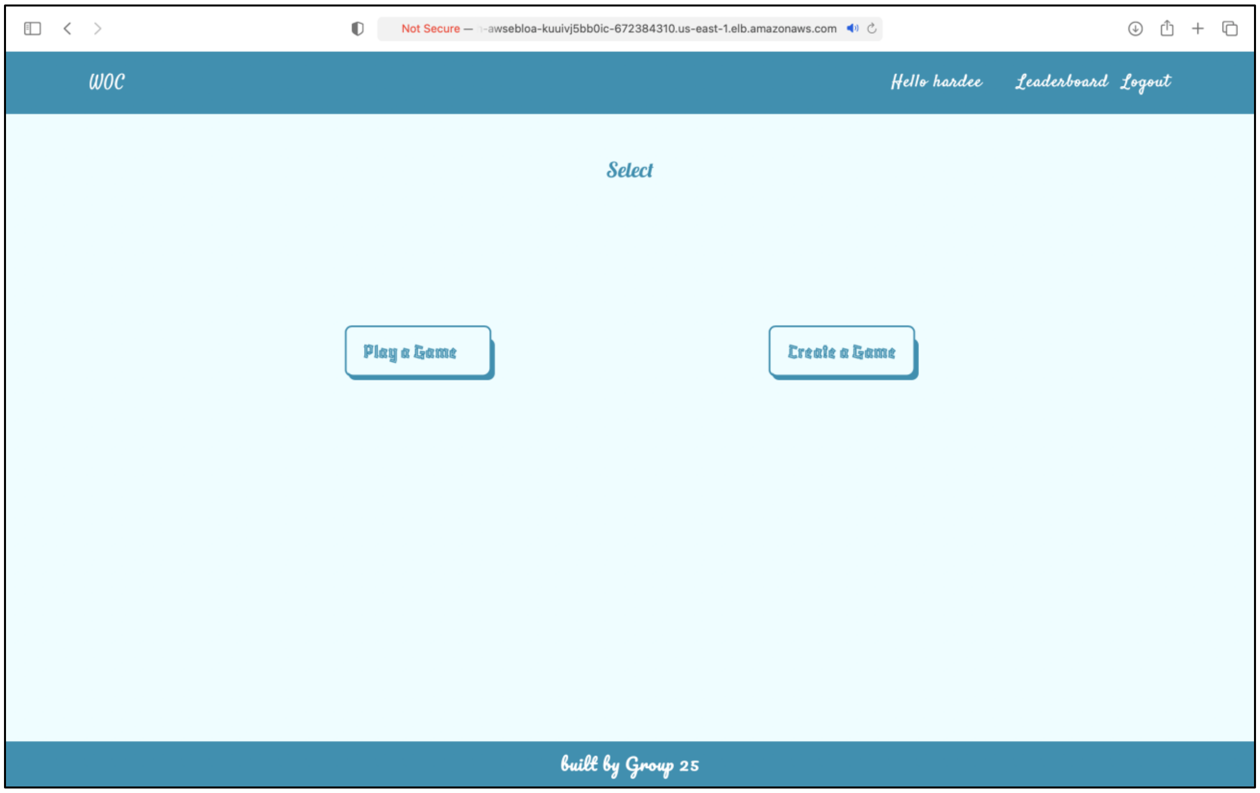
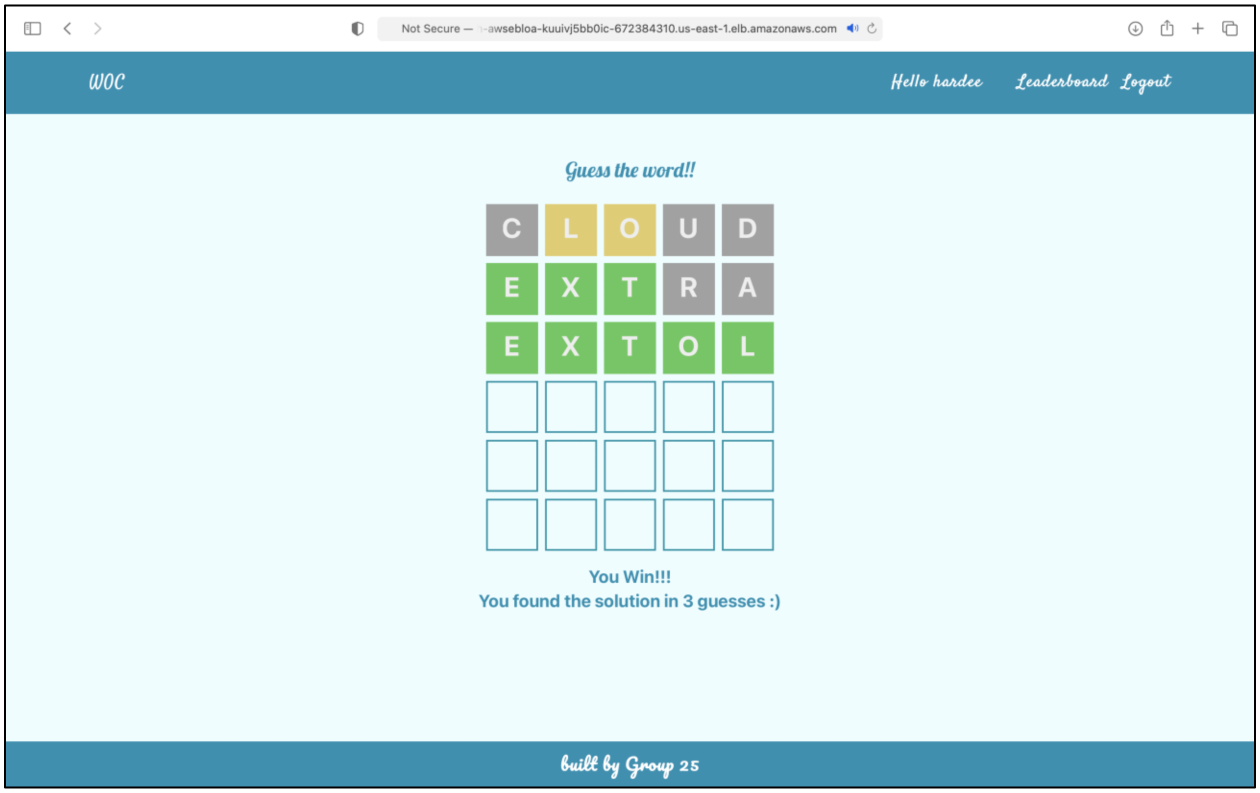
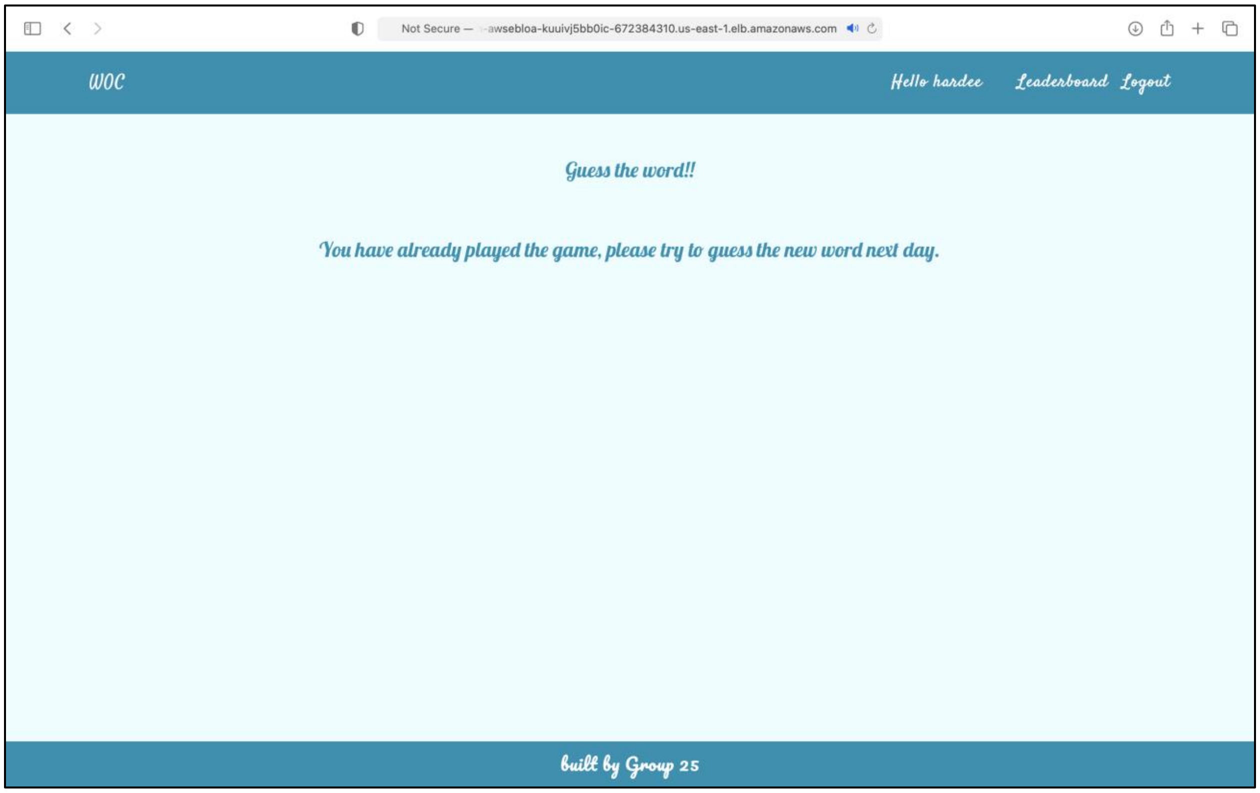


Figure 13: home page



You Win!!!
You found the solution in 3 guesses :)

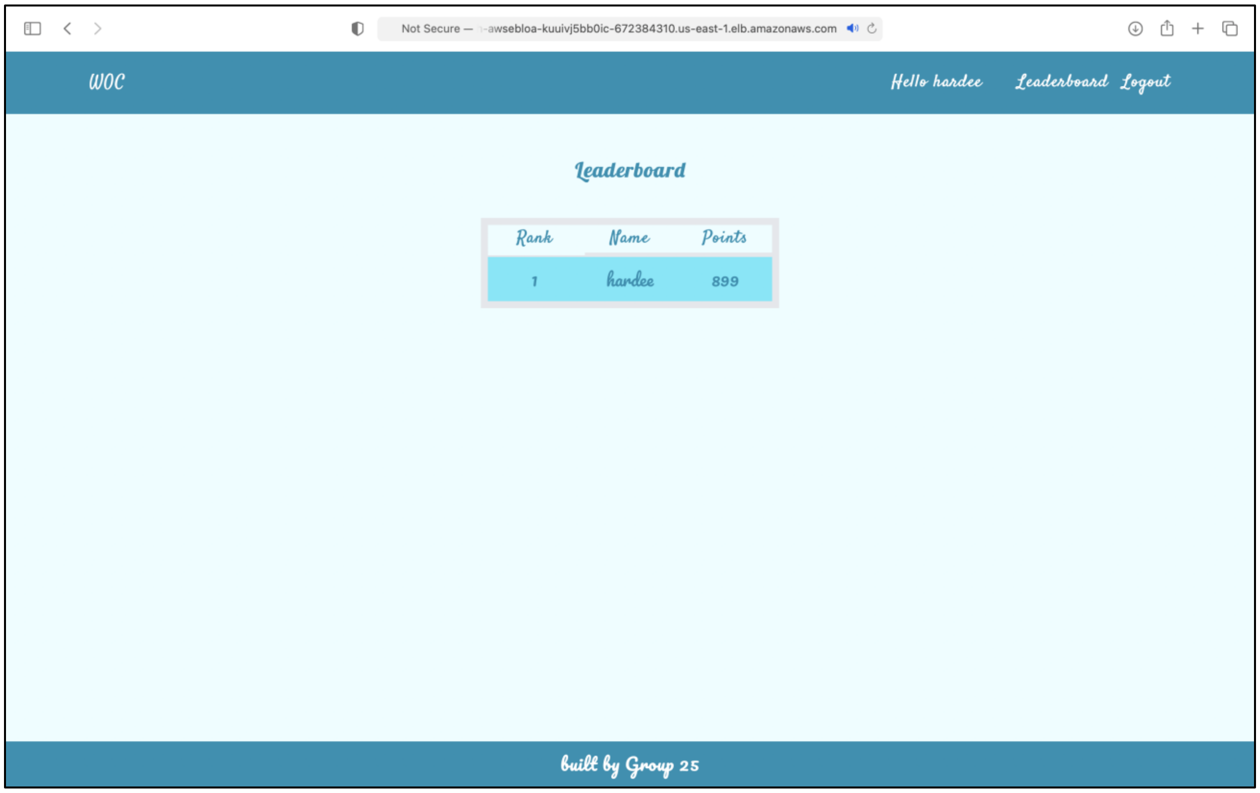
Figure 14: play game feature: if playing game for the first time in a day


Not Secure — awsebloa-kuuivj5bb0ic-672384310.us-east-1.elb.amazonaws.com
WOC
Hello hardee
Leaderboard
Logout
built by Group 25

Guess the word!!

You have already played the game, please try to guess the new word next day.

Figure 15: if the game is already played in that day


Not Secure — -awsebloa-kuuivj5bb0ic-672384310.us-east-1.elb.amazonaws.com
WOC
Hello hardee
Leaderboard
Logout
built by Group 25

Leaderboard

Rank	Name	Points
1	hardee	899

Figure 16: Leaderboard

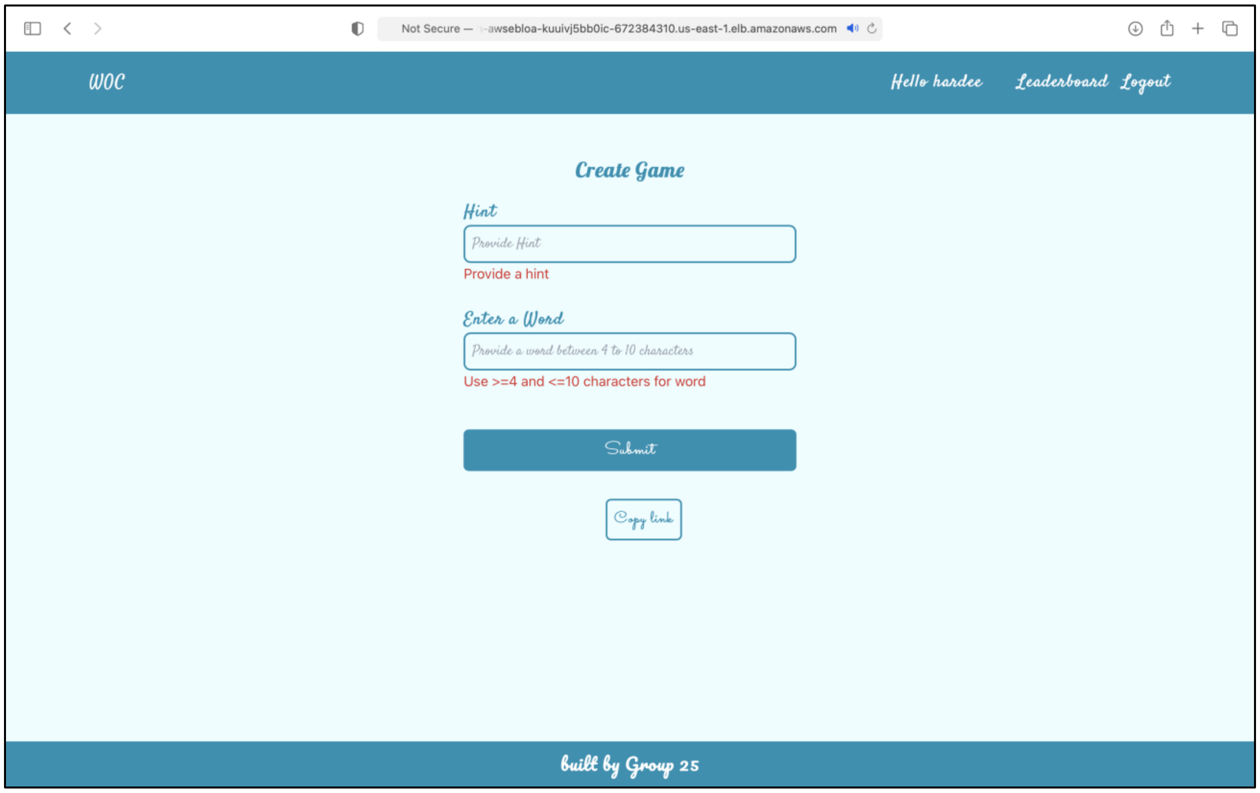

Not Secure — -awsebloa-kuuivj5bb0ic-672384310.us-east-1.elb.amazonaws.com
WOC
Hello hardee
Leaderboard
Logout
Create Game
Hint
Provide Hint
Provide a hint
Enter a Word
Provide a word between 4 to 10 characters
Use >=4 and <=10 characters for word
Submit
Copy link
built by Group 25

Figure 17: create game page with validations

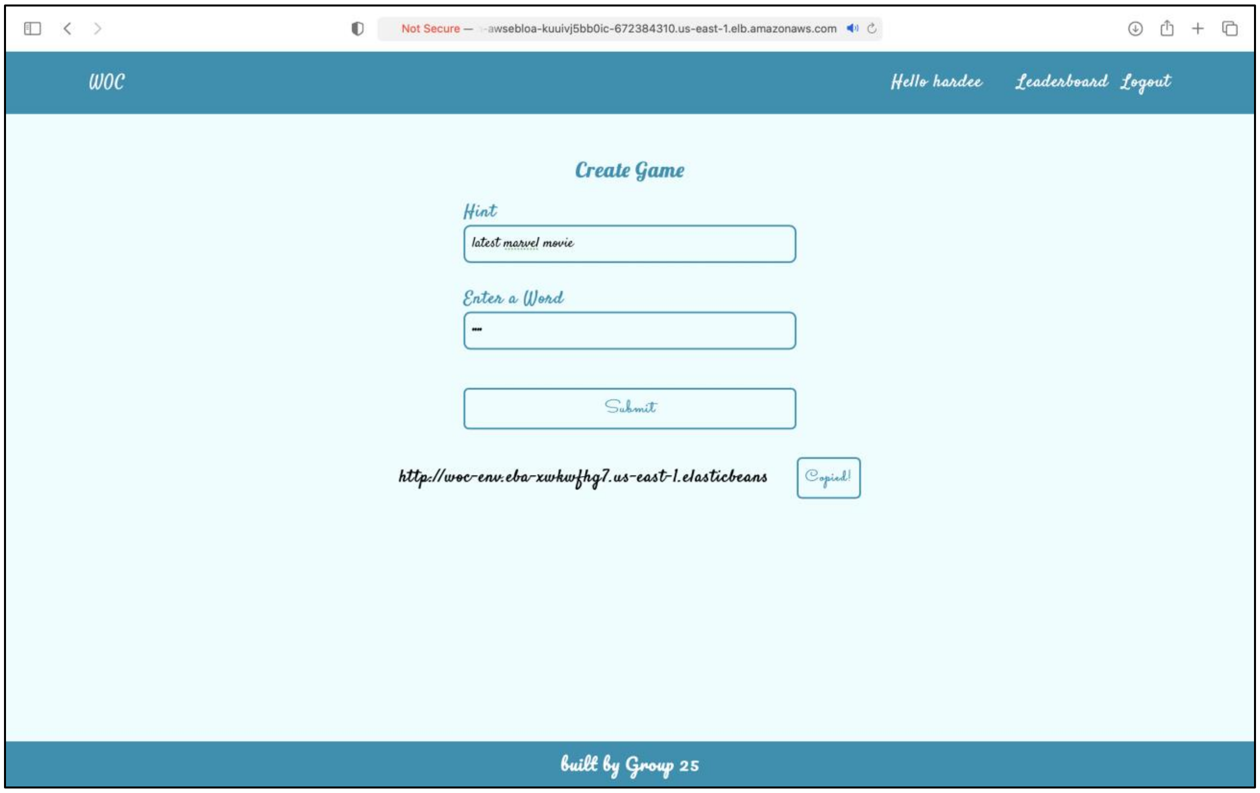

Not Secure — awsebloa-kuuivj5bb0ic-672384310.us-east-1.elb.amazonaws.com
WOC
Hello hardee
Leaderboard
Logout
Create Game
Hint
latest marvel movie
Enter a Word
Submit
http://woc-env.eba-xwkwfhg7.us-east-1.elasticbeans
Copied!
built by Group 25

Figure 18:Generate game feature for create game feature

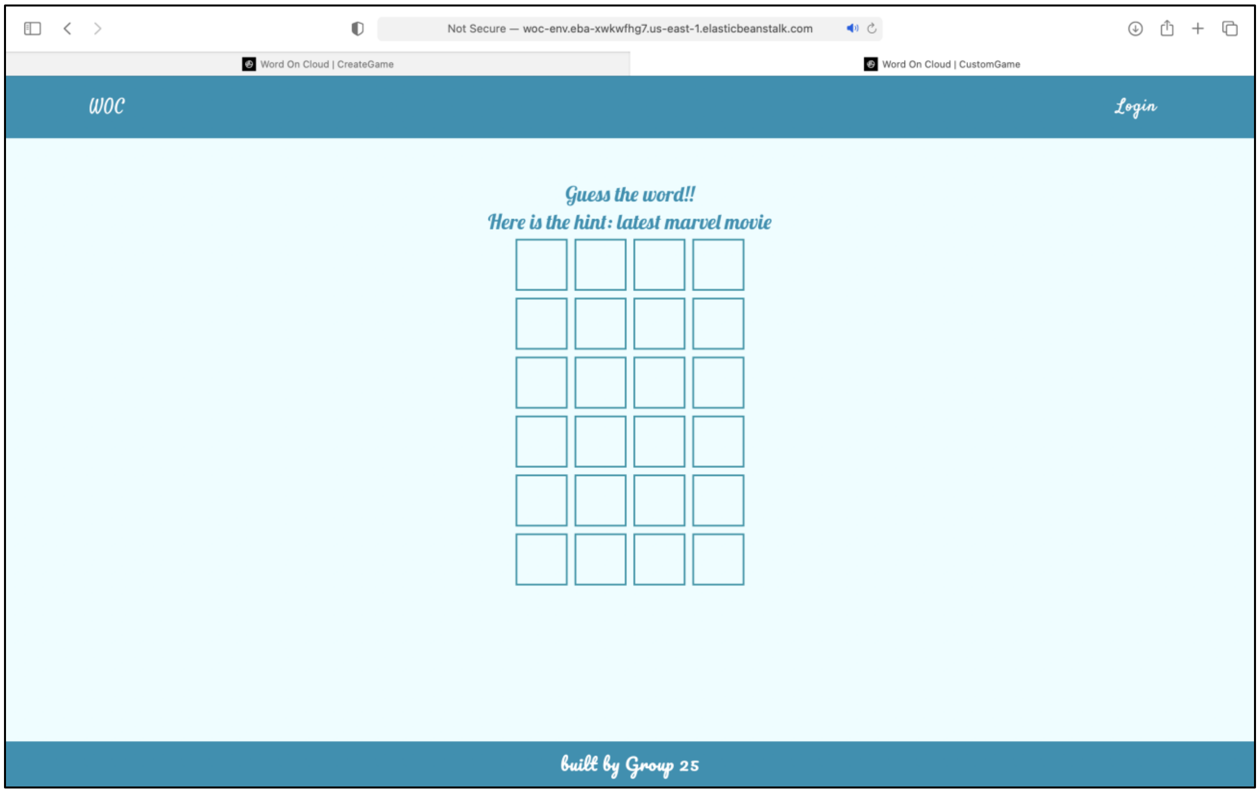
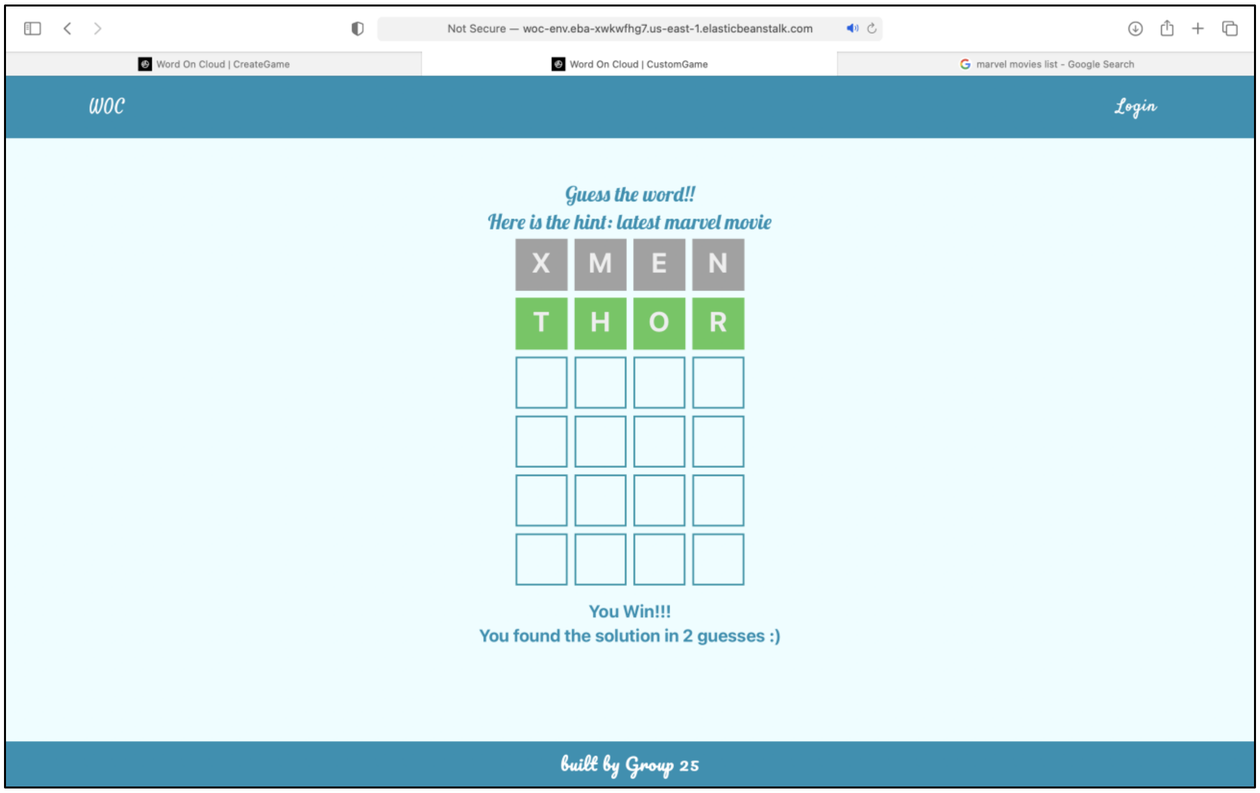

Not Secure — woc-env.eba-xwkwfhg7.us-east-1.elasticbeanstalk.com
Word On Cloud | CreateGame
Word On Cloud | CustomGame
WOC
Login
Guess the word!!
Here is the hint: latest marvel movie
built by Group 25

Figure 19: Accessing the link of create game


Not Secure — woc-env.eba-xwkwfhg7.us-east-1.elasticbeanstalk.com
Word On Cloud | CreateGame
Word On Cloud | CustomGame
marvel movies list - Google Search
WOC
Login
X M E N
T H O R
built by Group 25

Guess the word!!
Here is the hint: latest marvel movie

You Win!!!
You found the solution in 2 guesses :)

Figure 20: Playing the custom created game

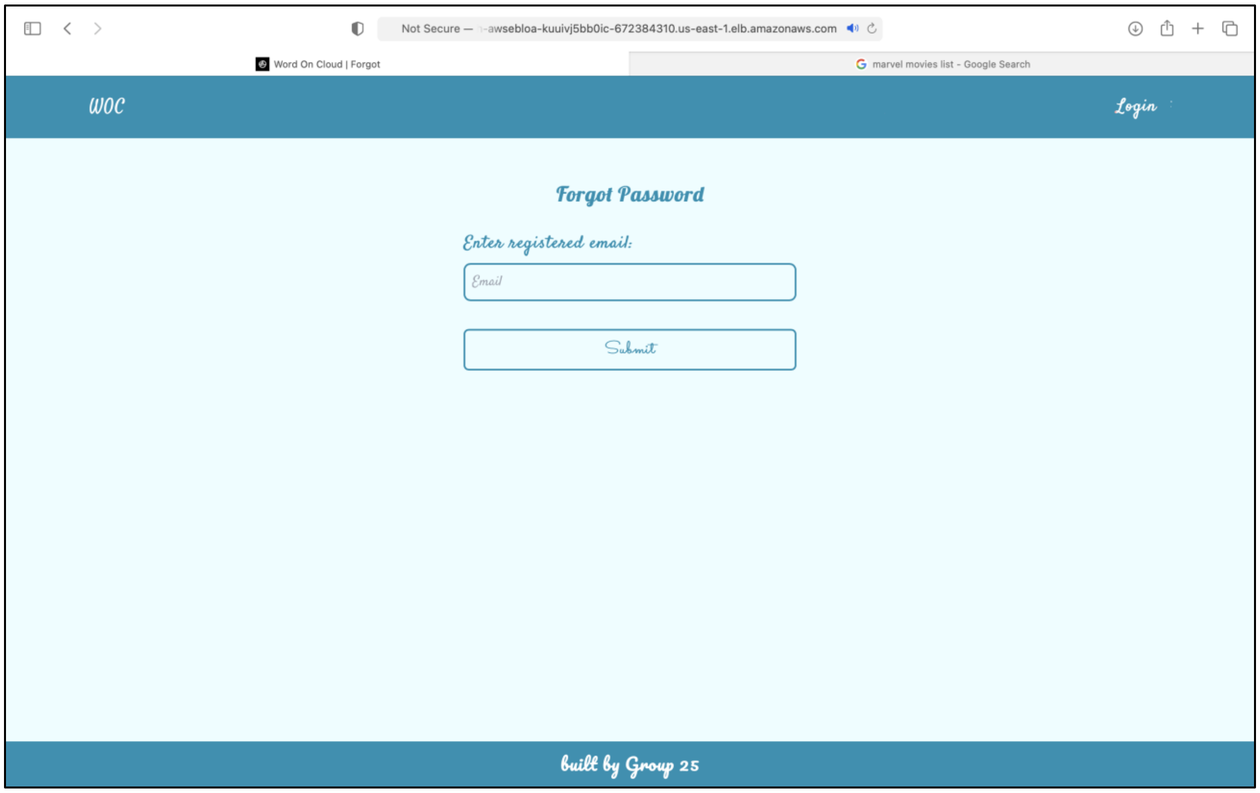
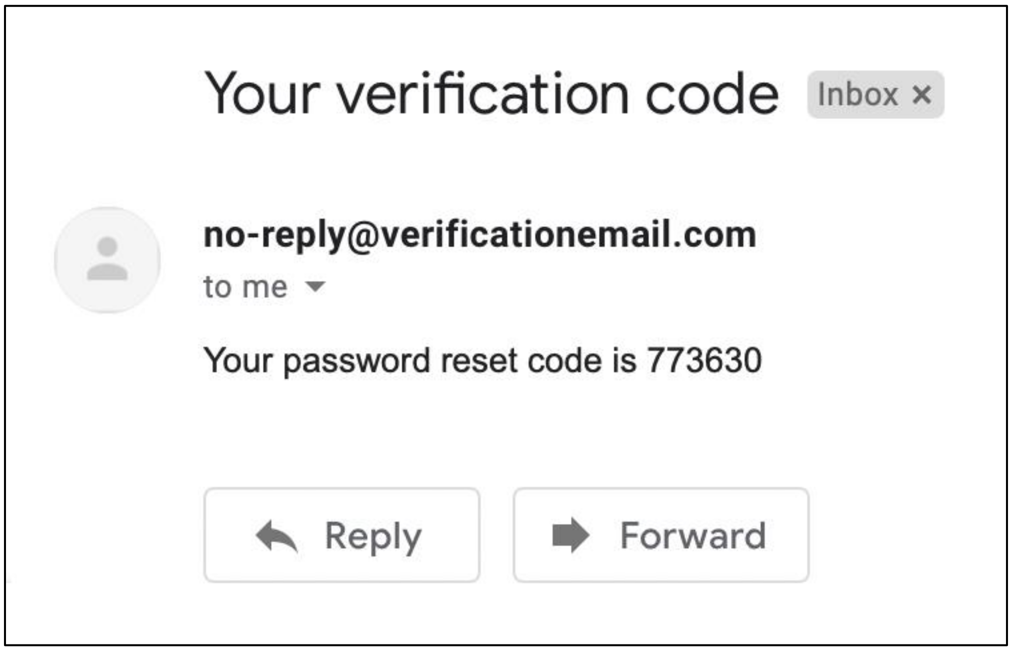

Not Secure — -awsebloa-kuuivj5bb0ic-672384310.us-east-1.elb.amazonaws.com
Word On Cloud | Forgot
marvel movies list - Google Search
WOC
Login
Forgot Password
Enter registered email:
Email
Submit
built by Group 25

Figure 21: Forgot password page



Your verification code

no-reply@verificationemail.com
to me

Your password reset code is 773630

Figure 22: email for password reset verification code

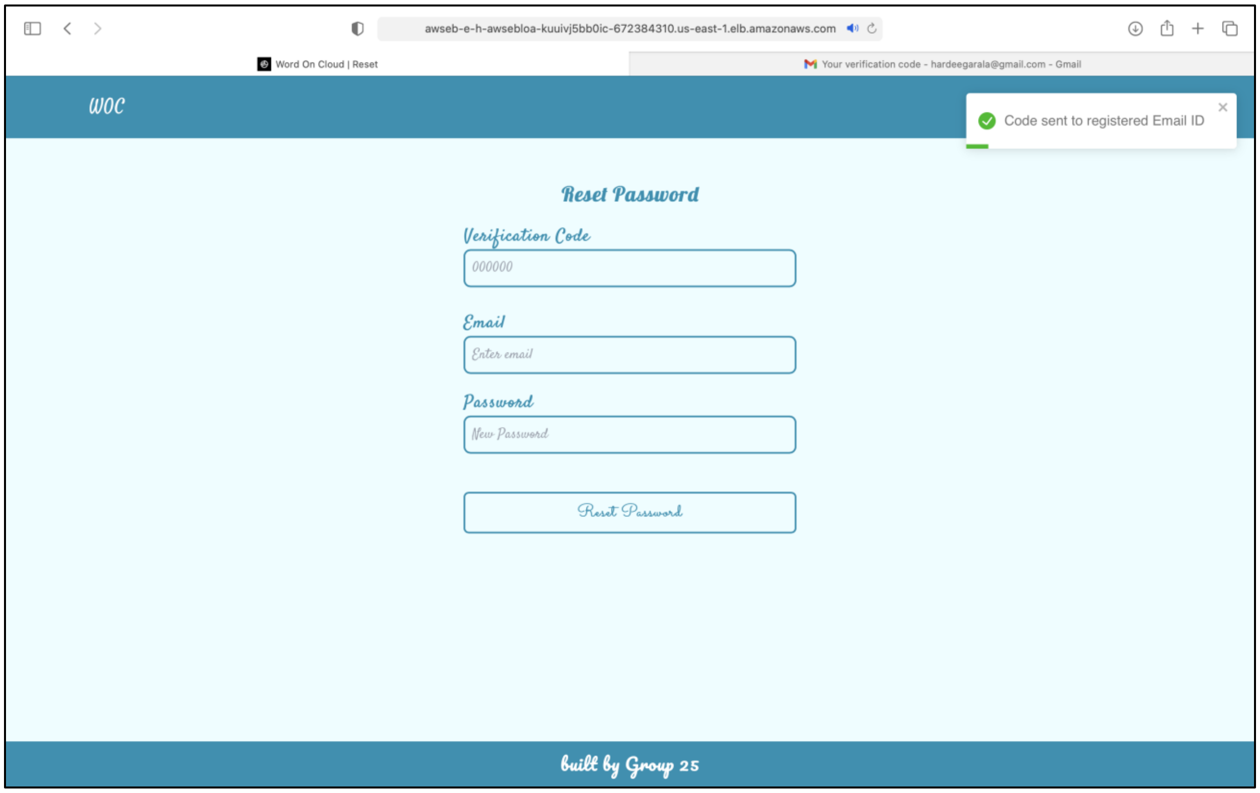

awseb-e-h-awsebloa-kuuivj5bb0ic-672384310.us-east-1.elb.amazonaws.com
Word On Cloud | Reset
Your verification code - hardeegarala@gmail.com - Gmail
WOC
Code sent to registered Email ID
Reset Password
Verification Code
000000
Email
Enter email
Password
New Password
Reset Password
built by Group 25

Figure 23: reset password page

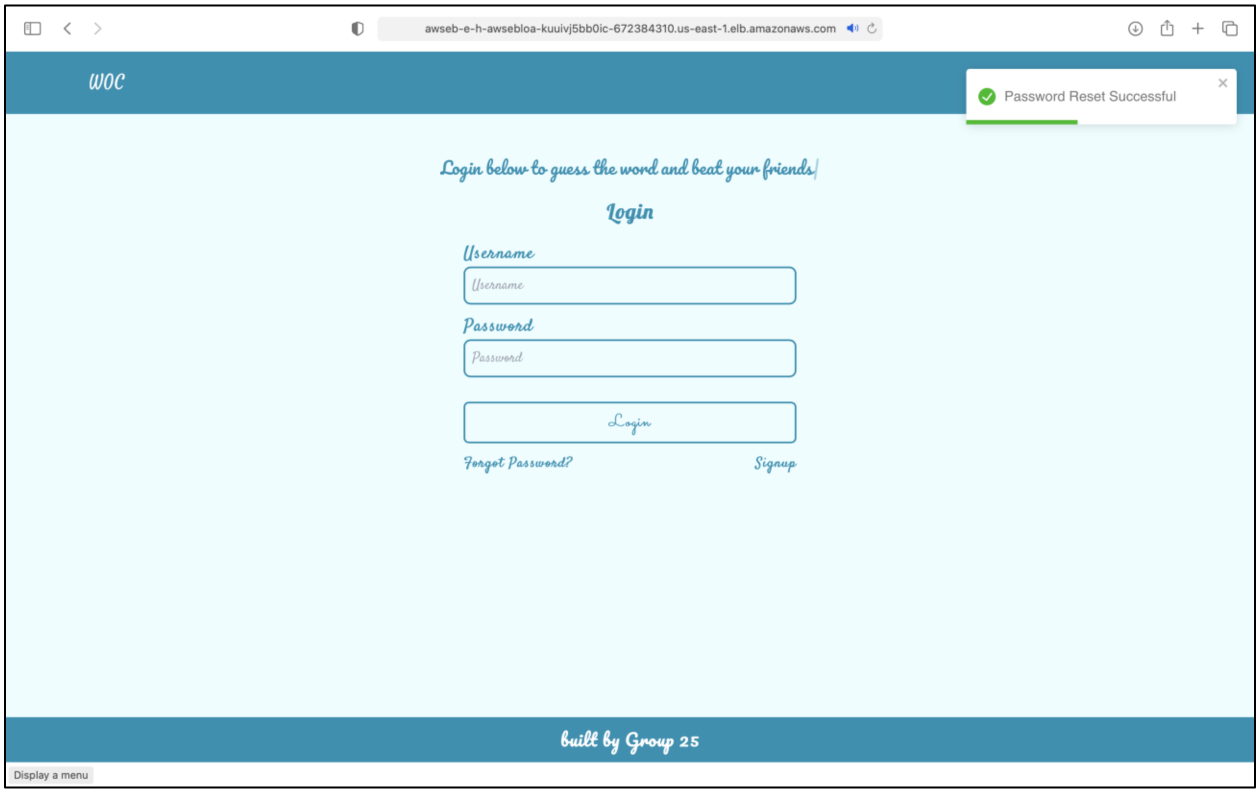

awseb-e-h-awsebloa-kuuivj5bb0ic-672384310.us-east-1.elb.amazonaws.com
WOC
Password Reset Successful
Login below to guess the word and beat your friends!
Login
Username
Password
Login
Forgot Password?
Signup
built by Group 25
Display a menu

Figure 24: password reset confirmation and redirection to login page

References:

[1] *Amazon Web Services, Inc. or its affiliates*, "Infrastructure security in Amazon VPC - Amazon Virtual Private Cloud," [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/infrastructure-security.html.

[2] A. Deshpande, "7 Advantages of ReactJS for Building Interactive User Interfaces", *Clariontech.com, 2022*. [Online]. Available: https://www.clariontech.com/blog/7-advantages-of-reactjs-for-building-interactive-user-interfaces. [Accessed: 25- Jul- 2022].

[3] E. Pulsifer and J. Short, "Building more cost-effective Lambda functions with 1 ms billing", *A Cloud Guru, 2022*. [Online]. Available: https://acloudguru.com/blog/engineering/building-more-cost-effective-lambda-functions-with-1-ms-billing. [Accessed: 25- Jul- 2022].

[4] "5 Ways to reduce data storage costs using Amazon S3 Storage Lens | Amazon Web Services", *Amazon Web Services, 2022*. [Online]. Available: https://aws.amazon.com/blogs/storage/5-ways-to-reduce-costs-using-amazon-s3-storage-lens/. [Accessed: 25- Jul- 2022].