

Quiz Master: Project Report

Group name: NULL Pointers

Group:

Parth Patel

Dev Patel

Shailen Sutradhar

Gautam Santhanu Thampy

Due Date: May 12th, 2025

Class: CMPE272 Sec 03

Prof: Andrew Bond

TABLE OF CONTENTS

TABLE OF CONTENTS.....	2
Introduction.....	4
Purpose	4
Intended Audience.....	4
Technical stack	4
Architecture & High-Level Design	6
Definitions and acronyms ¹	6
Project risks.....	7
Development process.....	7
Milestones & Time schedule (Agile Methodology)	8
Demo.....	10
Arriving at the home page	10
Logging in using email	10
Create Quiz.....	13
Attempting Quiz	14
My Quizzes tab.....	16
Technical Walkthrough	17
Code Repository	17
Testing using Vitest	17
AWS API Gateway.....	18
AWS S3	19
AWS Lambda Functions	19
AWS DynamoDB	22
Firebase Authentication	23
Deployment and CI/CD (Dev).....	23
Cloud Deployment	23
CI/CD (Continuous Integration Continuous Deployment):	25

Organization	28
Project group	28
Communication.....	29
GitHub: github.com/ParthPatel00/QuizMaster	29
Website: https://quizmaster.dedyn.io/	29
Citations.....	29

Introduction

Purpose

QuizMaster is designed to be an application that enables users to assess their knowledge and proficiency on a specific topic of their choice. Its core functionality is to generate quizzes based on user-provided content.

Users can upload a PDF document containing content for which they would like to test their knowledge in. QuizMaster will generate a quiz tailored to this content and provide it to the user in a simple, user-friendly interface. Upon completion, users can receive instant grading, feedback on correct and incorrect answers, and the ability to retake the quiz as many times as they like.

Furthermore, when logged in/signed up, QuizMaster will also allow users to save quizzes for future retakes.

Intended Audience

The QuizMaster tool is designed to empower the executives and students across schools, universities, enterprises and a portion of general audience.

- Students can upload lecture slides and textbooks to generate tests helping them prepare for exams and be able to test their knowledge on a topic.
- Enterprises can integrate the tool in their employee recruitment, training and assessment.
- Universities can use QuizMaster to automate composing examination questions and reducing the risk of cheating with the ability to generate unique questions for each test.
- General audience can utilize the tool to generate quizzes from books, documents and research papers.

Technical stack

Frontend: React, Typescript, Tailwindcss

Backend: AWS Lambda, API Gateway

File storage: Amazon S3

Database: AWS DynamoDB

LLM: Google Gemini

Authorization: Firebase

CI/CD: GitHub Actions, Code Build

Version control: GitHub, Git

Reasoning for selected tech stack:

React:

- Component based code for front-end makes it more readable and code is reusable for future uses
- Responsive and faster to load
- Better documentation and resources to create a user-friendly interface

AWS S3 Bucket

- For file storage of PDFs

AWS DynamoDB

- Used as a document store of JSON files received from Gemini

AWS Lambda:

- Affordable cost due to its serverless function and only queries made to the function are charged
- Easier integration with other AWS services
- Faster

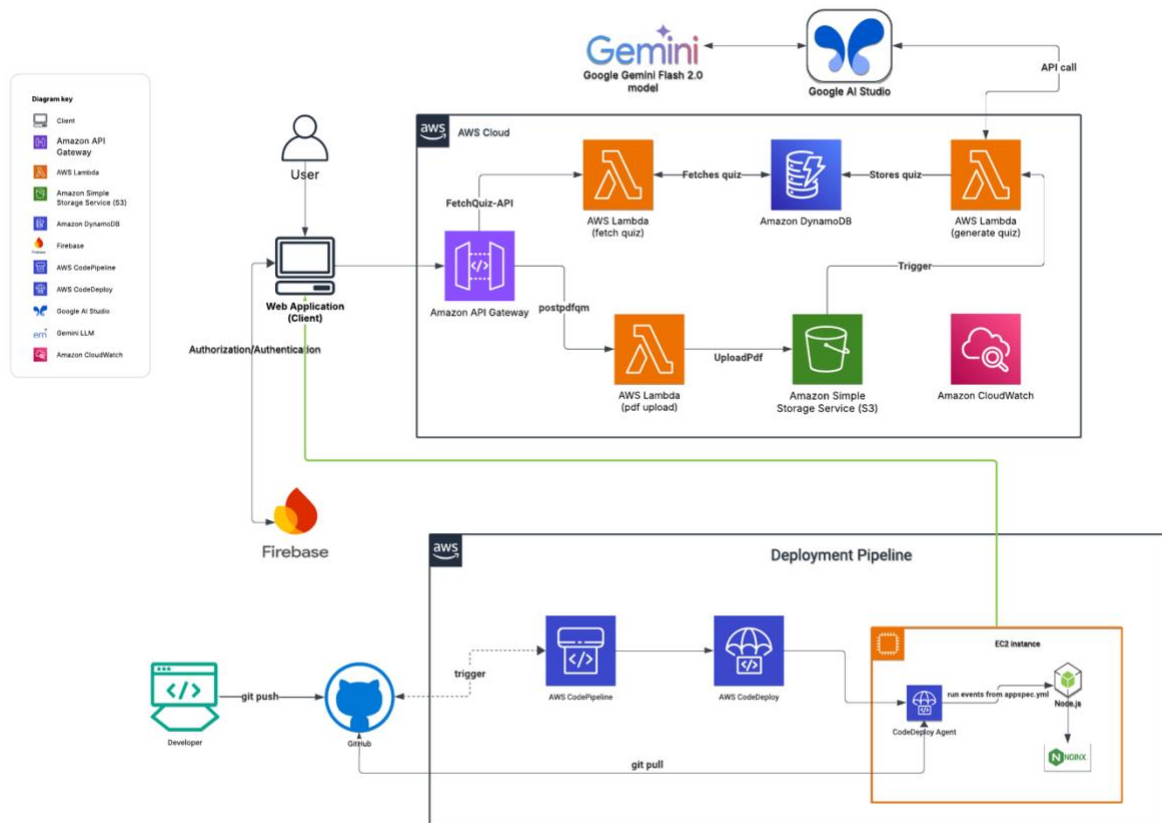
AWS API Gateway:

- Secure deployment of Rest APIs
- Scalable to handle large number traffic loads
- Seamless integration with AWS Lambda

Code Build:

- Offers free 100 build minutes/month
- Allows automated software release through multiple environments

Architecture & High-Level Design



Link to [Lucid Chart](#)

Definitions and acronyms¹

AWS: Amazon Web Services. A cloud computing platform that offers a variety of services, including storage, computing and content delivery.

EC2: Amazon Elastic Compute Cloud. A web service that let users run application in the Amazon Web Services cloud.

S3: Amazon Simple Storage Service. A cloud-based object storage service that allows users to store and retrieve data.

LLM: Large Language Model. A type of artificial intelligence that can understand and generate human language.

API Gateway: Application Programming Interface Gateway. A software tool that manages API calls between clients and backend services.

DynamoDB: Amazon DynamoDB. A cloud-based database service.

CI/CD: Continuous Integration and Continuous Deployment/Delivery. A software development practice that uses automation to speed up and improve the software development process.

Git: Global Information Tracker. A free, open-source version control system.

Project risks

Accessibility: The application must be accessible by the users all time. For server or some other failure, the user might not be able to access the website at all. Besides, the saved data might be inaccessible for the user.

Responsiveness: The webpage might not be responsive across devices with different screen sizes which might lead to broken layout, unreadable texts and a poor user interface.

File Handling: The application will not function if the user uploads a file too big for the LLM or a file of inappropriate type.

Performance: It might take long for the quiz to be generated which will lead to inefficiency. Sometimes, failure to generate quizzes can also happen.

Security: The authentication can fail, which will lead to unauthorized access. There can be breaches in stored quiz data leading to privacy and security concerns.

Incorrect or Unethical Quiz: The AI model might generate unrealistic or inappropriate questions and incorrect answers. It might also generate questions out of the given material.

Scalability: A high number of users trying to access the webpage can slow down the application and cause it to crash.

Development process

1. Requirements gathering
 - a. Coming up with features
 - b. Designing the website
 - c. Choosing an LLM
2. Creating the frontend
 - a. Creating a home page
 - b. Creating a navbar
 - c. Creating a quiz page
 - d. Creating the My Quizzes page

- e. Creating authorization
 - f. Integrating frontend buttons with AWS Gateway endpoints.
 - g. Testing
- 3. Creating an organization on AWS for team collaboration
- 4. Creating storage (S3 and DynamoDB)
- 5. Setting up API gateway and Lambda functions
 - a. Setting up
 - b. Implementing
 - c. Testing
- 6. Setting up Gemini account
- 7. Enabling Code Deploy and EC2 for deployment
 - a. Provisioning
 - b. Testing

Milestones & Time schedule (Agile Methodology)

Tasks to Accomplish	Total weeks	Complete by
Requirements gathering	1	Feb 10 th
Research which LLM model to use, and setup an account with it (ChatGPT/Gemini/Deep Seek etc.)		
Creating an AWS organization account for collaboration		
Develop a functional home page.	2	Feb 24 th
Creating an S3 bucket		
Setting up a Dynamo database		
Milestone #1 Team meeting + Testing + Next steps		
Create the Quiz frontend component	2	March 10 th
Implement the pdfUploadToS3 Lambda function		
Create NoSQL mock data for DynamoDB		

Test connection to AWS lambda		
Test trigger events to S3		
Test responsiveness of frontend		
Milestone #2 Team meeting + Testing + Next steps		
Setup Firebase and backend for authorization		
Do prompt engineering with LLM of choice.	1	March 17 th
Test authorization with Gmail, and new email		
Test example PDFs with prompts		
Milestone #3 Team meeting + Testing + Next steps		
Implement the Lambda function responsible for populating DynamoDB		
Implement backend for fetching data from DynamoDB	2	March 31 st
Test populate dynamo lambda function		
Test fetching data from dynamo lambda function		
Milestone #4 Team meeting + Testing + Next steps		
Provisioning EC2 and Code Build for deployment	2	April 14 th
End-to-end testing		
Milestone #5 Team meeting + Final Testing + Presentation planning		
Creating a presentation	Remaining time: ~ 1 Month	May 12 th
Assigning slides to each member		

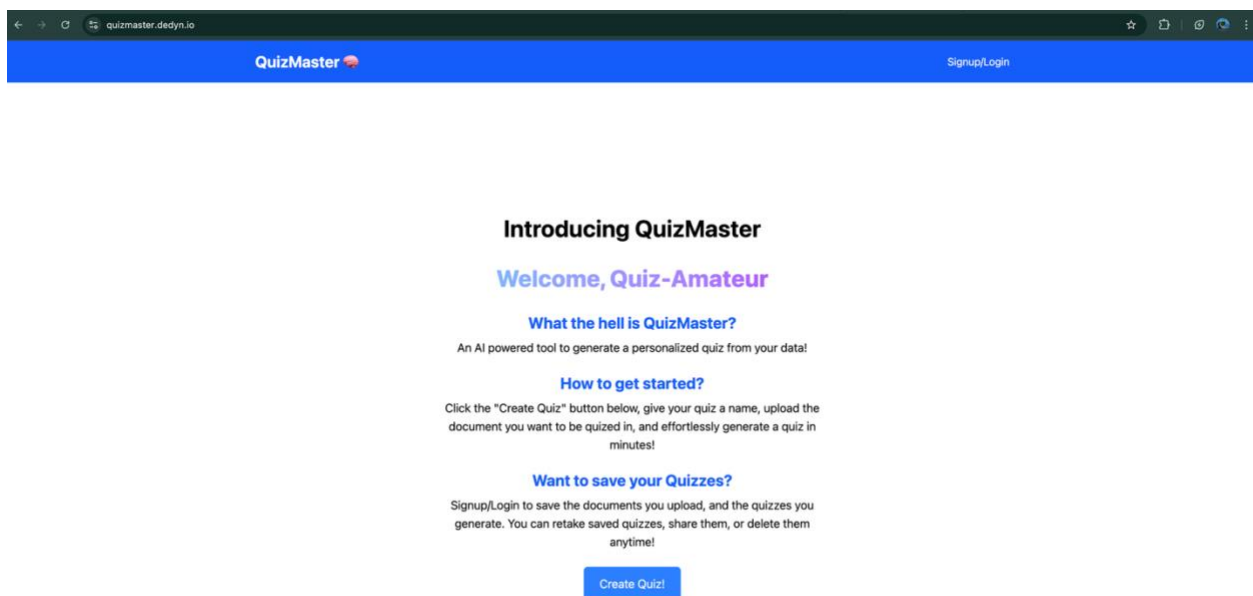
Practice presenting		
PRESENT PROJECT IN CLASS		

Demo

Arriving at the home page

The application starts at the QuizMaster homepage (<https://quizmaster.dedyn.io/>). The homepage features:

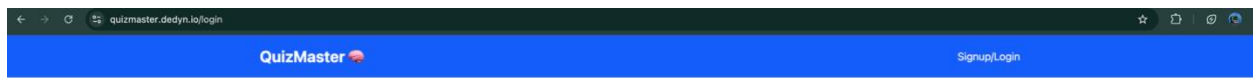
- A navbar at the top with a “Sign up/Login” button.
- A central area providing a brief introduction to the features of the app.
- A blue “Create Quiz” button that allows users to start creating their AI-generated quiz.



Logging in using email

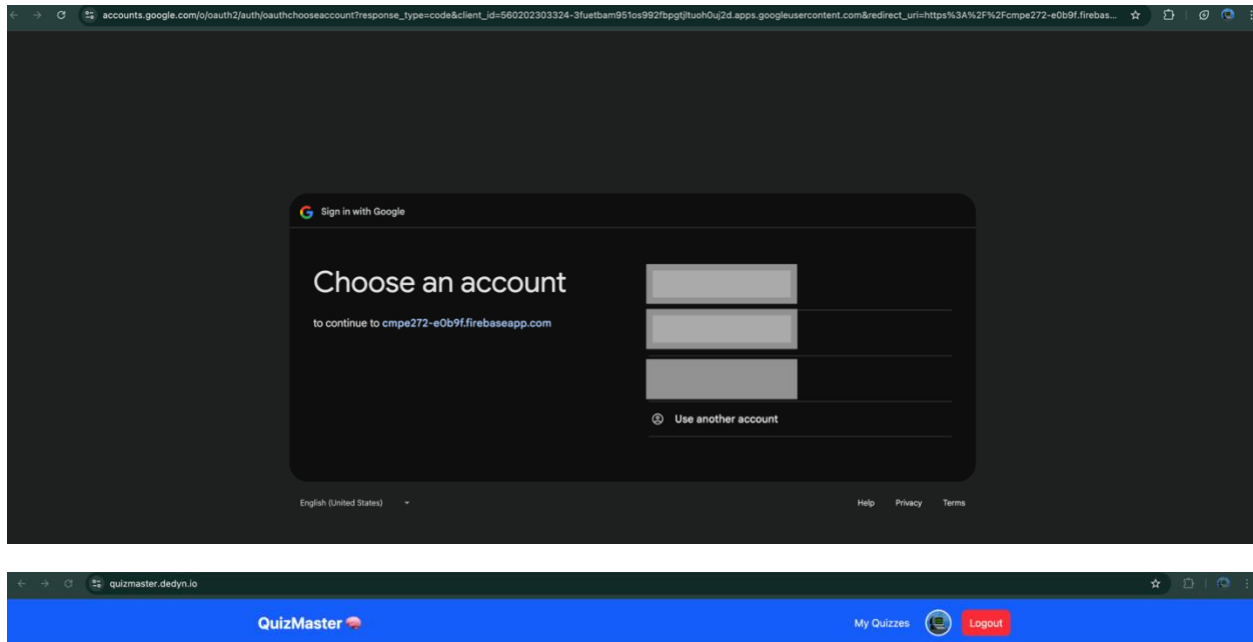
A key feature of this application is the ability for users to return to quizzes they've previously created through the “My Quizzes” tab. However, accessing this feature requires signing in.

- **Sign In or Sign Up:** The user can click the “Sign up/Login” button to sign in or create a new account.

A 'Welcome!' login and signup form. It features a 'Sign in with Google' button at the top, followed by an 'OR' separator. Below the separator are input fields for 'Email' and 'Password'. At the bottom are two buttons: a blue 'Sign In' button and a green 'Sign Up' button.

Option 1: Sign in with Google

- If the user selects “Sign in with Google,” they’ll be prompted to choose from their Google account emails.
- Upon signing in, the user will be redirected to a personalized homepage with an updated navbar featuring:
 - The “My Quizzes” tab.
 - Their Google account picture.
 - A “Logout” button.



Introducing QuizMaster

Welcome, Parth Patel

What the hell is QuizMaster?

An AI powered tool to generate a personalized quiz from your data!

How to get started?

Click the "Create Quiz" button below, give your quiz a name, upload the document you want to be quizzed in, and effortlessly generate a quiz in minutes!

Want to save your Quizzes?

Signup/Login to save the documents you upload, and the quizzes you generate. You can retake saved quizzes, share them, or delete them anytime!

Create Quiz!

Option 2: Sign Up Using a New Email and Password

- If the user chooses to sign up with a new email and password, they'll still be able to save quizzes, but the homepage layout will be slightly different.

The top screenshot shows the QuizMaster login page. It features a blue header with the QuizMaster logo and a 'Signup/Login' link. The main content area has a white box with the heading 'Welcome!'. Below this is a 'Sign in with Google' button, followed by 'OR', an email input field with 'test1234@quizmaster.com', a password input field, and two buttons: 'Sign In' (blue) and 'Sign Up' (green).

The bottom screenshot shows the user's dashboard after login. The header now includes 'My Quizzes' and a 'Logout' button. The main content area is empty.

Introducing QuizMaster

Welcome, Quiz-Amateur

What the hell is QuizMaster?

An AI powered tool to generate a personalized quiz from your data!

How to get started?

Click the "Create Quiz" button below, give your quiz a name, upload the document you want to be quizzed in, and effortlessly generate a quiz in minutes!

Want to save your Quizzes?

Signup/Login to save the documents you upload, and the quizzes you generate. You can retake saved quizzes, share them, or delete them anytime!

Create Quiz!

Create Quiz

When the user clicks on the "Create Quiz" button, they are prompted to:

- Enter a title for their quiz.
- Upload a PDF file.

File Constraints:

- The file must be in PDF format.
- The file size must be less than 5MB.

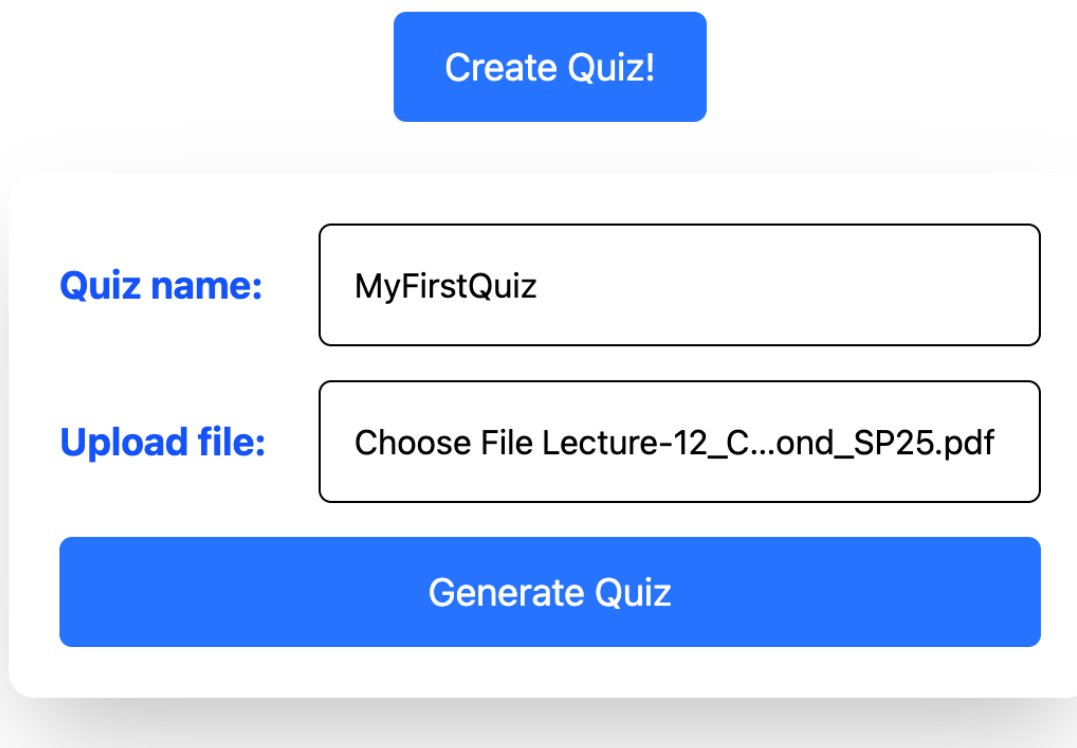
These constraints ensure ease of parsing and optimize token usage for the LLM model (Gemini). Once both inputs are completed, the user can click the “Generate Quiz” button.

Progress Indicators:

While the quiz is being generated, a status bar is displayed, which goes through the following steps:

- “Connecting to AWS...”
- “Uploading your file...”
- “Fetching Generated Quiz...”

Once all steps are completed, the user will be directed to the quiz page.



The image shows a user interface for creating a quiz. At the top, there is a blue button labeled "Create Quiz!". Below this, there is a white form with rounded corners. Inside the form, the label "Quiz name:" is followed by a text input field containing "MyFirstQuiz". Below that, the label "Upload file:" is followed by a file selection area showing "Choose File Lecture-12_C...ond_SP25.pdf". At the bottom of the form is a large blue button labeled "Generate Quiz".

Attempting Quiz

The quiz page consists of up to 5 questions (the current limit). Each question may be:

- True/False, or
- Multiple choice with 4 options, where one option is correct.

The user must select a single option for each question before submitting the quiz.

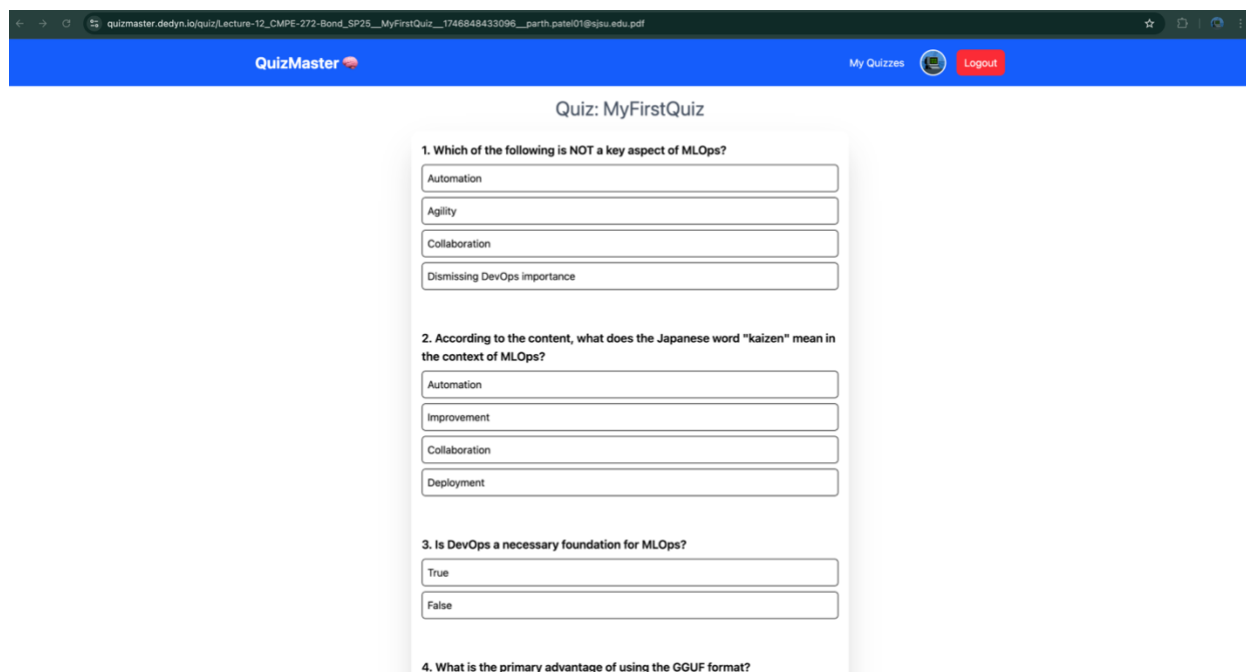
Post-Submission Feedback:

After submitting the quiz, the user will see their score. The feedback includes:

- Correct answers highlighted in green.
- Incorrect answers highlighted in red.

This feedback allows the user to learn from their mistakes and either retake the quiz immediately or later.

- The “Retake Quiz” button will reset all answers.
- The “Generate a New Quiz” button will take the user back to the homepage.



The screenshot shows a web browser window with the URL `quizmaster.dedyn.io/quiz/Lecture-12_CMPE-272-Bond_SP25_MyFirstQuiz_1746848433096_parth.patel01@ijsu.edu.pdf`. The page has a blue header with the "QuizMaster" logo and a "Logout" button. The main content area is titled "Quiz: MyFirstQuiz" and contains four questions:

1. Which of the following is NOT a key aspect of MLOps?
2. According to the content, what does the Japanese word "kaizen" mean in the context of MLOps?
3. Is DevOps a necessary foundation for MLOps?
4. What is the primary advantage of using the GGUF format?

4. What is the primary advantage of using the GGUF format?

Increased model size

Reduced model size and improved inference speed

Compatibility with only CPUs

Higher accuracy

5. What is RAG (Retrieval Augmented Generation) used for?

Fine-tuning LLMs

Improving the output of LLMs

Creating new LLMs

Building Chatbots

Quiz submitted! Correct answers are highlighted

Your Score: 4 / 5

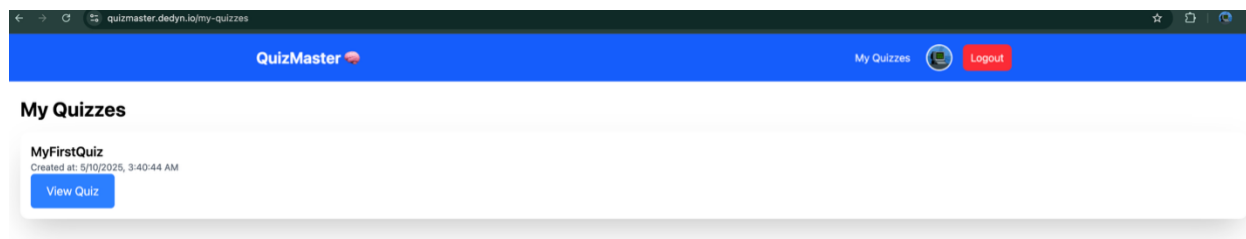
Retake Quiz

Generate a new Quiz

My Quizzes tab

The “My Quizzes” tab allows the user to see all quizzes they’ve created.

- Clicking on “View Quiz” will take the user to the quiz page again, where they can view the details of the selected quiz.



Technical Walkthrough

Code Repository

All code related to the frontend and client-side backend is stored on GitHub ([QuizMaster Repository](#)). The majority of the code resides in the frontend folder, where components are split into different subfolders for better readability.

Testing using Vitest

Testing of APIs was conducted using Vitest library:

There are a total of 7 test cases. They are as follows:

1. Renders the Create Quiz button
2. Shows error when trying to upload a file too big
3. Shows error when trying pass an empty name
4. Shows error when unable to fetch the quiz from Gemini
5. Fetches Quiz using GET operation from DynamoDB and renders correctly on QuizPage.
6. Performs the fetch Quiz operation when “My Quizzes” button is clicked
7. Performs exponential polling from DynamoDB instead of failing on the first try.

The test cases are run using the CLI command “`npm run test`” which runs the test cases as shown below

```

✓ src/App.test.tsx (7 tests) 78ms
  ✓ App Component > renders the app title
  ✓ App Component > shows error when trying to create an empty post
  ✓ App Component > shows error when trying to fetch a post without an ID
  ✓ App Component > shows error when trying to delete a post without an ID
  ✓ App Component > fetches posts using GET operation when the fetch button is clicked
  ✓ App Component > performs the POST operation when the create button is clicked
  ✓ App Component > performs the DELETE operation when confirmed in the modal

Test Files  1 passed (1)
Tests       7 passed (7)
Start at    18:26:30
Duration    556ms (transform 59ms, setup 0ms, collect 143ms, tests 78ms, environment 208ms, prepare 27ms)

PASS Waiting for file changes...
press h to show help, press q to quit

```

AWS API Gateway

Two API Gateways have been created:

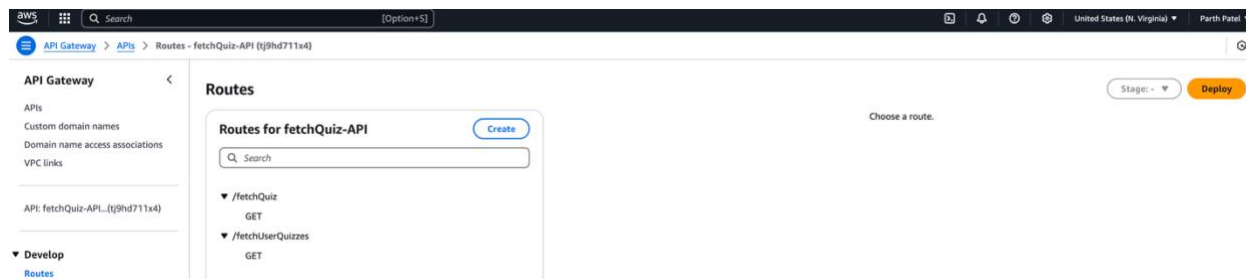
1. **postpdfqm Gateway** (REST Protocol):
 - Responsible for accepting the quiz name and PDF file, creating a unique destination location in an S3 bucket, and uploading the file.
 - Only handles POST operations.
2. **fetchQuiz-API Gateway** (HTTP Protocol):
 - Responsible for fetching the quiz from DynamoDB once created.
 - Also used to fetch quizzes in the "My Quizzes" section.
 - Only handles GET operations.

The screenshot displays the AWS API Gateway console. The top section shows a list of APIs:

Name	Description	ID	Protocol	API endpoint type	Created
fetchQuiz-API	Created by AWS Lambda	tj9hd711x4	HTTP	Regional	2025-02-13
postpdfqm		36f0au07n8	REST	Regional	2025-02-06

The bottom section shows the details for the **postpdfqm** API. It lists the resources and methods:

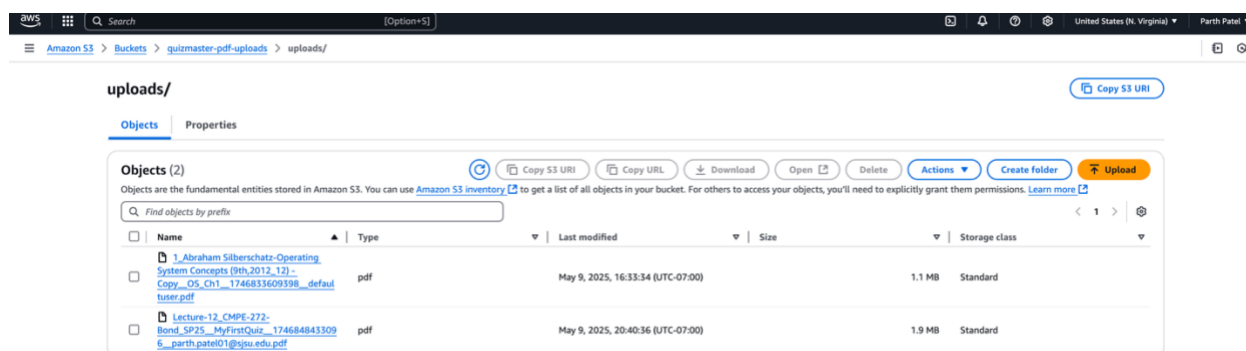
- Resources:**
 - POST**
 - /generate-presigned-url** (ANY, OPTIONS, POST, PUT)
- Methods (1):**
 - POST** (Integration type: Lambda, Authorization: None, API key: Not required)



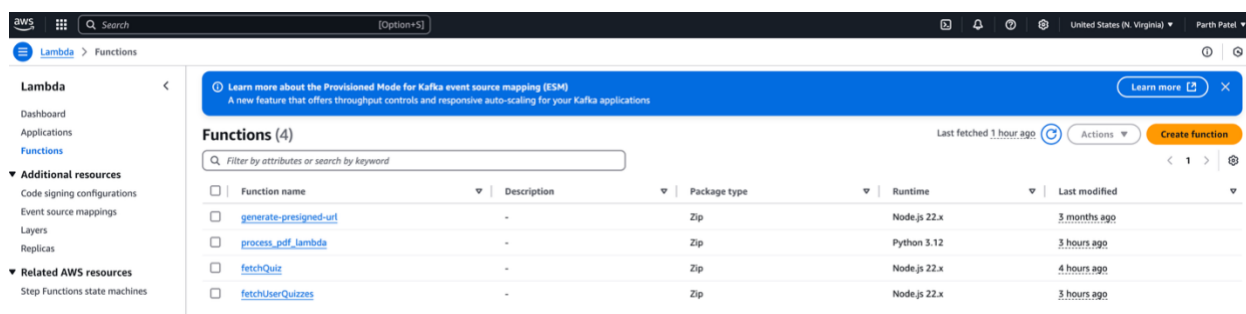
AWS S3

The S3 bucket serves as the landing zone for PDF files. Once a unique destination name is generated, the files are uploaded.

- The bucket can store up to 5GB of data under the free tier.
- Each file is stored with a unique name in the format:
`<name_of_file>__<quiz_name>__<datetime>__<email_if_provided>.pdf`



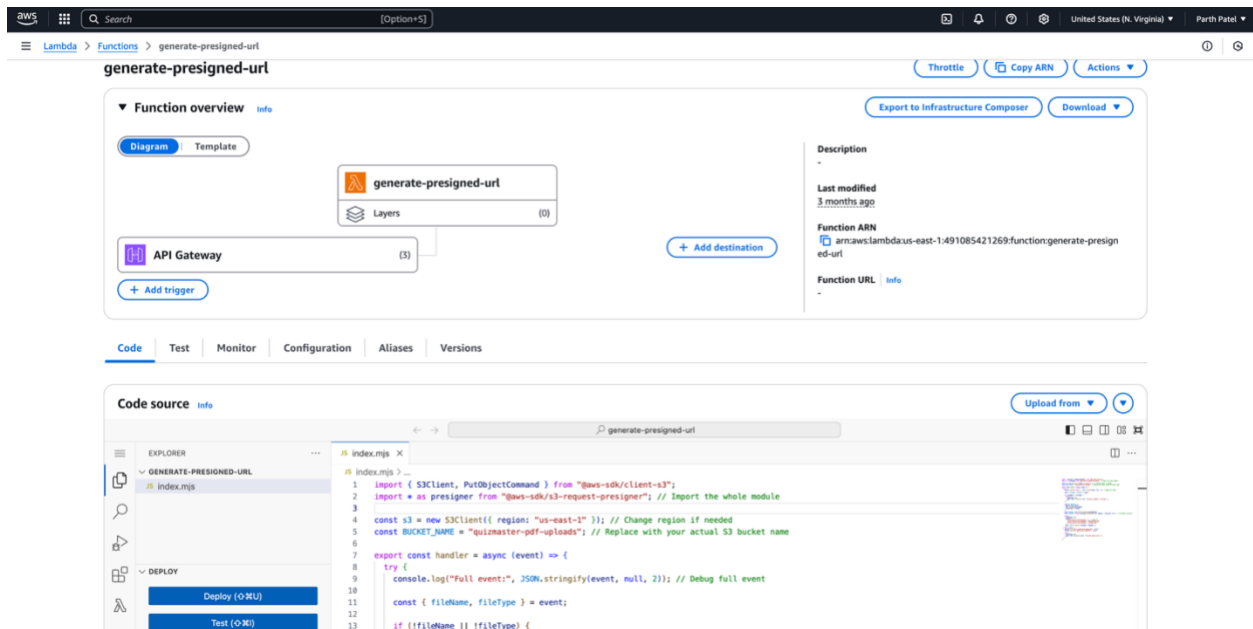
AWS Lambda Functions



There are four Lambda functions, each with a specific role:

1. Generate-presigned-url

- Accepts the quiz name and file via the API Gateway and uploads the file into the S3 bucket.



2. Process_pdf_lambda

- Triggered once the file is uploaded to S3.
- Parses the PDF, sends it along with a prompt to Gemini, and receives a JSON containing questions and answers.
- The JSON response is then stored in DynamoDB.

The screenshot shows the AWS Lambda console for the function 'process_pdf_lambda'. The 'Function overview' tab is active, displaying a diagram of the function's architecture. It shows an S3 bucket as the trigger, which points to the 'process_pdf_lambda' function. The function is configured with a single layer. The 'Code source' tab is also visible, showing the Python code for the function. The code defines a 'process_pdf' function that takes a bucket and key as input, logs the raw response, parses the JSON response, and extracts the document name by removing 'upload/' from the key. The function is deployed to the 'us-east-1' region.

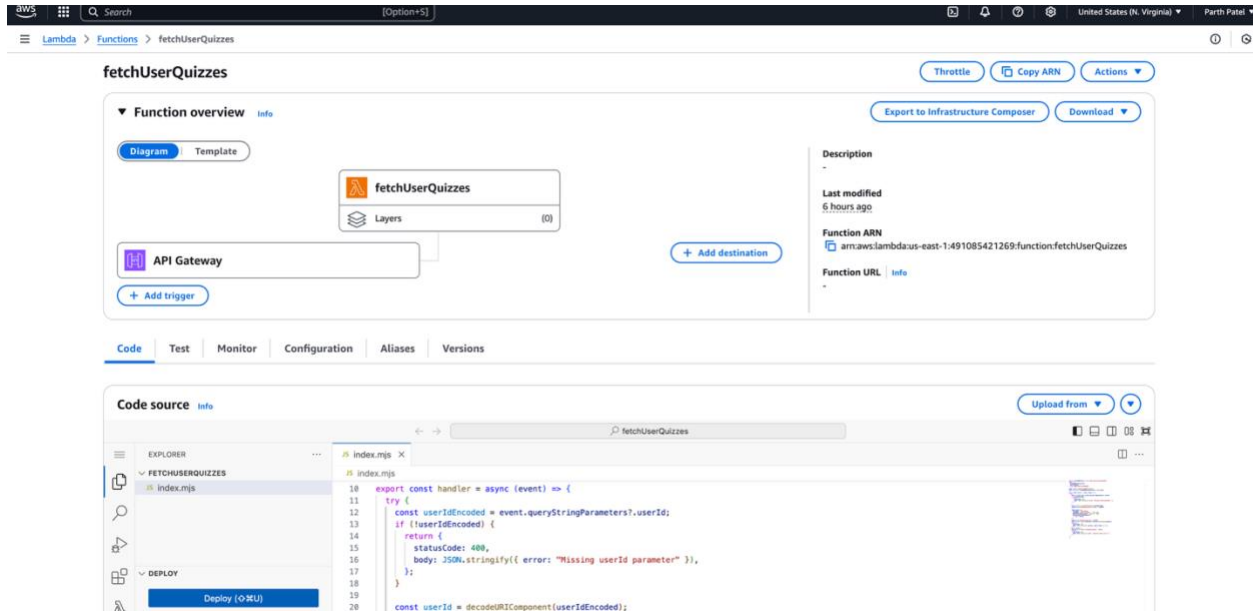
3. fetchQuiz

- While the previous functions execute, the client-side performs exponential polling on this function to check if the quiz is available in DynamoDB.
- If the quiz is found, it fetches and displays the data.
- If not found after 10 tries, a failure message is shown.

The screenshot shows the AWS Lambda console for the function 'fetchQuiz'. The 'Function overview' tab is active, displaying a diagram of the function's architecture. It shows an API Gateway as the trigger, which points to the 'fetchQuiz' function. The function is configured with a single layer. The 'Code source' tab is also visible, showing the JavaScript code for the function. The code defines a 'fetchQuiz' function that takes a quizId as input, decodes the URL component, logs the raw quizId, and then fetches the quiz data from a DynamoDB table named 'QuizSystem'. The function is deployed to the 'us-east-1' region.

4. fetchUserQuizzes

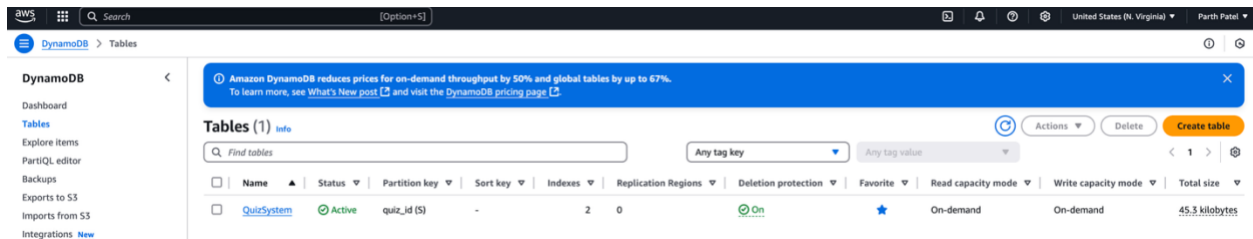
- Triggered when the user clicks "My Quizzes." It fetches quizzes generated by the logged-in user via a DynamoDB query.
- Accepts the user's email and returns a list of quizzes for display on the client side.



AWS DynamoDB

A DynamoDB table named QuizSystem is used, with an index on quiz_id. Table attributes include:

- quiz_id
- document_name
- error_message
- generated_at
- metadata
- questions
- quiz_name
- source_document
- status
- user_id



DynamoDB

Dashboard
Tables
Explore items
 PartiQL editor
 Backups
 Exports to S3
 Imports from S3
 Integrations **New**
 Reserved capacity
 Settings

▼ **DAX**
 Clusters
 Subnet groups
 Parameter groups
 Events

Find tables

QuizSystem

Scan Query

Select a table or index
Table - QuizSystem

Select attribute projection
All attributes

Filters - optional

Run Reset

Completed - Items returned: 9 - Items scanned: 9 - Efficiency: 100% - RCUs consumed: 5

Table: QuizSystem - Items returned (9)

Scan started on May 09, 2025, 20:46:01

quiz_id (String)	metadata	questions	quiz_name	source_document	status	user_id
2_Abraham_Silberschatz-Operating_System_Concepts_9th...	{ "question_...	[{ "M": { "q...	OS_Ch2	https://quizmaster...	success	dev.v.pate
Lecture-09_CMPE-272-Bond_SP25_Test_1746836259752...	{ "question_...	[{ "M": { "q...	Test	https://quizmaster...	success	patelspart
Lecture-12_CMPE-272-Bond_SP25_MyFirstQuiz_1746848...	{ "question_...	[{ "M": { "q...	MyFirstQuiz	https://quizmaster...	success	parth.pate
Lecture-12_CMPE-272-Bond_SP25_FCE_1746828404183...	{ "question_...	[{ "M": { "q...	FCE	https://quizmaster...	success	patelspart
1_Abraham_Silberschatz-Operating_System_Concepts_9th...	{ "source_d...		OS_Ch1		error	defaultus
Lecture-12_CMPE-272-Bond_SP25_FC_1746828374118...	{ "question_...	[{ "M": { "q...	FC	https://quizmaster...	success	defaultus
1_Abraham_Silberschatz-Operating_System_Concepts_9th...	{ "source_d...		OS_Ch1		error	defaultus
Lecture-11_CMPE-131-01_Bond_SP25_Lecture_11_17468...	{ "question_...	[{ "M": { "q...	Lecture_11	https://quizmaster...	success	patelspart

Firestore Authentication

Firestore Authentication is used to allow users to log in via email or Google account. A Firestore project has been set up with the necessary configurations, including sign-in providers and allowed domains.

Firebase

Project Overview

Project shortcuts

Authentication

Product categories

Build
Run
Analytics
AI

All products

Related development tools
Firebase Studio
Checks

CMPE272

Authentication

Users Sign-in method Templates Usage Settings Extensions

Sign-in providers

Add new provider

Provider	Status
Email/Password	Enabled
Google	Enabled

Advanced

SMS Multi-factor Authentication

Allow your users to add an extra layer of security to their account. Once enabled, integrated and configured, users can sign in to their account in two steps, using SMS. [Learn more](#)

MFA and other advanced features are available with Identity Platform, Google Cloud's complete customer identity solution built in partnership with Firebase. This upgrade is available on both the Spark and Blaze plans.

Upgrade to enable

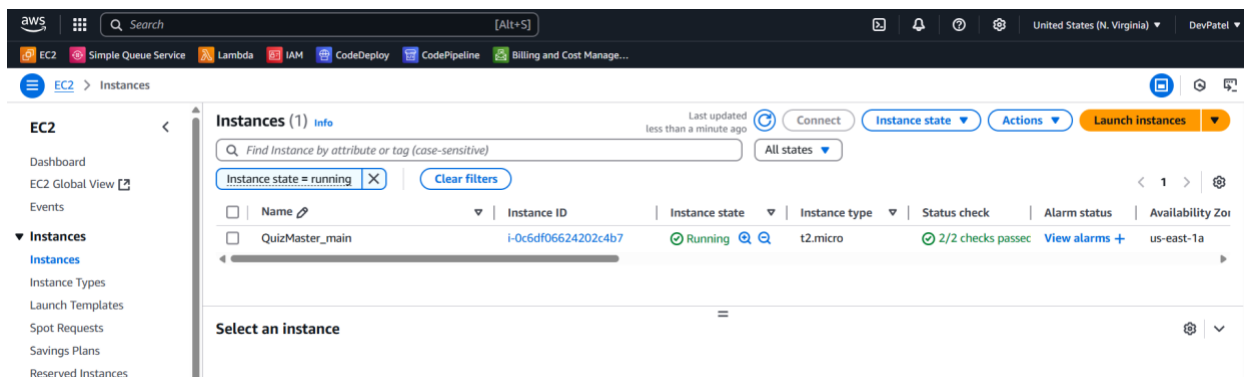
Deployment and CI/CD (Dev)

Cloud Deployment

For demo purposes and to avoid incurring unnecessary charges the application was deployed plainly on AWS EC2 using Nginx for reverse proxy to mask the Public IP of the EC2 instance and setup SSL security with OpenSSL.

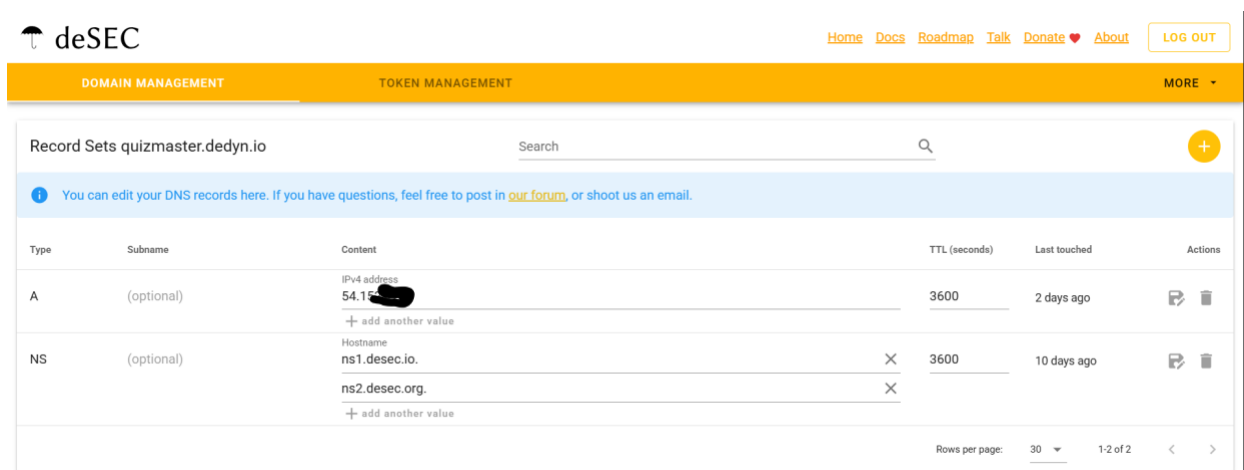
EC2 Instance was created with Ubuntu OS, t2.micro instance type, and Free tier eligible 30 GB storage, the Instance was setup in us-east-1 (N. Virginia) region for faster access and low cost. Security Group assigned to it had ports 80 and 443 were exposed publicly for HTTP and HTTPS requests.

AWS EC2



The domain quizmaster.dedyn.io was acquired from the desec.io DNS provider and the DNS record with CNAME A was updated to the public IP of the EC2 instance.

<https://desec.io/domains/quizmaster.dedyn.io>



The GitHub Repo was cloned into the EC2 Instance using AWS EC2 Instance Connect and after installing necessary dependencies using npm and installing the libraries required including node JS, OpenSSL, etcetera.

After testing out the frontend on Public IP using **npm run dev** the build for the application was created using **npm run build** that built the vite app and a dist folder was created contents of which were then copied into a folder at path `var/www/quizmaster` and then Nginx config file was edited to host this folder for frontend instead of the default Nginx frontend.

```
ubuntu@ip-172-31-84-215:~$ cat /etc/nginx/sites-available/quizmaster
server {
    server_name quizmaster.dedyn.io;

    root /var/www/quizmaster;
    index index.html;

    location / {
        try_files $uri /index.html;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/quizmaster.dedyn.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/quizmaster.dedyn.io/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}
server {
    if ($host = quizmaster.dedyn.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name quizmaster.dedyn.io;
    return 404; # managed by Certbot
}
ubuntu@ip-172-31-84-215:~$
```

Once that was done the Nginx was routing the domain to public IP and the application was visible on the domain. After that SSL was set up by installing certbot and then linking the newly obtained SSL certificate to the quizmaster.dedyn.io domain.

`sudo apt update`

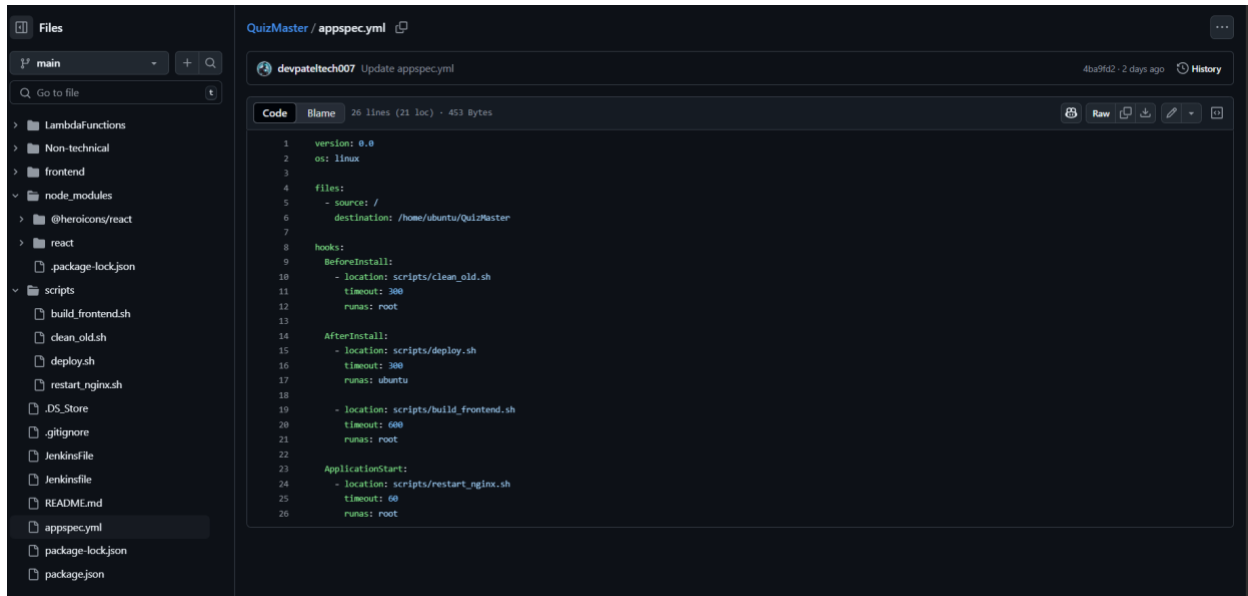
`sudo apt install certbot python3-certbot-nginx -y`

`sudo certbot --nginx -d quizmaster.dedyn.io`

CI/CD (Continuous Integration Continuous Deployment):

AWS CodePipeline was used to set up a CI/CD pipeline with source as GitHub repo and GitHub account owner had to authorize AWS CodePipeline access to GitHub hooks for listening to Push requests which then triggers the CI/CD pipeline to kickoff.

appspec.yml file was added to the root directory of GitHub Repo for AWS CodeDeploy to pull the stages of deployment with the events and execute them chronologically.



CodeDeploy Agent was installed on EC2 which is running as a service to execute the CodeDeploy commands.

```
May 10 00:00:16 ip-172-31-84-215 systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
ubuntu@ip-172-31-84-215:~$ sudo systemctl status codedeploy-agent.service
● codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent
   Loaded: loaded (/etc/init.d/codedeploy-agent; generated)
   Active: active (running) since Fri 2025-05-09 22:52:49 UTC; 2 days ago
     Docs: man:systemd-sysv-generator(8).
   Process: 516 ExecStart=/etc/init.d/codedeploy-agent start (code=exited, status=0/SUCCESS)
    Tasks: 4 (limit: 1129)
   Memory: 147.9M (peak: 280.1M)
      CPU: 35.079s
   CGroup: /system.slice/codedeploy-agent.service
           └─791 "codedeploy-agent: master 791"
              └─794 "codedeploy-agent: InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller of master 791"

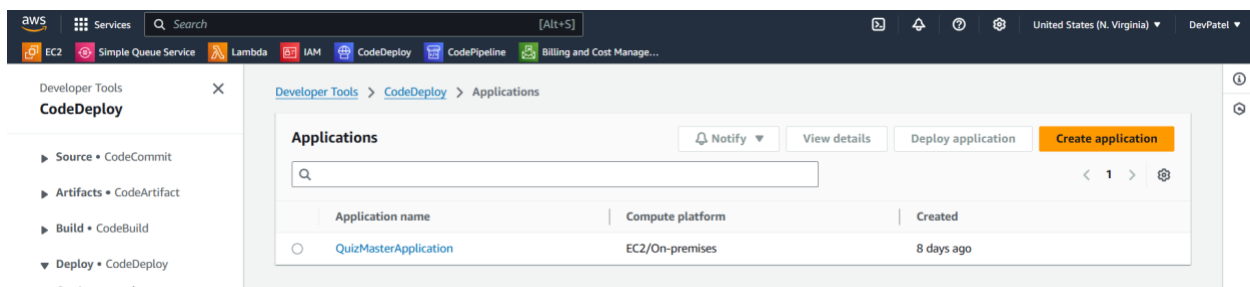
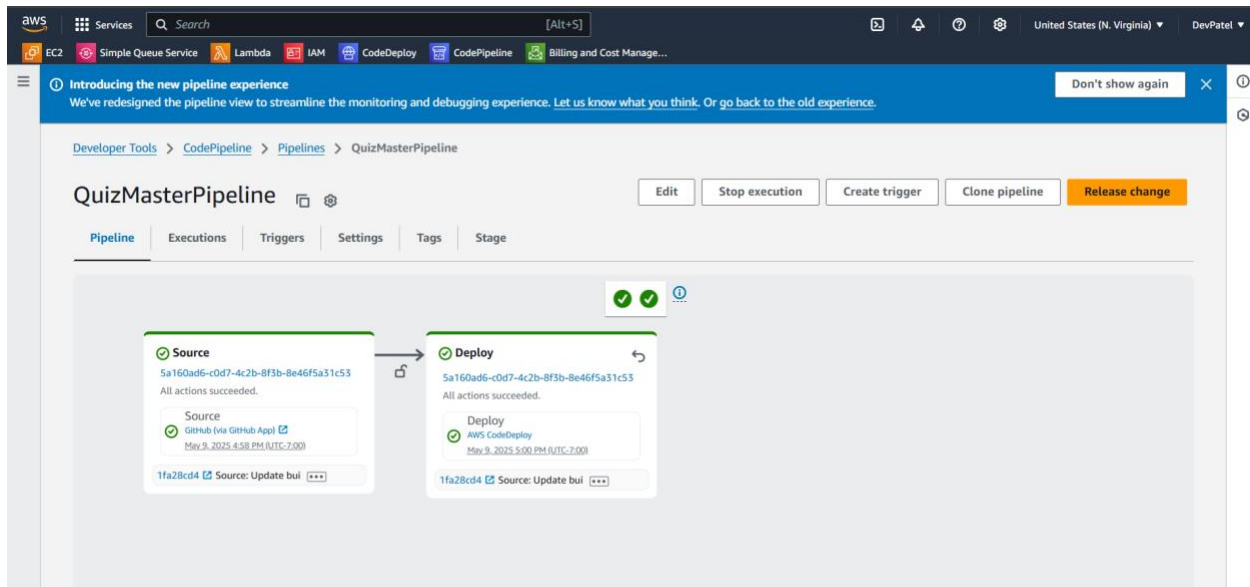
May 09 23:59:48 ip-172-31-84-215 su[3835]: (to root) root on none
May 09 23:59:48 ip-172-31-84-215 su[3835]: pam_unix(su:session): session opened for user root(uid=0) by (uid=0)
May 09 23:59:48 ip-172-31-84-215 su[3835]: pam_unix(su:session): session closed for user root
May 09 23:59:50 ip-172-31-84-215 su[3856]: (to ubuntu) root on none
May 09 23:59:50 ip-172-31-84-215 su[3856]: pam_unix(su:session): session opened for user ubuntu(uid=1000) by (uid=0)
May 09 23:59:56 ip-172-31-84-215 su[3906]: (to root) root on none
May 09 23:59:56 ip-172-31-84-215 su[3906]: pam_unix(su:session): session opened for user root(uid=0) by (uid=0)
May 10 00:00:16 ip-172-31-84-215 su[4011]: (to root) root on none
May 10 00:00:16 ip-172-31-84-215 su[4011]: pam_unix(su:session): session opened for user root(uid=0) by (uid=0)
May 10 00:00:16 ip-172-31-84-215 su[4011]: pam_unix(su:session): session closed for user root
ubuntu@ip-172-31-84-215:~$
```

There are 3 scripts under the **/scripts** folder in the repo root directory which contains bash scripts to remove the old files from the folder, pulls new changes into the EC2 and then do npm install and npm run build to build the frontend, replace it into Nginx hosted files and then restart Nginx. The site keeps serving the old front end till the pipeline has completed all the stages.

The CI/CD Pipeline works as follows:

1. A commit pushed to the GitHub Repo.
2. AWS CodePipeline triggered.

3. AWS CodeDeploy pulls the GitHub Commits and deploys them on EC2 using the CodeDeploy Agent installed on the EC2 Instance.



The screenshot displays the AWS CodeDeploy console for the application 'QuizMasterApplication'. The left sidebar shows the navigation menu with 'CodeDeploy' selected. The main content area shows the 'QuizMasterApplication' details, including its name and compute platform (EC2/On-premises). Below this, the 'Deployment groups' tab is active, showing a table with one deployment group: 'QMDeploymentGroup', which has a status of 'Succeeded' and a trigger count of 0. The bottom section shows the 'Revision details' for a specific revision, including its location, creation time, and a list of events (ApplicationStop, DownloadBundle, BeforeInstall, Install, AfterInstall, ApplicationStart, ValidateService) all of which succeeded.

QuizMasterApplication

Application details

Name	QuizMasterApplication	Compute platform	EC2/On-premises
------	-----------------------	------------------	-----------------

Deployment groups

Name	Status	Last attempted deploy...	Last successful deploy...	Trigger count
QMDeploymentGroup	Succeeded	May 9, 2025 5:00 PM (UT...	May 9, 2025 5:00 PM (UT...	0

Revision details

Event	Duration	Status	Error code	Start time	End time
ApplicationStop	less than one second	Succeeded	-	May 9, 2025 4:59 PM (UTC-7:00)	May 9, 2025 4:59 PM (UTC-7:00)
DownloadBundle	less than one second	Succeeded	-	May 9, 2025 4:59 PM (UTC-7:00)	May 9, 2025 4:59 PM (UTC-7:00)
BeforeInstall	less than one second	Succeeded	-	May 9, 2025 4:59 PM (UTC-7:00)	May 9, 2025 4:59 PM (UTC-7:00)
Install	less than one second	Succeeded	-	May 9, 2025 4:59 PM (UTC-7:00)	May 9, 2025 4:59 PM (UTC-7:00)
AfterInstall	25 seconds	Succeeded	-	May 9, 2025 4:59 PM (UTC-7:00)	May 9, 2025 5:00 PM (UTC-7:00)
ApplicationStart	less than one second	Succeeded	-	May 9, 2025 5:00 PM (UTC-7:00)	May 9, 2025 5:00 PM (UTC-7:00)
ValidateService	less than one second	Succeeded	-	May 9, 2025 5:00 PM (UTC-7:00)	May 9, 2025 5:00 PM (UTC-7:00)

Organization

Project group

Name	Tasks
Parth Patel	<ul style="list-style-type: none"> Frontend development

	<ul style="list-style-type: none"> • Client-side backend develop • FetchQuiz & FetchUserQuizzes • Lamda development • Authentication/Authorization development • Testing
Shailen Sutradhar	<ul style="list-style-type: none"> • Requirements Gathering • Generate_presigned_url Lambda development • Provisioning resources
Dev Patel	<ul style="list-style-type: none"> • Cloud Deployment • Continuous Integration /Continuous Deployment pipeline • EC2 Provisioning • Cost Optimization
Gautam Santhanu Thampy	<ul style="list-style-type: none"> • Process_pdf Lambda development • Prompt engineering

Communication

GitHub: github.com/ParthPatel00/QuizMaster

Website: <https://quizmaster.dedyn.io/>

Citations

1. [https://docs.aws.amazon.com/glossary/latest/reference/glossary.html#:~:text=Amazon%20Web%20Services%20\(AWS\),is%2Dcloud%2Dcomputing%2F%20](https://docs.aws.amazon.com/glossary/latest/reference/glossary.html#:~:text=Amazon%20Web%20Services%20(AWS),is%2Dcloud%2Dcomputing%2F%20).