

A serverless taxi booking system on AWS cloud

A

Project Report

*submitted in partial fulfillment of the
requirements for the award of the degree of*

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING with

Specialization in CCVT

by

SAP ID

Roll No.

Name

500087456

R2142201858

Mohit Bishesh

under the guidance of

Mr. Saurabh Shanu

Assistant Professor - Senior Scale,

Systemics Cluster

School of Computer Science



Systemics Cluster

School of Computer Science

University of Petroleum & Energy Studies

Bidholi, Via Prem Nagar, Dehradun, UK

April – 2023



CANDIDATE'S DECLARATION

We hereby certify that the project work entitled “**A serverless Taxi booking system on AWS cloud**” in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in CCVT, and submitted to the Systemics Cluster at School of Computer Science, University of Petroleum & Energy Studies, Dehradun, is an authentic record of my work carried out during the period from **January, 2023** to **April, 2023** under the supervision of **Mr. Saurabh Shanu**, Assistant Professor - Senior Scale, Systemics Cluster, School of Computer Science.

The matter presented in this project has not been submitted by me for the award of any other degree of this or any other University.

Name:

Mohit Bishesh

SAP ID:

500087456

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: _____2023

Mr. Saurabh Shanu
(Subject Faculty)



School of Computer Science

University of Petroleum & Energy Studies, Dehradun

Project Based Learning:

I

A serverless taxi booking system on AWS cloud cloud:

ABSTRACT (250-300 words)

The advent of cloud computing has brought about a paradigm shift in the way software applications are developed and deployed. One of the emerging trends in cloud computing is serverless computing, which enables developers to focus on code instead of server management. This paper proposes a serverless taxi booking system on AWS cloud that utilizes AWS Lambda, API Gateway, and DynamoDB services, AWS code commit, AWS Amplify, AWS Cognito to provide a scalable, highly available, and cost-effective solution.

The system architecture consists of a serverless backend that handles user requests for Creating an account, login using credentials or using third party like Google or Gmail, taxi booking, driver matching, and ride scheduling. The backend is implemented using AWS Lambda functions, which are event-driven and can scale automatically based on demand. API Gateway is used to expose the Lambda functions as RESTful APIs, which can be accessed by clients over the internet.

The system uses DynamoDB as a NoSQL database to store and retrieve user data, driver data, and ride data. DynamoDB is highly scalable and can handle large volumes of data with low latency. It also provides a flexible data model that can adapt to changing requirements.

The system implements several security features, including AWS Identity and Access Management (IAM) roles, encryption of data in transit and at rest, and network security groups. These features along with AWS Cognito ensure that user data is protected against unauthorized access and data breaches.

The serverless taxi booking system on AWS cloud offers several advantages over traditional server-based systems, including reduced infrastructure costs, increased scalability, and improved availability. It also enables developers to focus on code development and innovation, rather than server management. The proposed system can be extended to other transportation services such as car rental, ride-sharing, and public transit.

Keywords: serverless computing, scalability, availability, Infrastructure cost reduction, Cost-effectiveness

Contents

INTRODUCTION (6-7 pages).....	1-5
PROBLEM STATEMENT (1 page)	6
OBJECTIVE (1 page).....	7
RELATED WORK (6 pages).....	8-13
WHY DO WE NEED CLOUD FOR THE PROJECT (5-6 PAGES)	14-19
METHODOLOGY (4-5 pages).....	20-25
ALGORITHM (2-3 pages).....	26-27
RESULT AND DISCUSSIONS (8-10 pages)	28-35
PUBLIC CLOUD DEPLOYMENT (4-5 pages).....	346-40
SCREENSHOTS.....	41
Key Bibliography/References	42

INTRODUCTION

In recent years, the demand for transportation services has increased significantly due to urbanization, population growth, and technological advancements. Taxi booking systems have revolutionized the transportation industry, enabling users to book rides on-demand, making it more convenient and accessible.

The current need for a taxi booking system has grown significantly with the rapid growth of the ride-hailing industry. Traditional taxi services have been facing increasing competition from ride-hailing services like Uber, Lyft, and Grab, which have disrupted the transportation industry by providing users with a convenient and reliable means of transportation.

The rise of ride-hailing services has also contributed to the need for a taxi booking system as they provide a seamless user experience that allows users choose a ride as per his convenience and comfort which he can book through the web application or the mobile app and also have a look upon the location of the driver and ride approaching towards him. This level of convenience and flexibility is driving the demand for taxi booking systems.

Moreover, taxi booking systems also offer several advantages to taxi companies, including increased efficiency in dispatching, reduced downtime between rides, and better utilization of resources. By using a taxi booking system, taxi companies can also improve their customer service by providing more accurate ETA's (Estimated Time of Arriva), real-time tracking, and other value-added services that can help them retain customers.

The need for a taxi booking system has also been accentuated by the COVID-19 pandemic, which has increased the demand for contactless transactions and the need for social distancing. Taxi booking systems that allow for cashless payments, driver ratings, and contactless pickup and drop-off can provide a safer and more hygienic means of transportation for both drivers and riders.

In summary, the current need for a taxi booking system is driven by the growing demand for convenient and reliable transportation services, increasing competition in the ride-hailing industry, and the need for contactless transactions and social distancing. As the transportation industry continues to evolve, taxi booking systems will play a critical role in providing a seamless and safe experience for users and taxi companies alike.

To cater to the growing demand for taxi booking systems, developers are adopting new technologies and cloud-based solutions to improve scalability, security, and cost-effectiveness.

Although we have several cloud services provider like GCP, AWS, Azure, but Amazon Web Services is a leading provider and have a wide range of offerings with easy to use and works on-demand.

Amazon Web Services (AWS) is a popular cloud computing platform that provides a wide range of services for building serverless applications.

A taxi booking system on AWS can benefit greatly from a serverless architecture. One of the primary advantages of serverless computing is its scalability. A taxi booking system may experience high volumes of traffic during peak hours or events, while being relatively quiet during off-peak times. With a serverless architecture, our resources will automatically scale on demand, which will ensure the better utilization of the resources and reduce the cost. This allows the system to handle sudden spikes in demand without needing to provision additional servers or worry about capacity planning.

Another advantage of serverless computing for a taxi booking system is cost-effectiveness. With a traditional infrastructure, the system would require a dedicated set of servers and resources to handle the expected traffic. However, during quiet periods, many of these resources would be underutilized, leading to wasted costs. With a serverless architecture, the system only charges for the actual usage of the service, so during periods of low traffic, the cost of running the system would be significantly reduced. Additionally, serverless computing eliminates the need to manage and maintain infrastructure, further reducing operational costs.

Serverless computing can also improve the speed and agility of the taxi booking system. With a traditional infrastructure, developers need to spend time managing and configuring servers, which can be time-consuming and prone to error. With serverless computing, developers can focus on writing code that delivers business value, as the underlying infrastructure is abstracted away. This allows for quicker deployment of new features and updates, which can be essential in a fast-paced industry like taxi booking.

Serverless computing has achieved a lot of traction in the past few years due to its many benefits. It enables the developers to write the code with zero overhead of managing the servers. In such architecture, the CSP looks over the over the management of infrastructure, operating system, servers, and application runtimes. It allows developers to focus on writing code that implements business logic rather than managing servers.

In this project, I am creating a serverless taxi booking system on AWS cloud using various AWS services such as CodeCommit, DynamoDB , cloud watch, Cognito , Amplify, AWS Lambda function. , and API Gateway. The taxi booking system is a real-world application that has become popular due to the growing demand for on-demand transportation services. The system will allow customers to book a taxi online through a mobile app or a web application. The app will communicate with the backend serverless architecture built using AWS services.

Currently AWS is the most popular and leading cloud CSP that offers a wide range of services which can be used to test, build and deploy a serverless taxi booking system. There are several reasons why AWS is a good choice for deploying a serverless taxi booking system.

Firstly, AWS offers a pool of services based on pay as you go model which are used to test, build and deploy a serverless architecture. This includes services such as AWS Lambda for serverless computing, Amazon API Gateway for API management, Amazon S3 for object storage, Amazon DynamoDB for NoSQL database, and Amazon Cognito for user authentication and authorization. With this comprehensive range of services, you can build and deploy a serverless taxi booking system that meets your specific needs.

Secondly, AWS provides automatic scaling capabilities that can help your taxi booking system

handle a large number of users and requests. With AWS, you can scale up or down based on demand, which can help reduce costs and improve performance. This is especially beneficial for a taxi booking system, which needs to handle a large number of requests during peak hours.

Thirdly, AWS provides a secure infrastructure that can help protect your taxi booking system from cyber threats. AWS offers features such as VPC (Virtual Private Cloud), security groups, and network ACLs that can help you create a secure environment for your application. It is particularly important for a taxi booking application, which deals with sensitive user information such as payment details and personal information.

Fourthly, AWS provides a flexible platform that allows you to choose the services and tools that best fit your requirements. With AWS, you can choose from various programming languages, frameworks, and deployment models, allowing you to build and deploy your taxi booking system the way you want. This flexibility can help you customize your system to meet the unique needs of your business.

Finally, AWS is based on the pay-as-you-use model for billing and pricing, which helps us to reduce the charges (cost)s by only paying for the resources we want and for the duration we use. This can be especially beneficial for serverless architecture, where you only pay for the time, your code is running. This can help reduce the costs of running a taxi booking system and make it more cost-effective for small and medium-sized businesses.

In conclusion, AWS provides a comprehensive and flexible platform that can help you build and deploy a serverless taxi booking system that is scalable, secure, and cost-effective. The wide range of services, automatic scaling capabilities, secure infrastructure, flexibility, and pay-as-you-go pricing model make AWS an ideal choice for deploying a serverless taxi booking system.

In this project we are going to build a taxi booking system which will be a serverless web-app and will enable the customer to book a ride from a preferred location. The user can choose the location of pickup from the map, and for which we will be using [ArcGIS.com](https://www.arcgis.com) to show the map.

We will deploy our project in AWS and will use the AWS services such as, AWS Amplify, AWS Cognito, API Gateway, Lambda function, and DynamoDB.

We will Upload our HTML, CSS, and JS and other website content files on AWS Code Commit which will build a backend server for us.

Now whenever the user interacts with our website, he will be asked to create an account using his e-mail and phone number or he will get the option to authenticate first which he can do either by signing through the google or Gmail or via his actual existing e-mail id so that only the trusted and genuine users can access our website and book the ride.

As the user will be authenticated, he can choose the pickup location, after which APIs will be called through the API Gateway which will trigger the lambda function to finally book a ride and store the data of the user in the DynamoDB.

Here as the data is being stored to DynamoDB by triggering lambda function, its making it a bit dynamic. Once the user successfully books the ride, a display will appear which will show the booking confirmation and he can see the path on the map which the ride will follow to approach towards the user.

The user request flow diagram is well shown in the diagram below. It shows how the user request will be generated and will be executed as well as handled.

Let's us have a Glimpse of the AWS services we will using and reason for using them in our project. We will discuss how these services are essential for hosting a serverless web application and how they make things easy and reliable.

1. **AWS Amplify:** it is a service offered by AWS which helps the users to build and deploy a full-Stack application with highly scalability and security.

Amplify provides a wide range of services, including authentication, APIs, storage, analytics, hosting, and CI/CD, all integrated into a single platform. These services allow developers to add functionality to their application seamlessly and efficiently, reducing development time and costs.

So, if we were not using the cloud-based solution, then we need to setup our own infrastructure, and need to manage the servers and the backend application.

Also, amplify provides us a public link through which the customer can access our website, so we do not need to purchase for a public DNS for our website in the initial phases and hence is cost effective. Also, we have used our on-premise server, then we need to be worry about to setup the proper environment to run our code and manage the physical servers.

2. **AWS Cognito:** It is a service offered by AWS which helps us for Authentication and authorisation of the users accessing our website or application. It is fully managed service used for identity management and user authentication. It helps developers add user sign-up, sign-in, and access control to their applications easily and securely. Cognito provides a set of features that enable developers to create and manage user authentication and identity-related tasks with minimal coding and infrastructure setup.

One of the primary benefits of Cognito is its flexibility. Developers can choose from several authentication methods such as username/password, social identity providers (such as Facebook and Google), and multi-factor authentication (MFA). Cognito also supports custom authentication flows, enabling developers to implement their own authentication logic and workflows.

Cognito maintains user pool and identity pool. Under user pool, it maintains the data of the user and is responsible for authentication whereas identity pool will provide the permissions (authorization) to the users to after the successful authentication is done.

Whereas, if we have used on premise strategy, then the authorisation and authentication processes are really complex and challenging to perform (or merely impossible to be done for small Startups).

Hence, it increases the security of our website and easily filters the unknown (unauthorized users).

3. **DynamoDB:** It is the fully managed NoSQL database provided by the AWS to store the data in the form of rows and columns with high scalability. It is highly scalable so we only need to pay for the amount of storage we will use as per our traffic.

So, in our previous days, with low Expenditure cost, we do not need think for the overhead of storage capacity.

4. **Lambda function:** it helps us to run our code on high availability compute resources. It is event triggered and has the capacity to automatically scale.

with the help of lambda function on cloud, we are making our application as serverless. The lambda function also provides us the 1 million instructions free of cost and even responsible of scaling of resources in case of sudden increase in high request. Whereas, if was very difficult to process, manage and respond to a million of instructions in case of on premise setup and would have required high capex and opex with high skilled IT professionals.

We will develop our web app using HTML, CSS, and JS and will upload it along with other website content files on AWS Code Commit. Then we will integrate our Code Commit with AWS amplify by creating a CI/CD pipeline which will build a backend server for us.

Now whenever the user interacts with our website, he will be asked to create an account using his e-mail and phone number or he will get the option to authenticate first which he can do either by signing through the google or Gmail or via his actual existing e-mail id so that only the trusted and genuine users can access our website and book the ride.

Once they are registered, then they can successfully login for next time via their credentials.

As the user will be authenticated, he can choose the pickup location, after which APIs will be called through the API Gateway which will trigger the lambda function to finally book a ride and store the data of the user in the DynamoDB.

Here as the data is being stored to DynamoDB by triggering lambda function, its making it a bit dynamic.

Once the user successfully books the ride, a display will appear which will show the booking confirmation and he can see the path on the map which the ride will follow to approach towards the user.

PROBLEM STATEMENT (1 page)

We have identified that a recent change in the trend of transportation services has occurred. The traditional system of transportation generally creates a overhead for the passenger to go to a nearby taxi stand and look for a desire taxi to travel. The traditional taxi booking system offers a smaller number of choices for the taxi to the passenger. On the other hand, it also leads to high cost and sometimes consumes more time which leads to inconvenience and delay to the passengers. It is also less reliable for the customers as generally they do not have the prior driving experience with the particular chosen driver and hence are sometime risky in case of long tour or travelling to the remote areas.

Moreover, in case of non-serverless application for the taxi booking system there is always overhead to manage the infrastructure, operating system, memory and other resources. It also requires I huge capital investment i.e., a high CapEx and OpEx is required, which are generally a threat in case of newly emerging startups and organizations.

OBJECTIVE (1 page)

Our objective is to develop a serverless taxi booking system using AWS services and deploying it over the cloud which will provide a highly scalable and reliable solution to the passengers.

Provide a pool of available taxis to the customer so that they can have a better choice and a good comparison between the other. It will enable the customer to book a ride by going through the previous rating and feedback of the customer for a particular driver and saves a lot of time to the customer by providing them a better ride at their footsteps.

It will enable the passenger to create an account by verifying it through their email phone number or google so that only valid and authenticated users can access it.

Once the user is successfully registered then they can login to our web application. After the successful login the user will be promoted to a geographical map and will be asked to choose a pickup location and a taxi from the pool of available once.

Finally, after the successful booking they can see a taxi approaching their pickup location.

The objective of our serverless architecture is also to reduce the overhead of the developers to configure the underline h/w, operating system, servers, network, and other resources and only focus upon coding and development part.

This serverless architecture will require a less CapEx and OpEx expenditure and hence will provide new opportunities to the startups and emerging organizations to compete in this industry so as to provide more reliable, cheaper and efficient solution to the customers.

RELATED WORK (6 pages)

1. Uber:

Uber is one of the most well-known taxi booking systems, and it is built on AWS. Uber uses a variety of AWS services, including EC2, S3, and RDS, to power its platform. AWS helps Uber to scale quickly and efficiently to meet the demands of its users.

Uber is running on a cloud server infrastructure that is provided by Amazon Web Services (AWS). AWS provides a range of cloud services that are used by Uber to power its platform and enable it to deliver a seamless experience to its users.

At a high level, Uber's platform consists of various applications and services that are deployed on AWS. These services are designed to handle various aspects of the Uber experience, such as ride requests, driver matching, payment processing, and more. AWS provides the underlying infrastructure and services needed to power these applications and services.

For example, AWS provides compute services such as Amazon Elastic Compute Cloud (EC2), which enables Uber to spin up virtual machines to run its applications. AWS also provides database services such as Amazon Relational Database Service (RDS), which enables Uber to store and manage data related to rides, drivers, and users.

In addition to compute and database services, AWS also provides a range of other services that Uber uses to power its platform. These include networking, storage, security, analytics, and more. AWS also provides tools and services that enable Uber to monitor and manage its infrastructure and applications, such as Amazon CloudWatch and AWS Management Console. Overall, by running on AWS's cloud server infrastructure, Uber is able to achieve scalability, reliability, and performance needed to meet the demands of its millions of users. AWS's cloud services also provide Uber with the flexibility, agility, and cost-effectiveness needed to continually improve and innovate its platform.

Uber has built its platform on top of Amazon Web Services (AWS), which provides a wide range of cloud services to support the company's various applications and systems. By leveraging the cloud, Uber is able to achieve the scalability, reliability, and performance necessary to meet the demands of its millions of users.

Cloud also provides Uber with the ability to quickly deploy and manage its applications and infrastructure. With AWS, Uber can easily spin up new instances of its applications, update software, and perform maintenance tasks with minimal downtime. This helps to ensure that the platform is always running smoothly and that users have a seamless experience.

One of the key benefits of using cloud for Uber is scalability. Uber's platform experiences significant spikes in demand during peak times, such as rush hour or during major events. By using AWS's auto-scaling feature, Uber is able to automatically adjust its computing resources up or down based on demand. This ensures that the platform can handle high volumes of traffic without any issues, while also reducing costs during slower periods.

In addition to scalability, cloud also provides Uber with the ability to quickly deploy and manage its applications and infrastructure. With AWS, Uber can easily spin up new instances of its applications, update software, and perform maintenance tasks with minimal downtime.

This helps to ensure that the platform is always running smoothly and that users have a seamless experience.

One of the key ways in which Uber uses cloud is to achieve scalability. Uber's platform experiences significant spikes in demand during peak times, such as rush hour or during major events. By using AWS's auto-scaling feature, Uber is able to automatically adjust its computing resources up or down based on demand. This ensures that the platform can handle high volumes of traffic without any issues, while also reducing costs during slower periods.

Cloud also provides Uber with the ability to quickly deploy and manage its applications and infrastructure. With AWS, Uber can easily spin up new instances of its applications, update software, and perform maintenance tasks with minimal downtime. This helps to ensure that the platform is always running smoothly and that users have a seamless experience

2. Lyft:

It is a leading ride-sharing service, is running on a cloud server infrastructure that is provided by Amazon Web Services (AWS), one of the leading cloud providers in the market. Lyft uses AWS to power its platform and provide a seamless experience to its users. Lyft's platform consists of a number of different applications and services that are deployed on AWS. These services are designed to handle various aspects of the Lyft experience, such as ride requests, driver matching, payment processing, and more. AWS provides the underlying infrastructure and services needed to power these applications and services.

One of the keyways in which Lyft uses AWS is to achieve scalability. Lyft's platform experiences significant spikes in demand during peak times, such as rush hour or during major events. By using AWS's auto-scaling feature, Lyft is able to automatically adjust its computing resources up or down based on demand. This ensures that the platform can handle high volumes of traffic without any issues, while also reducing costs during slower periods.

Another way in which Lyft uses AWS is to achieve reliability and availability. AWS provides a range of services designed to ensure that applications and services remain available even in the face of hardware failures, network issues, or other problems. For example, AWS provides data replication and backup services that help to ensure that data remains available even in the event of a disaster or outage.

AWS also provides a range of security features and compliance certifications, which helps to ensure that Lyft's data is secure and protected. This includes features such as encryption, identity and access management, and threat detection and prevention. AWS also provides built-in backup and disaster recovery features, which helps to ensure that data is always available in the event of an outage or other issue.

Finally, AWS enables Lyft to leverage advanced technologies such as machine learning and artificial intelligence. By using AWS's machine learning services, Lyft is able to optimize its platform by analyzing user behavior and other data. This helps to improve the user experience, while also increasing efficiency and reducing costs.

Overall, by leveraging the scalability, reliability, security, and advanced technologies provided by AWS, Lyft is able to provide a seamless and reliable experience to its users, while also managing costs and maintaining data privacy and security.

One of the keyways in which Lyft uses cloud computing is to achieve scalability. Lyft's platform experiences significant fluctuations in demand during peak times and geographic locations. Cloud computing enables Lyft to dynamically scale up or down its computing resources based on demand. Lyft uses AWS's auto-scaling feature to automatically adjust its computing resources based on traffic patterns. This ensures that Lyft's platform can handle high volumes of traffic without any issues, while also reducing costs during slower periods.

Another way in which Lyft uses cloud computing is to achieve reliability and availability. Cloud computing enables Lyft to leverage the infrastructure and services provided by AWS, which are designed to ensure that applications and services remain available even in the face of hardware failures, network issues, or other problems. Lyft uses AWS's data replication and backup services to help ensure that data remains available even in the event of a disaster or outage.

Lyft also uses cloud computing to improve security. AWS provides a range of security features, compliance certifications, and identity and access management services that help to ensure that Lyft's data is secure and protected. AWS's security features include encryption, threat detection and prevention, and built-in backup and disaster recovery features.

Finally, Lyft uses cloud computing to leverage advanced technologies such as machine learning and artificial intelligence. Lyft uses AWS's machine learning services to analyze user behavior and other data, which helps to improve the user experience, increase efficiency, and reduce costs.

Overall, by leveraging cloud computing services provided by AWS, Lyft is able to provide a seamless and reliable experience to its users, while also managing costs and maintaining data privacy and security.

3. Grab:

Grab, a leading ride-hailing and mobile payments platform in Southeast Asia, runs on cloud server infrastructure provided by various cloud providers, including Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). Grab uses cloud computing to power its platform and provide a seamless experience to its users.

Grab's platform consists of several different applications and services that are deployed on the cloud. These services are designed to handle various aspects of the Grab experience, such as ride requests, driver matching, payment processing, and more. The cloud providers provide the underlying infrastructure and services needed to power these applications and services.

One of the keyways in which Grab uses cloud computing is to achieve scalability. Grab's platform experiences significant spikes in demand during peak times, such as rush hour or during major events. By using cloud auto-scaling features, Grab is able to automatically adjust its computing resources up or down based on demand. This ensures that the platform can handle high volumes of traffic without any issues, while also reducing costs during slower periods.

One of the primary ways in which Grab uses cloud computing is to achieve scalability. Grab's platform experiences significant spikes in demand during peak times, such as rush hour or during major events. By using cloud auto-scaling features, Grab is able to automatically adjust its computing resources up or down based on demand. This ensures that the platform can handle high volumes of traffic without any issues, while also reducing costs during slower periods.

Another way in which Grab uses cloud computing is to achieve reliability and availability. The cloud providers offer a range of services designed to ensure that applications and services remain available even in the face of hardware failures, network issues, or other problems. For example, they provide data replication and backup services that help to ensure that data remains available even in the event of a disaster or outage.

The cloud providers also provide a range of security features and compliance certifications, which helps to ensure that Grab's data is secure and protected. This includes features such as encryption, identity and access management, and threat detection and prevention. The cloud providers also provide built-in backup and disaster recovery features, which helps to ensure that data is always available in the event of an outage or other issue.

Additionally, Grab is leveraging advanced technologies such as machine learning and artificial intelligence provided by cloud providers. By using machine learning services, Grab is able to optimize its platform by analyzing user behavior and other data. This helps to improve the user experience, while also increasing efficiency and reducing costs.

Another way in which Grab uses cloud computing is to achieve reliability and availability. The cloud providers offer a range of services designed to ensure that applications and services remain available even in the face of hardware failures, network issues, or other problems. For example, they provide data replication and backup services that help to ensure that data remains available even in the event of a disaster or outage.

The cloud providers also provide a range of security features and compliance certifications, which helps to ensure that Grab's data is secure and protected. This includes features such as encryption, identity and access management, and threat detection and prevention. The cloud providers also provide built-in backup and disaster recovery features, which helps to ensure that data is always available in the event of an outage or other issue.

Finally, Grab is leveraging advanced technologies such as machine learning and artificial intelligence provided by cloud providers. By using machine learning services, Grab is able to optimize its platform by analyzing user behavior and other data. This helps to improve the user experience, while also increasing efficiency and reducing costs.

Overall, by leveraging the scalability, reliability, security, and advanced technologies provided by cloud providers, Grab is able to provide a seamless and reliable experience to its users, while also managing costs and maintaining data privacy and security.

4. Ola

Ola, one of the biggest ride-hailing organizations in India, is utilizing distributed computing to control its foundation and give a consistent encounter to its clients. Ola utilizes cloud administrations given by various cloud suppliers, including Amazon Web Administrations (AWS) and Microsoft Purplish blue.

One of the essential manners by which Ola utilizes distributed computing is to accomplish adaptability. Ola's foundation encounters huge spikes sought after during busy times, like busy time or during significant occasions. By utilizing cloud auto-scaling highlights, Ola can naturally change its registering assets up or down in view of interest. This guarantees that the stage can deal with high volumes of traffic with next to no issues, while additionally lessening costs during more slow periods.

One more manner by which Ola utilizes distributed computing is to accomplish dependability and accessibility. The cloud suppliers offer a scope of administrations intended to guarantee that applications and administrations stay accessible even despite equipment disappointments, network issues, or different issues. For instance, they give information replication and reinforcement benefits that assistance to guarantee that information stays accessible even in case of a catastrophe or blackout.

The cloud suppliers likewise give a scope of safety highlights and consistence certificates, which assists with guaranteeing that Ola's information is secure and safeguarded. This incorporates elements like encryption, personality and access the board, and danger discovery and avoidance. The cloud suppliers likewise give worked in reinforcement and calamity recuperation highlights, which assists with guaranteeing that information is generally accessible in case of a blackout or other issue.

Moreover, Ola is utilizing trend setting innovations, for example, AI and computerized reasoning given by cloud suppliers. By utilizing AI administrations, Ola can improve its foundation by investigating client conduct and different information. This assists with further developing the client experience, while additionally expanding proficiency and decreasing expenses.

Notwithstanding these advantages, Ola is likewise ready to exploit distributed computing to rapidly and effectively convey new highlights and administrations. The cloud suppliers offer a scope of instruments and administrations that make it simple for designers to fabricate and send applications, which assists Ola with remaining in front of its rivals and meet the developing necessities of its clients.

In rundown, Ola is utilizing distributed computing to accomplish versatility, dependability, and security, while additionally utilizing trend setting innovations to further develop the client experience and streamline its foundation. By utilizing cloud administrations given by numerous cloud suppliers, Ola can guarantee that its foundation stays accessible and receptive to clients, while additionally keeping up with information protection and security.

WHY DO WE NEED CLOUD FOR THE PROJECT (5-6 PAGES)

Since we are creating a serverless web application, we do not want to have any overhead for maintaining and managing the infrastructure, setting up the networks, operating system, storage, database and other resources.

Under the serverless architecture, our main aim is to majorly focus upon the developmental of our application and quickly build and deploy our web application to the uses as soon as any changes is made over our source code repository without any down time or human intervention.

When we have such an ideology cloud computing is our first choice as it provides us with various services which can be used to completely deploy and host our web application.

As we have used AWS amplify to build and deploy our complete HTML, CSS and Java script based web application and create a CI/CD pipeline through code commit so, if we were not using the cloud-based solution, then we need to setup our own infrastructure, and need to manage the servers and the backend application due to which the developer could freely focus upon the development part only rather than taking the overhead of infrastructure. So, it will need to less CapEx and OpEx investment, so it provides an opportunity to the emerging startups and organizations to come up with creative and innovative features on such applications.

Also, amplify provides us with a public link through which the customer can access our website, so we do not need to purchase a public DNS for our website in the initial phases and hence is cost effective. Also, we needed to use our on-premises server and need to be worry about to setup the proper environment to run our code and manage the physical servers.

Secondly, our application requires the authentication of the user which we have done with the help of Cognito service. So, if we have not used this cloud-based solution then we have to use on premise strategy for Authorization and authentication of the genuine users. Using on premise strategy for such task would have been really a complex and challenging task to perform (or merely impossible to be done for small Startups) as we need high IT experts and professionals to write and manage the complex security algorithms which would adversely require high investment.

Moreover, these user data are highly critical and confidential, so in case of on-premises strategy, managing and securing these data will create a burden for the developers and really be a challenging task or merely impossible for startups.

Next, we are required with a NoSQL database which will store the data in the form of rows and columns with high scalability. It is highly scalable so we only need to pay for storage we will use as per our traffic. Whenever any request from the passenger will come to book a taxi the booking, the booking details will be stored in the dynamo DB by invoking a function.

So, in our previous days, with low Expenditure cost, we do not need to think for the overhead of storage capacity.

So in the case of non-cloud solution, we need to purchase and setup our own database need to higher some database engineers look for all the management and storage of data.

The customers data are critical, so to maintain the privacy of the customer we need to set up certain mechanism for the security and redundancy of our database which in turn increases the overhead for the developers and the cost of investment as well.

Next, we have used lambda function which helps us to run our code on high availability compute resources. It is event triggered and has the capacity to automatically scale.

With the help of lambda function on cloud, we are making our application as serverless. The lambda function also provides us the 1 million of instruction for free of cost and even responsible of scaling of resources in case of sudden increase in high request. Whereas, it was very difficult to process, manage and respond to a million of instructions in case of on premise setup and would have required high capex and opex with high skilled IT professionals.

In our project, lambda function will respond to the events and the user requests such as http requests. Our lambda function will process the request from the customers that is mainly the API request to book a taxi.

Whenever a request from the user will come for the booking of a taxi the lambda function will be triggered by interacting with API calls and then it will automatically trigger The dynamo DB database which will store the booking details of the users to the dynamo DB table in the form of rows and columns. So if we would have not used the cloud based solution then we need to look for some manual solutions for the booking of a taxi fare as by using a cloud base solution we have automated the task using the event with triggers which is also more efficient, reliable and fast as compared to the manual works.

Also, the manual task would have been less reliable and would have required a large amount of time to perform the same task and it would lead the inconsistency and inconvenience to the customer.

Also, we have used the Amazon API Gateway to build and deploy a rest full API which can be accessible to our customer through the public internet. This API is essential for interacting the customers with cloud-based services and deals with the user request in the form of http calls.

If we have not used this cloud-based solution, then we would have needed to create our own API which might not be fully efficient and secure. On the other hand, we would also have the overhead to maintain and manage the API in case of on-premises architecture.

But in a cloud-based scenario we have secured the API with Cognito service and the maintenance and management of the API is done by AWS itself as it comes under the managing responsibility of the cloud service provider.

So, it reduces our time effort and increases the reliability efficiency and efficiency of our application.

Also, our serverless architecture requires a cloud watch for the maintaining of logs, metrics and monitoring over the proper functioning of the other services or the number of services used, pricing generated, resources underutilized, total no of requests handled or the number of IOPS made.

CloudWatch keeps on tracking the services based upon the metrics and sends us the alert alarm if any matrix is reached beyond its limit or the failure of any services occurs.

CloudWatch also helps us to achieve the automation as we do not require a manual intervention for it and get the quick and certain alarms for any such cases.

Whereas in case of on premise we need to set up an engineer for properly doing the monitoring task or need to write some algorithm to automate the monitoring. But in the case of on-premises monitoring is not an easy task as we need to monitor for each fraction of time.

Also, in the case of startups and emerging organizations, setting a dedicated engineer for monitoring, or writing their own algorithms for them monitoring is nearly impossible as it requires a huge amount of time, IT professionals and high expenditure.

Moreover, monitoring in case of on premise cannot be as efficient, and reliable as compared to cloud-based solution.

Using cloud services like AWS can provide a serverless taxi booking system with scalability, reduced infrastructure costs, high availability, security, and integration with other AWS services. These benefits can help ensure the success of the system and provide a better experience for both the customers and the service providers.

Also, Simple email service is also used for the authentication part. Each time A user registers to our application a verification code is generated via Amazon simple email service and is sent to the users entered mail address, and user needs to be submitted the verification code to prove the user authenticity and create a user account.

So, if we have not used cloud-based solution then we would have required to set up a dedicated host, engineer, infrastructure and a network for the same process. Also, the task would not have been automated as efficiently as compared with cloud and would have required more amount of time for the security code to be generated from the server and reaches the customer.

The late generation, delivery and verification of the security code would have been so common in case of non-cloud solution, and it would have really created an in ample number of convenience and bad user experience.

The above-mentioned benefits and the features enable developers to focus on developing the code and deploying code quickly and efficiently without worrying about managing the underlying infrastructure. The code written by the developer will be more efficient, smooth, and responsive which in more reliable, cost-effective, and scalable applications and provide a better user experience to the customers.

It also enables the developers to update with the changing trend and need of the market and focus on building up new features so as to increase the user experience as per the changing market need and compete with other providers to hold the customers for future.

Overall, using cloud computing for hosting a serverless application on AWS provides benefits such as automatic scaling, reduced cost of infrastructure, high availability, quick market delivery, ease to integration with other services, and enhanced security. These benefits enable developers to build more reliable, cost-effective, and scalable applications. These benefits can help ensure the success of the system and provide a better experience for both the customers and the service providers.

We have discussed the above advantages and the features of the services used in our application architecture are very much essential and crucial for server list taxi booking system. We have discussed the problems arrows in case of on-premises architecture rather than using a cloud based solution and approach.

Some of the other features and the advantages for using the cloud to build and deploy our web application are:

1. Cost of infrastructure

It is very clear that using a cloud-based solution for our serverless application we do not need to think for the infrastructure.

The system would not require a dedicated set of servers and resources to handle the expected traffic. However, during quiet periods, the resources can easily be scaled up or down depending upon the need and requirement. With a serverless architecture, the system only charges for the actual usage of the service, so during periods of low traffic, the cost of running the system would be significantly reduced. Additionally, it eliminates the need to manage and maintain infrastructure, further reducing operational costs.

The CSP charges the developer based upon the number of times they have used the services or the amount of execution for the deployed code has been invoked, so it also provides the features of pay as you use for the developers and provide a chance for the emerging startups and new organisations to come of with their innovative features and provide a better experience to the customer.

2. Quick delivery to the market

With the help of serverless architecture on cloud the developer needs to only focus upon the development of their application building new features detecting box and improving the quality of their code. Cloud enables us to quickly build and deploy to the any changes made by the developer within some fraction of time and hence it provides the updated features to the customers as soon as any change is made to the production server.

Nowadays we have some competitors on every offering over the internet, so customer requires the latest solution and features to their application or simply they will shift to another application or organization for the booking of taxi so the real time provisioning, building and deployment of cloud computing enable us to quickly provide new features to the customers so that they could have a good user experience and retain to use our application for the future purpose as well.

3. Automation

The cloud-based approach helps us to achieve the automation of the task. The lambda Function used in our serverless architecture works on the event-based triggers and whenever the sudden traffic increases or decreases as per the preferred expected users it can quickly provision or deprovision the resources required to handle the traffic with minimal cost.

We have also achieved the automation for authentication using Cognito user pool and identity, the booking details are automatically stored over the dynamo DB with the help of API gateway.

Also, when any change is made in our source control repository or the production server, with the help of CICD pipeline created it automatically triggers the amplify to provision, build, and deploy our updated application code and provide real time delivery to the customers.

4. High availability

In case of on-premises solution the failure of any service or server leads to the failure of other services as well and hence the failure of the complete application. Also, in case of on my solution it's not easy to handle the failures or protect the use a data.

But when we talk about the cloud base solution it is highly available and provides a fault tolerant feature to our application, the cloud based solutions are highly resilient and The cloud based solutions help us to create the multiple replica of our application and codes and also ensures us the feature of disaster recovery management in case of natural calamities like flood, earthquake, attacks etc. which help us to regain our application code and continue with our application to provide solution to the customer.

This feature increases the trust of the customer for any organisation and attracts more customers towards them it also help them to retain their customer for a longer duration of time and have a good customer feedback and reviews for their organisation.

5. Security

We have already discussed that the CSP offers a wide range of security features, it provides are the feature of key management identity and access management encrypts the data address and data in transit.

For example: we have used Cognito to add on the security feature to our application so as to filter and control the engagement of valid and genuine users to our application.

Both on-premises and cloud deployments require proper network security measures such as firewalls, DNS, Route 53, and VPNs. Whereas, in a cloud deployment, the cloud provider typically provides network security services that can be easily integrated into the application, such as VPC or Azure Virtual Network.

On the other hand, data security is critical for any application, and both on-premises and cloud deployments require proper measures such as role based access controls, encryption and backups. In a cloud deployment, the cloud provider typically provides various security services such as AWS Identity and Access Management (IAM), AWS Key Management Service (KMS), and Certificate Manager, which can be used to manage access to the system, encrypt data in rest and in data in transit, and manage SSL certificates.

6. Easy integration of the services

Cloud offers easy and wide range of integration with the services. It is very easy to navigate and connect to different services used to work together for an application.

For example: in our application the services like API gateway, Cognito, code commit, dynamo DB lambda, cloud watch and simple email services are easily integrated with each other mainly with the help of lambda to achieve that desire and proper functioning of our web application.

In the case of on-premises solutions the integration of services plays challenging task.

Moreover, cloud integration offers a wide range of scalability maintenance and cost effectiveness when compared with the on-premises integration of the services.

In case of on premise we often require huge IT professionals and experts with the deep domain knowledge for the integration where the failure of one task main lead to the failure of other and hence the failure of entire application as well.

Overall, using cloud computing for hosting a serverless application on AWS provides benefits such as automatic scaling, reduced cost of infrastructure, high availability, quick market delivery, ease to integration with other services, and enhanced security. These benefits enable developers to build more reliable, cost-effective, and scalable applications. These benefits can help ensure the success of the system and provide a better experience for both the customers and the service providers.

METHODOLOGY (4-5 pages)

At the very first we developed our web application, which is done by following agile and develops methodologies like scrum i.e. defined a product backlog, divided the task into multiple sprints, defined sprint backlog and then worked on each sprint to develop our application.

We have used HTML, CSS, and JavaScript for developing our web application. HTML is used for the structuring of our website while we have used CSS for the styling part and Java script so as to introduce the logic of backend and functional behavior of our web application. Based on the requirements and use cases, design the architecture of the system. The architecture should be scalable, fault-tolerant, and secure. It should utilize serverless technologies such as AWS Lambda, API Gateway, and DynamoDB to provide a cost-effective and efficient solution.

Then we have done the requirement gathering and understood the change in the trend of transportation and requirement as per the customers as well as drivers. We have identified the new features we can add to our web application so as to provide a good experience to the customer like booking the ride, tracking the ride, ratings of the driver, choosing a ride from a pool etc.

Then we have understood about the cloud-based services which will be used to create our architecture, like dynamo DB for NoSQL database, Amazon S3 for file storage, Cognito for authentication, code commit for source code management, CloudWatch for the monitoring of logs, a lambda function for event-based triggers and API gateway to handle the post request from the website.

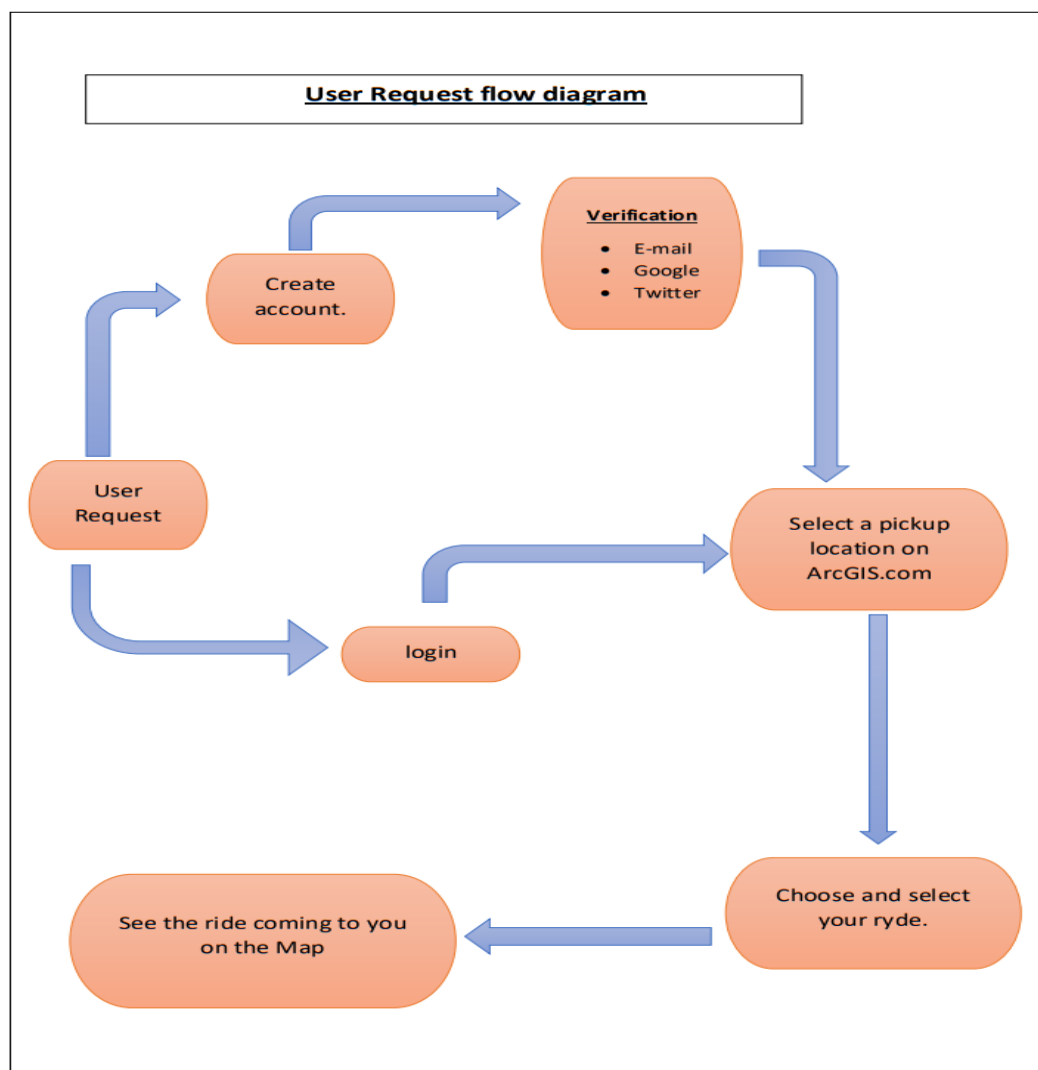
Once we had done with designing the architecture, we have moved forward with the implementation which involves the setting up a AWS account, creating IAM user, configuring lambda function, rest full API using API gateway, setting up Cognito user pool and identity pool, AWS code commit for source code management and dynamo DB for storing the details of the rides booked. The Lambda functions should handle the business logic of the system, such as booking a ride, managing ride requests, and tracking the ride in real-time. The DynamoDB tables should store the data related to customers, drivers, rides, and ride history.

To ensure optimal performance and detect any issues or errors, it is important to monitor the serverless taxi booking system on AWS cloud. This involves setting up monitoring tools to track the system's performance metrics, such as response times, error rates, and resource utilization. Regularly reviewing these metrics can help identify any issues that may arise and allow for quick resolution.

As demand for the taxi booking system grows, it may be necessary to scale the resources to handle increased demand. This can be done by adding additional Lambda functions, increasing the capacity of DynamoDB tables, or increasing the size of the API Gateway instance. By monitoring the system and scaling the resources as needed, the serverless taxi booking system can continue to provide a reliable and efficient service to customers and drivers alike.

Then we have finally moved with the testing and deployment of our web application using AWS Amplify. We have created a CI/CD pipeline using by integrating our Code Commit and Amplify so that if any change occurred from the developer end, then without any manual interruption and with zero downtime, we could provide the updated web application with added features to our customers.

Now let us discuss the user request flow diagram and the complete features of our application. Whenever the user visits our application for the very first time, they are required to register upon our application so that only the genuine and authorized user could move further to book a taxi which will increase the security, resilience, reliability and affordability of the taxis. So, at very first the user will be required to create an account by either using their email or phone number.



{ fig→ 1 user request flow diagram }

Once the user has entered a valid email or a phone number, Cognito will check the validity of details and then a verification code using Amazon simple email service is sent to the users' mail or the phone number. The user is then required to enter the verification code so must prove his authenticity and then successfully logged in. The user can also simply login through their Google account. After, a successful login is done the user will be promoted with a geographical map of their location. The user can navigate through the map as per his or her wish and can point to a place to select his or her pick up location.

The user will then asked to choose a taxi from the pool of available ones and can choose it by comparing the pricing, feedback of the driver and the type of taxi he/she wants. Once the user had successfully done with all the steps and confirm the booking, he will see a prompt which will say, a taxi is approaching towards you location and he can see a taxi approaching towards his location and the amount of time it will be required to approach the pickup location.

As a whole, the serverless taxi booking system's UI-based architecture comprises a user interface for customers and drivers, API Gateway, AWS Lambda, Amazon DynamoDB, Amazon S3, Amazon CloudFront, Amazon Cognito, Amazon SNS, Amazon SQS, and AWS CloudFormation. This design offers a cost-effective, scalable, and dependable solution for booking taxi services.

The AWS service his which we will be using to completely deploy our web application and built the serverless architecture code commit amplify Cognito dynamo DB lambda function API gateway and CloudWatch.

Let us discuss the internal architecture of our application.

- 1. AWS Code Commit:** It is a service offered by AWS used for storing our source code and repository files. It is a completely managed by AWS and is like a private gift depository which is highly scalable and reliable.

It enables us to keep a track of the previous versions of our website for web application and also allows us to shift to the previous version whenever required. It is very easy to use and integrate with the other services like code pipeline, code deploy, code build.

We have used code commit for version control and source code management of our repository files and to create a CI/CD pipeline with Amplify to build and deploy our application code with zero down time and no manual intervention.

- 2. AWS Amplify:** it is a service offered by AWS which helps the users to build and deploy a full-Stack application with highly scalability and security.

Amplify provides a wide range of services, including authentication, APIs, storage, analytics, hosting, and CI/CD, all integrated into a single platform. These services allow developers to add functionality to their application seamlessly and efficiently, reducing development time and costs.

So, in our application we have used AWS amplify and connected it with code commit to create a continuous integration and continuous deployment pipeline. Whenever any changes are made from our developer to code commit, the amplify service suddenly comes into play, build and deploy our application file.

- 3. AWS Cloud Watch:** CloudWatch is an easy to use monitoring service offered by AWS which is easily integrated with all services. It is used for logs management, collecting matrices, setting alarms. Monitors our application and send a alert alarm in case of over CPU utilization, disc uses, network utilisation or other resources.

In our application we have use CloudWatch to maintain the logs and monitor the API invokes and post request it also monitors over the proper execution of lambda function and event-based triggers.

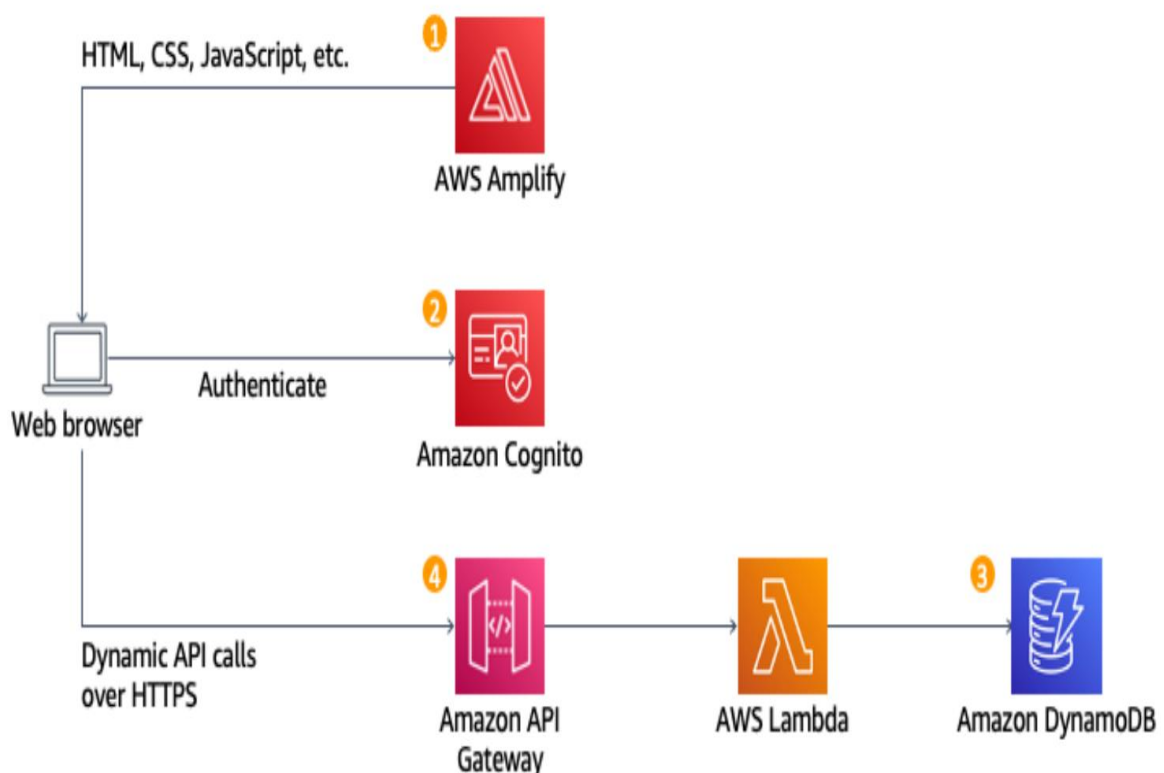
4. **AWS Cognito:** It is a service offered by AWS which helps us for Authentication and authorisation of the users accessing our website or application. It is fully managed service used for identity management and user authentication. It helps developers add user sign-up, sign-in, and access control to their applications easily and securely. Cognito provides a set of features that enable developers to create and manage user authentication and identity-related tasks with minimal coding and infrastructure setup.

One of the primary benefits of Cognito is its flexibility. Developers can choose from several authentication methods such as username/password, social identity providers (such as Facebook and Google), and multi-factor authentication (MFA). Cognito also supports custom authentication flows, enabling developers to implement their own authentication logic and workflows.

Cognito maintains user pool and identity pool. Under user pool, it maintains the data of the user and is responsible for authentication whereas identity pool will provide the permissions (authorization) to the users to after the successful authentication is done.

Whereas, if we have used on premise strategy, then the authorisation and authentication processes are complex and challenging to perform (or merely impossible to be done for small Startups).

Hence, it increases the security of our website and easily filters the unknown (unauthorized users).



{ Fig→2 our serverless architecture }

- 5. DynamoDB:** It is the fully managed NoSQL database provided by the AWS to store the data in the form of rows and columns with high scalability. It is highly scalable so we only need to pay for the amount of storage we will use as per our traffic.

So, in our previous days, with low Expenditure cost, we do not need think for the overhead of storage capacity.

Each time I user will request to book arrived a lambda function will be triggered and the request will be send to The dynamo DB table.

So after the booking of each ride the data for the ride will be sent to The dynamo DB table and stored over it.

- 6. Lambda function:** it helps us to run our code on high availability compute resources. It is event triggered and has the capacity to automatically scale.

with the help of lambda function on cloud, we are making our application as serverless. The lambda function also provides us the 1 millions of instruction for free of cost and even responsible of scaling of resources in case of sudden increase in high request. Whereas, if was very difficult to process, manage and respond to a million of instructions in case of on premise setup and would have required high capex and opex with high skilled IT professionals.

An IAM role is already associated with every lambda function which define what other AWS services this lambda function is allowed to access and interact with.

Since we are using CloudWatch and dynamo DB, so we will be defining ruoes and policies for the lambda function to read and write the items over the CloudWatch and dynamo DB table.

ALGORITHM (2-3 pages)

Developing and deploying a taxi booking system on the cloud involves the use of various algorithms to ensure optimal performance and efficiency. Some of the algorithms that can be used include:

1. **Routing algorithms:**

Routing algorithms are used to calculate the most efficient route for a taxi to take from its current location to the passenger's pickup location and then to the destination. These algorithms take into account various factors such as traffic conditions, distance, and time of day to determine the optimal route.

Routing algorithms are an essential component of developing and deploying a taxi booking system on the cloud. These algorithms are responsible for calculating the most efficient route for a taxi to take from its current location to the passenger's pickup location and then to the destination. There are several different types of routing algorithms that can be used, including:

1.1 Dijkstra's Algorithm: Dijkstra's algorithm is a popular routing algorithm that is used to find the shortest path between two points in a graph. In the context of a taxi booking system, this algorithm can be used to find the shortest route between a taxi's current location and the passenger's pickup location, taking into account factors such as traffic, road conditions, and distance.

The basic idea behind Dijkstra's algorithm is to start at the initial vertex and explore all its neighboring vertices, updating the distances of each vertex as it goes along. It then selects the vertex with the minimum distance and repeats the process until the destination is reached. The algorithm uses a priority queue to keep track of the vertices with the smallest distance.

In the context of a taxi booking system, Dijkstra's algorithm can be used to find the shortest path between the taxi's current location and the passenger's pickup location. This algorithm takes into account various factors, such as the road network, traffic conditions, and the distance between the two points.

The main advantage of Dijkstra's algorithm is that it guarantees finding the shortest path between two points in a graph. However, it can be computationally expensive when used on large and complex graphs, which can be an issue in some taxi booking systems that need to process many requests simultaneously.

Despite its limitations, Dijkstra's algorithm remains a valuable tool for developing and deploying taxi booking systems on the cloud. By selecting the most appropriate routing algorithm for the specific needs of the system, developers can ensure that the system is optimized for efficiency and accuracy, resulting in a better experience for passengers and drivers alike.

2. Pricing algorithms:

Pricing algorithms are used to calculate the fare for each ride based on factors such as distance, time of day, and demand. These algorithms use machine learning techniques to analyze past data and predict future demand, allowing the taxi booking system to adjust prices dynamically to maximize revenue. The pricing algorithm calculates the fare based on the distance traveled, time taken, and other relevant factors such as toll charges, peak hours, and demand. It is crucial for the pricing algorithm to be accurate and reliable so that both the passenger and the driver are satisfied with the fare charged.

In a cloud-based taxi booking system, pricing algorithms can be continuously optimized using machine learning and data analytics to ensure that the fares are competitive, fair, and reflective of the actual cost of the ride. For example, by analyzing data such as weather, traffic conditions, and historical demand, the algorithm can automatically adjust fares in real-time to ensure that the supply and demand are balanced and that the pricing is fair.

Pricing algorithms are critical for ensuring the profitability of a taxi booking system, as they directly impact the revenue generated by the system. Therefore, it is essential to develop and deploy a pricing algorithm that is transparent, reliable, and meets the needs of both passengers and drivers.

In conclusion, pricing algorithms are a crucial component of developing and deploying a taxi booking system on the cloud. By leveraging advanced machine learning and data analytics techniques, pricing algorithms can be continuously optimized to ensure that the fares are competitive, fair, and reflective of the actual cost of the ride. This helps to ensure the profitability of the system while providing a reliable and convenient service to passengers and drivers alike.

3. Recommendation Algorithm:

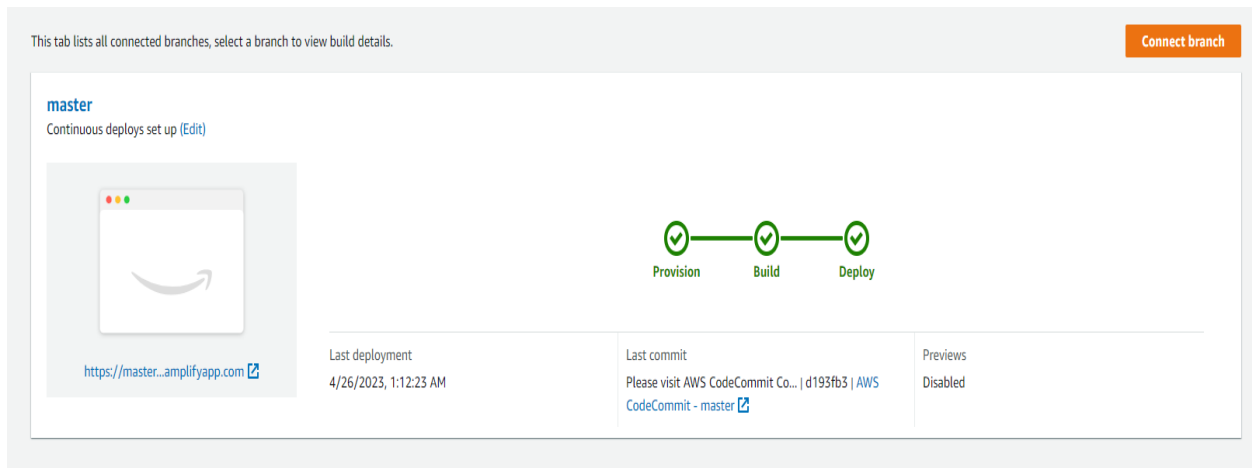
Recommendation algorithms are an essential component of developing and deploying a taxi booking system on the cloud. These algorithms help to provide personalized recommendations to passengers based on their previous ride history, preferences, and behaviour. The recommendation algorithm uses machine learning techniques to analyze data such as ride history, location, and user behavior to suggest relevant options to the passenger. For example, the algorithm can recommend preferred routes, popular destinations, and even preferred drivers to the passenger.

In a cloud-based taxi booking system, recommendation algorithms can be continuously optimized using real-time data analysis to ensure that the recommendations are personalized, accurate, and relevant. For example, by analyzing data such as passenger behavior, ride history, and location data, the algorithm can provide personalized recommendations to each passenger.

Recommendation algorithms are critical for ensuring customer satisfaction and loyalty. By providing personalized recommendations to passengers, the taxi booking system can enhance the user experience and build customer loyalty.

RESULT AND DISCUSSIONS (8-10 pages)

With every change, an auto provision, build and deploy is triggered in Amplify.



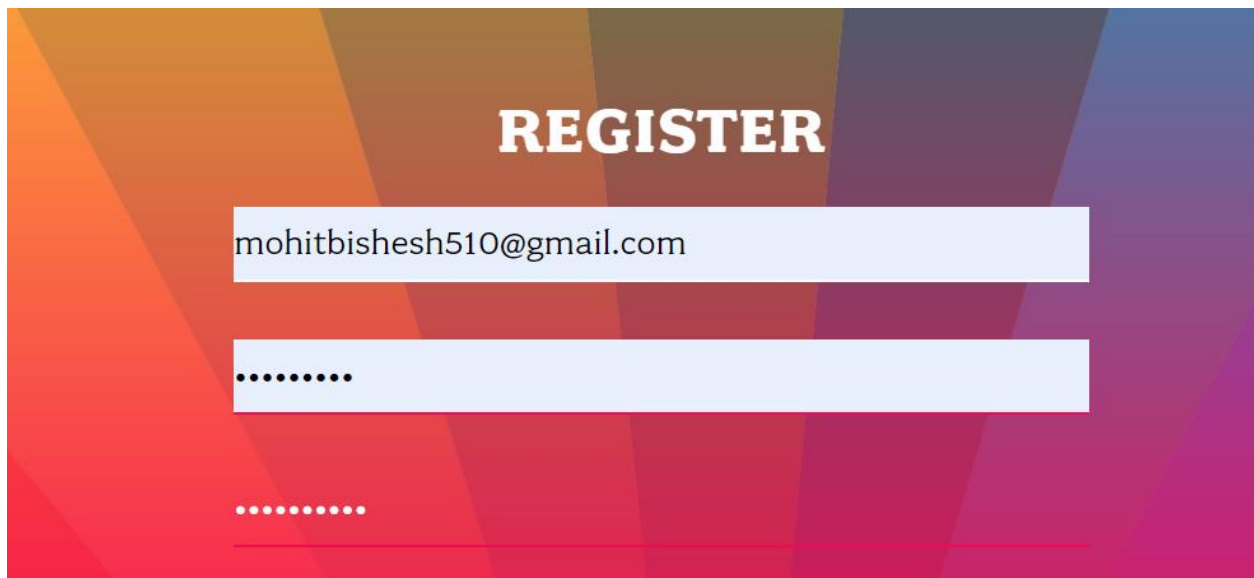
{ Figure 3 }

Open the url shown to navigate your website. We can clearly see our website and navigate through it .



{ Figure 4 }

Click on signup and register. Enter your credentials to register.

A registration form titled "REGISTER" in bold white text on a colorful geometric background. It features three input fields: the first contains the email "mohitbishesh510@gmail.com", the second contains masked characters ".....", and the third contains masked characters ".....".

REGISTER

mohitbishesh510@gmail.com

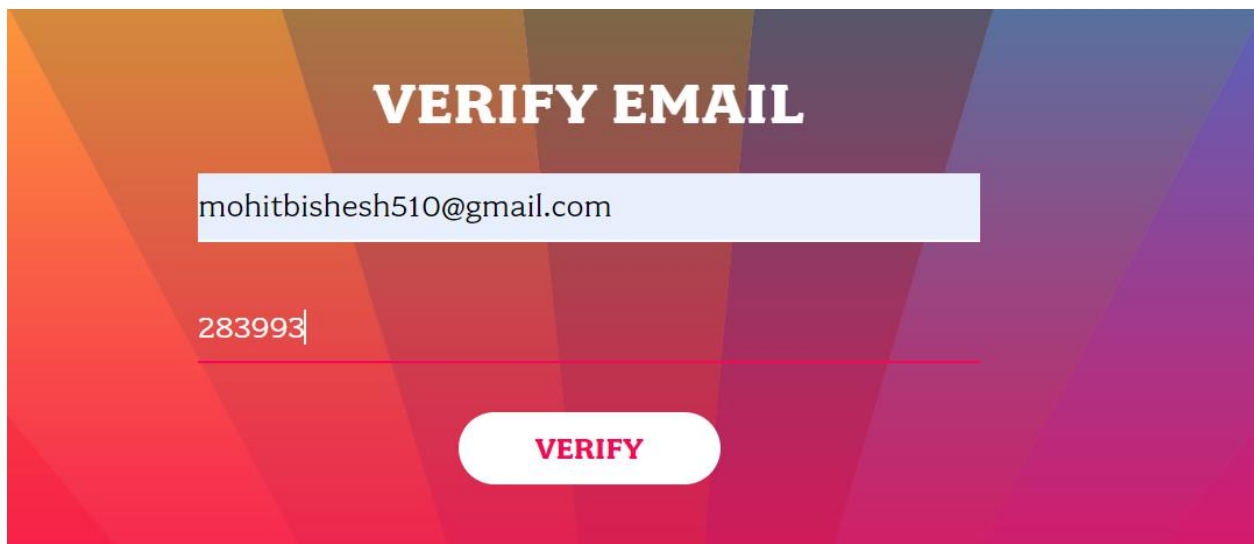
.....

.....

{Figure 5}

A verification code is sent to your mail. Enter the code to verify.

We have successfully received the Verification code in our email address that is our cognito pool is working well and fine.

A verification form titled "VERIFY EMAIL" in bold white text on a colorful geometric background. It features two input fields: the first contains the email "mohitbishesh510@gmail.com" and the second contains the verification code "283993". Below the fields is a red "VERIFY" button.

VERIFY EMAIL

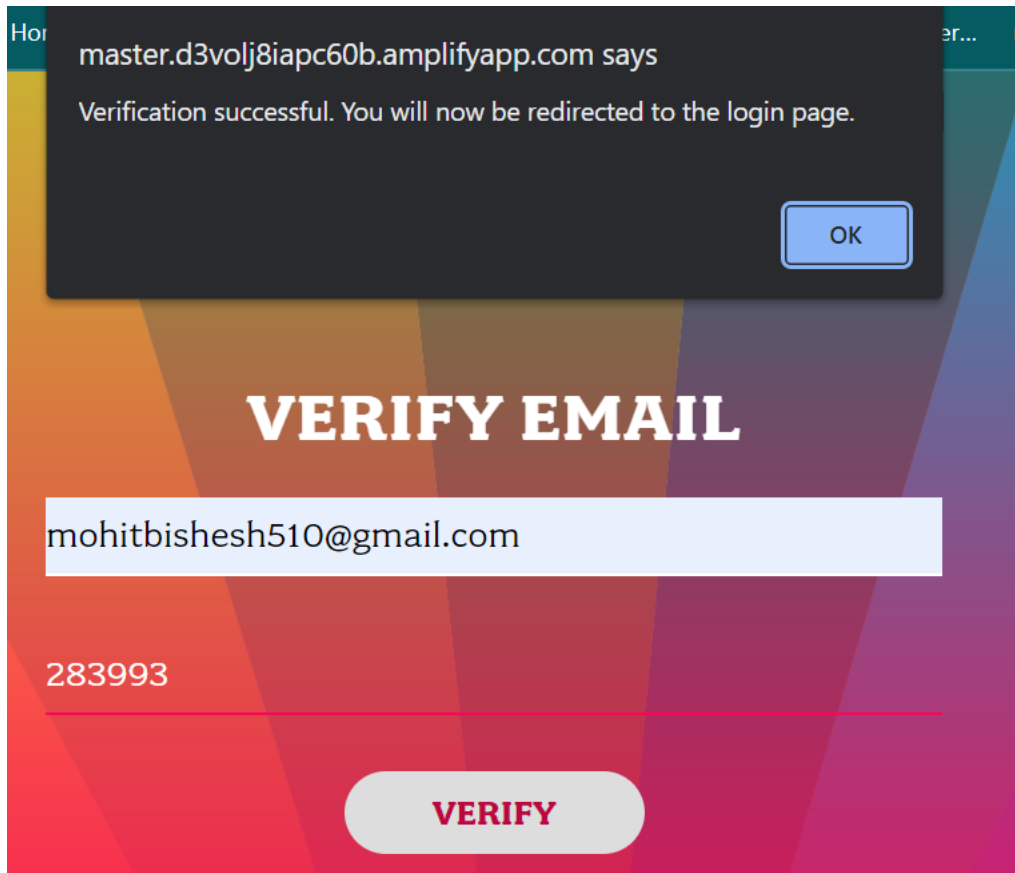
mohitbishesh510@gmail.com

283993

VERIFY

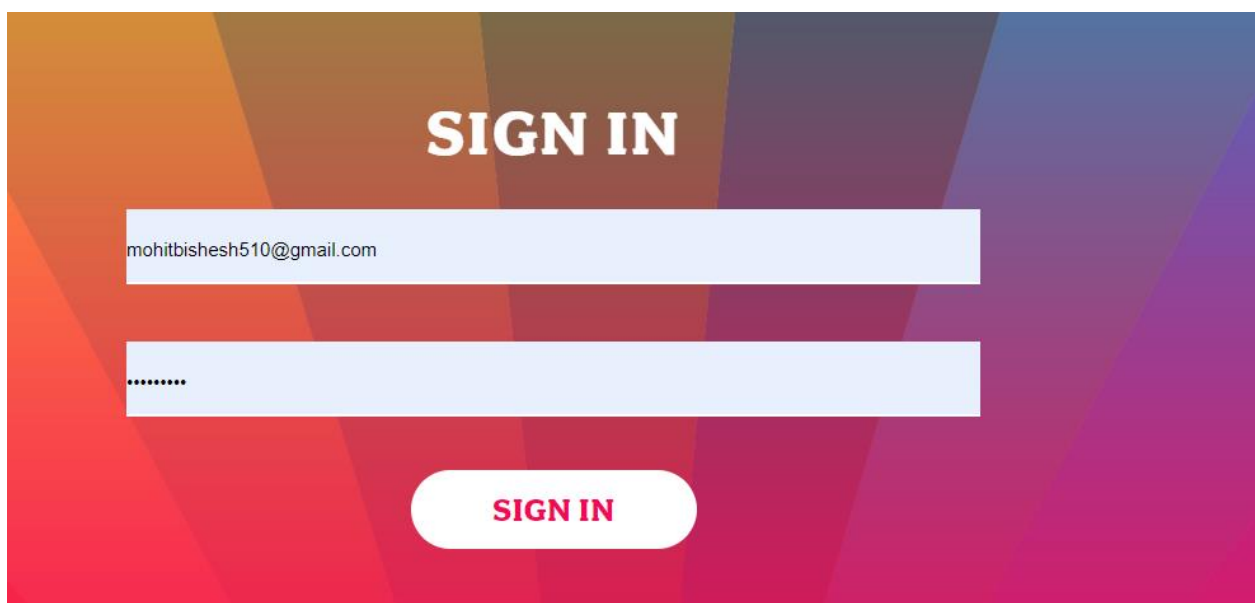
{Figure 6}

A verification successful prompt appears.



{Figure 7}

Enter your credentials to sign in.



{Figure 8}

A geographical map as per your location is appeared.



{Figure 9}

Select on the map to choose a pickup location.



{Figure 10}

As discussed earlier, the deployment of a taxi booking system on AWS cloud offers various benefits such as scalability, cost savings, high availability, advanced security features, easy accessibility, and global reach. The application leverages the advantages of cloud computing to provide a reliable and scalable solution for taxi booking services. In this section, we will discuss the results and the discussion of deploying the taxi booking application on the AWS cloud.

Results:

- Scalability:

The AWS cloud provides scalability features that enable the taxi booking application to handle an increased number of users during peak hours. This ensures that the application can handle a large number of requests and maintain its performance.

A taxi booking system needs the on-demand scalability that public cloud infrastructure offers. The system must be able to scale up or down as necessary because it must manage different levels of demand. Flexible scaling solutions are available from public cloud providers, enabling the system to add or remove resources in real-time according to demand. This means that the system can scale up to handle a lot of requests during peak times, and it can scale down to save money during off-peak times.

- Cost Savings:

By leveraging AWS cloud services, the taxi booking application can reduce the infrastructure cost associated with the traditional on-premise solution. The cloud allows the application to pay only for the resources that are being used.

Taxi reservation systems can be deployed on public cloud infrastructure for considerable cost savings. Pay-as-you-go pricing models are available from public cloud providers, allowing businesses to only pay for the services they really utilise. As a result, businesses won't need to spend money on pricey gear, software, and upkeep. Additionally, long-term users can receive discounts from public cloud providers, which can result in long-term cost savings for businesses. The majority of public cloud service providers adopt a pay-per-use pricing structure, where customers only pay for the services they actually use. For taxi reservation systems, the elimination of the need to buy and maintain pricey hardware and software infrastructure can result in significant cost savings. Users only pay for the resources they really use using pay-per-use pricing, which may be adjusted up or down in response to changing demand. Users are exempt from paying for idle resources, which has a considerable financial advantage.

- High Availability:

The taxi booking application is deployed across multiple availability zones in the AWS cloud. This ensures that the application remains available even in case of a hardware failure or a network issue. When using an on-premises solution, the failure of one server or service causes the failure of additional services, and ultimately the entire application. It's also difficult to manage faults or safeguard the use of data when using my solution.

However, when it comes to cloud-based solutions, they are extremely resilient, offer a fault-tolerant feature to our application, and are highly available.

In addition to providing us with the function of disaster recovery management in the event of natural calamities like flood, earthquake, attacks, etc., cloud-based solutions enable us to build multiple replicas of our application and codes. This allows us to recover our application code and carry on with our business.

In addition to providing us with the feature of disaster recovery management in the event of natural calamities like flood, earthquake, attacks, etc., cloud-based solutions enable us to create multiple replicas of our applications and codes. This enables us to continue using our applications to provide solutions to customers.

- Advanced Security Features:

The AWS cloud provides advanced security features such as network security, data encryption, and identity and access management. The taxi booking application leverages these features to ensure that the user's data is secure.

Multi-factor authentication (MFA) is one of the most important security technologies made available by public cloud service providers. Before gaining access to the system, MFA requires users to submit two or more kinds of authentication, such as a password and a fingerprint. This adds an extra layer of security on top of conventional usernames and passwords, making it more challenging for hackers to access the system without authorization. Encryption is yet another essential security element provided by public cloud service providers.

To secure data from unauthorised access, encryption services are used to encrypt data both in transit and at rest. Any comprehensive security solution must include encryption as it adds an additional degree of security. Access controls are another essential security element that public cloud service providers offer. Administrators can limit access to resources based on roles and permissions by using access controls. This minimises the chance of data breaches and other security issues by ensuring that only authorised users may access the resources they need to do their work.

- Easy Accessibility and Global Reach:

The taxi booking application is accessible from any location and can be used by anyone with an internet connection. The application can also be accessed from different devices, including smartphones, tablets, and laptops.

The public cloud also offers a variety of tools and services that can assist broaden accessibility and reach across borders. No matter where they are in the world, clients should be able to easily and quickly access the taxi booking system thanks to the use of content delivery networks (CDNs), which may be used to cache content in data centres all over the world. This enhances performance and dependability, much to how traffic can be distributed around several data centres using global load balancers.

And finally, it is easy to maintain and keep track of the taxi booking system thanks to the administrative tools and APIs that public cloud providers frequently offer. This allows taxi reservation services to measure usage promptly, monitor system performance, and identify areas that require improvement, guaranteeing that the

Discussion:

The deployment of the taxi booking application on the AWS cloud provides several advantages. By using the cloud-based solution, the application can be scaled up or down to meet the demands of the users, which is essential during peak hours. This ensures that the application remains available and provides a seamless experience to the users.

Moreover, by using the AWS cloud, the taxi booking application can reduce the infrastructure cost associated with the traditional on-premise solution. This can be attributed to the pay-as-you-go pricing model, where the application pays only for the resources that are being used.

The deployment of the taxi booking application across multiple availability zones in the AWS cloud ensures high availability, which is critical for a taxi booking system. This ensures that the application remains available even in case of a hardware failure or a network issue.

The AWS cloud also provides advanced security features such as network security, data encryption, and identity and access management. By leveraging these features, the taxi booking application can ensure that the user's data is secure.

Finally, the deployment of the taxi booking application on the AWS cloud provides easy accessibility and global reach. The application can be accessed from any location and can be used by anyone with an internet connection. This enhances the user experience and ensures that the application is available to a large number of users.

In conclusion, the deployment of the taxi booking application on the AWS cloud provides several benefits, including scalability, cost savings, high availability, advanced security features, easy accessibility, and global reach. These advantages ensure that the application remains available, secure, and provides a seamless experience to the users.

One of the key advantages of deploying the taxi booking application on AWS cloud is the ability to leverage the various AWS services for scalability, reliability, and security. The use of services such as EC2, S3, RDS, and Lambda makes it easier to deploy and manage the application. In addition, the use of auto-scaling and load balancing ensures that the application can handle traffic spikes and maintain high availability. The integration of cloud watch for monitoring and logging also makes it easier to identify and troubleshoot issues.

Another advantage of using AWS cloud is the flexibility it offers in terms of deployment options. The application can be deployed in a variety of ways, such as using Elastic Beanstalk, Kubernetes, or ECS. This provides the ability to tailor the deployment to meet specific requirements for the application. For example, Elastic Beanstalk offers a managed platform that simplifies the deployment process, while Kubernetes offers more granular control over the deployment.

The use of microservices architecture also offers several benefits for the taxi booking application. It allows for greater flexibility and agility in development, as changes can be made to individual services without affecting the entire application. It also makes it easier to scale the application horizontally by adding more instances of specific services. However, it does add complexity to the overall architecture and requires careful consideration of service boundaries and communication. In terms of pricing, AWS offers a pay-as-you-go model that allows for cost optimization. The use of reserved instances and spot instances can also help to reduce costs further. However, it is important to monitor usage and adjust as needed to avoid unexpected expenses.

Overall, deploying the taxi booking application on AWS cloud provides several advantages in terms of scalability, reliability, security, flexibility, and cost optimization. However, it does require careful consideration of architecture and deployment options to ensure optimal performance and cost-effectiveness.

PUBLIC CLOUD DEPLOYMENT (4-5 pages)

Public cloud organization of a taxi booking framework includes facilitating the framework on a cloud foundation that is overseen by a cloud specialist co-op and available to the general population over the web. There are a few cloud specialist co-ops accessible, including Amazon Web Administrations (AWS), Microsoft Sky blue, and Google Cloud Stage, among others. In this segment, we will examine the organization of a taxi booking framework on AWS, one of the most famous cloud specialist co-ops.

Whenever we had finished with planning the engineering, we have pushed ahead with the execution which includes the setting up an AWS account, making IAM client, designing lambda capability, rest full Programming interface utilizing Programming interface door, setting up Cognito client pool and personality pool, AWS code commit for source code the executives and dynamo DB for putting away the subtleties of the rides booked. The Lambda capabilities ought to deal with the business rationale of the framework, for example, booking a ride, overseeing ride demands, and following the ride continuously. The DynamoDB tables ought to store the information connected with clients, drivers, rides, and ride history.

To guarantee ideal execution and distinguish any issues or mistakes, it is vital to screen the serverless taxi booking framework on AWS cloud. This includes setting up checking apparatuses to follow the framework's exhibition measurements, for example, reaction times, blunder rates, and asset usage. Consistently investigating these measurements can assist with distinguishing any issues that might emerge and take into consideration speedy goal.

As interest for the taxi booking framework develops, it could be important to scale the assets to deal with expanded request. This should be possible by adding extra Lambda capabilities, expanding the limit of DynamoDB tables, or expanding the size of the Programming interface Door case. By observing the framework and scaling the assets on a case by case basis, the serverless taxi booking framework can keep on giving a dependable and effective help to clients and drivers the same.

Then we have at long last moved with the testing and arrangement of our web application utilizing AWS Enhance.

We have made a CI/CD pipeline utilizing by coordinating our Code Commit and Intensify so that on the off chance that any change happened from the designer end, with practically no manual interference and with zero margin time, we could furnish the refreshed web application with added highlights to our clients.

When the client has entered a substantial email or a telephone number, Cognito will check the legitimacy of subtleties and afterward a confirmation code utilizing Amazon basic email administration is shipped off the clients' mail or the telephone number. The client is then expected to enter the confirmation code so should demonstrate his credibility and afterward effectively sign in. The client can likewise essentially login through their Google account.

Later, an effective login is done the client will be advanced with a geological guide of their area. The client can explore through the guide according to their desire and can highlight a spot to choose their get area.

The client will then be requested to pick a taxi from the pool of accessible ones and can pick it by contrasting the estimating, criticism of the driver and the kind of taxi he/she needs.

When the client had effectively finished with every one of the means and affirm the booking, he will see a brief which will say, a taxi is drawing nearer towards you area and he can see a taxi drawing nearer towards his area and how much time moving toward the pickup location will be required.

There are several reasons why public cloud is a preferred choice for deploying a taxi booking system.

Firstly, public cloud infrastructure provides on-demand scalability, allowing the taxi booking system to scale up or down as needed to handle varying levels of demand. This means that the system can be easily scaled up during peak hours to handle a large number of requests and scaled down during off-peak hours to reduce costs.

Secondly, deploying a taxi booking system on public cloud infrastructure provides high availability and reliability. Public cloud providers have multiple data centers in different regions, ensuring that the system can continue to operate even in the event of a disaster or outage in one region.

Thirdly, deploying on public cloud infrastructure provides cost savings as it eliminates the need for companies to invest in expensive hardware, software, and maintenance. Public cloud providers offer a pay-as-you-go pricing model, where companies only pay for the resources they use.

Fourthly, public cloud infrastructure provides advanced security features to protect the system from cyber threats. Cloud providers offer features such as firewalls, identity and access management, encryption, and monitoring tools to ensure that the system remains secure.

Finally, public cloud infrastructure provides easy accessibility and global reach. Cloud providers have data centers all over the world, ensuring that users from different regions can access the system with low latency and high data transfer speeds.

It is a help presented by AWS utilized for putting away our source code and store documents. It is a totally overseen by AWS and resembles a confidential gift storehouse which is profoundly versatile and solid.

It empowers us to monitor the past renditions of our site for web application and furthermore permits us to move to the past adaptation at whatever point required. It is exceptionally simple to utilize and coordinate with different administrations like code pipeline, code convey, code construct.

We have utilized code commit for variant control and source code the board of our storehouse records and to make a CI/Compact disc pipeline with Enhance to fabricate and send our application code with zero personal time and no manual mediation.

Whereas, amplify on public cloud is a help presented by AWS which assists the clients with building and send a full-Stack application with exceptionally versatility and security.

Intensify gives many administrations, including validation, APIs, stockpiling, examination, facilitating, and CI/Compact disc, all coordinated into a solitary stage. These administrations permit designers to add usefulness to their application consistently and productively, diminishing advancement time and expenses.

In this way, in our application we have utilized AWS enhance and associated it with code resolve to make a persistent joining and consistent organization pipeline. Whenever any progressions are produced using our designer to code commit, the enhanced administration unexpectedly becomes possibly the most important factor, construct and send our application document.

Some of the other features for deploying a taxi booking system over public cloud are:

1. Scalability:

Public cloud infrastructure provides on-demand scalability, which is critical for a taxi booking system. As the system has to handle varying levels of demand, it needs to be able to scale up or down as needed. Public cloud providers offer flexible scaling options, allowing the system to add or remove resources in real-time based on demand. This means that during peak hours, the system can scale up to handle a large number of requests and during off-peak hours, it can scale down to reduce costs.

One of the key ways that public cloud providers enable scalability is through the use of virtualization and containerization technologies. By using these technologies, public cloud providers can create a pool of virtual resources that can be dynamically allocated to applications and services as needed. This allows taxi booking companies to easily scale their systems up or down, depending on demand.

In addition to virtualization and containerization, public cloud providers also offer a range of autoscaling tools that can be used to automatically scale resources in response to changes in demand. For example, a taxi booking system could be configured to automatically spin up additional virtual machines during periods of high demand, and then spin them down again during periods of low demand. This ensures that the system remains responsive and efficient, regardless of the level of demand.

2. High Availability:

Public cloud providers have multiple data centers in different regions, ensuring that the system can continue to operate even in the event of a disaster or outage in one region.

This means that users can access the taxi booking system without interruption, ensuring a high level of availability.

Additionally, public cloud providers offer features such as load balancers and auto-scaling to ensure that the system can handle varying levels of demand. Load balancers distribute incoming traffic across multiple servers, ensuring that no single server becomes

overloaded. Auto-scaling, on the other hand, automatically adds or removes resources based on demand. For instance, if the demand for a taxi booking system increases, auto-scaling can add more servers to ensure that the system remains responsive to users.

Public cloud providers also offer backup and disaster recovery options to ensure that data is protected and can be recovered in the event of an outage. They typically offer multiple backup options, including snapshots, replication, and backups to a secondary location, ensuring that data can be recovered quickly in the event of an outage.

3. Cost Savings:

Deploying a taxi booking system on public cloud infrastructure can provide significant cost savings. Public cloud providers offer a pay-as-you-go pricing model, where companies only pay for the resources they use. This eliminates the need for companies to invest in expensive hardware, software, and maintenance. Additionally, public cloud providers offer discounts for long-term usage, which can help companies save money in the long run.

Public cloud providers typically offer a pay-per-use pricing model, where users only pay for the resources they consume. This model eliminates the need to purchase and maintain expensive hardware and software infrastructure, which can be a significant cost-saving benefit for taxi booking systems. With pay-per-use pricing, users only pay for the resources they need, which can be scaled up or down as demand fluctuates. This provides a significant cost-saving benefit, as users are not required to pay for unused resources.

4. Advanced Security Features:

Public cloud providers offer advanced security features such as firewalls, identity and access management, encryption, and monitoring tools to ensure that the taxi booking system remains secure. These security features are typically more advanced than what companies can provide in their own on-premise data centers, making public cloud infrastructure a more secure option.

One of the most significant security features offered by public cloud providers is multi-factor authentication (MFA). MFA requires users to provide two or more forms of authentication, such as a password and a fingerprint, before they can access the system. This provides an additional layer of security beyond traditional usernames and passwords, making it more difficult for hackers to gain unauthorized access to the system.

Another critical security feature offered by public cloud providers is encryption. Encryption services are used to encrypt data at rest and in transit, ensuring that data is protected against unauthorized access. Encryption provides an additional layer of security and is an essential component of any comprehensive security solution.

Access controls are also a crucial security feature offered by public cloud providers. Access controls allow administrators to restrict access to resources based on roles and permissions. This ensures that only authorized users can access the resources they need to perform their jobs, reducing the risk of data breaches and other security threats.

5. Easy Accessibility and Global Reach:

Public cloud providers have data centers all over the world, ensuring that users from different regions can access the taxi booking system with low latency and high data transfer speeds. Additionally, public cloud providers offer content delivery networks (CDNs) that can cache frequently accessed content closer to the user, reducing latency even further.

Another advantage of public cloud accessibility is the ability to scale up or down quickly and easily. As the demand for taxi services fluctuates throughout the day, the public cloud can automatically scale up or down the resources needed to meet demand. This ensures that the system remains accessible and responsive, even during periods of high demand.

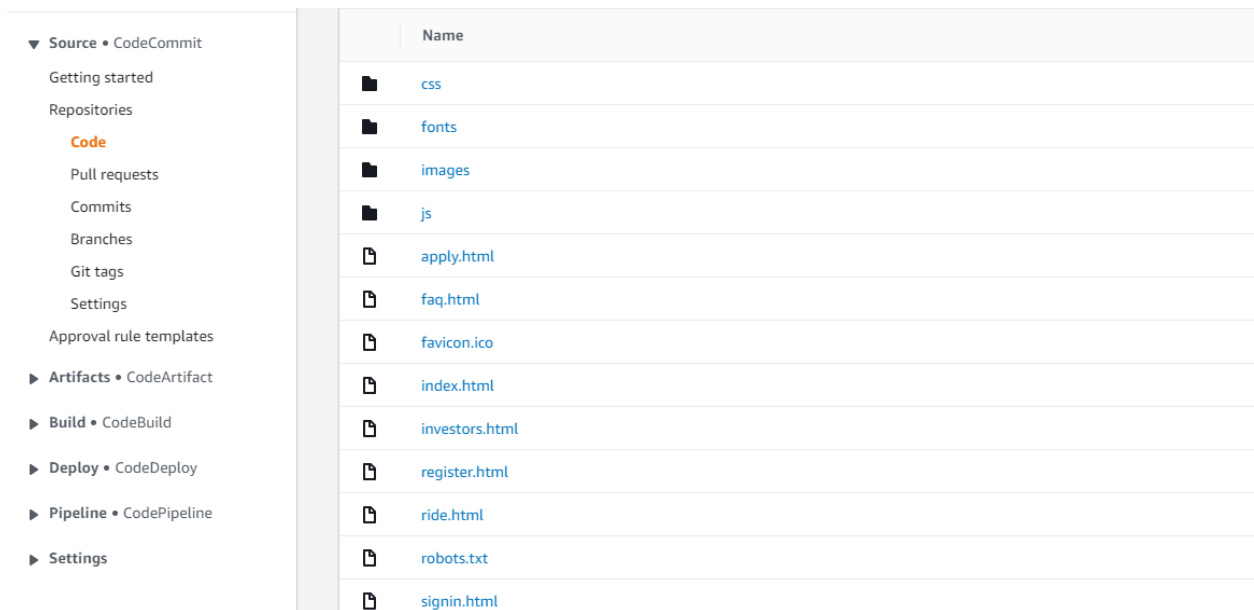
A variety of tools and services are also available on the public cloud to help increase accessibility and reach internationally. In order to ensure that customers can access the taxi booking system quickly and simply, regardless of their location, material Delivery Networks (CDNs) can be utilised to cache material in data centres all over the world. Similar to how traffic can be divided among various data centres using global load balancers, this improves performance and dependability.

And last, the administration tools and APIs that public cloud providers often provide make it simple to manage and keep track of the taxi booking system. This enables taxi booking companies to quickly assess usage, keep tabs on system performance, and pinpoint areas that need improvement, ensuring that the system is always available and responsive.

In conclusion, public cloud providers are the best option for hosting a taxi booking system because of their simple accessibility and extensive global network. The public cloud makes it simple to expand the system into new markets and scale up or down as needed to meet demand because it has data centres spread out throughout the globe and a variety of tools and services to increase performance and dependability.

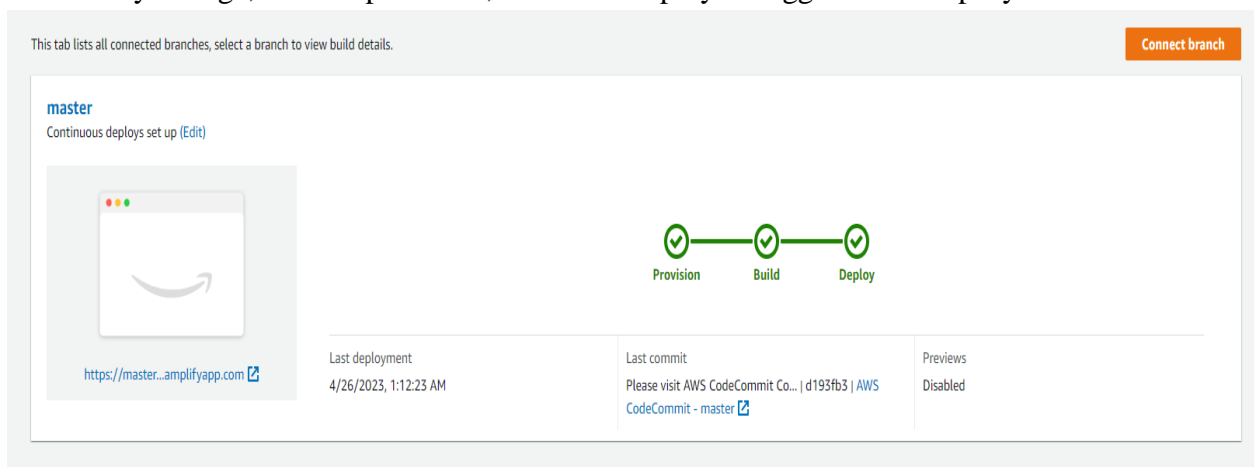
SCREENSHOTS

Is it used for version control and source code management of our code file.



{figure → 11 source code commit}

With every change, an auto provision, build and deploy is triggered in Amplify.



{Figure 12 shows the auto build process}

Key Bibliography/References

- [1] G. McGrath and P. R. Brenner, "Serverless Computing: Design, Implementation, and Performance," 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), Atlanta, GA, USA, 2017, pp. 405-410, doi: 10.1109/ICDCSW.2017.36.
- [2] R. A. P. Rajan, "Serverless Architecture - A Revolution in Cloud Computing," 2018 Tenth International Conference on Advanced Computing (ICoAC), Chennai, India, 2018, pp. 88-93, doi: 10.1109/ICoAC44903.2018.8939081.
- [3] D. Kelly, F. Glavin and E. Barrett, "Serverless Computing: Behind the Scenes of Major Platforms," 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), Beijing, China, 2020, pp. 304-312, doi: 10.1109/CLOUD49709.2020.00050.
- [4] S. Gandhi, A. Gore, S. Nimbarte and J. Abraham, "Implementation and Analysis of a Serverless Shared Drive with AWS Lambda," 2018 4th International Conference for Convergence in Technology (I2CT), Mangalore, India, 2018, pp. 1-6, doi: 10.1109/I2CT42659.2018.9058237.
- [5] S. Neela, Y. Neyyala, V. Pendem, K. Peryala and V. V. Kumar, "Cloud Computing Based Learning Web Application Through Amazon Web Services," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2021, pp. 472-475, doi: 10.1109/ICACCS51430.2021.9441974.
- [6] R. A. P. Rajan, "Serverless Architecture - A Revolution in Cloud Computing," 2018 Tenth International Conference on Advanced Computing (ICoAC), Chennai, India, 2018, pp. 88-93, doi: 10.1109/ICoAC44903.2018.8939081.
- [7] J. Surbiryala and C. Rong, "Cloud Computing: History and Overview," 2019 IEEE Cloud Summit, Washington, DC, USA, 2019, pp. 1-7, doi: 10.1109/CloudSummit47114.2019.00007.
- [8] J. Surbiryala and C. Rong, "Cloud Computing: History and Overview," 2019 IEEE Cloud Summit, Washington, DC, USA, 2019, pp. 1-7, doi: 10.1109/CloudSummit47114.2019.00007.
- [9] S. Namasudra, P. Roy and B. Balusamy, "Cloud Computing: Fundamentals and Research Issues," 2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM), Tindivanam, India, 2017, pp. 7-12, doi: 10.1109/ICRTCCM.2017.49.
- [10] S. Patidar, D. Rane and P. Jain, "A Survey Paper on Cloud Computing," 2012 Second International Conference on Advanced Computing & Communication Technologies, Rohtak, India, 2012, pp. 394-398, doi: 10.1109/ACCT.2012.15.

