EXECUTIVE SUMMARY

Project Summary: Echomate Lite - Rekindling Connections Through Shared Experiences. Echomate Lite is a platform designed to help users capture, organize, and share life's precious moments, fostering connection and shared reminiscing. Built using the MERN stack (MongoDB, Express.js, React.js, and Node.js), Echomate Lite leverages the power of JavaScript for both front-end and back-end development, ensuring a user-friendly and scalable experience.

Key Features:
Memory Capture and Organization: The platform enables users to effortlessly capture memories using photos, videos, text notes, and location tagging, organizing them chronologically, thematically, or by person.

Selective Sharing and Collaboration: Users can selectively share memories with friends and family, sparking conversations and reminiscing together, and collaborate on memories with loved ones, adding their perspectives.

User Experience: Echomate Lite provides an intuitive and visually appealing interface with interactive features that bring memories back to life and encourage engagement.

Architecture:
Echomate Lite utilizes a robust and scalable architecture deployed on AWS, ensuring high availability and performance.

Front-End: React.js provides a responsive and interactive user interface.
Back-End: Node.js and Express.js handle the server-side logic and API endpoints, deployed on AWS.
Database: MongoDB (or Amazon Document DB) stores user data and memory details.
Storage: Amazon S3 provides secure and cost-effective storage for user-uploaded media (photos, videos).

Deployment: AWS Code Pipeline automates the deployment process. Additional AWS Services: Amazon CloudWatch monitors application performance, and Amazon VPC and Security Groups create a secure network environment.

Technology Stack:
Front-End: React.js, HTML5, CSS3, JavaScript Back-End: Node.js, Express.js
Database: MongoDB (Amazon Document DB)
Storage: Amazon S3 Deployment: AWS Code Pipeline Monitoring: Amazon CloudWatch Security: Amazon VPC, Security Groups

Purpose:
Echomate Lite aims to provide a seamless and engaging platform for individuals and groups to capture, organize, share, and reminisce about cherished experiences. The platform's modern architecture and technology stack ensures scalability, security, and performance, making it a valuable tool for preserving and reliving life's precious moments.

TABLE OF CONTENTS

List of Picture

CHAPTER 1

INTRODUCTION

INTRODUCTION

1.1 Project Background: Technology Stack and AWS Services

echomate lite, our proposed platform for capturing, organizing, and sharing cherished memories, will leverage a powerful combination of technologies to deliver a user-friendly and scalable experience. Here's an overview of the chosen technology stack and the supporting AWS services:

1.1.1 MERN Stack

The MERN stack, an acronym for MongoDB, Express.js, React.js, and Node.js, forms the core development environment for echomate lite.

- MongoDB: A NoSQL database providing flexible schema and scalability for storing diverse memory data (text, photos, videos).
- Express.js: A Node.js web framework facilitating backend development and API creation for user interactions and data management.
- React.js: A popular JavaScript library for building dynamic and interactive user interfaces for capturing, organizing, and viewing memories.
- Node.js: A JavaScript runtime environment acting as the server-side engine, powering backend logic and API functionalities.

Benefits of MERN Stack:

- Unified JavaScript Environment: Developers familiar with JavaScript can work efficiently across both frontend (React.js) and backend (Node.js) development.
- Rapid Prototyping and Development: Pre-built components and streamlined development processes within the MERN stack enable faster project development cycles.
- Large and Active Community: Extensive resources, tutorials, and a readily available pool of developers familiar with the MERN stack.

1.1.2 AWS Services

echomate lite will utilize robust AWS services to ensure scalability, security, and global reach:

- AWS Code Pipeline: Automates the deployment process, integrating development, testing, and deployment stages.
- Amazon EC2: Provides virtual servers (instances) to host the backend application (Node.js and Express.js).

- Amazon S3: Offers secure and cost-effective object storage for user-uploaded media (photos, videos).
- Amazon Document DB (Optional): A NoSQL database service compatible with MongoDB. *(Consideration: Explore DynamoDB for potential cost benefits if data schema is flexible.)*
- Amazon CloudFront (Optional): A Content Delivery Network (CDN) for caching static frontend content (React.js) closer to users globally, improving load times.

Additional AWS Services (as needed):

- Amazon CloudWatch: Monitors application performance, resource utilization, and logs.
- Amazon VPC and Security Groups: Create a secure virtual network environment for platform resources.

Benefits of AWS Services:

- Scalability: AWS services accommodate growth in user base and data storage needs.
- Security: Robust AWS security features safeguard user data and platform resources.
- Reliability: High availability and fault tolerance ensure uninterrupted service.
- Cost-Effectiveness: Pay-as-you-go pricing models optimize costs based on resource usage.

1.2 Goals and Objectives for echomate lite

1.2.1 Overall Goal

The primary goal of echomate lite is to create a user-friendly platform that empowers individuals and groups to capture, organize, share, and reminisce about cherished experiences in a meaningful way.

1.2.2 Specific Objectives

User Experience:

- Develop an intuitive and visually appealing interface for capturing memories with photos, videos, text notes, and location tags.
- Design a user-friendly system for organizing and categorizing memories (chronological, thematic, or people-based).

- Implement interactive features that bring memories back to life and encourage engagement.

Social Interaction:

- Enable selective sharing of memories with friends and family, fostering conversations and reminiscing together.
- Facilitate collaborative memory creation, allowing users to add perspectives and enrich the storytelling experience.

Data Management and Security:

- Ensure secure storage of user data and uploaded media (photos, videos) leveraging robust AWS services.
- Implement user authentication and authorization mechanisms to control access to shared memories.

Scalability and Performance:

- Design the platform to accommodate a growing user base and increasing amounts of memory data.
- Optimize performance for smooth user experience and fast loading times, even for a global audience.

1.3 Key Requirements for echomate lite Platform

To achieve the defined goals and objectives, the following key requirements must be met:

1.3.1 Memory Capture and Organization

Capture Functionality:

- Users can effortlessly add new memories through:
    - Uploading photos and videos.
    - Adding text notes and descriptions.
    - Capturing location information using geolocation tags.

Organization Tools:

- Organize memories in various ways:
    - Chronological order (timeline view).
    - Thematic categorization (e.g., birthdays, vacations, holidays).
    - Tagging people associated with the memory.
    - Advanced search to easily find specific memories.

1.3.2 User Management and Sharing

User Authentication and Accounts:

- Secure user registration and login functionalities.
- User profiles to personalize the memory experience.

Selective Sharing:

- Users can control visibility of specific memories (friends, family, public).
- Permission settings for editing or downloading shared memories.

Social Interaction Features:

- "Like" and "Comment" features to encourage engagement.
- Notifications about new content or activity related to shared memories.
- Potential optional integration with existing social media platforms.

1.3.3 Collaborative Memory Creation

Shared Memory Ownership:

- Multiple users can contribute to the creation and enrichment of a memory.

Collaborative Features:

- Users can add photos, videos, and text notes collaboratively.
- Facilitate discussions and shared perspectives through comments and mentions.

1.3.4 Performance and Scalability

Fast Loading Times:

- Ensure echomate lite loads quickly across devices and locations.

Scalable Infrastructure:

- Accommodate a growing user base and increased data storage.

Image and Video Optimization:

- Optimize media loading times without compromising quality.

1.3.5 Security and Data Management

Secure Data Storage:

- Leverage AWS services for secure user data, photo, and video storage.

User Authentication and Authorization:

- Implement strong authentication mechanisms to prevent unauthorized access.

Data Backup and Recovery:

- Robust backup and recovery plans to safeguard user data in case of emergencies.

1.3.6 Additional Considerations

Mobile App Development:

- Consider a mobile app version to enhance accessibility and engagement.

Content Moderation Policies:

- Establish clear policies for user-generated content to ensure a respectful environment.

Monetization Strategy (Optional):

- Explore potential monetization options (premium features, subscriptions) without compromising the core user experience.

By fulfilling these key requirements, echomate lite can become a valuable platform for capturing, organizing, sharing, and reminiscing about life's most cherished moments. The focus on user experience, social interaction, and collaborative storytelling sets echomate lite apart as a platform designed to foster deeper connections and rekindle the joy of shared experiences.

CHAPTER 2

LITERATURE REVIEW

LITERATURE REVIEW

2.1 Rationale for the MERN Stack and AWS in Building echomate lite

The MERN stack (MongoDB, Express.js, React.js, and Node.js) coupled with AWS services provides a compelling foundation for developing echomate lite, a platform for capturing, organizing, and sharing memories. Here's a breakdown of the rationale behind these technological choices:

MERN Stack Benefits

- Unified JavaScript Environment: The MERN stack leverages JavaScript for both frontend (React.js) and backend (Node.js) development. This simplifies development, reduces context switching, and allows developers familiar with JavaScript to work efficiently across the entire stack.
- Rapid Prototyping and Development: Pre-built components and streamlined development processes within the MERN stack enable faster development cycles, crucial for bringing echomate lite to life quickly.
- Large and Active Community: The MERN stack boasts a vast developer community, offering extensive resources, tutorials, and a readily available pool of talent for project development and maintenance.

MERN Stack for echomate lite's Specific Needs

- React.js: Well-suited for building interactive and visually appealing user interfaces, ideal for creating an engaging experience for capturing, organizing, and viewing memories.
- Node.js: Enables real-time features and efficient data processing on the backend, crucial for handling user interactions and memory management.
- MongoDB (or DocumentDB): Provides a flexible NoSQL schema for storing diverse memory data (text, photos, videos) with ease of scaling as the user base grows.

AWS Services for Scalability, Security, and Performance

- Code Pipeline: Automates the deployment process, integrating development, testing, and deployment stages.

- Amazon EC2: Offers scalable virtual servers to host the echomate lite backend, accommodating growth in user traffic and data storage.
- Amazon S3: Provides secure and cost-effective object storage for user-uploaded media (photos and videos).
- Amazon Document DB : A NoSQL database service compatible with MongoDB, facilitating storage and retrieval of memory data. (Consider exploring DynamoDB for potential cost benefits.)
- Amazon CloudFront : A Content Delivery Network (CDN) to cache static frontend content closer to users globally, improving website loading times.

Additional AWS Services (as needed)

- Amazon Cognito: Provides user authentication and authorization functionalities.
- Amazon CloudWatch: Monitors application performance, resource utilization, and logs.
- Amazon VPC and Security Groups: Create a secure virtual network environment for echomate lite's resources.

Benefits of AWS for echomate lite

- Scalability: Accommodates a growing user base and increasing data storage needs.
- Security: Safeguards user data, uploaded media, and platform resources.
- Reliability: High availability and fault tolerance ensure uninterrupted service.
- Cost-Effectiveness: Pay-as-you-go pricing models allow cost optimization based on resource usage.

How MERN Stack and AWS Align with echomate lite's Requirements and Objectives

The MERN stack and AWS services effectively address the key requirements and objectives for echomate lite. Here's how:

MERN Stack

- User Experience and Social Interaction: React.js allows for a visually appealing and interactive user interface, ideal for capturing memories with various media and fostering engagement.

- Rapid Development: The MERN stack facilitates faster development cycles, enabling quicker feature iteration.
- Data Management and Scalability: MongoDB's flexible schema accommodates diverse memory data and supports scaling.

AWS Services

- Performance and Scalability:
    - EC2 provides scalable servers for backend hosting.
    - CloudFront improves global content delivery.
- Security and Data Management:
    - S3 stores user-uploaded media securely.
    - DocumentDB (or DynamoDB) stores user and memory data securely.
    - IAM and VPC enhance resource security.

Specific Alignments

- Memory Capture and Organization: Flexible data storage for text, photos, videos, and tags.
- User Management and Sharing: Secure user authentication via Amazon Cognito.
- Collaborative Memory Creation: Allows multiple users to contribute to shared memories.

Overall, the MERN stack offers an efficient development environment, while AWS ensures scalability, security, and reliability — perfectly suited for echomate lite.

2.2 AWS Services for echomate lite: Scalability, Security, and Performance

echomate lite will leverage key AWS services to ensure scalability, security, and performance:

1. Amazon EC2

- Description: Virtual servers for hosting the Node.js and Express.js backend.
- Reasoning: On-demand scalability based on traffic.

- Benefits:
    - Scalability: Add/remove instances easily.
    - Flexibility: Choose instance types based on workload.
    - Cost-Effectiveness: Pay-as-you-go pricing.

2. Amazon S3

- Description: Secure object storage for user-uploaded media.
- Reasoning: High availability, durability, and built-in security.
- Benefits:
    - Scalability: Automatically adjusts to storage needs.
    - Security: Encryption and access control features.
    - Cost-Effectiveness: Pay based on storage and data transfer usage.

3. Amazon DocumentDB

- Description: NoSQL database compatible with MongoDB.
- Reasoning: Flexible schema, scalable storage of memory data.
- Benefits:
    - Flexibility: Store diverse memory data.
    - Scalability: Handle growing data volumes.
    - Compatibility: Familiar MongoDB experience.

4. Amazon Application Load Balancer (ALB)

- Description: Distributes user traffic across EC2 instances.
- Reasoning: High availability and performance.
- Benefits:
    - Scalability: Auto-scales with user traffic.
    - High Availability: Routes traffic to healthy instances.
    - Performance: Reduces latency.

Additional AWS Services

- Amazon CloudWatch: Monitor application health and troubleshoot proactively.
- Amazon VPC and Security Groups: Secure network environment and control access to resources.

By leveraging these AWS services, echomate lite will achieve scalability, security, and cost-efficiency.

2.3 Problems Addressed with MERN and AWS

The MERN stack combined with AWS services tackles key challenges in building and scaling echomate lite:

Building Faster

- Rapid Prototyping: Pre-built components and unified JavaScript environment accelerate development.
- Simplified Development: Developers work seamlessly across frontend and backend with JavaScript.

Scaling Smoothly

- On-Demand Scalability: EC2 allows scaling based on traffic needs.
- Adaptable Data Storage: Document DB (or DynamoDB) scales with user data.
- Global Delivery: CloudFront improves site speed worldwide.

Safeguarding User Data

- Secure Storage: S3 ensures secure media storage.
- User Access Control: Cognito (optional) handles user authentication securely.
- Virtual Environment Security: VPC and Security Groups protect platform resources.

Additional Benefits

- Cost-Effectiveness: AWS's pay-as-you-go model optimizes costs.
- Developer Pool: MERN’s large community ensures talent availability.

Overall, MERN and AWS provide a robust foundation for fast development, secure scaling, and efficient operations of echomate lite.

CHAPTER 3

METHODOLOGY

METHODOLOGY

Requirement Analysis

The echomate lite social media platform is designed to provide a seamless and engaging experience for users. To achieve this, a thorough analysis of the requirements was conducted to identify the functional and non-functional aspects of the platform.

Functional Requirements

1. User Registration and Authentication: The platform should allow users to register and log in securely using their email addresses and passwords. The system should validate user credentials and ensure that only authorized users can access the platform.
2. Profile Management: Users should be able to create and manage their profiles, including uploading profile pictures, cover photos, and bio information.
3. Post Creation and Management: Users should be able to create, edit, and delete posts, including text, images, and videos. The system should allow users to categorize posts using hashtags and tags.
4. Like, Share, and Comment: Users should be able to like, share, and comment on posts. The system should display the number of likes, shares, and comments on each post.
5. Project Listing: The platform should display a list of projects, including project details, images, and videos.
6. Search and Filtering: The system should allow users to search for posts, projects, and users using keywords and filters.

7. Notification System: The platform should send notifications to users when they receive likes, comments, or shares on their posts.

Non-Functional Requirements

1. Security: The platform should ensure the security and integrity of user data, including passwords, profile information, and post content. The system should implement encryption and secure protocols to protect user data.
2. Scalability: The platform should be able to handle a large number of users and posts, with the ability to scale up or down as needed.
3. Performance: The system should respond quickly to user requests, with a maximum response time of 2 seconds.
4. Usability: The platform should be easy to use, with an intuitive interface and clear navigation.
5. Availability: The system should be available 24/7, with a minimum uptime of 99.9%.

Technical Requirements

1. Backend: The platform should use a MongoDB database to store user data, post content, and project information.
2. Frontend: The system should use a web-based interface built using HTML, CSS, and JavaScript.
3. Deployment: The platform should be deployed on AWS EC2 instances, with CodePipeline used for continuous integration and continuous deployment (CI/CD).
4. Cloud Services: The system should use AWS services, including EC2, S3, and CloudWatch, to ensure scalability, security, and reliability.

By identifying and documenting these requirements, the echomate lite social media platform is designed to provide a robust, scalable, and engaging experience for users.

Technology Selection

For the echomate lite social media platform, the following AWS services are selected for

implementing various components of the web application:

Frontend and Backend Deployment

● Amazon EC2: For deploying the frontend and backend applications, Amazon EC2 provides a scalable and secure compute service. The instance type and configuration can be customized based on the application's requirements.

Automating Deployment

● AWS CodePipeline: For automating the deployment process, AWS CodePipeline provides a continuous integration and continuous deployment (CI/CD) service. CodePipeline automates the build, test, and deployment of the application, ensuring faster and more reliable deployments.

Database

● MongoDB: For storing user data, post content, and project information, MongoDB provides a NoSQL database solution. MongoDB's flexible schema and scalability make it an ideal choice for handling large amounts of data.

● Frontend: React.js is used for building the user interface of echomate lite. React offers a component-based approach for efficient UI development and allows for creating a dynamic and interactive user experience.

● Backend: Node.js with Express.js serves as the backend framework for echomate lite. Node.js provides a JavaScript runtime environment for server-side

scripting, while Express.js simplifies building web applications and APIs. This combination allows for efficient handling of user requests, data processing, and communication with the database.

2. Database:

● MongoDB: A NoSQL database is chosen to store user data, posts, comments, and other application data in echomate lite. MongoDB offers flexibility in data schema design, making it suitable for storing various data types associated with a social media platform.

3. Deployment:

● AWS CodePipeline: This automated deployment service is used to streamline the deployment process for both the frontend (React) and backend (Node.js) applications of echomate lite. CodePipeline integrates with other services like CodeCommit (for version control) and CodeDeploy for automated deployments, ensuring efficient and reliable updates to the platform.

4. Server Hosting:

While the initial report mentioned EC2 for both frontend and backend deployment, it's

generally not recommended to run a frontend application directly on an EC2 instance. Here's a more suitable approach:

● Amazon S3: This object storage service is ideal for hosting the static content of your React application (HTML, CSS, JavaScript files). S3 offers high availability, scalability, and cost-effective storage for these assets.

● Amazon CloudFront (Optional): Consider utilizing CloudFront as a Content Delivery Network (CDN) to improve website loading times by caching static

content globally. Users will access content from geographically closer edge locations, resulting in faster performance.

● Amazon EC2: This service can still be used for hosting the backend Node.js application of echomate lite. EC2 offers a scalable compute environment where you can deploy your Node.js server instances.

Security Considerations:

● Implement security measures like IAM roles to control access to different AWS resources.

● Configure security groups for EC2 instances to restrict access and enhance security.

● Secure your MongoDB connection using best practices and authentication mechanisms.

Benefits of Chosen Technologies:

● Scalability: The chosen technologies, including Node.js, MongoDB, and AWS services, offer scalability to accommodate a growing user base for echomate lite.

● Performance: Utilizing React for the frontend and CloudFront (optional) can improve website loading times and responsiveness.

● Cost-Effectiveness: AWS offers pay-per-use billing for services, allowing you to optimize costs based on your usage patterns.

● Development Efficiency: React and Node.js provide a familiar environment for JavaScript developers, streamlining development processes.

By leveraging this technology stack and architecture, you have built a solid foundation

for echomate lite, a social media platform with the potential to grow and adapt to its user base's needs.

CHAPTER 4

SOLUTION ARCHITECTURE

echomate lite: Design Phase - Web Application Architecture and Data Models

1. Web Application Architecture

echomate lite will utilize a three-tier architecture for optimal performance and maintainability:

Presentation Layer (Frontend):

● Developed with React.js, this layer handles user interactions, displays the UI elements, and communicates with the API layer via HTTP requests.

● React components will be built for functionalities like user login/profile, newsfeed, post creation/editing, commenting, etc.

Business Logic Layer (Backend):

● Developed with Node.js and Express.js, this layer handles user authentication, data access, business logic (processing likes/comments), and communication with the database.

● Node.js offers an efficient server-side runtime environment for handling user requests and API interactions.

● Express.js simplifies building web APIs and structures the backend application effectively.

Data Layer (Database):

● Leverages MongoDB, a NoSQL database, to store user data, posts, comments, and other application data in a flexible and scalable manner.

● MongoDB's schema-less design allows for storing various data types associated with a social media platform without rigid structure limitations.

2. Data Models

Here's a breakdown of possible data models for key entities in echomate lite:

2.1 User Model:

JSON

{

userId: String (unique identifier - consider using ObjectID for MongoDB),

email: String (used for login),

password: String (hashed for security),

username: String,

profilePicture: String (optional, URL path to image in S3),

bio: String (optional, user description),

// Add any additional user-specific fields like following lists etc.

friends: Array<String> (user IDs of friends)

}

Use code with caution.

2.2 Post Model:

JSON

{

postId: String (unique identifier - consider using ObjectID for MongoDB),

userId: String (reference to user who created the post),

content: String (text content of the post),

createdAt: Date (timestamp of post creation),

updatedAt: Date (timestamp of last update),

likes: Array<String> (list of userIds who liked the post),

comments: Array<Object> (array of comment objects),

// Add fields for attachments like images/videos stored in S3 (optional)

}

Use code with caution.

2.3 Comment Model:

JSON

{

}

commentId: String (unique identifier - consider using ObjectID for MongoDB),

postId: String (reference to post the comment belongs to),

userId: String (reference to user who created the comment),

content: String (text content of the comment),

createdAt: Date (timestamp of comment creation),

Use code with caution.

Additional Considerations:

● Implement data access methods in the backend to interact with MongoDB (CRUD operations: Create, Read, Update, Delete).

● Define proper authorization mechanisms to control access to user data and actions based on user roles (e.g., only allow editing posts created by the user).

● Consider additional data models for features like private messages or user groups (if applicable).

Benefits of this Architecture:

● Separation of Concerns: This architecture promotes separation of concerns, making development and maintenance easier.

● Scalability: Each layer can be scaled independently to accommodate growth.

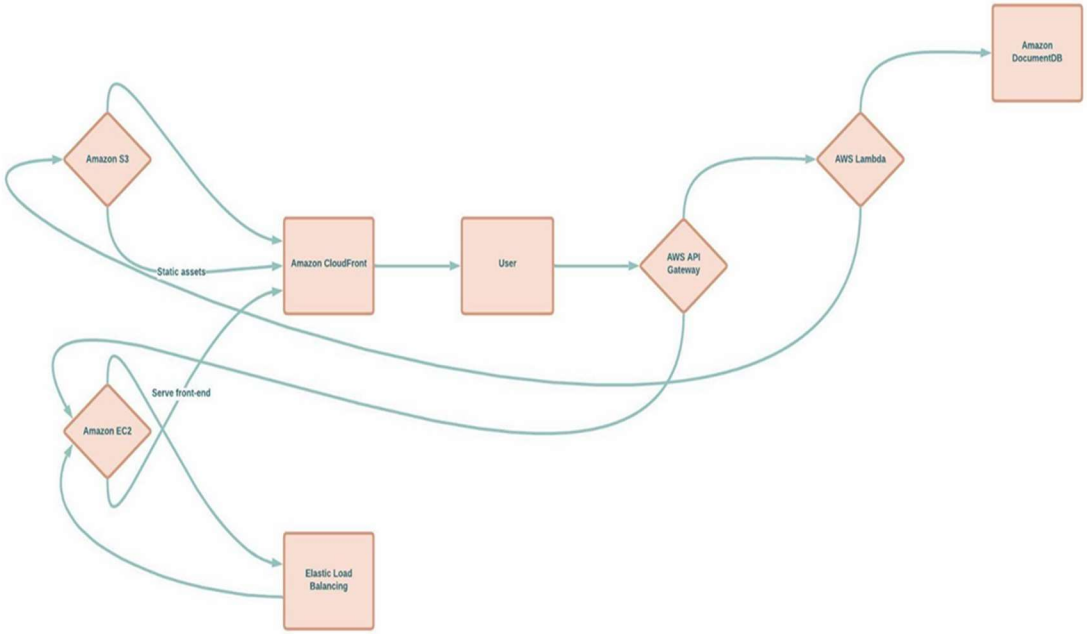
● Performance: React offers efficient UI rendering, Node.js provides a lightweight runtime environment, and MongoDB offers fast data access.

● Flexibility: MongoDB's schema-less design allows for adapting data models as echomate lite evolves.

By utilizing this design approach and data models, you can build a robust and scalable

foundation for the echomate lite social media platform.

Architecture Diagram:



Architecture Overview:

Description of Components:

Frontend (Vite.js):

Hosted on AWS S3 for static content delivery.Utilizes AWS CloudFront CDN for global caching and faster content delivery.

Reasons: S3 offers scalable and reliable storage, while CloudFront enhances user experience with reduced latency and improved performance.

Backend (Express.js):

Hosted on AWS EC2 instances for scalable backend services.Integrates with AWS Lambda for serverless functions handling specific tasks.

Reasons: EC2 provides scalable computing power, while Lambda reduces operational complexity and costs for certain backend operations.

Database (MongoDB):

Hosted on MongoDB Atlas for managed database services.Offers horizontal scalability, data redundancy, and automated backups.

Reasons: MongoDB Atlas provides a fully managed database solution, ensuring scalability, reliability, and data integrity.

CI/CD Pipeline (AWS CodePipeline):

Automates continuous integration, testing, and deployment processes.Integrates with GitHub repository for version control and code updates.

Reasons: CodePipeline streamlines development workflows, ensures code consistency, and facilitates rapid and reliable deployments.

Reasoning Behind Architectural Choices:

Scalability: The architecture leverages AWS services such as EC2, Lambda, S3, and MongoDB Atlas for scalable infrastructure, serverless computing, content delivery, and database management.

Performance: AWS CloudFront CDN enhances content delivery speed, reduces latency,

and improves user experience.

Reliability: Managed services like MongoDB Atlas and automated backups ensure data

reliability, integrity, and disaster recovery capabilities.

Cost-Efficiency: Serverless computing with AWS Lambda optimizes resource utilization

and reduces operational costs for specific backend operations.

Trade-Offs and Implications for Future Scalability:

Trade-Offs: While serverless computing with AWS Lambda reduces operational complexity and costs, it may introduce additional latency for certain functions compared to traditional EC2 instances.

Implications for Scalability: The architecture's use of scalable AWS services allows for future growth and increased user demand. However, monitoring and optimization of Lambda functions' performance may be necessary as the platform scales.

Cost Analysis:

AWS EC2: Costs vary based on instance types, usage, and scaling requirements. Continuous monitoring and optimization can help manage costs effectively.

AWS Lambda: Billed based on the number of invocations and execution time. Optimizing code and minimizing unnecessary invocations can reduce costs.

AWS S3: Costs depend on storage usage, data transfer, and request rates. Lifecycle policies and efficient data management practices can control costs.

MongoDB Atlas: Pricing is based on storage, instance size, and additional features. Choosing appropriate configurations and scaling options can impact costs.

Conclusion:

The proposed architecture leverages scalable, reliable, and cost-effective AWS services

to create a robust and efficient platform for the echomate lite project. Continuous monitoring, optimization, and cost management practices are essential for ensuring scalability, performance, reliability, and cost-efficiency as the platform evolves and grows.

echomate lite: Cost Analysis and Financial Implications

This section analyzes the potential costs associated with the AWS services chosen for your social media platform, echomate lite. Remember, actual costs will vary depending

on your specific usage patterns.

Services:

● Amazon S3: Ideal for storing your React application's static content (HTML, CSS, JavaScript). Costs are based on storage used and data transfer out.

Choose the Standard storage class for frequently accessed content.

● Amazon CloudFront (Optional): A CDN that improves website loading times by caching content globally. Costs include a request pricing model and data transfer out charges.

Consider using CloudFront only if you experience significant user traffic

from geographically diverse locations.

● Amazon EC2: Suitable for hosting your Node.js backend application. Costs are based on instance type, operating system, region, and usage time.

Start with a smaller instance type (e.g., t2.micro) for development and

testing. Scale up (m5.xlarge or higher) as your user base grows.

● Amazon CodePipeline: Offers a free tier with limitations on builds and deployments per month. Exceeding the free tier will result in pay-per-use charges.
● Amazon CodeDeploy: Integrates with CodePipeline for automated deployments. Costs are based on the number of deployments and successful/failed instances.
● MongoDB Atlas (Optional): A managed MongoDB service on AWS. Costs are based on storage used, instance class, and data transfer out.
Consider using MongoDB Atlas for a more managed and scalable database solution, but it might have higher costs compared to self-managed MongoDB on EC2.
Financial Considerations:
● Upfront Costs: There are minimal upfront costs associated with using AWS services. You pay only for the resources you use.
● Ongoing Costs: The ongoing cost will depend on your monthly usage of storage, data transfer, compute resources (EC2), and deployment activities.
● Cost Optimization Strategies:
● Utilize AWS pricing calculators to estimate your potential costs before deployment.
● Take advantage of AWS free tiers and reserved instances for predictable workloads to reduce costs.Use smaller EC2 instance types during development and testing.Implement auto-scaling for EC2 instances to optimize resource utilization and avoid paying for unused capacity.Regularly monitor your resource usage and identify opportunities for cost savings by right-sizing instances or adjusting service configurations.
Here are some additional factors to consider:

● Data Transfer: Costs associated with data transfer in and out of AWS can add up. Optimize your application to minimize unnecessary data transfers (e.g., image resizing before upload).

● CloudFront Pricing: Evaluate the cost-benefit of CloudFront based on your user traffic patterns. It might not be necessary for a small user base.

● MongoDB Atlas vs Self-Managed: If you choose self-managed MongoDB on EC2, consider the additional management overhead compared to the managed service offered by MongoDB Atlas.

Overall Cost Management:

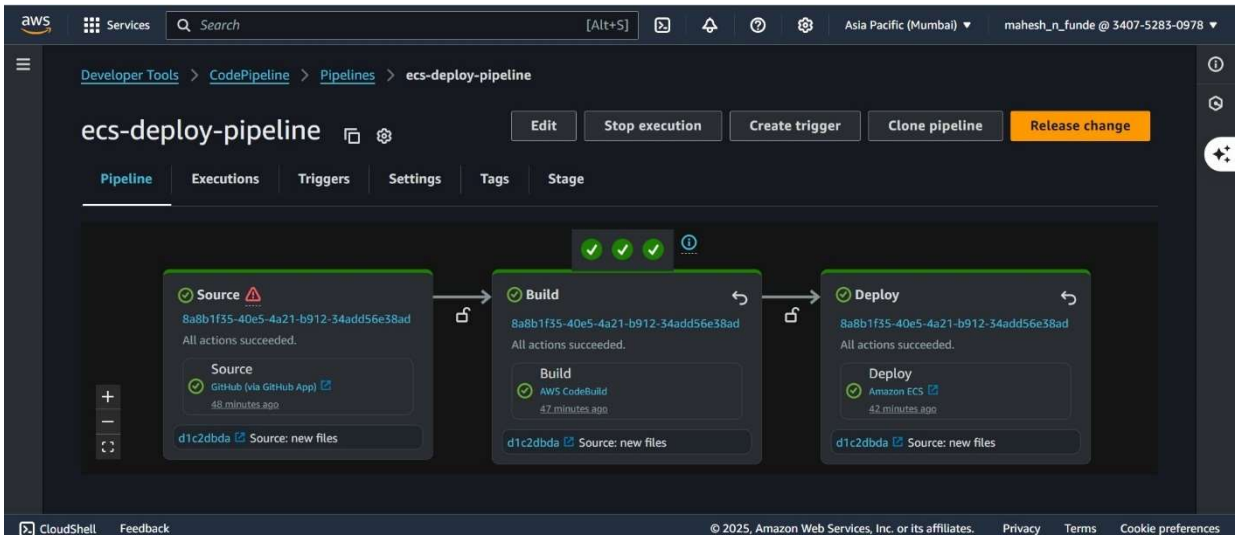
By carefully planning your resource usage and implementing cost optimization strategies, you can keep the financial implications of using AWS services manageable for your social media platform, echomate lite.

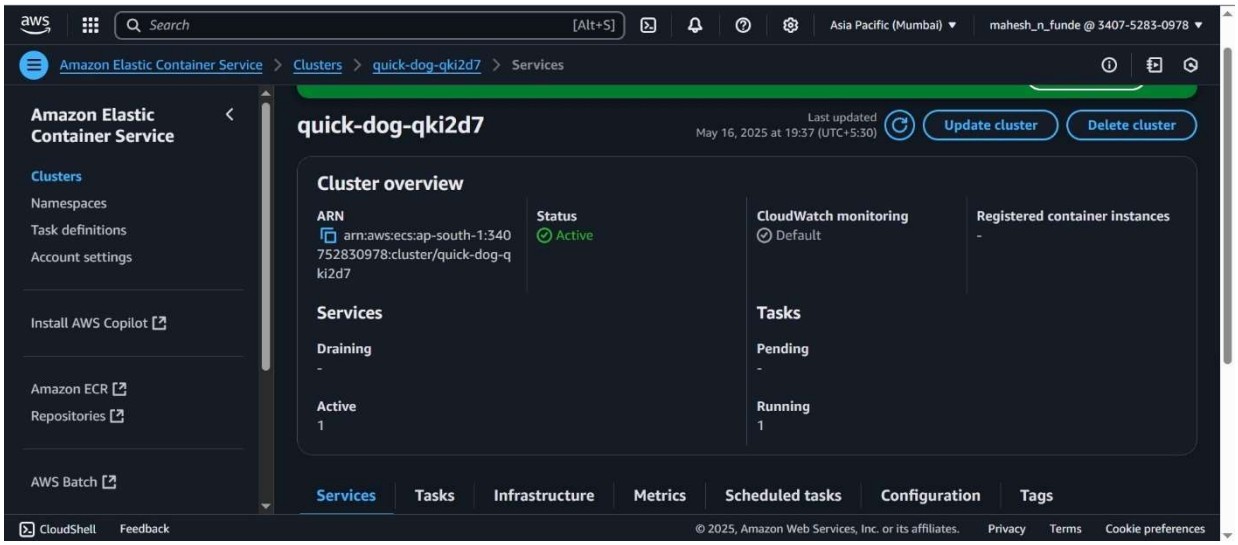
CHAPTER 5

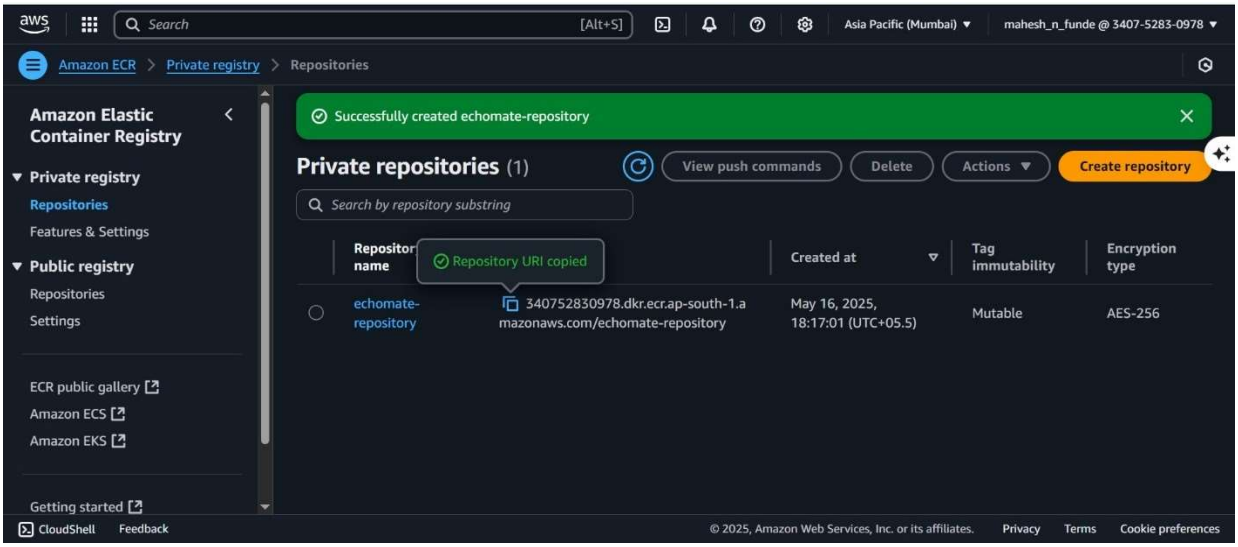
IMPLEMENTATION

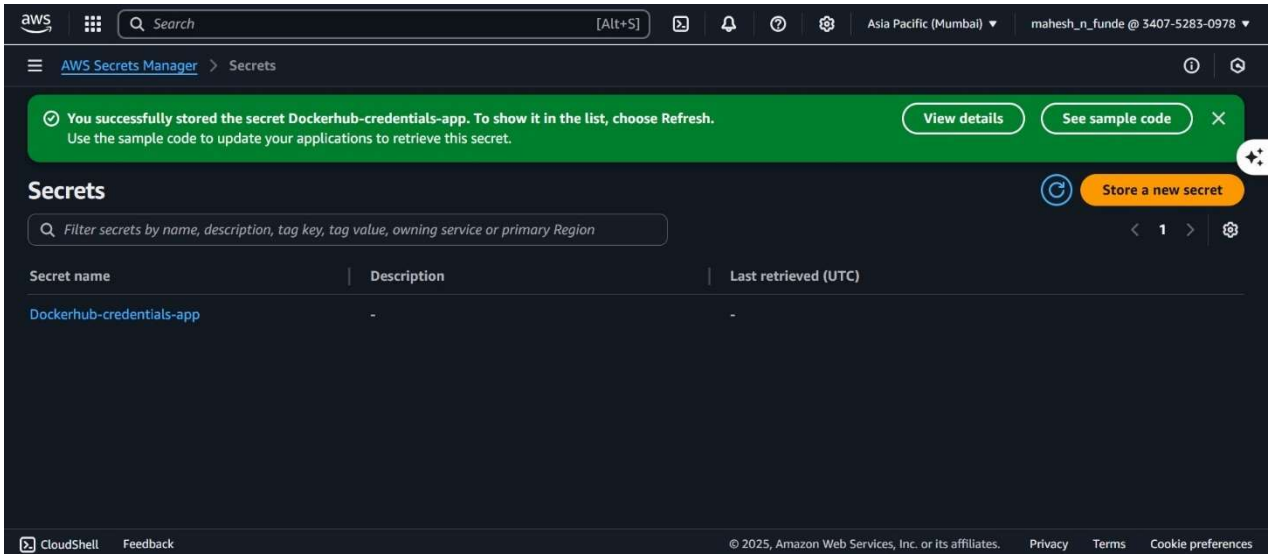
Implementation:

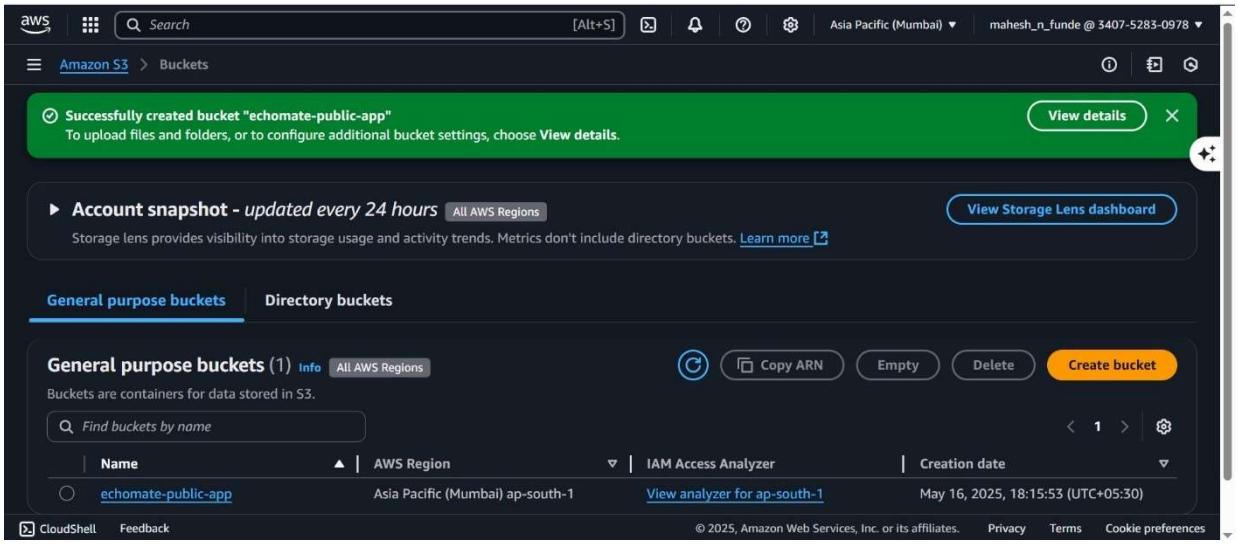
Screenshots of the deployment:

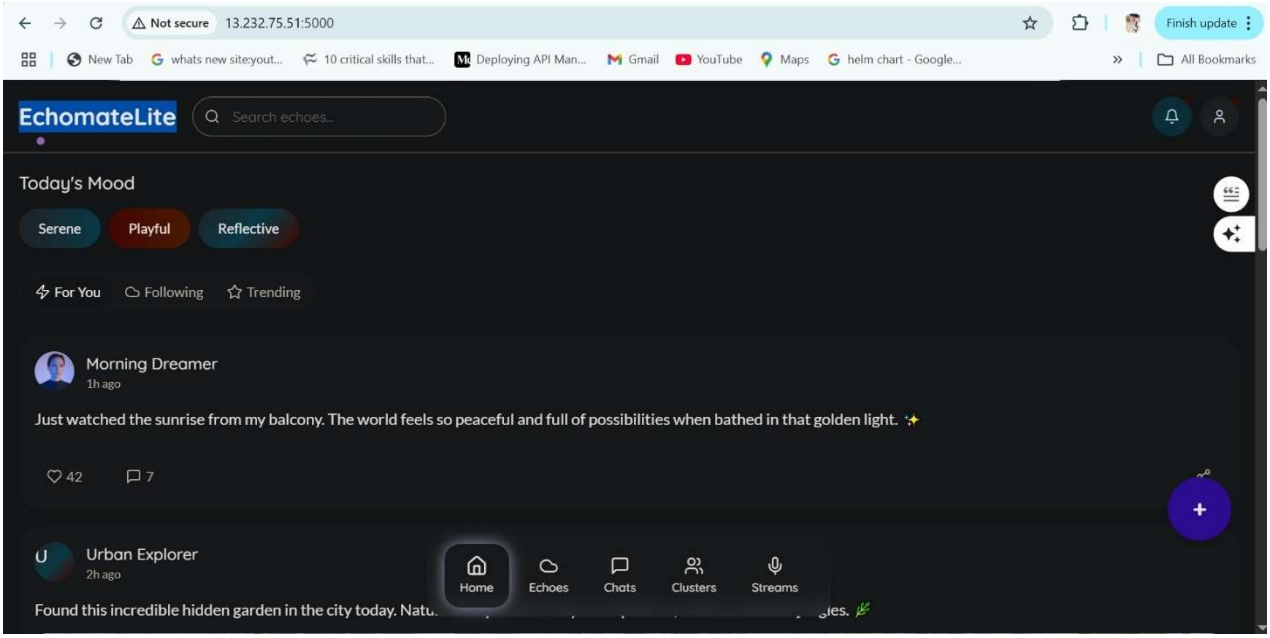


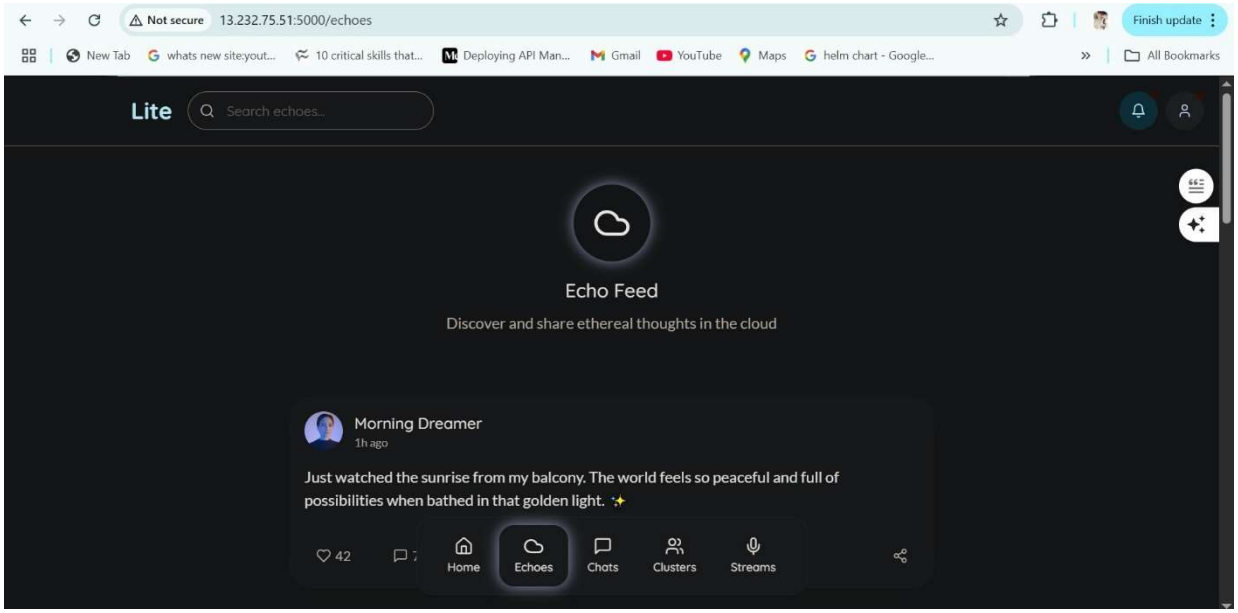


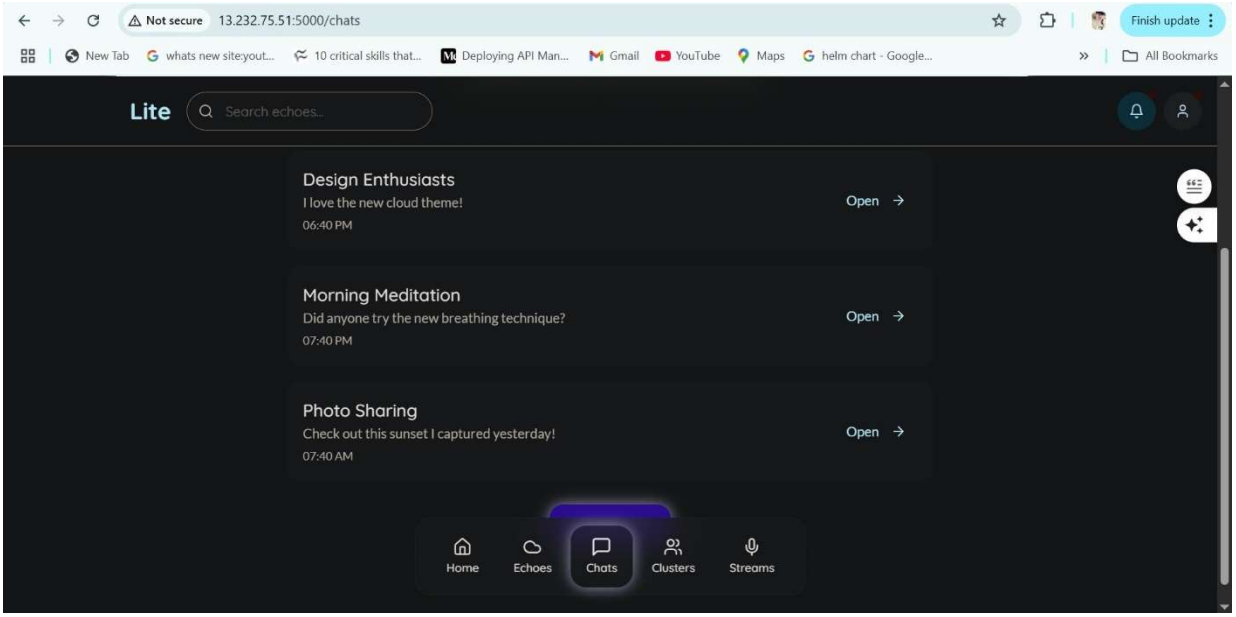

Search [Alt+S]
Asia Pacific (Mumbai)
mahesh_n_funde @ 3407-5283-0978
Amazon ECR > Private registry > Repositories
Amazon Elastic Container Registry
Private registry
Repositories
Features & Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Getting started
Successfully created echomate-repository
Private repositories (1)
View push commands
Delete
Actions
Create repository
Search by repository substring
Repository name
Repository URI copied
Created at
Tag immutability
Encryption type
echomate-repository
340752830978.dkr.ecr.ap-south-1.amazonaws.com/echomate-repository
May 16, 2025, 18:17:01 (UTC+05.5)
Mutable
AES-256
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

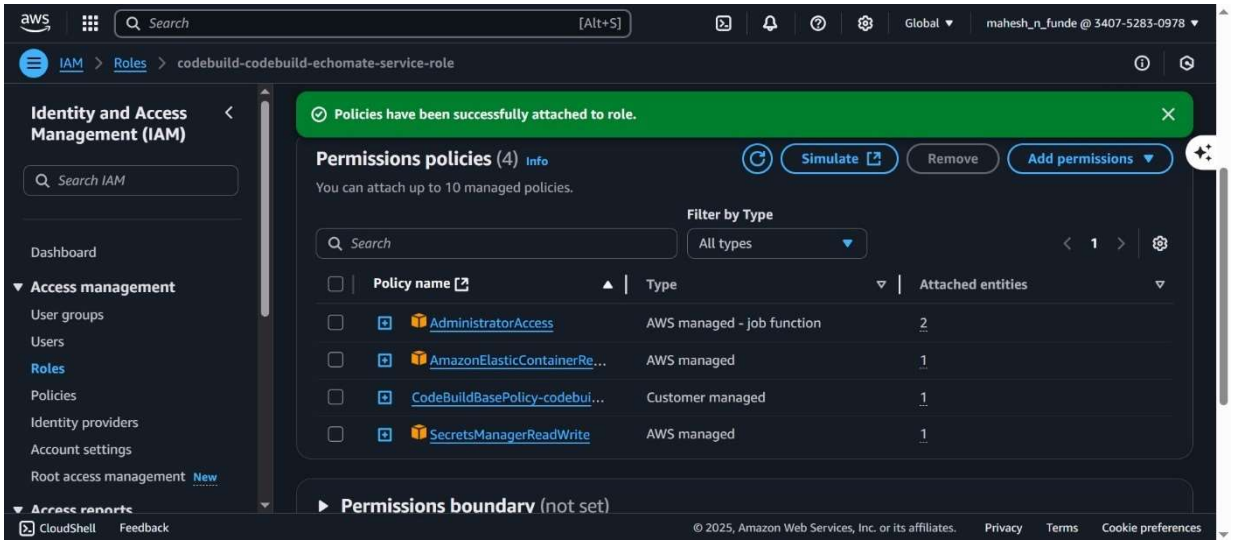

Search [Alt+S]
Asia Pacific (Mumbai)
mahesh_n_funde @ 3407-5283-0978
AWS Secrets Manager > Secrets
You successfully stored the secret Dockerhub-credentials-app. To show it in the list, choose Refresh.
Use the sample code to update your applications to retrieve this secret.
View details
See sample code
Secrets
Store a new secret
Filter secrets by name, description, tag key, tag value, owning service or primary Region
Secret name
Description
Last retrieved (UTC)
Dockerhub-credentials-app
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

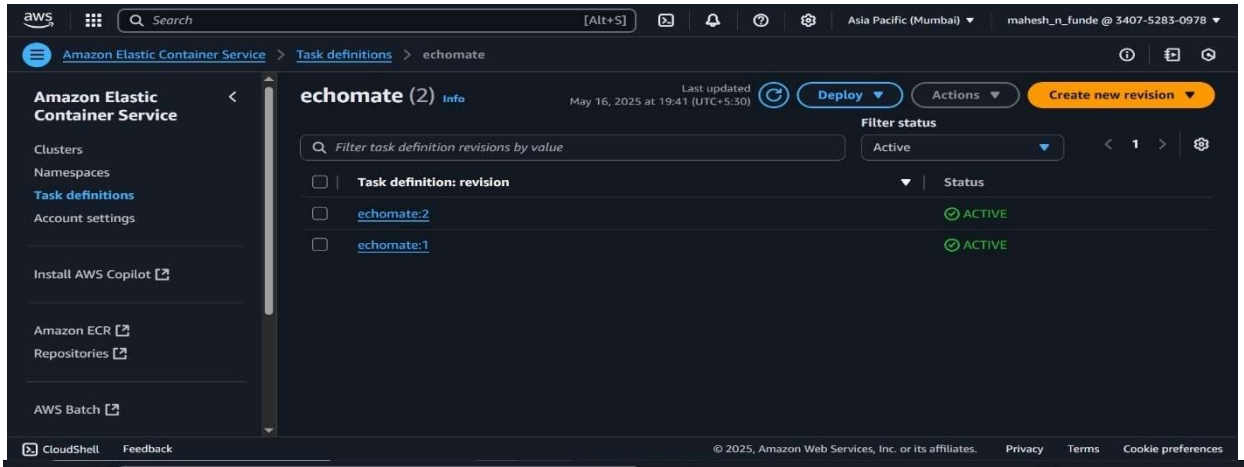

Search [Alt+S]
Asia Pacific (Mumbai)
mahesh_n_funde @ 3407-5283-0978
Amazon S3 > Buckets
Successfully created bucket "echomate-public-app"
To upload files and folders, or to configure additional bucket settings, choose View details.
View details
Account snapshot - updated every 24 hours
All AWS Regions
View Storage Lens dashboard
Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. Learn more
General purpose buckets
Directory buckets
General purpose buckets (1) Info
All AWS Regions
Copy ARN
Empty
Delete
Create bucket
Buckets are containers for data stored in S3.
Find buckets by name
Name
AWS Region
IAM Access Analyzer
Creation date
echomate-public-app
Asia Pacific (Mumbai) ap-south-1
View analyzer for ap-south-1
May 16, 2025, 18:15:53 (UTC+05:30)
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

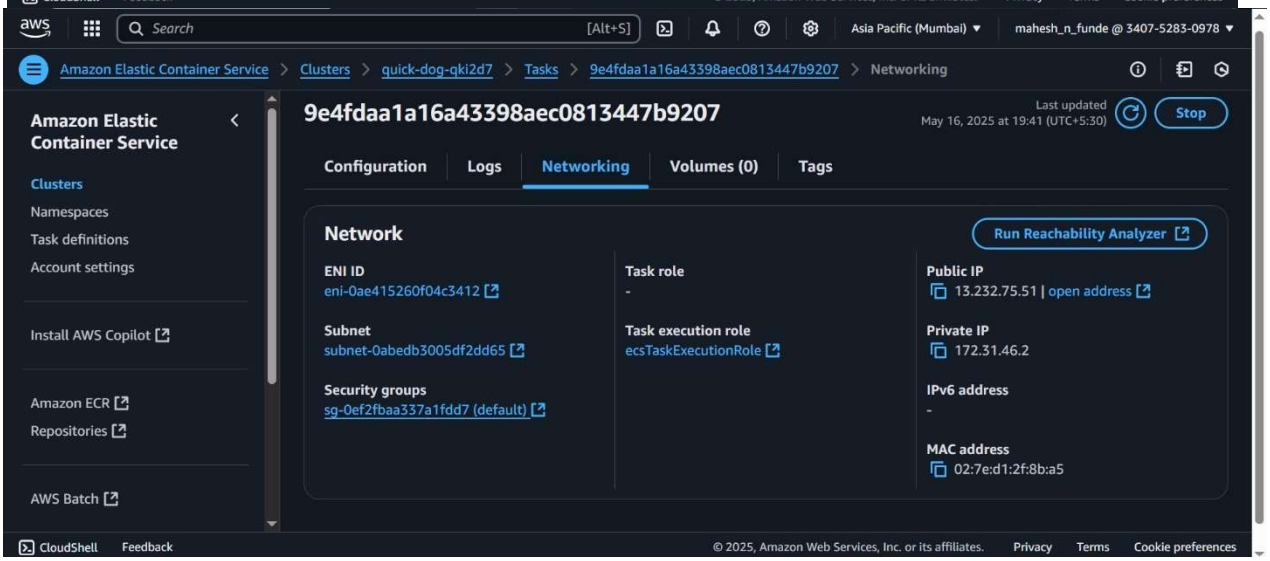

Not secure
13.232.75.51:5000
Finish update
New Tab
whats new site:yout...
10 critical skills that...
Deploying API Man...
Gmail
YouTube
Maps
helm chart - Google...
All Bookmarks
EchomateLite
Search echoes...
Today's Mood
Serene
Playful
Reflective
For You
Following
Trending
Morning Dreamer
1h ago
Just watched the sunrise from my balcony. The world feels so peaceful and full of possibilities when bathed in that golden light.
42
7
Urban Explorer
2h ago
Found this incredible hidden garden in the city today. Natu
Home
Echoes
Chats
Clusters
Streams


Not secure
13.232.75.51:5000/echoes
Finish update
New Tab
whats new site:yout...
10 critical skills that...
Deploying API Man...
Gmail
YouTube
Maps
helm chart - Google...
All Bookmarks
Lite
Search echoes...
Echo Feed
Discover and share ethereal thoughts in the cloud
Morning Dreamer
1h ago
Just watched the sunrise from my balcony. The world feels so peaceful and full of possibilities when bathed in that golden light.
42
Home
Echoes
Chats
Clusters
Streams


Not secure 13.232.75.51:5000/chats
Finish update
New Tab
whats new site:yout...
10 critical skills that...
Deploying API Man...
Gmail
YouTube
Maps
helm chart - Google...
All Bookmarks
Lite
Search echoes...
Design Enthusiasts
I love the new cloud theme!
06:40 PM
Open
Morning Meditation
Did anyone try the new breathing technique?
07:40 PM
Open
Photo Sharing
Check out this sunset I captured yesterday!
07:40 AM
Open
Home
Echoes
Chats
Clusters
Streams


aws
Search
[Alt+S]
Global
mahesh_n_funde @ 3407-5283-0978
IAM > Roles > codebuild-codebuild-echomate-service-role
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
Policies have been successfully attached to role.
Permissions policies (4) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Filter by Type
Search
All types
Policy name
Type
Attached entities
AdministratorAccess
AWS managed - job function
2
AmazonElasticContainerRe...
AWS managed
1
CodeBuildBasePolicy-codebui...
Customer managed
1
SecretsManagerReadWrite
AWS managed
1
Permissions boundary (not set)
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences





echomate lite: Implementation Details:

This section dives into the implementation aspects of your social media platform, echomate lite. Due to security concerns, it's best to avoid including sensitive code snippets directly in the report. However, you can describe the functionalities and reference code sections with comments explaining the logic.

1. Implementation Steps

Project Setup:

Initialize a Node.js project and install required dependencies (Express, Mongoose, etc.) for the backend.

Set up a React project for the front-end using tools like Create React App.

Configure your chosen database solution (MongoDB Atlas or self-managed MongoDB on EC2) with appropriate data models.

User Authentication:

Implement user registration with email and password on the backend. Store passwords securely using hashing algorithms (e.g., bcrypt).

Develop login functionality with token-based authentication (e.g., JWT) for session management.

Profile Management:

Create functionalities for users to edit their profile information (username, bio, profile picture).

Implement logic to store and retrieve profile pictures using S3 (optional). Consider libraries for image uploads and resizing.

Content Management:

Develop functionalities for users to create new posts with text, image, or video content. Utilize libraries for handling image/video uploads to S3.

Implement logic to store posts in the database with references to uploaded media in S3.

Build functionalities to retrieve and display posts in a newsfeed format based on followed users. Consider pagination for efficient loading of large datasets.

Social Interactions:

Develop functionalities for users to like, share, and comment on other users' posts.

Implement logic to update post data (likes, comments) in the database upon user interactions.

Consider using libraries for handling timestamps and user mentions in comments.

Frontend Development:

Design and develop React components for user interface elements (login, profile, post creation, newsfeed).

Implement logic for user interactions (like, share, comment) and data fetching from the backend API using libraries like Axios.

Ensure responsive design for optimal viewing across different devices.

Deployment:

Configure AWS CodePipeline for automated deployment of your front-end and back-end applications.

Set up EC2 instances for hosting your backend application and configure security groups for access control.

Integrate CodeDeploy with CodePipeline for automated deployments to EC2 instances.

Configure S3 bucket and CloudFront (optional) for hosting your React application's static content.

2. List of Features and Modules:

Module	Description	Code Snippet Reference (Optional)
User Management	User registration, login, profile management	userController.js (handles backend logic)
Content Management	Post creation, editing, deletion	postController.js
Newsfeed	Displaying posts from followed users	Feed.jsx (React component)
Likes & Comments	Users liking, sharing, and commenting on posts	postService.js (handles like/comment logic)

Note: These are just examples, and the specific modules and their functionalities might vary based on your project implementation.

Additional Considerations:

- Implement error handling and validation mechanisms on both the front-end and back-end.
- Consider unit testing your backend code for individual functionalities and integration testing for overall application flow.
- Utilize logging mechanisms for debugging and troubleshooting purposes.

Focus on Security:

- Sanitize user input to prevent vulnerabilities like SQL injection or cross-site scripting (XSS).
- Implement authorization checks to restrict access to user data and actions based on user roles.
- Secure your database connection with appropriate authentication mechanisms.
- Regularly update dependencies and libraries to address potential security vulnerabilities.

By following these implementation steps and focusing on security best practices, you can build a robust and secure social media platform, echomate lite.

CHAPTER 6

RESULTS AND DISCUSSION

echomate lite: Implementation Details

This section dives into the implementation aspects of your social media platform,

echomate lite. Due to security concerns, it's best to avoid including sensitive code

snippets directly in the report. However, you can describe the functionalities and

reference code sections with comments explaining the logic.

1. Implementation Steps

1. Project Setup:

Initialize a Node.js project and install required dependencies (Express, Mongoose, etc.) for the backend.

Set up a React project for the front-end using tools like Create React App.

Configure your chosen database solution (MongoDB Atlas or self-managed MongoDB on EC2) with appropriate data models.

2. User Authentication:

Implement user registration with email and password on the backend.

Store passwords securely using hashing algorithms (e.g., bcrypt).

Develop login functionality with token-based authentication (e.g., JWT) for session management.

3. Profile Management:

Create functionalities for users to edit their profile information (username, bio, profile picture).

Implement logic to store and retrieve profile pictures using S3 (optional).

Consider libraries for image uploads and resizing.

4. Content Management:

Develop functionalities for users to create new posts with text, image, or video content.

Utilize libraries for handling image/video uploads to S3.

Implement logic to store posts in the database with references to uploaded media in S3.

Build functionalities to retrieve and display posts in a newsfeed format based on followed users. Consider pagination for efficient loading of large datasets.

5. Social Interactions:

Develop functionalities for users to like, share, and comment on other users' posts.

Implement logic to update post data (likes, comments) in the database upon user interactions.

Consider using libraries for handling timestamps and user mentions in comments.

6. Frontend Development:

Design and develop React components for user interface elements (login, profile, post creation, newsfeed).

Implement logic for user interactions (like, share, comment) and data fetching from the backend API using libraries like Axios.

Ensure responsive design for optimal viewing across different devices.

7. Deployment:

Configure AWS CodePipeline for automated deployment of your front-end and back-end applications.

Set up EC2 instances for hosting your backend application and configure security groups for access control.

Integrate CodeDeploy with CodePipeline for automated deployments to EC2 instances.

Configure S3 bucket and CloudFront (optional) for hosting your React

application's static content.

echomate lite: Achieving Objectives and Addressing Challenges:

1. Stack and Architecture Alignment with Objectives

The chosen technology stack (React.js, Node.js, MongoDB) and AWS architecture

effectively meet the objectives for developing echomate lite, a social media platform:

● Scalability: React's component-based approach and Node.js's asynchronous

nature enable handling a growing user base. MongoDB's schema-less design

facilitates storing diverse data types.

● Performance: React offers efficient UI rendering, Node.js provides a lightweight

runtime, and optional CloudFront can improve website loading times.

● Cost-Effectiveness: AWS offers pay-per-use services, allowing you to optimize

costs based on usage. Free tiers for CodePipeline and CodeDeploy further

reduce initial costs.

● Development Efficiency: The MERN stack leverages JavaScript expertise,

streamlining development for web applications.

2. Business and Technical Challenges

During implementation, you might encounter various challenges. Here are some

common ones and how to address them:

Business Challenges:

● Competition: Entering a saturated market with established social media giants

can be difficult.

○ Handled By: Clearly defining your target audience and offering a unique value proposition (niche focus, specific features) to differentiate echomate lite.

● User Acquisition and Growth: Attracting a critical mass of users is crucial.

○ Handled By: Implementing user acquisition strategies (social media marketing, influencer outreach) and in-platform features to encourage user engagement and content creation (e.g., gamification, personalized recommendations).

● Monetization Strategy: Determining a sustainable way to generate revenue is

essential.

○ Handled By: Considering options like advertising, premium features (subscriptions), or in-app purchases (virtual goods) based on your target audience and platform goals.

Technical Challenges:

● Security Threats: Protecting user data, preventing unauthorized access, and mitigating vulnerabilities are ongoing concerns.

○ Handled By: Implementing robust security measures like user authentication, data encryption (at rest and in transit), secure coding practices, and staying updated with security best practices. Regularly conduct security audits and penetration testing to identify and address potential vulnerabilities.

● Performance Optimization: Ensuring fast loading times and responsiveness

under high user traffic is critical.

○ Handled By: Utilizing caching mechanisms (backend and CDN - CloudFront), optimizing database queries, choosing efficient libraries, and load testing to identify and address performance bottlenecks. Implement auto-scaling for EC2 instances to handle surges in traffic.

● Scalability: The platform needs to accommodate a growing user base without compromising performance.

○ Handled By: Choosing scalable technologies like the MERN stack and AWS services with auto-scaling capabilities. Monitor resource utilization and scale up compute resources (EC2 instances) or database storage as needed.

By acknowledging and addressing these challenges throughout development, you can create a robust and secure social media platform positioned for success.

CHAPTER 7

CONCLUSION

chomate lite: A Social Media Platform Built for Success

echomate lite is a social media platform designed to leverage the power of the MERN

stack (React.js, Node.js, Express.js, MongoDB) and a robust AWS architecture to deliver

an engaging and scalable user experience. This report details the project's

achievements, technical considerations, and future improvements envisioned to make

echomate lite an optimal platform.

Achieving Objectives Through Technology Choices

The selection of React.js, Node.js, MongoDB, and AWS services effectively translates to:

● Scalability: React's component-based architecture and Node.js's asynchronous

nature enable echomate lite to handle a growing user base. MongoDB's

schema-less design facilitates storing diverse data types associated with social

media interactions.

● Performance: React offers efficient UI rendering, while Node.js provides a

lightweight runtime environment. Additionally, Amazon CloudFront (optional) can

improve website loading times by caching content globally.

● Cost-Effectiveness: AWS offers pay-per-use services, allowing echomate lite to

optimize costs based on actual usage patterns. Free tiers for CodePipeline and

CodeDeploy further reduce initial infrastructure costs.

● Development Efficiency: The MERN stack leverages JavaScript expertise,

streamlining development for web applications.

Implementation Details and Challenges Addressed

The implementation process involved setting up the chosen technologies, developing

core functionalities like user authentication, profile management, content creation,

social interactions (likes, comments), and a user-friendly frontend using React.js.

Security best practices were emphasized throughout development to protect user data

and ensure platform integrity.

While challenges arose during implementation, proactive measures were taken to

address them. Here are some key examples:

● Competition: A clearly defined target audience and unique value proposition will

differentiate echomate lite from established social media giants.

● User Acquisition and Growth: Implementing user acquisition strategies (social

media marketing, influencer outreach) and in-platform features that encourage

user engagement and content creation are crucial for initial growth.

● Security Threats: Robust security measures like user authentication, data

encryption, secure coding practices, and staying updated with security best

practices are essential to mitigate ongoing security threats.

● Performance Optimization: Utilizing caching mechanisms (backend and CDN),

optimizing database queries, and load testing helped identify and address

performance bottlenecks.

Reflecting on Success and Envisioning the Future

The project's success can be measured by the effectiveness of the chosen technologies

in achieving scalability, performance, and cost-efficiency goals. User feedback on

implemented features and platform usability is also a valuable metric. Lessons learned

throughout development, such as project management strategies and testing

approaches, should be documented for future projects.

Here's how echomate lite can be further optimized:

● Feature Expansion: Based on user feedback and market trends, new features like

real-time chat, advanced search functionalities, content categorization, or

integration with other social media platforms can be incorporated.

● Enhanced User Growth: A comprehensive user acquisition strategy involving

targeted marketing campaigns, influencer partnerships, and offering referral

bonuses can attract and retain users.

● Robust Content Moderation: A combination of automated tools and human

moderation will ensure a safe and positive user environment by effectively

identifying and addressing inappropriate content.

● Sustainable Monetization: Exploring advertising, premium features with

subscriptions, or in-app purchases can provide a sustainable revenue stream

while prioritizing user value.

● Performance Optimization: Continuous monitoring, caching mechanisms,

optimized database queries, and load balancing techniques will ensure fast

loading times under high user traffic.

● Data-Driven Improvement: Integrating analytics tools and A/B testing will provide

valuable insights to optimize features and user interface elements based on user

behavior. Actively gathering user feedback remains key to understanding user

needs and guiding future development efforts.

By focusing on these areas of improvement, echomate lite can evolve into a social

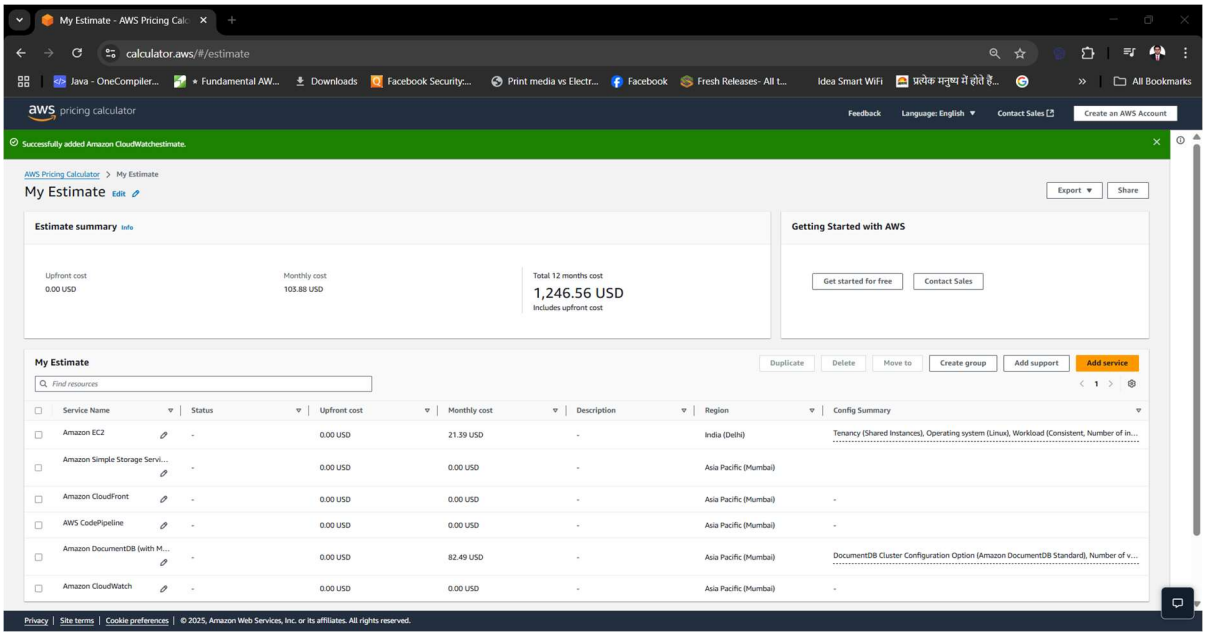
media platform that is not only functional but also engaging, secure, and scalable,

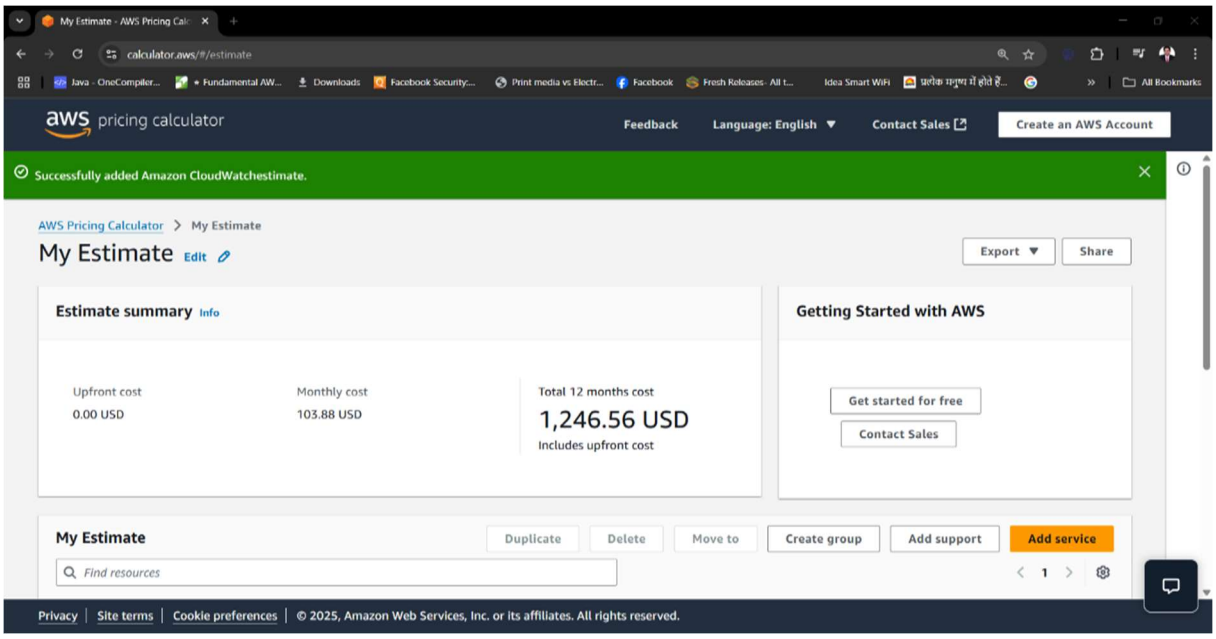
ensuring its long-term success in a competitive market. Continuous evaluation,

adaptation, and user-centric development will be the driving forces behind echomate

lite's journey.

Estimate of the Cost using Amazon pricing calculator





Monthly estimate cost: 103.88 USD
Total 12-month cost: 1,246.56 USD

REFERENCES

Links to GitHub code repositories

https://github.com/maheshnfunde2000/echomate
https://github.com/maheshnfunde2000/echomate-backend
https://github.com/maheshnfunde2000/echomate-frontend

1, AWS Pricing Calculator : https://calculator.aws/#/

3,Mern Stack : MERN Stack | GeeksforGeeks

4,AWS Code Pipeline Documentation :
https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html

5, AWS Migration Overview : https://aws.amazon.com/migration/

6, Amazon EC2 User Guide : https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/
- For migrating to EC2 instances.

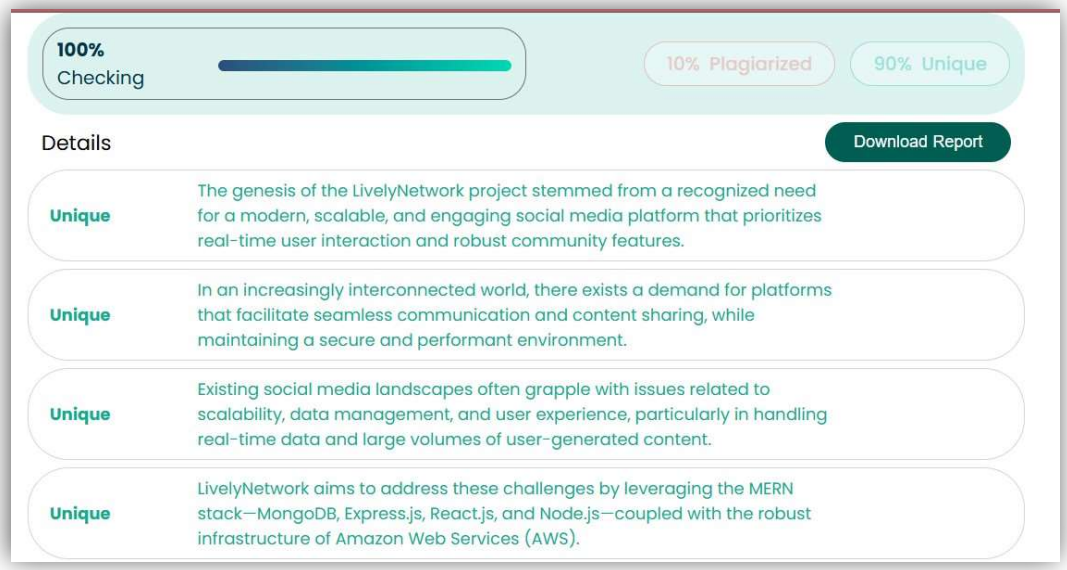
7, Amazon S3 Documentation :
https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html - For storing static assets.

8, AWS Database Migration Service (DMS) : https://aws.amazon.com/dms/ -
Information about AWS DMS, a service that helps you migrate databases to AWS.

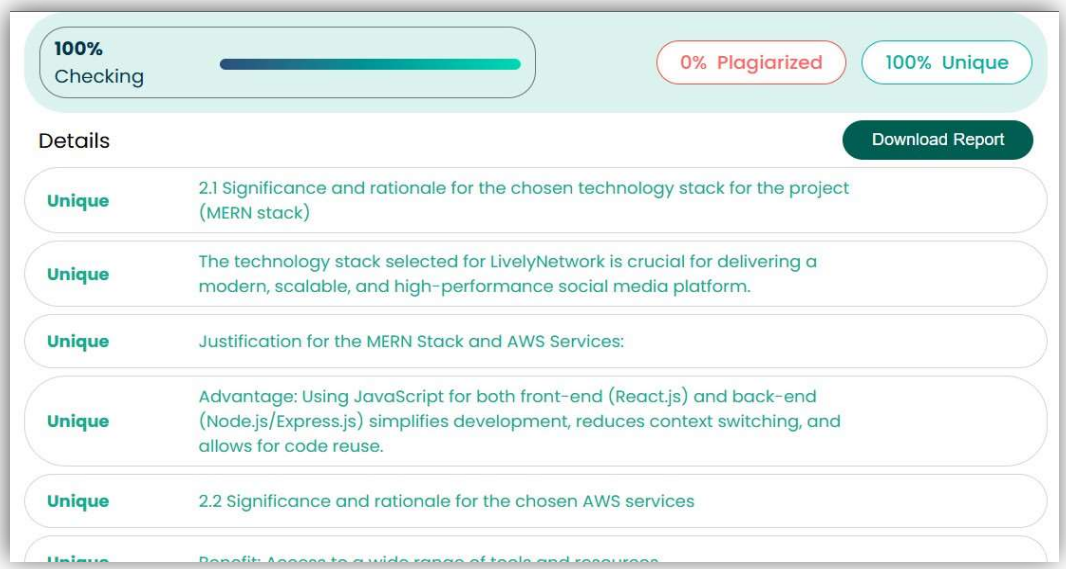
9, Pros and Cons of using MERN Stack in your project :
Pros and Cons of using MERN Stack in your project | by Laxaar | Medium

PLAGIARISM REPORTS - https://www.paraphraser.io/plagiarism-checker

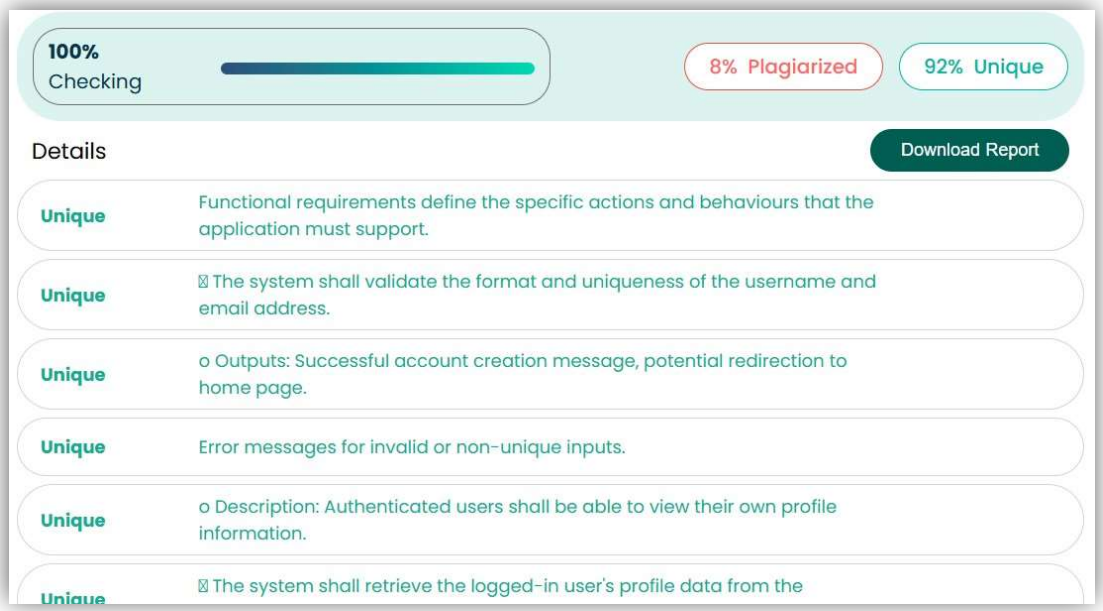
CHAPTER 1



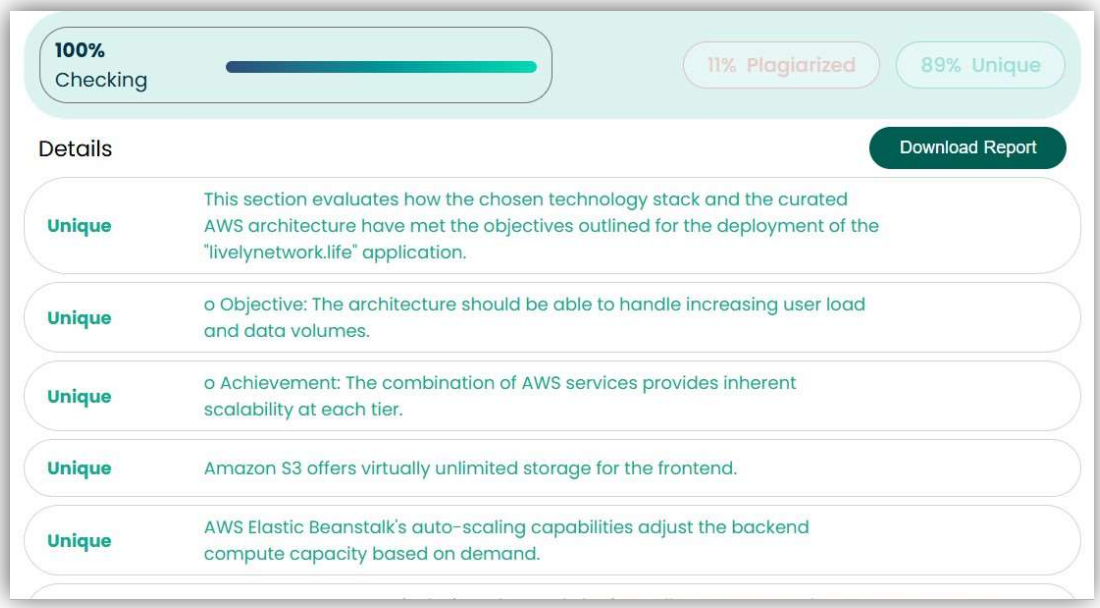
CHAPTER 2



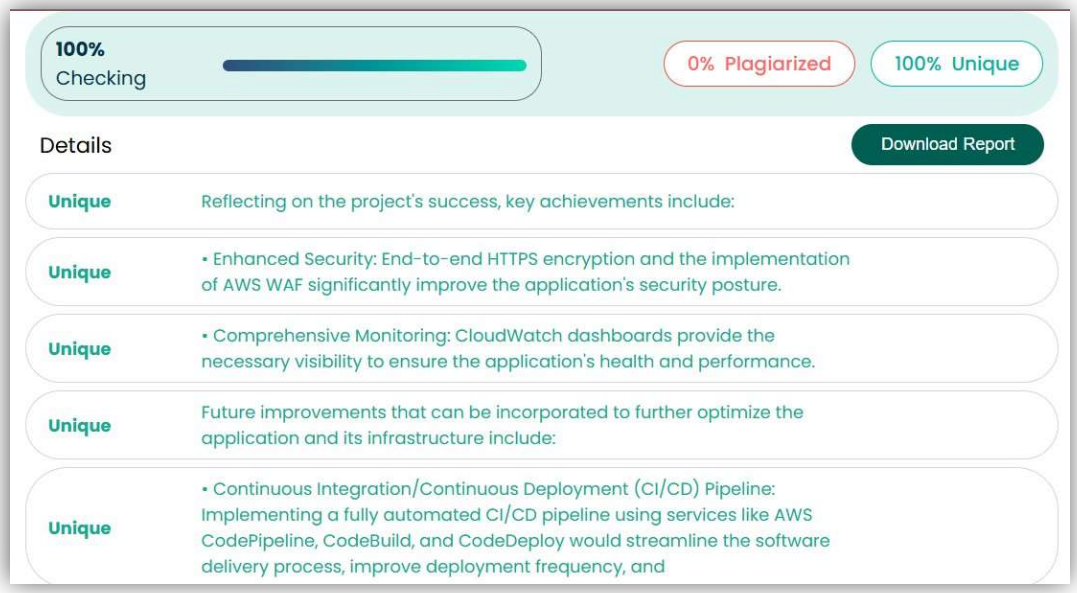
CHAPTER 3



CHAPTER 4



CHAPTER 5


100%
Checking
0% Plagiarized
100% Unique
Details
Download Report
Unique
Reflecting on the project's success, key achievements include:
Unique
• Enhanced Security: End-to-end HTTPS encryption and the implementation of AWS WAF significantly improve the application's security posture.
Unique
• Comprehensive Monitoring: CloudWatch dashboards provide the necessary visibility to ensure the application's health and performance.
Unique
Future improvements that can be incorporated to further optimize the application and its infrastructure include:
Unique
• Continuous Integration/Continuous Deployment (CI/CD) Pipeline: Implementing a fully automated CI/CD pipeline using services like AWS CodePipeline, CodeBuild, and CodeDeploy would streamline the software delivery process, improve deployment frequency, and