**Serverless Webpage**

Submitted to Manipal University Jaipur

towards the partial fulfillment for the award of the degree of

**Bachelor of Technology in Computer & Communication Engineering**

By

**Onkar Deokate**

219303061

Section A

Under the Guidance of

**Ms. Anshika Malsaria**

A close up of a sign

Description automatically generated

**Department of Computer & Communication Engineering**

**Manipal University Jaipur**

**2023-2024**

**Theme Park Management System**

1. Introduction

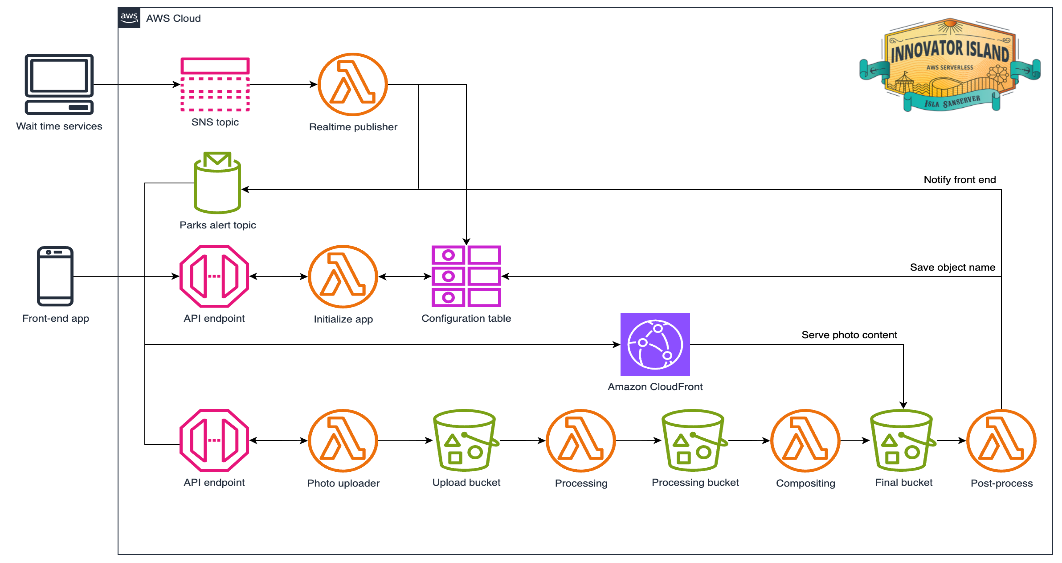
This document aims to provide a comprehensive understanding of the system's architecture, components, features, and deployment procedures. The Theme Park Management System is a sophisticated solution designed to efficiently manage the operations of a theme park, offering a range of functionalities including real-time updates, photo processing, multilingual support, analytics, and event-driven notifications.

2. System Overview

The Theme Park Management System consists of two main components: the frontend application and the backend infrastructure. The frontend, developed as a Progressive Web App (PWA) using Vue.js, serves as the primary interface for park visitors. It interacts seamlessly with the backend, which is built on AWS serverless infrastructure, facilitating data processing, storage, and communication functionalities.

3. Architecture

The system architecture embraces a micro-services approach, leveraging various AWS services to construct a scalable, resilient, and cost-effective solution. Key components include AWS Lambda for compute, API Gateway for RESTful API creation, Amazon S3 for storage, DynamoDB for NoSQL database functionality, Cognito for user authentication, CloudFront for content delivery, Kinesis for data streaming, EventBridge for event-driven architecture.



4. Getting Started

* Prerequisites:

- An AWS account with appropriate permissions.

- Installation of AWS Cloud9 IDE for development purposes.

- Git installed on your local machine.

- Node.js runtime environment.

- Amplify CLI and SAM CLI for streamlined deployment procedures.

* Setup Instructions:

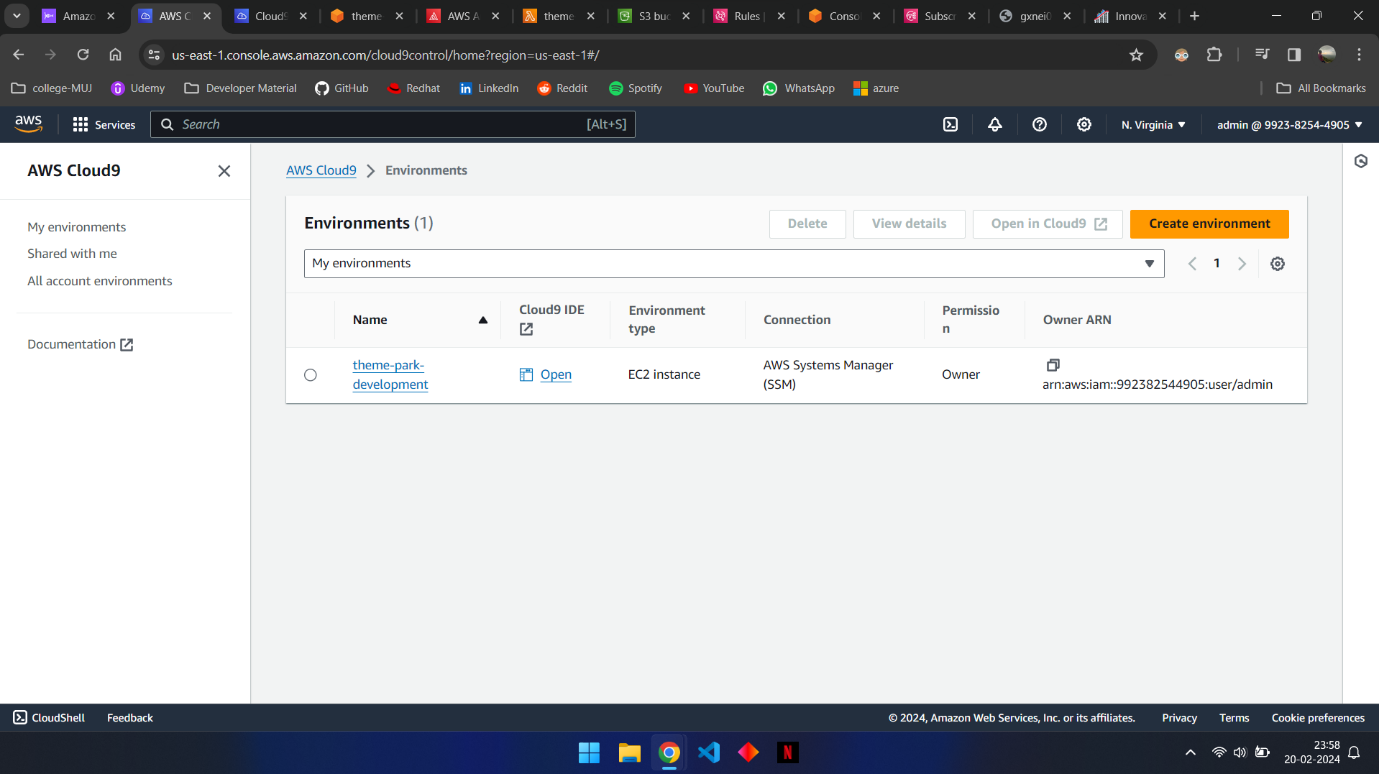
1. Clone the project repository from GitHub.

2. Establish an AWS Cloud9 environment within your AWS account.

3. Utilize SAM CLI to deploy the backend infrastructure.

4. Configure and deploy the frontend application using Amplify CLI.

5. Execute comprehensive tests to validate deployments and ascertain functionalities operate as expected.

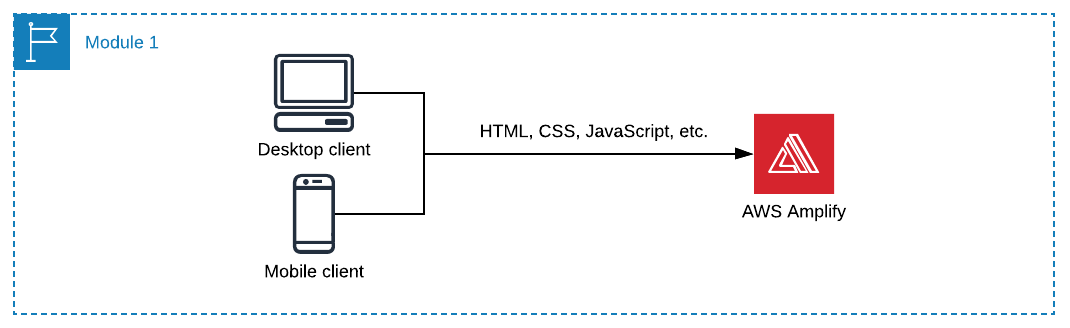


A screenshot of a computer

Description automatically generated

5. Frontend Configuration

The frontend application is seamlessly hosted on AWS Amplify Console, providing an integrated Git-based workflow for continuous deployment. It encompasses static web resources, comprising HTML, CSS, JavaScript, and image files, meticulously served via Amazon S3. The frontend effectively interfaces with the backend through API Gateway endpoints, facilitating real-time ride updates, photo processing functionalities, and multilingual translations.



A screenshot of a computer

Description automatically generated

6. Backend Infrastructure

The backend infrastructure is meticulously constructed utilizing AWS serverless services orchestrated via AWS SAM. It encompasses AWS Lambda functions, driving core business logic, API Gateway for seamless RESTful API creation, S3 buckets for static asset storage, DynamoDB tables for persistent data storage, Cognito for secure user authentication, and CloudFront for optimized content delivery. The backend seamlessly supports real-time features, photo processing capabilities, translation support, analytics, and event-driven notifications. A diagram of a blockchain function

Description automatically generated

A screenshot of a computer

Description automatically generatedA computer screen with a white box

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA computer screen shot of a computer

Description automatically generatedA screenshot of a computer

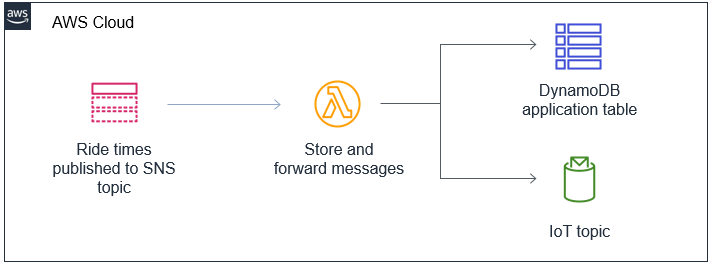
Description automatically generatedA screenshot of a computer

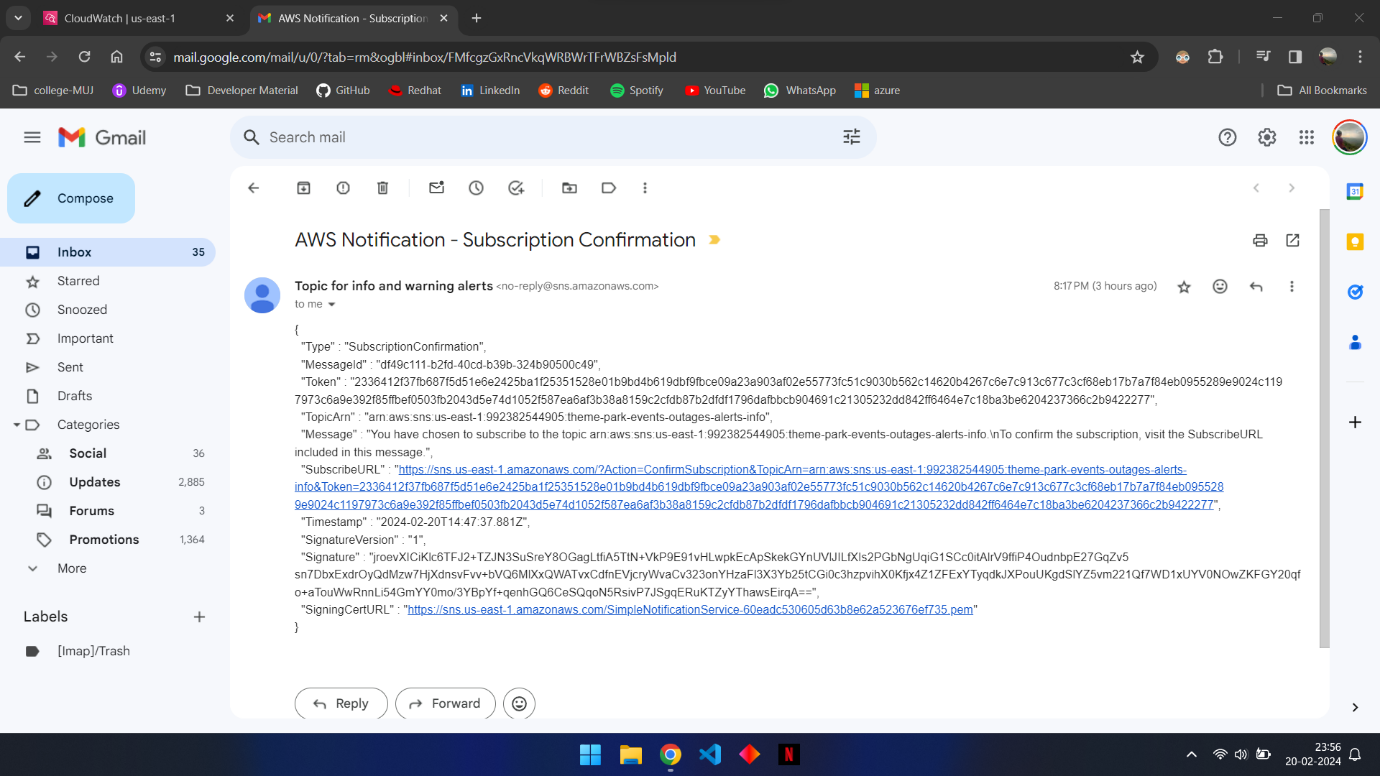
Description automatically generatedA screenshot of a computer

Description automatically generated

7. Real-time Features

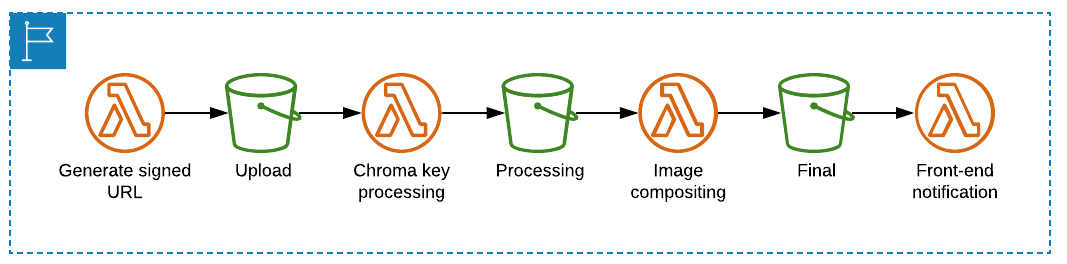
Real-time ride updates are seamlessly disseminated to park visitors leveraging AWS Lambda functions, SNS topics, and IoT Core. Periodic updates are systematically published to a designated SNS topic, subsequently triggering Lambda functions to fetch, process, and disseminate the latest ride data sourced from DynamoDB. The processed data is then channelled to the frontend application via IoT Core, meticulously displayed to visitors in real-time.





8. On-ride Photo Processing

Park visitors are afforded the opportunity to capture and personalize on-ride photos through the photo processing feature. Upon photo upload, Lambda functions execute chromakey processing to eliminate background elements, conduct compositing to embed theme park graphics, and undertake post-processing to finalize the image. The processed photo is diligently stored within an S3 bucket, subsequently integrated into the frontend application for visitor viewing and sharing purposes.



9. Translation Support

The frontend application extends support for multilingual translations, enhancing accessibility for diverse audiences. Amazon Translate is judiciously employed to generate translations for resource files, systematically deployed through Amplify Console. Visitors are empowered to seamlessly toggle between diverse languages, thereby accessing park information and directives in their preferred language.

A screenshot of a computer screen

Description automatically generated

A computer screen shot of a computer screen

Description automatically generated

10. Event-based Architecture

The Theme Park Management System strategically embraces an event-driven architecture to bolster scalability and flexibility. Amazon EventBridge proficiently orchestrates event routing and processing between disparate components, effectively decoupling producers, and consumers of events. Configured rules effectively filter and route events predicated on predefined patterns, thereby enabling seamless integration with extant infrastructure. Real-time alerts and notifications are judiciously triggered based on pertinent event patterns, ensuring expedited responses to critical incidents.

A diagram of a company

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated

A diagram of a theme park

Description automatically generated

A close-up of a logo

Description automatically generated

A screenshot of a phone

Description automatically generated

11. Conclusion

The Theme Park Management System stands as a paragon of innovation, delivering a comprehensive solution adept at proficiently managing theme park operations. From real-time updates to photo processing functionalities, multilingual support, analytics, and event-driven notifications, the system ensures a seamless and enriching experience for park visitors whilst optimizing operational efficiency for park management. By harnessing AWS serverless technologies and embodying a micro-services architecture, the system guarantees scalability, resilience, and cost-effectiveness.

A screenshot of a game

Description automatically generated

A person with his mouth open and hands on his face

Description automatically generatedA person with his mouth open and hands on his face

Description automatically generated

https://main.djn4xp1089ovf.amplifyapp.com/#/

12. References

- AWS Documentation: https://docs.aws.amazon.com/

- Vue.js Documentation: https://vuejs.org/v2/guide/

- SAM CLI Documentation: https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.html

- Amplify CLI Documentation: https://docs.amplify.aws/cli

- OpenCV Documentation: <https://docs.opencv.org/master/>

- https://catalog.us-east-1.prod.workshops.aws/workshops/74d0f3be-7108-4bba-8136-00617a988535/en-US/0-introduction

This documentation offers a detailed insight into the Theme Park Management System, presenting a structured overview of its architecture, functionalities, and deployment strategies. It serves as a comprehensive resource for stakeholders, developers, and enthusiasts seeking a deeper understanding of the system's design and implementation intricacies.