Senior Development Project Documentation

### **Project Overview**

This project aims to demonstrate the creation of a cloud-based solution using AWS services to generate and manage vanity phone numbers. The core components of this system include an AWS Lambda function, DynamoDB, Amazon Connect, and a web application for displaying results.

### **System Components**

1. AWS Lambda Function: Converts provided phone numbers into vanity numbers. The top 5 vanity options are generated based on predefined criteria emphasizing ease of memorability and potential customer engagement.
2. DynamoDB: Stores phone numbers and their corresponding top 5 vanity numbers for efficient retrieval.
3. Amazon Connect: Ideally used to create a contact flow that announces the top 3 vanity options to the caller based on the Lambda function's output.
4. Web Application: Displays vanity numbers for the last 5 callers, providing a simple UI for user interaction.

### **Challenges and Limitations**

1. Amazon Connect with AISPL Account:
   * Issue: The AWS account provided by Amazon Internet Services Pvt. Ltd (AISPL) does not support the creation of Amazon Connect instances.
   * Impact: Unable to demonstrate the Amazon Connect integration within the provided AWS environment.
   * Resolution: Proposed using a global AWS account for demonstration purposes or alternative third-party services that mimic similar functionalities.
2. AWS Free Tier Limitations:
   * Issue: Some of the intended functionalities exceed the capabilities provided in the AWS Free Tier, especially concerning the number of transactions and performance scalability.
   * Resolution: Limited the scope of real-time data processing and provided recommendations for a paid AWS plan for full-scale deployment.

### **Development and Deployment**

* Utilized AWS CloudFormation for resource deployment to ensure consistency and reproducibility across different AWS environments.
* Encountered issues with IAM roles and permissions that required adjustments to ensure secure and unrestricted interactions between services.

### **Shortcuts and Compromises**

* Error Handling: Due to time constraints, comprehensive error handling across all service interactions was not implemented.
* Security: Minimal security configurations were applied, which would not suffice in a production environment.

### **Future Enhancements**

With more time and resources, enhancements would focus on:

* Robust Security: Implement advanced security measures including fine-grained IAM policies and encryption at rest and in transit.
* Performance Optimization: Refine the code and AWS configurations to handle higher loads and optimize cost.
* Extended Functionality: Integration of additional AWS services like AWS Pinpoint for direct SMS feedback based on vanity number selections.

### 

### 

### 

### 

### 

### **Architectural Diagram**



**Conclusion**

This project showcases the ability to leverage AWS services for building scalable applications, despite facing significant challenges related to account limitations and service availability. The solutions and workarounds proposed herein demonstrate a thorough understanding of cloud architecture principles and AWS service capabilities.