Detailed Technical Design

**Project Name**: Customer Communications

**Problem Statement**

Businesses often need to communicate with customers during various stages of the product lifecycle. Currently, product owners rely on the customer communication platform team for creating, viewing, editing, or updating customer communications, which slows down the time-to-market for new products and updates.

**Scope**

The project aims to enable business owners to create and manage their own customer communication letters, utilizing generative AI for prompt-based content creation. Key objectives include:

1. Allow business owners to generate customer communication templates using AI-powered prompts.
2. Store the templates securely in a centralized location.
3. Provide the ability to access, edit, and update these templates when needed.
4. Enable composed letters to be automatically delivered through print or online channels once finalized.
5. Leverage GitHub Copilot for code generation and solution development.
6. Utilize AWS services, React, and language models (LLMs) for the technical design.

**High level Technical Design**:

A diagram of a software company

Description automatically generated

Detailed Solution Design

**Detailed Solution Design**

**Frontend - React Application**

1. **Template Generation:** Design a React-based user interface that allows business users to interact with the Anthropic Claude-v2.1 model through Amazon Bedrock, generating letter templates based on prompts.
2. **Template Editing and Storage:** Enable users to customize the generated templates and save them securely in an Amazon S3 bucket.
3. **Template Composition:** Once a template is finalized, it will be processed by a composition platform to create a customer communication that can be sent via email or print.
4. **Template Management:** Provide access to templates throughout the product lifecycle, allowing updates or changes to be made when necessary.

**Backend - API and Integration**

* **API Gateway:** Set up a REST API using API Gateway to serve as the interface for interactions with the LLM model.
* **Model Configuration:** Utilize the Anthropic Claude-v2.1 model for text generation, conversation, and complex reasoning tasks. Configure the model to access external data sources for Retrieval-Augmented Generation (RAG) and integrate with the knowledge base.
* **Knowledge Base:** Store all communication templates in Amazon S3, creating a searchable knowledge base.
* **Search and Retrieval Integration:** Combine the search functionality with Amazon Bedrock's LLM capabilities to generate context-aware responses based on relevant templates.(temparature – for creating less preditive but more creative content – top p – combination of word is more creative )

**Backend - AWS Infrastructure**

1. **Lambda Functions:** Create AWS Lambda functions to handle Bedrock API requests, using Python for implementation. Ensure that the Lambda functions have the appropriate IAM role and permissions to access Bedrock.
2. **Data Storage:** Use Amazon S3 to store communication templates and serve as an external data source for RAG and the knowledge base.

**GitHub Copilot Integration**

* **Code Completion:** Use GitHub Copilot to assist with code completion, providing suggestions for functions, variables, or logic blocks.
* **Code Generation:** Describe desired functionality in natural language, and let Copilot generate the corresponding code.
* **Debugging:** Utilize Copilot's suggestions to fix errors, optimize code readability, and improve performance.
* **Test Case Generation:** Generate test cases automatically by describing desired functionality in natural language.
* **Commit Text Generation:** Automatically create commit messages by selecting files and allowing Copilot to generate descriptive text based on the code changes.