Concepts Guide

NOTE: This is a sample of a Concepts Guide I have written for an e-commerce application that is hosted on AWS. Any client brand names and feature specifications have been omitted from this document.

Overview

[Brand name] is a marketing technology company that offers their e-commerce clients the power to increase brand engagement and get new revenues. With [Brand name], e-commerce retailers are able to connect with individuals by [brand technology description].

What Is [App Name]?

[Brand name] launched the [App name] to provide store owners with a solution that optimizes e-commerce [brand description here].
Note: Currently, the application is only available for [brand] store owners.

The application allows store owners to:

- Install the application directly from [website]
- Create and customize [sales tool]
- Configure [sales tool] to target customers
- View reporting metrics
- Receive payments

For the next phase of the [App name], the goal of the project is to improve scalability and existing features of the application. The main areas of improvement are:

- Improving the API gateway infrastructure
- Migrating from JS SDK to Web SDK
- Migrating to a public reporting API infrastructure
- Improving empty states to improve the customer experience
- Improving and automating operations processes
- Automating payment detail updates
- Meeting [brand] application requirements

How Does It Work?

The [App name] application is hosted on the Amazon Cloud Platform.

Figure 1 illustrates a high-level architecture of the application comprising its main components and interactions.

[Image of high-level application architecture]

Figure 1. High-Level Architecture of the [App name] Application

The application comprises the following high-level interactions:

- **Store Owner and Frontend:** The store owner installs the application in the App Store and uses the application frontend to create [sales tool and more description].
- **Frontend and Backend:** The frontend makes REST API calls to the backend to collect data and respond to user requests. The frontend is a web application that is written in React, and the backend technical stack comprises Amazon cloud components, REST APIs, and GraphQL.

- **Backend and Store System:** The backend makes queries to the GraphQL Admin and receives event notifications.

- **Backend and Platform:** The backend fetches data from the [brand name[Platform using API endpoints.

- **Store and Platform:** The Store obtains [feature] details from the platform.

- **Customer and Store:** The customer views [feature] displayed in the Store after a transaction.

Features of [App name]

The application comprises the following features:

- Creating and Customizing Placements
- Configuring Offers and Targeting
- Viewing Reporting Metrics
- Receiving Payments

Creating and Customizing [Feature name]

[Description of feature.]

Configuring [Feature name] and Targeting

Store owners can configure the categories to show to their customers. On the **Settings** page, they can block industry verticals or sub-verticals to hide those related categories from customers.

Viewing Reporting Metrics

On the **Overview** page, store owners can view a dashboard that contains the following metrics:

- **Revenue:** The total revenue earned by store owners.
- **Value per Transaction:** The revenue earned per order.
- **Transactions:** The total number of orders from the store.

The metrics on the dashboard can be filtered by date and currency.

Receiving Payments

To receive payments, store owners need to provide a PayPal email address in the **Settings** section. The revenue amount depends on the number of orders customers make.