To: Ron Roe, Alex Kim
From: Lawrence McDaniel
Date: 23-November 2017
Re: FAT Brands Data Loader Design Document and System Architecture

Application URL: http://dataloader.fatbrands.com
Source Code Repository: https://github.com/lpm0073/Fatbrands-Loader

Business Objective. The Data Loader ("DL") will enable a small group of accounting employees or 3rd party delegates to upload weekly store-level sales and traffic data to the FAT Brands consolidated data warehouse. Stores use a variety of POS systems, and accordingly they report in a variety of formats. A single point person in the LA corporate office is responsible for receiving and consolidating the weekly performance reports and putting all of the data into a standardized format recognizable by the DL. The DL recognizes a single standardized file format which is a tab-delimited text files with several standardized column names. This point person is assumed to be the primary user of the DL.

The DL provides a basic set of utility features:

- A file viewer that shows uploaded files (name, size, load date) in a standard file system tree view layout
- An uploader
- A simple set of database reporting tools intended to provide minimal feedback to the user that uploads were successful and are reflected in the database contents

Minimum System Requirements. the DL is browser based. It requires a stable Internet connection and a browser, preferably Chrome though other browsers probably work fine as well. DL runs on any kind of desktop computer, including Windows, OS X and Linux desktop operating system. Technically, DL will also run from the browser of iPads, Galaxy Pads and even mobile phones, however, this would be a cumbersome experience.

Technical Design Overview. The DL is a serverless web app relying on a host of open source libraries and frameworks and pay-as-you-go cloud-based infrastructure services. Technology infrastructure services are provided by Amazon Web Services (AWS). Briefly summarizing, Amazon makes available on a pay-as-you-go basis all of the IT infrastructure services that they use for their own web properties. Thus, all of these are managed services requiring minimal post-deployment administration, and their scalability far exceeds the needs of our project.

Front End. The DL front end is a single-page web app built entirely with Angular 5, a free javascript framework developed by Google, and, Bootstrap, an open-source application framework developed by Twitter. Angular/Bootstrap is a popular combination nowadays as it enables small teams to rapidly create commercial-grade, device agnostic enterprise-class software products. As an added bonus, Angular projects are entirely browser based, which means that they can be served from a low-cost cloud drive as opposed to a Linux-based web

server. Accordingly, Angular projects tend to require minimal post-deployment support and administration.

Back End. The DL depends on several cloud-based services. We are using AWS exclusively for back-end services. There are a few noteworthy advantages to this approach. First, the features provided by each individual AWS service tend to far exceed the requirements of typical projects, which means that leveraging these services enables us to provide robust software while working very fast. Second, AWS services are designed to easily interoperate, which simplifies system integration. Additionally, these are managed services which means that we are not encumbered with administration nor cost-of-ownership related to any given service that we use. Lastly, these are serverless services which means that we are not encumbered with provisioning server resources to match user demand. Following is a summary of the specific back-end services DL will use.

AWS Cognito	Cognito is a full-featured user authentication and user management platform. It keeps track of passwords, enforces policies for user names and passwords, communicates with end users via email and SMS text message when they forget their password or want to change it for any reason. It maintains logs of login and account activity. It synchronizes users' application data across devices. This is tedious mundane stuff. But, it's important to get this stuff right.
AWS Cloudfront	Cloudfront is Amazon's content delivery network. We are using Cloudfront as a superior low-cost alternative to a traditional web server.
AWS S3	S3 means "Simple Storage Service". It is Amazon's flagship cloud infrastructure product. Our web pages, images and CSS and JS files will be stored on S3. We will also store original copies of users' upload source files in S3.
AWS RDS	RDS means "Relational Database Service" and is Amazon's new managed database service. We use RDS to host our MySQL warehouse database. This is the "core" infrastructure service for this project.
AWS DynamoDB	DynamoDB is a "NoSQL" database. It is a low-cost alternative to MongoDB. We use DynamoDB to store non-critical data such as user login activity.
AWS Lambda	Lambda is a newish concept. This is a service in which we are able to run snippets of our computer code on Amazon's computer servers, and we only pay for the computing cycles consumed each time our code runs. This is an extremely economical and scalable alternative to "renting" our own server from a cloud provider like AWS. We will create Lambda to do the real processing behind the DL application. For example, when DL validates an

upload file, or inserts the file contents to our MySQL database, it will be a Lambda function that does this work for us.

AWS API Gateway

The DL application depends on a REST interface to retrieve and post data. AWS API Gateway provides a user-friendly integrated development environment that allows us to create and host a commercial-strength API for our application that connects to the various AWS services like for example, the RDS MySQL database, the DynamoDB, our S3 cloud drive, or our Lambda functions.