Margin Markup - Proposal

Contents

Project Pricing

We request a work description, including the number of approximate hours and the hourly rate. We also request a payment structure that requires segmented payments upon accomplishing specific milestones.

PDF Mockup is Companion Document

Rather than copying into this document copious imagery from the mockup, "**COGS Prototype1-96.pdf**," provided by the customer, the mockup PDF is rather considered part of this document. Refer to the PDF when considering this document.

Terminology

- COGS—cost of goods and services
  - Current COGS – COGS based on cost of ingredients obtained from currently-used purveyors;
  - Potential COGS – COGS computed by obtaining all ingredients from the purveyor with the lowest price for each ingredient.
- GP—gross profit = retail - COGS
- Ingredient—food item which, when combined with others, comprises a recipe or menu item.

- Margin Markup—working title for this software product.
- menu item—a recipe composed of ingredients.
- Purveyor—a food vendor who sells ingredients to restaurants.
- recipe—same as a menu item
- vendor—this term is not used by restaurateurs; rather use the term purveyor.

Purpose / Summary

This document, along with the mockup PDF ("COGS Prototype 1-96.pdf"), describe the proposed software system. Read both documents carefully and make sure all essential elements are included and are described accurately and completely in the quote.

Summary Narrative

Restaurant chefs and managers are responsible for pricing menu items and purchasing foodstuffs such that the business can be profitable. Food vendors, universally referred to as purveyors in the restaurant industry, sell ingredients to restaurants. Pricing changes over time and different purveyors have different pricing for the same ingredients. Food professionals must not only devise menu items which attract food consumers, but they must also determine how much profit is made from each menu item. To remain profitable and competitive, they must also keep track of shifting prices and obtain ingredients from different purveyors who offer a better price, when necessary. Large chains have homegrown systems and software in place to automate and otherwise streamline this process. But most local, non-chain and small-chain restaurants do all this by hand. The Margin Markup system is a web-based SAAS (software as a service) which automates this process for restaurateurs.

Elements Not Included

Any element not explicitly listed in the mockup PDF ("COGS Prototype 1-96.pdf") or in this document should not be included without consent from the customer.

In addition, the following elements should not be included. These are future enhancements possibilities:

- Help content—mainly because nothing beyond a help link is specified in the mockup PDF. The help content will be provided on an additional URL.

  Include a link (
) to a URL that opens in a new tab or something similar.
- Graphic design—glyphs, explicit color schemes, and other assets to be supplied by customer
- Application hosting & backup -- provide a quote for hosting. The hosting will use the url “app.marginmarkup.com”.

System Functions

Most Functionality Specified in the Mockup PDF

The mockup PDF describes or alludes to all system functionality. The information below adds meat to parts of the mockup.

Security / Account Management

Each user will have a login and password. If desired, the system could support social logins, i.e., login using Google / GMail, Facebook, or Twitter.

New Accounts and Self Sign Up

There will be a “register” link on the main login page, which will allow new users to create their own accounts. New accounts will have a free trial period.

Application Security Model

All of the application functions essentially support the same role (see Summary Narrative.) As there is only one role, it does not make sense to partition access to application functions based on user or roles. For this reason, there will only be one type of system user – the end user. Later, in a future add-on project, it may make business sense to create a super user who can perform certain maintenance tasks, for instance.

Platform and Architecture

The system can be built using a combination of Microsoft and open source technologies. We are also very open to suggestions of other technologies that may be more agile and/or open source for further development. While not all will be specified at this point, the following are "short list" candidates:

- **ASP.NET MVC**—this architecture is thin, testable, performant, and provides a high degree of control over the rendered HTML. In general, ASP.NET MVC is faster and more scalable than ASP.NET Web Forms.
- **ASP.NET Single Page Application**—SPA is a paradigm for developing rich client-sideapplications. Rather than always loading physically different pages when navigating among different functions, Javascript is used to switch between "virtual pages" which all reside on the same physical web page. While there are tradeoffs, this approach can make the real and perceived response time more closely match that of a native / non-web application. http://www.asp.net/single-page-application
- **ASP.NET Identity 2.0**—rather than reinventing the wheel, we will utilize this mature identity system. ASP.NET Identity already implements everything we need, including two-factor authentication (should this be desired), social login providers, account confirmation, password recovery, and account lockout. http://www.asp.net/identity
- **Database**—Microsoft SQL Server will provide the data store. The free Express Edition 2014 can utilize up to 1G memory and have multiple databases, each up to 10G in size. This will be more than adequate for hundreds to several thousands of users. Later, the

database can be scaled up to apay-edition of SQL Server. CSS used this approach successfully for the first seven years of operation of a web-based SAAS in use by 30 hosted counties with hundreds of users. All 30 counties performed well sharing a single instance of SQL Server Express Edition with Advanced Services (64-bit).

App Components

- java script for pop ups
- Ajax for all searches
- all screens will be provided and can be sliced up as png's for the software
- interface with Chargify Small Business for subscription payments
- Automatic account creation with free acct for 14 days. Link from website
- API from http://www.supermarketapi.com or http://corp.groceryserver.com/ for retail vendors. Here is the info for part of the API's of retail outlets: http://www.supermarketapi.com. They provide a free acct for 14 days. Feel free to test with your own creds and we can purchase when ready. This only covers Walmart, Sams, etc … not the specific purveyors.
- Ability to use personal API's for preferred purveyor pricing
- Several purveyors simply are not offering API's directly to customers because there isn't existing technology that requires it. Therefore, we need to look at options to streamline the data for our customers. I've struggled through several options and am open to your solutions. The solutions (which is in the demo) should:
  - Allow customers to use existing CSV files.
  - Update the data with minimal user interaction

  There is a service at http://delray.io that states it can pull CSV data from FTP folders and by other means and create an API that can be used in Margin Markup. This may be a service that we can pay for if it can be integrated into the software (in which we setup the API for them and then set the calls to their FTP account to update as needed).

Requirements for end users:

Browsers

The latest version of following browsers will be supported:

- Google Chrome
- Internet Explorer
- Mozilla Firefox
- Apple Safari
- Opera

Local Storage

Local storage is a recent advent in modern browsers. Browsers usually have a one-time consent to the system's usage of local storage. As stated by http://www.w3schools.com/HTML/html5_webstorage.asp:

With HTML local storage, web pages can store data locally within the user's browser.

Earlier, this was done with cookies. However, local storage is more secure and faster. The data is not included with every server request, but used ONLY when asked for. It is also possible to store large amounts of data, without affecting the website's performance. The data is stored in name/value pairs, and a web page can only access data stored by itself.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

All modern browsers support local storage.