Barket Project Proposal

10/23/2017

Team 1

Kate Strombom – kstrombom@ku.edu

Jack Stevens – jackstevens@ku.edu

Denis Sehic – denissehic21@gmail.com

Hanwen Cao – hanwencao@ku.edu

Project Description

The premise of our project is to treat bar related items as stock in a mini-stock market. This means that the pricing of each item is dynamic and will increase or decrease based on the amount of stock being purchased of items in the "market". This idea has two underlying opportunities. The first is to capitalize on the novelty of the idea in the United States market by attracting patrons to the bar utilizing this software. We believe that the new format of purchasing items will become popular for its novelty and game-like purchasing. We also will capitalize on an opportunity to improve a restaurant or bars sale of stock. By having the pricing be dynamic, restaurants can begin to sell more items that would normally not get sold often. They can also heavily capitalize on the popular drinks and meals sold.

Our project will be a locally hosted web application. The backend will consist of the algorithm, which will take multiple input criteria to build a customer designed model for pricing. Based on sales made in a POS system, the algorithm will perform calculations within a configurable time period and present the pricing changes on the customer friendly page. The frontend will consist of a clean user interface for presenting the items being sold similar to that of a stock market page. The bar owner will also be able to configure the look of the user interface to create a look like the bar's style.

Project Milestones

MILESTONE	DATE TO BE COMPLETED
DIAGRAMS – WIREFRAME, USE-CASE, OBJECT, ETC. ALGORITHM DESIGN	November 1, 2017
INITIAL PROJECT SET UP	December 1, 2017
ALGORITHM IMPLEMENTATION	February 1, 2018
RESTAURANT OWNER FRONTEND DESIGN	February 15, 2018
CUSTOMER FRONTEND DESIGN	March 15, 2018
TESTING AND TWEAKING	April 15, 2018

Project Budget

We intend to utilize open source software and languages for the creation of this project so that the cost involved will only be labor related. The below are the backend, front-end, and database technologies we have chosen to work with so far. They are all free but will require time to become accustomed to because they are mostly new technologies. We do not foresee Node.js and PostgreSQL requiring much learning time though based on prior experiences. All these technologies will be required as soon as the design milestone has been finished.

- Trello
- Github
- Node.js
- PostgreSQL
- Express.js
- Angular
- Ionic

Much later, near April 2018, depending on our progress, we may want to deploy the project from our localhost to a server that would likely be utilized in the restaurant to test in a more realistic

environment. We intend to find a free or cheap server to test with if this is the case. I believe KU currently offers free servers to students to test with.

GitHub

https://github.com/kstrombom/stock-restaurant

Work Plan

Milestones

Tasks	Strombom	Cao	Stevens	Sehic
Designs and Diagrams	X	X	X	X
Initial project set up	X	X	X	X
Database	X		X	
Algorithm			X	X
Algorithm testing	X	X		
Backend			X	X
Frontend	X	X		
Final testing and tweaking	X	X	X	X

Gantt Chart

For the full chart with bar tracking go to:
https://app.smartsheet.com/b/publish?EQBCT=3cbb6aac770744549f6f2b4bdbba8b91

Task Name	Start	Finish	Assigned To	Duration
Initial Planning	9/24/2017	10/2/2017		**7d**
Concept discussion	9/24/2017	9/27/2017	All	**4d**
Research technologies	9/28/2017	10/2/2017	All	**3d**
Diagrams	10/2/2017	11/3/2017		**25d**
Gantt Chart	10/2/2017	10/6/2017	All	**5d**
Algorithm and Database Schema	10/9/2017	10/13/2017	Jack, Denis	**5d**
Web architecture pattern	10/9/2017	10/13/2017	Kate, Jessica	**5d**
Project Proposal and Video	10/15/2017	10/27/2017	All	**11d**
Backend API documentation	10/29/2017	11/3/2017	Jack, Denis	**6d**
Wireframes	10/29/2017	11/3/2017	Kate, Jessica	**6d**
Initial Project set up	11/5/2017	11/21/2017		**13d**
Initial Stack Set up	11/5/2017	11/21/2017	All	**13d**
Migrate and seed databases with test data	11/19/2017	11/21/2017	All	**3d**
Backend	11/27/2017	4/27/2018		**79d**
Write test scripts	11/27/2017	12/8/2017	Jack, Denis	**10d**
Implement initial service code	1/17/2018	2/9/2018	Jack, Denis	**18d**
Implement API for POS to Barket DBs	2/12/2018	2/21/2018	Denis Sehic	**8d**
Implement API for Barket DB to Service	2/22/2018	2/28/2018	Denis Sehic	**5d**
Implement Config file logic and GUI	2/12/2018	2/28/2018	Jack	**13d**
Implement basic POS GUI executable	3/1/2018	3/16/2018	Denis Sehic	**12d**
Implement algorithm	3/1/2018	3/16/2018	Jack	**12d**
Test, Tweak, Optimize	3/26/2018	4/27/2018	Jack, Denis	**25d**
Frontend	11/27/2017	4/20/2018		**74d**
Learn Stack, implement simple website	11/27/2017	12/8/2017	Kate, Jessica	**10d**
Add database connection and models	1/17/2018	2/7/2018	Kate, Jessica	**16d**
Implement drink component	2/8/2018	2/16/2018	Kate Strombom	**7d**
Implement tier component	2/18/2018	2/28/2018	Kate Strombom	**9d**
Implement personalized config components	2/8/2018	2/28/2018	Jessica	**15d**
Implement website view	3/1/2018	3/9/2018	Kate, Jessica	**7d**
Implement mobile view	3/12/2018	4/6/2018	Kate, Jessica	**15d**
Test, Tweak, Optimize	4/9/2018	4/20/2018	Kate, Jessica	**10d**

Preliminary Project Design

Algorithm

The algorithm will utilize a known calculation for the dynamic pricing of supply and demand. At initial set up for the client (bar owner) we will have them interact with a simple GUI written in C#.NET that takes in their desired profit margin, the time period over which prices change, and can set minimum and maximum prices for each item in their inventory. The GUI will also allow the user to remove certain

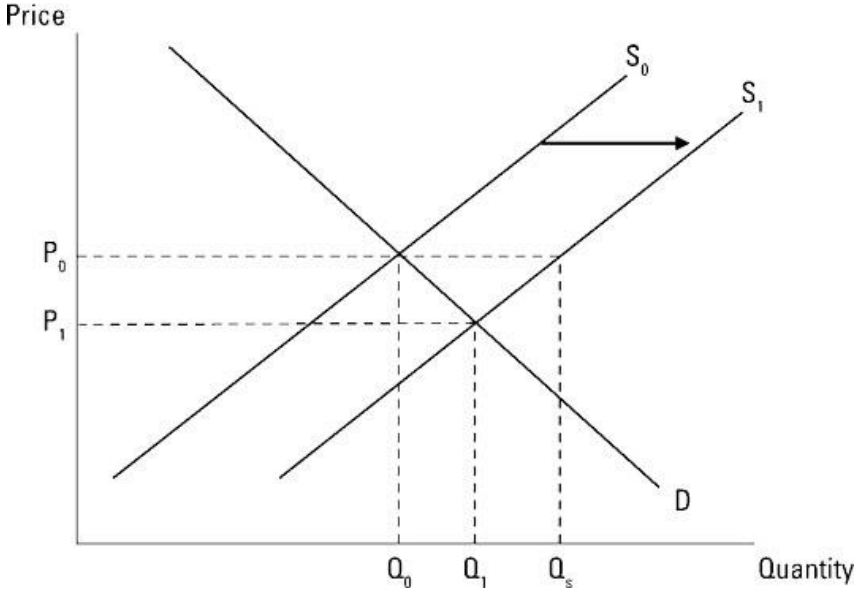
stock from participating in the dynamic pricing. Finally, the GUI will allow the user to group certain items in tiers. For example, the groups could look like cocktails, beer on tap, beer in bottle, wine, etc.

The algorithm will take in a vector<vector<vector<transaction>>> input. The innermost vector will contain all the transactions over the past time period for a particular stock item. The middle vector will represent the tier and the outermost vector will represent the whole of the transactions. The calculation will then be made and update the price in the POS and the customer-facing database.

The gist of the calculation is that every item has a minimum and a maximum price point. Based on the desired profit margin and the amount of stock remaining the prices will change to best fit that margin and get rid of stock. To do this an equilibrium point is calculated for each stock item. This is performed at the tier level and at the overall inventory level.

In addition to the dynamic pricing algorithm, we will calculate the change in pricing from the last time period to simulate a stock market.

Supply and Demand Chart



Tech Stack

Our technical stack can be split into two parts, the service running on the server and the web application stack.

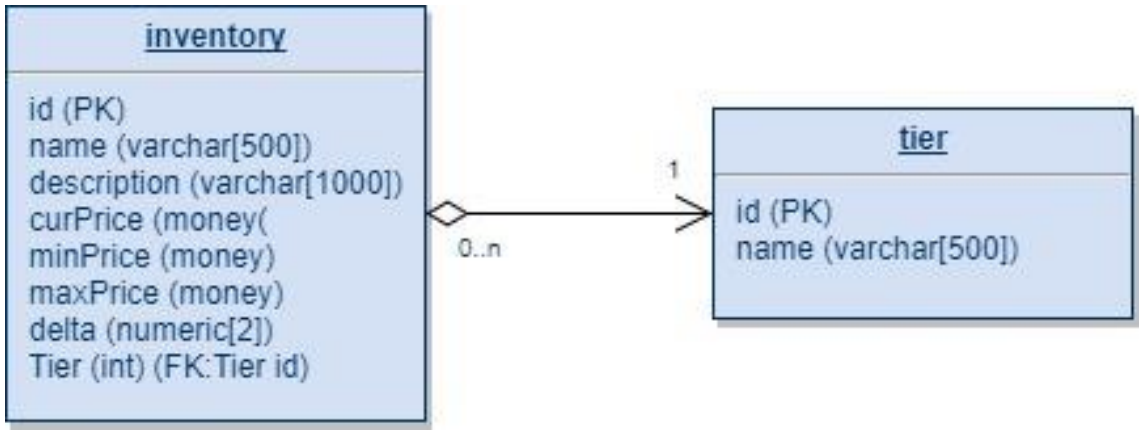
The service will be written in C# and will continually perform the algorithm calculations every time period. There are also two parts to the service that will interact with the set up, but will be in the form of an executable. The executable will be a basic POS that the servers would see for demonstration and adhoc testing purposes. The second part of the executable will be the configuration GUI. This configuration will store the profit percentage, min and max price points, and allow for grouping into tiers and exclusion from the dynamic pricing. These GUI's for now will be written in C#.NET, but will likely need to be written for Macs as well if time permits.

The web application will be written in Angular, express, and ionic. Angular was chosen because it has data bindings and is a well known framework with great documentation. Plus, a member on the team

was interested in trying Angular out. LESS will be utilized for the styling. Express.js and Node.js will be controlling the building and backend logic for the web application. Ionic will be used to transform the native web application into a native app for android and iOS phones. The architecture pattern will be Model – View – View Model (MVVM). There will be two pages in the web application: the customer view and the management configuration. The management configuration page will allow management to choose a color scheme for the customer page, assign a name for the restaurant, and upload an icon. This information will be stored in a configuration file on the server. The customer view page will consist of three main components: a title bar, a tier, and a stock item. The tier will contain stock items.


Database Schema

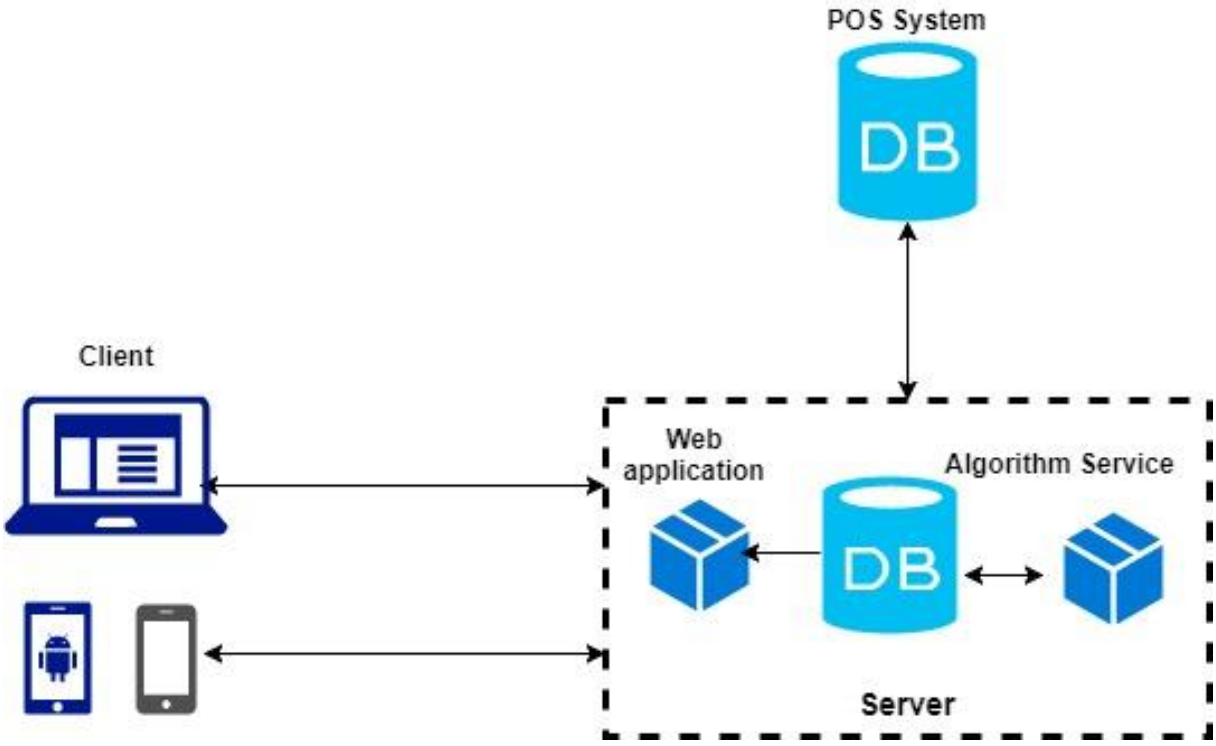
All data for the web application and the service will be stored in a PostgreSQL database. This language was chosen because it is an open source language with heavy documentation and a relational database was desired for the architecture. POS systems are often represented relationally and we wanted to keep consistent so that integration of a POS system in the future would be easier.



Overall Architecture

The server will house the Barket database, the algorithm service, the two executables, and the web application. It will be a locally hosted server, which also makes development easy. As stated above, the profit margin and the time period may be configured by the bar owner. Every cycle of the time period, the algorithm service will contact the POS system via a basic API and retrieve the transactions, items in stock, and vendor price. The algorithm will then perform calculations on this data and store the final price point and its delta in the Barket database. The final price point will also be stored into the POS system so that the bar owner may still use all the software he has already purchased with his POS.

When the calculation is finished the service will send a server side event to the client, updating the price and deltas in the model, thus updating the view. No information from the client web and mobile applications will be stored in the database. All the configuration required will be stored in a config file in the web application source file.



Ethical and Intellectual Property Issue

Ethical Issue

Before examining our project using ethical systems of thinking, we must decide if ethics even applies to our project. To sum up our project: we are creating an application intended to make purchasing alcohol a more interactive and fun experience for customers. The goal is also to increase the profits of the bar owner through novelty and to ensure that their stock moves evenly and quickly. Creating this project is not a voluntary choice and alcohol is a known hot topic amongst our society. When we enter college, for example, we are forced to take an online tutorial examining alcohol consumption and the effects it may have on the body and mind. Alcohol can kill people in overconsumption (an obvious societal no-no) and it could be argued that by making the drinking experience more fun with our project that we are

possibly inviting more alcohol related problems to arise with patrons that want to keep on buying and testing the mini-stock market.

Utilitarianism is a way to examine ethics by considering the greatest amount of utility that our project's actions may take. The action our project takes is to increase profits for the bar owner by making a fun drinking experience for patrons and using an algorithm to ensure profit margins are reached. Our audience could just be bar owners because that is the group of people our project is geared towards. However, Barket is also intended to be used by bar patrons and potential patrons so we may want to include them in our audience as well. Finally, we examine the utility of this project on the audience. Bar owners are a small subset of our audience, but they would clearly be affected most by Barket. For this subset of the audience Barket is not necessary, but could potentially boost profits and move stock for bar owners, meaning more money for their lives. This would also be a very useful tool for the struggling bar owner looking to stand out from the crowd. When looking at the customers or potential customers, our project's utility is lacking. The customer does not technically need our tool to order drinks and for the customers who like stability in pricing it would almost be a negative for our product to enter their established bar. For those customers who find our product fun, it would introduce a bit of excitement and new-ness to their ordering routine. We believe that most customers would enjoy our product. Because our product has very little utility for most bar patrons, a negative utility for a small subset of patrons, and a beneficial utility for bar owners, our project passes the utilitarian test.

In applying the Respect-for-Persons test, we must analyze the alternative actions we have available. One option is to not make this project and do something else, which could potentially have its own consequences but let's assume we pick something with no consequences. The other option would be to create the pricing system for general purpose, instead of gearing it towards bar patrons. This option would not have any foreseeable consequences to society, however, if we adopted this concept it would greatly increase the scope of our project. We believe this would be a great option to pursue if we decide to continue the project past the senior design time. We intend to make the system as modular as possible to accommodate changes such as these for the future. Finally, the consequences of our current course of action. We put ourselves in the situation where we were at a bar with this pricing system. The four of us agreed that it would indeed increase the amount that we purchased, but that by using our personal judgement this system would not entice us to overconsume.

Intellectual property

The concept we are creating has been done before and therefore opens us to risk from intellectual property complications. There is currently a group based in the United States called Drink Exchange who is also applying a stock market concept to bars. The algorithm we intend to utilize is based on well-known economic principles and therefore does not fall under intellectual property. Software copyrights can be applied to the expression of this idea, i.e. the Source code and tech stack we intend to utilize. The idea itself is not able to be copyrighted. Because we have designed the tech stack of Barket from scratch, we highly doubt that any competitors will have an identical implementation. Further research needs to be done on the patents given to an idea like this, but for now we assume that the implementations of the idea are quite different.

Change Log

Section	Changed
Project Description	*
Project Budget	*
Project Milestones	
Github	