CROWGLE

Final Report

Arun Balachandran Ganesan
Clifton Lin
Jared Pryor
Sriram Ramasubramanian
Molly Samuels

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The distribution of coupons and deals has drastically changed over the past few years; instead of cutting them out from the Sunday newspaper, we print them out from e-mails and websites, or show them on our mobile phones. The evolution of technology has been a key factor in this rapid change of the coupon industry. With all the sensor data that is now available from the current generation of mobile devices and with the maturing of online profiles, merchants now have the opportunity to provide more effective deals to the appropriate customer demographic. Our goal has been to explore this exciting space of modern deal distribution.

Presently, there are two main types of solutions that are available: the first type provides daily deals for customers to claim such as Groupon, Living Social, etc.; the second type provides deals based on check-in with mobile devices like foursquare and Facebook places. We first identified the overall direction of this space by exploring context-aware deal distribution possibilities for mobile users. From our user research, we saw that customers are often bombarded by coupons through e-mail or postal mail that aren't relevant to them. Merchants expressed that they recognize the need to distribute deals to customers, and have attempted to leverage social media, but do not know how effective it is in increasing traffic to their business.

Crowgle is a deal distribution service that addresses to the needs identified by the merchants and customers. It introduces a "haggling" concept where customers can watch deals as they fluctuate and claim a deal at whatever point that they like. Merchants can dynamically change the amount of the discount for the deal they have posted based on the number of customers who are watching the deal or who have claimed the deal. We have analyzed the technical and financial feasibility to understand that Crowgle is a profitable solution that really puts merchants in control and provide the right deal at the right time for individual customers.

WHAT IS CROWGLE?

Crowgle (crowd-haggling) is a mobile service that allows merchants to post deals for their products and services, and then adjust the price of the deal based on the customer response. Customers will use Crowgle to browse a list of offers from merchants nearby their current location or according to the product categories and/or merchants. When customers see an offer that they are interested in, they can choose to watch it, and will be notified if the price of the product or service goes up or down. At any time, a customer can choose to claim an available deal, meaning they purchasing the item and can use the Crowgle application to redeem the product or service with the particular merchant.

Crowgle allows merchants to monitor the customer response to their posted deals in real time and adjust the prices of the deals accordingly. The customer data obtained from using Crowgle can be used to determine the best price point to sell a certain product or service at, which will help the merchant maximize the number of people who ultimately purchase it. By subscribing to the Crowgle service, merchants will be able to post their offers to the "Crowglers" and will have access to insightful analytics about their customers.

CASE FOR DESIRABILITY

User Research

Throughout the semester, we conducted directed story telling sessions with 18 merchants of local stores and large national chain stores, and with 15 consumers. Based on our findings from the merchants, we decided to focus our design on the smaller, local merchants rather than large chain restaurants and shops, partly because most of the large chain stores have already established their own unique coupon systems that their customers actively use. We would like to highlight the following key findings that had the greatest influence on our final design:

1. Customers face coupon overload
Consumers are bombarded with coupons that come through e-mail and postal mail. This results in coupon overload, where they ignore all or most of these coupons and will miss out on deals that may have actually interested them. We learned that our design must include a feature that filters the coupons based on what each costumer really wants, allowing them to avoid a lengthy process of searching for desired coupons.

2. Customers forget to carry coupons
When customers find a coupon that they want to use, either from e-mail or postal mail, they often forget to bring it with them while they are dining or shopping. We learned that our mobile solution should allow consumers to find and access coupons right when they need them. Coupons should be redeemed straight from the user's mobile phone, eliminating the need to carry around a paper copy, or to memorize a coupon code.

3. Merchants hold Happy Hours to bring customers in during off-peak hours
Many local restaurants hold Happy Hours to bring in customers early in the evening or very late at night right before closing, and the merchants have found these kinds of deals to be very beneficial to their business. Our service would emulate these Happy Hours by allowing merchants to create a "Happy Hour" in real time, whenever they want. If a merchant needs to bring in more customers, they'll post a deal to try and attract additional people. They can then alter that deal based on the customer response (offer a better deal if there aren't enough people, or decrease the offer if too many people have shown up). The goal in our service is to give the merchants a lot of control in the deals that they offer to customers, so that they don't have later regrets for the deals that they post.

4. Merchants cannot calculate the effect of social media

Our research showed us that many local merchants have tried to use social media by creating Facebook pages and Twitter accounts. However, they find it difficult to understand what effect their social media campaigns have on their business. We determined that our design must include a powerful analytics platform to help the merchants learn more about their customers and the deals that they post through the Crowgle Service.

Competitive Analysis

We analyzed competitors who have a presence in the mobile coupon space. We determined that the most successful competitors come in two forms:

1. Services that offer one deal to customers per a day based on their current location
2. Services that use a customer's current location, and their action of "checking in" to a place to offer coupons

Groupon is a service that sends a new local deal (in one of 500 cities across the world) each day to a list of subscribers. The value of the deal is usually a 50% discount of the normal price of a service and/or product, and the "groupon" is usually valid for a year. However, just agreeing to buy a groupon isn't enough to get it. In order for a deal to become valid, a certain number of people must commit to buying that deal. Groupon has become so popular, that the deals are usually on by the first few hours of posting.

According to their website, Groupon has sold over 42 billion deals. For each deal on Groupon, the company will take a 40-50% cut from the money that the customers pay. The typical Groupon demographic is a young, educated female who is middle or upper class. Groupon is easily the biggest player in this space, having infamously turning down Google's offer to acquire the company for $6 billion. While many merchants find that these deals from Groupon are successful, some have reported that they were unable to handle the influx of business that the deal brought in.

Living Social offers a single deal each day, that potentially gives as much as a 90% discount at various businesses in a specific area. Once you have bought a deal, you can share a link with people you know, giving them the option to buy it, and if 3 people buy a deal based off your link, the deal that you purchased is free. The site claims to have experts identifying the best merchants in local areas for the deals. Unlike other coupon services, there isn't a limit to the number of people who are able to buy a particular deal. In addition to the deals, Living Social offers "Escapes", which are trips that can be bought at a discount, and new ones are offered each week. The same rules that apply for the deals apply for the escapes (there is no limit to how many people can buy the escape, and if you get three people to purchase it, your escape is free). There also is a section that offers "365 things to do" in a city, giving tips on various products and services for customers to try.

Google Offers was launched very recently as Google's answer to the success of Groupon. The service is currently in a Beta release phase in Portland, OR. Like Groupon, Google sends a daily deal to its e-mail subscribers, and takes a cut of all the Offers that are sold. Unlike Groupon, the Offers are not necessarily good for a year, but rather for a period of time like 24 hours. Google claims that businesses that create Offers will reach new customers that have prepaid for a service or product.

Foursquare is location based social network in which users to check-in to places on their mobile phones, allowing them to share their current location with their friends and discover new places nearby. Foursquare allows businesses to "claim" these places, which lets the business access statistics about the place and track loyal customers, promote specials, and create deals. When users check in to a place, they can receive a deal that the merchant has created for them. They can also see a list of nearby places, with an icon indicating if there is an available deal. Foursquare offers a variety of deals, so that the merchant can target frequent customers, new customers, or the "mayor" (the user with the most check-ins at that place). This service is free to both merchants and customers to use.

Facebook recently launched a deals feature, to go along with Facebook Places. Like Foursquare, Facebook Places allows users to check in to locations and share this information with their friends. When a user launches Facebook Places, they see a list of nearby places, which include details such as an icon representing the available deals at a given location. Users can also check in with their friends to receive group deals. The service is free to both customers and merchants.

Gowalla, like Facebook Places and Foursquare, also allows users to check-in to different places. Merchants can claim their business on Gowalla and offer promotions when a user checks in. Gowalla does not allow users to discover other places nearby that are running promotions. Like other check-based services, Gowalla is free for the customers and merchants.

While it is not an online coupon service, we also investigated Ebay to learn more about their auctions model in order to better structure our own "haggling" model.

Ebay is a site that allows you to buy a variety of items by bidding on them in an auction against other people. Also, you can become a seller, where you create an auction for a particular item, and sell the item to a buyer if your criteria for an auction is met (such as a minimum price, or if the customer is willing to pay a high enough price to immediately get the item). Buyers have the option to watch particular auctions that they are interested in with out bidding, and sellers can set up “Ebay Stores” to create an area of all the stuff they are selling. There is a review system so that buyers and sellers can rate their experiences on buying, which can help other potential buyers and sellers to avoid bad experiences. Ebay does have insurance to pay back buyers if they purchase an item, but never actually receive it.

Competitive Analysis Comparison Chart

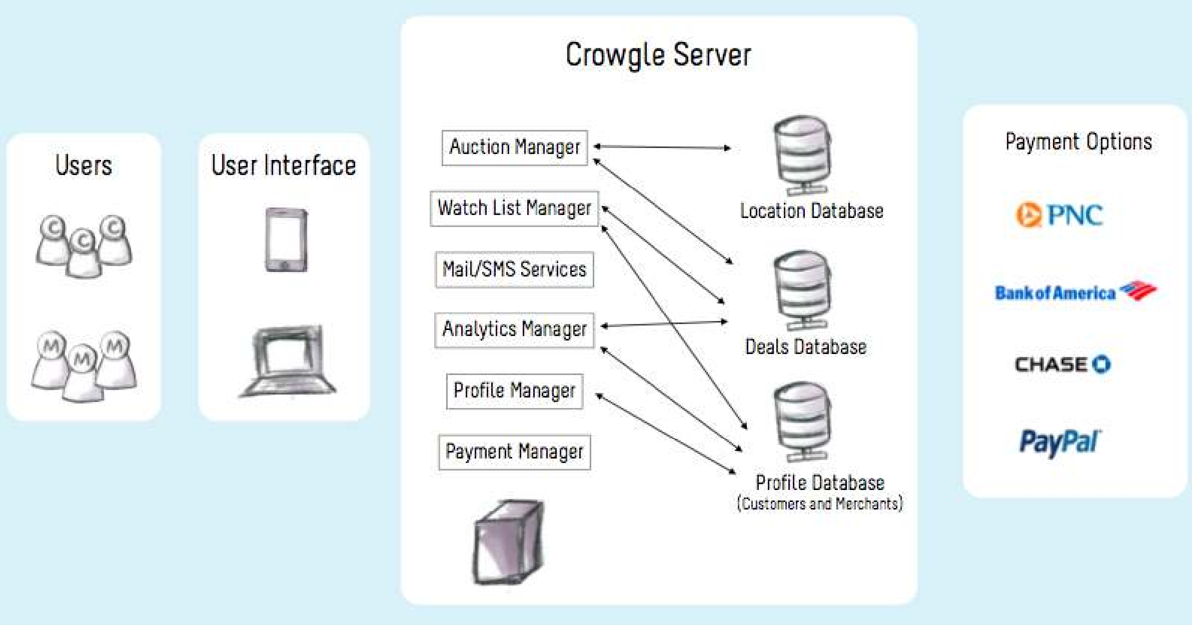
	Charge to Consumers	Charge to Merchants	Social Media Component	Merchant Analytics	Customers "Check in"	Push Notifications	E-mail Notifications
Groupon	Free	50% of coupon sales	N	Y	N	N	Y
Living Social	Free	50% of coupon sales	N	Y	N	N	Y
Google Offers	Free	50% of coupon sales	N	Y	N	N	Y
Foursquare	Free	Free	Y	Y	Y	N	N
Facebook Deals	Free	Free	Y	Y	Y	N	N
Gowalla	Free	Free	Y	N	Y	N	N

How is Crowgle Different?

Crowgle allows users to find nearby deals on their mobile phone. Instead of just providing one daily deal, the deals on Crowgle are highly dynamic, changing constantly so that you won’t expect to see the same pattern of deals. When using the application, users search for deals based on categories, and the system queries automatically by location for these deals. This allows users to get nearby deals without the trouble of checking in. When the location information is unavailable, the system returns results based on criteria indicated from the user profile.

Merchants can fluctuate the price of their deals based on the customers’ response, including the number of customers that are following the deal and the number who actually claim the deal. Making decisions on deals based on real-time analytics allows merchants to host precise “happy hours” to effectively maximize revenue and balance inventory. They can also control the amount of customers entering their store, preventing overflow at a particular times when the deal provided is very good. Furthermore, Crowgle provides a more affordable solution of effective deal distribution by only taking a 10% commission.

TECHNICAL FEASIBILITY



Description

The Block Diagram describes all of the different components of the Crowgle Service. We start out with the users, the merchants and the customers, who are able to access Crowgle in two ways: through their mobile phone browser, or through a traditional browser on a desktop or laptop computer. The mobile phone interface to Crowgle is primarily used for posting deals, modifying deals, watching deals, or buying deals, while the traditional browser is can be used for everything in the mobile interface, plus viewing the analytical information over the deals a merchant has offered over time. The Crowgle service will be accessible from multiple mobile phone platforms because it will not be designed as a native application, but a website that will be accessed through a mobile browser; similarly, the Crowgle service can be accessed from any web browser (on a laptop or desktop computer) because it will come in the form of a website.

The next part of the block diagram is the backend Crowgle Server, which handles all of the requests from the merchants and customers, appropriately manages information about the deals and all users of the service, and makes connections to a variety of payment services when customers decide to to buy a deal. The server will be divided into multiple modules that will each encapsulate the functionality specific to that area of the Crowgle Service. Each of these modules will communicate back to the merchants and customers through the desired interface, and will persist important information (location, deals, profiles) into the appropriate database server that is running.

Use Case Examples

To further illustrate the block diagram, please go to the appendix to see a couple use cases show which parts of the block diagram are used for each step.

Development Timeline

Based on our use cases, we determined that it would take 62 working days days (or 3 months) to launch Crowgle. Below is our breakdown of how we reached this time estimate:

Time to develop Consumers Mobile Application:
User Interface for Search - 2 days
Watch a deal with visualization - 3 days
Mobile payment - 5 days
Maps integration - 1 day
Location querying - 2 days
Login - 1 day
Profile - 1 day
Add a Merchant to Favorites - 1 day
Total: 16 days

Time to develop Merchants Mobile Application:
UI to post deal - 2 days
Change deal - 1 day
Monitor deal - 3 days
Redeem a deal - 5 days
Login - 1 day
Profile setup - 2 days
Total: 14 days

Time to develop Server:
Database - 1 day
Location database - 2 days
APIs for mobile - 5 days
Analytics - 5 days
Machine Learning - 10 days
Total: 23 days

Time to develop Website
General Login - 1 day
Show analytics - 3 days
Total: 4 days

Alpha Testing: 5 days

Assumptions:
Each of the estimates include the testing of the individual components. The alpha testing phase is for the system as a whole.

Total days of work: **62 days** (3 months of development).

FINANCIAL VIABILITY

Costs

Based on our estimate of 3 months to develop the Crowgle Service, we determined that our start up costs will total **$53,100.** This figure includes cost for office space, servers (Google App Engine and Amazon Web Services), and legal fees.

Our average running costs total **$159,321 per month.** These costs include office space, server space, legal fees, and the salaries and benefits of our developers and sales people. Our development team will begin with 3 people, increasing to 5 by the end of the first year. We've estimated that each sales person can be responsible for 30 merchants. We will start with 2 sales people, increasing the sales team to 10 by the end of the year.

Revenue

Our revenue projections are based on the following formula:

(50 people purchase a deal on average) * ($15 per average transaction) *
(10% commission for Crowgle)
= **$75 per deal**

We've estimated that at running time, we will have 300 merchants posting an average of 10 deals a month, producing to the following monthly revenue:
(3000) * ($75) = **$225,000 per month**

Profitability

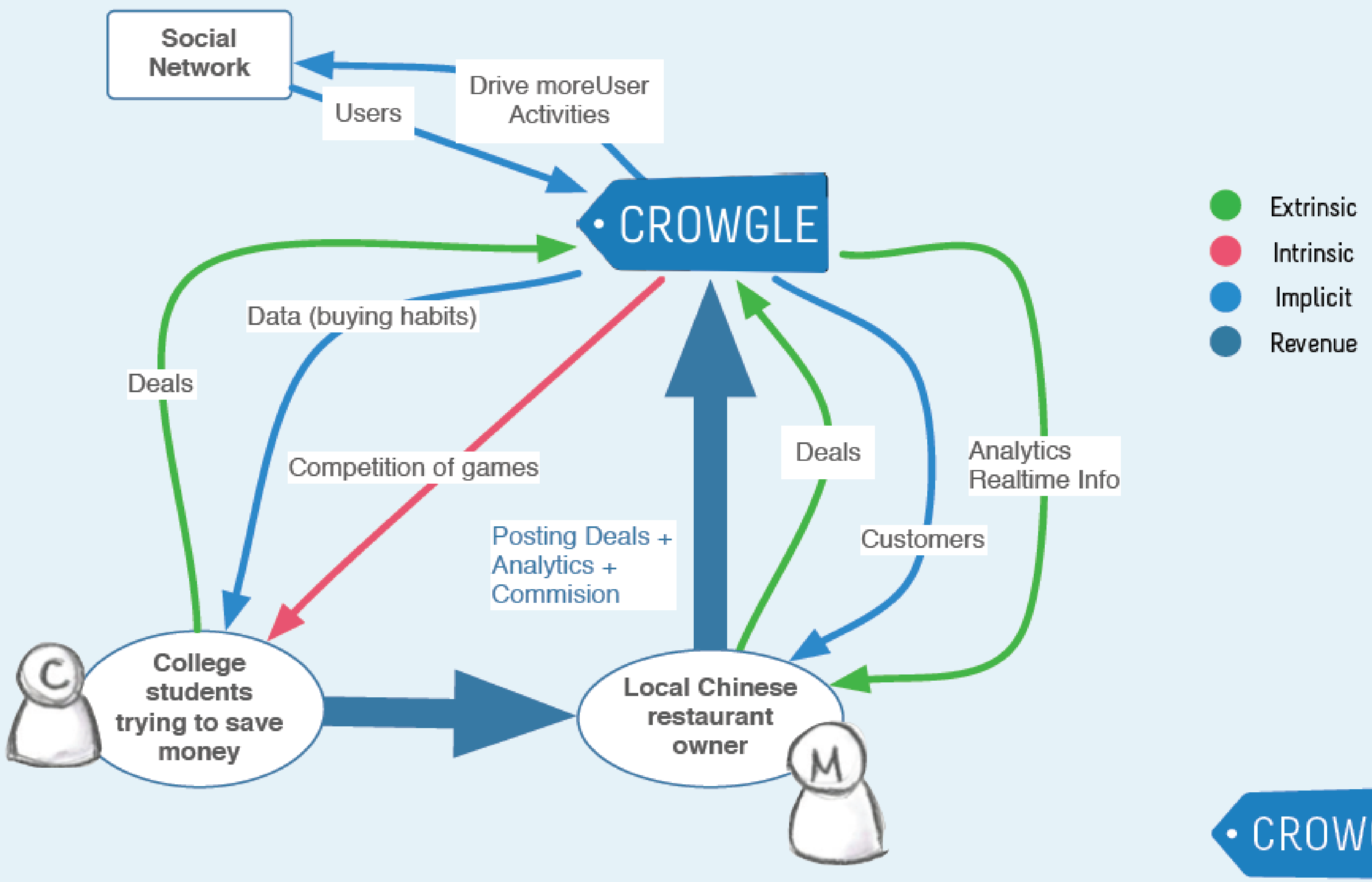
Based on the figures above, we expect to break even on our costs after the 6th month of operation. By the end of the first year, our profits should average **$1.8 million per year (per city).**

CONCLUSION

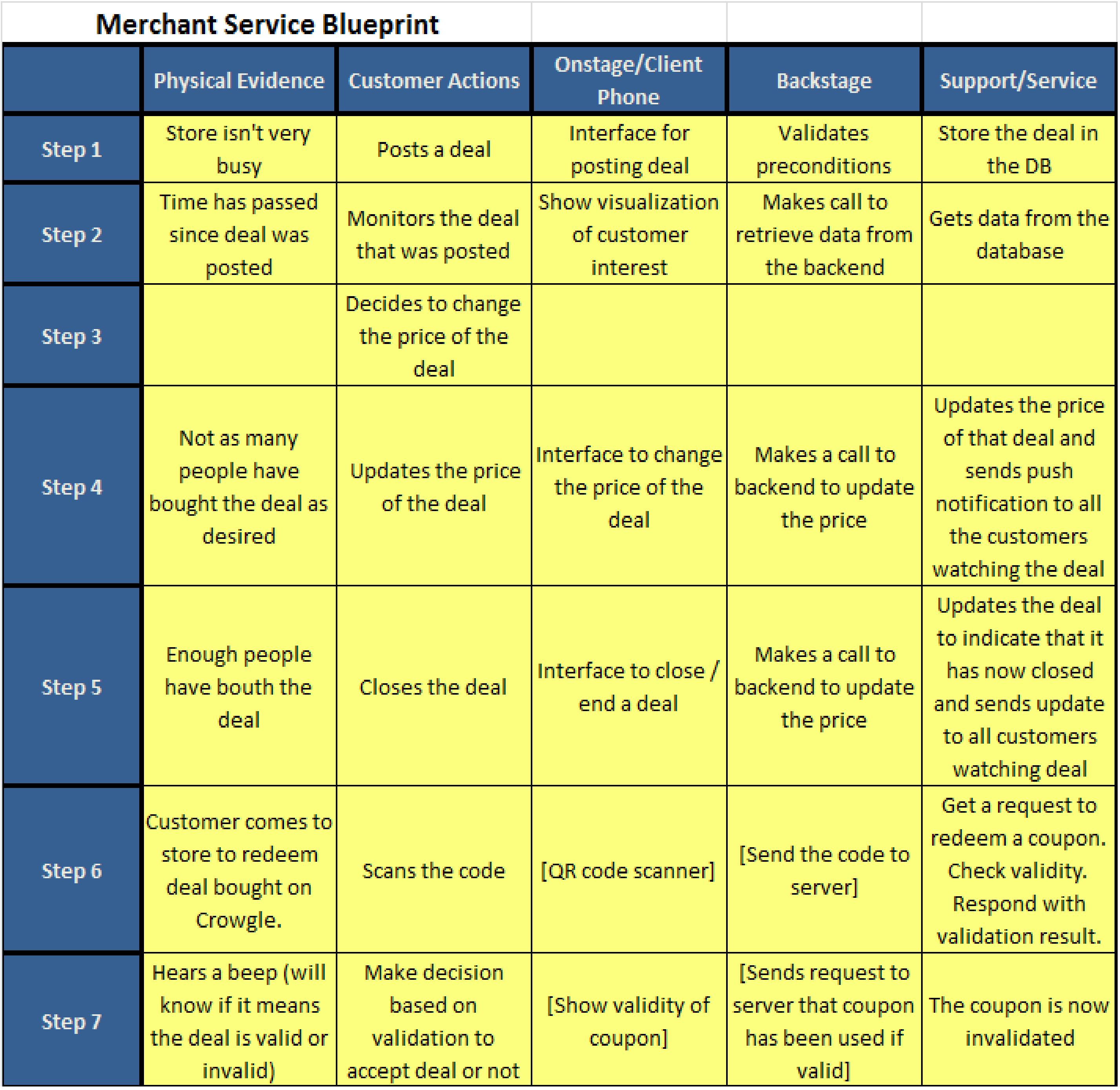
Crowgle brings to the table what isn’t available in the current market: an easy way for merchants to evaluate the deals they are providing; an innovative game layer that makes sense for the mobile coupon space. It is a solution that stands in the forefront leveraging current technological capabilities with a feasible revenue model that will be attractive to merchants. From our analysis, we believe that Crowgle has a strong potential for success in this competitive realm of deal distribution services.

APPENDIX

Value Flow Diagram



CROWGLE

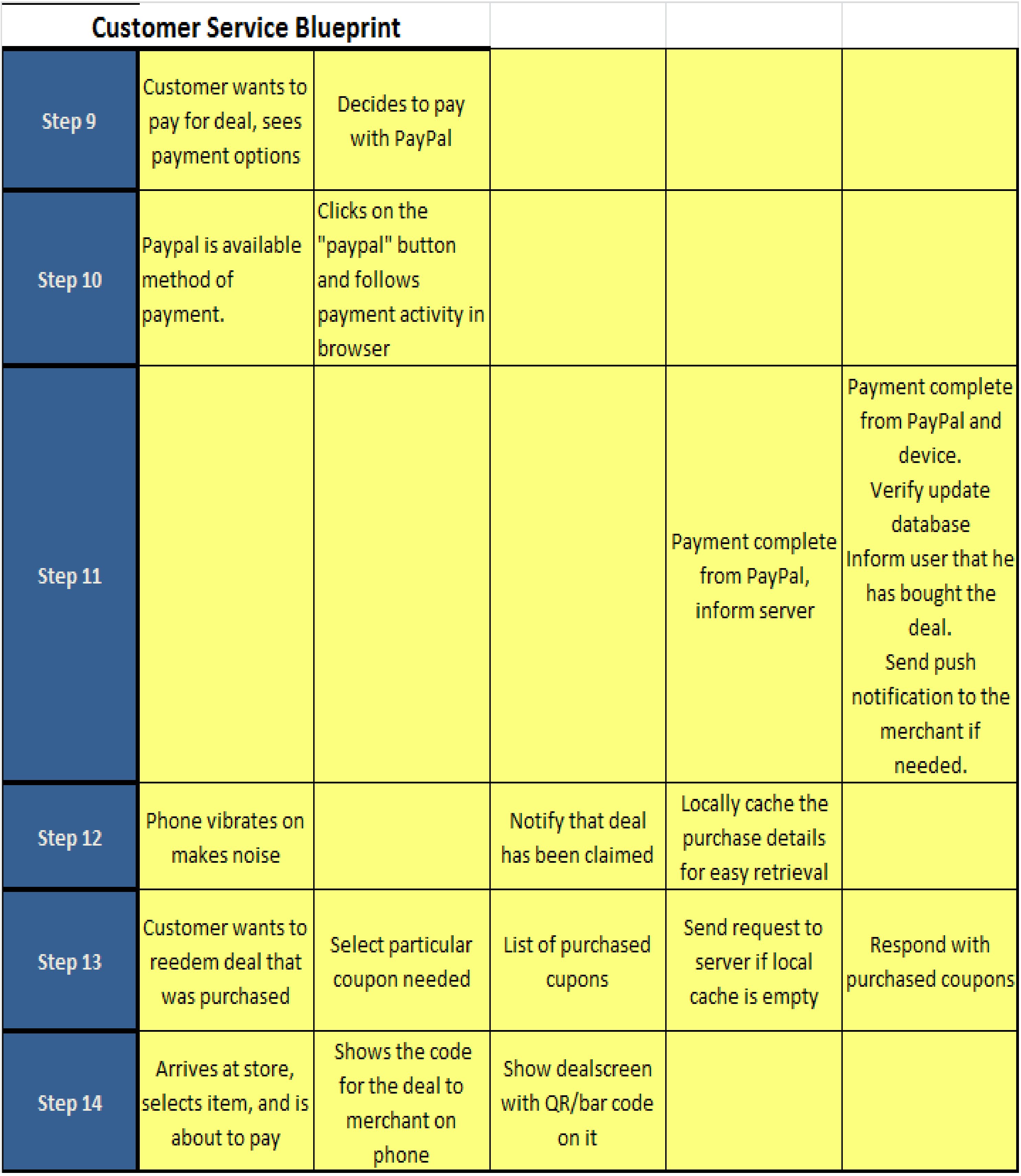


Merchant Service Blueprint

	Physical Evidence	Customer Actions	Onstage/Client Phone	Backstage	Support/Service
Step 1	Store isn't very busy	Posts a deal	Interface for posting deal	Validates preconditions	Store the deal in the DB
Step 2	Time has passed since deal was posted	Monitors the deal that was posted	Show visualization of customer interest	Makes call to retrieve data from the backend	Gets data from the database
Step 3		Decides to change the price of the deal			
Step 4	Not as many people have bought the deal as desired	Updates the price of the deal	Interface to change the price of the deal	Makes a call to backend to update the price	Updates the price of that deal and sends push notification to all the customers watching the deal
Step 5	Enough people have bouth the deal	Closes the deal	Interface to close / end a deal	Makes a call to backend to update the price	Updates the deal to indicate that it has now closed and sends update to all customers watching deal
Step 6	Customer comes to store to redeem deal bought on Crowgle.	Scans the code	[QR code scanner]	[Send the code to server]	Get a request to redeem a coupon. Check validity. Respond with validation result.
Step 7	Hears a beep (will know if it means the deal is valid or invalid)	Make decision based on validation to accept deal or not	[Show validity of coupon]	[Sends request to server that coupon has been used if valid]	The coupon is now invalidated

Customer Service Blueprint

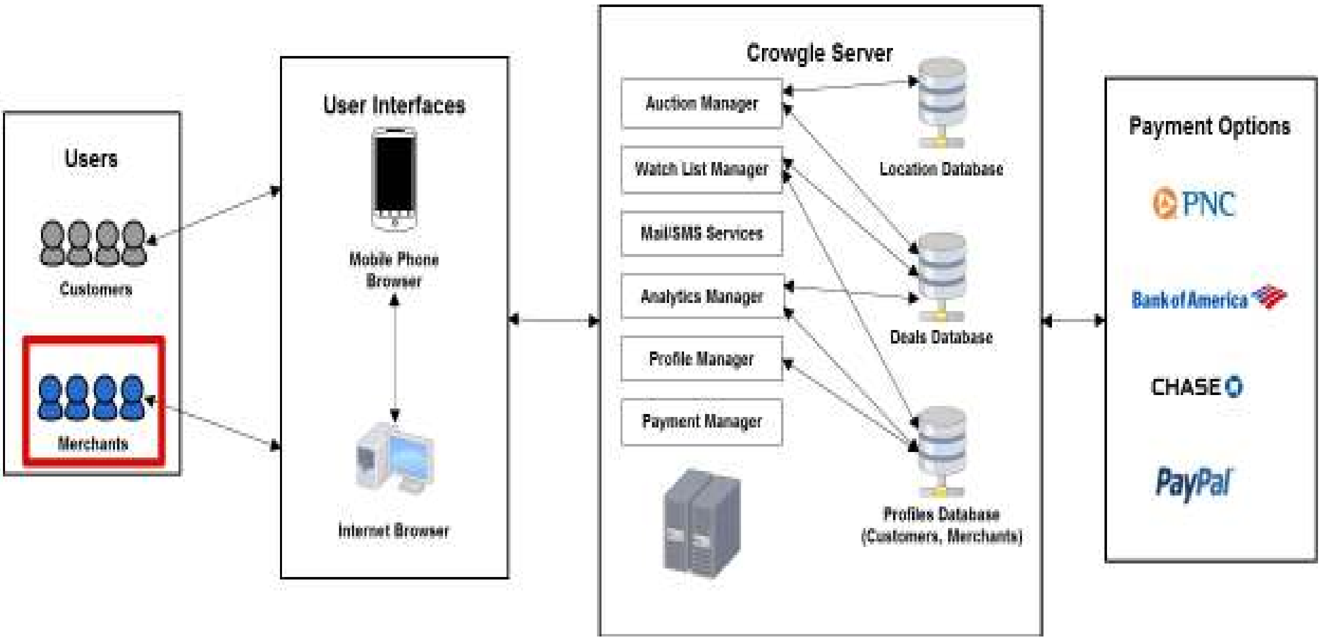
	Physical Evidence	Customer Actions	Onstage/Client Phone	Backstage	Support/Service
Step 1	Customer is interested in finding a deal	Decides to browse deals by location	Search UI	App queries current loc. & sends req to server along with loc.	Server finds results in DB based on query
Step 2			Shows results for query		
Step 3	App on the phone	Decides to view details		Send a request to server	Fetch visualization data, stats, and and send it back
Step 4			Display the details of the deal		
Step 5	Customer is interested in the current deal	Decides to "watch" & clicks "watch" button		Send a request to server	Server tracks all the people watching a particular deal
Step 6	Phone vibrates or makes noise		Shows push notification of price change on customer's phone		
Step 7	Customer sees notification on phone	Customer sees the price update of the deal	Show deal with the updated price		
Step 8	Decides to claim the deal after looking at updated details	Hits "Buy" button	Interface for deal	Send a buy request for this deal for this customer	Update deal to reflect that customer has tentatively bought this deal. Send notification to the app to process payment



Customer Service Blueprint					
Step 9	Customer wants to pay for deal, sees payment options	Decides to pay with PayPal			
Step 10	Paypal is available method of payment.	Clicks on the "paypal" button and follows payment activity in browser			
Step 11				Payment complete from PayPal, inform server	Payment complete from PayPal and device. Verify update database Inform user that he has bought the deal. Send push notification to the merchant if needed.
Step 12	Phone vibrates on makes noise		Notify that deal has been claimed	Locally cache the purchase details for easy retrieval	
Step 13	Customer wants to reedem deal that was purchased	Select particular coupon needed	List of purchased cupons	Send request to server if local cache is empty	Respond with purchased coupons
Step 14	Arrives at store, selects item, and is about to pay	Shows the code for the deal to merchant on phone	Show dealscreen with QR/bar code on it		

Use Case 1: Merchant posting a deal

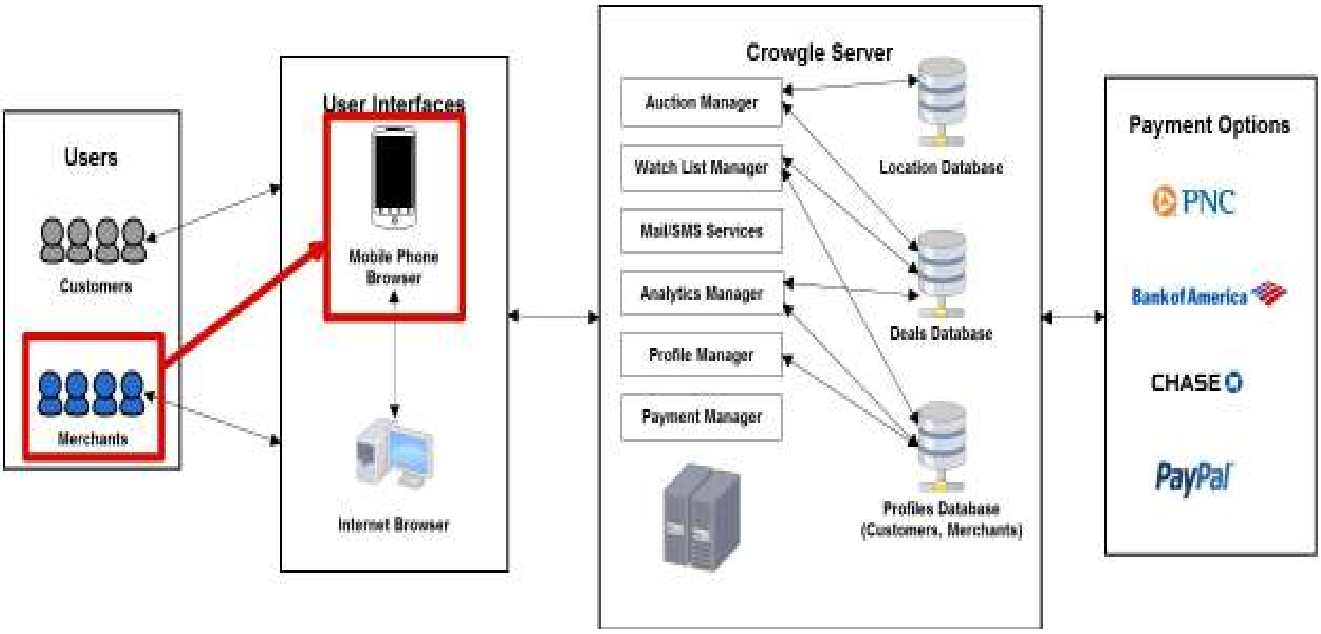
Step 1:The Merchant accesses the Crowgle service through a mobile phone



Use Case 1: Merchant posting a deal

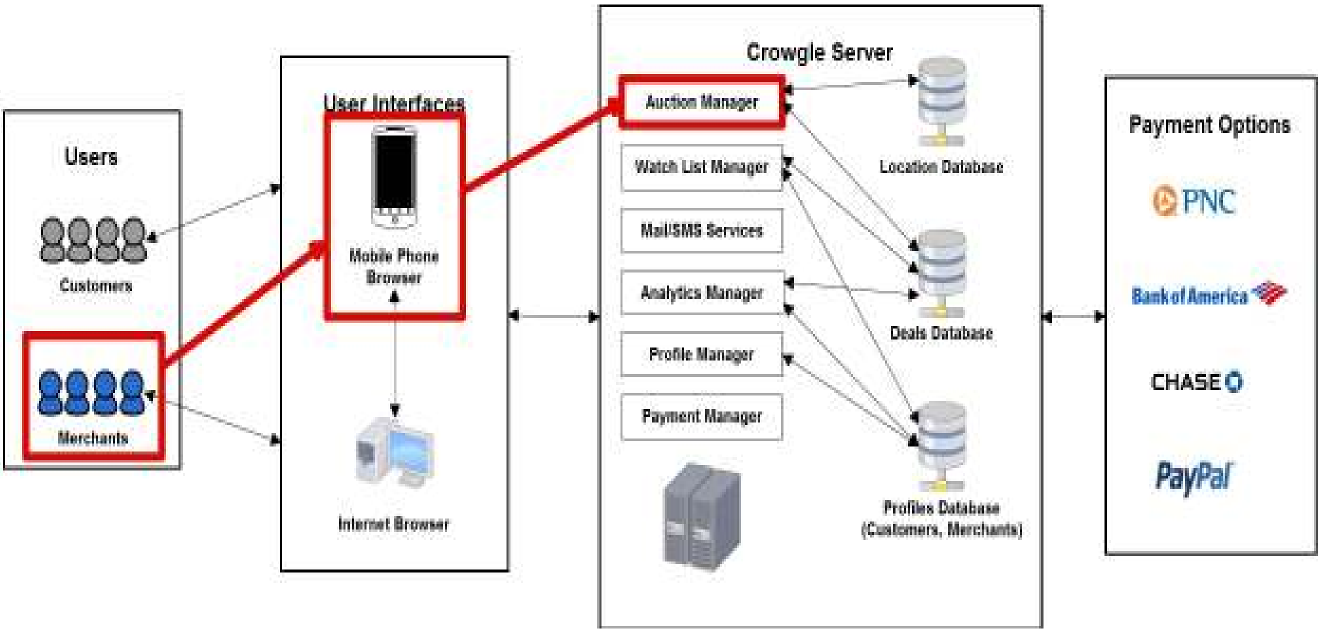
Step 2: The Merchant logs in, and bring up the screen to post a new deal

Step 3: The Merchant fills out the appropriate information, then presses the button to create this deal



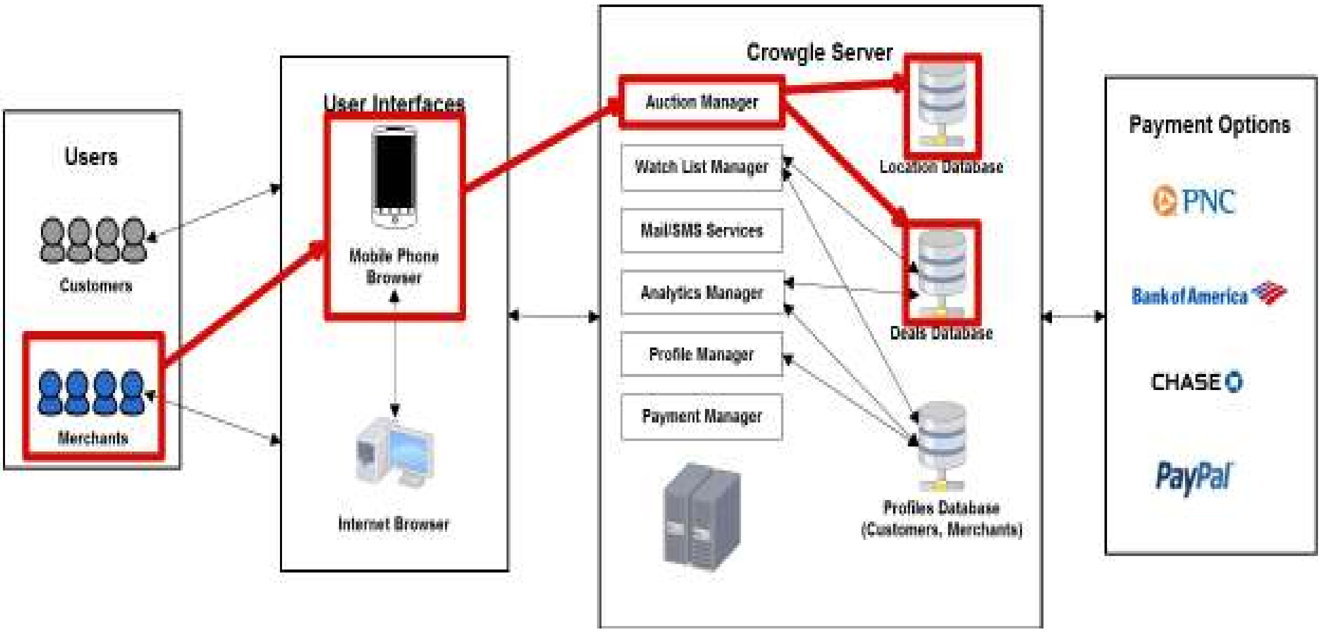
Use Case 1: Merchant posting a deal

Step 4: The request to create a new deal is sent to the Auction Manager. In this module, the potential deal is validated.



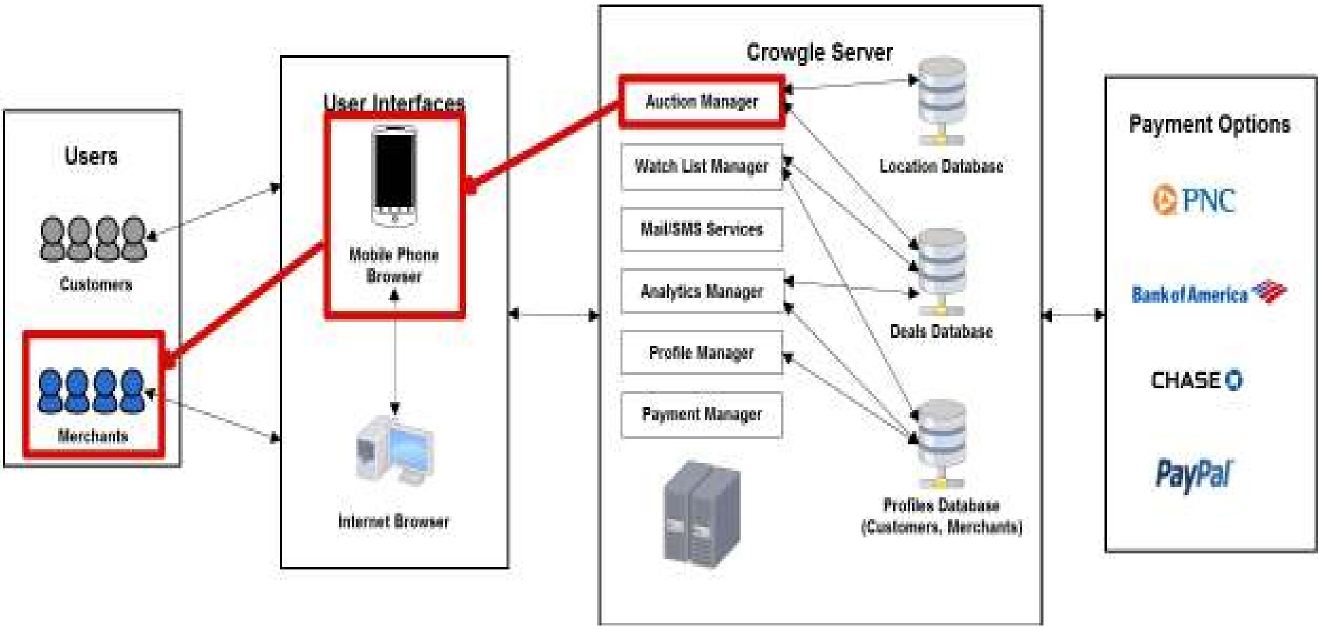
Use Case 1: Merchant posting a deal

Step 5: If everything is correct, a new deal will be created and will become active at the appropriate time (either now or a time in the future), and a copy of the deal and that it was created by a particular merchant will be stored in the appropriate database locations. If everything wasn't correct in the system, a corresponding deal will not be made.



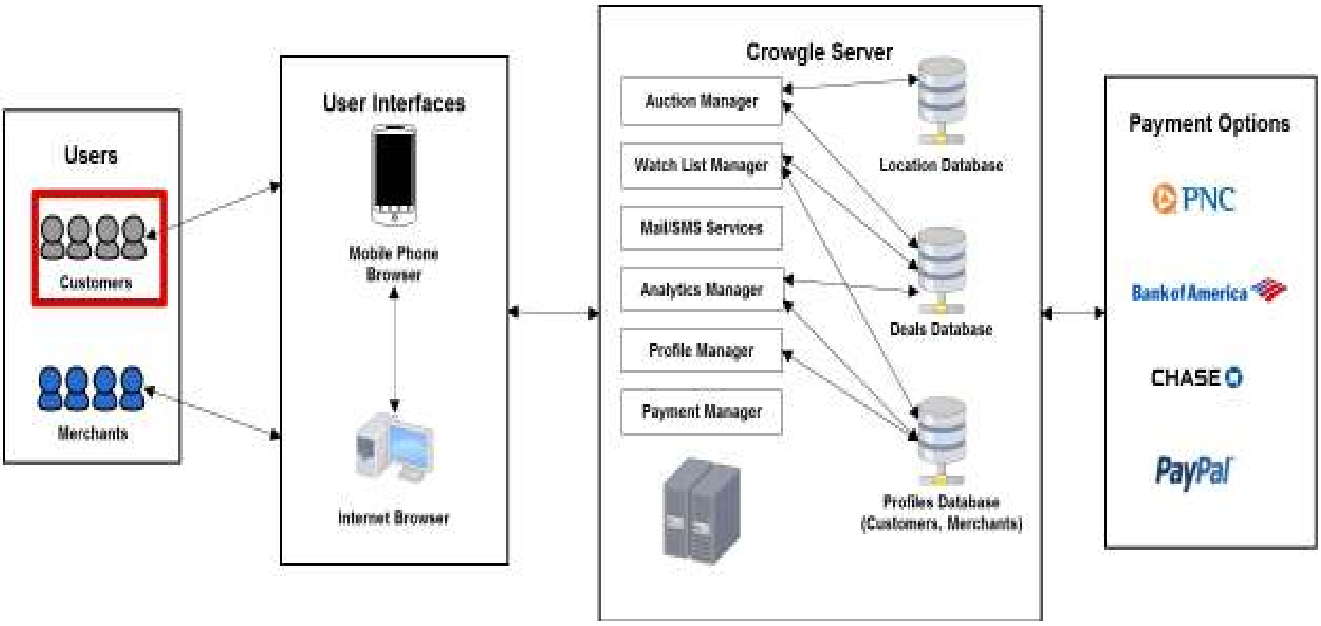
Use Case 1: Merchant posting a deal

Step 6: The Auction Manager will send an update back to the Mobile Browser to indicate whether the deal was successfully created or not.



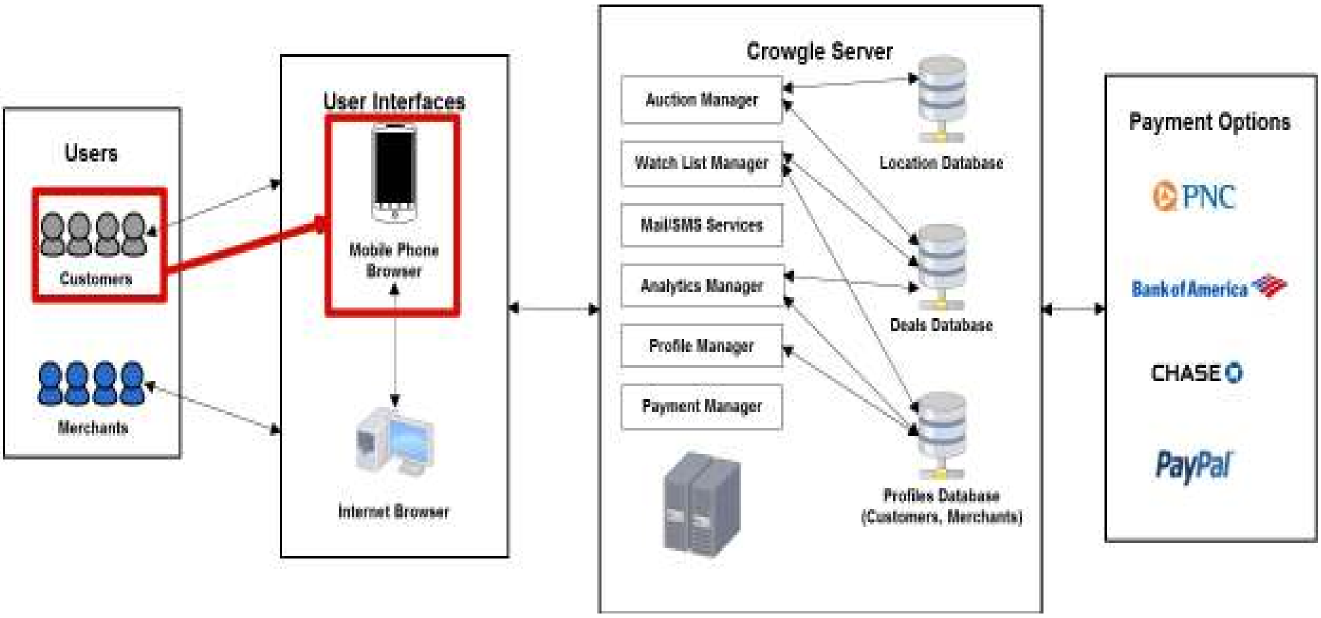
Use Case 2: Customer Watches a deal

Step 1: The Customer accesses the Crowgle service through a mobile phone



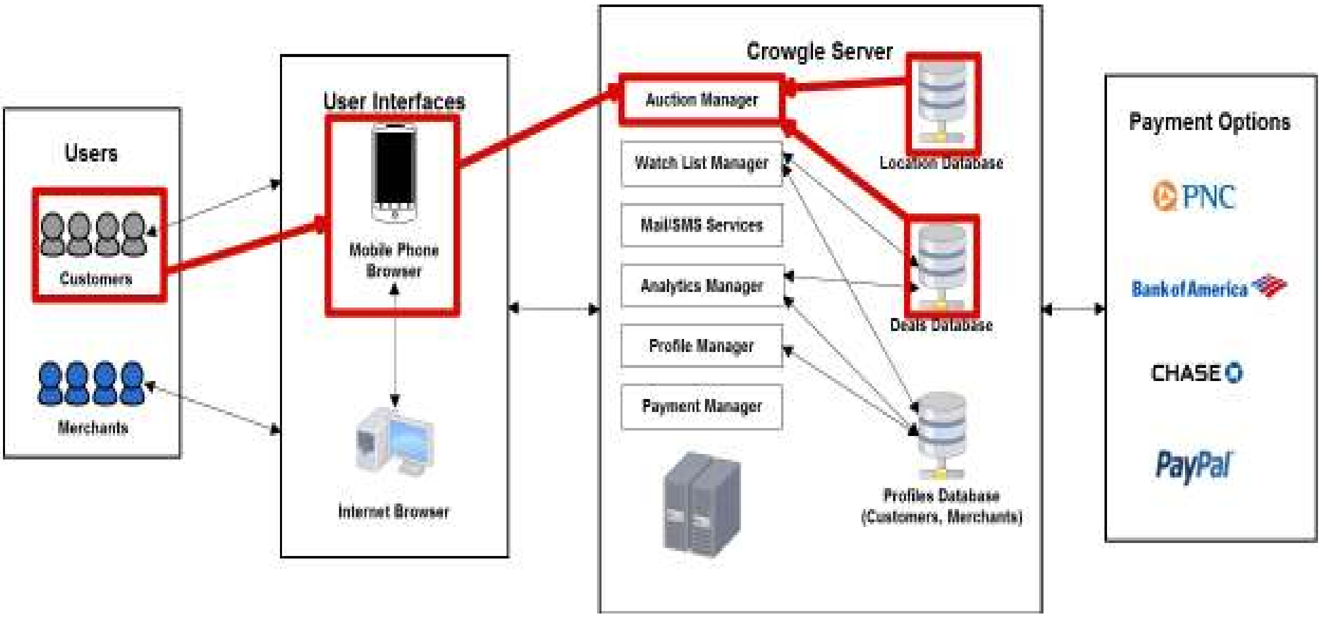
Use Case 2: Customer Watches a deal

Step 2: The Customer logs in and goes to the section of the application to see what deals are close to their current position.



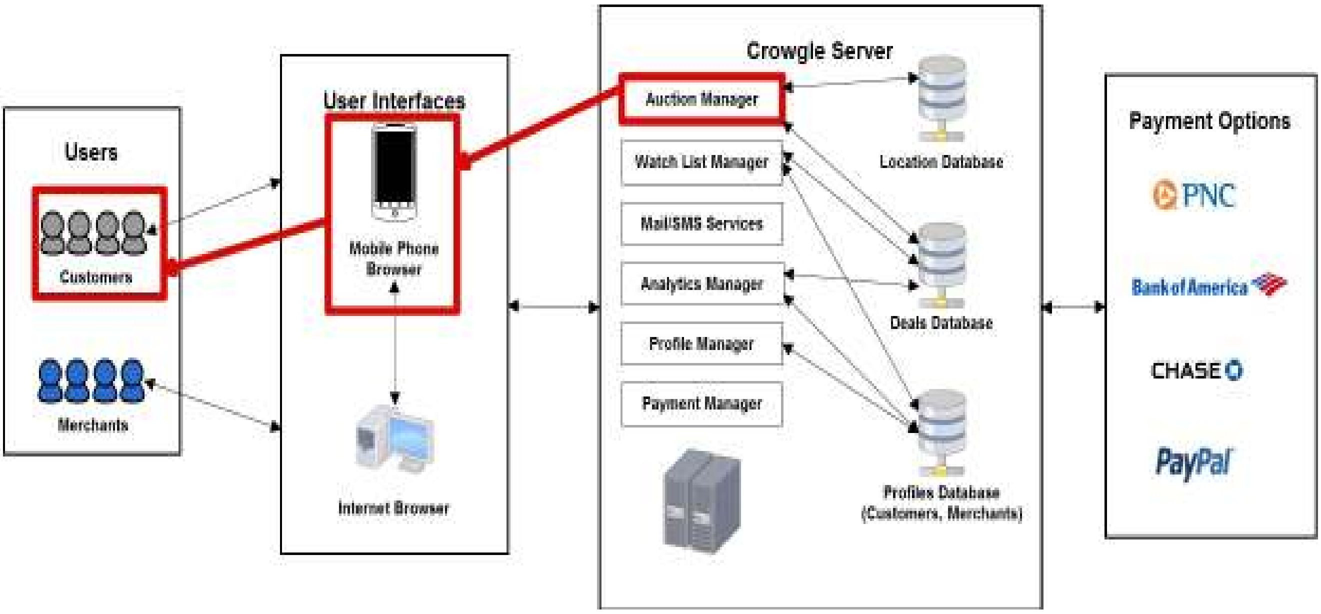
Use Case 2: Customer Watches a deal

Step 3: A request is sent to the Auction Manager to retrieve all of the deals that are close to a specific location. The Auction Manager retrieves this information through a combination of data collections and database requests.



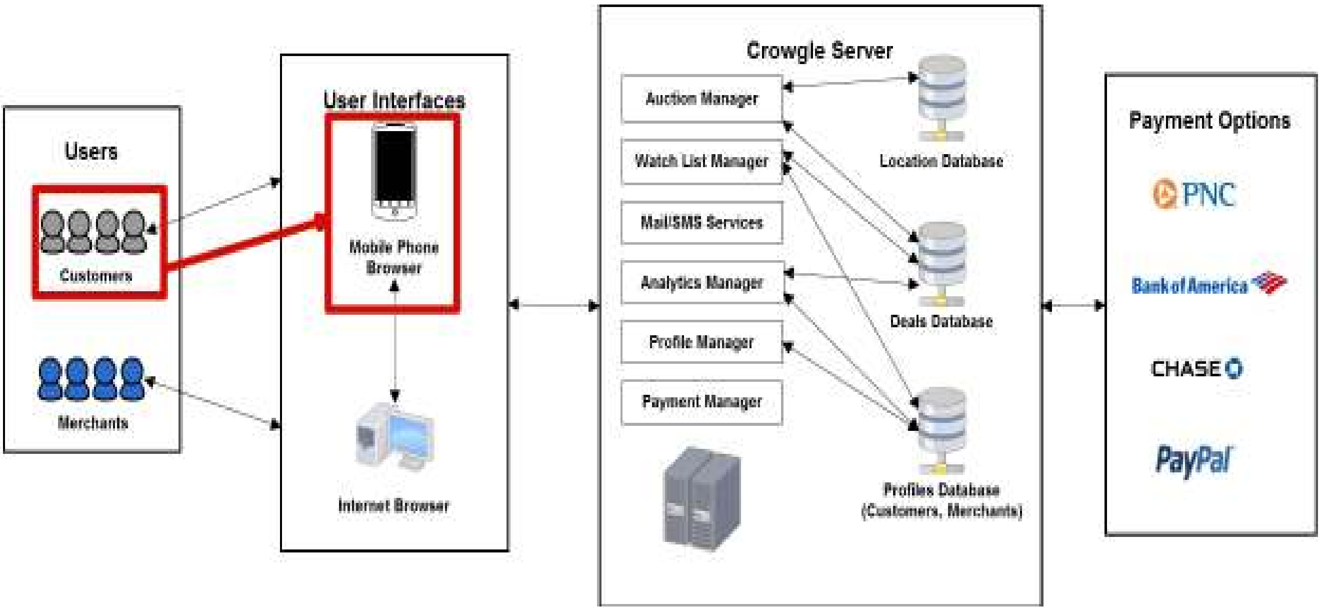
Use Case 2: Customer Watches a deal

Step 4: The Auction Manager sends the deals meeting this criteria back to mobile browser, and now the customer is able to see all of available deals close to them.



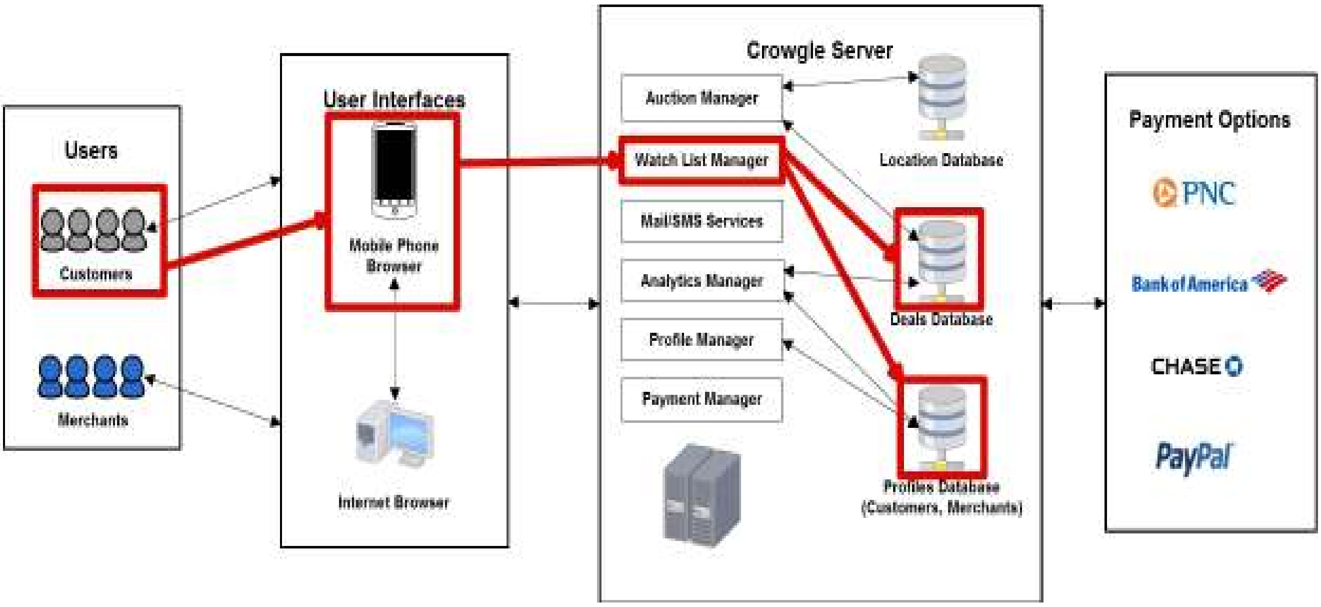
Use Case 2: Customer Watches a deal

Step 5: The customer selects a deal from the list, and chooses to watch it by pressing the appropriate button.



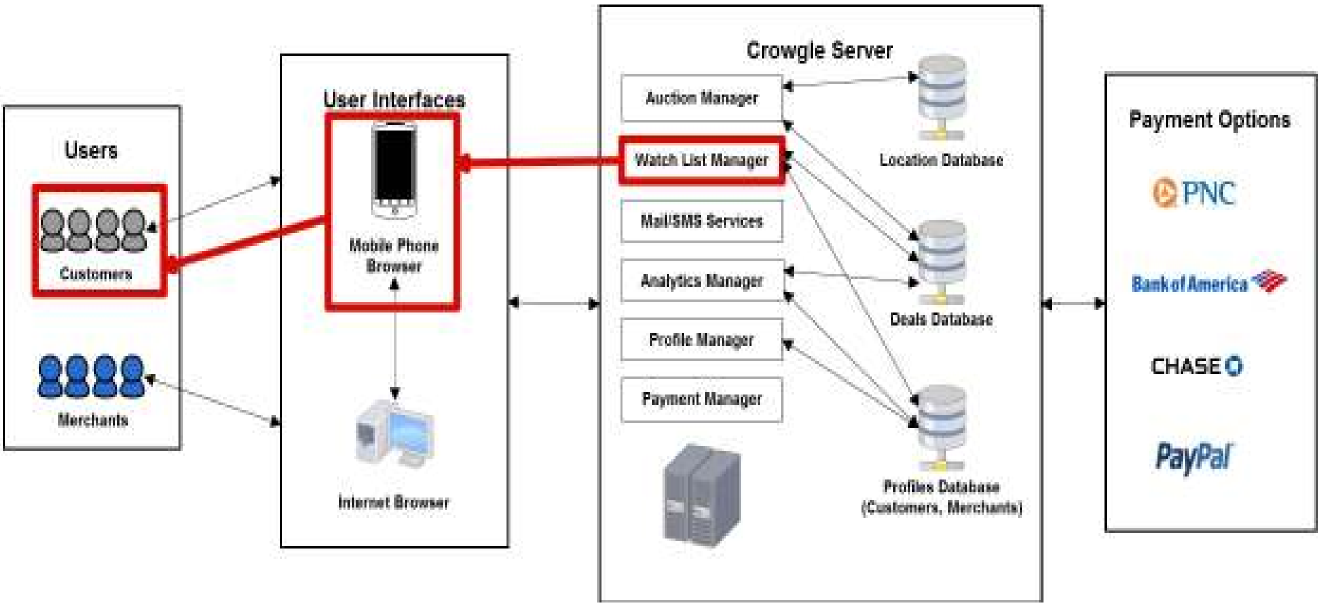
Use Case 2: Customer Watches a deal

Step 6: A Request is sent to the Watch List Manager, which configures this particular customer to be updated (according to their criteria) when certain aspects of the deal change. This information gets updated in the data collection, and is persisted into the Deals and Profiles databases.



Use Case 2: Customer Watches a deal

Step 7: The Watch List Manager sends an update to the mobile browser that the Customer is now watching the deal, and will get updated when changes are made.



Consumer Use Cases

Browse deals (by location, category, favorite merchants)

View a deal

    show number of people "watching"

    number of people "purchased"

    price fluctuation so far

Watch a deal

    receive push notification on change

Un-watch a deal

Purchase a deal

Save a merchant to favorites

View redeemed deals (history)

Use Case - C1 - Browse Deals by location

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login. There is a data connection and GPS available with the mobile.

Success Guarantee: The system shows deals based on consumer's request; shows a message to say there are no deals, in case there are no deals available.

Main Success Scenario:

1. The consumer navigates to the search screen.
2. The application shows the different search options: a. location, b. categories, c. favorite merchant
3. The consumer selects the option "location."
4. The application queries for the consumer's location using the device's GPS.
5. The application shows the user about the location found for verification.
6. The user confirms the location.
7. The application sends the request to the server.
8. The application shows the list of deals available to the user.

Extensions:

4a. The location information is unavailable

    4a1. The application retries to find the user's location.

    4a2. The application informs the user about the status of the location, found or unable to find.

6a. The location information found is wrong.

    6a1. The application queries for the location again.

    6a2. The application shows the new location.

7a. The application is not able to contact the server.

7a1. The application retries contacting the server.

7a2. The application reports error if it cannot contact the server after X retries.

8a. There are no deals available nearby.

8a1. The application reports that there are no deals available nearby.

Use Case - C2 - Browse Deals by category

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login.

Success Guarantee: The system shows deals based on consumer's request; shows a message to say there are no deals, in case there are no deals available.

Main Success Scenario:

1. The consumer navigates to the search screen.
2. The application shows the different search options: a. location, b. categories, c. favorite merchant
3. The consumer selects the option "category."
4. The application shows the user the list of available categories.
5. The user selects a category.
6. The application sends the request to the server.
7. The application shows the list of deals available to the user.

Extensions:

6a. The application is not able to contact the server.

6a1. The application retries contacting the server.

6a2. The application reports error if it cannot contact the server after X retries.

7a. There are no deals available nearby.

7a1. The application reports that there are no deals available nearby.

Use Case - C3 - Browse Deals by favorite merchants

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login. The customer has at least one merchant marked as a favorite

Success Guarantee: The system shows deals based on consumer's request; shows a message to say there are no deals, in case there are no deals available.

Main Success Scenario:

1. The consumer navigates to the search screen.
2. The application shows the different search options: a. location, b. categories, c. favorite merchant

3. The consumer selects the option “favorite merchants.”
4. The applications sends a request to the server to get the list of favorite merchants.
5. The application shows the list of favorite merchants of the user.
6. The user selects a merchant from the list.
7. The application sends the request to the server.
8. The application shows the list of deals available to the user.

Extensions:

4a. The application is not able to contact the server.

4a1. The application retries contacting the server.

4a2. The application reports error if it cannot contact the server after X retries.

5a. The user has no merchants listed as a favorite.

5a1. The application shows an error message conveying that there are no favorite merchants for the user.

7a. The application is not able to contact the server.

7a1. The application retries contacting the server.

7a2. The application report error if it cannot contact the server after X retries.

8a. There are no deals available nearby.

8a1. The application reports that there are no deals available nearby.

Use Case - C4 - View a particular deal in detail

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used atleast once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login. The consumer is shown a list of deals as a result of either of the use cases C1, C2 or C3.

Success Guarantee: The system shows a detailed view of a particular deal.

Main Success Scenario:

1. The user selects one of the deals listed to him as a result of use cases C1, C2 or C3.
2. The system sends a request to the server to get details about the deal.
3. The system shows the user the details of the deal.

Extensions:

2a. The application is not able to contact the server.

2a1. The application retries contacting the server.

2a2. The application reports error if it cannot contact the server after X retries.

Use Case - C5 - Watch a deal

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used atleast once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login. The consumer is shown a detailed view of a particular deal.

Success Guarantee: The system acknowledges the user for watching the deal.

Main Success Scenario:

1. The user selects the option to "watch" a deal.
2. The system sends a request to the server.
3. The server responds back a success message.
4. The system notifies the user on successfully allowing him to watch a deal.

Extensions:

2a. The application is not able to contact the server.

2a1. The application retries contacting the server.

2a2. the application reports error if it cannot contact the server after X retries.

3a. The server responds back with a negative message due to concurrency issues (all remaining deals are claimed before the user's request reaches the server).

3a1. The system notifies the user with an appropriate negative message.

Use Case - C6 - User selects a deal to be notified after X people buys it

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used at least once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login. The user has successfully watched a deal.

Success Guarantee: The system notifies the user of a successful push notification setup.

Main Success Scenario:

1. The user has successfully watched the deal as a result of use case C5.
2. The user selects "notify" option.
3. The system provides the user with different choices for push notification.
4. The user selects "buy" and enters the number X.
5. The system sends a request to the server.
6. The server responds with a success message after adding required entries to the database.
7. The system notifies the user about successfully enabling the push notification option.

Extensions:

3a. The device does not have push notification capabilities.

3a1. The system disables "notify" option in step 2.

3a2. The system informs the user about the lack of push notification capabilities.

3a3. The system might show other notification options.

5a. The application is not able to contact the server.

5a1. The application retries contacting the server.

5a2. the application reports error if it cannot contact the server after X retries.

6a. The server responds with a negative message due to database issues.

6a1. The system notifies the user with an error message.

Use Case - C7 - User selects a deal to be notified after when price changes

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used at least once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login. The user has successfully watched a deal.

Success Guarantee: The system notifies the user of a successful push notification setup.

Main Success Scenario:

1. The user has successfully watched the deal as a result of use case C5.
2. The user selects “notify” option.
3. The system provides the user with different choices for push notification.
4. The user selects “price”.
5. The system sends a request to the server.
6. The server responds with a success message after adding required entries to the database.
7. The system notifies the user about successfully enabling the push notification option.

Extensions:

3a. The device does not have push notification capabilities.

3a1. The system disables “notify” option in step 2.

3a2. The system informs the user about the lack of push notification capabilities.

3a3. The system might show other notification options.

5a. The application is not able to contact the server.

5a1. The application retries contacting the server.

5a2. the application reports error if it cannot contact the server after X retries.

6a. The server responds with a negative message due to database issues.

6a1. The system notifies the user with an error message.

Use Case - C8 - User receives a push notification on change in price or number of people buying it

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used at least once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login. The user is watching a deal and has setup push notifications for changes with respect to the deal.

Success Guarantee: The system notifies the user with a push notification and takes the user to the deal’s detailed view.

Main Success Scenario:

1. The system shows a push notification to the user.
2. The user taps on the notification.
3. The system sends a request to the server to get more details about the deal.
4. The server responds back with the details of the deal.
5. The system shows the details of the deal to the user.

Extensions:

3a. The application is not able to contact the server.

3a1. The application retries contacting the server.

3a2. the application reports error if it cannot contact the server after X retries.

Use Case - C9 - Un-watch a deal

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used at least once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login. The user is currently watching a deal.

Success Guarantee: The system successfully un-watches a deal for the user.

Main Success Scenario:

1. The user is on the detailed view of a deal that he is currently watching.
2. The user selects “Un-watch”.
3. The system sends a request to the server.
4. The server responds with a success message for removing the user from the watch list of the deal.
5. The system notifies the user about the successful un-watching of the deal.

Extensions:

3a. The application is not able to contact the server.

3a1. The application retries contacting the server.

3a2. the application reports error if it cannot contact the server after X retries.

4a. The server responds with a negative message due to database issues.

4a1. The application informs the user about the issue at the server side.

4a2. The application asks the user to retry after some time based on the issue.

Use Case - C10 - User purchases a deal

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used at least once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login. The user decides to purchase a deal.

Success Guarantee: The system informs the user about the successful purchase of the deal.

Main Success Scenario:

1. The user clicks on “buy” button on a certain deal’s detailed view.
2. The system shows a popup message to confirm his action.
3. The user selects “Yes”.
4. The system sends a request to the server to temporarily lock a deal.
5. The server responds back with a success message.
6. The system transfers the user to PayPal’s mobile payment.
7. The system is notified about the successful completion of the payment.
8. The system sends the request to the server to confirm the payment.
9. The server responds back with a success message.
10. The system notifies the user with a success message in the form of a purchased coupon.

Extensions:

4a. The application is not able to contact the server.

4a1. The application retries contacting the server.

4a2. the application reports error if it cannot contact the server after X retries.

5a. The server responds with a negative message.

5a1. The application informs the user with the error.

7a. The application is not notified by the successful completion of payment.

7a1. The application contacts the PayPal server after some timeout.

7a2. PayPal responds back with the status of the payment. The application takes appropriate steps.

8a. The application is not able to contact the server.

8a1. The application retries contacting the server.

8a2. The application reports error if it cannot contact the server after X retries.

8a3. The application informs the user to check back after some time.

9a. The server responds back with a negative message.

9a1. The application retries again after sometime.

9a2. The application informs the user about the issue, if the server has not got a confirmation from the PayPal.

Use Case - C11 - Add a merchant as a favorite

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used at least once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login. The user is currently viewing a deal.

Success Guarantee: The system informs the user on successfully adding a merchant as a favorite.

Main Success Scenario:

1. The user clicks on the name of the merchant.
2. The system displays details about the merchant, including location.
3. The user selects “Add as Favorite”.

4. The system sends a request to the server.
5. The server responds with a success message.
6. The system informs the user about the successful adding of the merchant as a favorite.

Extensions:

4a. The application is not able to contact the server.

4a1. The application retries contacting the server.

4a2. the application reports error if it cannot contact the server after X retries.

5a. The server responds with a negative message.

5a1. The application informs the user about the error.

Use Case - C12 - Remove a merchant as a favorite

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used at least once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login. The user is currently viewing a deal. The merchant is already a favorite for the user.

Success Guarantee: The system informs the user on successfully removing a merchant as a favorite.

Main Success Scenario:

1. The user clicks on the name of the merchant.
2. The system displays details about the merchant, including location.
3. The user selects “Remove as Favorite”.
4. The system sends a request to the server.
5. The server responds with a success message.
6. The system informs the user about the successful adding of the merchant as a favorite.

Extensions:

4a. The application is not able to contact the server.

4a1. The application retries contacting the server.

4a2. the application reports error if it cannot contact the server after X retries.

5a. The server responds with a negative message.

5a1. The application informs the user about the error.

Use Case - C13 - View purchased deals

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used at least once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login.

Success Guarantee: The system shows a list of purchased deals that are yet to be redeemed.

Main Success Scenario:

1. The user navigates to the deals screen.
2. The user selects the “Purchased deals” option.
3. The system sends a request to the server to get the purchased deals.
4. The server responds with a list of purchased deals.
5. The system displays the list of deals to the user.

Extensions:

3a. The application is not able to contact the server.

 3a1. The application retries contacting the server.

 3a2. the application reports error if it cannot contact the server after X retries.

4a. The server responds with an error due to database issues.

 4a1. The application informs the user about the error.

4b. The user has no deals purchased so far.

 4b1. The server responds with a success message having no deals.

 4b2. The application displays a message saying there are no deals purchased by the user.

Use Case - C14 - View redeemed deals

Primary Actor: Consumer

Scope: Mobile Application

Level: User goal

Preconditions: The application is installed and has been used at least once.

Minimal Guarantee: The consumer has an account with the system and the system provides enough help to login.

Success Guarantee: The system shows a list of redeemed deals of the user.

Main Success Scenario:

1. The user navigates to the deals screen.
2. The user selects the “Redeemed deals” option.
3. The system sends a request to the server to get the purchased deals.
4. The server responds with a list of redeemed deals.
5. The system displays the list of deals to the user.

Extensions:

3a. The application is not able to contact the server.

 3a1. The application retries contacting the server.

 3a2. the application reports error if it cannot contact the server after X retries.

4a. The server responds with an error due to database issues.

 4a1. The application informs the user about the error.

4b. The user has no deals redeemed so far.

 4b1. The server responds with a success message having no deals.

 4b2. The application displays a message saying there are no deals redeemed by the user.

Merchant Use Cases

Post Deal

Monitor Deals

- Provide advice/tips based on history
- Include a visualization
- Push Notifications (customize and/or turn off)

Modify Existing Deals

Cancel deal

Redeem deal for customers

View Analytics

- Deal “sweet spot”
- Best performing deals (option to rerun)
- Performance of deals by time/place (if mult. locations)

Use Case - M1 - Post Deals

Primary Actor: Merchant

Scope: Mobile Application

Level: User goal

Preconditions: User has already generated a deal scheme

Minimal Guarantee: Previous deals, profile remains intact

Success Guarantee: Deals made public for the consumers

Main Success Scenario:

1. Prompts input of merchandise name
2. User inputs merchandise name
3. Application prompts for input on type, starting amount, system monitors inputs
4. User inputs deal details

5.Application prompts for a date, warns user if number isn’t reasonable

6.User inputs dates

7. User submits the deal to the system

Extensions:

7a. Fail to enter mandatory field

7b. Application reports failure to the user, brings user back to the input interface with missing field highlighted

Use Case - M2 - Monitor Deals

Primary Actor: Merchant

Scope: Mobile Application

Level: User goal

Preconditions: Have an ongoing deal posted

Minimal Guarantee: Previous deal settings not affected

Success Guarantee: Stats update in real-time
Main Success Scenario:
1. User clicks to monitor a particular deal
2. Application brings up the monitor interface, queries data from the server
3. User browse through different statistic
4. User exits the monitor interface when finished
Extensions:
4a. User decides to change a deal and clicks the modify deal button
4b. Triggers Use Case M3

Use Case - M3 - Modify Existing Deals

Primary Actor: Merchant
Scope: Mobile Application
Level: User goal
Preconditions: Have an ongoing deal posted.
Minimal Guarantee: Previous deal settings not affected.
Success Guarantee: Updates the price of that deal and the system sends push notification to all the customers watching the deal.
Main Success Scenario:
1. Opens Crowgle Merchant app on phone.
2. Clicks on a deal to view details from the deals list view.
3. Click the Adjust Deal button on the deal monitor interface.
4. Application directs user to the deal adjustment interface.
5. User uses the slider interface to modify deal amount, hits OK
6. Application directs user to confirmation interface with highlight (color) indication for modified field.
7. User hits Confirm button to change deal amount.
8. Application contacts server to send push notification of deal change to customers following that deal.
9. Application directs user back to the deals list view and briefly highlights the deal just modified.

Extensions
1a. Logs in with user's merchant profile on Crowgle website
3a. Decides not to change deal status and hits the Back button
3b. Decides to cancel the deal and remove it entirely, trigger M3 Cancel Existing Deal use case

Use Case - M4 - Cancel Existing Deals

Primary Actor: Merchant
Scope: Mobile Application
Level: User goal
Preconditions: Have an ongoing deal posted
Minimal Guarantee: Previous deal settings not affected.
Success Guarantee: Updates the price of that deal and the system sends push notification to all the

customers watching the deal.

Main Success Scenario:

1. Opens Crowgle Merchant app on phone.
2. Clicks on a deal to view details from the deals list view.
3. Click the Cancel Deal button on the deal monitor interface.
4. Application directs user to the canceling confirmation interface.
5. User clicks the Confirm button to remove the deal.
6. Application contacts server to move the deal to archive.
7. Server sends push notification to customers that are following the deal.

Extensions

1a. Logs in with user's merchant profile on Crowgle website

3a. Decides not to change deal status and hits the Back button

3b. Decides to just modify the deal amount, trigger M2 Modify Existing Deal use case

4a. User decides not to cancel the deal, hits back button.

4a1. Application directs user back to deal details page.

Use Case - M5 - Redeem deal for customer

Primary Actor: Merchant

Scope: Mobile Application, Physical Storefront

Level: User goal

Preconditions: Customers show up with a digital or physical copy of deal purchase receipts.

Minimal Guarantee: QR code on receipt invalid, redemption failed. Inventory not effected.

Success Guarantee: Redemption successful, applications notifies server to update deals redeemed.

Main Success Scenario:

1. Customer presents their Crowgle application on his/her smartphone with the Deal Confirmation receipt.
2. User opens up Crowgle Merchant application and press the Scanner from the menu.
3. Application brings up the scanner interface, accessing the camera from the smartphone.
4. User hovers smartphone over the QR code on the receipt presented.
5. Application detects QR code, sends request to server to query match, display matching
6. Server matches the incoming information to the data of deals sold for that particular merchant.
7. Server finds a match, update that as redeemed, sends success notification back to the application.
8. User sees success message, hands customer the merchandise to complete the transaction.
9. Application updates the redeemed deal count on the deal details page.

Extensions

1a. Customer presents a physical printout of the receipt with the QR code on it.

7a. Server couldn't match incoming information with ones in the database, sends failure notification back to the application.

Use Case - M6 - View Analytics Sweet Spot

Primary Actor: Merchant

Scope: Mobile Application

Level: User goal

Preconditions: Ongoing deals that are previously posted by the user.

Minimal Guarantee: Analytics taking too long to load, everything is not effected.

Success Guarantee: User quickly views Sweet Spot and makes necessary judgements

Main Success Scenario:

1. User clicks on the Analytics tab.
2. Application sends request to server for data in the user's deal archive.
3. Application brings up the Analytics interface and displays general Analytics.
4. User clicks on the Sweet Spot button on the Analytics interface.
5. Application calculates and defines the Sweet Spots and plots it on the chart.
6. User clicks on one particular Sweet Spot.
7. Dialogue box pops up showing the particulars of that Sweet Spot.

Extensions

4a. User clicks on Best Performing Deals, triggers M7 - View Analytics Best Performing Deals.

4b. User clicks on Performance of Deals, triggers M8 - View Analytics Deal Performance.

Use Case - M7 - View Analytics Best Performing Deals

Primary Actor: Merchant

Scope: Mobile Application

Level: User goal

Preconditions: Ongoing deals that are previously posted by the user.

Minimal Guarantee: Analytics taking too long to load, everything is not effected.

Success Guarantee: Application quickly loads and display the ranking of deals that attracted the most customers.

Main Success Scenario:

1. User clicks on the Analytics tab.
2. Application sends request to server for data in the user's deal archive.
3. Application brings up the Analytics interface and displays general Analytics.
4. User clicks on the Best Deals button on the Analytics interface.
5. Application sorts the archived deals and displays the top 5 (most # of customer claimed)
6. User clicks on one particular deal.
7. Application sends request to server for deal detail history for that particular deal.
8. Directs the user to deal details interface with the analytics information.

Extensions

4a. User clicks on Sweet Spots, triggers M6 - View Analytics Sweet Spot.

4b. User clicks on Performance of Deals, triggers M8 - View Analytics Deal Performance.

5a. User clicks on "Next 5" on the list.

> 5a1. Application list out the next 5 deals in the ranking (now 10)

Use Case - M8 - View Analytics Deal Performance

Primary Actor: Merchant
Scope: Mobile Application
Level: User goal
Preconditions: Ongoing deals that are previously posted by the user.
Minimal Guarantee: Analytics taking too long to load, everything is not effected.
Success Guarantee: Effectively queries and displays the visualization of prior deals according the location, time.

Main Success Scenario:

1. User clicks on the Analytics tab.
2. Application sends request to server for data in the user's deal archive.
3. Application brings up the Analytics interface and displays general Analytics.
4. User clicks on the Performance button on the Analytics interface.
5. Application initially queries according to location, plots it on a simplified map
6. User clicks on Time on the filter tab.
7. Application queries according to time, and plots the deals on a horizontally scrollable timeline.
8. User clicks on one particular deal.
9. Application sends request to server for deal detail history for that particular deal.
10. Directs the user to deal details interface with the analytics information.

Extensions

4a. User clicks on Best Performing Deals, triggers M7 - View Analytics Best Performing Deals.
4b. User clicks on Sweet Spot, triggers M6 - View Analytics Sweet Spot.

Expenditure	Base Cost	Yearly Cost Only	1st month	2nd month	3rd month	4th month
Office Space (5400 sq. ft)	8100	X	8100	8100	8100	8
Initial Office Setup	25000					
supplies (including computers)	20000	X	1000	1000	1000	1
Hardware Maintanence	2000					
Building Maintanence		12000				
Promotion and Adverising			3000	3000	3000	4
Salary	X	X	36000	36000	36000	37
Insurance + Other benefits	X	X	3100	3100	3100	3
Lawyers & Consultants (40 man hours per year	X	18800				
servers	X	X	200	200	200	
Total Expenditure	53100	30800	107,067	109,633	112,200	117
					412800	530366.66666
Number of deals			100	100	100	
Avg cost per deal			15	15	15	
Avg sales per deal			50	50	50	
Revenue			75000	75000	75000	187
					225000	412
Profit						

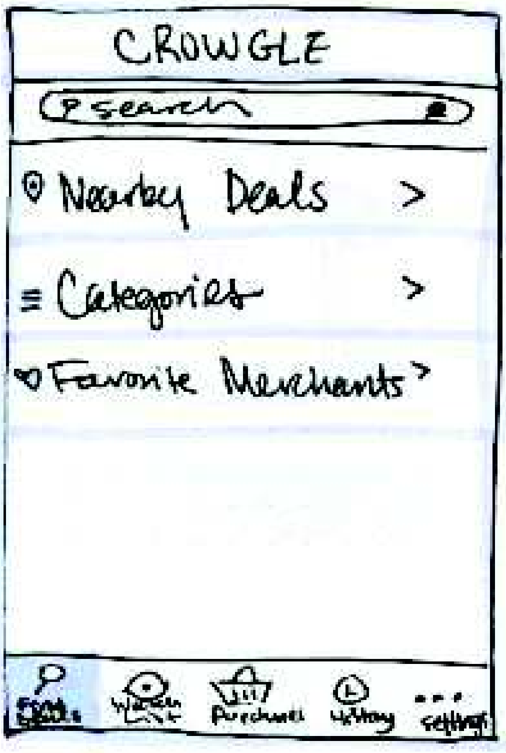
	5th month	6th month	7th month	8th month	9th month	10th month
3100	8100	8100	8100	8100	8100	8
1000	1000	1000	1000	1000	1000	1
4000	4000	4000	4000	5000	5000	5
7500	37500	51000	72500	86000	99500	108
3100	3100	3940	5200	6040	6880	7
500	500	500	800	800	800	1
,567	120,133	137,040	162,667	180,573	197,480	209
6667	650500	787540	950206.666666667		1328260	
		65628.3333333333				
250	250	600	1400	2100	3000	4
15	15	15	15	15	15	
50	50	50	60	60	60	
7500	187500	450000	1260000	1890000	2700000	4387
2500	600000	1050000	2310000		6900000	
		Break even				

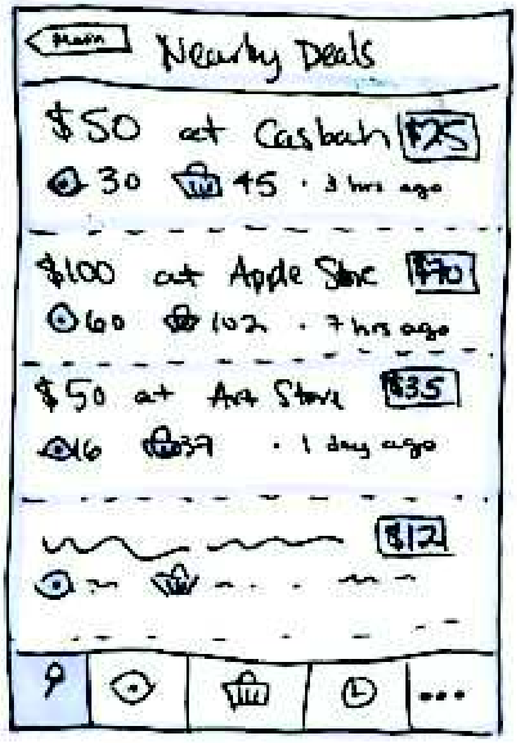
	11th month	12th month	TOTAL (1 year)	TOTAL (3 years)	TOTAL (5 years)	
3100	8100	8100	97200			
			5000			
1000	1000	1000	32000			
5000	5000	5000	50000			
3500	108500	108500	817500			
7300	7300	7300	59460			
			18800			
,000	1,000	1,000	7,500			
,667	212,233	214,800	1,087,460			
		1964960	Requested Funding: $1.1M			
						Avg Running C
						159321.666666
4500	4500	4500	21400			
15	15	15				
65	65	75				
7500	4387500	5062500	20737500	277219353	804702010	
		20737500			8 Billion Dollars	
			19650040	1.5M$?		
			19.7 Million Dollars	1564378.33333333		
		1.87 correct da		18.7 Million Dollars in Profit		

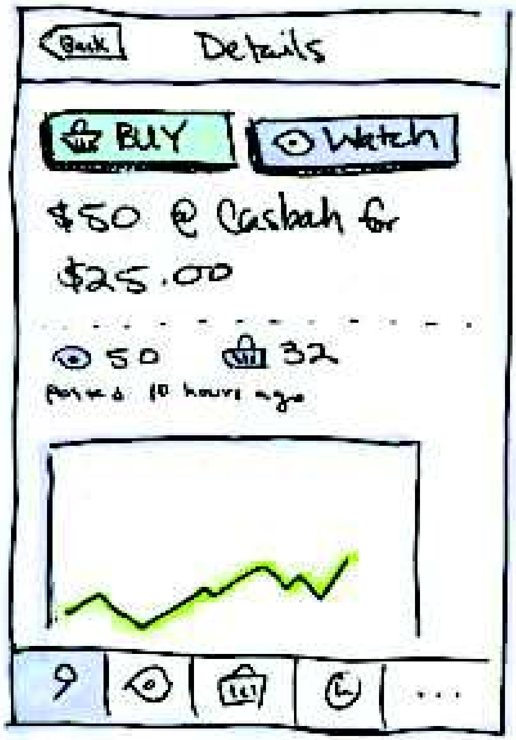
ost

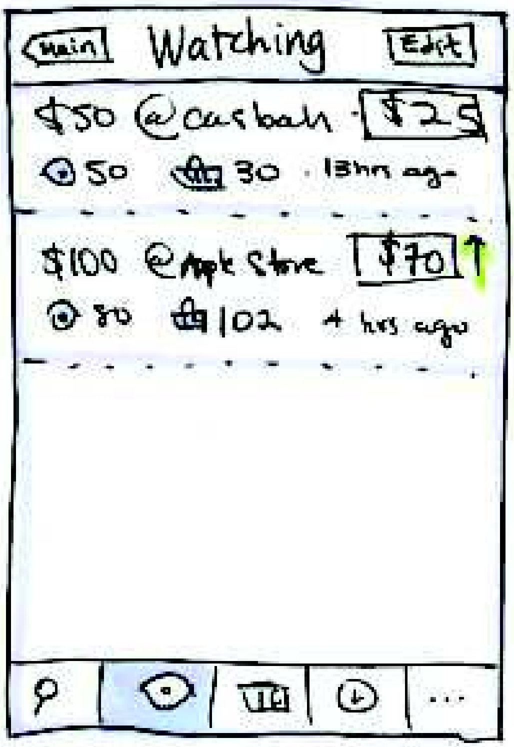
667

Customer Wire Frames

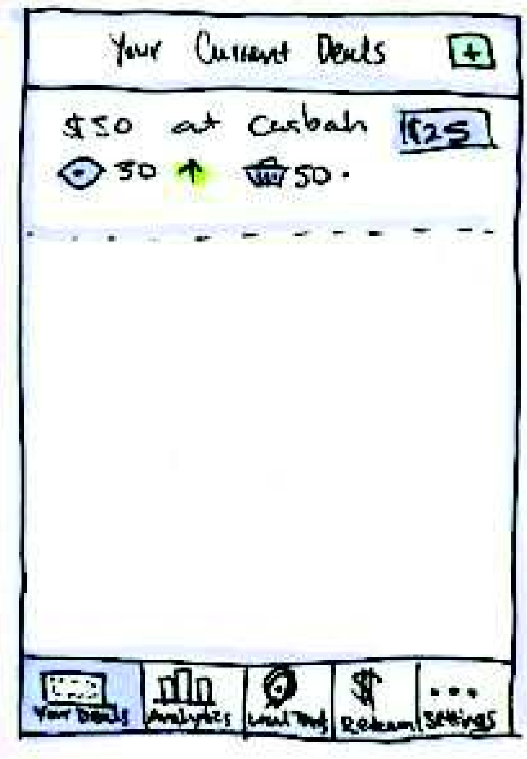


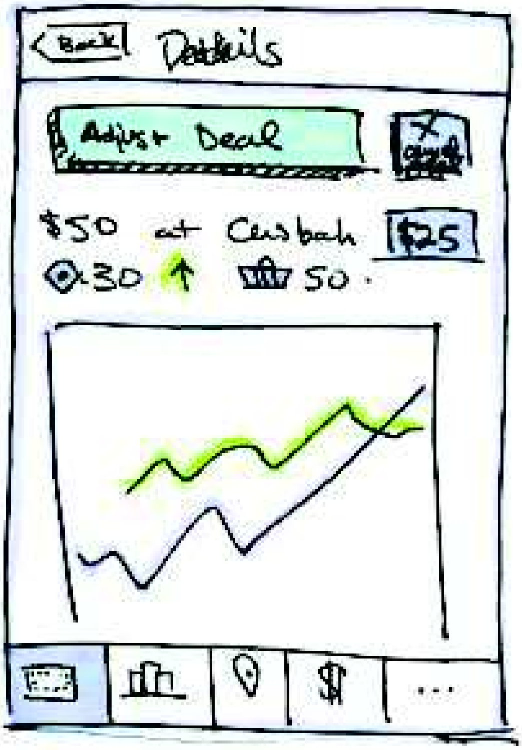


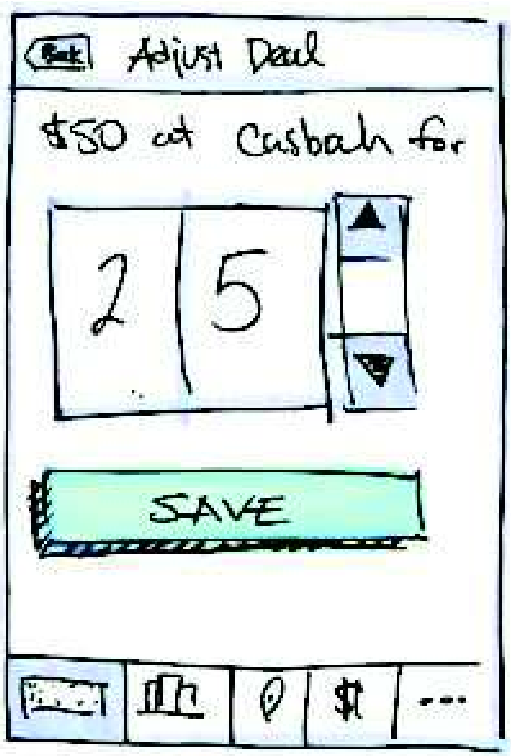




Merchant Wire Frames


Your Current Deals
+
$50 at Casbah
$25
30
50
Your Deals
Analytics
Local Tools
Redeem
Settings
...


Back
Details
Adjust Deal
X
$50 at Casbah
$25
30
50
...


Back
Adjust Deal
$50 at Casbah for
2
5
SAVE
...