The Feiner Things

Jenee Benjamin, jlb2229
Dan Mercado, dm2497
Orlando Pineda, omp2114
Varun Ravishankar, vr2263

Purpose

Yelptastic is a web app that allows users to bookmark businesses on Yelp and let them sort, filter, and delete them as well. Yelptastic's code is located on Github.

Overall Process

Our overall process for creating Yelptastic followed a typical design cycle. We first up came up with use scenarios and then used them to create mockups. We iterated over the mockups many times, trying to decide the best way to display bookmarks, search through and add new bookmarks, and display a unified interface for searching and filtering through bookmarks by category, rating, distance, and date added. Once we were satisfied with the mockups, we started programming the site. We broke up the process into two parts, the front-end and the back-end, and the group split up to work on each part. We would sync up our work when one group member completed a part, and finish integrating that portion on the site. Throughout the process, we looked through the site and evaluated our design decisions to better improve the user experience. We altered our search boxes to resolve consistency issues between other forms on the web and ours, worked on adding spacing to improve readability, and to improve our error handling. To sum up, we went through the following process:

- Brainstorming
- Use Scenarios
- Look over feedback on use scenarios
- Start looking at different ways to visualize search results
- Create Yelptastic mockups
- Get feedback on Yelptastic mockups and iterate
- Program Yelptastic
- Test site and look for design issues

Roles and Responsibilities

We split the project into two parts: the back-end, which was concerned with querying the Yelp API and parsing the results, and the front-end, which took the data from the back-end and displayed it to the user.

Orlando and Dan worked on the front-end, while Jenee and Varun worked on the back-end. Varun worked on querying the Yelp API and writing functions to deal with geolocation, while Jenee worked on adding the API results to localstorage.

Target Users

We are targeting young adults, 18–25 year olds with our application. These young adults are comfortable using technology and may or may not be in college. They live in urban and suburban areas and are surrounded by a variety of restaurants and businesses. They utilize Yelp Favorites to make it easier for them to find good places to eat and to keep tabs on the businesses. They like to use the tags and notes features so that they can make note of and identify unique characteristics of establishments that they have favored. They also use these features to help them better organize their favorites. Because the number of favorites a user can have is unlimited, users can search through their favorites and have the options of filtering by category, distance, and rating and/or sorting by distance, date added, and rating.

Personas:

Greg is a 17 year old college freshman who recently entered a relationship with his girlfriend Gina. He is majoring in Computer Science and minoring in mathematics so that he could find a steady career and support himself in the future. As a busy college student, Greg does not have time to cook, and goes out to eat at restaurants often with his girlfriend. Greg has a Yelp account and uses it to find good restaurants in New York City. Once he finds a restaurant that he likes, he goes onto Yelp and adds that restaurant to his favorites with a little note about what made his experience at the restaurant a good one. Greg tries many different restaurants and tends to lose track of restaurants that he adds to his favorites. The new Yelp Favorites application helps him organize them and allows him to quickly add new businesses and delete businesses that he no longer likes or that no longer exist.

Steve is a 21-year-old college student, trying to graduate with a degree in creative writing with the least amount of work possible. Steve loves partying with his frat brothers, but his friends have stopped funding his party animal antics after he puked on the couch and left without cleaning it up. Steve is now on his own and is too upset to leave his house to get food. As a result, Steve is using Yelp to find the closest restaurants that will deliver to his area, so that he does not have to pay a delivery fee. He adds those businesses to his favorites for future reference so that he can simply go to his favorites and get the information for those places easily.

Use Cases:

Scenario One:

Greg wants to take Gina out for their one-month anniversary. He is planning on having dinner in the area. He lives near Times Square and plans on walking around the area and visiting the attractions. He first wants to find a restaurant nearby. To find a place to eat that he knows is good, Greg visits Yelptastic and goes to his favorites. Once he is on the Favorites homepage, he types “food” in the search box. He filters his results by rating, by selecting “Ratings 2 Stars and Up” title, and looks at his results that are in and near the Times Square area. He reads a note that he left for ”Dallas BBQ" previously which says that the service there was amazing. With its convenient location and good notes, Dallas BBQ is Greg’s ideal choice of restaurant, and he decides to take Gina there.

In addition, Greg wants to look up the common tourist attractions to take Gina to after dinner. Greg goes back to his Favorites and searches through his favorites tagged as “entertainment”. He then filters those results by rating to see the best attractions with 2 stars and up near his location (Times Square). He sees Monty Python’s Spamalot, which he had previously visited; however, he decides that this is a bad place to go to for a date, and decides to delete the favorite so that it no longer shows up. He then looks through his results again and plans to bring Gina to the M&M store after dinner.

Scenario Two:

Steve no longer dines out and is no longer living his party animal lifestyle. To adjust to this new way of living, Steve wants to create a list of favorite restaurants that sells cheap food and that will deliver to his house. He knows that pizza is one of the cheapest foods in his area and that pizza places nearby will deliver at no charge, since he lives in their surrounding area. To go about makeing a list of these places, Steve uses the Yelp Favorites application. He goes to the Yelp Favorites page, types “pizza” into the “Search For New Favorites” box and enters his zip code “10025” as his location. Steve looks through the list of nearby pizza places and adds to his favorites pizza places with good ratings. As he adds each business to his favorites, Steve adds various tags like “good_pizza”, “pizza_looks_yummy”, or “John’s_fav” to the pizza place favorite. He does this by typing the tag in the popup dialog box that appears once he clicks “Add Favorite” for that particular restaurant, and then he saves it. After Steve has added shops to his favorites, he goes to his Favorites homepage, and searches through them by searching for “pizza” in the “Search Favorites” box. Because of the various businesses that have tags that have “pizza” in them, he gets plenty of results. He sorts the businesses by rating and finds the highest rated pizza place- Koronets. He sees Koronets’ phone number there, and places a call to order a large pie with anchovies.

Design Decisions

The first major design decision we made was to target a younger, more technologically-savvy demographic. This group of users would be familiar with review sites like Yelp, and would be used to searching for businesses online to find interesting restaurants or shops. This meant that we could remain consistent with Yelp’s look and feel for each result and depend on the user to know what iconography and basic terminology meant.

We made major design decisions when creating our mockups. Our first mockups and initial thoughts when writing the use scenarios involved lists of items that could be filtered and searched through, with a search bar at the top and a filter box similar to Amazon’s filters on the right. Categories would be displayed in a grid. When a user clicked on a category, they would be taken to another grid of subcategories. For businesses with a category that was two or three tiers down (such as Restaurants > Chinese > Dim Sum), the user would need to click through two categories to see their actual result, and the user would need to click on the back button to go back and browse through another category. We decided that this option did not offer the user much *user control*, and so decided against the grid approach. We also decided against combining this view with a list view, as too many views would simply confuse the user and would not offer much *consistency*.

We initially wanted to sort by price, as price is a major criterion for deciding which restaurant to eat at or which shop to buy from. However, the Yelp API does not offer a way to filter results by price, nor does it offer a field to view its dollar sign rating (one dollar sign to five dollar signs). We would have had to crawl the web to show prices, and so we decided against it. However, we did realize that this would have been a feature that our target audience would appreciate, and given more time, we would have liked to implement this.

We had trouble deciding where to include tags and notes. Initially, we wanted to include a tag cloud, like in Wordpress or other Web 2.0 sites. Users would be able to click on a tag to view all the items with that tag. This was awkward though, so we decided to just display all the tags each item had and allow the user to search tags, notes or bookmarks.

We knew that pagination was important to implement: one of the downsides of the list approach to display bookmarks is that you would have to keep scrolling to see your bookmarks, and then scroll all the way back up to search or filter again. However, we decided against display page numbers: the Yelp API returns the total number of results, and so it is possible to calculate the appropriate number of pages. However, these page numbers do not mean anything outside of Yelp’s API; while they do tell the user how many results there are, if the user cannot find what they want on the first few pages, the search results are not displayed in the best order possible and so are not doing their job. We thus decided to stick to Previous page and Next page for simplicity’s sake, and to remain as *minimal* as possible.

One of the major issues we had was with displaying errors. We knew that error checking was important, both to give the user quick feedback and to prevent wasting API calls. We were able to take care of most of the user-caused errors right away, but had trouble handling Yelp API errors. Because the Yelp API is using JSONP to make API requests, and because JSONP does not have any error handling built-in, we were unable to detect when Yelp API errors occurred. Errors could be caused by reaching API limits, for example, or because Yelp was down. We found a library that could detect general error conditions, but would be unable to detect what error was occurring. We were thus forced to just show a generic error page when an API error was thrown, which is not very user-friendly. We provided a generic solution as well: to go back and try again. On the other hand, we decided that users should not have to be concerned with what errors were occurring, and so decided that this was not a terrible scenario.

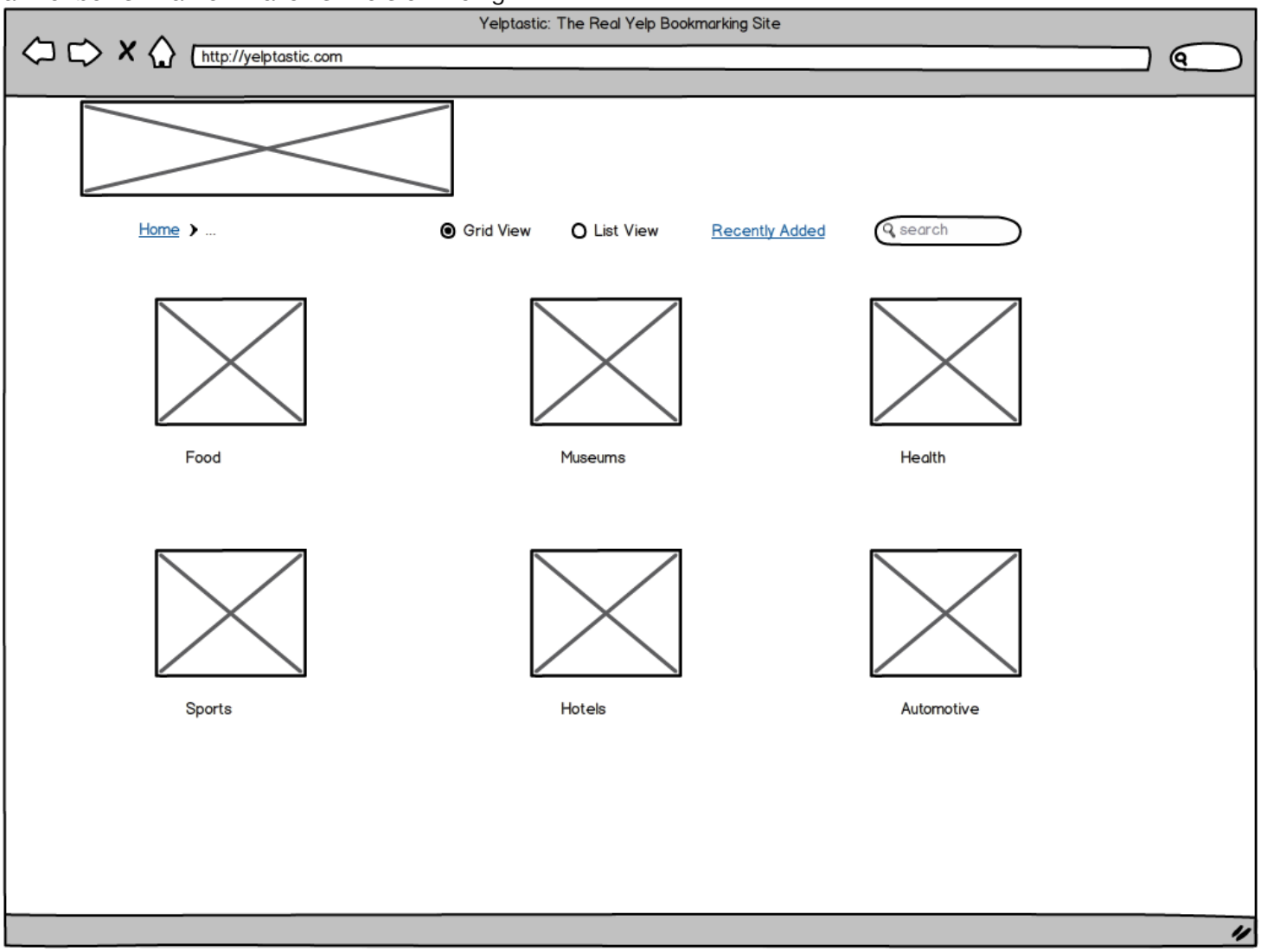
Our final design took inspiration from Amazon, Newegg, and Yelp itself. We made a category listing similar to Newegg's category listing, and implemented filtering like on Amazon, where multiple filters could be active at once. This ensured that our site was *consistent*, and also made sure that users could *recognize* familiar UI patterns and use those to more easily navigate our site. We also made sure that we had some basic *help* instructions to ensure that the user could figure out how to use Yelptastic.

Prototyping and Testing Process

Prototyping

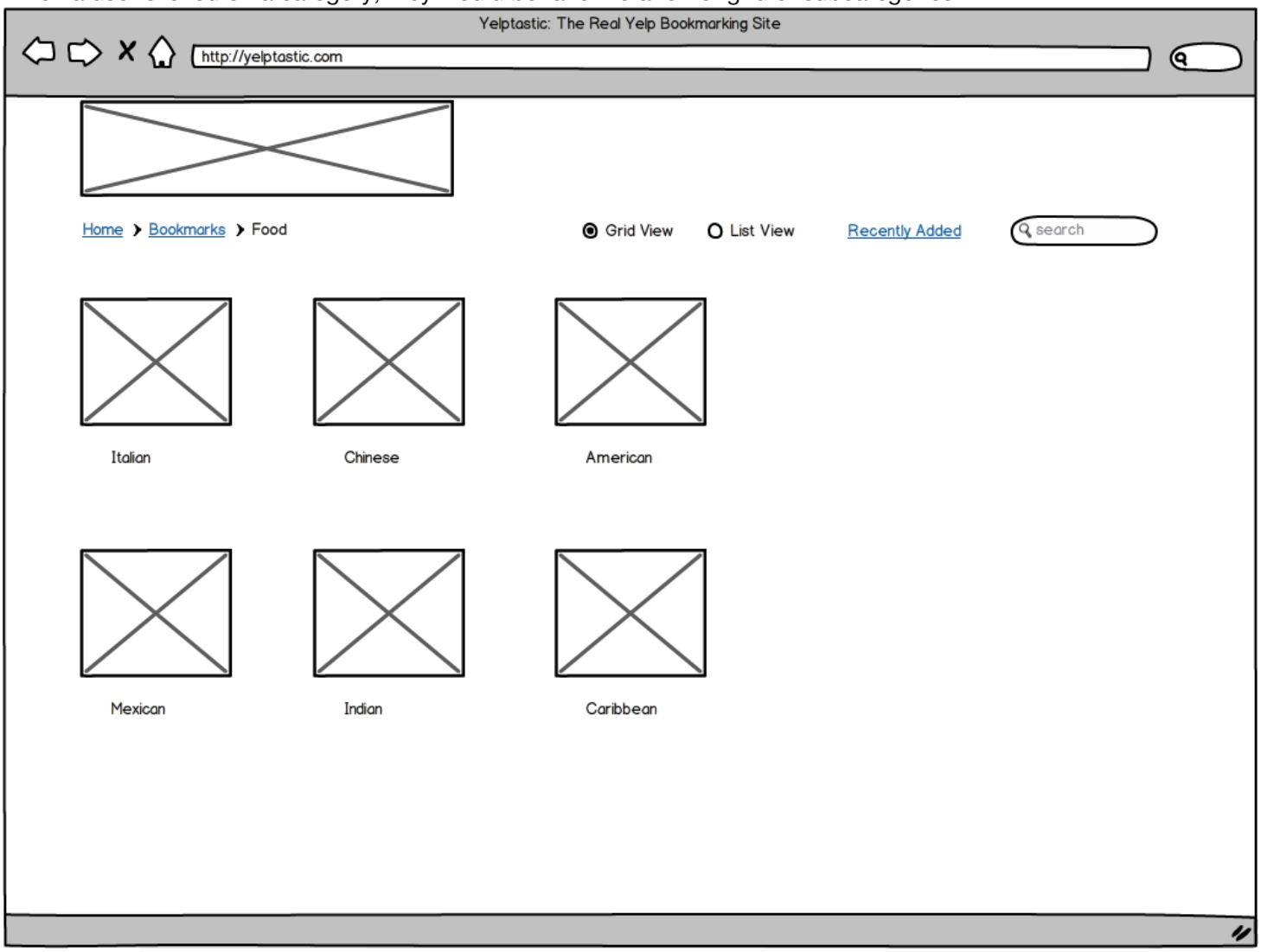
Our prototyping process consisted of mockups drawn in Balsamiq and of sketches drawn on whiteboards. We chose not to use more sophisticated lo-fi prototyping techniques like paper prototypes because of their cumbersome nature and the time involved in making them. We found that we were better able to communicate our ideas through prototypes in Balsamiq, and that they were easy enough to make once you understood how to use the tools provided.

Our first mockups involved lists of items that could be filtered and searched through, with a search bar at the top and a filter box similar to Amazon's filters on the right.

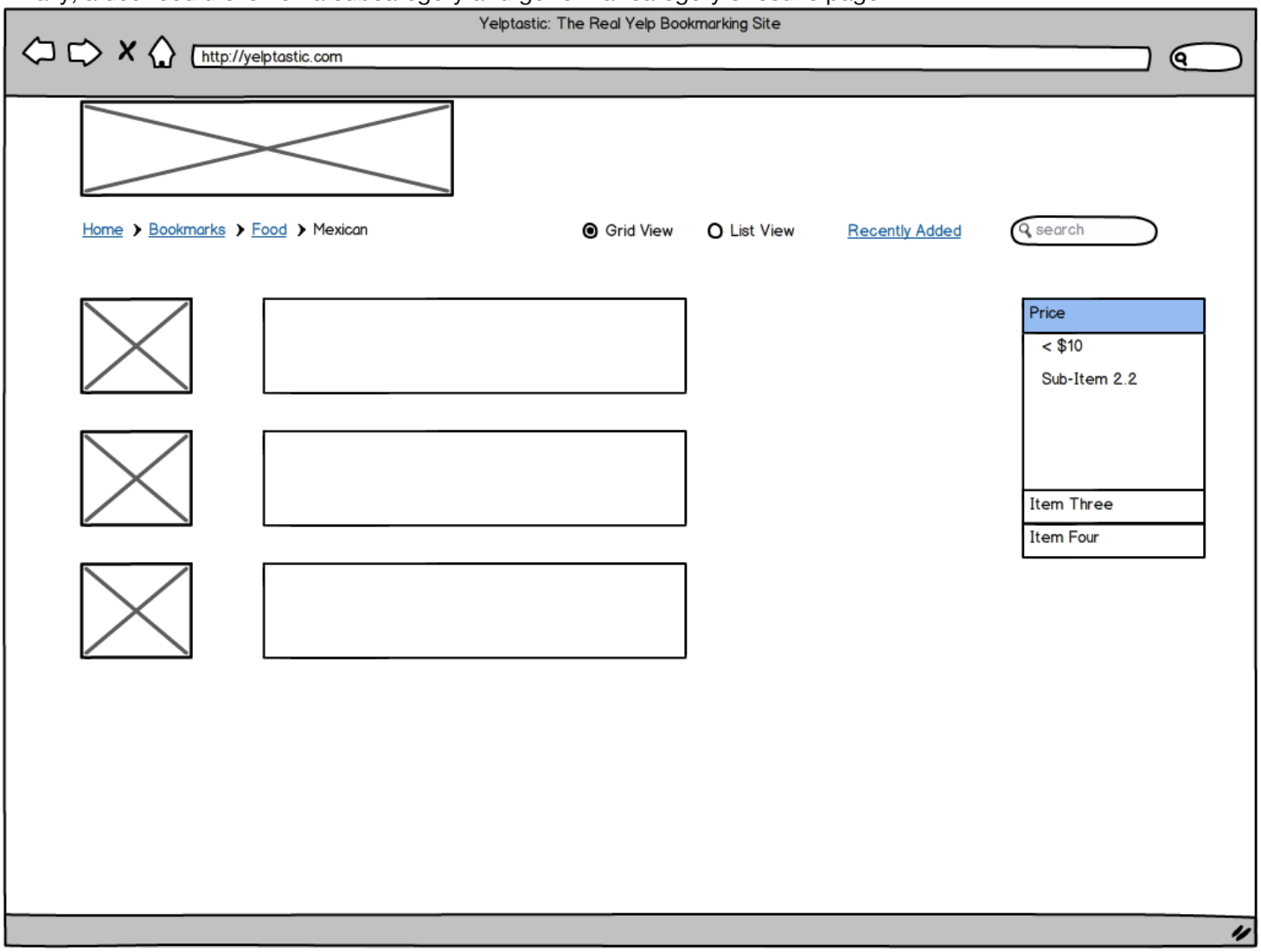


Categories would be displayed in a grid.

When a user clicked on a category, they would be taken to another grid of subcategories.

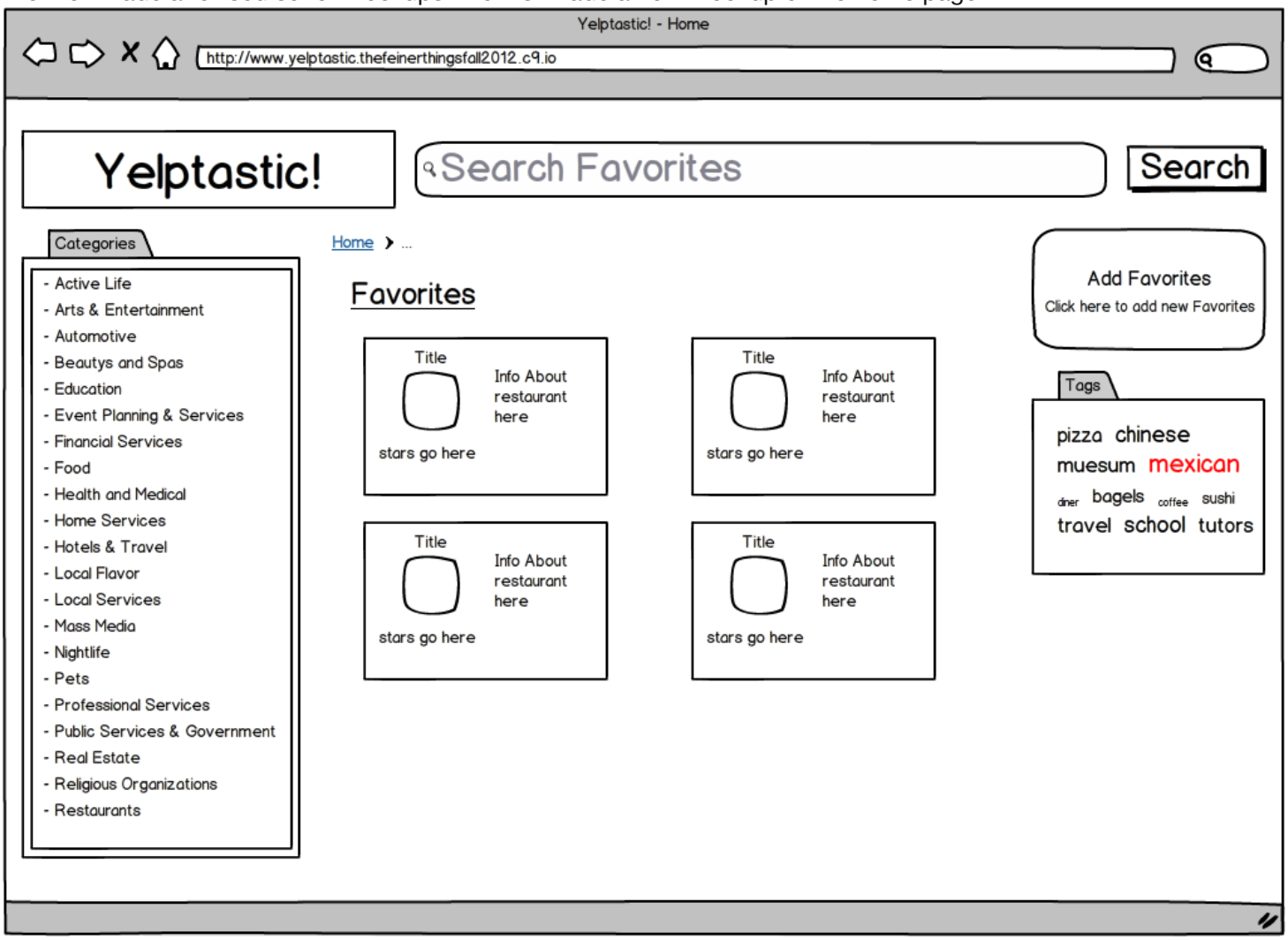


Finally, a user could click on a subcategory and go to that category's results page.



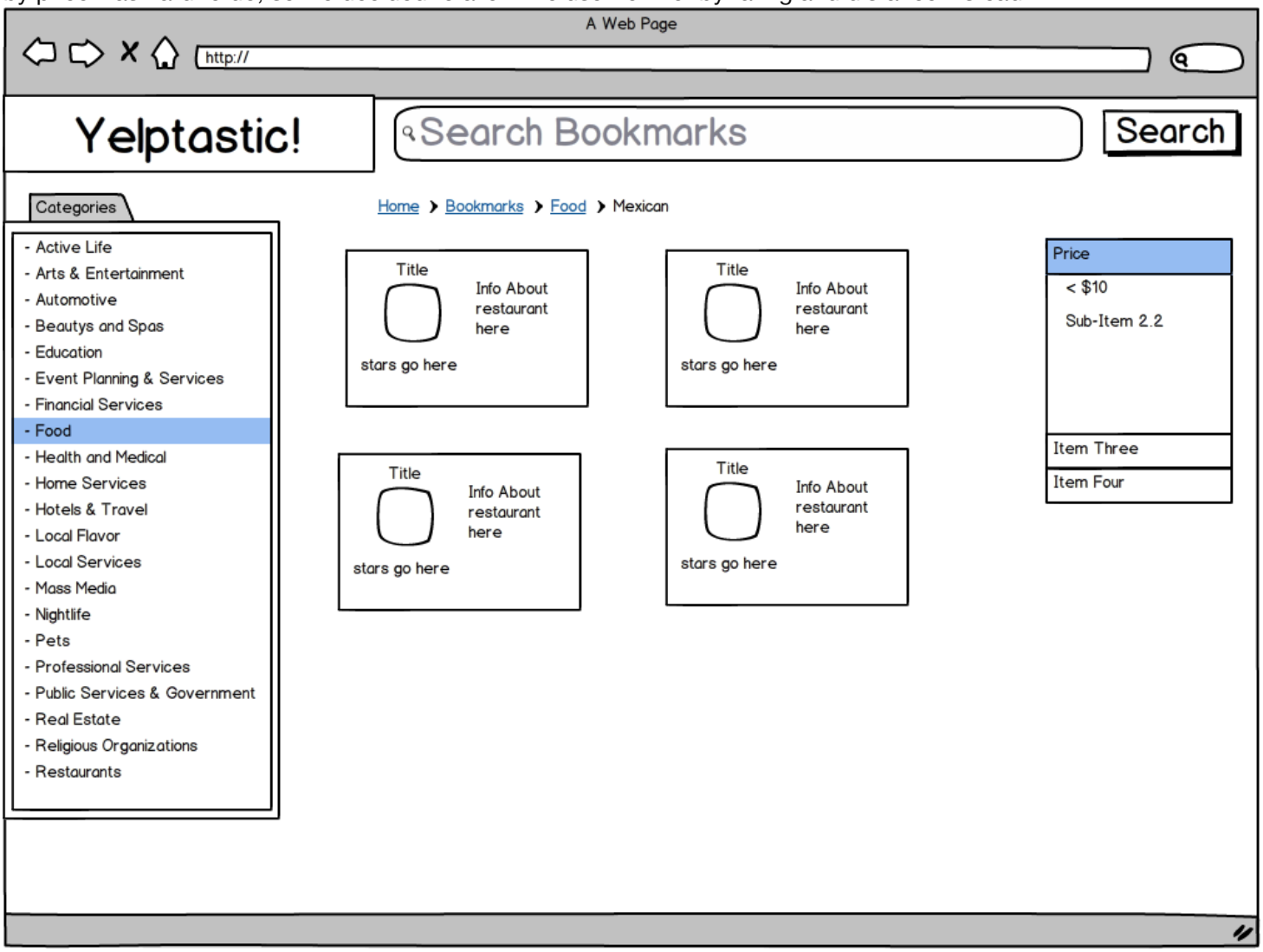
For businesses with a category that was two or three tiers down (such as Restaurants > Chinese > Dim Sum), the user would need to click through two categories to see their actual result, and the user would need to click on the back button to go back and browse through another category. We decided this was inefficient, and so got rid of the grid view. We also decided against combining this view with a list view, as too many views would simply confuse the user.

We then made a revised set of mockups. We first made a new mockup of the home page.

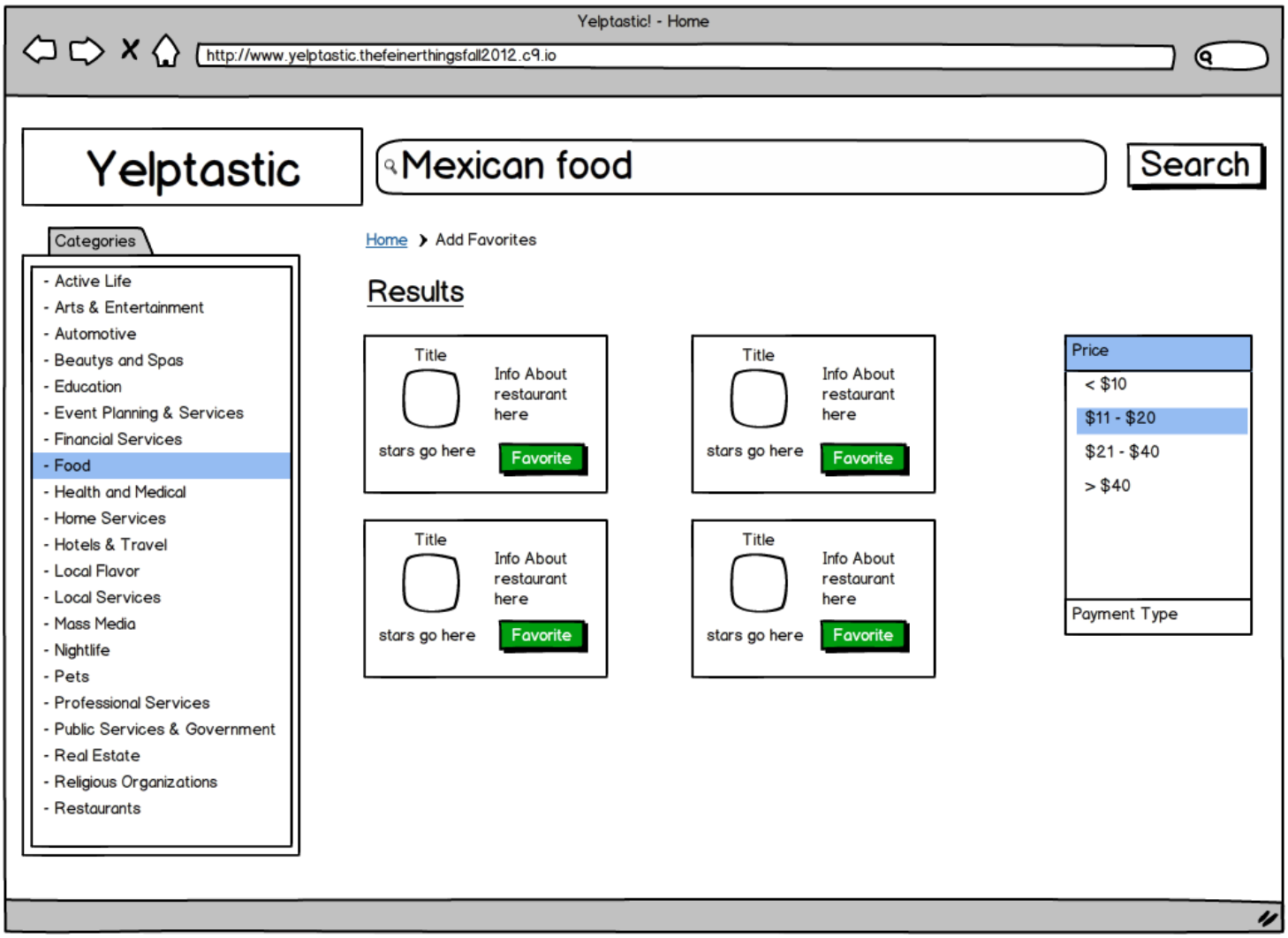


In this mockup, users would see their favorites displayed as a grid, but the grid would consist of results, not of categories. Users would also have the option to search through their favorites, or to add new favorites. This dialog ended up getting redesigned. On the left side, users would be able to see the list of categories that describe these favorites. We also added a breadcrumb bar to allow the user to navigate back and forth between pages.

The next mockup of the search page would allow the user to filter through their results. We discovered that filtering by price was hard to do, so we decided to allow the user to filter by rating and distance instead.



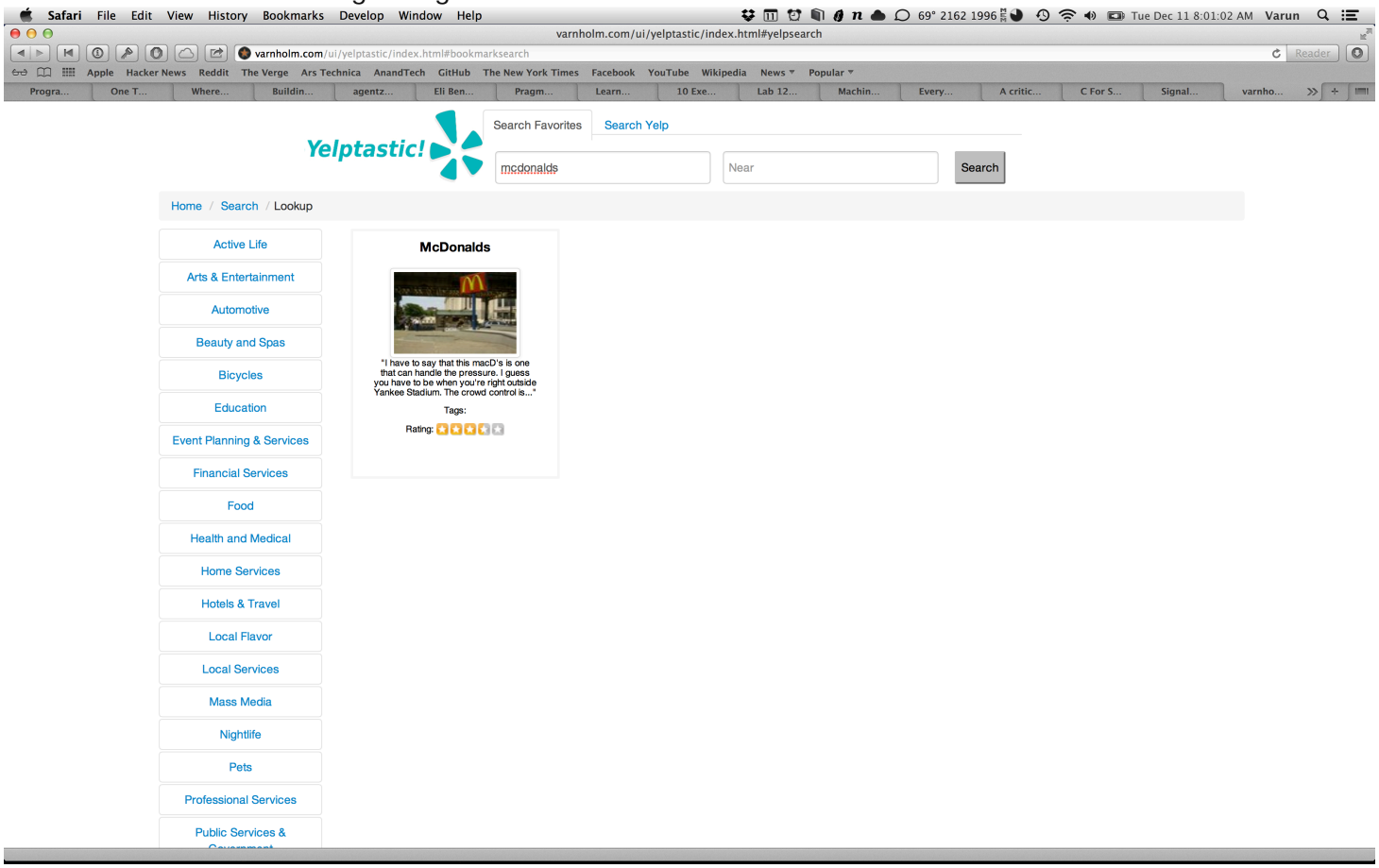
The final mockup is of the add bookmarks page. If the user searches for something like Mexican food, results would be displayed as a grid and would let the user favorite a restaurant or other business. Again, the user would be able to filter through the results. We did not like how the use would have to go back to the home page to search through their bookmarks again, so we decided to combine the add bookmarks and the search bookmarks bars into one tab bar.



Testing

We tested Yelptastic throughout the coding process. While we did note visual and aesthetic issues, we focused on usability issues as much as possible. Everybody in the group tested the page as they were coding, but usability tests were spaced out and done when a group member had time.

Varun was the first one to do testing. He focused on error handling, application responsiveness, and consistency. Varun had feedback on making button labels more informative, and on making sure that entering blank text into one of the search boxes would not force a search that would in turn simply throw an error. He also noted that when searching through bookmarks, there was no indication that results were clickable, and noticed that there were sometimes issues when trying to use the keyboard to submit a search query (ideally, navigation would be equally straightforward when using the mouse or keyboard). He also noted errors with the breadcrumb bar, where adding bookmarks and then searching through bookmarks would lead to a bookmarks bar with both results.



Software Engineering

Yelptastic is written in HTML, CSS, and Javascript. It uses many HTML5 features, like Local Storage, Geolocation, and the History API. Open-source components used in Yelptastic include:

- jQuery
- underscore.js
- Bootstrap
- Handlebars.js
- ouath

Tools we used include:

- Github
- Cloud9

All code for Yelptastic can be found on our team's Github repository.