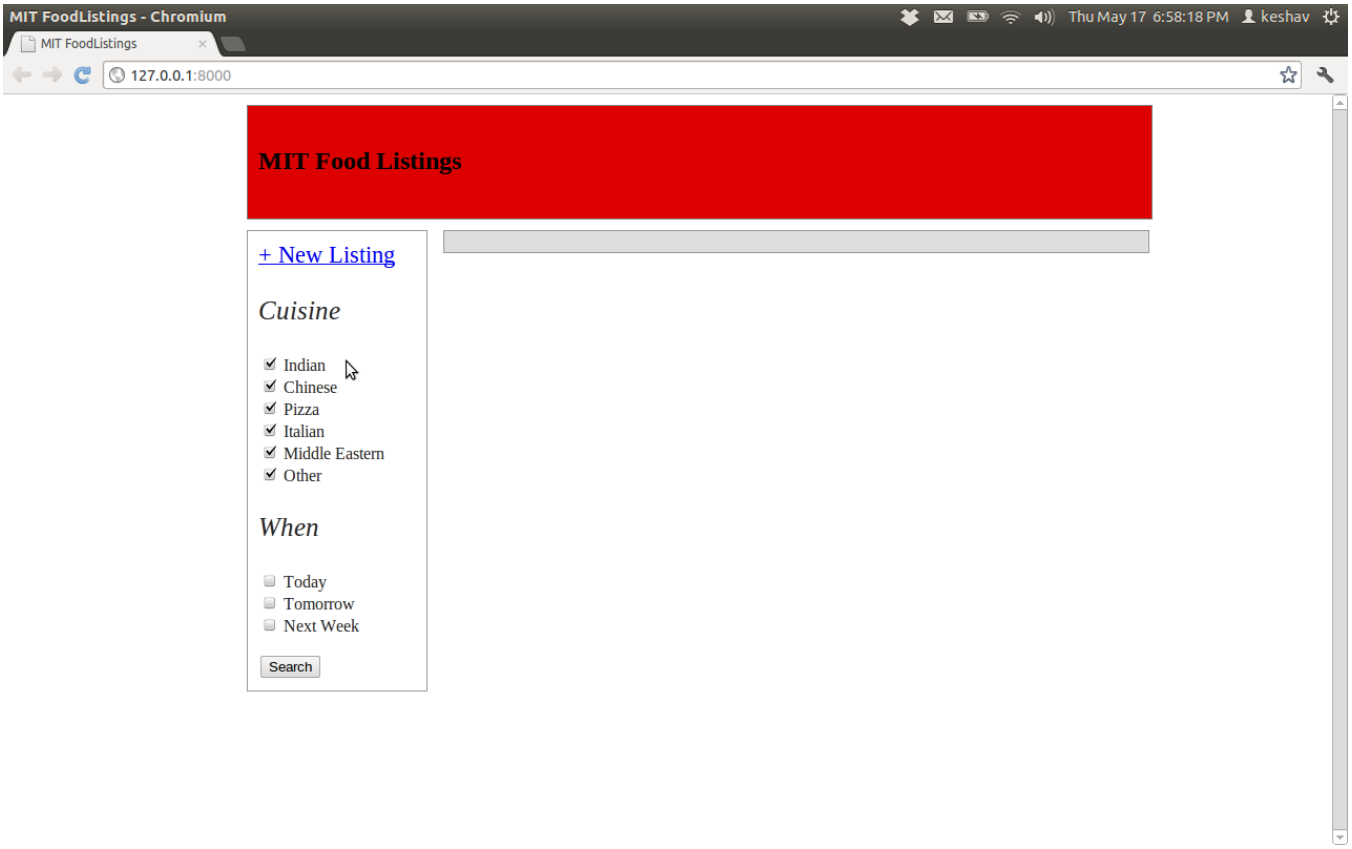
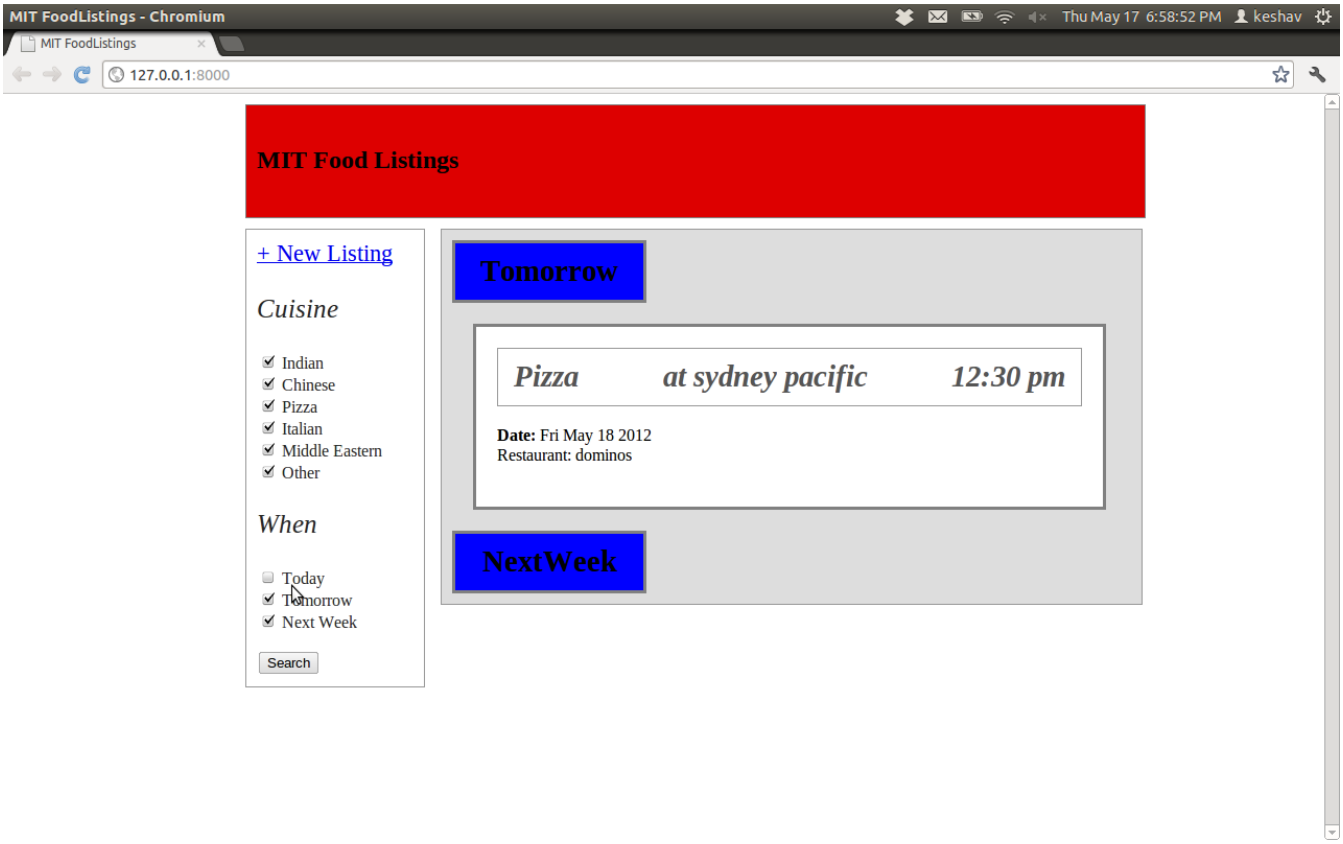
Design
When someone first arrives at our site, they see the following:



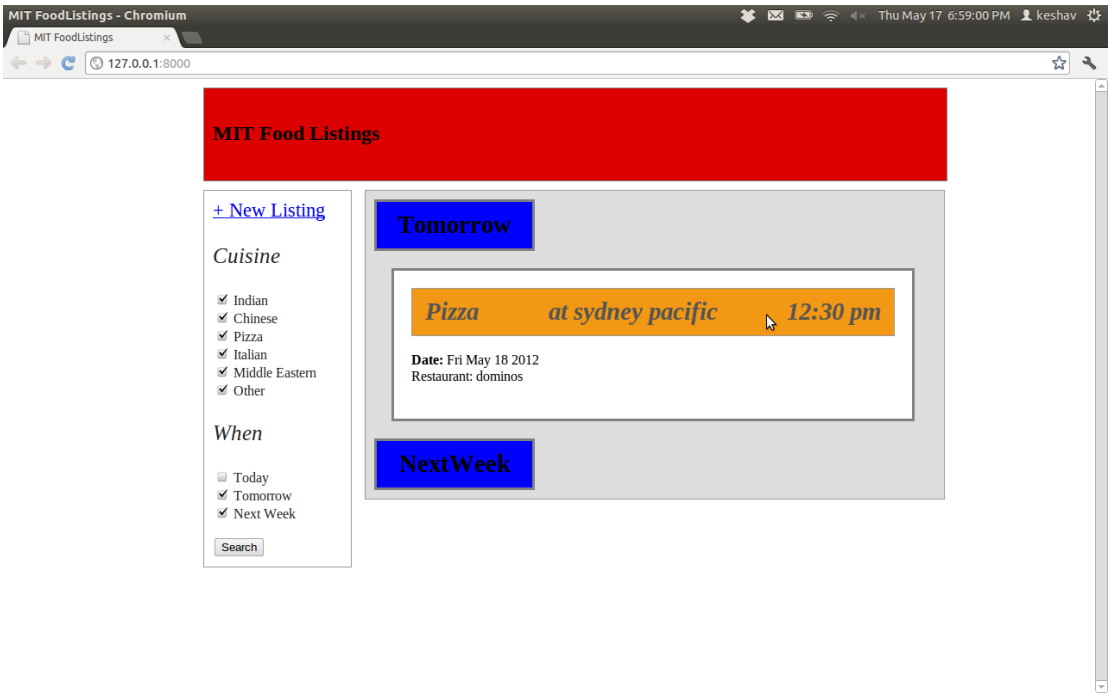
They see a blank screen with a filtering bar on the left (our default behavior is to have all options clicked, because most users don't care about type of food and are generally looking for food ASAP. This decision was based on user testing, before which we had all options unclicked, but users complained, and we felt that they were correct). We show the options unclicked just to show what the naked website looks like.

We considered different schemes for the filter bar, including the possibility of including more filters. However, we wanted to make the main focus of the website to improve learnability, as most users will only use the website maybe 10 minutes a week, and don't have the time or the need to become "expert" users. Hence, we kept the design of the search bar very simple and easy to learn, instead of adding extra features that would make it complex (but perhaps more efficient for expert users to pinpoint the exact events they desire). ANother reason for this tradeoff was presumably most users of our site aren't too picky (because they are looking for free food), so only simple filters should suffice for most users.

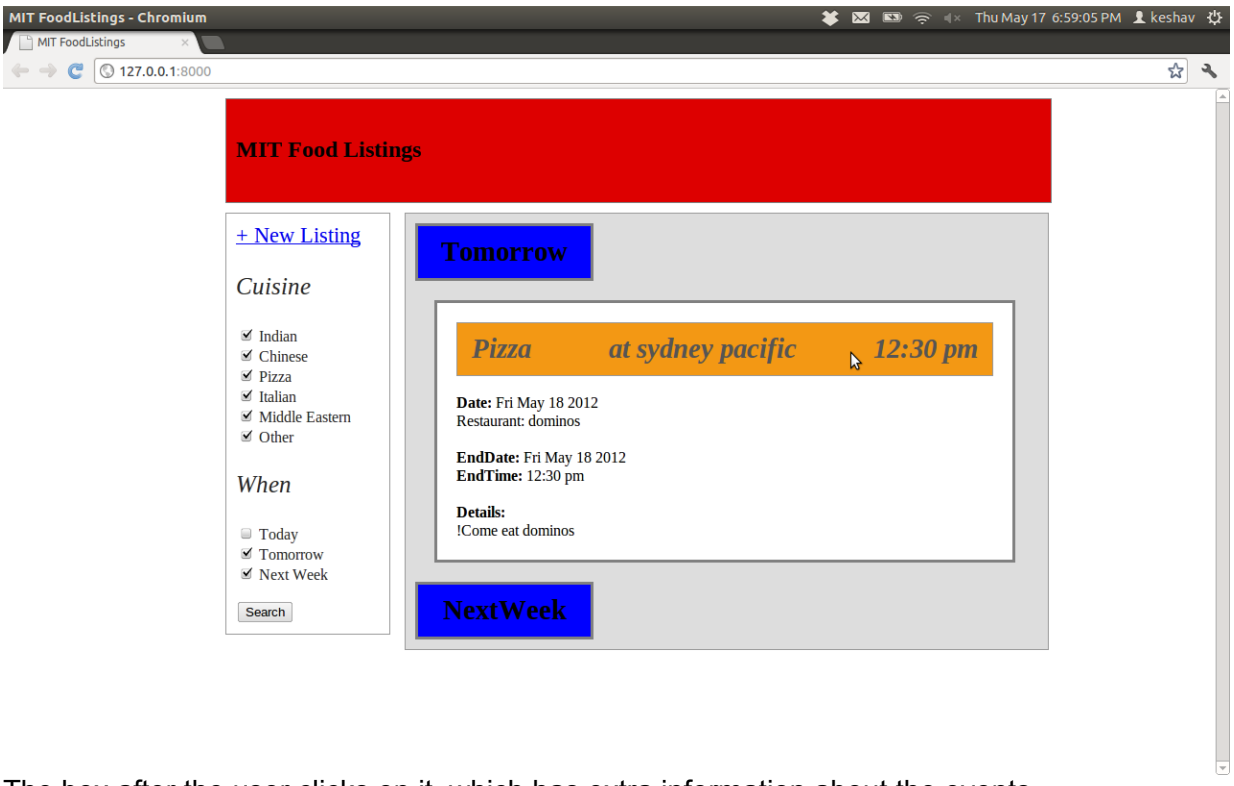


If a user clicks on one of the dates, a box shows up with that date, under which all the events which match the filters and dates are shown. The important information (location, time and type of food) are shown in large font. In early computer prototypes, we had all the information in the same font, which users complained were unreadable, and they suggested focusing on the important information. We also provide an affordance (box) around the important information, which seems clickable. If a user clicks on the box, they see additional information about the event. Furthermore, when scrolling over the box, the background changes color, giving another affordance that it is clickable. These features were present since the first computer prototype. We show a screenshot below with this color change demonstrated.

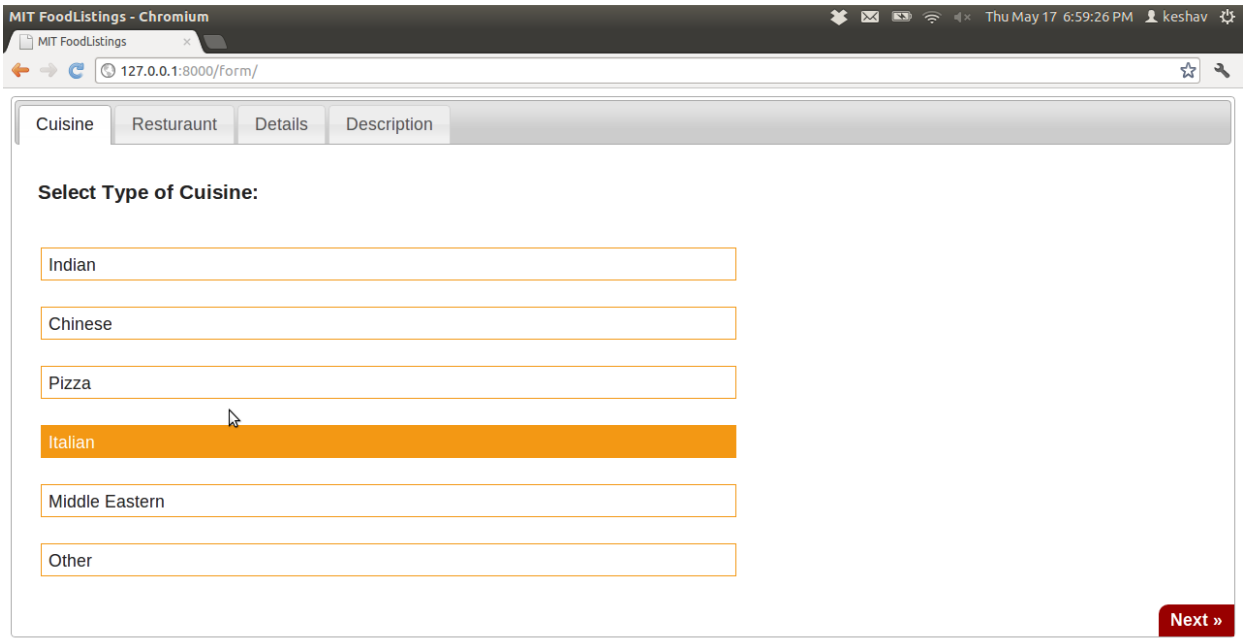
The other important design decision we made was how to display the listings (straight format where you scroll vertically, or some more complicated design arrangement, like a calendar arrangement or something similar). However, we decided to just a listing type interface for two primary reasons. First, it is very simple to learn, and many users would be familiar with interfaces similar to this. Second, its efficient for most people, because most people are just looking for the “next” event with free food, which is the first event displayed (and it is displayed prominently). Very few users would find a more complicated interface useful, because users were generally not interested in comparing events (we got this feedback in user testing), and this complicated interface would “hide” the most important event (by making it smaller).



The box before a user clicks on it



The box after the user clicks on it, which has extra information about the events.

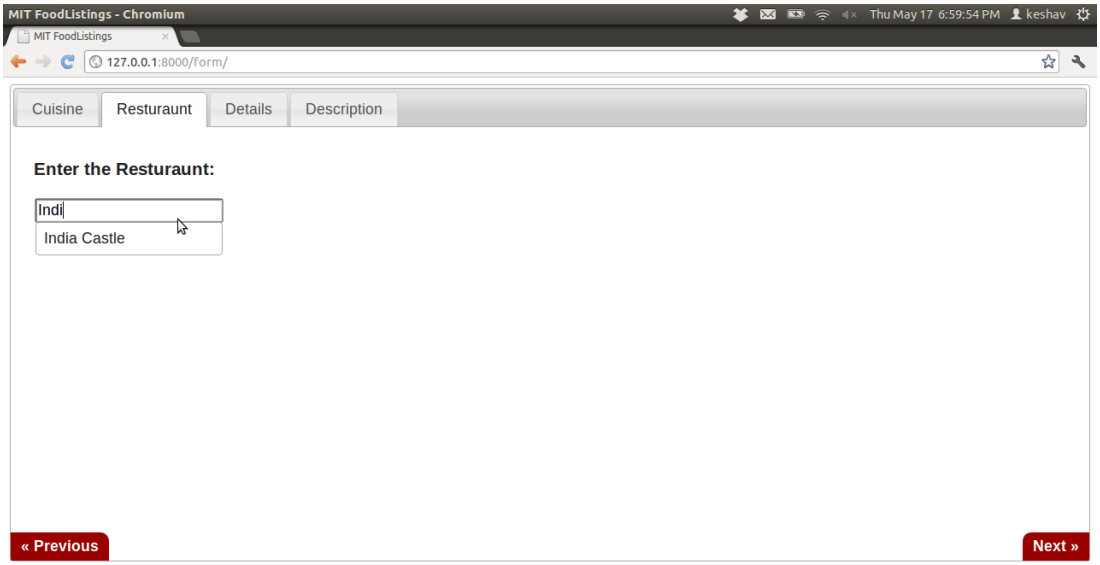


The other thing a user can do is add an event. They can do so by clicking new listing on the previous screen (which has clear affordances of clickability). In this screen, the user can choose the type of cuisine. We got negative feedback about the affordances for selecting the type of cuisine in user testing and in heuristic evaluation, and would change it in the final version. Another thing users noticed was there was no simple way to go back.

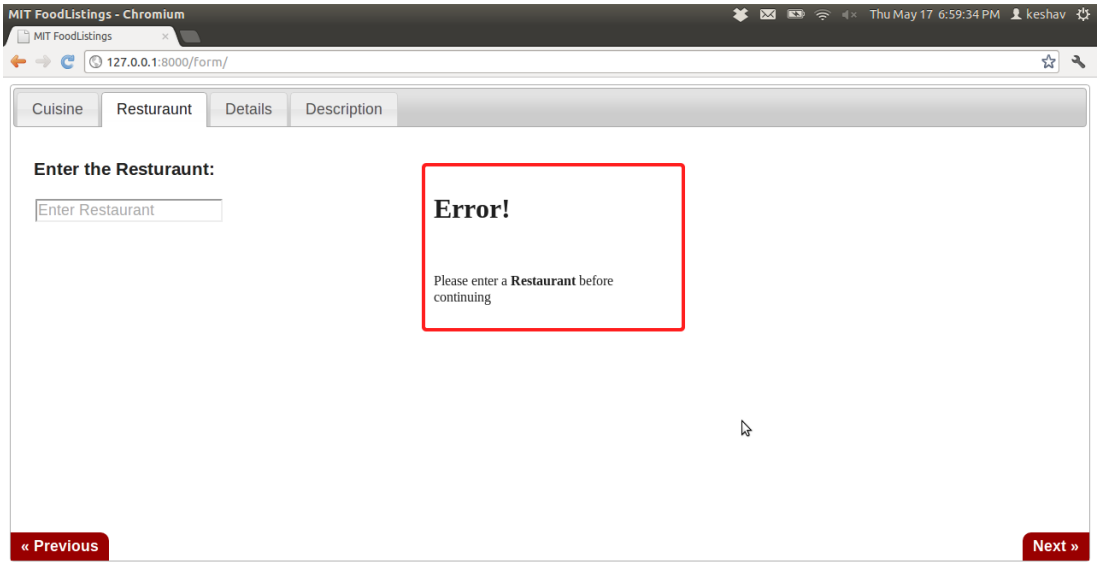
One important design decision was to make the process of creating an event completely isolated from the search page. We did this for two primary reasons. First, most users are here only for search. Second, in paper prototyping, when we tried to integrate the two onto the same page, the design confused many users, creating security issues because users accidentally created events.

Another design decision we made was to split the form into multiple stages, as opposed to keeping the different categories on the same page. We tried both in paper prototyping, but decided to split the design because of positive feedback from users -- mainly over security and learnability concerns. The unified design offered better efficiency for expert users.

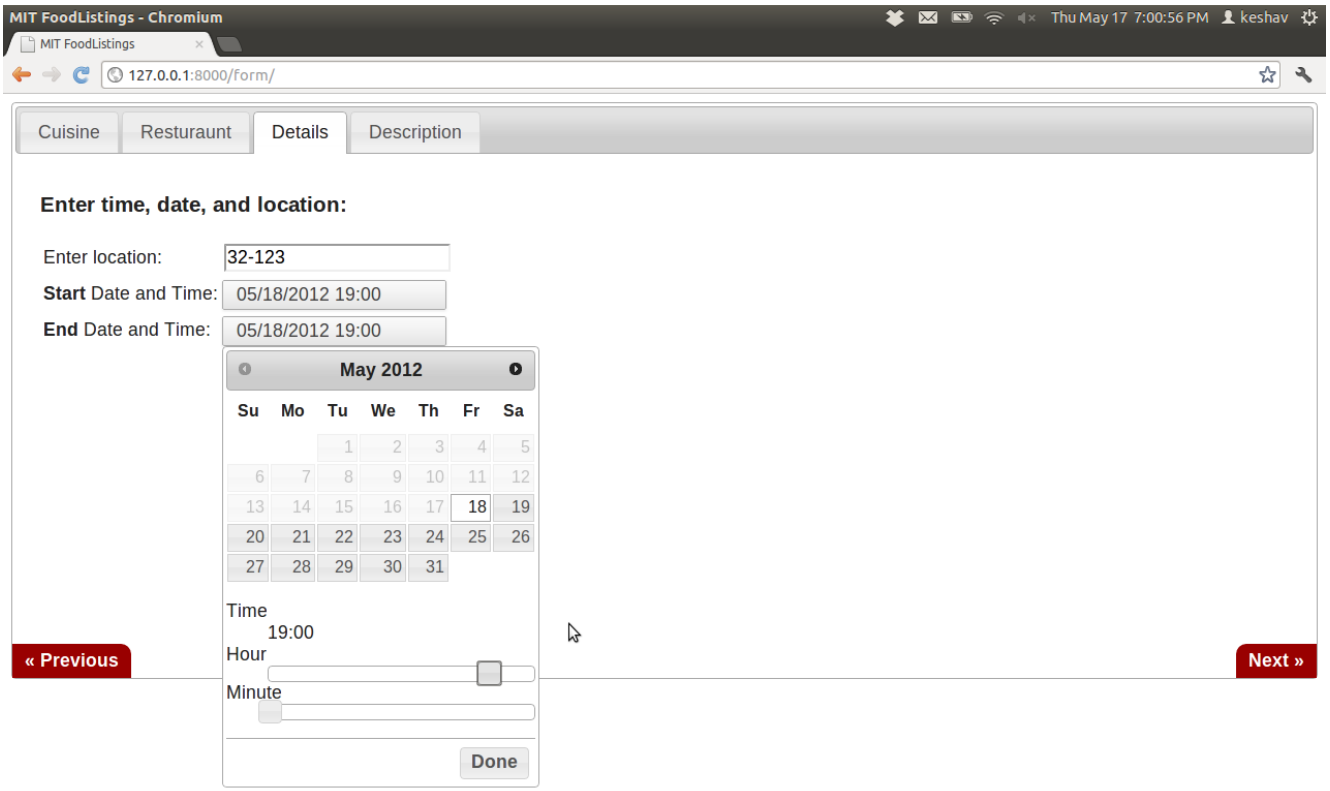
After selecting a cuisine, the user can click next.



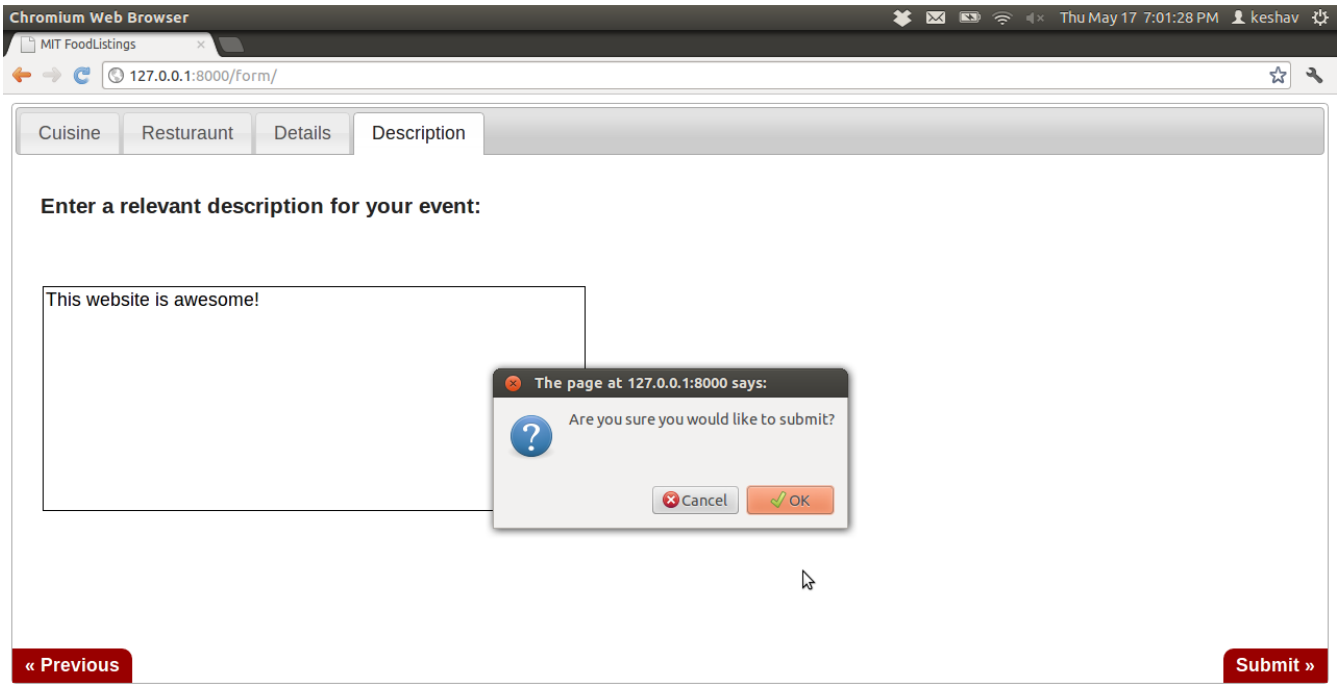
The next page is where the user enters the restaurant name. We added the auto complete feature after noticing that users often made typing mistakes.



We also check on each page whether the user has entered all the information for that page (form validation). This increases security, but decreases efficiency.



In the next page, the user can enter location and time. We added auto complete to the location box after feedback from heuristic evaluation. We also added a time/calendar widget, although the widget is not perfect in picking time (Fitz's law issues). The reasons for not improving the widget is discussed in the implementation section.



We finally have the last screen where the user can enter detailed information for the event. THe user can also easily go back to any of the previous sections using the tabs at the top. Finally there was a confirm popup. In the final version, we have a confirm page instead of a popup, because that actually serves a useful purpose, whereas most users when presented with a popup click it without reading it. This was done after feedback from user testing, when users didn't pay attention to the popup.

Another thing we added in the final design was a way to exit the event creation process without clicking the back button. We did this by adding a clearly visible logo, which when clicked upon takes the user to the home screen. We also did this after user testing.

Implementation

We had an extremely decoupled design which was our intention from the start. Our content is completely separated from the style. So we have for each page separate HTML files which just showcase content and CSS files which style the content. This was done really efficiently by putting semantically consistent id attributes and using div wrappers very judiciously so that when we use selectors in CSS, it makes for very readable code. This allows us to efficiently debug and change any style we want. In addition we also use AJAX on the site. When you add a listing it automatically gets added to the page without a refresh. We tried to give the site a pretty web 2.0 interface by using Javascript through the site. We add event listeners and handlers on a lot of elements which then caused more information or filtering of information without a reload.

One of the key decisions we made while implementing the backend was using div layout and not table layout. This was because table does not make semantic sense especially since the data is organised more in a section style rather than table style. Using div also helps make ajax calls less cluttered. Using tables in this case would have totally slowed down the update too because table layout algorithms are much more cumbersome and heavy than div layout algorithms. Also, semantically using Ajax with div is much easier.

Another key decision we made was using the timepicker widget to select time. Ideally we would have liked to implement this widget ourselves and optimise its User Interface to be consistent with UI Heuristics, however it was proving too cumbersome and complicated so we decided to stick to just using the widget instead. The time picker widget poses serious Fitt's Law issues but we thought it was better to use it and implement other features than building the widget on our own.

Evaluation

Method of Testing:

We used a very similar protocol to user test for GR6 as we did in GR3. We took a set of 4 users. 3 undergraduate students of 3 different years, all from different dorms, and a grad student from Tang.

Each user was briefed about the system information.
Each user was asked to complete all 4 tasks one by one.
We did not provide a demo, because we felt the interface wasn't complex enough to require one, and to keep the user completely new to the interface.
Ishaan was the facilitator, while Keshav and Viksit observed.

BRIEFING
We are an online listing site specific to MIT where people can upload listings of the free food events that happen at MIT. Numerous events such as company info sessions , club events, etc offer free food for those who attend. We want to create a platform where this information is available all at one place. In true Web 2.0 like manner, anyone can create a listing for anyone to view. The only caveat and a fair one is that you be an MIT student.

TASKS
We used the same tasks from GR3 with a slight modification to task 3 (because we did not embed calendar functionality into the site) but with a different set of users because we wanted to test how we had fared in our design after taking input from earlier rounds of testing.

1. This was the simplest task.
"You are very hungry and want to eat Mexican/chinese/pizza food immediately. It is 7:45 PM and you are sitting famished in the Stata Center ."

2. " Anne Hunter just sent out an email. Oracle is planning to hold an info-session at 32-123 on

Friday, March 16 at 6PM. They will have free-food from Qdoba . Create a listing for this event and help the cause of free food at MIT “

3. “To cut down on expenses, for the next one month you are experimenting by having Vegetarian food every time its available for free on campus “

TESTING

TASK 1

All four users could get the task accomplished in reasonable amount of time. The cuisines come all selected. Most users played with the checkboxes a couple of times to "get"
the live filtering that was happening. They would then go on to unselect others but mexican/ chinese/pizza. Because the default list of results is sorted begining from current day, and because of the heading "Today", all of them could identify which portion of the results to manually scan depending on the time of the day - 7:45pm => Close to the lower half of the listings.

Additionally, one user played around with the checkbox for tomorrow and today, to just see how it changed the results. 2 users also tried clicking the Search button.

Two of them said the "Search" button was helpful que, others said it was confusing, because it "didnt do anything". That is because filters are applied live via ajax.

TASK 2

All users were able to accomplish the task; some were quicker than others though.

Some common things that were pointed out:

We had a scipt to enforce forward linear behavior through the 4 tabs before a user could chose to navigate to a particular tab out of sequence. 3 of the 4 users disaproved of the choice. They felt too "bound". It also lead to some error conditions:
One user wanted to go back once selecting the cuisine (first tab), but the error kept on forcing him to fill in the next tabs for two times, until he reached the final tab with the submit button. At this point he was allowed to go to the first tab, but was left clearly unhappy.

Another user hit the back button of the browser to accomplish something similar, and landed on the home page instead. (This happens because the form is actually just one page divided into 4 tabular sections.)

Most users asked for a preview of the form before submission. Otherwise they had to manually check all 4 tabs to make sure all information entered was correct.

Because we did not incorporate user accounts, the interface does not allow for delete / edit.

2 of the users said they would have appreciated the feature.

One user asked for a redirect to his listing on the front page, to get visual feedback to know that his listing had been posted.

TASK 3

This was the least well defined of the tasks. This gave us the opportunity to see how a user would “play around” with the website for some bit.

Some users played around with all cuisines but individual days. Other chose all possible times, but one cuisine at a time.
Users wished they could have more flexibility on time filters as well, just like cuisine. Some infact suggested a dedicated vegeterian checkbox filter would indeed be a great asset.
Some suggested a button to reset filters while some others mentioned that a select all/unselect all button would be nice.

Reflection

We wish we had more time to be able to implement all the features that we wanted and got ideas for after the paper prototype and heuristic evaluations.

We wish we thought about browser compatibilaty and scalibility of our code in advance. We could have pruned our designs and at least have made some design decisions based on these aspects. For example, some aspects like layouts were simple to do before hand but became more complicated as the design got more complex. Getting all the forms and divs and boxes to allign correctly was a mess, but if we thought about the code design earlier, I think we would have been better off.

We thought it was surprising how much asking questions and talking to users actually helped in diagnosing UI problems (much more so than UI heuristics). This was partially I think because users were much more likely to complain about the user interface in person and in a fluent conversation than to write every minute detail in a feedback form. Some subtle problems were hard to diagnose in paper prototypes such as affordances, and user testing really helped improve our affordances.

The heuristic evaluations, in our opinion, were the most useless part of the feedback process (but still a little useful). This was largely because much of the feedback was things we were planning to do, but didn;t get to (we thought gr4 was due way too early in the semester). The other reason was I think people might have been trying to get to the 15 comments by finding

rather minute details, instead of giving truly insightful feedback.

The other thing we wish we did was thing about implementation details when paper protyping. Generally, we ran into the problem that if we got feedback on a design, we sometimes ran into trouble implementing it. On the other hand, we couldnt get paper prototype feedback on the designs that we were able to implement.