Danny Chiao, Sherwin Wu, Rebecca Odim, Ibrahim Abdussabur

MIT Event+

A student group event planner for MIT

1 Overview

1.1 Minimum Viable Product Description

In our minimum viable product, we aimed to produce a basic system where student groups can:

1. create group accounts that can create and manage events
2. view a calendar of all events, and create/edit events via calendar.

We integrated this service with the ASA Group Database for authentication. Accounts can only be created if the email address exists in the ASA database as an officer mailing list. Additionally, the officer mailing list must confirm the account via email.

1.2 Final Product Description

The final product implements additional features on the checklist of tasks needed to plan an event at MIT as well as improves upon existing features. We have create a smooth and user-friendly UI for each of the features included in our Minimum Viable Product. Additional features include:

1. Event planning checklist that streamlines the event planning process with an easy to use and customizable checklist, complete with auto-generated suggested checklist items.
2. Integration with Yelp or other food catering services to find and place orders.
3. Upload posters and draft/send emails for publicity.
4. Create and save a budget.
5. MIT Events Calendar
    a. any interest club/group event on MIT Events Calendar will show up in MIT Event+ as well

1.3 Key Design Challenges

1. How to authenticate MIT student groups?
2. How to display all the events in an understandable manner?
3. How can we recommend appropriate restaurants for a certain group's event?

2 Detail

2.1 Data Representation

Differences from Object Model
Omitting calendar object originally present in object model since only have one master calendar. Functionality of displaying events, detecting event collision, etc. will be abstracted to event and timeblock.

Integration with Other Services
Each integration feature will be represented as a separate object that belongs to each event (i.e. yelp catering will be managed by a FoodManager object, for example, etc.)

2.2 Key Design Decisions

How to authenticate the user
There were multiple possible ways to authenticate the user, and this decision influenced what our notion of a user was:

We could choose to use MIT certificates to authenticate individuals and then pull their student groups from their associated mailing lists. We decided against this approach since the overhead of getting MIT certificates working did not give us significant functionality. Further, there was a risk of somehow filtering out MIT mailing lists.

Another option was to use MIT ASA group emails to authenticate groups. A "user" on our website is then a student group and this email authentication process ensures that we have no unwanted users. This simplified our design, solved security risks, and saved us time, so we went with this approach. We do have to, however, repeatedly update our database of ASA student groups as new groups always emerge.

Event Display
Another design challenge was how to display events to a user. We wanted a way for the user to pick a date and time for their event so that it did not conflict with other events happening on that day. One possible design that was rejected was to show events in the way that one sees events on Google Calendar. This design was rejected because we wanted to find a better way to display event's times so that you can easily see all of the times at once. A user on Google Calendar's day view has to scroll down the page to see all of the events. If there are a bunch of events on that day, a user sees many different colors and event names squished on the screen.

We chose to have a month view using the javascript library FullCalendar and to make our own day view. Our day view was based off of the way the website Hipmunk displays flight information. This way it is visually clear whether your event would conflict with another. One downside to our design is that our month view expands a lot when there are many events during a month. Another downside is that there is no way to filter our calendar by group. Student groups often have very different audiences and it would be useful to be able to only see the events your members are most likely to attend.

Restaurant Suggestion
One particular interesting design challenge was how we were going We decided to base our restaurant suggestion feature around a system of tags that describe a certain student group. These tags were taken from what student groups reported themselves in the ASA database, and we prepopulated a newly created user/group

with the tags that were in the ASA database for that group.

We decided to use this tags approach to suggestion as opposed to other suggestion methods for several reasons. One method we considered was to try and do some pseudo-natural language processing on a student group's name to try and infer what type of food they may order. However, the issue with this was that it might be much more work than it is work (it would be hard to parse every single club name and infer a restaurant category off of it), but more importantly, the accuracy of the suggestion cannot be guaranteed. While some clubs are more intuitive, like the Chinese Students Club ordering food from Chinese restaurants, we realized that some clubs, like the Casina Rueda Group, are associated with Hispanic and Caribbean roots, and information like that would be very hard, if not impossible to see just from the group name.

Another approach we considered was allowing a "social" aspect of suggestions, where we would give restaurant suggestions to a group based on what other groups have ordered from the past. This created an issue of specificity for which student groups would care about where other groups have ordered. For more neutral groups this may not matter (i.e. Leadership Training Institute *could* benefit from seeing that the MIT Logs ordered cheap Thai food from this one restaurant), but for more restaurant oriented groups this may break down (i.e. Korean Students Association doesn't care that the Muslim Student Association ordered food from a Middle Eastern place). This level of noise would have been near impossible to filter out, because we don't know which clubs would or wouldn't care about where other clubs ordered from.

Our ultimate decision, we believe, created very good specificity and accuracy. For one, we are using tags that the clubs *themselves* declared about themselves, so we know with a very high certainty that these tags are accurate in describing the club. Further, we cross reference these tags (fuzzily to maximize matching) with Yelp restaurant categories, which in itself provides a very comprehensive list of restaurants. When there is a match (such as on Korean or Vietnamese), we are guaranteed a very specific and accurate restaurant suggestion. Further, we allow the user to create tags themselves, so this system is constantly improving from user input. However, our design still has some flaws. For example, some clubs with multiple tags that match restaurant categories will inevitably prefer one over the other, but our system right now only randomly selects a tag to suggest restaurants. Also, not all clubs reported tags in the ASA database, so some clubs will start with no tags.

3 Critique

Summary from user's perspective

Positive

- Clean and mostly intuitive UI, users know where to go.
- Integration with student group information provides unique functionality
- Each feature is very intuitive to use.
- Flow of the event planning website is very easy to pick up.
- Site is bug-free (from what we could tell)

Negative

- Events are not scalable (once a user has a lot of events, it will get hard to see them all and manage them)
- It is not immediately intuitive how to check off a checklist item
- It would be nice to an "export to excel" feature for the budget
- On the events calendar interface, it would be nice to limit the number of events shown per day
- If you set an event time, and you go back to the calendar, it's hard to see where you planned the event previously
- It would be nice to talk about location when planning an event
- It is not entirely clear what should be put in an event description and there is no obvious incentive to type a description in the first place
- An event should be able to belong to multiple groups

Summary from developer's perspective

Positive

- Thin and DRY controllers
- Routes are RESTful
- Ruby on Rail code very readable and nicely commented
- CSS files are nicely modularized

Negative

- Javascript not DRY and hard to read
- Need more tests
- Third party code should be separated into a vendor folder

Most and least successful decisions

Most successful

- Modular design - each feature is handled by a separate object that is linked to an event, and each feature can stand alone apart from the others. This makes it very intuitive for the user to use all the features, but also allowed for very nice division of work when developing. Further, if one feature goes down, the rest of the features are perfectly functional and fine.

Least successful

- Not allowing for recurring events. This would significantly clean up the UI and make events easier to manage. New inexperienced exec members can then easily pinpoint previous decisions and user flow is very straightforward.

Priorities for improvement

- Allow for recurring events
- Clean up assets code
- Make app more scalable over time
- General UI polish