Danny Chiao, Sherwin Wu, Rebecca Odim, Ibrahim Abdussabur

MIT Event+

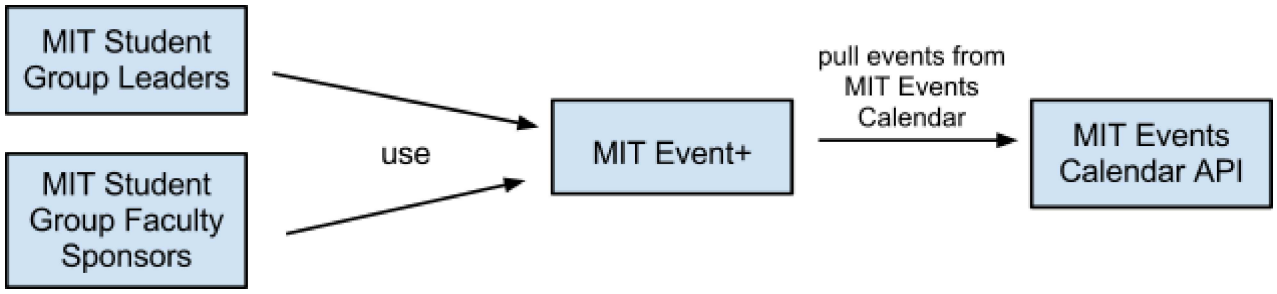
A student group event planner for MIT

1 Overview

1.1 Purpose and Goals

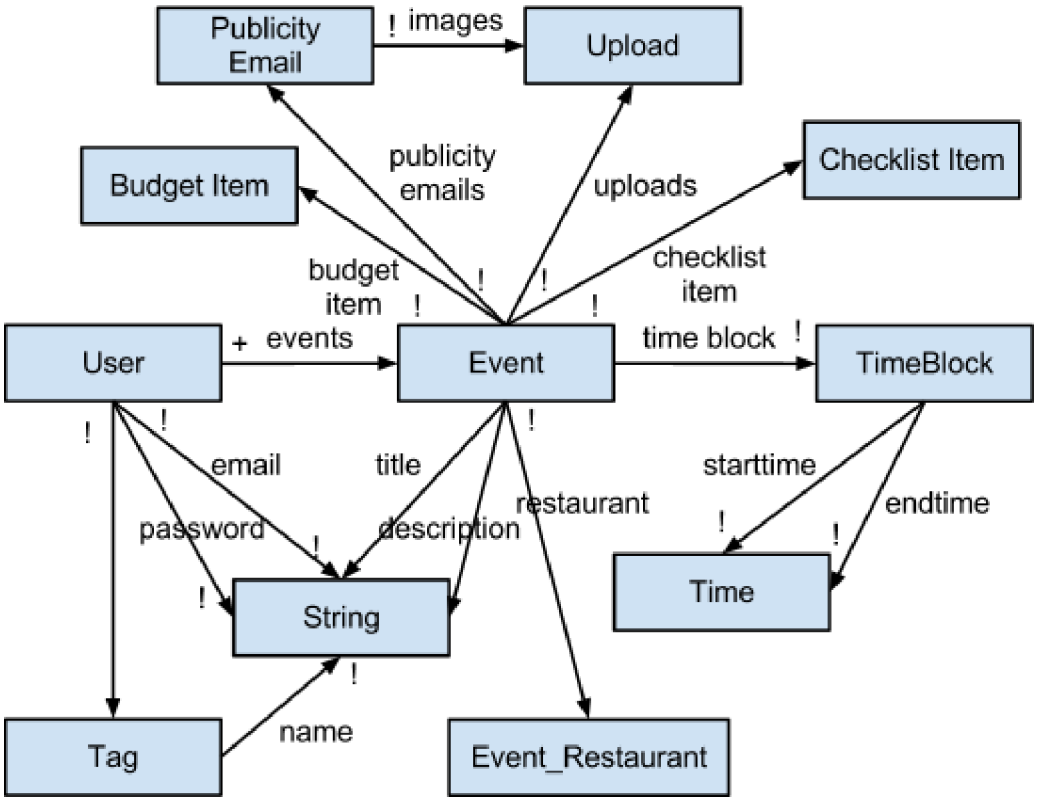
The goal of this website is to streamline event planning at MIT by congregating all aspects into one easy-to-use interface. Existing solutions supply general functionality, but do not address the specific challenges an MIT student group faces when planning events. MIT also provides various services that address a specific aspect of event planning, but often these are poorly documented and fragmented. We aim to create a system that integrates with MIT's services to give a step-by-step event planning service and eliminate the learning curve that comes with planning events at MIT.

1.2 Context Diagram



2 Domain

2.1 Object Model



***** TimeBlock **-** Each event has a notion of a timeblock that can be easily compared to other timeblocks to determine conflicts.

* Event_Restaurant **-** An instance of a Yelp restaurant that is associated with a single event.

2.2 Event Model

FirstTimeUser ::= signUp authorizeAccount*

NormalUser ::= (login (createEvent **|** EditEvent **|** deleteEvent)* logout)

EditEvent ::= (setTime **|** setRestaurant **|** sendPubEmail **|** uploadFile **|** makeChecklistItem **|** toggleChecklistItem **|** editBudget)

Definitions of Non-obvious Events:

- **authorizeAccount:** Authorize each new group by sending an authentication email to the MIT student group email.
- **sendPubEmail**: Sends a publicity email to the group's exec mailing list. In addition, stores the pub email on file to assist in making of future publicity emails.
- **setRestaurant:** Chooses a restaurant to get food catering from. Does not make any physical order.
- **editBudget:** Adds or deletes budget items to create a complete budget for an event.

3 Behavior

3.1 Feature Description

- **Student Group Account Validation:** When a student group registers for our service, they must be registering with the "@mit.edu" mailing list for their club. Further, the email must belong to an ASA recognized student group. Finally, we will send an email to the student group mailing list so they can validate the account.
- **Student Group Account Management:** Anyone in a student group can register their student group, and after being validated, they can login and create events.
- **MIT Event Calendar:** Users will see a calendar displaying events at MIT so that student groups can easily avoid conflicts with other events when planning their own.
- **Event Creation**: Student groups can create an event and add it to the calendar.
- **Event Checklist**: Student groups have a pre-generated checklist of things to do to plan the event and can add items of their own.
- **File Manager:** Student groups can upload files to each event for archiving or management purposes
- **Publicity Emails:** Student groups can send rich text-edited emails to their student group. Previous event publicity emails can be easily ported in to make recurring event email sending quick and easy.
- **Budget Creation:** Student groups can create a budget for each event and manage budget items.
- **Student Group Tagging System:** A student group's tags from the ASA database will automatically be pre-populated when that group creates an account on MIT Event+, and they can add or delete tags for their account. The tags are used for the restaurant suggestion system.
- **Yelp Event Restaurant Selection:** Student groups can choose a restaurant to order food from based on suggestions or they can search themselves. Restaurant selection is powered with the Yelp API.

3.2 Security Concerns

- **Spam Attacks:** A random user could try to create a fake student group and spam our service by either adding a bunch of nonsense events to our calendar or spamming our services (that we will add later). We prevent this type of attack by validating email addresses that you register (through the ASA database list of officer email lists, which we assume will be accurate).
- **Fake Credentials:** If a user is able to register as another student group, they would be able to hijack their account and register events on their behalf. We prevent this by sending a validation email to the registered student group mailing list.
- **Authentication:** Under no circumstances should a nonregistered user/group be able to make events. All features of our service must be done by a logged in account.

3.3 Operations

GET /

Effects: Returns a welcome page

POST /login

Requires: An email and password

Modifies: Session

Effects: Authenticates and logs user in

DELETE /logout

Requires: Group is logged in

Modifies: Session

Effects: Destroys session data for group

POST /event

Requires: User is logged in, event details

Modifies: Event, Calendar

Effects: Creates new event and puts it on the calendar

POST /event/edit

Requires: User is logged in, ID of event, details to edit. User must own event

Modifies: Event

Effects: Modifies content for event.

DELETE /event

Requires: User is logged in, ID of event. User must own event.

Modifies: Event, Calendar

Effects: Destroys an event on the calendar

POST /checklist_item

Requires: User is logged in, ID of event. User must own event.

Modifies: ChecklistItem

Effects: Creates a checklist item bound to an event

DELETE /checklist_item

Requires: User is logged in, ID of event. User must own event.

Modifies: ChecklistItem

Effects: Destroys a checklist item

POST /budget_items

Requires: User is logged in, ID of event. User must own event

Modifies: BudgetItem

Effects: Creates a BudgetItem bound to an event

DELETE /budget_items/:id

Requires: User is logged in, ID of budget item. User must own event

Modifies: BudgetItem

Effects: Destroys a BudgetItem

POST /events/:event_id/uploadFromRedactor

Requires: User is logged in, ID of event, file, file must be an image type, User must own event

Modifies: Upload

Effects: Creates an Upload

POST /events/:event_id/publicity_emails

Requires: User is logged in, ID of event, publicity email, User must own event

Modifies: PublicityEmail

Effects: Creates a PublicityEmail

POST /events/:event_id/uploads

Requires: User is logged in, ID of event, file, User must own event

Modifies: Upload

Effects: Creates an Upload

DELETE /events/:event_id/uploads/:id

Requires: User is logged in, ID of upload and event. User must own event

Modifies: Upload

Effects: Destroys an Upload

GET /events/getevents

Requires: User is logged in, start and end time range.

Effects: Returns json of all events

GET /events/getresources

Requires: User is logged in, start and end time range.

Effects: Returns json for all events with only name and id parameters for timeView implementation

POST /checklist_items

Requires: User is logged in, ID of event, checklist_item text,, User must own event that checklist item belongs to

Modifies: Checklist_Item

Effects: Creates a Checklist Item

DELETE /checklist_items/:checklist_item_id

Requires: User is logged in, ID of checklist_item,, User must own event that checklist item belongs to

Modifies: Checklist_Item

Effects: Deletes a certain checklist item

POST /checklist_item/edit_text

Requires: User is logged in, ID of checklist_item, User must own event that checklist item belongs to, new text of checklist item

Modifies: Checklist_Item

Effects: Updates the text of a checklist item to what was just passed in.

POST /checklist_items/:checklist_item_id/toggle_checked

Requires: User is logged in, ID of checklist item, User must own event that checklist item belongs to

Modifies: Checklist_Item

Effects: Toggles the checked boolean for a checklist item, changing it from true to false or vice versa.

POST /tags

Requires: User is logged in, name of new tag

Modifies: Tag

Effects: Creates a new tag for the currently logged in user with a name of what is passed in as name.

DELETE /tags/:tag_id

Requires: User is logged in, ID of tag

Modifies: Tag

Effects: Deletes a certain tag with the given id.

POST /checklist_items/:checklist_item_id/toggle_checked

Requires: User is logged in, ID of checklist item, User must own event that checklist item belongs to

Modifies: Checklist_Item

Effects: Toggles the checked boolean for a checklist item, changing it from true to false or vice versa.

GET /events/:event_id/yelp

Requires: User is logged in, ID of event

Effects: Returns the yelp restaurant suggestion/selection page for the event with the passed in id

POST /events/:event_id/yelp_search

Requires: User is logged in, ID of event, search term and search zip

Modifies: None

Effects: Returns a hash of Yelp API search results with the given search term and zip.

POST /events/:event_id/select_restaurant

Requires: User is logged in, ID of event, yelp_id, yelp_name, yelp_url, yelp_phone (all yelp fields are information on the seleted restaurant

Modifies: Event_Restaurant

Effects: Creates an Event_Restaurant object belonging to the given event and stores some information about the restaurant

POST /event/:event_id/deselect_restaurant

Requires: User is logged in, ID of event, yelp_id

Modifies: Event_Restaurant

Effects: Deletes an Event_Restaurant object belonging to the given event with given yelp_id.

DELETE /event/:event_id/clear_restaurants

Requires: User is logged in, ID of event

Modifies: Event_Restaurant

Effects: Deletes all Event_Restaurant objects belonging to a given event.

3.4 User Interface

For user interface screenshots and a flow diagram (ui_flow_diagram.jpg) see the /deliverables/eventplus_ui/ directory in our GitHub repository.