**Group Members:** Angelica Casuela, Terri Gan, Nica Medrano

**Delegation of Tasks:**

* Methods aside from sign-in – Nica
* Sign in (entities, method, token) – Angelica
* Entities + Repository + Initializer – Terri

**Internal Deadline: 20 Nov, 11AM - 1PM**

**Proposed system:**

The proposed project of the students is to create an event management system. The main goal of the system is to track when the events of different student organizations in the Ateneo happen in order to notify the Ateneo community about these events. Aside from checking the events on a specific day, the system could also be used to check the venues taken and if the event scheduled if open to all or not.

Possible people who can benefit:

* Office of Student Activities
* Student Organizers
* Ateneans in general

Possible Outputs:

* Events in a month
* Details of one event
* Events of one org in a semester (arranged by date and time)
* Events happened in a specific venue (arranged by date and time)

**Requirements:**

Descriptions

CONTROLLER

The controller can currently add an event along with its details such as the event name, venue, date, time, name of the org involved, event date, description of the event, kind of event, and lastly if it’s open to all or just members only. The controller once triggered returns a confirmation statement of event added.

The controller also enables the user to query the events in a month.

Here are the proposed plans for controller:

* The controller will have deleteEvent and ~~updateEvent~~, which enables the user to edit the values for one, more, or all fields in the details of the event.
* The controller will have viewEvent, which enables the user to view a single event and its details once the user inputs its event name
* The controller will have venueEvents, which enables the user to view the events happening in the inputted venue name over the span of time (either a month or semester)
* The controller will have orgEvents, which enables the user to view the events of the inputted organization acronym for a span of time (either a month or a semester)
* The controller will have login, which enables a user to login to the system before anything else

COMPONENTS

The component named “EventManager” has a method newEvent, which will get the details from the controller inputted by the user and then save in the repository autowired to this specific component.

The proposed plans for the components are as follows:

* Two other components we can add are VenueInitializer and OrgInitializer to have the initial database for both the Venue entity and the Org entity.
* The EventManager component will have methods that can query the inputted the event and either delete or update the details about it
* An additional Login component will justify the authentication of the user

ENTITIES AND REPOSITORIES

The entity involved is the Event Entity which has the following details:

* Event\_id
* Event\_name
* Venue
* Org\_name
* Open to all?
* Category/ Type

All of which have getters and setters set.

In addition, the EventRepository is connected with the Event Entity with a method findBydateMonth to yield the list of events in an inputted month, which is handled by the controller.

The proposed plan for the Entities and Repositories are as follows:

* Suppose we have a database of users who will interact with the system, then we can set a User Entity with the following columns:
  + user\_id
  + user\_name
  + user\_password
* Along with the user entity, we will also have a Token Entity with the following columns:
  + token\_id
  + token\_code
  + user\_id (as foreign key)
* Suppose we have a database of venues around Ateneo, then we can set a Venue entity with the following columns:
  + venue\_id,
  + venue\_name,
  + venue\_buildingName,
  + venue\_roomNumber
  + venue\_hasAircon (where it is a boolean of Yes and No if aircon exists in the venue)
* Suppose we have a database of organizations in Ateneo, then we can set a Ateneo\_Org entity with the following columns:
  + Org\_id
  + Org\_name
  + Org\_acronym
  + Org\_accreditation (where it is a boolean of Yes and No if its accredited or emerging)
* We will have getter/setters for these new entities
* We will create UserRepository, TokenRepository, VenueRepository, and AteneoOrgRepository for the newly made entities
  + UserRepository will have methods that will verify the authentication of the user
  + TokenRepository will have findByTokenCode to verify the access of the user in utilizing the system
  + VenueRepository will have findByVenueName for the controller to return the list of events held in that venue for a specific time span
  + AteneoOrgRepository will have findByOrgAcronym for the controller to return the list of events the inputted org will hold for a specific time span
* We will then align the Event entity with the necessary foreign keys to connect the venue entity and ateneo\_org entity (such as changing the org\_name from Event entity into org\_id and the like)
  + The EventRepository will have findByEventName for the controller to return the details of the inputted event name