UPDate

(An Event Sharing Android App)

Analysis Model

Submitted to:

Asst. Prof. Ma. Rowena C. Solamo
Faculty Member
Department of Computer Science
College of Engineering
University of the Philippines, Diliman

Submitted by:
Bacister, Camille Grace
Lenon, Mikaela Jun
Ubias, Zachary James

In partial fulfillment of Academic Requirements
for the course
CS 191 Software Engineering I
of the
1st Semester, AY 2015-2016

Revision Control

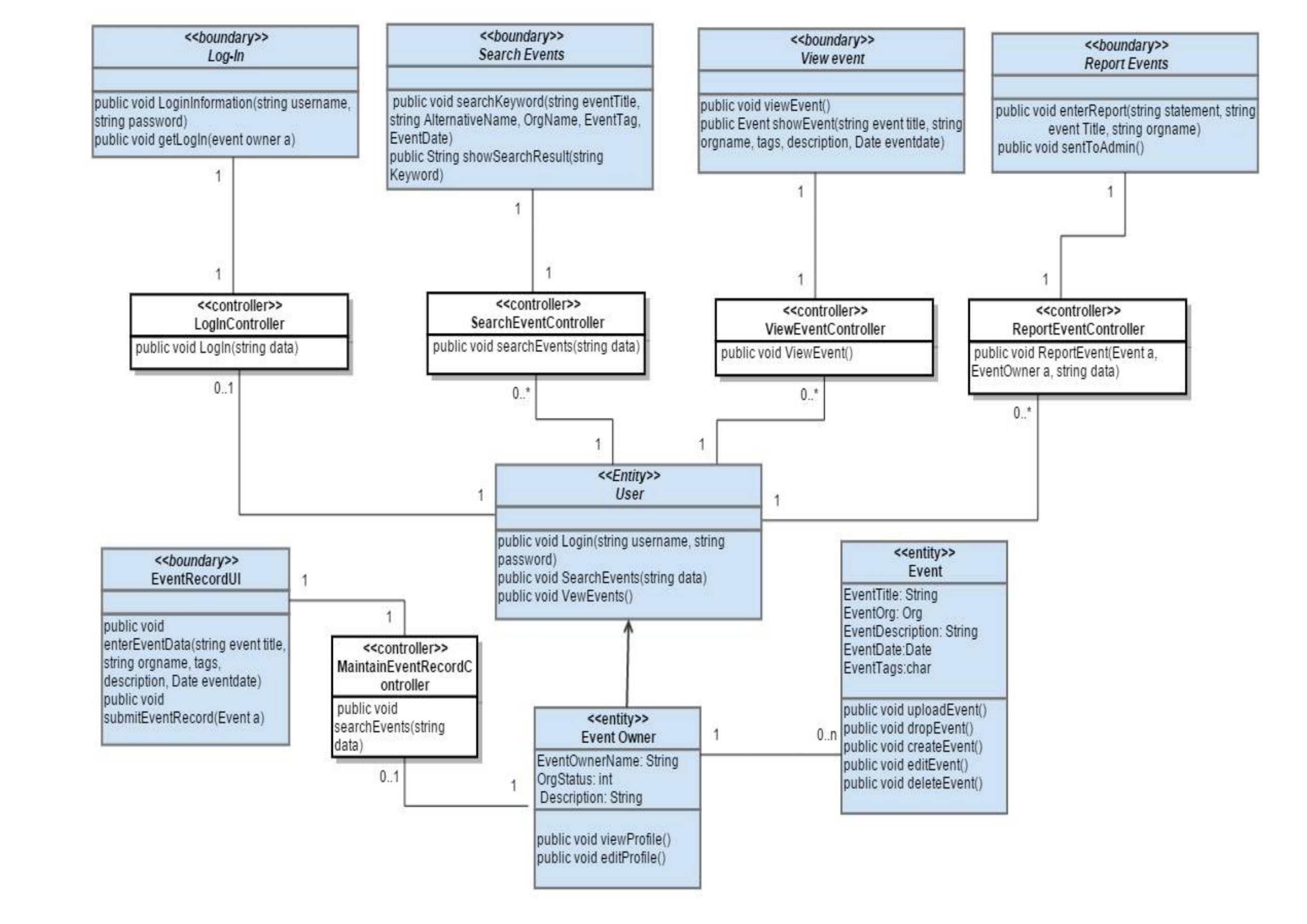
History Revision:

Revision Date	***Person Responsible***	***Version Number***	***Modification***
10/27/15	Mika Lenon	1.0	Initial Document; Boundaries
10/27/15	Mika Lenon	1.1	Class Diagram
10/27/15	Camille Bacister	1.2	Control Class
10/28/15	Camille Bacister	1.3	Entity Class
10/29/15	Mika Lenon	1.4	Updated Description

System Name: Post and Search Event System

Description: The system divided the classes of the project into three different sub-classes: boundaries, controllers, and entities. It clearly shows what interaction will happen or what screen will appear (boundaries) with a specific action (control), and which data (entities) are used or altered in doing so.
This simply enumerates the functions of the Event Sharing application to easily show to the client the relationship and interaction of each class to the other classes.

Class Diagram:



<<boundary>>
Log-In
public void LoginInformation(string username, string password)
public void getLogIn(event owner a)
<<boundary>>
Search Events
public void searchKeyword(string eventTitle, string AlternativeName, OrgName, EventTag, EventDate)
public String showSearchResult(string Keyword)
<<boundary>>
View event
public void viewEvent()
public Event showEvent(string event title, string orgname, tags, description, Date eventdate)
<<boundary>>
Report Events
public void enterReport(string statement, string event Title, string orgname)
public void sentToAdmin()
<<controller>>
LogInController
public void LogIn(string data)
<<controller>>
SearchEventController
public void searchEvents(string data)
<<controller>>
ViewEventController
public void ViewEvent()
<<controller>>
ReportEventController
public void ReportEvent(Event a, EventOwner a, string data)
<<Entity>>
User
public void Login(string username, string password)
public void SearchEvents(string data)
public void VewEvents()
<<boundary>>
EventRecordUI
public void enterEventData(string event title, string orgname, tags, description, Date eventdate)
public void submitEventRecord(Event a)
<<controller>>
MaintainEventRecordController
public void searchEvents(string data)
<<entity>>
Event Owner
EventOwnerName: String
OrgStatus: int
Description: String
public void viewProfile()
public void editProfile()
<<entity>>
Event
EventTitle: String
EventOrg: Org
EventDescription: String
EventDate:Date
EventTags:char
public void uploadEvent()
public void dropEvent()
public void createEvent()
public void editEvent()
public void deleteEvent()
1
1
1
1
1
1
1
1
0..1
0..*
0..*
0..*
1
1
1
1
1
1
0..1
1
1
0..n

Boundary Classes:

Class Name	**Description**
Search Events UI	The user can search different events by typing in the event title, organization, date, or tags. The system will then show a list of events that satisfies the data the user entered.
View Events UI	After showing a list of events, the user can choose any event from those to show its details: Event organizer (organizations), short description, and Event date.
Report Events UI	In case there will be a suspicious event, the user can report it to the admin by filling up a form –which includes information like Event Title, Event Organization, and the statement or report.
Event Record UI	Event maintenance is done by the Event Owner. He can upload and add, modify or delete an event. This boundary class basically make changes to an Event A.
Log-In UI	To Log-in, the system will ask the username and password as inputs from the User.

Control Classes:

Class Name	**Description**
Log-In Controller	Log-In controller regulates the interaction between the User and the Log-In UI. Say the Log-In UI asks for the username but the User inputs an invalid username, then the system will check the User entity for data and then the Log-In Controller will direct you to Log-In UI which will prompt an “Log-In Failed. Invalid username.” However, if you input a valid username and the system recognizes it, the Log-In Controller will direct you to Log-In UI which will prompt “Log-In Successful!”
Search Event Controller	Search Event Controller links User and Search Event UI. For example a user wants to search an event, then the system willc check data from the User and prompt possible searches on Search Event UI. If there none, then it will prompt “No search found” on the Search Event UI.
View Event Controller	View Event Controller connects the View Event boundary to the User Entity. Once the user opt to check some events, the system will check the data from the User entity then direct you to the View Event UI where the data will be displayed. If there are no data on the User Entity, “No details about the event” will prompt on the View Event UI, however if there are any data, it will just display the details on the View Event UI.
Report Event Controller	Report Event Controller bridge Report Events and the User Entity. Once a user send a Report Event, the system will add a report data on the User Entity and therefore notifying the event owners.
Maintain Event Record	Maintain Event Record maintains the interaction between the Event Record UI and the Event Owner. Event owners can edit their events through the Event Record UI then the system will send the updates of the data on the Event Owner Entity.

Entity Classes:

Class Name	Description
User	User is the parent of the Event Owner. The User can use Search Event, View Event, Report Event and Log-in.
Event Owner	Event Owner is the child of User, thus It inherits attributes from the user entity. Aside from the Search Event, View Event, Report Event and Log-In, event owners are allowed to Upload, Delete and Manage/Modify the events they created.
Events	Events is the related to Event owner by association, meaning an Event Owner “has an” Event. However, an event owner can have 0 to many number of events but an event can only be managed by an Event Owner.