Software Design Document

For

gatorEvents

Version 1.0
September 18, 2015

IC3

Prepared by:

Gagan Sharma
Vineet Kaushik

Shubham Gupta
Chang Long

Jiayong Li
Zhaohe Xu

Table of Contents

Revision History

Name	Date	Reason for Changes	Version

1.0 Introduction

This Software Design Document covers a range of issues related to gatorEvents and offers an extensive description of the architecture and design of all the components of the system.

With the help of this document, developers and the project manager/scrum master can gain an in-depth understanding of the technical underpinnings as well as the overall operation of the entire system.

1.1 Goals and objectives

The objective of gatorEvents is to provide the entire UF community a comprehensive yet easy to navigate catalog of events that happen around the University of Florida.

Presently, information about events happening around the campus is distributed over too many channels, many of which can be hard to discover for new and sometimes even older members of the community. It can be just as hard to stay updated about events. This is the problem that gatorEvents will redress.

- Therefore, the goals of the project can be summarized into delivering the following features and functionalities:
- The ability to browse events through a simple, intuitive yet attractive interface.
- A 'Favorites' feature for collecting all event categories of interest to the user under one banner.
- The ability to RSVP for an event.
- A 'My events' feature to view all the events one is attending.
- Receive reminders about events one has signed up for and notifications in case of a change of venue or time.
- Search for events using a search button.

1.2 Statement of scope

A description of the software is presented. Major inputs, processing functionality and outputs are described without regards to implementation detail. Rank the major processing functionality from the developer's point of view. Use a simple ranking system such as: essential, desirable and future requirements. This should represent what you think your team can accomplish in the time frame of a semester. The essential requirements, you are sure you can complete. The desirable requirements you hope to complete, but are not sure about. The future requirements, you have strong doubts about. Strive to balance the desires of your client with the reality of the time it takes to develop a SW product.

gatorEvents comprises two applications:

1. A web application used by event organizers to create and post events. Organizers will also be able to view past events on this application.
2. An Android application for users (students/staff/faculty) to look for events that they would be interested in attending.

The above are the umbrellas for various functionalities that will be offered to users through gatorEvents. The underlying functionalities of the system have been categorized into two groups:

1. Essential Requirements
2. Desirable Requirement

Essential Requirements

1. **User Registration (Web and Android)**
   a. Android app users must register with their ufl.edu IDs.
   b. Organizers must also register themselves on the web back-end using their ufl.edu ID if applicable.
   c. Both the systems shall require a verification code, which will be sent to the user's email.

2. **Categories (Android)**
   a. This feature presents events to users in a fashion that is logical and not overwhelming.
   b. Three to four broad categories such as Sports, Recreation, Culture, Academics which branch out into sub-categories e.g. Football, Basketball, Gator Night etc.

3. **Post Events (Web)**
   a. A functionality that helps organizers spread the message about new events through the UF community.
   b. Once logged in organizers can access this functionality through the "Add Event" button and fill relevant details in an online form.
   c. The details are stored in a database when the form is submitted. This data is then made available to app users in a clearly readable format.

4. **Favorites (Android)**
   a. Many times users will not want to navigate through multiple screens to access the few categories/sub-categories they are interested in. To avoid such repetition of effort, they can add their favorite categories/sub-categories to the "Favorites" section.
   b. It will help prevent an information overload on the user who is interested only in a few activities thereby improving user experience.
   c. This will not only help users to improve their use of the app, it will also help in creating a more meaningful news feed for upcoming events.

5. **Edit/Delete Events (Web)**
   a. Organizers will be provided the functionality to change details of the events that have already been posted to the app.
   b. Similarly they will also be able to delete an event in case it is cancelled.
   c. Changes / Deletions will be updated to the android app along with the appropriate information.

6. **Notifications (Android)**
   a. This allows the organizers to inform the android app users about changes made to an event in a reliable way.
   b. Also serves as a reminder for events that are around the corner, and the user may have forgotten.
   c. A log of previous notifications/messages is also made available to the user.

<u>Desirable Requirements</u>

1. **Calendar (Android)**
   a. A personal calendar that allows user

2. **MyEvents (Android)**
   a. Users will want an easier, direct route to the events they have signed up for.
   b. This feature will serve as the list of the aforementioned events. Any events the user RSVPs yes for, will be added to this section, which is accessible through the main menu placed on the header navigation bar.

3. **News Feed (Android)**
   a. "Latest Events" will be featured on the home screen of the application in a news feed like structure.
   b. Users can scroll down to view if any interesting events have been posted recently by simply reading the news feed instead of navigating categories and subcategories.

1.3 Software context

The business model for the project entails offering the app for free to students as well as organizers in the initial stages. Once the number of users of the app reaches a threshold, the project will be introduced in other colleges and universities.

Once this milestone has been achieved, the team plans to approach investors for funding. The cost of serving multiple college communities and organizers will be substantial. Funds would be necessary to buy servers to handle the growing user base. Development cost would remain steady as the team has sufficient members to maintain the system and manage future releases.

While in the beginning there will be no charges for users of the android app as well as the web back-end, eventually organizers holding commercial events will be charged for publicizing their events through the application. This cost may be shared by the users in that they will be shown on-screen ads on their devices. But this will happen only if we are forced to reduce the fee charged to organizers in order to maintain the attractiveness of this product

1.4 Major constraints

There are some sticking points that could adversely impact the project. Those are as follows

- The team has limited experience working with the technologies required to build the system e.g. Android, GitHub, web development. The team will have to invest extra time to learn these technologies and apply in them in a meaningful manner.
- Finding participants (mainly organizers) and coordinating with them to test the viability of the project could prove to be an uphill task. Some of them may not be willing to change their current practices with regard to publicizing their events.
- Time is also a factor worth considering. The team has only a semester to create this project and may not be able to execute all features/functionalities.

2.0 Data design

2.1 Internal software data structure

The architectural pattern being implemented is the Model-View-Controller paradigm with two views, where one of them is the Android app and the other is the web application to be used by organizers.

The internal data structure can therefore be two parts also, specifically, two client - server groups with the same server providing for both clients.

The main data storage unit will be a MySQL database. This is where most of the data of the project with regard to events will be stored.   The Android app will also use a SQLite database to leverage the storage benefits offered by the device. In addition to this, the app will also make use of Shared Preferences to store some key pair values. More details about the MySQL database is provided in further sections of this document.

As most members of the development team are familiar with JSON, it will be the interchange format that we shall use to communicate with the MySQL database.

2.2 Global data structure

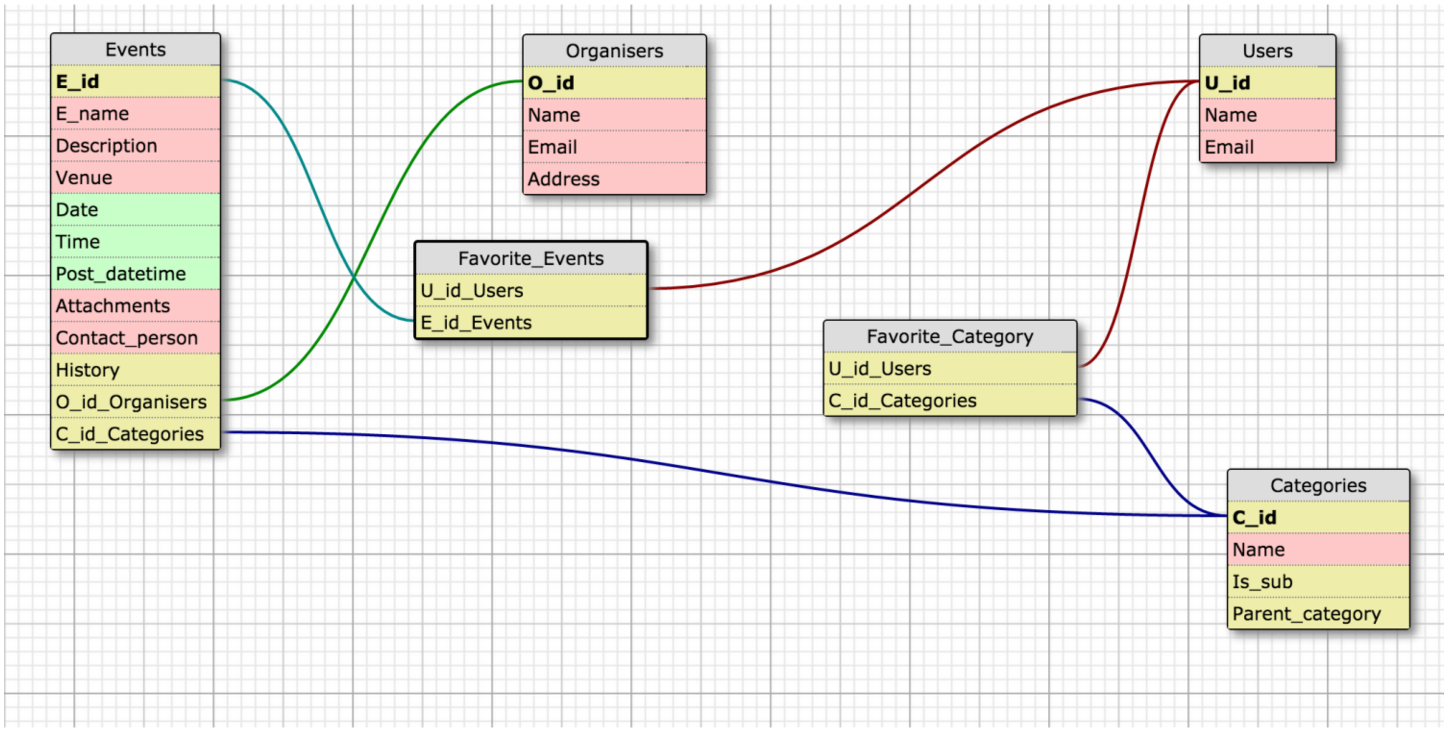
The major part of the project relies on a MySQL database for accessing and storing persistent data. The Android app will be using a SQLite database which will be synchronized with the MySQL database at regular intervals or when a new session is created.

2.3 Temporary data structure

User sessions will be stored in the form of authorization tokens in Shared Preferences. These tokens will be valid for 3 months after which the user will be re-prompted to enter their log-in details.

2.4 Database description

Database(s) created as part of the application is (are) described. (Provide enough detail to create the database.



3.0 Architectural and component-level design

A description of the program architecture is presented.

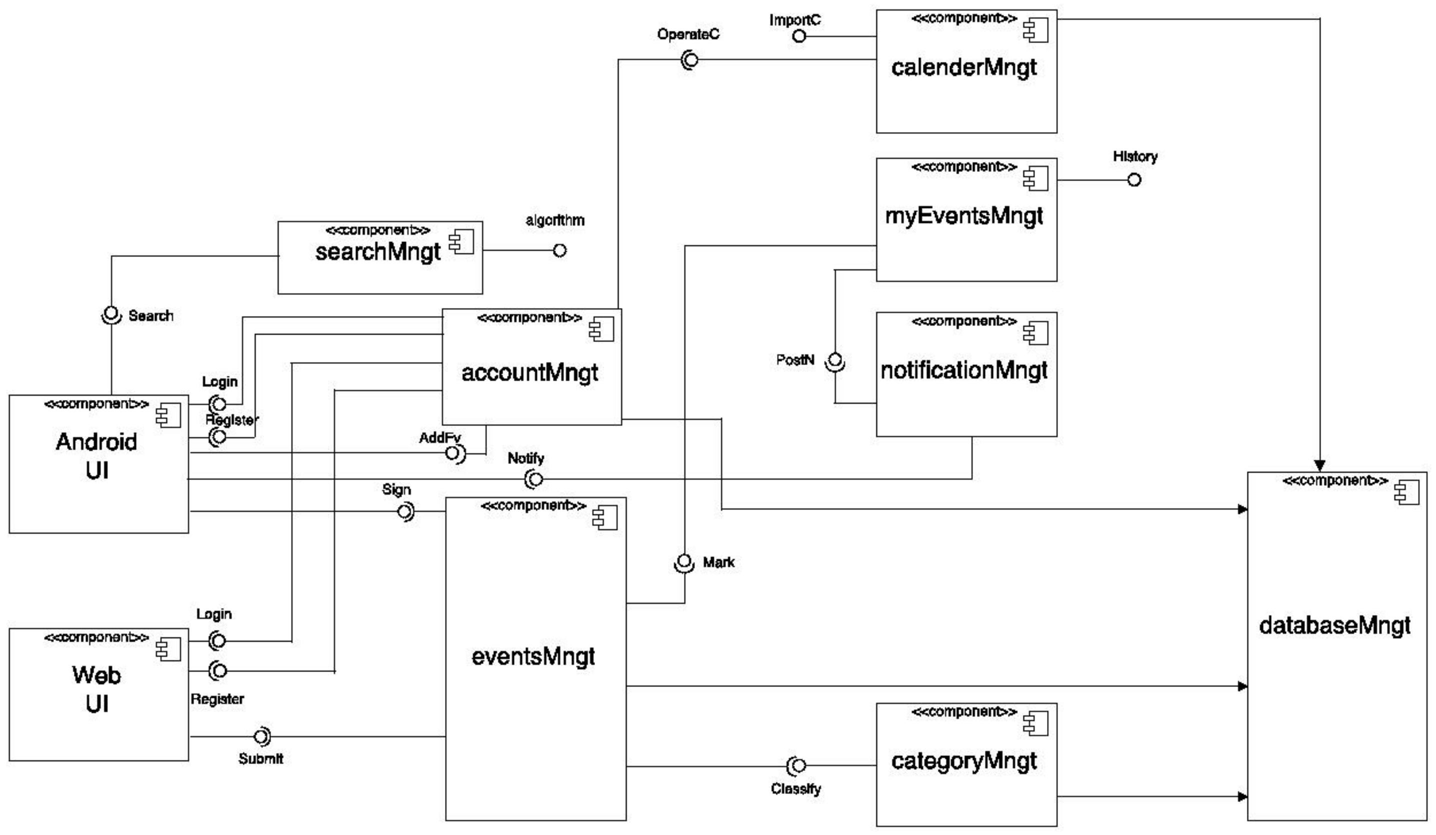
3.1 System Structure

gatorEvents is based on the MVC pattern.

The project is composed of two views, the Android UI and the web UI for the web application. Likewise, there are two separate controllers, one each for the web application and the Android app. The model comprises the MySQL database on the web server and the SQLite database on the device.

For the Android application, the code for the functionality is described in Java and the user interface is coded in XML. The front end for the web application is written HTML 5.0, CSS3 and Javascript (JQuery). The backend will be handled using Flask framework for Python.

3.1.1 Architecture diagram



3.2 AccountManagement Component

3.2.1 Processing narrative (PSPEC) for AccountManagement

The AccountManagement component is responsible for user registration and login. While uses, organizers and administrators have different levels of privilege and even different code, they follow the same pattern and are hence clubbed under a single component. This component also handles encryption of passwords and creates authorization token for the user.

3.2.2 AccountManagement interface description.

AccountManagement takes data it requires from the user through text fields. The output of this component is primarily directed to the database in the form of SQL queries wrapped in HTTP POST requests.

3.2.3 AccountManagement processing detail

This component handles simple user registration and login and therefore does not require any specific algorithms for processing its operation.

3.2.3.1 Restrictions/limitations for AccountManagement

The component does not rely on any other external factors for its operation and hence faces no restrictions as such.

3.2.3.2 Performance issues for AccountManagement

The AccountManagement component does not operate on large data structures. Hence it will rarely face any performance issues. The web application may slow down if the internet connection is slow. The Android application saves user registration details in Shared Preferences of the device and login entries are checked against the same single set of data. Hence AccountManagement is guaranteed to run speedily at all times.

3.2.3.3 Design constraints for AccountManagement

There are no real design constraints with regard to this component.

3.3 EventManagement Component

3.3.1 Processing narrative (PSPEC) for EventsManagement

EventsManagement handles a variety of tasks related to events. EventsManagement provides the user with events in the news feed section as well as in the categories navigation screen. EventsManagement also responsible for posting new events to the database and editing those events in case the organizers wishes to change details of the event or delete it altogether.

3.3.2 EventManagement interface description

Input to the EventsManagement is provided mostly in form of GET requests to the server side python code. The web interface may also provide text input from the post event form to the back end. Users on android do not provide any text field input to this component.
Outputs from the management are mainly in the form of text descriptions and an accompanying image. This is what is provided to the Android App in the MyEvents section and the listview of events listed in the various sub categories of the application.

3.3.3 EventManagement processing detail

The EventsManagement component does not employ any algorithms to carry out its function. As the operation of the component is limited to fetching, writing and updating the database and no heavy computational operations are to be optimized, there is no need for sophisticated algorithms in this component.

3.3.3.1 Restrictions/limitations for EventManagement

The component does not rely on any other external factors for its operation and hence faces no restrictions as such.

3.3.3.2 Performance issues for EventsManagement

While EventsManagement does not operate on large amount of data, it can face performance issues with regard to updating the news feed and the events list under sub-categories. This is because on fetching events on app launch depends on the strength of the internet connection of the device. A slow internet connection may prolong the time required to load these events.

3.3.3.3 Design constraints for AccountManagement

There are no real design constraints with regard to this component.

3.4 NotificationManagement Component

3.4.1 Processing narrative (PSPEC) for NotificationManagement

The NotificationManagement component is responsible for notifying the user about the event they signed for as attending and also about the events coming up under their favorite categories.

3.4.2 NotificationManagement interface description.

NotificationManagement does not have any input. These are generated when there are any changes to the event the user has signed for as attending or the information about the event is updated or the sub category for the event has been modified. The notifications for all these would be send as the notifications on the Android Device.

3.4.3 NotificationManagement processing detail

Notifications are sent to the user with the help of Google Cloud Message's Downstream Messaging feature. The algorithms involved with this feature are abstracted by this feature.

3.4.3.1 Restrictions/limitations for NotificationManagement

The Users should configure their device in such a way that it allows notifications from this application to be sent to the user's device.

3.4.3.2 Performance issues for NotificationManagement

There are no performance issues for notification management.

3.4.3.3 Design constraints for NotificationManagement

There are no real design constraints with regard to this component.

3.5 SearchManagement Component

3.5.1 Processing narrative (PSPEC) for SearchManagement

The SearchManagement component is responsible for searching the events by the keywords or the time frame selected by the user.

3.5.2 SearchManagement interface description.

SearchManagement takes data it requires from the user through text fields or the spinner for time. The output of this component is primarily directed to the database in the form of SQL queries wrapped in HTTP POST requests.

3.5.3 SearchManagement processing detail

This component handles search and requires algorithms like Full text Search to be implemented to get the correct search result in real time.

3.5.3.1 Restrictions/limitations for SearchManagement

The accuracy and ability of the search is defined by how effective the search algorithm has been implemented. The skills of the development team might also put some limitations on its effectiveness.

3.5.3.2 Performance issues for SearchManagement

The performance of the search algorithm will depend on the processing capability of the Android Device.

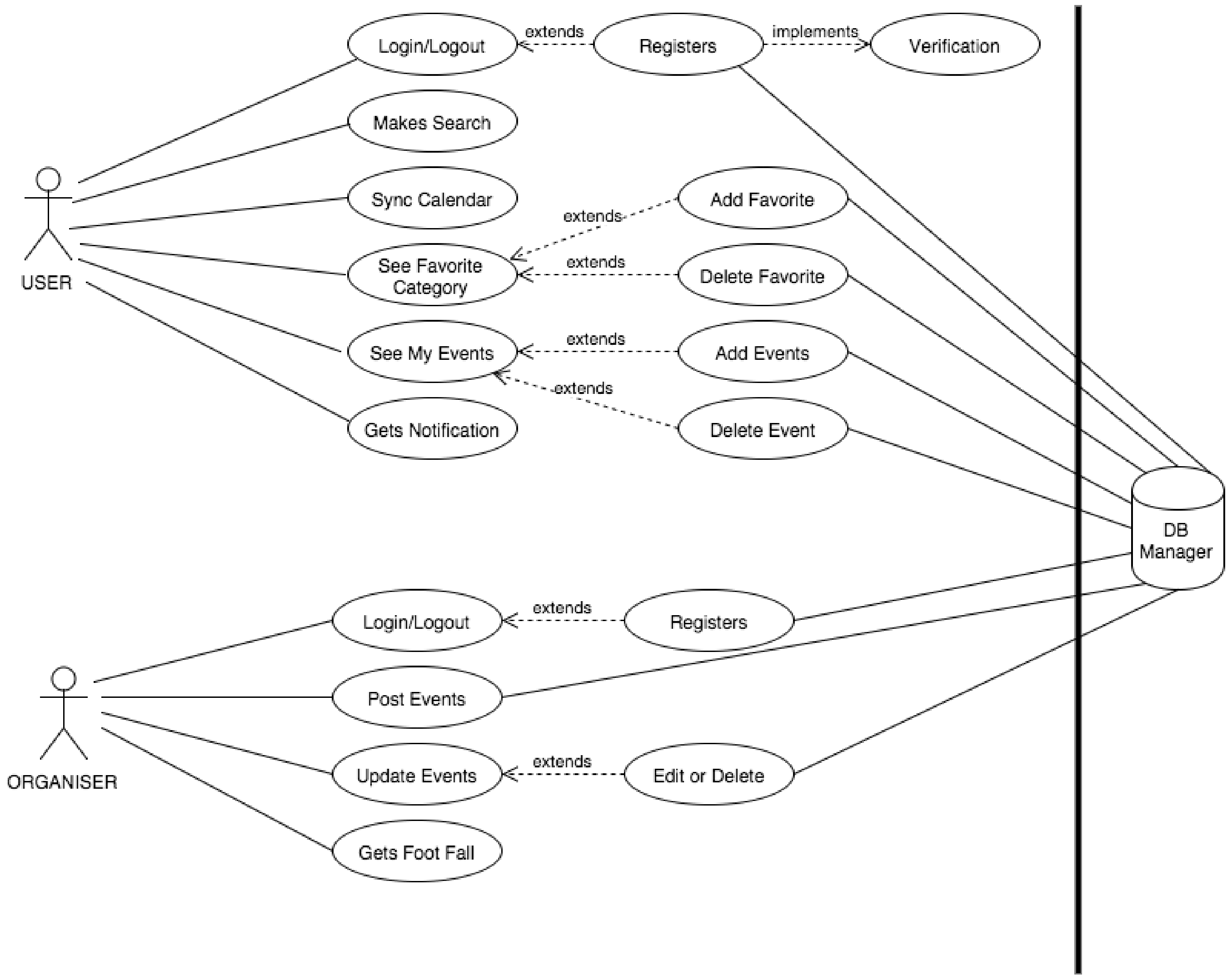
3.5.3.3 Design constraints for SearchManagement

There are no real design constraints with regard to this component.

3.6 Dynamic Behavior for all Components

3.6.1 Interaction Diagrams

<u>3.6.1.1 Use case diagram</u>



This diagram includes all use cases in the system. And here are use case tables to show some detailed information.

Number	A1	
Name	user_login()	
Description	User accesses the system and views the functions and activities currently available for him to participate.	
Priority	5 (1 = lowest priority, 5 = highest priority)	
Preconditions	User chooses to use the apps and has account.	
Postconditions	User accesses the system.	
Primary Actor(s)	User	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Log in" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	User must fill in their information e.g. user name, password.
	3	Click the login button.
	4	Jump to home page.
Extensions	**Step**	**Branching Action**
	3a	System notifies user inputs invalid use name or password.
	3b	System notifies user hasn't register the account and then jump to the registration page.
Open Issues	1	System displays there is no wireless.

Number	A2	
Name	search()	
Description	User searches events and activities or schedule via system.	
Priority	2 (1 = lowest priority, 5 = highest priority)	
Preconditions	User chooses to use the apps.	
Postconditions	User see the search results on the screen.	
Primary Actor(s)	User	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Search" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click the "Search" button.
	3	Input the key words which user wants to query.
	4	Click the "Search" option button.
Extensions	**Step**	**Branching Action**
		No Alternative paths that the use case may take
Open Issues	3	System displays there is no wireless.

Number	A3
Name	calendar()
Description	User syncs and update his or her own schedule or calendar via system.
Priority	2(1 = lowest priority, 5 = highest priority)
Preconditions	User chooses to use the apps.
Postconditions	User update and see the new calendar after sync.
Primary Actor(s)	User
Secondary Actor(s)	User accounts database
Trigger	Click the "Sync Calendar" button on the screen.

Main Scenario	Step	Action
	1	Launch the application from the phone's home screen.
	2	Click the "Sync Calendar" button.
Extensions	**Step**	**Branching Action**
		No Alternative paths that the use case may take
Open Issues	2a	System displays there is no wireless.
	2b	System is unable to link the user's calendar on phone.

Number	A4	
Name	fvt_show()	
Description	User views favorite category.	
Priority	5 (1 = lowest priority, 5 = highest priority)	
Preconditions	User chooses to use the apps.	
Postconditions	User can see the posted and updated favorite category.	
Primary Actor(s)	User	
Secondary Actor(s)	User accounts database	
Trigger	Click the "See Favorite Category" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click the "See Favorite Category" button.
Extensions	**Step**	**Branching Action**
		No Alternative paths that the use case may take
Open Issues	2a	System displays there is no wireless.
	2b	System is unable to find the user's favorite category.

Number	A5	
Name	myEv_show()	
Description	User views the events user chooses and stores in the apps.	
Priority	5 (1 = lowest priority, 5 = highest priority)	
Preconditions	User chooses to use the apps.	
Postconditions	User can see the events user chooses and stores in the system.	
Primary Actor(s)	User	
Secondary Actor(s)	User accounts database	
Trigger	Click the "See My Events" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click the "See My Events" button.
Extensions	**Step**	**Branching Action**

		No Alternative paths that the use case may take
Open Issues	2	System is unable to display the correct saved events.

Number	A6	
Name	notification()	
Description	User gets system notifications on an urgency basis or a change of detail basis. Users can also access the notifications log within the app itself.	
Priority	4(1 = lowest priority, 5 = highest priority)	
Preconditions	User chooses to use the apps.	
Postconditions	User receives the correct notification.	
Primary Actor(s)	User	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Gets Notification" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click notification button to browse the page listing all past notifications.
Extensions	**Step**	**Branching Action**
		No Alternative paths that the use case may take
Open Issues	2a	System is unable to display the past events.
	2b	User is unable to order the desired notification.
	2c	System is unable to receive the user's notification requirements.

Number	B1
Name	user_register()
Description	The user of the android app shall have the ability to register themselves with the system. This is optional for the users of the app.
Priority	3 (1 = lowest priority, 5 = highest priority)
Preconditions	User chooses to use the apps and hasn't get account.
Postconditions	User owns an account in this app.

Primary Actor(s)	User	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Register" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click the register button.
	3	User must fill in their information e.g. Name, UFL ID.
	4	Enter verification code.
	5	Jump to home page.
Extensions	**Step**	**Branching Action**
	3a	System notices user's ID is occupied by other users
	3b	System notices user's password is too weak.
Open Issues	2a	System is unable to let user log in.
	2b	System displays there is no wireless.
Number	B4a	
Name	user_fvt_add()	
Description	This features allow user to add the favorite and the updated information will be shown on the user's mobile application.	
Priority	3 (1 = lowest priority, 5 = highest priority)	
Preconditions	User chooses the function "See Favorite Category".	
Postconditions	User views the updated information after the new events being added.	
Primary Actor(s)	User	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Add Favorite" button on the screen.	
Main Scenario	**Step**	**Action**

	1	Launch the application from the phone's home screen.
	2	Click "See Favorite Category" button
	3	Select your favorite events or activities
	4	Click "Add Favorite" button
Extensions	**Step**	**Branching Action**
	1	User can also choose log in system firstly.
Open Issues	3	System fails to select your favorite.
	4	System fails to add the favorite and save the information.

Number	B4b	
Name	user_fvt_del()	
Description	This features allow user to delete the favorite and the updated information will be shown on the user's mobile application.	
Priority	3 (1 = lowest priority, 5 = highest priority)	
Preconditions	User chooses the function "See Favorite Category".	
Postconditions	User views the updated information after the old events being deleted.	
Primary Actor(s)	User	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Delete Favorite" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click "See Favorite Category" button
	3	Select your old or expired events or activities
	4	Click "Delete Favorite" button
Extensions	**Step**	**Branching Action**
	1	User can also choose log in system firstly.

Open Issues	3	System fails to select your favorite.
	4	System fails to delete the favorite and save the information.

Number	B5a	
Name	evt_add()	
Description	This features allow users to add the event they posted and the updated information will be shown on the user's mobile application.	
Priority	3 (1 = lowest priority, 5 = highest priority)	
Preconditions	User chooses the function "See My Events"	
Postconditions	User views the updated information after the new events being added.	
Primary Actor(s)	User	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Add Events" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click "See My Events" button
	3	Select your interested events or activities
	4	Click "Add Events" button
Extensions	**Step**	**Branching Action**
	1	User can also choose log in system firstly.
Open Issues	3	System fails to select the interested events.
	4	System fails to add the events and save the information.

Number	B5b	
Name	evt_del()	
Description	This features allow users to delete the event they posted and the updated information will be shown on the user's mobile application.	
Priority	3 (1 = lowest priority, 5 = highest priority)	

Preconditions	User chooses the function "See My Events"	
Postconditions	User views the updated information after the old events being deleted.	
Primary Actor(s)	User	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Delete Events" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click "See My Events" button
	3	Select your existed events or activities
	4	Click "Delete Events" button
Extensions	**Step**	**Branching Action**
	1	User can also choose log in system firstly.
Open Issues	3	System fails to select the expired events.
	4	System fails to delete the events and save the information.

Number	C1	
Name	user_verify_email()	
Description	System is required to send the verify email to user.	
Priority	1 (1 = lowest priority, 5 = highest priority)	
Preconditions	User finishes both the user name and password input.	
Postconditions	User will receive the verify email and get a new account.	
Primary Actor(s)	User	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Register" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.

	2	Click “Register” button.
Extensions	**Step**	**Branching Action**
	1	User can also choose log in system firstly.
Open Issues	2	System fails to save user name and password.
	3	System fails to send the verify email.

Below is for organizer.

Number	A7	
Name	org_login()	
Description	Organizer accesses the system and views the functions and activities currently available for him or her to participate.	
Priority	5 (1 = lowest priority, 5 = highest priority)	
Preconditions	Organizer chooses to use the apps and has account.	
Postconditions	Organizer accesses the system.	
Primary Actor(s)	Organizer	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Log in" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Organizer must fill in their information e.g. user name, password.
	3	Click the login button.
	4	Jump to home page.
Extensions	**Step**	**Branching Action**
	3a	System notifies organizer inputs invalid user name or password.
	3b	System notifies organizer hasn't register the account and then jump to the registration page.
Open Issues	1	System displays there is no wireless.

Number	A8
Name	evt_post()
Description	This feature is the main functionality of the web back-end system. Event organizers can post their information through this function.
Priority	5 (1 = lowest priority, 5 = highest priority)
Preconditions	Organizer chooses the function "See My Events"
Postconditions	Organizer views the added event's content.
Primary Actor(s)	Organizer

Secondary Actor(s)	User accounts database	
Trigger	Click the "Post Events" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click "Log in" button
	3	Once logged in, organizers click the "Post Events" button.
	4	Enter details in the template form as provided by the website e.g. title of the event, time and location, description of the activity and attachments (if any).
	5	Submit completed template.
Extensions	**Step**	**Branching Action**
	2	Organizer may be requested to create a new account by system.
Open Issues	2	System fails be logged in.
	4	System fails to edit the events content and save the information.
	5	The content or template is unable to be turned in.

Number	A9	
Name	evt_submit()	
Description	This feature is to update the events in the system, like edit or delete.	
Priority	5 (1 = lowest priority, 5 = highest priority)	
Preconditions	User chooses the function "Update Events"	
Postconditions	User views the updated information after the new events being added or deleted.	
Primary Actor(s)	Organizer	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Update Events" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Organizers click the "Update Events" button.

Extensions	**Step**	**Branching Action**
	1	User can also log in firstly.
Open Issues	2	System fails to update all the events.
Number	A10	
Name	org_lookfor_num()	
Description	This feature is to estimate how many people will participate this event to attract more people come.	
Priority	5 (1 = lowest priority, 5 = highest priority)	
Preconditions	Organizer chooses the function " Gets Foot Fall "	
Postconditions	Organizer views how many people will participate this event and in this event.	
Primary Actor(s)	Organizer	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Gets Foot Fall" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Organizers click the "Gets Foot Fall" button.
Extensions	**Step**	**Branching Action**
	1	Organizer can also log in firstly.
Open Issues	2	System fails to update the number of people in this event.

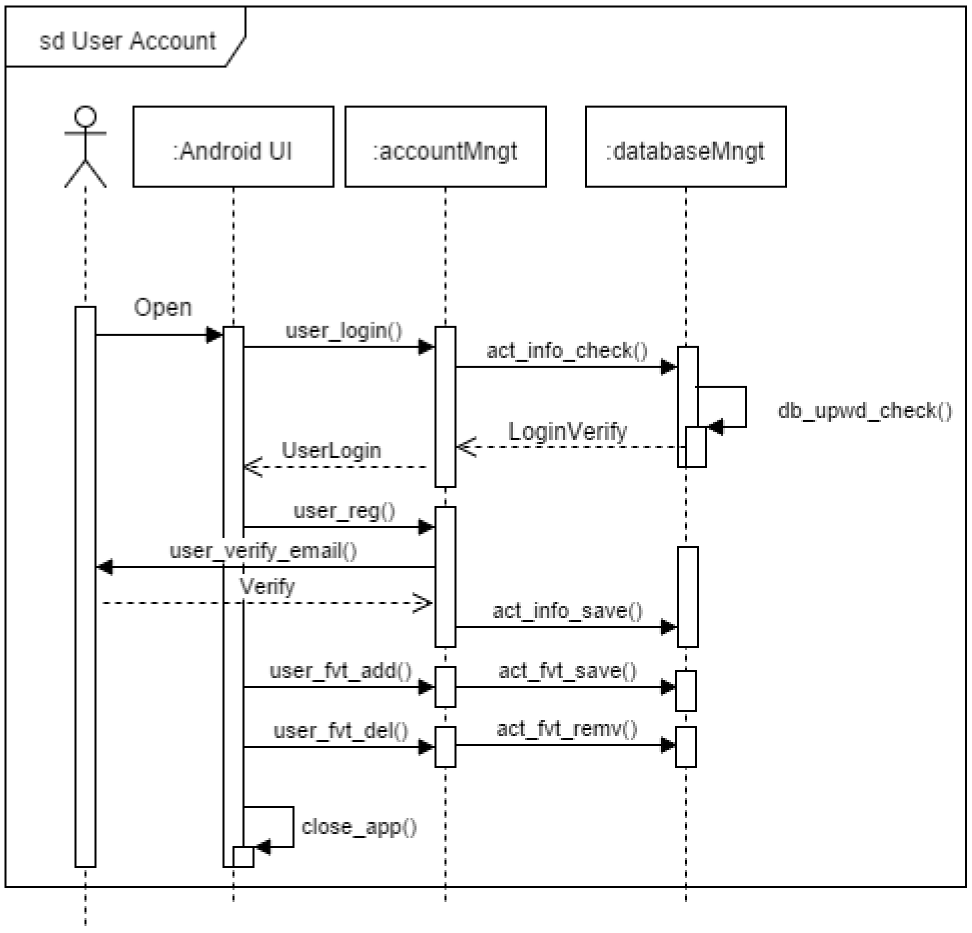
Number	B7
Name	org_reg()
Description	The organizer of the android app shall have the ability to register themselves with the system. This is optional for the users of the app.
Priority	3 (1 = lowest priority, 5 = highest priority)
Preconditions	Organizer chooses to use the apps and hasn't get account.
Postconditions	Organizer owns an account in this app.
Primary Actor(s)	Organizer
Secondary Actor(s)	User accounts database
Trigger	Click the "Register" button on the screen.

Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click the register button.
	3	Organizer must fill in their information e.g. Name, UFL ID.
	4	Enter verification code.
	5	Jump to home page.
Extensions	**Step**	**Branching Action**
	3a	System notices organizer's ID is occupied by other users
	3b	System notices organizer's password is too weak.
Open Issues	2a	System is unable to let organizer log in.
	2b	System displays there is no wireless.

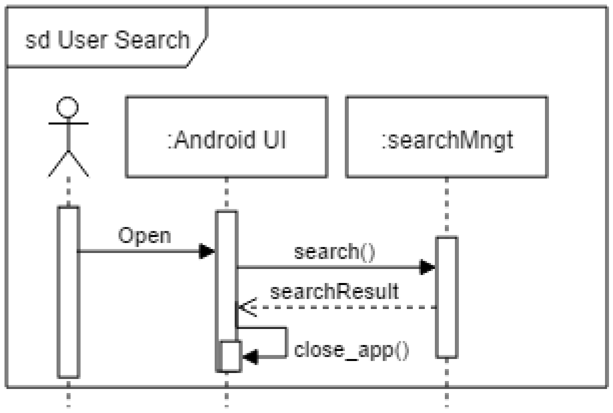
Number	B9a	
Name	org_evt_edit()	
Description	This features allow organizers to edit the event they posted and the updated information will be shown on the user's mobile application.	
Priority	3 (1 = lowest priority, 5 = highest priority)	
Preconditions	Organizers chooses the function "Update Events"	
Postconditions	Organizers views the updated information after the new events being edited.	
Primary Actor(s)	Organizers	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Edit" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click "Update Events" button
	3	Select your interested events or activities

	4	Click "Edit" button
Extensions	**Step**	**Branching Action**
	1	Organizers can also choose log in system firstly.
Open Issues	3	System fails to select the interested events.
	4	System fails to edit the events and save the information.

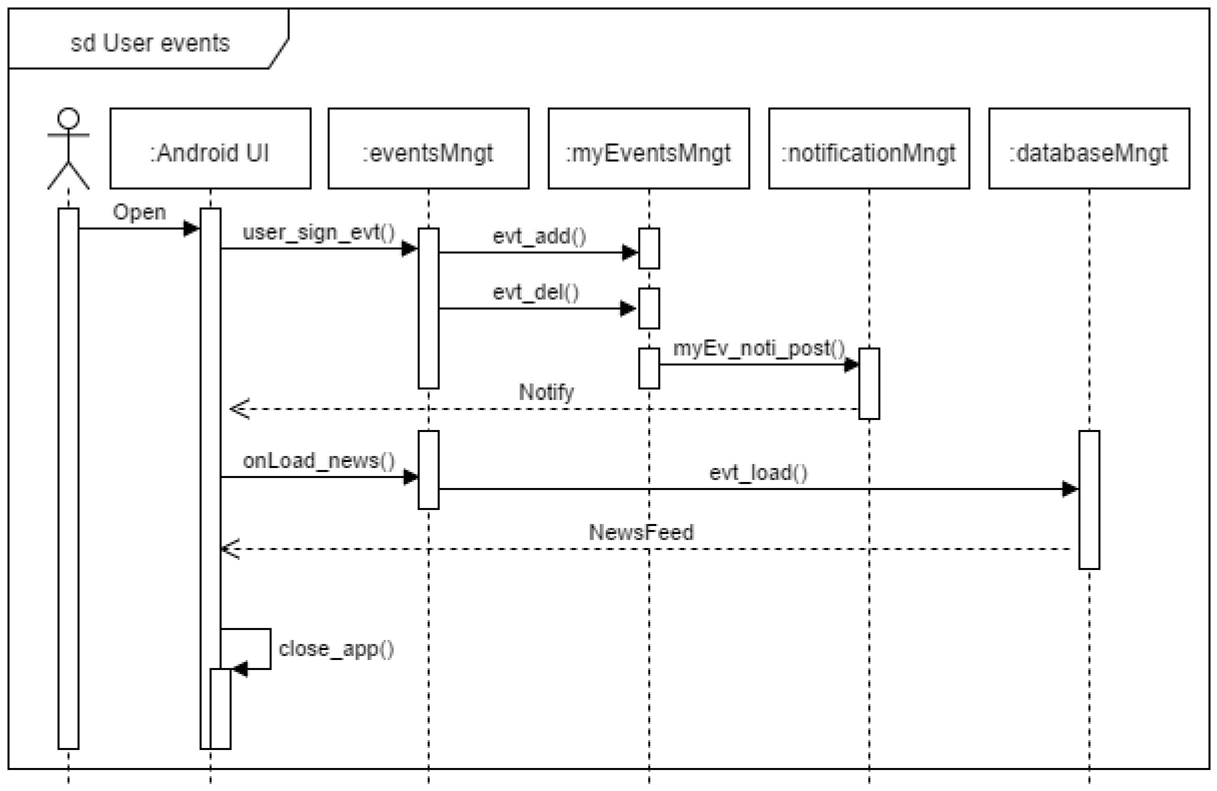
Number	B9b	
Name	org_evt_del()	
Description	This features allow organizers to delete the event they posted and the updated information will be shown on the user's mobile application.	
Priority	3 (1 = lowest priority, 5 = highest priority)	
Preconditions	Organizer chooses the function "Update Events"	
Postconditions	Organizer views the updated information after the old events being deleted.	
Primary Actor(s)	Organizer	
Secondary Actor(s)	User accounts database	
Trigger	Click the "Delete" button on the screen.	
Main Scenario	**Step**	**Action**
	1	Launch the application from the phone's home screen.
	2	Click "Update Events" button
	3	Select your existed events or activities
	4	Click "Delete" button
Extensions	**Step**	**Branching Action**
	1	Organizer can also choose log in system firstly.
Open Issues	3	System fails to select the expired events.
	4	System fails to delete the events and save the information.



This is the sequence diagram for account management, to show how the system executes login, registration, adding favorite and deleting favorite.



This is the sequence diagram for search management to show how does the system do search.



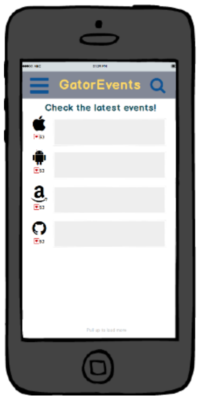
This is the sequence diagram for whole events management -- the most important part of the system.

4.0 User interface design

4.1 Description of the user interface

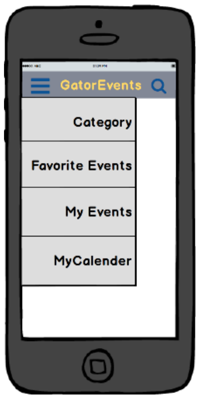
4.1.1 Screen images

4.1.1.1 Android Home Page



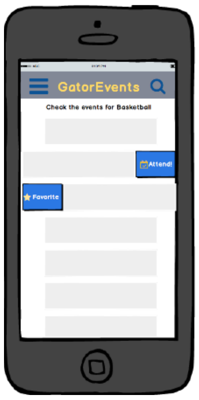
- On the home page, there is a News Feed which gives information about all the latest events posted on the Application by the organizers.
- Any user can use search by clicking the button at top right and enter some key words.
- Additional operations are placed in the menu, which can be accessed by clicking the button at top left.
- Users who wants to sign for events or add some favorite categories should press Login/Register at the bottom of the home page.

4.1.1.2 Android Menu



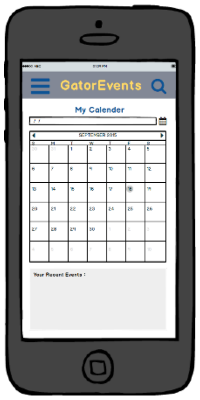
- Click 'Category' to look up all categories the system has and then the user can mark some of them as their favorite.
- Click 'Favorite Events' to add some favorite categories or subcategories, and then they can look for some events of these categories quickly. They can also make changes to their favorite selection by adding new and deleting the previously selected categories.
- Click 'My Events' to see the events signed by the user as attending and they can also sign out from an event they don't want to attend.
- Click 'My Calendar' to show the events the user has signed for in a very decent and informative format. They can also search events by selecting a time frame in which they are free to attend an event.

4.1.1.3Android Events



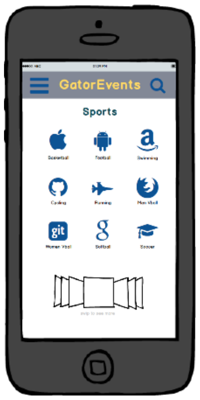
- Slide the bar towards the left to sign in for the event.
- Slide the Favorite bar right to mark this as the favorite event.
- The marked events will show up in the myEvents tab.

4.1.1.4 Android Calendar



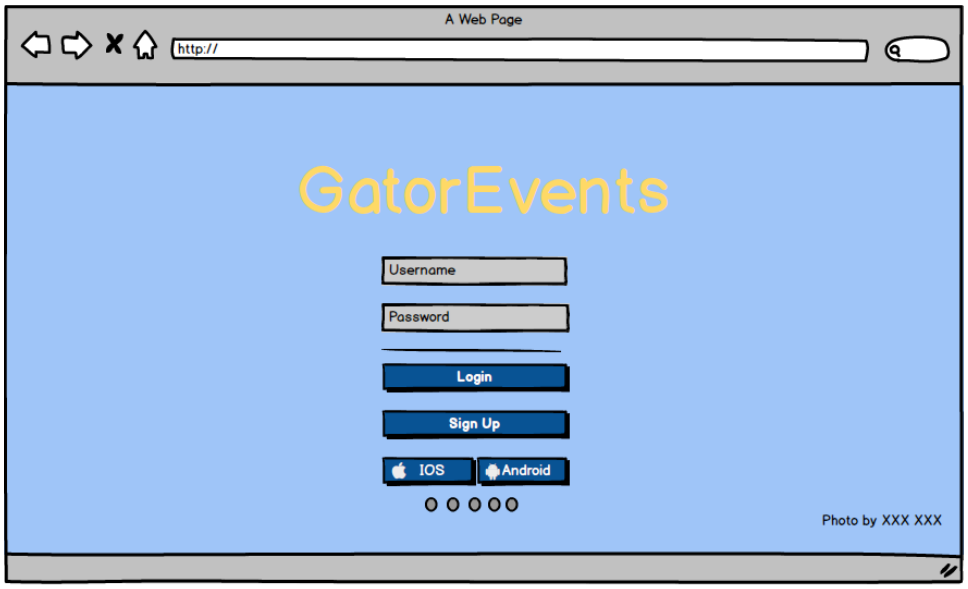
- This will help the user see the events they are attending on a particular date in a very easy and informative manner.
- They just need to select a particular date and they can see the events and timings for the same just below the calendar.

4.1.1.5 Android Categories



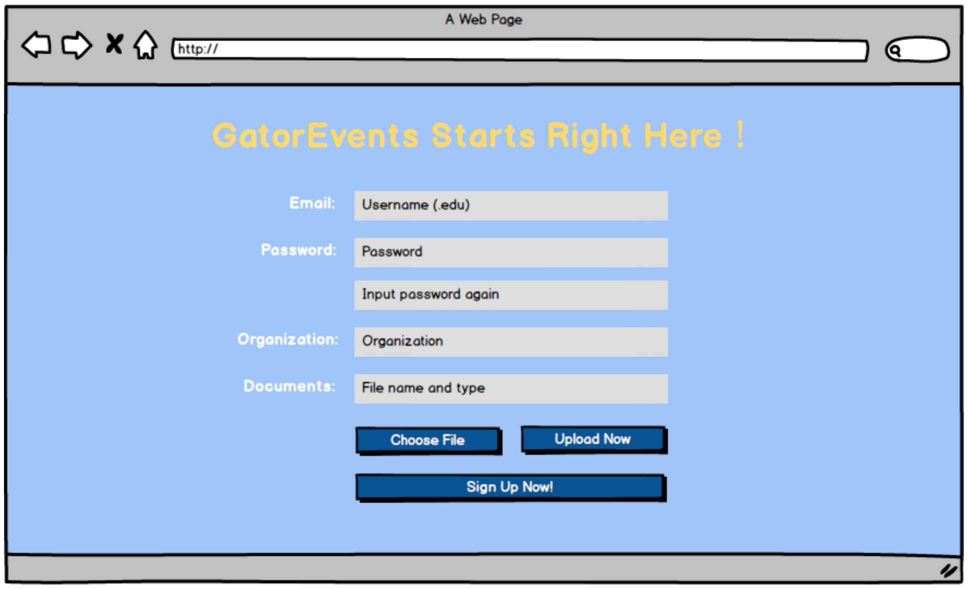
- This will help the user see all the categories the system has in a layout that is very informative and easy for the user to select and proceed further.

4.1.1.6 Web Home page



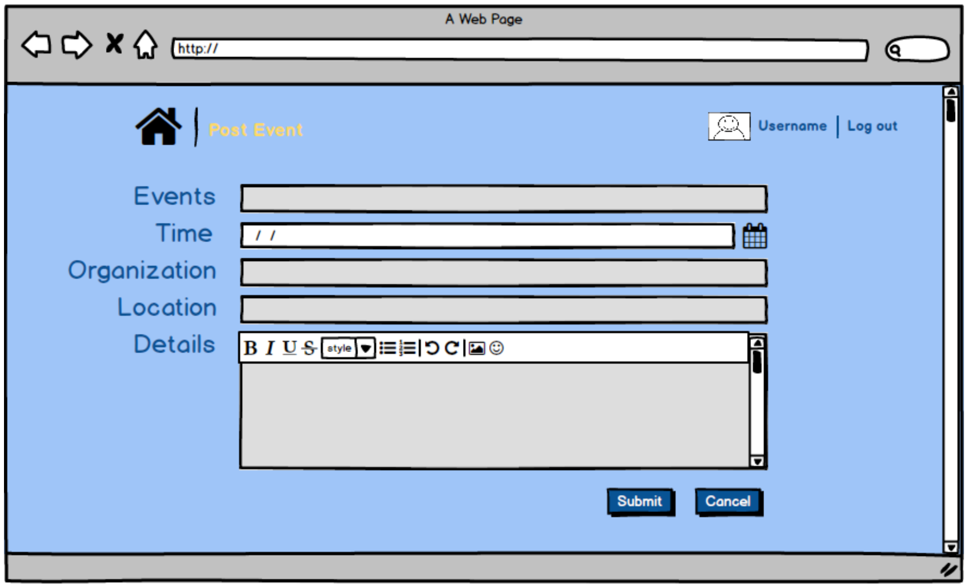
This is the home page for organizers to do their further operation such as login and registration by pressing the button mentioned in this image.

4.1.1.7 Web Registration



This is the page for organizers to sign in to the system, and they must identify themselves by uploading some identification files.

4.1.1.8 Web Events Submit



- This page is to be used by the organizers to post the events they are hosting.
- They just need to fill in some of the text field, text area and a drop down menu to give the description of their events.

4.1.2 Objects and actions

For making it a friendly user interface we have a layout that is a standard one. There are text fields to be filled by user to move on further. Then we have some bars to select the event as attending. The app will also ask for double confirmation of the user before making an update to the existing stuff on the application.

For organizers the web application is so designed such that they can get useful data from the system about the people they are going to see at the event. They can also their past events and upload every minute detail about the upcoming event using a portal that is very easy to fill.

4.2 Interface design rules

1. Strive for consistency
Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent color, layout, capitalization, fonts, and so on should be employed throughout. Exceptions, such as required confirmation of the delete command or no echoing of passwords, should be comprehensible and limited in number.
For gatorEvents system, all pages regardless of Android pages and web pages have the same layout and color style, in addition, all buttons and menus such as “Category” and “Favorite Events” are in the same style to maintain consistency.

2. Cater to universal usability.
Recognize the needs of diverse users and design for plasticity, facilitating transformation of content. Novice to expert differences, age ranges, disabilities, and technological diversity each enrich the spectrum of requirements that guides design. Adding features for novices, such as explanations, and features for experts, such as shortcuts and faster pacing, can enrich the interface design and improve perceived system quality.
For gatorEvents system, the use interfaces are designed for smart phone, therefore any student and staff in UF who has a smart cell-phone and is well versed with using smart phone applications would find the interface similar and therefore easy to use and all age groups can use it without any extra knowledge.

3. Offer informative feedback
For every user action, there should be system feedback. For frequent and minor actions, the response can be modest, whereas for infrequent and major actions, the response should be more substantial. Visual presentation of the objects of interest provides a convenient environment for showing changes explicitly.
For gatorEvents system, there are three types of informative feedback besides login and registration. One is notification for users to inform them of how soon the events they signed for will come. The second is a kind of statistic feedback for the organizers to let them know how many people are willing to take part in the event. The other one is a system feedback for every one including users and organizers to message them when the server is down unexpectedly.

4. Design dialogs to yield closure
Sequences of actions should be organized into groups with a beginning, middle, and end. Informative feedback at the completion of a group of actions gives operators the satisfaction of accomplishment, a sense of relief, a signal to drop contingency plans from their minds, and an indicator to prepare for the next group of actions.
For gatorEvents system, after every operation such as login, registration, add/delete favorite category and sign for event being done, a one-line simple string will be popped at the bottom of the screen. In addition, when users or organizers want to delete something like favorite category and events, there will be a double check to confirm he/she really wants to remove it.

5. Prevent errors
As much as possible, design the system such that users cannot make serious errors; for example, gray out menu items that are not appropriate and do not allow alphabetic characters in numeric entry fields. If a user makes an error, the interface should detect the error and offer simple, constructive, and specific instructions for recovery.

For the gatorEvents system, if the user enters wrong password but there is a matched user name in the database table, there is no need for user to fill in the user name again, he/she just has to enter the password again. Other situations like the one mentioned above would be dealt in the same way as this one.

6. Permit easy reversal of actions
As much as possible, actions should be reversible. This feature relieves anxiety, since the user knows that errors can be undone, and encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data-entry task, or a complete group of actions, such as entry of a name-address block.
For gatorEvents system, since every Android device has a back button, users can cancel the operation they did first but does not want the system to execute it actually can bring the operation back by just pressing the back button on their device.
Even in the Web Application the organizer can cancel the operation by clicking cancel button.

7. Support internal locus of control
Experienced users strongly desire the sense that they are in charge of the interface and that the interface responds to their actions. They don't want surprises or changes in familiar behavior, and they are annoyed by tedious data-entry sequences, difficulty in obtaining necessary information, and inability to produce their desired result.
For gatorEvents system, we have detailed documents such as requirement document, design document and many logical diagrams to make the system more structural, as a result, the experienced user can easily understand what one specific interface is responsible for and know about the data flow.

8. Reduce short-term memory load
Humans' limited capacity for information processing in short-term memory (the rule of thumb is that we can remember "seven plus or minus two chunks" of information) requires that designers avoid interfaces in which users must remember information from one screen and then use that information on another screen. It means that cell phones should not require re-entry of phone numbers, web-site locations should remain visible, multiple-page displays should be consolidated, and sufficient training time should be allotted for complex sequences of actions.
These underlying principles must be interpreted, refined, and extended for each environment. They have their limitations, but they provide a good starting point for mobile, desktop, and web designers. The principles presented in the ensuing sections focus on increasing users' productivity by providing simplified data-entry procedures, comprehensible displays, and rapid informative feedback to increase feelings of competence, mastery, and control over the system.
For gatorEvents system, we just have Android platform for users and web back-end for organizers, so it is unnecessary to mention something in common between these two platforms. Of course, we will set cookie for saving the main profile of users or organizers to make sure that they do not need remember anything on the screen or page.

4.3 Components available

Linear Layout: The Android application of the system is in Linear Layout and there is also a Linear Layout embedded to show every page except for calendar page and category page.

Grid Layout: This kind of layout will be used in category page to show more information in one page.

Button: Each function in the system has a button to listen user operation.

Text Field: The system we designed will use many kinds of text field such as password field and email field to get the input from users or organizers.

Image View: This is used to show images on the screen.

Scroll View: Every page which has many pieces of information will have a scroll view to help user easy to read.

Swipe View: All events will be shown in text view embedded in Swipe View to realize the function as sliding left to add and sliding right to delete.

Calendar View: This view is used to show the user's calendar in the specific page.

4.4 UIDS descriptions

The system uses Mockup to design user interface diagrams first and then using Android Studio to realize what we design, and all of them will be in the same style.

5.0 Restrictions, limitations, and constraints

Special design issues which impact the design or implementation of the software are noted here.

5.1 Language limitation

Since we just have three Sprints, it is hard to demonstrate our system in some other languages besides English. We can make more effort to take some main languages into consideration if needed.

5.2 Scope restriction

Our system is just for UF students and some organizations which have relationship with UF, so not anyone who wants to use this system have permission to log in this system. In addition, the mentioned events in the system are just about UF.

5.3 Server constraint

Because our system is a limited one, there is no need for us to use a server that has enormous capacity, which means if there is a lot of unexpected users using it at the same time, it will not be allowed.

5.4 Platform constraint

The reason why we have this constraint is that we just have Android devices and PCs but not Apple devices, so our system only can be run on the following platform: Android (4.0.2~5.1.1), Chrome, Firefox, Internet Explorer (ver. 9+).

5.5 Development experience restriction

Since just some of our members have experience of developing a software system, they must learn something new, and then make something new. Because of this restriction, this software could not be up to the user expectations as it depends on their learning and bring that into implementation.

6.0 Testing Issues

Testing would be done for four layers to make sure every feature and function is performing as per the requirements. These layers include Unit Testing, Component Testing, Product Testing and finally getting all the three of them together to do the System Testing as a whole.

6.1 Classes of tests

6.1.1 Black Box Testing: Black Box Testing would be done at each level of testing mentioned above to make sure that all the boundary value conditions have been taken into consideration. A cause effect graph would be built for each test to verify whether a given cause results into the expected output or not. To exhaustively test the system partition testing would be done to make sure that all invalid and valid conditions are covered and the system is showing an expected behavior in that particular input.

6.1.2 White Box Testing: White Box Testing would be done to check the internal logic and structure in order to make sure that the developer has written the right code for the given requirements. It will be done at the lower level, the Unit and Component Level so that when we will integrate these modules the results are not catastrophic. Testing of code would be done in such a way that the developer who has written a code would get it test by somebody else rather than himself to increase the effectiveness of testing as they might have some ego attachment with their own piece of code.

6.2 Expected software response

6.2.1 Register (ReqId: R-001)

6.2.1.1 User Name:

Invalid Input (TestID: T-001): Numerals, Special Characters except for Space, Null.

Output: Warning String under the text field “Name should be English Letters Only” in red.

Valid Input (TestID: T-002): English Letters with or without space.

Output: Move on Registration.

6.2.1.2 Email Address

Invalid Input (TestID: T-003): Wrong style Email address and Email already registered.

Output: Warning “Wrong Email address format” under the text field or “User already registered” in red.

Valid Input (TestID: T-004): Email in the right format.

Output: Move on Registration

6.2.1.3 Password

Invalid Input (TestID: T-005): Password length less than 6 letters.

Output: "Password should be at least 6 characters long" under the text field in red.

Valid Input (TestID: T-006): Password equal to 6 or more characters.

Output: "Successful Registration" and redirected to home page.

6.2.2 Post or Edit (ReqId: R-002)

6.2.2.1 Event Name

Invalid Input (TestID: T-007): Null value or only special characters or numerals.

Output: "The Name of the event should contain English letters" under the text field in Red.

Valid Input (TestID: T-008): Event Name with English Letters and a combination of English Letters, Numerals and Special Characters.

Output: Move on Event Posting or Editing.

6.2.2.2 Description

Invalid Input (TestID: T-009): Null value

Output: Event Description can't be null.

Valid Input (TestID: T-010): Strings with English Letters.

Output: Move on Event Posting

6.2.2.3 Attachments

Invalid Input (TestID: T-011): Formats such as doc, xls or exe or the size of file greater than 5MB.

Output: "Wrong Format" or "File too large to upload".

Valid Input (TestID: T-012): Format such as PNG, JPG, PDF.

Output: Move on Event Posting.

6.2.2.4 Venue

Invalid Input (TestID: T-013): Null Value.

Output: Event Venue can't be Null.

Valid (TestID: T-014): Combination of numerals and strings.

Output: Move on Event Posting.

6.2.3 Search (ReqId: R-003)

No invalid input

Valid input (TestID: T-015): Enter some letters and click “Search” button

Output: Jump to the right view and show the right results of search.

6.2.4 Calendar (ReqId: R-004)

No invalid input

Valid input (TestID: T-016): Choose time in the calendar and click “OK” button

Output: Jump to the event search that satisfy this frame.

6.2.5 myEvents (ReqId: R-005)

6.2.5.1 Add Events

No invalid input

Valid input (TestID: T-017): Sign for one event.

Output: Jump to the “myEvents” view and right information of this event should be mentioned.

6.2.5.2 Delete Events

No invalid input

Valid input (TestID: T-018): Sign out one event.

Output: Remove this user’s U_id_Users in the ‘Favorite_Events’ table from the database and the panel of this event should be deleted in the ‘myEvents’ view.

6.2.6 Favorite (ReqId: R-006)

6.2.6.1 Add Favorite

No invalid input

Valid input (TestID: T-019): Choose a favorite category.

Output: The right favorite category is added into the ‘Favorite_Categories’ table in the database and users can check it in the “Favorite” view.

6.2.6.2 Delete Favorite

No invalid input

Valid input (TestID: T-020): Delete a favorite category.

Output: Remove the corresponding U_id_Users in the ‘Favorite_Categories’ table from database and uses can check it in the “Favorite” view.

6.3 Performance bounds

6.3.1 Screen transitions

The application should have smooth screen transitions for friendly usage. It should be ensured that there is no time lag while we switch between the pages. It absolutely can give users a good experience.

6.3.2 Database query fetching

The application is going to have images stored in the database which requests a good database speed to fetch them in real time so that each user can switch between the pages and get information simultaneously.

6.3.3 Server capacity

According the analysis, the application should be used by at least 10,000 students on campus, therefore to ensure all of them being served simultaneously, the server capacity should be increased by buying more bandwidth.

6.4 Identification of critical components

Those components that are critical and demand particular attention during testing are identified.

6.4.1 Account verification

6.4.1.1 Password verification

Since login is the first step a user should follow, the password verification is the most important and the highest priority component to be identified. The system must confirm whether the password is right for the user before allow a user to get in.

6.4.1.2 Email verification

The system asks organizers and users to provide with their email address to make sure that the user or organizer registering is actually the real one. To do this, the application sends a verification email to email address provided by the user or organizer, which they need to verify to proceed further. Otherwise the system will reject their registration.

6.4.2 Event post and edit

Because event is the most critical component in the system, users should be notified immediately if any updates or changes occur to an event they signed for. The database should also be updated accurately about the same event so that it gives the right information on the application.

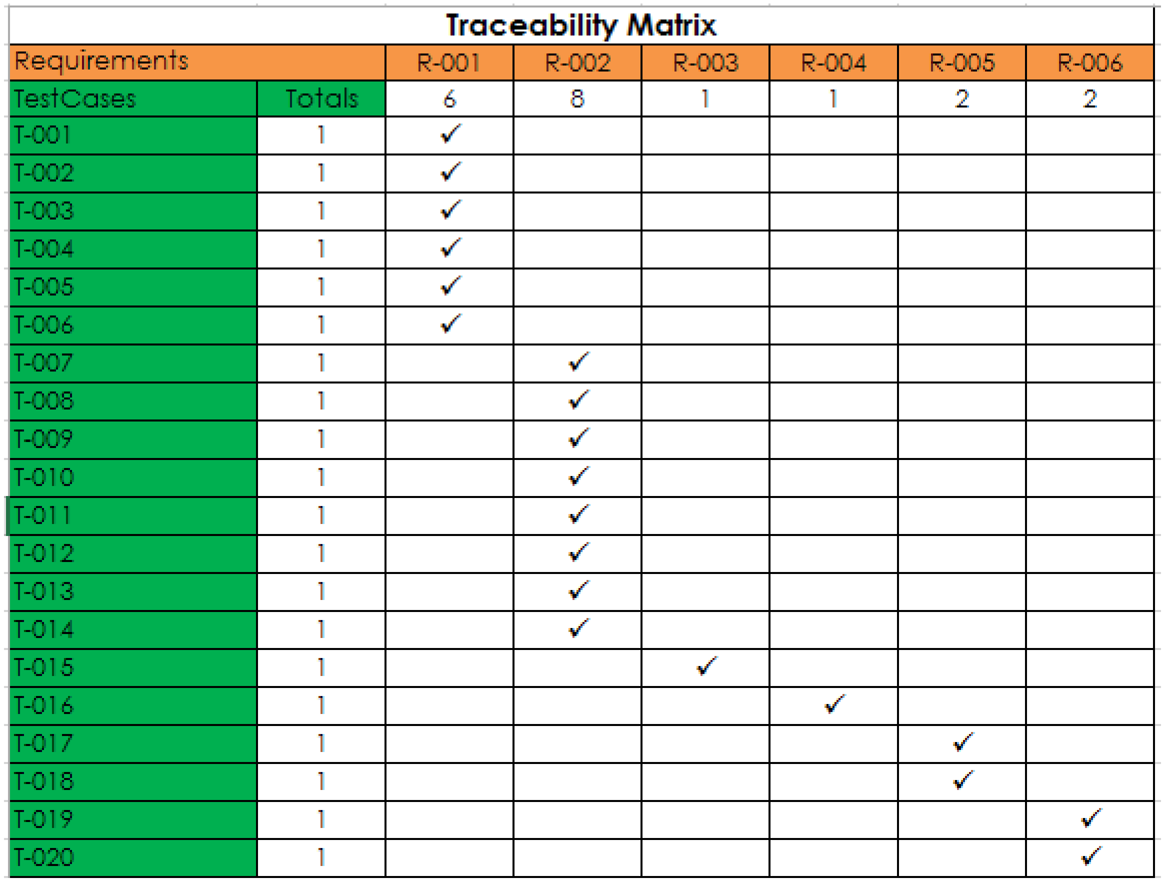
6.4.3 Notification

Notification is an important component of the system as it gives flexibility to the user to select from various options of getting notifications about the events they signed for by selecting the time they want to get them, so the system must be real time to ensure them the same.

7.0 Appendices

Presents information that supplements the design specification.

7.1 Requirements traceability matrix



Traceability Matrix							
Requirements		R-001	R-002	R-003	R-004	R-005	R-006
TestCases	Totals	6	8	1	1	2	2
T-001	1	✓					
T-002	1	✓					
T-003	1	✓					
T-004	1	✓					
T-005	1	✓					
T-006	1	✓					
T-007	1		✓				
T-008	1		✓				
T-009	1		✓				
T-010	1		✓				
T-011	1		✓				
T-012	1		✓				
T-013	1		✓				
T-014	1		✓				
T-015	1			✓			
T-016	1				✓		
T-017	1					✓	
T-018	1					✓	
T-019	1						✓
T-020	1						✓

7.2 Packaging and installation issues

Special considerations for software packaging and installation are presented.

Packaging requires that the Android application be signed. The Android system will not install applications which are not signed. After this the final APK package will be zipped.

No special considerations are warranted here beyond installation of gatorEvents app through Android Marketplace and/or running of the environment through the SDK emulator.

7.3 Design metrics to be used

Stability will be determined by measuring and calculating the ease to change packages without affecting other packages within the application. Abstractness will be evaluated subjectively by analyzing code readability, modularization of classes and methods, and overall structure of the code.