**EventUp!**

**Team 4 - Design Document**

*Yan Abusamen, Samuel Carbone, Joseph Dafforn, Joseph Davey, Jackson Didat, Siraj Zaneer*

**Purpose:**

At Purdue University, events occur almost every single day but the issue is a majority of students are not even aware of them. To even get a small amount of people to be aware of these events requires organizers to send Facebook invites, use different department mailing lists, put in ad requests for TV boards, etc. which in turn requires students to be see these advertisements. “EventUp!” aims to solve this issue by providing organizers a centralized platform to advertise their events as well as allow students at Purdue a simple and reliable method of finding and being notified of these events.

Functional requirements:

1. Events:
   1. Any registered organizer can create an event. Information such as description, location, and time can be added.
   2. Event can be associated with an organization.
   3. Event can be edited later by the event’s organizer to change the details.
2. Checking into event:
3. Any user can check into an event granted that they are within vicinity of event based on device geolocation.
   1. Application will prompt user to check into the event if they are within Geofence of the event.
   2. User can opt in or out of auto-check ins.
4. Map of Event:
   1. Any user can see a map of local area which displays pins of events.
   2. User can click on a pin to go to event information page.
   3. User can click on a pin accessory button to navigate to event using Apple Maps
5. Event list page:
   1. User can see local events in a scrollable list with relevant information.
   2. User can sort/filter events by event metadata.
   3. User can search events by event metadata.
6. Viewing event information
   1. User can click on a map pin or list cell to see the event metadata.
   2. User can see the information on each event in the events feed.
   3. User can search for event metadata and view event information in the search tab.
   4. User can report inappropriate events.
7. Suggested events
   1. User can scroll through suggested events based on how many participants are attending the event.
   2. User can search through suggested events based on event metadata.
   3. User can sort/filter suggested events based on event metadata.
8. Chat room
   1. Users can anonymously chat with other users in rooms associated with events.
   2. Chat room can be enabled or disabled at organizers discretion.
9. Notifications
   1. User will be notified of upcoming events.
   2. User will be notified of events close by that they can check into.
   3. User will be notified of events they have RSVP’d to when they are close to the location of the event.

Non Functional requirements:

1. Client application
   1. Application should be able to run on an iOS Device.
   2. Application should not take up too much memory (utilize database to store event information).
   3. Application should work fluently with Apple maps to deliver directions to the user.
2. Database
   1. Database should be able to hold event information.
   2. Database should be able to store information about registered event planners.
   3. Database should be able to store information on all users.
3. Performance
   1. Client application should be responsive and not drop frames.
   2. Race conditions on should be avoided through the correct use of threading.
   3. Database should respond to queries quickly and be able to scale easily as user base increases.

**Design Outline:**

This project will be an application that displays information about local events to users with information about events being stored in a database that can be accessed and edited in real-time. We will be using the client-server model to allow us to communicate with the database from the clients. Our client will be created using the Model-view-controller architectural pattern allowing for efficient and effective development.

Components:

1. Client:
   1. iOS Application which users will use to leverage the use of our system.
2. Server:
   1. Firebase to communicate with the database.
3. Database:
   1. Firebase to keep track of events and users.

Components of our system: We will have Firebase interact with the user to give the current event details and number of users

Describe the interactions between individual systems:

Since our system is comprised of a client, a server, and a database interactions will occur as the following:

1. Client to server:

The client will send requests for information to the server such as a query for events or an event planner’s information.

1. Server to database:

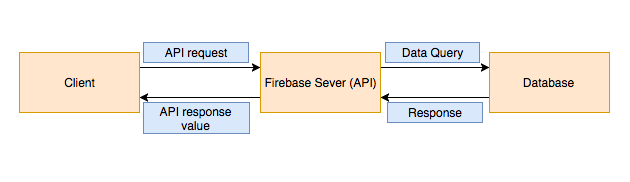
The server will query the database for the information that was requested by the client.

1. Database to server:

The database will return the data which was queried by the server.

1. Server to client:

The server will send back to user the data that it received from the query to the database.



**Design Issues:**

Issue 1: What feature will the application initially launch on startup?

* **Option 1: The map of events**
* Option 2: A home screen displaying icons of all features
* Option 3: The events homepage (Creating, search, attend)
* Option 4: A feed of events based on the user’s preferences

We were considering launching into a home screen with icons (option 2), however we decided to go with option 1 because we will be implementing a navigation bar, which keeps the interface more simple, and users can easily navigate the different features all in one place.

Issue 2: What backend framework should we use?

* Option 1: Parse
* **Option 2: Firebase**
* Option 3: MySQL

After considering our options for the backend framework, we decided to use Firebase because it allows us to build the app fast, without managing all of the infrastructure. That is the most compelling feature of firebase, as this quick launch will allow us to save time in development and focus more on the user interface and smooth functionality as a whole. Firebase is also backed by google, which means we don’t have to be concerned with how large the app becomes. Another reason is how well Firebase is integrated with iOS, this makes it easy to include in our project making it the overall best option in this case.

Issue 3: How can the user browse the current events?

* **Option 1: A scrollable list of events**
* Option 2: Pins on Map
* Option 3: Swipe-able information cards

There are a few interfaces we considered using when looking at the current events. Current apps have a swipeable list of information cards, which is very user-friendly. However, this method would most likely be difficult to implement, as well as cause the user to have to swipe many times just to see a few events. Option 2 is also a good alternative, but the user will not be able to see what each event is; they must click on the pin in order to be able to view details. In the end, we chose option 1, because this allows the user to see all current events. They will be able to scroll through all the events easily and quickly to find something they are interested in. This will engage the user and encourage them to use the application more, since they can quickly become a participant of an event.

Issue 4: How can the user browse future events?

* Option 1: Pins on a map
* **Option 2: A scrollable list of future events**
* Option 3: Swipeable info cards

Since these events are in the future, there will not be a way to put pins on the map, because this would interfere with and draw users away from current events. For option 3, it is feasible to create swipeable info cards, but this would again be difficult to implement as well as it would not be as easy for the user to quickly browse the events. This brings us to option 2, creating a scrollable list of future events. This would allow the user to quickly browse the future events and RSVP to their favorite events.

Issue 5: What method should we use to guide users to events?

* Option 1: Google maps
* **Option 2: Apple maps**
* Option 3: MapQuest

There are multiple choices that we could pick for a map. Google maps has potential, but since we are programming on iOS, it would be easier to use the built in Apple maps capabilities. We will most likely choose to go with Apple Maps because it is already present on the device and it will take up less memory space. Lastly for MapQuest, this method is slightly outdated and does not come built in on all iPhones, so it would not be feasible to implement this method.

Issue 6: How do we check users into the event?

* Option 1: Automatically check them in when in the vicinity
* **Option 2: Let the user check in when they arrive**
* Option 3: Let the event coordinator check them in when they arrive

Option 1 allows the user to not worry about checking in when at the event and leaving that to the applications back end. The problem with option 1 is that a user might not be checked in because GPS is not always perfectly accurate. Option 2 lets the user check in manually and make sure that they get checked in; but, that puts the responsibility on the user that many people might not want. Option 3 would allow the event coordinator to make sure that people don’t check in without actually attending the event.

Issue 7: How do we check for user permissions?

* **Option 1: Give the user permission if the event belongs to that user**
* Option 2: Not include special user permissions
* Option 3: Give all users permission to change events

There are a few options that we could have gone with in regards to giving users permissions. The first is that we could not include special user permissions, but this option is not feasible since there would not be a way for an event planner to edit details for the event they created. The second option is giving all users permission to change any events, but this is a bad option for obvious reasons. If we gave all users permission to change any event, there would be no accountability and all events would be changed maliciously. Therefore, the last option is the one we are choosing, since it will give just the right amount of permissions to each user that has created an event.

Issue 8: What method should we use for users to chat at the events?

* **Option 1: Firebase**
* Option 2: Create our own chat system
* Option 3: Do not include chat

The best option for this user issue is using Firebase. This is the best option because Firebase takes care of managing users, events and chats. This will allow us to create a new chat for each event, and it will take care of deleting the chats after each event is over. Creating our own chat system would create more work for our group and would likely not be as accessible as Firebase. Lastly, we should include a chat because it will be very useful for users to be able to communicate details about the event that can’t be otherwise displayed to other participants.

Issue 9: How do we calculate user ratings?

* Option 1: Amount of events attended determines rating
* Option 2: Peer rating system
* **Option 3: Average of event ratings**

Although there are a few options we can use to calculate user ratings, there is really only one method we can use. We have decided to go with option 3 because we feel this is the most accurate way to determine how well an event was put on. After the event is over, we will take an average of the ratings on a scale of one to five, and show the average for the event. Option 1 is a good alternative but basing the rating on the number of events attended is biased towards users who attend many events, but this may not be the case for most participants. Using this system would give users a more negative rating if they don’t attend many events. The second method is a peer rating system, but this has many faults since there is no predictability to how users will rate an event, and it may not be consistent over multiple events/users.

Issue 10: How should users edit their event details after the event has been created?

* **Option 1: Let them edit the description, the date, the time, and location.**
* Option 2: Let the user only edit the description, the date, and the time.
* Option 3: Let the user only edit the description.

For option 1, it allows the event creator the freedom to change all the aspects of the event; but, at the expense of the attenders confusion. Changing details of an event might cause some confusion with the user because they might be confused on when and where the event is. For option 2, it might reduce some of the confusion caused by changing events details; but, still allows the creator the freedom to change some of the attributes of an event. For option 3, it takes away the creator’s freedom to change their event; but, it reduces all confusion from users that might want to attend the event.

Issue 11: What framework should we use to access user’s location?

* Option 1: Track user’s location in the background
* **Option 2: Use location only when the app is open**
* Option 3: Not use the user’s location

For option 1 we could track the user’s location in the background, but this is unethical and would violate the user’s privacy. There are some apps that have done this in the past and have gotten in trouble because they were tracking locations all the time. Option 2 is what we chose to go with, since we will only have access to the location when the app is open. This respects the privacy of the users and also allows us to calculate the location and automatically check in users by using geolocation. The third option is not feasible, since we need to know the user’s location. If we don’t have the location, we will not be able to show the user events near them, as well as their location will not show up on the map. This would make it harder to help the user discover new events, and would probably discourage users from participating in events, since many events would be too far away.

Issue 12: How should we authenticate users that want to create parties?

* **Option 1: Have the user sign in with a Facebook account**
* Option 2: Have the user sign in with a Google account
* Option 3: Have the user sign in with a Twitter account

Option 1 is the best option because most people have Facebook accounts and it would allow users to post the events they create to their feed if they wish to. Option 2 would be another good option because a lot of people also have Google accounts and Google accounts are easy to create. Option 3 is not a great option because although many people have Twitter accounts, the authentication is not nearly as good as Google or Facebook.

Issue 13: How can user’s join events

* **Option 1: Search for events using search-bar**
* Option 2: Join events via Map pins
* Option 3: Join events via location filter

All options are great options. They all allow users to easily join an event that they are interested in. Option 1 would help them find specific events instead of having to search the entire map for a certain event. Option 2 would allow the user to easily see all events on the map and how close they are to each one. Option 3 lets users see locations where they might be interested in going to. Option 1 will probably be the best option because it gives the user the freedom to search easily and gives them the freedom to look up specific events quickly.

Issue 14: How will user’s know if events are sponsored or not?

* **Option 1: Pin has logo, or color-coded**
* Option 2: Sponsor Info is in description of event
* Option 3: It is unbeknownst to user if event is sponsored

We have decided that having color coded pins is the best option because it makes it clear to the users that the event is sponsored. It is important that we differentiate the sponsored and organizational events from the user created events, because our users will be looking for different types of events. This is a simple, yet impactful way to benefit our users.

Issue 15: How will we filter out spam among users?

* **Option 1: Limit user’s to certain number of events per day/week**
* Option 2: Look at event/attendee ratio
* Option 3: Manually cap certain users

Option 1 is an easy way to limit the number of events a user creates; but, doesn’t keep them from creating multiple events. It also helps the application developers keep track of users and not have them making a huge amount of events daily. Option 2 may not work because sometimes some events may just be a little smaller than others. Some events like niche callouts will have a low attendance which would potentially get shut down unjustly. Option 3 requires the developers to find certain users that they believe are violating the rules and limit their use. That would be a large task for the developers because there could be a large number of users that are violating rules and that would take a lot of time to find all of the users using the application for the wrong reasons.

Issue 16: How to interface with Firebase database

* Option 1: Implement our own interface for the database
* **Option 2: Use the iOS Framework for Firebase**
* Option 3: Not interface with the database

We have a few options when it comes to interfacing with the database. The best option is to use the iOS framework for Firebase. This will allow us to easily interact with the database to get information on users, events and managers. Using Firebase is the most simple and also easiest to implement since the Firebase API is well documented and easily integrates with our project. The first option is a good alternative, but creating our own interface would

Issue 17: How to prevent multiple check-ins from a user

* **Option 1: Flag it locally**
* Option 2: Flag it based on unique identification from phone
* Option 3: Limit based on number of users checked in

There are a few ways we can prevent multiple check-ins from the user. The first is to flag it locally. This means that we keep track on the user’s device if they have been checked into the event already, and if they have then prevent them from checking in again. This will keep the current number of participants very close to the real number. The second option is to flag it based on unique identification from the phone. This is a good alternative, since we can keep track of the phone’s unique identification. This could raise privacy concerns though, since we are keeping track of where the user is trying to check in. The third and final option is to limit check ins based on how many users have RSVP’d. This is not a feasible option, since it is possible that there are participants who did not previously RSVP. This means that if the user didn’t check in, they wouldn’t be able to attend the event. This goes against the basis of our app, since we want people to be able to attend whenever they want, whether that be last minute or RSVP in advance. In the end, we chose to go with option 1, since it provides us with the best way to keep track of the actual number of participants present at each event without violating the privacy of the user.

Issue 18: How will users receive notifications regarding new events

* **Option 1: push notifications**
* Option 2: in-app notifications
* Option 3: email/text notifications

Option 1 is the quickest way to reach the user. It will pop a notification instantly without having the user open up another application. Option 2 also lets the user see the notification without leaving the application; but, we can use existing iOS features to display the push notifications. Option 3 is not a good option because it is too slow and users may not notice the notification if it sent through the email.

Issue 19: How will inappropriate events be handled

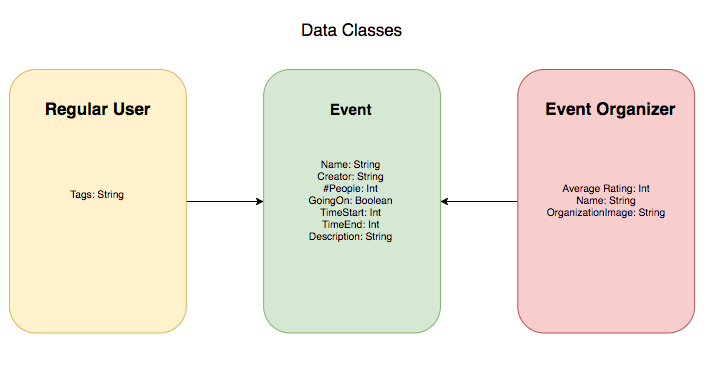
* **Option 1: User reports events deemed inappropriate**
* Option 2: App filters events itself
* Option 3: No filtering of inappropriate events

We have decided to go with option 1 because we want our users to post and at the same time, we don’t want users spamming reports without a reason. This is why we have decided to review and make a decision on an event when an inappropriate event is seen and reported by a user. This is the best option because it allows us to be the middleman in deciding if a controversial event is actually inappropriate.

Issue 20: How will events in the same place/vicinity be displayed

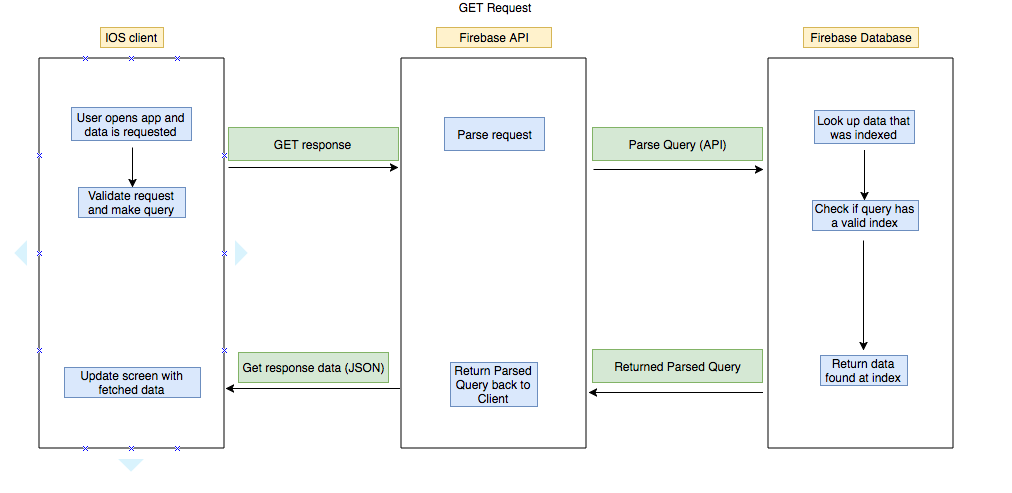
* Option 1: As overlapping zones on the map
* **Option 2: As pins with small distance between them on the map**
* Option 3: Create a conflict for the two events

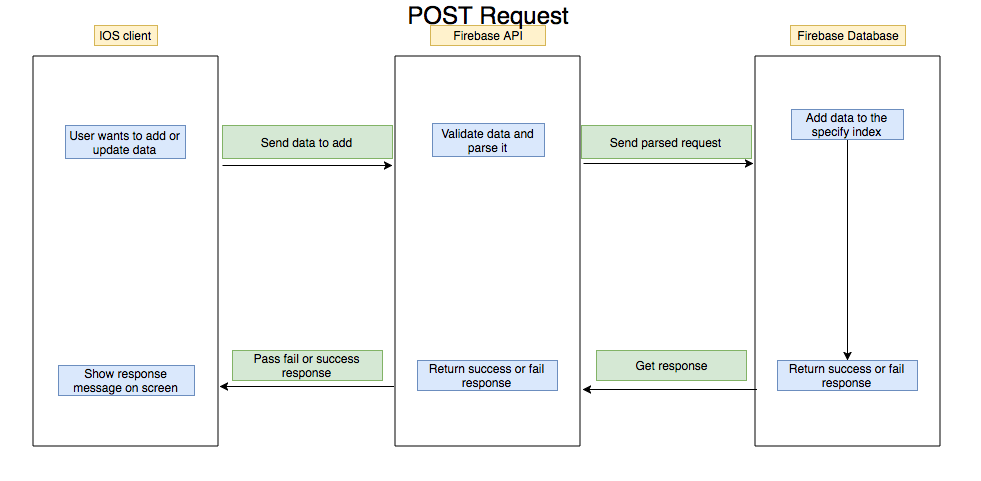
After considering the options for this issue, we have decided that option 2 is the best choice because it is the most visible. If we were to implement option 1, there would be confusion with the overlap of the zones. It is much more simple and practical to clearly have the events displayed as pins. We quickly rules out option 3 because we want users to be able to have events that are close in location, while also clearly showing their location.

****

The data classes will hold information for regular user’s (Once which just attend the events). The Regular user will store the user’s favorites locally, and suggest events based on those favorites. The biggest data class is the Event, which will hold the description of the event, number of people, creator, etc. That is the main data class that will hold the most information. The Event organizer will hold the Organizer’s average rating, the user’s name, and the organization that they are associated with to display in their events.

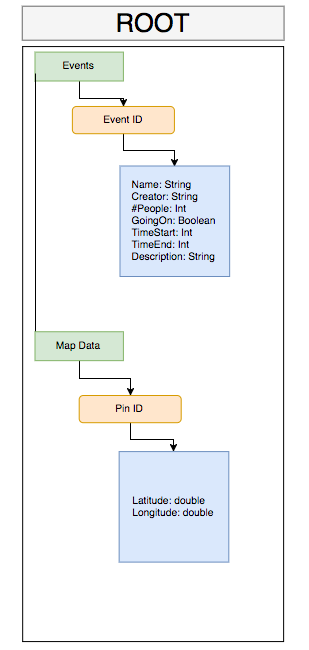
**GET/POST protocols**

****

****

**Database Layout**

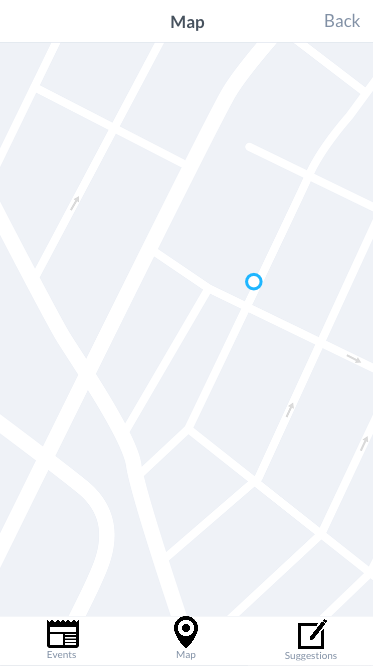
The database will hold the events and it’ll hold all the event pins for the map. When users click on the map it’ll GET data from the Map Data, and drop the pin on the map. Once the pin is clicked the events data will be requested and show to the user.

****

**App state diagram**

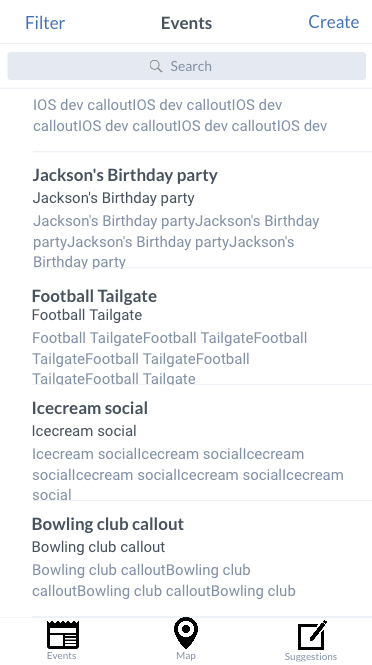
**Map**

This is the main map that will be shown to the user’s when they open the app or want to see event activities. User’s will be able to click on pins that will represent events and information about the event will be displayed.

****

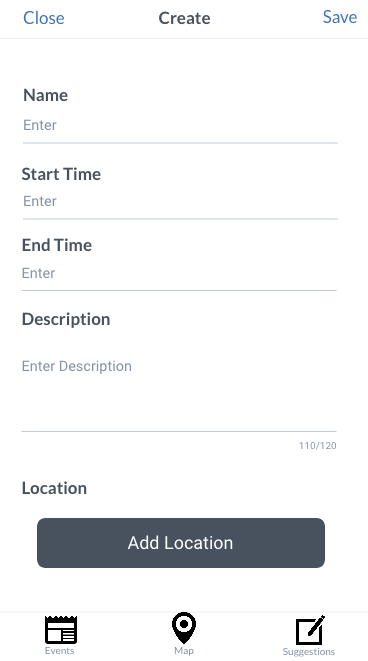
**Event List**

This is going to show a list of events, and the list can be filtered based on tags. It has a search bar which you can also use to find an event by its title. There is an option to create the event on the top right which will lead to the next screen.

****

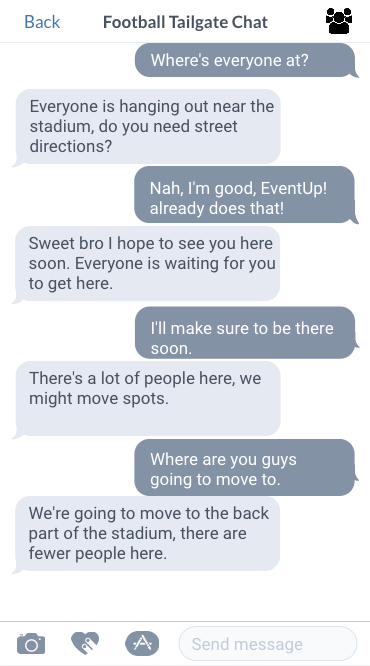
**Create Event**

This screen will allow organizers to create an event and type in the required fields. Then at the end the organizer will be able to add location by clicking the button. This will open a map and the organizer will be able to pick the location of the event.

****

**Event Group Chat**

This screen will be the chat interface the user will see whenever they want to talk to people currently at the event. They can use the chat to ask questions about the event and just meet new people.

****