# EventHype

## Project Specification

Artur Grigoryan
Thomas Halstead
Russell Fenenga
Jonnathan Petote

CS125

## Abstract

Our project will help people search for events around them from a mobile application as well as the ability to create events. We believe that this is a major area that still has not implemented a system that can be searched through and that is what our team and project aims to solve.

## Introduction and Problem Definition

Currently to find events on college campuses or in any community people rely on seeing flyers or being invited to a Facebook page by someone they know. This system is highly inefficient and there is no real to search for specific types of events on the fly. These events could range from small events like attempting to get a pickup game of soccer together at a local park to huge events like concerts. Currently it is very hard to find a list of all these possible events going on around a user's location and that is what we aim to solve.

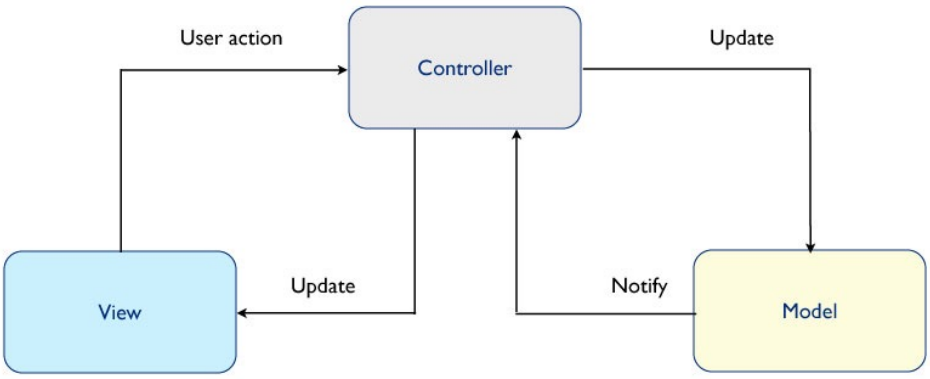
Our system will require multiple parts that will be outlined in this report. The first of which is our backend which will house our SQL database as well as our API for interacting with our database. This API will connect to our iOS app which will provide the users the functionality to view all the events currently active around them as well as filter what types of events they see and how far around them they can see events. This document aims to outline the technical aspects of this project and how we are going to structure all the components of this system.

## Current State of Art

Currently there are no systems that operate in the same area that we are planning on targeting. Facebook has events but searching for events isn't easy and they aren't sorted by location around you so there isn't any reason to use it for searching for events.

## iOS App Architecture

The general architecture of the iOS app is going to be incredibly standard to fit within Apple's guidelines. The project will be built using the Model View Controller design pattern. We decided to go with this choice because it is the standard for iOS apps and fits the needs of our project perfectly. The first thing we will discuss is what each part of the app is going to perform and how it is going to work. The picture below represents how the MVC pattern functions.



# iOS UI

## Event Feed

●●●●○ Carrier 1:20 PM 100%

Events

EDITORS' PICKS

POSTED FEBRUARY 4, 2017 #DESIGN

### How to create beautiful typography

Typography is the visual component of the written word. It is for the benefit of the reader , not the writer.

READ ARTICLE →

POSTED FEBRUARY 2, 2017 #CLUB

### Night out on the town

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt elit in mi imperdiet, sed hendrerit nibh scelerisque.

READ ARTICLE →

# Event Content

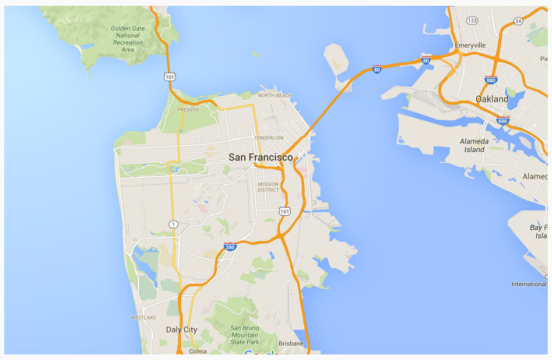
POSTED FEBRUARY 4, 2017 #DESIGN

## How to create beautiful typography

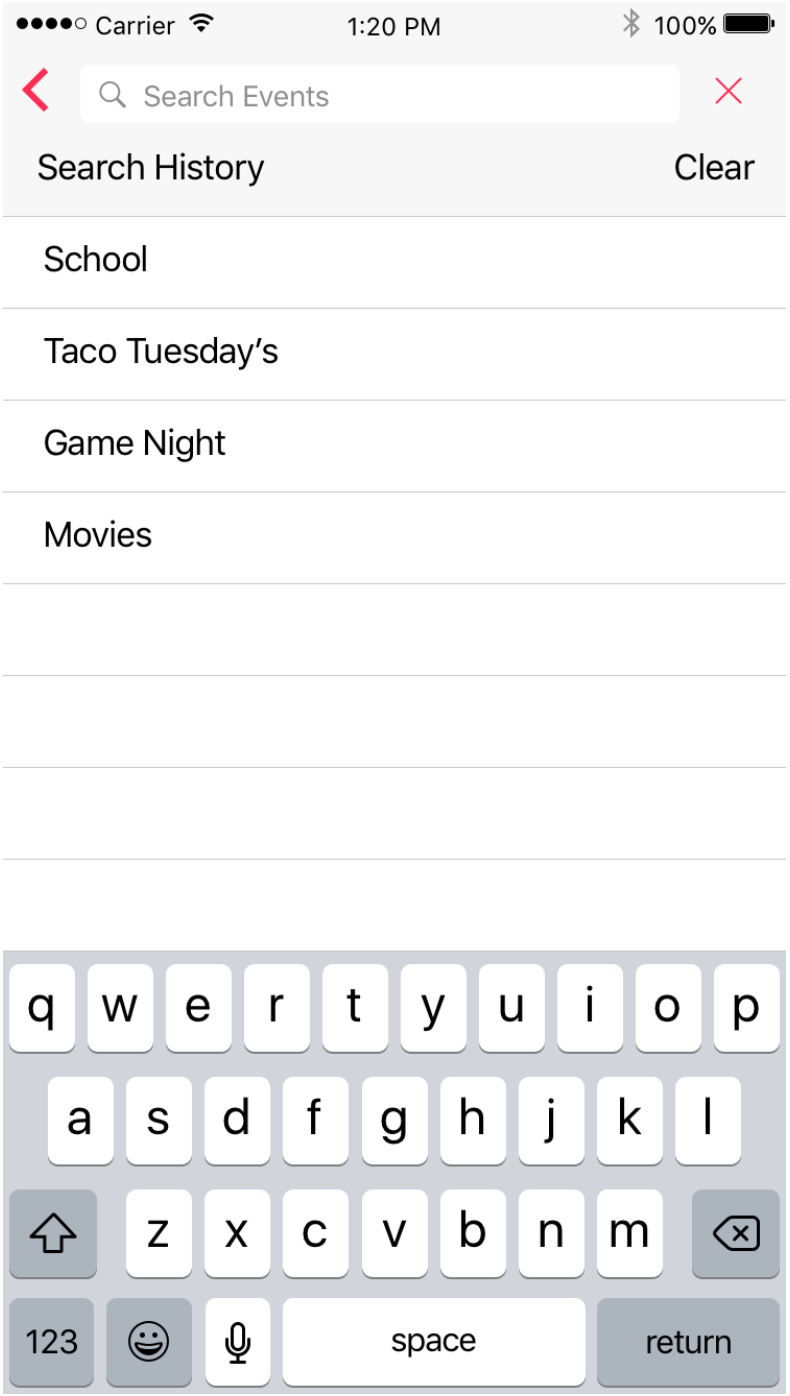
Typography is the visual component of the written word. It is for the benefit of the reader , not the writer. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt elit in mi imperdiet, sed hendrerit nibh scelerisque. Interdum et malesuada fames ac ante ipsum primis in faucibus. Vivamus ultricies velit non gravida hendrerit. Duis lobortis eget lacus sed mollis.

Vulputate quam est viverra augue. Vestibulum tristique, nisi id dignissim egestas, dui orci bibendum libero, quis porta ante elit vitae elit. Proin placerat est ac sagittis malesuada. Aliquam sit amet dui at metus porta posuere ac non nulla. Cras ligula nisi, blandit vitae dui quis, luctus sollicitudin nunc. Vestibulum vel justo semper, vestibulum dui vehicula, egestas dui. Maecenas lacus.

888 Brannan St, San Franscisco, CA 94103



Search


Carrier
1:20 PM
100%
Search Events
Search History
Clear
School
Taco Tuesday's
Game Night
Movies
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
return

Add Event

●●●●○ Carrier 1:20 PM 100%

Add Event

Add Event Photo

Title

Location

Description

Tags

Post Event

# Mobile Model

The goal of the model in this project is going to be handling all the loading of data with our database through the API we are building. This means that this is going to be the most imperative part of the iOS app so making sure that this is built correctly is highly important. The model will consist of multiple classes that represent the data that we are going to be loading through the API, the most important of which is the Event class. This class will contain properties for:

i. Location (Lat, Long)

ii. Event Title

iii. Event Description

iv. Event Image

v. Tags describing event type.

This class is going to be the core model of our app and will also contain convenience methods for interacting with it. One important method will be a serializer which will take the class and serialize the data to be passed into the API. Along with this the class will contain methods for updating the different properties listed above.

Outside of this class the other important class will be the API layer which will also be included in the model. The API layer acts a wrapper around our api allowing us to make calls to nicely named methods in this class from our controller to load events or update events. One important thing that the model has to deal with is notifying the rest of the app that an event has been loaded or that a current event has been updated. iOS provides multiple ways to do this but the easiest way to handle this is

going to be creating a delegate that our controller will conform to which is called every time that a new event is loaded from our API or updated. This takes us to the next part of the app architecture which is the controller.

## Mobile Controller

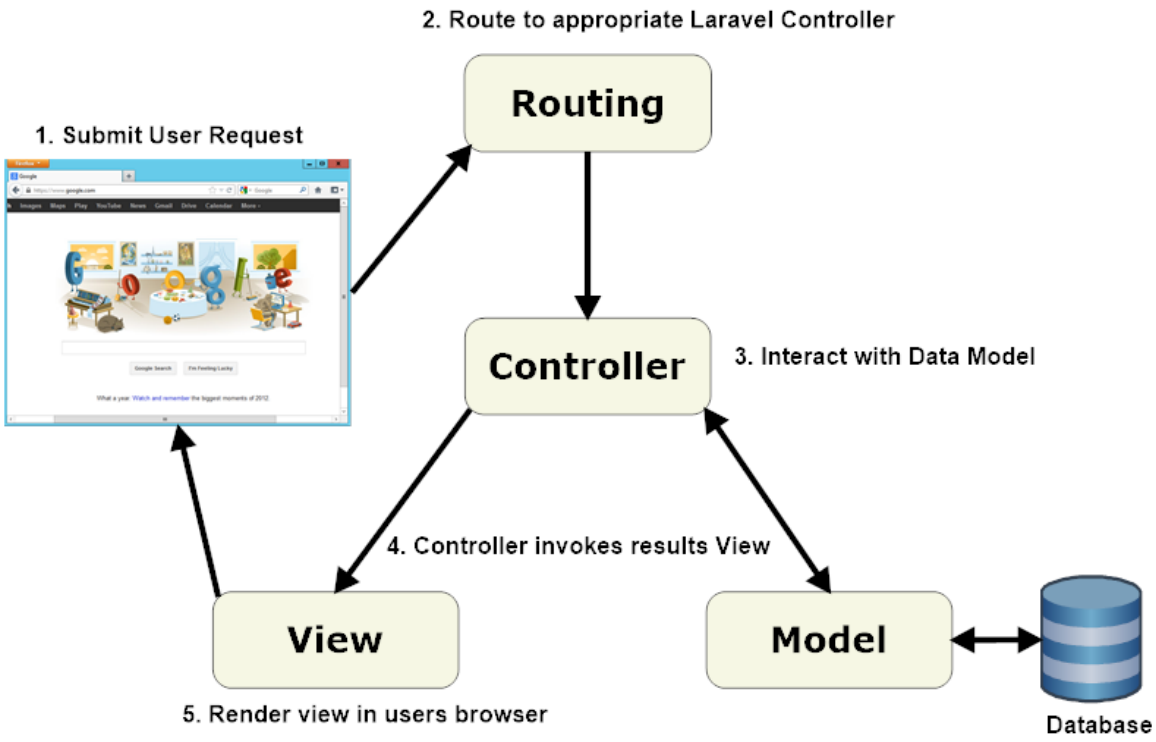
The controller acts as the middleman between the Model and the View and in our case is going to perform 2 main tasks. The first task our controller is going to be responsible for is handling user actions in the app and updating the view accordingly. The other, more important, functionality of the controller is going to be calling our API layer and creating new instances of our Event class. Our first controller class is going to be MapViewController which will handle all the displaying of the icons for events on a map. This also requires that it needs to know when new data has been received by our server. This means that our controller will be communicating both ways with our view (both updating and receiving actions from it) as well as communicating both ways with our model (sending data to the model and receiving data from the model). This means that this controller is going to be the most robust of all the controllers the project is going to have. The other main controller that this project will have is a controller for creation of Events. This controller will just respond to user input and then communicate with the model to create a new event.

## Mobile View

The last component of the architecture for the iOS app is the view. The view for this app is going to consist of two main screens. The main screen will be a map view that is centered around the users current location and is populated with markers for all the current events. There will be a button in the top right corner for creating a new event that will transition to a new page for creating the event. On the bottom of the map view there will be a filter which will allow the user to filter the events shown based on tags and radius around where they currently are. The view will only talk directly to the controller thus keeping independence and coupling to a minimum with the model.

## Desktop App Architecture

The API for the entire project requires a MySQL Database with the ability to make POST and GET API requests. To accommodate the API requests we are using a Laravel MVC framework. Laravel has an ORM that will make the queries against the MySQL Database, and the framework Models allow us to make modifications to the data before serving it via API. The modifications include evaluation algorithms, adjustments dependent on the client location, and other requirements that the project might have. The Desktop App will use the MVC model mentioned previously, and an additional routing server that formats incoming user requests and parses the URI.




## Desktop Model

The Laravel Model will use the Eloquent ORM to make the queries and evaluate the data. Each database table has a corresponding "Model" which is used to interact with that table. Models allow you to query for data in your tables, as well as insert new records into the table.

It will also be creating the index, based on the query data returned. We will be using the “Faker” OSS repo to create the initial Event data for the application

## Desktop Controller

Laravel Controllers can group related request handling logic into a single class. The Controller for the Desktop App will serve to functions. First, it will format the Model

data in JSON to be returned to the Mobile or Desktop client. Second, since we're also creating a Desktop version of the app, the Controller of the Desktop App will create the objects that are required to be rendered on the Desktop Client view.

## Desktop View

The Laravel View – called Blade – allows the controller data to be formatted using HTML/CSS/JS and be displayed on the user's device. We will use the Laravel Blade to render the client pages on the screen.

## Software

For the iOS portion of this project there are multiple pieces of software we are going to use. For designing the user interface of the app we are going to us Sketch which is a vector designing software and Paintcode which allows us to design interactive elements and it generates native code for them. The app will be written in Objective C as opposed to Swift. We decided to go with Objective C because of the stability that it has with Cocoa's Library APIs vs. Swift which is still evolving and breaks a lot if a new version were to release while we are working on this project. When it comes to building the actual app we are going to be using Xcode which is Apple's IDE for iOS Development.

One major component of this project is going to be making calculations based off of location and distances so we are going to need to write software that handles the math for calculating what events are inside of the area defined by the user.

## User Studies and Evaluation

For us, adding an analytics library into the app is going to be the easiest way for us gather information on the how the users are using the app. This will gather info on how many of the users are creating events vs. looking at events. For testing the app we will have to populate the app with demo data until people start actually using the application.

## Conclusion

Our app aims to solve the problem of finding local events to get involved in. There are systems that exist for creating events and interacting with attendees but there are no systems for actively being able to find ones based off the users location. Our filtering system will allow users to only search for events that they are interested in getting involved with. We hope that our app will provide a new avenue for people to get connected with people who they share similar interests with.