Party Pass

Requirements Specification
CompSci320, Spring 2017

Team Party Hardy

Manager: Tamara Zbitnaja

Nick Merlino
Roman Ganchin
Josh Silvia
Michael Gallant
Matt Cassano
Will Sattanuparp

Table of Contents

1 - Introduction

1.1 - Purpose of the System

Party Pass is a mobile application that aims to streamline the party registration process as well as the warning issuing system from administrators to party hosts. By utilizing the functionality that Party Pass provides, users can be given chances to control their parties themselves before administrative forces interject. The benefit to administrators is that they do not have to waste as many resources by needlessly travelling to the party in order to issue a warning in person.

1.2 - Commercial Aspects

A symbiotic environment in which users can provide party information and where administrators can contact the party host(s) to issue warnings is very attractive to both parties involved. This relationship has a clear application to the registration of a house party, or even fraternity party, to the local police department. Many law enforcement agencies have limited resources due to budget or employee availability. By removing the unnecessary trips they take, it augments the available forces they have at any time. Party hosts will benefit from the peace of mind knowing they will have warnings before their parties are meet with personal interactions from law enforcement.

1.3 - Stakeholders

The stakeholders in this application include the developers, users, as well as any administrative body that has a need to monitor and control parties in their environment.

2 - Overall System Description

2.1 - Basic Functionality

A new user will be able to register for an account using a username and password, assuming both are at least 6 characters long and the username isn't already taken. Users will be able to log into the app by entering their username and password. If their account is not the admin account, then the app will open their personalized view containing the list of their registered parties. There is a single administrative account, which is created by the developers and distributed only to trusted administrators. If their account is the administrative account, the app will open the administrative view containing the list of all parties registered by all users. If they do not have an account, then the user will be able to register for a user account.

Users will be shown a list of their currently registered parties. They will be able to create a party, and each party will automatically expire once 24 hours have passed after the set end time of their event. The party registration information is the party's address, start time, end time, and the accounts for any other hosts associated to the party. If the other hosts do not have accounts, then their name and phone number can be entered instead. This registration information can be modified or deleted at any time if it has not already expired. Users will be able to logout and return back to the login page.

Administrators will be shown the list of parties that haven't expired yet, sorted oldest to newest by the start time of each party. They will also be able to filter the list by searching for a party filtered by the party address or by a host name associated to the party. When selecting a party from the list, a pop up will appear displaying the party information, the host(s) information, and an option to send a warning to the hosts with a personalized message. Admins will also have access to a map view containing pins with each party. When a pin is tapped, the same pop up as described before will appear, with the option to warn the party with a message. Administrators will be able to logout and return back to the login page.

2.2 - System Distribution

This is a standalone app that does not interact with any other products. The application is cross-platform, therefore it is built with the cross-platform frameworks of AngularJS and Ionic. For the functionality of the system, it interacts with a centralized database containing:

1) User accounts, each containing
    a) User account identification number
    b) Username
    c) Hash of password
    d) List of party identification numbers
2) Party registrations, each containing
    a) Party identification number
    b) Address of party
    c) Start time of party
    d) End time of party
    e) List of user identification numbers who are “hosts”
    f) List of names and phone numbers for “hosts” that do not have an account

2.3 - External User Types

User - view, add, edit, and remove party registrations

Administrator - view party registrations, either in a list or in a map, and issue warnings to party hosts

3 - Functional Requirements

3.1 - Use Cases

1. **Login Page**
    1.1. User registers for an account
    1.2. User/admin logs in
2. **User Interacts with Party Registrations**
    2.1. User registers a party
    2.2. User views parties in a list
    2.3. User views a party
    2.4. User edits a party
    2.5. User deletes a party
3. **Administrator Interacts with Party Registrations**
    3.1. Admin views parties in a list
    3.2. Admin views parties in a map
    3.3. Admin views a party
    3.4. Admin sends warning to party's hosts

Use Case ID:	UC_1.1
Use Case Name:	User registers for an account
Actor(s):	User
Description:	User makes a username and password that are associated with their account
Trigger:	User selects “sign up” on the login view
Preconditions:	User is on the login view
Postconditions:	User is on their main view
Normal Flow:	1) User selects “sign up” on the login view 2) Transition to sign up view 3) User enters desired username and password into fields 4) User selects “sign up” 5) Username and password for new account is stored in database 6) User is transitioned to their main view
Alternative Flows:	1) User selects “sign in” option and is returned to the sign in view
Exceptions:	1) The username is taken, so the user is prompted to choose a new one before attempting to sign up again 2) The username or password are not at least 6 characters long
Includes:	Mockup-1 (Login view) Mockup-2 (Sign up view) Mockup-3 (User main view)

Use Case ID:	UC_1.2
Use Case Name:	User/admin logs in
Actor(s):	User/admin
Description:	User/admin logs into the app
Trigger:	User/admin selects “sign in” button
Preconditions:	1) User/admin has an account 2) User/admin is on the login page 3) User/admin has entered both a username and password into login fields
Postconditions:	1) User is on their main view
Normal Flow:	1) User/admin selects “sign in” button 2) The entered credentials are verified to an account from the database. Specifically, the username must match to an account’s username and the hash of the entered password must match to the hash of the password stored in the database under that username’s account. 3) Transition to the user’s main view
Alternative Flow:	1) If the username/password are verified, but to the administrator account, then the admin is transitioned to the admin main view
Exceptions:	1) The entered credentials do not match an account, so the user is notified that no account matched their username/credentials
Includes:	Mockup-1 (Login page) Mockup-3 (User main view) Mockup-4 (Admin main view)

Use Case ID:	UC_2.1
Use Case Name:	User registers a party
Actor(s):	User
Description:	Users are able to submit their party information.
Trigger:	User selects “add party”
Preconditions:	User is logged in and on their main view
Postconditions:	Party’s information (address, start & end time, and host info) is stored in the database. The user is returned to their main view.
Normal Flow:	1. User selects “add party” button 2. “Register Modification view” pop up form appears 3. User fills out form with the event’s address, date, start time, end time, and hosts associated to the party. Hosts will be entered by their account or by their name and phone number if they do not have an account. 4. User taps the “Save changes” button. 5. User is returned to their main view
Alternative Flow:	1. User taps the “Cancel” button and is returned to their main view
Exceptions:	3. User does not enter valid information in the register party form: The “save changes” button will not be selectable and will remain greyed out until all requirements are filled in. Valid information is considered > 0 characters for their names, valid US phone numbers, valid time format, and a valid US address. There must not be more than 1 entry at the same address with overlapping start/end time as well.
Includes:	Mockup-3 (User main view) Mockup-4 (Register party view)

Use Case ID:	UC_2.2
Use Case Name:	User main list view
Actor(s):	User
Description:	User views all his registered parties in a list
Trigger:	User logs in.
Preconditions:	User is in their main view.
Postconditions:	User is in their main view
Normal Flow:	1. Use logs in 2. User is transitioned to their main view which contains a scroll menu of containing all registered parties
Alternative Flows:	1. If there are no active parties, the list will simply be empty
Exceptions:	-
Includes:	Mockup-3 (User main view)

Use Case ID:	UC_2.3
Use Case Name:	User views a party
Actor(s):	User
Description:	User will be able to view data of previously created party
Trigger:	User selects a cell representing a party
Preconditions:	User has created a party, and is on the home screen
Postconditions:	User is on the home screen
Normal Flow:	1. User selects a cell representing a party 2. User views the information 3. User selects “cancel” button 4. User is returned to their home screen
Alternative Flows:	1. User modifies and saves the information, refer to UC2.4
Exceptions:	
Includes:	Mockup-3 (User main view) Mockup-5 (Party modification view)

Use Case ID:	UC_2.4
Use Case Name:	User edits a party
Actor(s):	User
Description:	User will be able to modify data inputted for previously created parties
Trigger:	User selects a cell representing a party
Preconditions:	User has created a party, and is on the home screen
Postconditions:	User is on the home screen, modified information is stored
Normal Flow:	1) User selects a cell representing a party 2) User edits the text boxes containing their information 3) User selects “Update” button 4) User is returned to the home screen
Alternative Flows:	1) User selects “back” button, changes are lost and the user is returned to their home screen
Exceptions:	1. User is not able to save changes until all fields are filled in
Includes:	Mockup-3 (User main view) Mockup-5 (Party modification view)

Use Case ID:	UC_2.5
Use Case Name:	User deletes a party
Actor(s):	User
Description:	Users will be able to delete their party
Trigger:	User selects the “delete party” button within the “edit party” view
Preconditions:	User has created a party, and is currently in the “edit party” view.
Postconditions:	User is taken back to the home screen. Party entry is removed from database.
Normal Flow:	1. User selects “delete party” button 2. The party entry is removed from the database 3. User is taken back to home screen
Alternative Flows:	1. User selects “cancel”, and is transitioned to the home screen
Exceptions:	-
Includes:	Mockup-3 (User main view) Mockup-5 (Party modification view)

Use Case ID:	UC_3.1
Use Case Name:	Administrator views all registered parties in a list
Actor(s):	Admin
Description:	Admin is able to view a list of all registered parties
Trigger:	Admin logs in
Preconditions:	Admin is in main view.
Postconditions:	Admin is in main view.
Normal Flow:	1) Admin logs in 2) Transition to the admin main view which contains a scroll menu of containing all registered parties nearby.
Alternative Flows:	1) If there are no active parties, the list will simply be empty 2) Admin uses search bar to filter parties
Exceptions:	-
Includes:	Mockup-6 (Admin main view)

Use Case ID:	UC_3.2
Use Case Name:	Administrator map view
Actor(s):	Admin
Description:	Admin views a map with pins for all registered parties.
Trigger:	Admin selects the “map view” button.
Preconditions:	User is in Admin mode and is in their main view.
Postconditions:	Admin is able to view map of registered parties.
Normal Flow:	1. Admin selects the “view map” button from the home screen. 2. Application switches to screen with a map containing all registered parties
Alternative Flows:	-
Exceptions:	-
Includes:	Mockup-6 (Admin main view) Mockup-7 (Admin map view)

Use Case ID:	UC_3.3
Use Case Name:	Administrator views information for a party
Actor(s):	Admin
Description:	Admin will be able to view an individual party's contact information, time of party, and party location.
Trigger:	Admin selects the cell representing the party if in main view. If in map view then they select the pin representing the party from the map.
Preconditions:	Admin is in their main view or map view
Postconditions:	Admin is on party modification view
Normal Flow:	1. Admin selects a cell/pin representing the party 2. Application retrieves information about individual party (Host contact information, party location, time of party). 3. Application switches to party modification view.
Alternative Flows:	1) Admin selects "back" option and is returned to their main view / map view
Exceptions:	1) If party is deleted before the info page is rendered, then the admin will be notified that the party does not exist anymore
Includes:	Mockup-6 (Admin main view) Mockup-7 (Admin map view) Mockup-8 (Admin party view)

Use Case ID:	UC_3.4
Use Case Name:	Administrator warns party hosts
Actor(s):	Admin
Description:	Admin sends a warning to the hosts of the party
Trigger:	Admin selects “warn party” button from a party view
Preconditions:	Admin is on a party view
Postconditions:	Admin successfully sent the hosts an in-app warning
Normal Flow:	1. Application opens a popup window with an optional text entry to add to the warning 2. Admin selects send, to issue the warning. 3. Application sends warning message to the hosts of the party.
Alternative Flows:	1) Admin selects to cancel the warning message, and is returned to party view
Exceptions:	-
Includes:	Mockup-6 (Admin main view) Mockup-7 (Admin map view) Mockup-8 (Admin party view)

4 - Non-Functional Requirements

4.1 - Performance

This application will allow a user to register a party by entering the required information about their party. The user will be able to view a list of all of their party registrations. They will be able to add new parties, edit current parties, and delete parties. Administrators will have a single account that has certain credentials, and when logged in with their credentials they will have access to all parties for all users. They will also be able to send a warning message to a party's hosts through the app.

4.2 - Security

The stored information will be encrypted on a centralized database, ensuring that sensitive information is not stored locally unencrypted on each device. Also, the passwords will be stored as a hash and not as the password itself in case the encryption is broken. There is also only a single administrative account, ensuring that only privileged users will be able to access the sensitive data.

4.3 - External Standards

Upon creating an account, users will agree to the terms of service, which will be available to view online, which details how administrators will be able to view their party registration information and contact the hosts if desired.

5 - Future Requirements

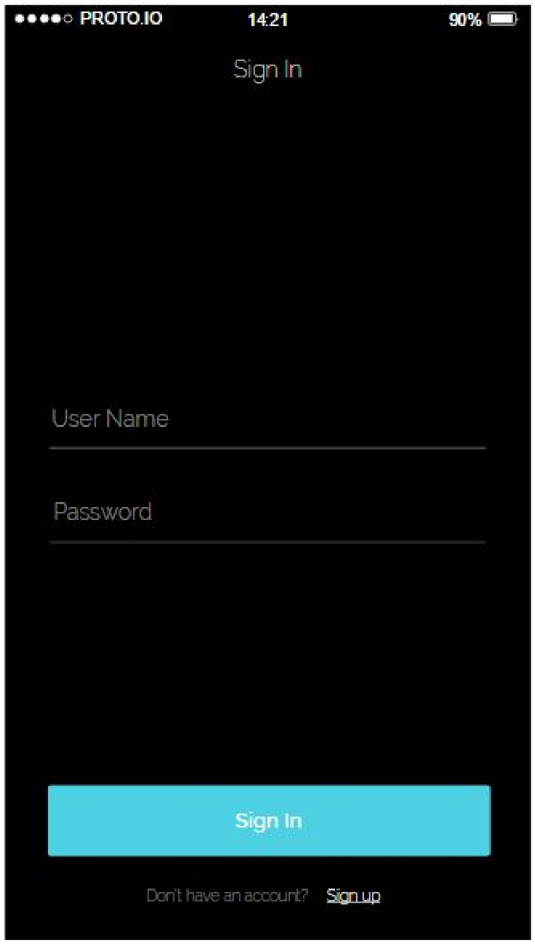
5.1 - Anticipated Features

5.1.1 - Users being able to retrieve/reset password if forgotten

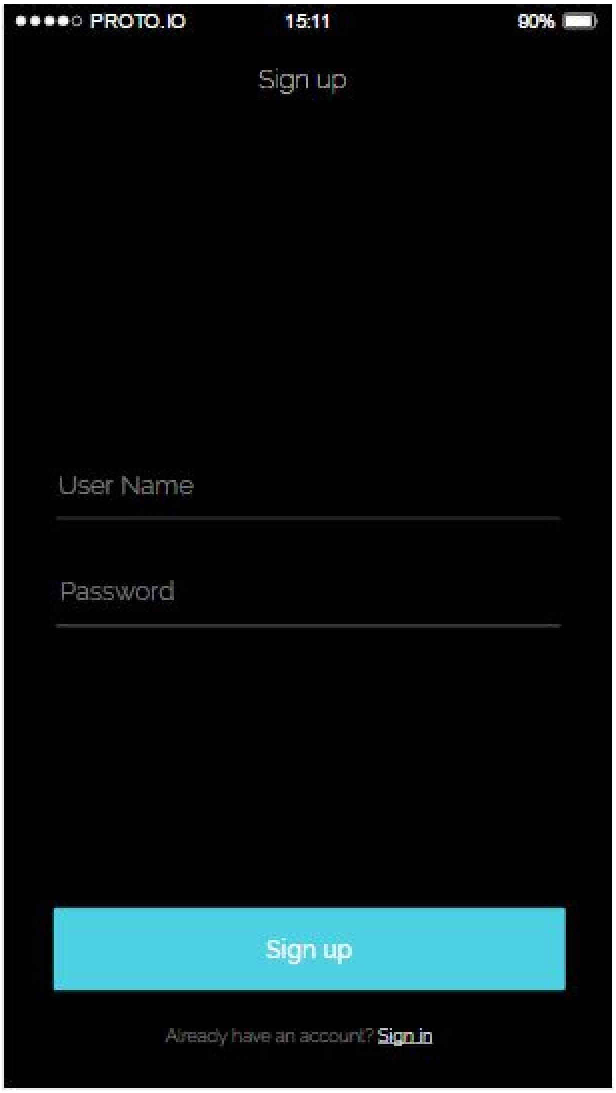
In future versions of the application, users may be able to request a password reset to their email. This would require the app to keep track of emails, validate them, and implement a password reset functionality. In the current version of the app, if a user forgets his password he must create a new account.

6 - User Interface Mockups

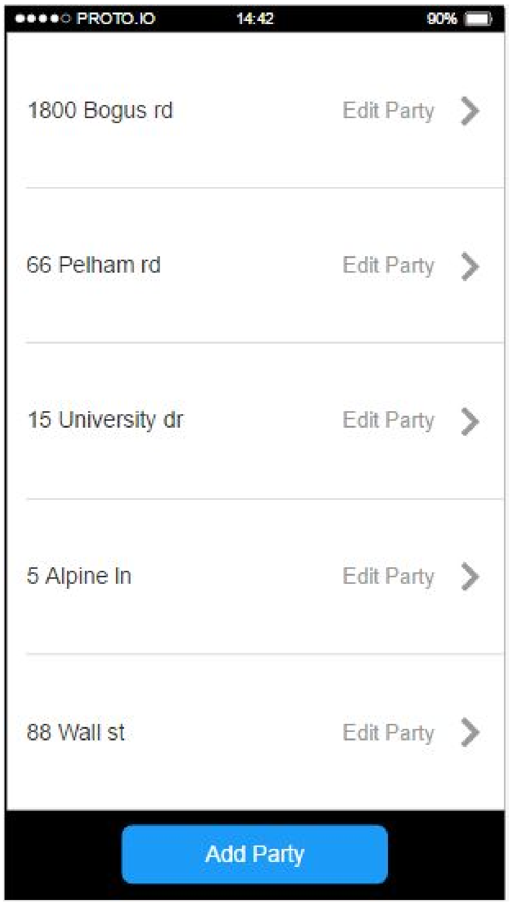
Mockup-1 - Login View



Mockup-2 - Sign Up View

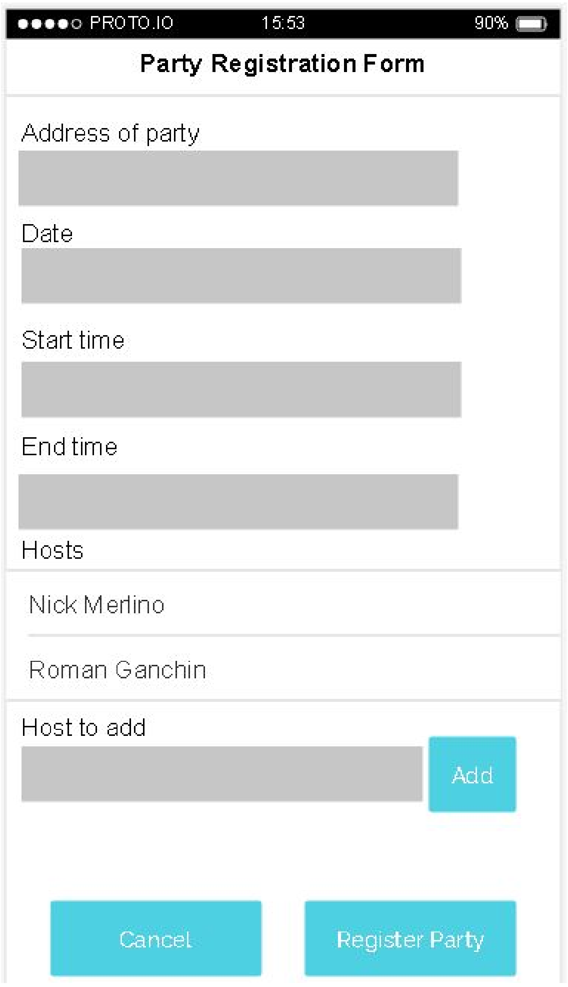


Mockup-3 - User Main View

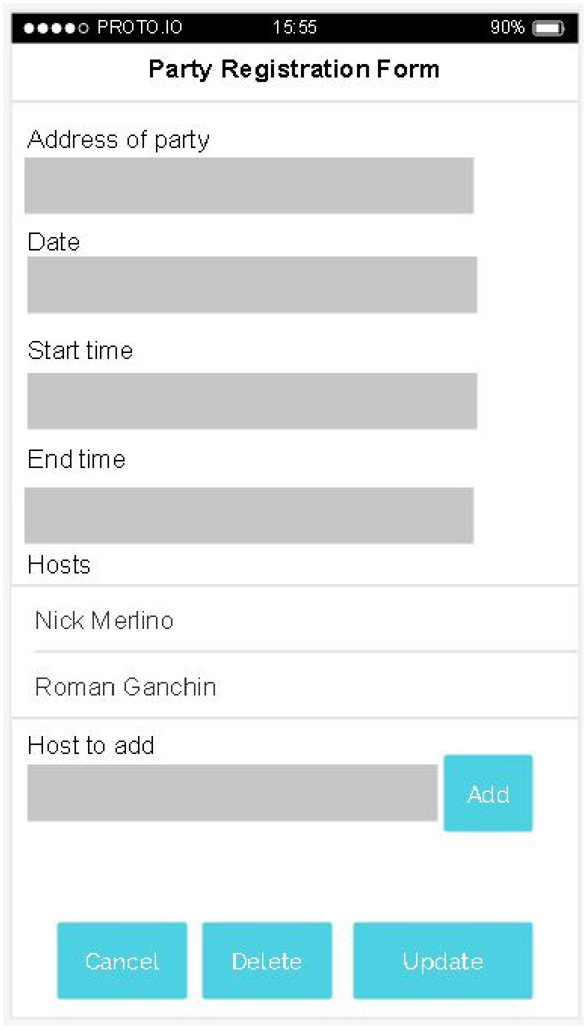


1800 Bogus rd — Edit Party
66 Pelham rd — Edit Party
15 University dr — Edit Party
5 Alpine ln — Edit Party
88 Wall st — Edit Party

Mockup-4 - Register Party View



Mockup-5 - Party View



Party Registration Form

Address of party

Date

Start time

End time

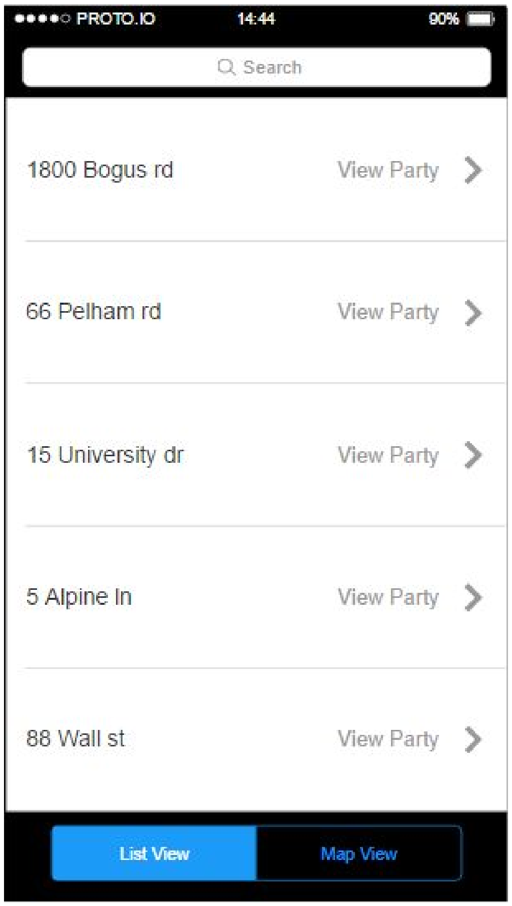
Hosts

Nick Merlino

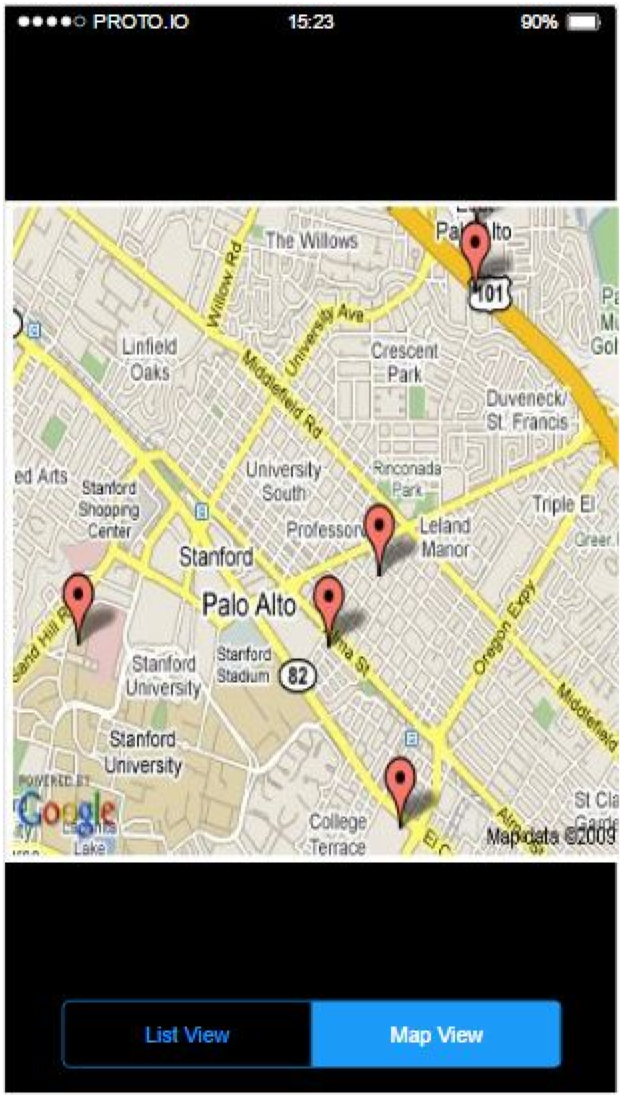
Roman Ganchin

Host to add

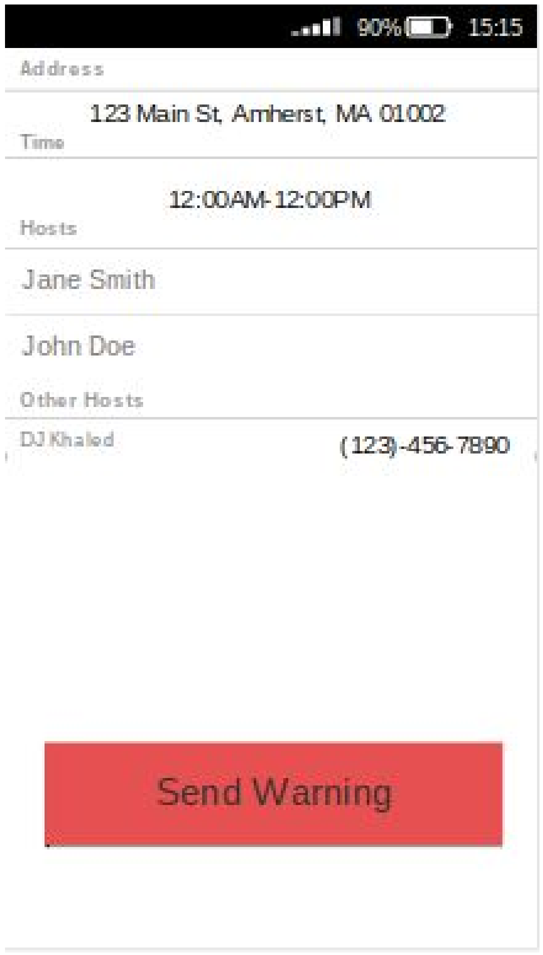
Mockup-6 - Administrator Main View



Mockup-7 - Administrator Map View



Mockup-8 - Administrator Party View



90% 15:15

Address

123 Main St, Amherst, MA 01002

Time

12:00AM-12:00PM

Hosts

Jane Smith

John Doe

Other Hosts

DJ Khaled (123)-456-7890

7 - Glossary

- **Framework** - An abstraction in which software can be easily utilized in order to extend upon its functionality
- **Centralized Database** - A single database that is in a controlled and secure environment, not on each device using the application
- **Party host** - One of the points of contact for the admin to issue warnings to. Their contact info is supplied on party registration.
- **Administrator -** The user type which has special privileges to view registered parties and send notifications to the users that have registered a party.
- **User** - Refers to someone who does not have administrative access but instead is a general user
- **Party registration** - The act of providing the address, start time, end time, and the phone numbers along with the accounts of other hosts or their names and phone numbers

8 - Resources

For the map view in the application we will look to use the Google Maps API. To account for the fact that the application should be cross-platform we will use the Ionic 2 framework. For the database we will use MongoDB and then NodeJS to eventually run the application. For testing, the combination of Karma and Jasmine will be used. Jasmine is the framework for the tests and Karma serves as the runner.