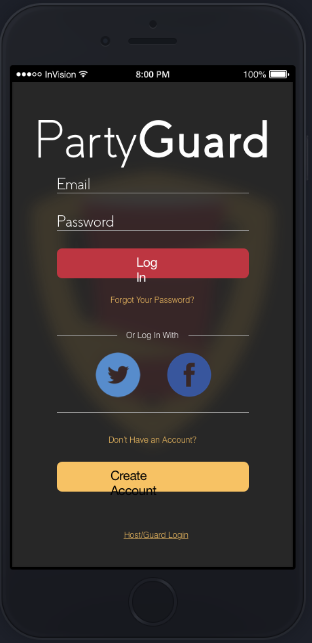
## **Use cases**

## **Basic User:**

## **Feature 1: Type of user**



**Introduction**

Based upon the person there are two types of users one is Host user (fraternity leader or member assigned to look after this app), regular member of the fraternity and person who guards the regular member.

**Inputs**

Select the button based on your classification whether a host, normal or guard user.

**Processing**

When a person clicks the appropriate button they will be presented by the respective login screen.

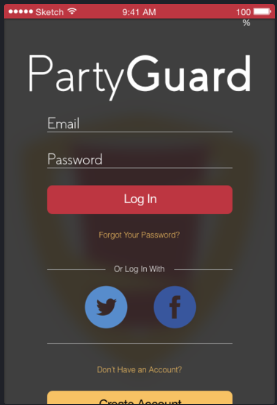
**Outputs**

The user will be redirected to their home page with successful login.

**User interface**

This screen has app logo, name and two buttons for the appropriate user selection.

## **Feature 2: Respective User Login**



**Introduction**

User is presented with login screen when pressed login page button and they are also given the option to login using their twitter, google, Facebook login credentials.

**Inputs**

* Email – It can contain special characters such as “@” and “.”. It should contain minimum 3 characters and maximum 15 characters length.
* Password – It can be alphanumerical, contain at least one uppercase letter, special character and number. It should contain minimum length of 8 and maximum length of 16. We will use encryption algorithms to encrypt the password and send it to the database.
* Can use third party for logging in such as twitter, google, Facebook.

**Processing**

* When user clicks forgot password they are taken to password reset page.
* When user uses third-party for logging in they are taken to the respective authentication pages.
* If they are new and doesn’t have an account they can create one using specific link provided.

**Outputs**

When user enters the username, password or uses third party to sign in and clicks login they are taken to the homepage if credentials are correct and matched.

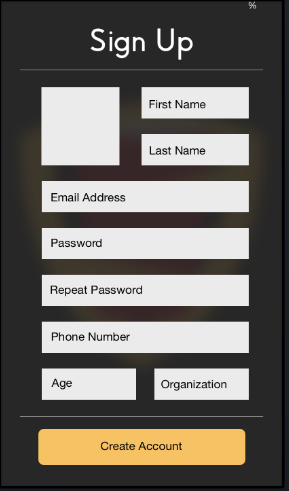
**Error Handling**

* If credentials are not met they are given a prompt, and if the user uses forgot password option the possible communication errors between application and database will be resolved by IOS and Web services team together by communicating the specific error log between each other and user is given respective prompt to avoid application crash.
* While using the third party sign in option the possible errors associated with the API are handled by getting appropriate error message from the API.

**User interface**

* Text fields for username and password, login/create account button with third party sign in options.

## **Feature 3: Registering new user**



**Introduction**

New user can create an account by entering the details listed in the picture.

**Inputs**

The details of the user include but not limited to

* Profile picture- Image of type .png or .jpg
* First Name – Should contain only alphabets
* Last Name - Should contain only alphabets
* Email - It can contain special characters such as “@” and “.”It should contain minimum 3 characters and maximum 15 characters length.
* Password - It can be alphanumerical, contain at least one uppercase letter, special character and number. It should contain minimum length of 8 and maximum length of 16. We will encrypt the password using encryption algorithm like RSA before sending to database.
* Phone number- Contains only numbers
* Age-Contains numerical values
* Organization-Contains only alphabets

**Processing**

When user enters all the details necessary the create account button is activated.

**Outputs**

When create account button is clicked their account is create and credentials are saved in database and email is sent to the user.

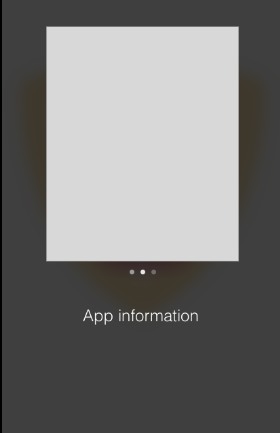
**Error Handling**

* To prevent the wrong input error, the user is given prompts about the input criteria while entering the information.
* When creating account if there is any network error user information will be stored in buffer and after resolving the network error the information will be fed to database for registration.
* If credentials are not according to the criteria the user will be given a prompt or dialog box till they satisfy the requirements.

**User interface**

In this screen, we have text fields for user details including photo upload field and a create account button.

**Feature 4: Application Guide**



**Introduction**

When user clicks application guide button or link they are provide with instructions on how to use the app with screenshots and a small video.

**Inputs**

Login information in previous page will be input over here.

**Processing**

Useful material is provided to the user which helps them understand on how to use the app.

**Outputs**

Images, documents, video.

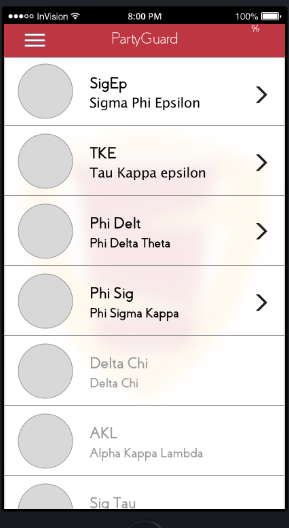
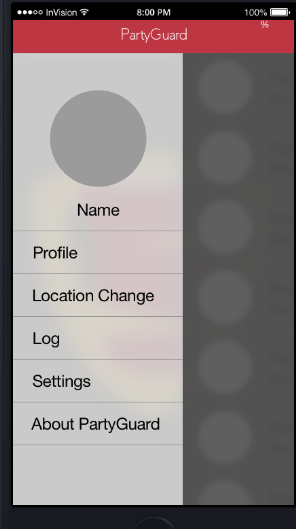
**Error Handling**

If images and videos are not loaded the application should provide progress bar or buffering bar in the meantime application will initiate the loading process again several times and if the error report shows the same error it will be displayed to user asking for some time to resolve it.

**User interface**

This user interface has slider with different pages containing text, documents and videos.

## **Feature 5: Home screen with menu**



**Introduction**

List of fraternities and organizations is given in the home page.

**Inputs**

User can edit his profile and choose a fraternity house.

**Processing**

Fraternities are enabled and fetched from database if they allow the usage in the app.

**Outputs**

Details about fraternity are generated.

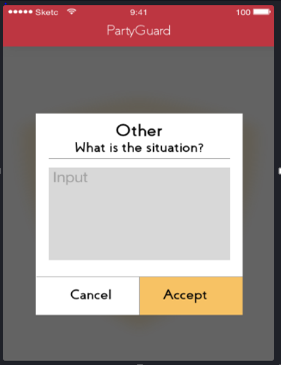
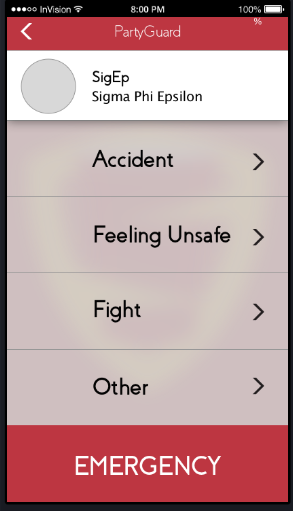
**Error Handling**

If the fraternity list is not loaded from the database, application will try to get the error report and will resend the call of request to database for getting the data.

**User interface**

We will have menu as a drop down option and the home screen has a clickable list view with different fraternities and organizations.

## **Feature 6: Issue Reporting**



**Introduction**

Various issues can be report by clicking the respective button and if the issue is not listed they can create a new issue.

**Inputs**

* Clicking respective issue link or creating new one.
* Clicking the emergency button as a last resort.
* Specify the situation if “Other” and click on accept.

**Processing**

When user click particular issue they can report it to the organization for action and if not addressed immediately they can use emergency button which will create a panic alarm and informs the host user& cops about the issue.

**Outputs**

* Fraternity is informed about the issue.
* Cops are informed in case of emergency.
* Dialog box with “Assistance is on the way!”

**Error Handling**

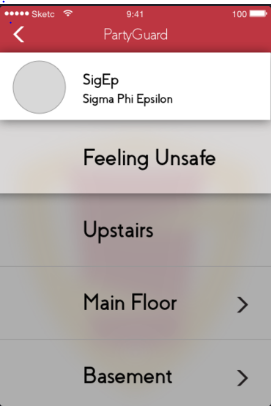
* If issue is not reported the app shouldn’t crash and should give appropriate message.
* The application will try to post the report several times till success and if presented with same error or different errors the user will be asked for some time.

**User interface**

Issue reporting screen also has a clickable list view with various issues and an emergency button. Once the issue is selected user will get a chance to select his/her location so that fraternity team locate the person reported in that location.

## 

## **Feature 7: User Location**



**Introduction**

When user wants to select the location where the incident is being taken place.

**Inputs**

User’s incident issue will be the input.

**Processing**

Respective action is performed when user chooses the location.

**Outputs**

Dialog box with the summary of issue, location and any comments mentioned will be popped up.

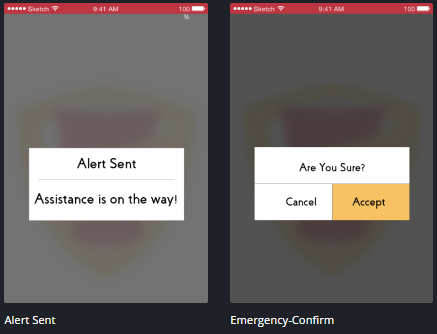
**Error Handling**

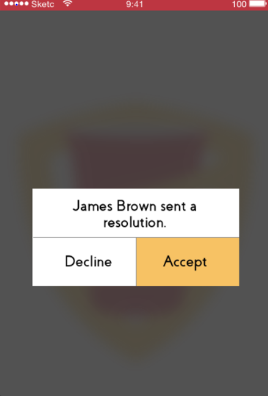
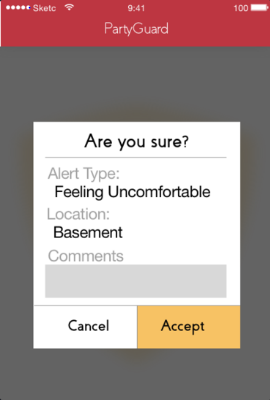
* The app shouldn’t crash if user selects undesired option.
* To prevent this error the user will be asked for confirmation as second step of authorization.

**User interface**

* Choose a location among the different locations within a fraternity.

## **Feature 8: Message prompts & Confirmation Dialogues**





**Introduction**

When a user commits a mistake/error or clicks something they are given prompts and confirmation dialog respectively.

**Inputs**

User either accepts or cancels the emergency confirmation message.

**Processing**

Respective action is performed when user hits accept or decline.

**Outputs**

Confirmation dialog box will appear according to the situation.

**Error Handling**

The error of selecting the undesired option is prevented by giving the user appropriate warning and confirmation messages/dialogue box.

**User interface**

* We have toast messages for different actions happening in the app which lets the user be aware of things going on.
* For taking confirmations we have dialog box prompt which has accept or cancel buttons with appropriate message in the dialog box.

## **Feature 9: Log of Incidents**

**Introduction**

User can see the list of incidents that had happened in the past.

**Inputs**

Can click a particular incident which will show the date of occurrence, person resolved, fraternity contacted at that time and incident details.

**Processing**

User will be directed to the appropriate information related to the incident.

**Outputs**

The user is presented with the details of date of occurrence, person resolved, fraternity contacted at that time and incident details.

**Error Handling**

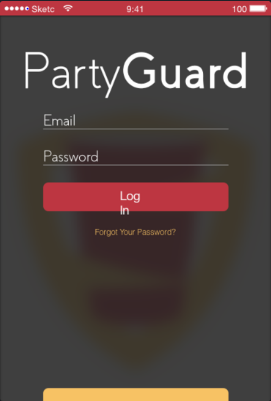
* The application will give a loading bar if log details are not loaded and meantime the loading initiation is triggered again till success and if error persists the user is given appropriate message and will be asked for some time for solving the issue.

**User Interface**

* It contains the incidents that were resolved by the fraternities in the past days.
* They are arranged in a sequential manner so that user can have a look at them in the glance.
* Clicking on an incident directs to the more details related to that particular incident.

**Host/Guard User:**

## **Feature 1: Host User Login**



**Introduction**

* When the user clicks on Host/Guard login on the first page he will be led to another page for the host user to login.

**Inputs**

* Host/Guard User must provide details such as username and password in order to login.
* Username: It can be alpha numerals. Should be a minimum of 5 characters to 16 characters.
* Password: It must contain alpha numerals and one special character. Should be minimum length of 8 characters and maximum length of 16 characters.

**Processing**

* Login details provided by the user should be validated.
* On logging in user is directed to PG Team page.

**Outputs**

* Host/Guard User will be directed to PG Team page on successful login otherwise an alert message with invalid login will pop up.

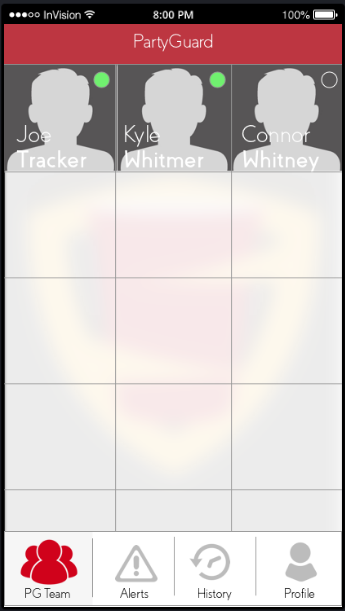
**Error Handling**

* If credentials are not met they are given a prompt, and if the user uses forgot password option the possible communication errors between application and database will be resolved by IOS and Web services team together by communicating the specific error log between each other and user is given respective prompt to avoid application crash.
* While using the third-party sign in option the possible errors associated with the API are handled by getting appropriate error message from the API.

**User interface**

* Text fields for username and password, login/create account button with third party sign in options.

## **Feature 2: PG Team**

****

**Introduction**

* On successful login user is directed to this page.
* All the guards with in the fraternity are available in this PG Team page.
* If host user clicks on a guard, then their profile shows up.
* “Green” circle on the guard shows that they are the members who responded for alerts.
* Host/Guard user can navigate through other tabs such as Alerts, History and Profile.

**Inputs**

* Host/Guard user chooses a guard to know his details.

**Processing**

* If user chooses a guard then their profile is displayed.

**Outputs**

* Profile of guard will be the output.

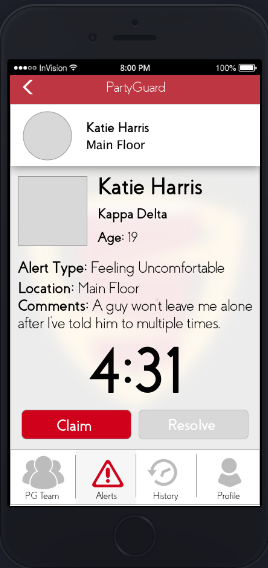
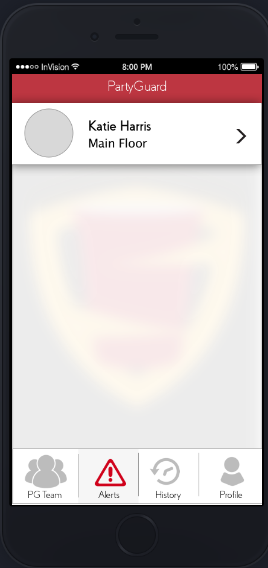
**Error Handling**

* Guards pertaining to that fraternity should be displayed, otherwise an error message pops up.

**User Interface**

* The PG team screen has thumbnail view with all the team member names, their picture along with their online status.

## **Feature 3: Alert Screen**



**Introduction**

Host/Guard user is navigated to Alerts tab when they click on it.

**Inputs**

Previous alerts are displayed and the user can select any to know in detail.

**Processing**

* When none of the guards is handling the alert, “claim” button will be enabled. Any member from the fraternity can claim the alert and confirm it.
* On claiming the alert, the fraternity member has to send the resolution confirmation request to the basic user by clicking on “Resolve” button.
* Host/Guard User either accepts or rejects the request.

**Output**

When the user accepts the request, then a confirmation message is shown up for fraternity member.

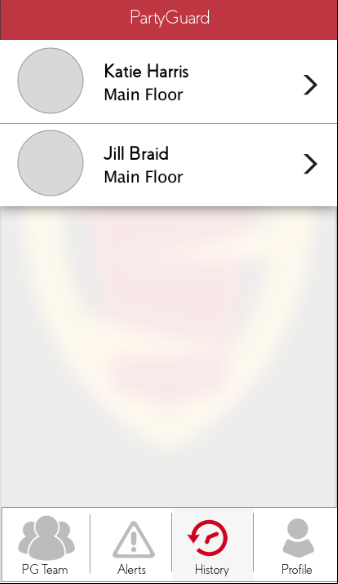
**Error Handling**

* Alert information should be available to user and if error occurs the information loading is done again till success or to find the error for resolving it.

**User Interface**

The alerts screen has clickable list view with all the alerts that host can select and look into.

## **Feature 4: History**



**Introduction**

Previous issues that are resolved are available in “History” tab.

**Inputs**

Host/Guard User clicks on the “History” tab to check out previous resolved issues.

**Processing**

When the basic user confirms that the issue was resolved, only then that issue will be displayed under “History” tab.

**Outputs**

List of resolved issues appears in this tab.

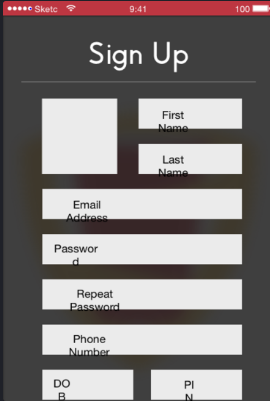
**Error Handling**

If no issues were available that are resolved then application will try to re fetch the details till success or the user is asked for some time if error is strange and doesn’t resolve by itself.

**User Interface**

History screen has clickable list view with all the events reported and resolved in the past.

## **Feature 5: Host/Guard Profile Screen**



**Introduction**

Host/Guard User can edit his profile when clicked on the “Profile” tab.

**Inputs**

Host/Guard User could edit his profile by providing any of the following details.

* Profile picture- Image of type .png or .jpg
* First Name – Should contain only alphabets
* Last Name - Should contain only alphabets
* Email Address- It can contain special characters such as “@” and “.”It should contain minimum 3 characters and maximum 15 characters length.
* Password - It can be alphanumerical, contain at least one uppercase letter, special character and number. It should contain minimum length of 8 and maximum length of 16. We will encrypt the password using encryption algorithm like RSA before sending to database.
* Phone number- Contains only numbers
* DOB and PIN-Can contain only numbers.

**Processing**

* Host/Guard user can edit his profile.
* They can have a profile picture too.

**Outputs**

Host/Guard User details will be available.

**Error Handling**

* To prevent the wrong input error, the user is given prompts about the input criteria while entering the information.
* When creating account if there is any network error user information will be stored in buffer and after resolving the network error the information will be fed to database for registration.
* If credentials are not according to the criteria the user will be given a prompt or dialog box till they satisfy the requirements.

**User Interface**

The host profile screen has the picture of the host with details like name, email address, contact info, pin.