

Functional Specification v4

Marked Private

Creative360

Friday 27th May, 2016

Contents

1	Prototypes	4
1.1	Mobile App - UX	4
1.2	Mobile App - UI	4
2	Mobile Application views	5
2.1	General Interaction	5
2.2	Loader View	5
2.2.1	Design	5
2.2.2	Interactions	6
2.3	First View	7
2.3.1	Design	7
2.3.2	Interactions	7
2.3.3	API	7
2.4	Sign in View (Guest, Host)	8
2.4.1	Design	8
2.4.2	Interactions	8
2.4.3	API	8
2.5	Reset Password View (Guest, Host)	9
2.5.1	Design	9
2.5.2	Interactions	9
2.5.3	API	9
2.6	Join an Event View (Guest)	10
2.6.1	Design	10
2.6.2	Interactions	10
2.6.3	API	11
2.7	Start an Event View (Host)	12
2.7.1	Design	12
2.7.2	Interactions	12
2.7.3	API	13
2.8	Adding Another Code View (Host)	14
2.8.1	Design	14
2.8.2	Interactions	14
2.8.3	API	15
2.9	Verifying Code View (Guest)	16
2.9.1	Design	16
2.9.2	Interactions	16
2.9.3	API	16
2.10	Verifying Codes View (Host)	17

2.10.1	Design	17
2.10.2	Interactions	17
2.10.3	API	18
2.11	Sign Up View (Blank) (Guest, Host)	19
2.11.1	Design	19
2.11.2	Interactions	20
2.11.3	API	20
2.12	Sign Up View (Filled) (Guest, Host)	21
2.12.1	Design	21
2.12.2	Interactions	22
2.12.3	API	22
2.13	Starting an Event View (Host)	23
2.13.1	Design	23
2.13.2	Interactions	23
2.13.3	API	24
2.14	Location Address View (Host)	25
2.14.1	Design	25
2.14.2	Interactions	25
2.14.3	API	25
2.15	Set Date View (Host)	26
2.15.1	Design	26
2.15.2	Interactions	26
2.15.3	API	27
2.16	Communicator View (Guest)	28
2.16.1	Design	29
2.16.2	Interactions	29
2.16.3	API	30
2.17	Information about Event View (Guest)	31
2.17.1	Design	31
2.17.2	Interactions	31
2.17.3	API	32
2.18	Communicator View (Host, Blank)	33
2.18.1	Design	33
2.18.2	Interactions	34
2.18.3	API	34
2.19	Communicator View (Host)	35
2.19.1	Design	35
2.19.2	Interactions	36
2.19.3	API	36
2.20	Live Stream View (Host, Guest)	37
2.20.1	Design	37
2.20.2	Interactions	38
2.20.3	API	38
2.21	Options View (Guest, Host)	39
2.21.1	Design	39
2.21.2	Interactions	39
2.21.3	API	40
2.22	Privacy and Terms View (Guest, Host)	41

2.22.1	Design	41
2.22.2	Interactions	41
2.23	Help View (Guest, Host)	42
2.23.1	Design	42
2.23.2	Interactions	42
2.24	Edit Profile View (Guest, Host)	43
2.24.1	Design	43
2.24.2	Interactions	44
2.24.3	API	44
2.25	Changing Password View	45
2.25.1	Design	45
2.25.2	Interactions	45
2.26	Confirming New Password View	46
2.26.1	Design	46
2.26.2	Interactions	46
2.26.3	API	46
2.27	Edit Event Details View	47
2.27.1	Design	47
2.27.2	Interactions	48
2.27.3	API	48
2.28	Categories of Invited Guests View	49
2.28.1	Design	49
2.28.2	Interactions	49
2.28.3	API	50
2.29	Messages View	51
2.29.1	Design	51
2.29.2	Interactions	51
2.29.3	API	51
2.30	New Message View (Guest, Host)	52
2.30.1	Design	52
2.30.2	Interactions	52
2.30.3	API	53
2.31	1-1 Message View (Guest, Host)	54
2.31.1	Design	54
2.31.2	Interactions	54
2.31.3	API	54
2.32	Media View	55
2.32.1	Design	55
2.32.2	Interactions	55
2.32.3	API	55
3	Non-functional Requirements	56
3.1	Mobile Application	56

Chapter 1

Prototypes

1.1 Mobile App - UX

<https://invis.io/T96P5B3YF>

Password: *creative360*

1.2 Mobile App - UI

<https://invis.io/Y96D6DM4V>

Password: *creative360*

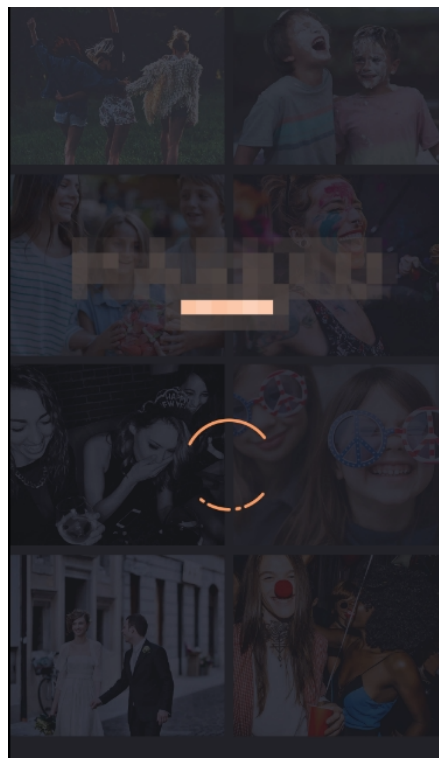
Chapter 2

Mobile Application views

2.1 General Interaction

If User force closes the app it then opens on the last event user was part of. If User closes the app without force closing it from the Task Manager than the app opens on the last screen he was on (usual iOS app behavior).

2.2 Loader View



2.2.1 Design

1. Logo
2. Loader

2.2.2 Interactions

User sees Loader View every time he loads the app.

2.3 First View

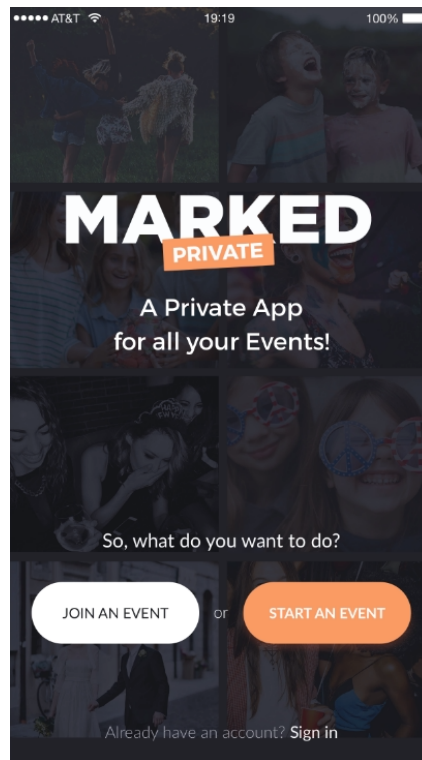


Figure 2.1: Design 1.

2.3.1 Design

1. Logo
2. Join an Event Button
3. Start an Event Button
4. 'Sign in' hyperlink

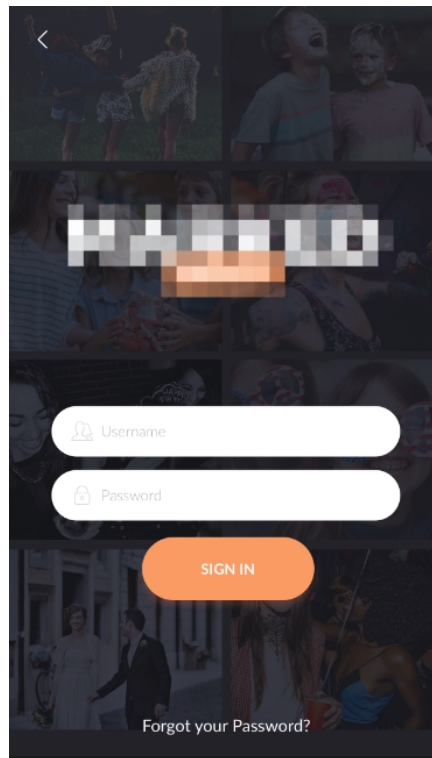
2.3.2 Interactions

The First View shows the application's two action buttons, that enable the user to join an existing event or starting a new one. The 'Sign In' hyperlink is also available in the bottom of the screen.

2.3.3 API

1. Join an event - verification for guest code.
2. Start an event - verification for host code.
3. Sign in - verification for username and password.

2.4 Sign in View (Guest, Host)



2.4.1 Design

1. Logo
2. Username Text Field
3. Password Text Field
4. Sign in Button
5. 'Forgot your password' hyperlink
6. Back Button

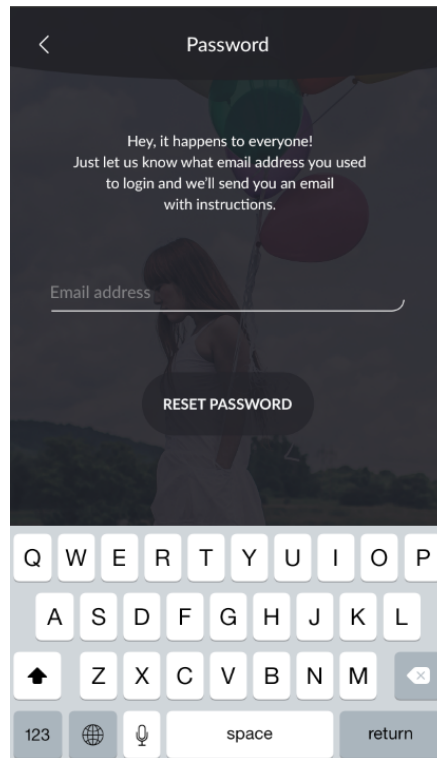
2.4.2 Interactions

After clicking on 'Sign In' hyperlink, application checks if: Username and Password is correct. If it is, application goes to the last event User was on. If it isn't, application shows information about incorrect input data. In case the password is incorrect, User can use "Forgot password" option. By clicking on the Back Button User goes back to First View.

2.4.3 API

1. Verification for username and password.
2. Forgot password - send reset link to user.

2.5 Reset Password View (Guest, Host)



2.5.1 Design

1. Reset password title
2. Description text
3. Email Text Field
4. Reset password button
5. Back button

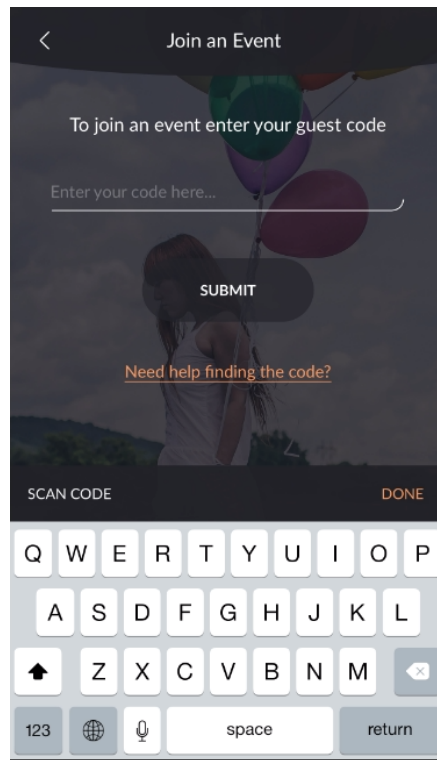
2.5.2 Interactions

After clicking on 'Forgot password' hyperlink, User goes to this screen. He enters his registered email and clicks Reset password button. After that he gets an email with password reset link that opens the screen for resetting the password. By clicking on the Back Button User goes back to Sign In View.

2.5.3 API

1. Verification for email.
2. Forgot password - send reset link to user.

2.6 Join an Event View (Guest)



2.6.1 Design

1. 'Join an Event' title
2. Enter your code Text Field
3. Submit Button
4. 'Need help' hyperlink
5. 'Scan code' Button
6. 'Done' Button
7. (Optional) 'Back' Button
8. (Optional) 'Menu' Button

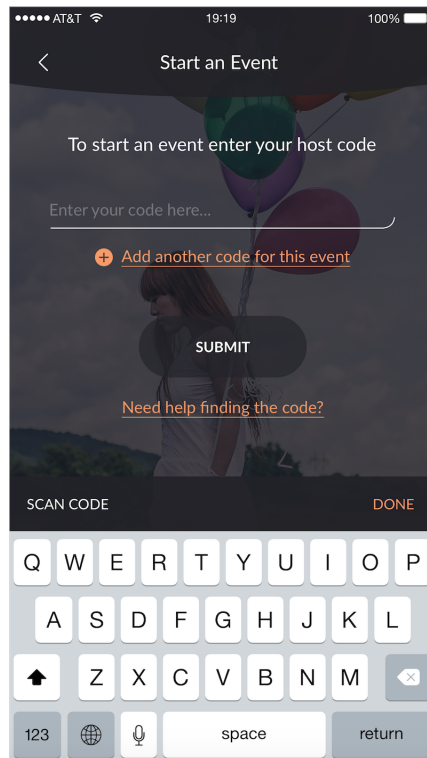
2.6.2 Interactions

After clicking on 'Join an event' button the application opens 'Join an event' view where Guest can enter the Guest code. Guest can also scan a code with his camera by pressing 'Scan Code' and giving access to his camera. By clicking 'Submit' button guest can proceed to the next step. By clicking 'Done' the keyboard closes. If Guest came from the Sign in screen by clicking on the Back Button he goes back to that Sign In screen. If Guest came from the menu he sees a hamburger Menu Button that he can click to go to other screens. If Guest presses on Need Help hyperlink he is navigated to Help screen with How to find the Guest code answer open.

2.6.3 API

1. Verification for guest code

2.7 Start an Event View (Host)



2.7.1 Design

1. 'Start an Event' title
2. Host code Text Field
3. 'Add another code' hyperlink
4. Submit Button
5. 'Need help finding the code' hyperlink
6. Scan code Button
7. Done Button
8. Back Button

2.7.2 Interactions

IMPORTANT: 1. screen should say "To start an event enter your host code", instead of "guest code". 2. By default there should be no Host code text field 2, only 1 field.

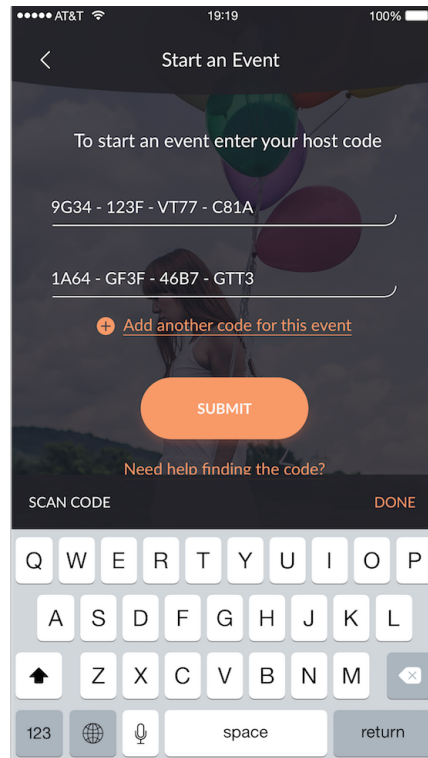
After clicking on 'Start an event' button the application goes to 'Start an event view' where Host can enter 1 or more host codes to be able to start an event. By clicking 'Add another code...' a new Host code Text Field is added below the last Text Field. If a Host accidentally added a Text Field but doesn't have a code for it (like on the screenshot) the app will let him in

anyway. Host can also scan a code with his camera by pressing 'Scan Code' and giving access to his camera. The code will then be autofilled in the code field. By clicking 'Submit' button Host can proceed to the next step. By clicking 'Done' the keyboard closes. If Host came from the Sign in screen by clicking on the Back Button he goes back to that Sign In screen. If Host came from the menu he sees a hamburger Menu Button that he can click to go to other screens. If Host presses on Need Help hyperlink he is navigated to Help screen with How to find the Host code answer open.

2.7.3 API

1. Verification for host code

2.8 Adding Another Code View (Host)



2.8.1 Design

1. 'Start an Event' title
2. Host code Text Field
3. 'Add another code' hyperlink
4. Submit Button
5. 'Need help finding the code' hyperlink
6. Scan code Button
7. Done Button
8. Back Button

2.8.2 Interactions

After clicking on 'Start an event' button the application goes to 'Start an event view' where Host can enter 1 or more host codes to be able to start an event. By clicking 'Add another code...' a new Host code Text Field is added below the last Text Field. If a Host accidentally added a Text Field but doesn't have a code for it (like on the screenshot) the app will let him in anyway. Host can also scan a code with his camera by pressing 'Scan Code' and giving access to his camera. By clicking 'Submit' button Host can proceed to the next step. By clicking

'Done' the keyboard closes. If Host came from the Sign in screen by clicking on the Back Button he goes back to that Sign In screen. If Host came from the menu he sees a hamburger Menu Button that he can click to go to other screens. If Host presses on Need Help hyperlink he is navigated to Help screen with How to find the Host code answer open.

2.8.3 API

1. Code(s) verification

2.9 Verifying Code View (Guest)

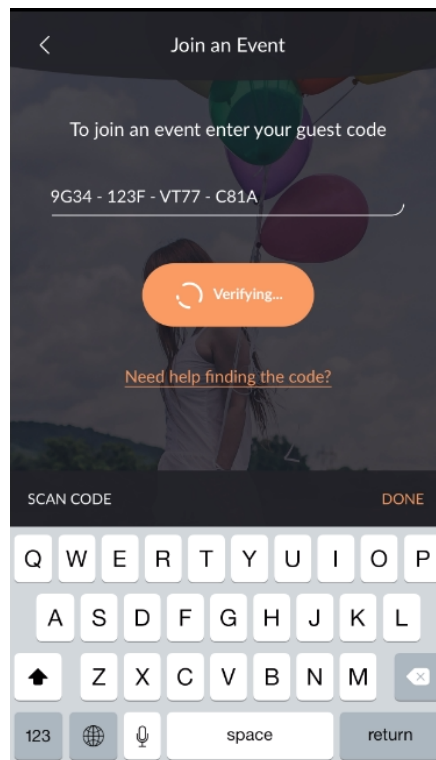


Figure 2.2: Verifying Code View

2.9.1 Design

1. 'Join an Event' title
2. Enter your code Text Field
3. Verifying Loader
4. 'Need help' hyperlink
5. Scan code Button
6. Done Button
7. Back Button

2.9.2 Interactions

After entering the Guest code, the application verifies the code. The process is depicted by Verifying Loader. All buttons and links are inactive when we verify the code(s).

2.9.3 API

1. Guest code verification

2.10 Verifying Codes View (Host)

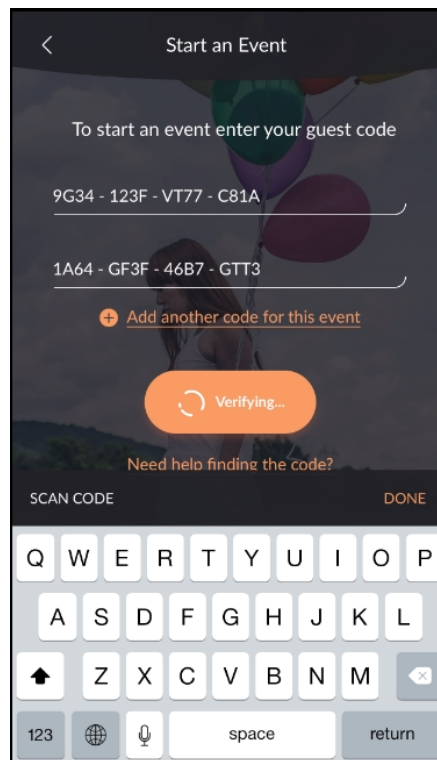


Figure 2.3: Verifying Code View

2.10.1 Design

1. 'Start an Event' title
2. Enter your code Text Field
3. Add another code Text Field
4. 'Add another code' hyperlink
5. Verifying Loader
6. 'Need help finding the code' hyperlink
7. Scan code Button
8. Done Button
9. Back button

2.10.2 Interactions

After entering the codes into the Text Fields the application verifies the Host code(s). The process is depicted by Verifying Loader. All buttons and links are inactive when we verify the code(s).

2.10.3 API

1. Host code(s) verification

2.11 Sign Up View (Blank) (Guest, Host)

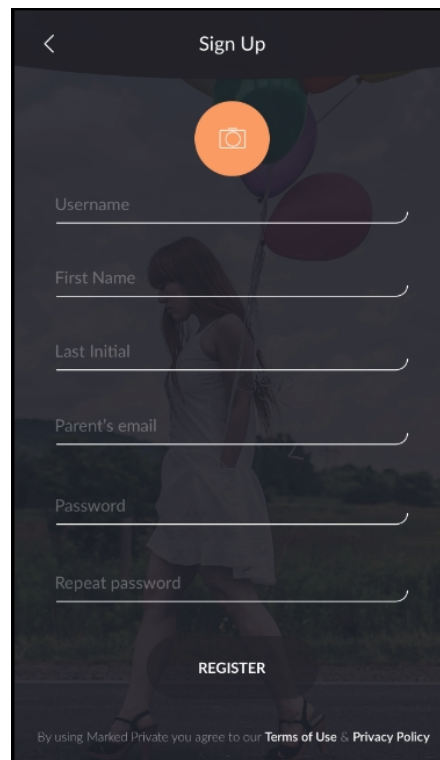


Figure 2.4: Sign Up View

2.11.1 Design

1. 'Sign up' title
2. Take photo / Choose from Camera Roll Button
3. Username Text field
4. First name Text field
5. Last name Text field
6. Email Text field
7. Password Text field
8. Repeat password Text field
9. Register Button
10. Privacy and Terms link

2.11.2 Interactions

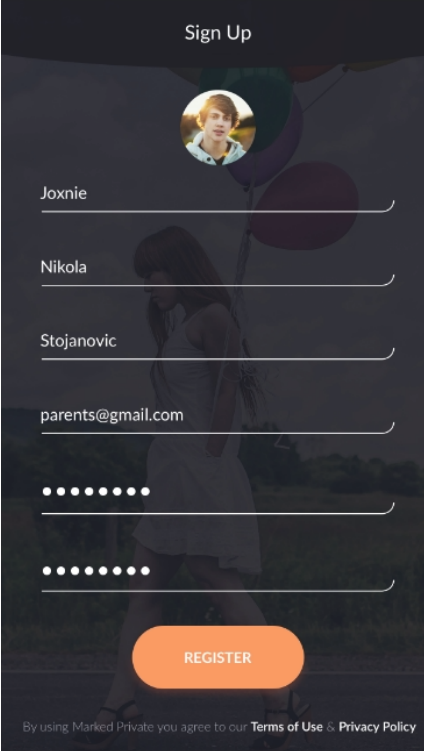
After entering the required data into all text fields and clicking on 'Register' button, the application signs up the new user and goes to the next page. After clicking on Take photo / Choose from Camera Roll button adding user's profile picture is available. By clicking on 'Privacy and Terms' link User sees a Privacy and Terms screen. The 'Register' button is inactive until all the fields are filled in. The avatar is optional. If user doesn't add an avatar we show a generic placeholder avatar. There's no back button on this screen, as the codes have already been redeemed.

Notes: UI shows 'Privacy and ToS', but it should say 'Privacy and Terms'. Order should be first name, last name, email, username, password, repeat password.

2.11.3 API

1. Sign up - verification for username, first name, last initial, parent's email and password.

2.12 Sign Up View (Filled) (Guest, Host)



Sign Up

Joxnie

Nikola

Stojanovic

parents@gmail.com

.....

.....

REGISTER

By using Marked Private, you agree to our [Terms of Use](#) & [Privacy Policy](#)

Figure 2.5: Register View

2.12.1 Design

1. 'Sign up' title
2. Take photo / Choose from Camera Roll Button
3. Username Text field
4. First name Text field
5. Last name Text field
6. Email Text field
7. Password Text field
8. Repeat password Text field
9. Register Button
10. Privacy and Terms link

2.12.2 Interactions

After entering the required data into all text fields and clicking on 'Register' button, the application signs up the new user and goes to next screen. After clicking on Take photo / Choose from Camera Roll button adding user's profile picture is available. By clicking on 'Privacy and Terms' link User sees a text + OK button popup. The 'Register' button is inactive until all the fields are filled in. The avatar is optional. If user doesn't add an avatar we show a generic placeholder avatar. There's no back button on this screen, as the codes have already been redeemed. After clicking Register Button Guest goes to the respective event, and Host can go to setting up the new event.

Note: UI shows 'Privacy and ToS', but it should say 'Privacy and Terms'. Order should be first name, last name, email, username, password, repeat password

2.12.3 API

1. Create new user
2. Return registration state

2.13 Starting an Event View (Host)

Start an event

Event name e.g., John's Party*

Location name e.g., Home

Location Click to Select >

Start time Click to Select >

End time Click to Select >

Anything else people should know (e.g., Bring a sleeping bags) optional

START AN EVENT

Figure 2.6: Starting an Event View

2.13.1 Design

1. 'Start an Event' title
2. Event name text field
3. Location select Button
4. Time from Button
5. Time to Button
6. Event details text field
7. Start Event Button

2.13.2 Interactions

IMPORTANT: Should be 'Start an Event', not 'Start an event'. After entering the Event's Name, Location, Time and other details into Text Fields the application creates a new event. When user presses Location it should go to Location Address View (Host). All the fields except for event name are optional. When Host clicks on Start time and End time buttons he sees a date and time picker (<http://i.stack.imgur.com/n9gLq.png>), where he can pick date and time and press done to hide it. Start an event button is inactive until all the necessary fields are filled in.

2.13.3 API

1. Create event

2.14 Location Address View (Host)

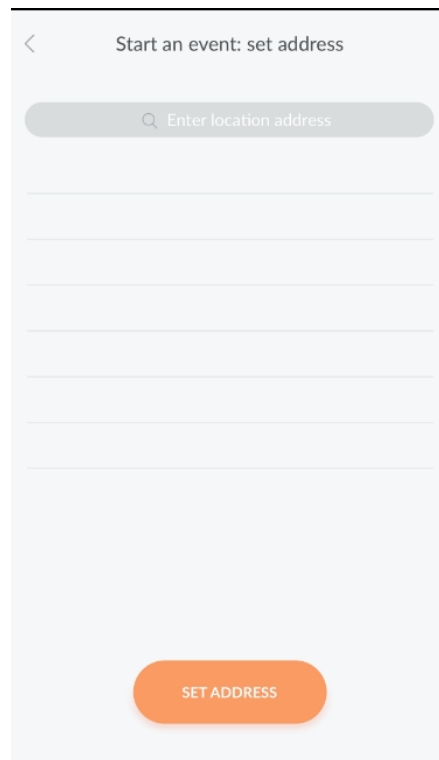


Figure 2.7: Location Address View

2.14.1 Design

1. Enter location address input field
2. List of locations (blank)
3. Set Address Button
4. Back Button

2.14.2 Interactions

IMPORTANT: Should be 'Start an Event: Set Address', not 'Start an event: set address'. After find the location's address the application sets it as a location of a created event. In this view we can also set detailed events location. Set Address button is inactive until the location's address is selected. By clicking on Back Button Host goes back to Starting an Event View.

2.14.3 API

1. Create event - address
2. Implemented with Apple maps API

2.15 Set Date View (Host)

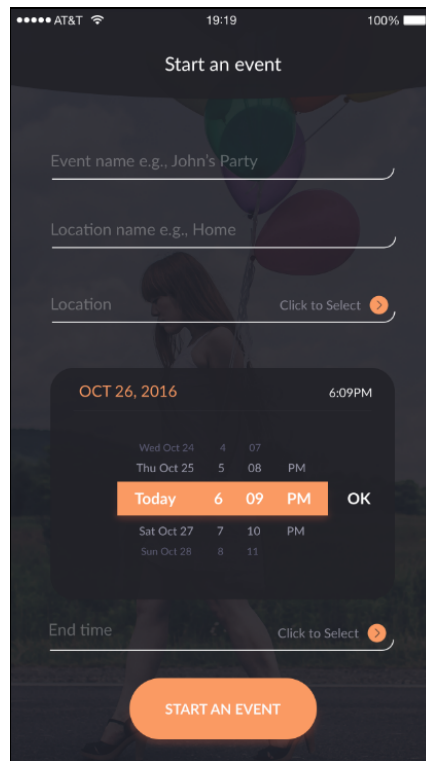


Figure 2.8: Set Date View

2.15.1 Design

1. 'Start an event' title
2. Event name Text field
3. Location button
4. Time from Button
5. Time to Button
6. Start an event Button

2.15.2 Interactions

IMPORTANT: Should be 'Start an Event', not 'Start an event'.

In this section the date of the newly created event is being set. After entering the Event's Name and Location the event's date and time can be set by using 'Click to Select' button and choosing the 'from' (Time from Button) 'to' (Time to Button) dates. When Host clicks on Start time and End time buttons he sees a date and time picker (<http://i.stack.imgur.com/n9gLq.png>), where he can pick date and time and press OK to hide it. Clicking on 'Start an Event' button saves all the aforementioned info, creates an event and redirects to Event's home screen. Start an event button is inactive until all the required fields are filled. Event's time can be chosen by picking the start and end time, they will be displayed with pm/am.

2.15.3 API

1. Create event

2.16 Communicator View (Guest)

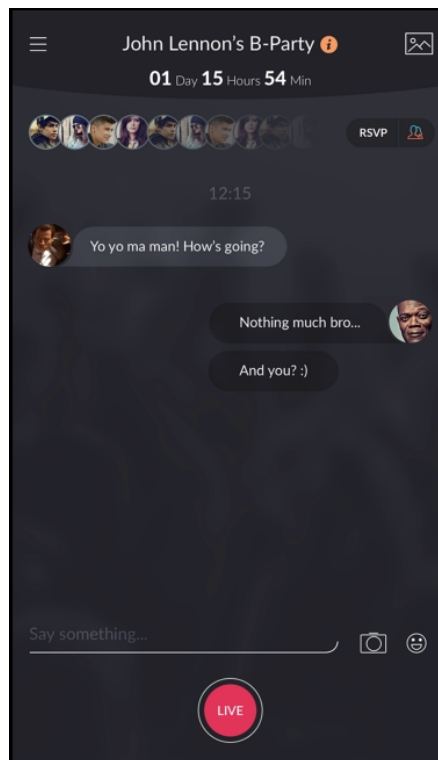


Figure 2.9: Communicator View, (Guests 1)

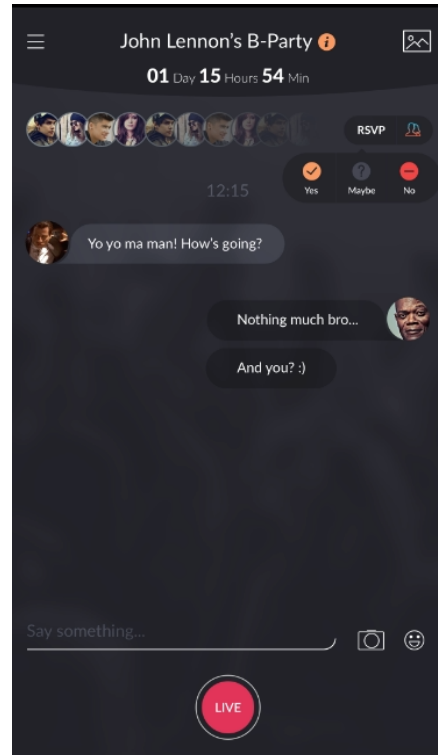


Figure 2.10: Communicator View, (Guests 2)

2.16.1 Design

1. Countdown
2. 'Name of event' title
3. List of invited guests
4. RSVP Button
5. Space for messages
6. Media button
7. Say something... Text field
8. Send photo Button
9. Send emoticon Button
10. Live stream Button
11. Info Button
12. Menu Button

2.16.2 Interactions

Upon registering Guests see the chat screen where system immediately automatically messages them inviting them to the event. Messages arrive one after another with a 1 sec interval. Messages are compiled as templates from the even info that the Host indicated when creating the event. Sample messages plot: 1. "Hi [name]! Welcome to [event name]." 2. "It will take place on [event date] at [location]." 3. (Optional, depends on whether Host entered additional event details) "Some additional info: [event details]." 4. "Can't wait to see you. Are you coming? Please RSVP now to join in." After message 4. RSVP opens to signal a Guest where he can RSVP. After RSVP-ing Guests can see the group chat, if any, right below the automated messages. By clicking on a 'Guests' icon Guest opens a Who's coming screen. By clicking on 'RSVP' button Guest can RSVP to the event, with Yes / No / Maybe answer. (<https://invis.io/9Z75BJ0AK>). By clicking on the avatars of people Guest opens a Who's coming screen. (You can click around here: <https://invis.io/7V75BE86Z>.) This screen is a group chat that any Guest or Host can see. New messages can be posted by clicking on the 'Say something' Text field and clicking "Send" on the keyboard. Additionally, after clicking on the 'Photo' icon a Guest can Take a photo / video or choose it from Camera Roll; after clicking on the 'Emoticon' icon a Guest can choose among standard emojis. Sending emojis are animated Periscope-style (<http://browniefed.com/blog/react-native-periscope-hearts-animation/>). By clicking on 'Live' button Guest can start a live stream that all other Guests and Hosts can see. The Communicator View then changes to full-screen live video stream. You can switch between live stream and Communicator view by tapping anywhere on the chat. You can control the stream with pause / stop buttons. On the top of the screen you can see the duration of the stream. Guests can also click on the 'Media' button to see a list of media shared, if any. The last image shared in the chat automatically becomes the background of the Communicator View. After the Live stream is over the full-screen live video stream changes to the Communicator View.

2.16.3 API

1. Implemented with chat SDK

2.17 Information about Event View (Guest)

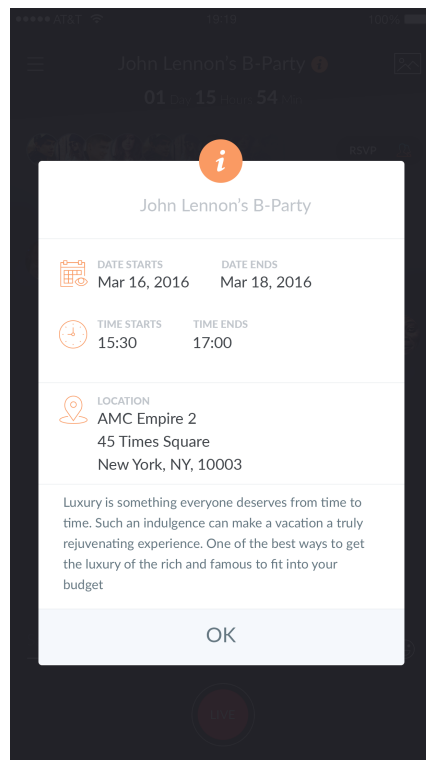


Figure 2.11: Information about Event View

2.17.1 Design

1. Name of Event
2. When Event is Starting
3. When Event is Ending
4. Event's Location
5. Details about Event
6. OK Button

2.17.2 Interactions

IMPORTANT: design doesn't reflect this but Date line should say: "Date: March 5th - March 29th", time line should say: "Time: 1:00pm - 2:00pm" (we use AM / PM). If event lasts one day there should be no end date block.

This view opens from a Guest Chat View by clicking on the 'i' icon near the title of the party. In this section the general information about the event is displayed. Clicking 'OK' button closes this section.

2.17.3 API

1. Return event details

2.18 Communicator View (Host, Blank)

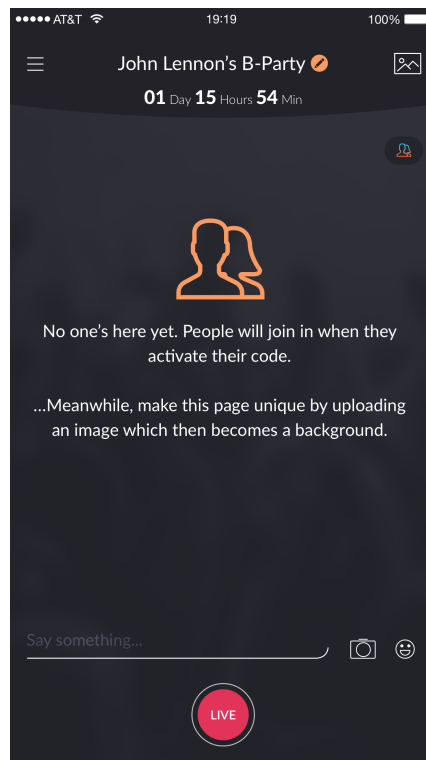


Figure 2.12: Communicator View (Host, Blank)

2.18.1 Design

1. Event name
2. Countdown to Event
3. Media Button
4. Space for invited guests
5. Say something Text field
6. Send photo Button
7. Send emoticon Button
8. Live stream Button
9. Edit Event Details Button
10. Menu Button

2.18.2 Interactions

After clicking on 'Edit Event Details' button, the application goes to the Edit Event Details screen (<https://invis.io/VT76IHQ52>). 'Count down to Event' section shows days, hours and minutes left to the event. New messages can be posted by clicking on the 'Say something' Text field and clicking "Send" on the keyboard. Additionally, after clicking on the 'Photo' icon a Host can Take a photo / video or choose it from Camera Roll; after clicking on the 'Emoticon' icon a Host can choose among standard emojis. Sending emojis are animated Periscope-style (<http://browniefed.com/blog/react-native-periscope-hearts-animation/>). By clicking on 'Live' button Guest can start a live stream. The "Communicator View" then changes to full-screen live video stream. You can switch between live stream and Communicator view by tapping anywhere on the chat. After the Live stream is over the full-screen live video stream changes to the Communicator View. Last uploaded photo becomes a background.

2.18.3 API

1. Return event guests

2.19 Communicator View (Host)

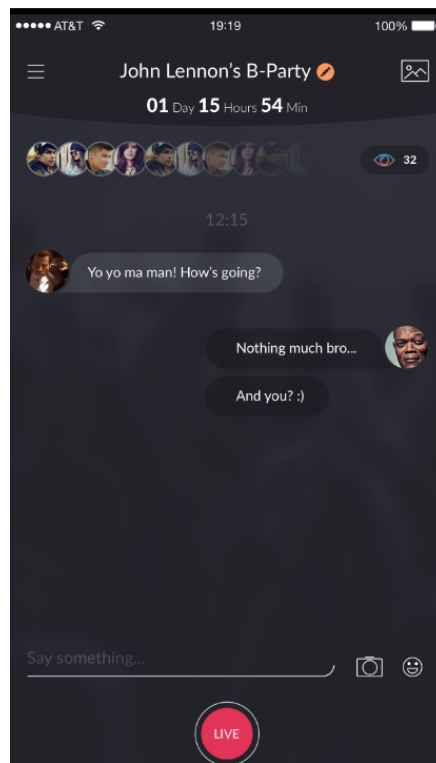


Figure 2.13: Communicator View (Host)

2.19.1 Design

1. 'Name of event' title
2. Countdown to Event
3. List of signed up guests
4. Number of signed up guests
5. Space for messages
6. Media button
7. Say something... Text field
8. Send photo Button
9. Send emoticon Button
10. Live stream Button
11. Edit Event Details Button
12. Menu Button

2.19.2 Interactions

When the first Guest signs up Hosts see the chat screen. After clicking on 'Edit Event Details' button, the application goes to the Edit Event Details screen (<https://invis.io/VT76IHQ52>). By clicking on a 'Number of signed up guests' icon Host opens a Who's coming screen. By clicking on the avatars of people Host opens a Who's coming screen. (You can click around here: <https://invis.io/7V75BE86Z>.) This Communicator View is a group chat that any Guest or Host can see. New messages can be posted by clicking on the 'Say something' Text field and clicking "Send" on the keyboard. Additionally, after clicking on the 'Photo' icon a Host can Take a photo / video or choose it from Camera Roll; after clicking on the 'Emoticon' icon a Host can choose among standard emojis. Sending emojis are animated Periscope-style (<http://browniefed.com/blog/react-native-periscope-hearts-animation/>). By clicking on 'Live' button Host can start a live stream that all other Guests and Hosts can see. The "Communicator View" then changes to full-screen live video stream. You can switch between live stream and Communicator view by tapping anywhere on the chat. Hosts can also click on the 'Media' button to see a list of media shared, if any. The last image shared in the chat automatically becomes the background of the Communicator View. After the Live stream is over the full-screen live video stream changes to the Communicator View.

2.19.3 API

1. Implemented with chat SDK

2.20 Live Stream View (Host, Guest)

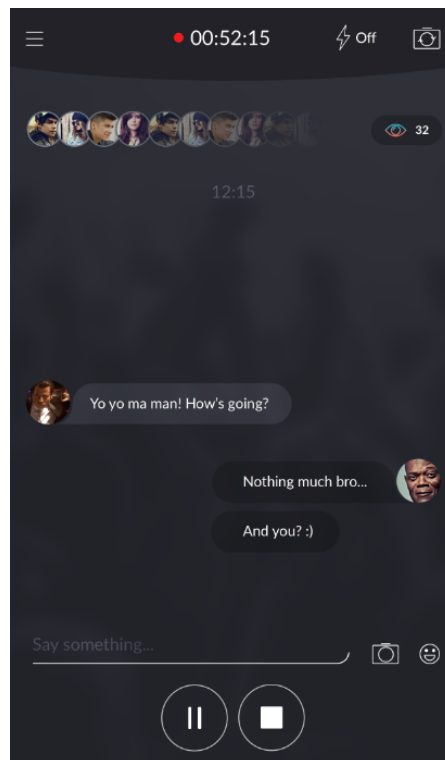


Figure 2.14: Live Stream View (Host, Guest)

2.20.1 Design

1. Flash button
2. Stream duration indicator
3. Switch camera button
4. Avatars of signed up guests
5. Number of signed up guests
6. Space for messages
7. Media button
8. Say something... Text field
9. Send photo Button
10. Send emoticon Button
11. Live stream Button
12. Pause button
13. Stop button
14. Menu Button

2.20.2 Interactions

By clicking on 'Live' button Host or Guest can start a live stream that all other Guests and Hosts can see. The "Communicator View" then changes to full-screen live video stream. Users can switch between live stream and Communicator view by tapping anywhere on the chat. The User who is streaming can pause a stream, as well as end the stream. Everyone else will just see a Communicator View when the stream is off, but they won't see the Live button to be able to start a new stream. By clicking on a 'Number of signed up guests' icon Host opens a Who's coming screen. By clicking on the avatars of people Host opens a Who's coming screen. (You can click around here: <https://invis.io/7V75BE86Z>.) This screen is a group chat that any Guest or Host can see. New messages can be posted by clicking on the 'Say something' Text field and clicking "Send" on the keyboard. Additionally, after clicking on the 'Photo' icon a Host can Take a photo / video or choose it from Camera Roll; after clicking on the 'Emoticon' icon a Host can choose among standard emojis. Sending emojis are animated Periscope-style (<http://browniefed.com/blog/react-native-periscope-hearts-animation/>). Hosts can also click on the 'Media' button to see a list of media shared, if any. After the Live stream is over the full-screen live video stream changes to the Communicator View.

2.20.3 API

1. Implemented with chat SDK

2.21 Options View (Guest, Host)

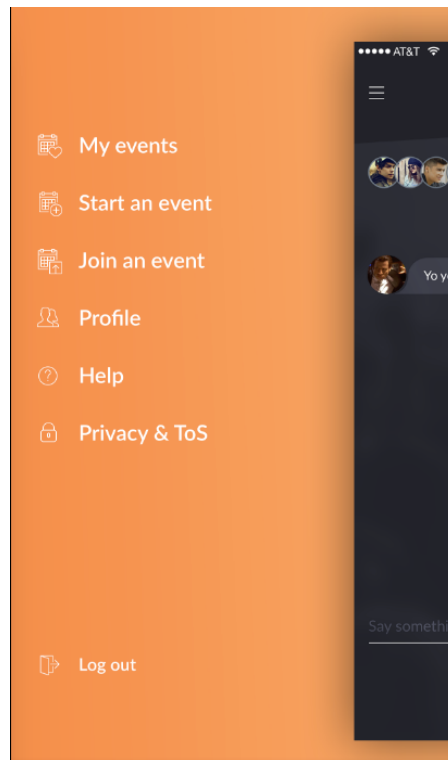


Figure 2.15: Options View

2.21.1 Design

1. My events Overlap
2. Start an event Overlap
3. Join an event Overlap
4. Profile Overlap
5. Help Overlap
6. Privacy and Terms Overlap
7. Log out Overlap

2.21.2 Interactions

After clicking on My events overlap, the application goes to My events View. After clicking on 'Start an Event' overlap, the application goes to the screen where Host can enter Host code(s). After clicking on 'Join an Event' overlap, the application goes to the screen where Guest can enter Guest code. After clicking on 'Profile' overlap, the application goes to user's Profile View. Clicking on 'Help' overlap, opens help screen with faq / contact form. Privacy and Terms opens a respective screen (<https://invis.io/K876IGQH6>). After clicking on 'Log out' button the user is being logged out. Note: UI shows 'Privacy and ToS', but it should say 'Privacy and Terms'.

2.21.3 API

1. Logout

2.22 Privacy and Terms View (Guest, Host)

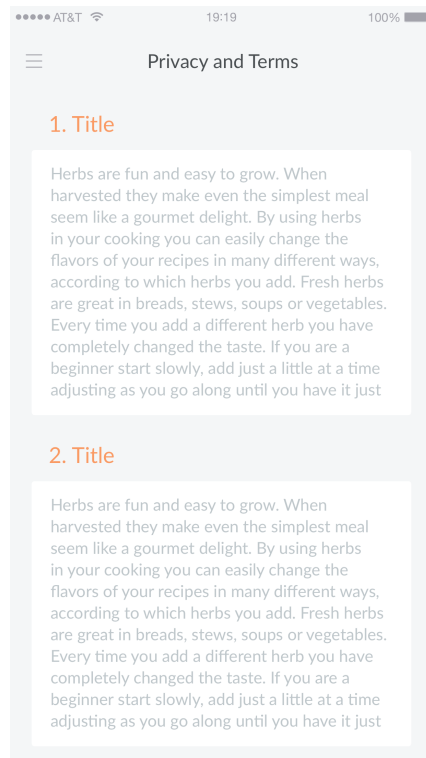


Figure 2.16: Privacy and Terms View

2.22.1 Design

1. (Optional) Menu Button
2. (Optional) Back Button
3. Privacy and Terms title
4. Subtitles
5. Text

2.22.2 Interactions

User can scroll this screen to see Privacy and Terms text. Depending on where he came from he can press Menu Button to go back to navigating between screens, or press Back Button to go back to Sign Up screen.

2.23 Help View (Guest, Host)

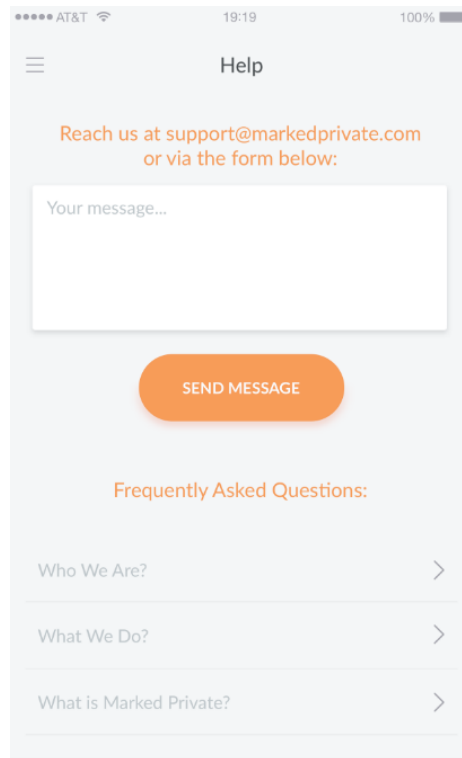


Figure 2.17: Help View

2.23.1 Design

1. Menu button
2. Help title
3. Description text
4. Message... text field
5. Send message button
6. Frequently Asked Questions: title
7. Questions accordion title
8. Questions text

2.23.2 Interactions

User can enter text and send it to app's support email. User can press a collapse answer to view it. By default the first question is open, all the rest are collapsed. By pressing Menu Button User goes back to navigating between screens.

2.24 Edit Profile View (Guest, Host)

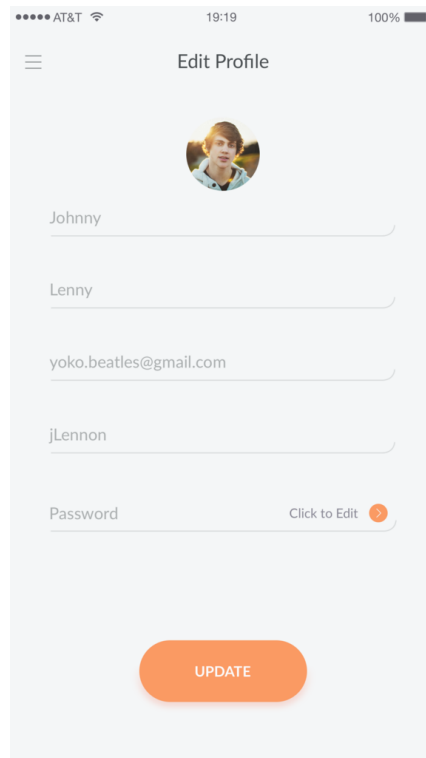


Figure 2.18: Edit Profile View

2.24.1 Design

1. 'Edit profile' title
2. Avatar (Take photo / Choose from Camera Roll) Button
3. Username Text field
4. First name Text field
5. Last initial Text field
6. Parent's email Text field
7. Password Text field
8. Repeat password Text field
9. Update Button
10. Menu Button

2.24.2 Interactions

After updating the required data and clicking on 'Update' button, the application updates User's profile details. After clicking on Take photo / Choose from Camera Roll button updating user's profile picture is available. If user doesn't have an avatar we show a generic placeholder avatar. By clicking on the menu button User can navigate to other screens.

2.24.3 API

1. Update Profile - rewriting data for User's first name, last initial, parent's email and password.

2.25 Changing Password View

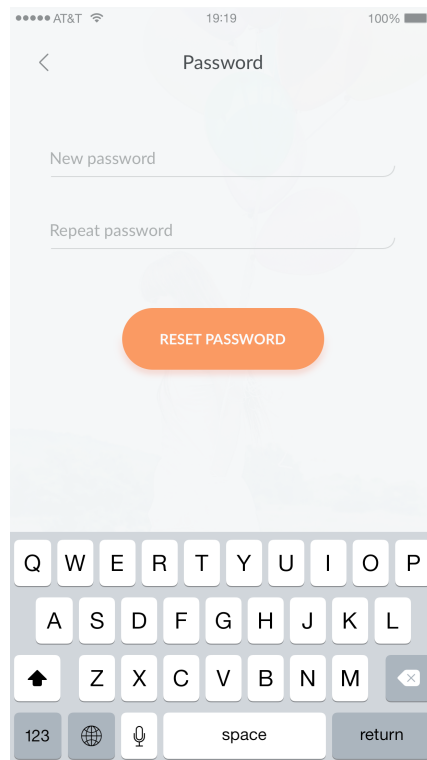


Figure 2.19: Changing Password View

2.25.1 Design

1. 'Password' title
2. Current password Text Field
3. New password Text Field
4. Repeat password Text Field
5. Update Button

2.25.2 Interactions

After entering user's password into the 'Current Password' Text Field (not reflected in the UI, but will be part of the app), providing the new one twice into 'New password' and 'Repeat password' Text Fields and clicking on 'Update' button the password is being changed.

2.26 Confirming New Password View

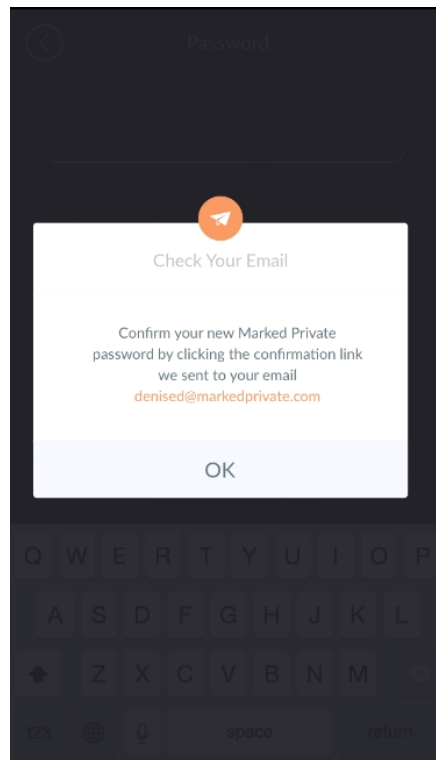


Figure 2.20: Confirming New Password View

2.26.1 Design

1. 'Check Your Email' title
2. Info about about change password
3. OK Button

2.26.2 Interactions

The application shows the information about the neccessity of confirming the new password by clicking on a link sent to the user's email address. Clicking on 'OK' button closes this section.

2.26.3 API

1. Change password for given user

2.27 Edit Event Details View

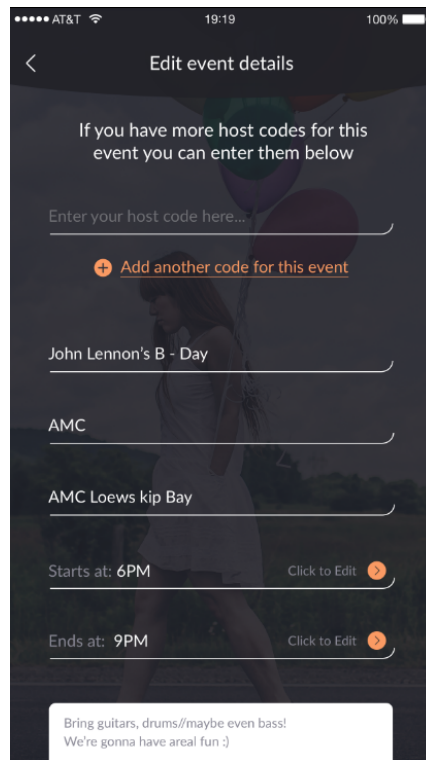


Figure 2.21: Edit Event Details View

2.27.1 Design

1. 'Back' button
2. 'Edit event details' title
3. 'Enter your host code' Text Field
4. Add another code Hyperlink
5. Name of event Text Field
6. Location of event button
7. Time from Button
8. Time to Button
9. Details about event
10. 'Save' button

2.27.2 Interactions

IMPORTANT: design should say 'Edit Event Details', instead of 'Edit event details'. Location field should have 'Click to edit' button, just like on Starting an event view.

Clicking on 'Add another code' hyperlink gives an option to add more than one code. When Host clicks to add another Host code a new text field appear below the previous text field. The information about Event's location, date etc. can be edited in the proper Text Fields and Buttons in this View. By clicking on Back Button Host goes back to Communicator View. All the fields except for event name are optional. When Host clicks on Start time and End time buttons he sees a date and time picker (<http://i.stack.imgur.com/n9gLq.png>), where he can pick date and time and press done to hide it. Start time / end time indicates the time that was set during creating the event, or is not filled if Host didn't set time when creating the event. Event's time can be chosen by picking the start and end time, they will be displayed with pm/am. By clicking on 'Save' button app verifies the code(s), if necessary, and saves updated event details.

2.27.3 API

1. Update event details

2.28 Categories of Invited Guests View

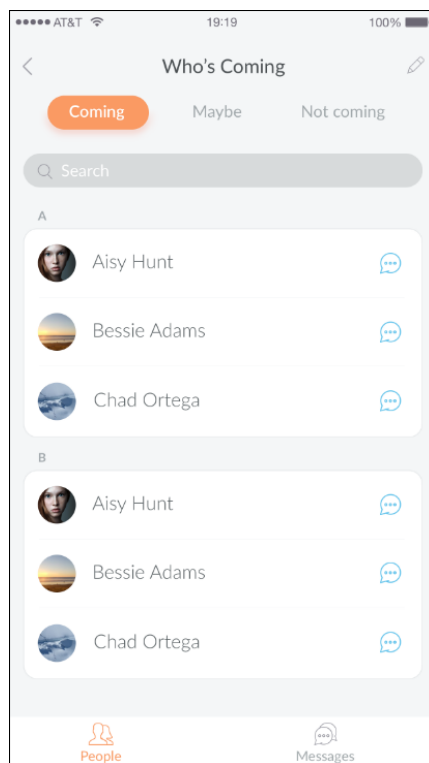


Figure 2.22: Categories of Invited Guests View

2.28.1 Design

1. 'Who's Coming' title
2. Coming/Maybe/Not coming Buttons
3. Search bar
4. Alphabetically placed list of invited guests
5. People Button
6. Messages Button
7. Back Button

2.28.2 Interactions

User can switch between messages and people (who's coming). Clicking on Coming/Maybe/Not coming button groups all invited guests into three groups of guest who declared that will come, will possibly come or will not come. By using 'Search bar' any guest from the list can be searched by first name. Clicking the pencil icon starts a new conversation with a user. By clicking on the back button user goes to chat screen.

2.28.3 API

1. Return list of invited guests with categories

2.29 Messages View

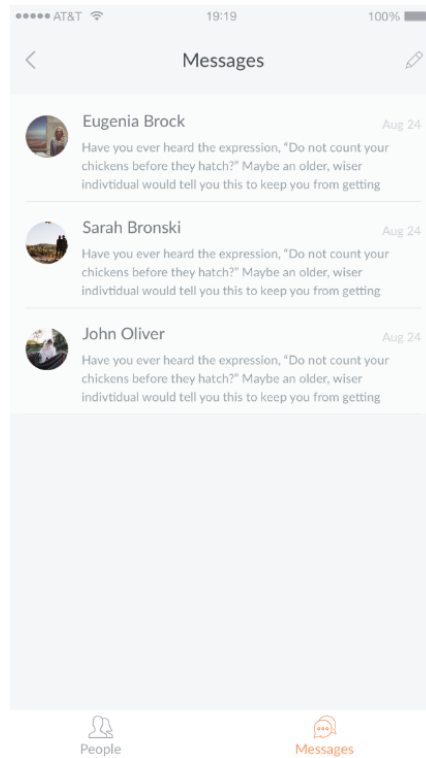


Figure 2.23: Messages View

2.29.1 Design

1. 'Messages' title
2. Chronologically placed messages
3. People Button
4. Messages Button
5. Back Button

2.29.2 Interactions

This view shows the messages displayed chronologically (new to old). User can switch between messages and people (who's coming). Clicking the message opens a 1-1 Message View. Clicking the pencil icon starts a new conversation with a user (or group of users). When a new message comes up User gets a push notification 'You got a new message from [First Name].' By clicking on the back button user goes to chat screen.

2.29.3 API

1. Return messages

2.30 New Message View (Guest, Host)

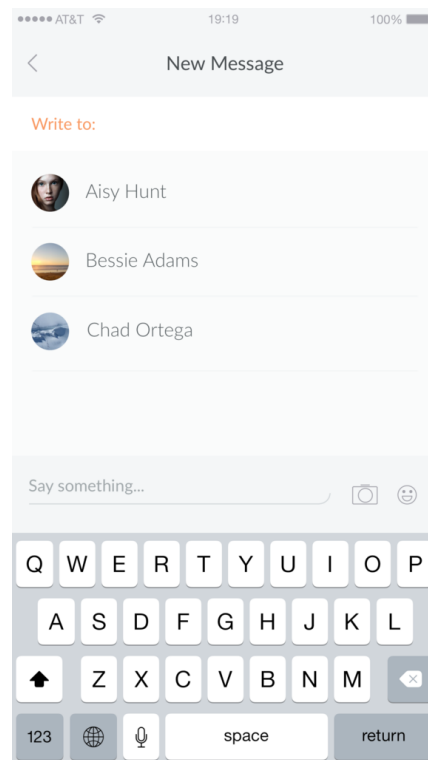


Figure 2.24: New Message View

2.30.1 Design

1. Back Button
2. New Message Title
3. Write to: search field
4. User avatar
5. First name
6. Last initial
7. Send emoticon button
8. Send photo / video button
9. Your message... text box

2.30.2 Interactions

User gets on this screen after clicking the 'Pencil' icon on the Messages or Who's coming screen. By clicking Back Button User goes to the previous respective screen. When User open this screen the keyboard is open by default to let User search for other Users. User can select

multiple Users to start a group chat, or select 1 User to start 1-1 chat. User can select multiple people to chat to. Selecting people works like on Facebook Messenger. New messages can be posted by clicking on the 'Say something' Text field and clicking "Send" on the keyboard. Additionally, after clicking on the 'Photo' icon a Guest can Take a photo / video or choose it from Camera Roll; after clicking on the 'Emoticon' icon a Guest can choose among standard emojis. Send- ing emojis are animated Periscope-style (<http://browniefed.com/blog/react-native-periscope-hearts-animation/>). When looking for people the emoji / camera buttons are inactive, they only become active when the chat starts.

2.30.3 API

1. Search for Guests and Hosts.

2.31 1-1 Message View (Guest, Host)

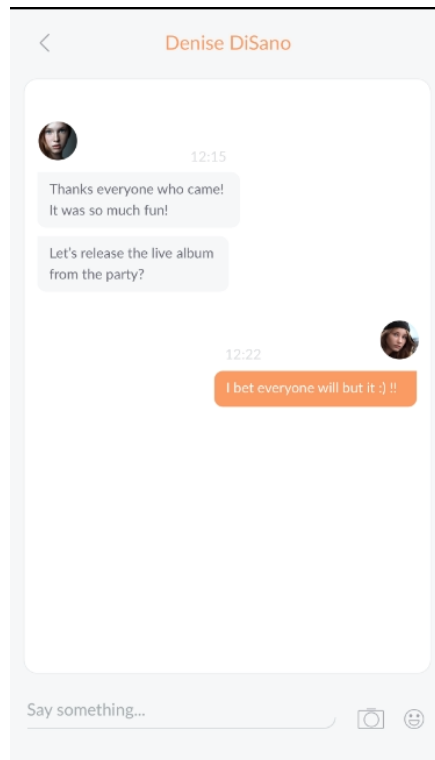


Figure 2.25: 1-1 Message View

2.31.1 Design

1. Name of guest
2. Messages
3. 'Say something' input field
4. Camera button
5. Emoji button
6. Back button

2.31.2 Interactions

This is a View for private messages. Any message can be posted in this View's 'Say something' input field. User can also post photos / videos by pressing Camera button and choosing them from the Camera Roll or snapping them. User can also post emoji by pressing Emoji button. Emojis are posted normally, as messages, unlike Chat screens where they are posted like animations.

2.31.3 API

1. Implemented with chat SDK

2.32 Media View



Figure 2.26: Media View

2.32.1 Design

1. 'Media' title
2. Clickable miniatures of photos
3. Back button

2.32.2 Interactions

Clicking on the miniatures enlarges the pictures (<https://invis.io/3V76ITD78>). Clicking on the back button goes back to the Chat screen.

2.32.3 API

1. Return photos

Chapter 3

Non-functional Requirements

3.1 Mobile Application

1. Application works on iOS devices:
 - iOS 8,
 - iOS 9.
2. The application works in portrait and landscape modes (vertical and horizontal).
3. Views for iPads are displayed in 2x mode.
4. The application works only in the on-line mode i.e. when the device has an access to the Internet and the operating system allows the application to use the network services.
5. The application does not offer any functionalities in off-line mode.
6. Notifications implemented by Apple Push Notification Service.
7. The application will be developed in Swift.
8. API implemented with following features:
 - (a) Amazon Web Services
 - (b) MySQL 5.7.10
 - (c) PHP 7.0.5 with Laravel framework
9. API returns results in JSON format.
10. Server meets all the requirements of API and database.
11. Chat functionality implemented with QuickBlox
<http://quickblox.com/developers/IOS>
12. Livestream functionality implemented with Live Stream SDK.
<http://livestreamsdk.com/#documentation>
13. The list of devices for the application:

iPhone 4S

iPhone 5

iPhone 5S

iPhone 5C

iPhone 6

iPhone 6 Plus

iPhone 6S

iPhone 6S Plus

iPad 2

iPad (3rd Gen)

iPad (4th Gen)

iPad Air

iPad Mini (2nd gen)