Bristech Speaker Relationship Management System

Elias Kassell Raymond, Roman Bromidge, Soo Yee Lim,
Aneesh Anand, Abby Weng, Martin Dimitrov

15th December 2017

Contents

1 Overview

1.1 Client

Bristech is a monthly technology themed meeting at the Engine Shed taking place on the first Thursday of each month. They attract a wide range of people: those who work in tech, want to learn about tech or those who are simply interested in the field.

They host 2 speakers at each event and provide food and drinks. They have a website with information about Bristech and their talks and currently use meetup.com in order to manage sign-ups for each event. Users must also manually sign in when they arrive at the event, though this is not always a sure-fire to way to record attendance. Speakers at events are all tech aficionados, examples of which include professional developers, company founders, academic researchers and general hobbyists.

1.2 Domain

Events are free to attend, and as such Bristech sustains itself through Sponsorship. There are not many upfront costs of hosting the events, other than the hiring of the venue and supplying free drinks and beverages to attendees. It provides an exciting opportunity and a pleasant environment, for both gregarious and introverted people who are interested in similar tech-related subjects, to converse and learn from each other.

1.3 Project

The project is to develop a more elegant way for speakers, managers and attendees to interact in the run-up to and during events. The key motivation for the new platform is to go beyond what meetup currently offers Bristech and its users. A large part of this is that meetup at this moment has ownership of all the user accounts so having its own platform gives the control over to Bristech. This then enables them to perform their own data analytics on the users so that they can target specific events to groups of people based on the events they've previously attended. Essential features of our app will include keeping track of members who attend, providing feedback functionality for users to speakers, and a distinguished Bristech membership with clear benefits of being a member. These problems stem from the fact our client has noticed that his events have a high drop out and no-show rate and would like to decrease the percentage of people who sign up for an event but then don't turn up. As all the events are usually very busy with a sizeable waiting list, having people not turn up to events means that other people who could have come may not be able to attend as the booking slot is taken even though it is not used. Feedback functionality is a symptom of increasing, expanding and retaining member participation.

1.4 Our solution

Our solution is a holistic mobile application. Members of the event will be able to view past and upcoming events, register for events, provide feedback on events, view feedback on talks, and be prompted to provide feedback when attending a talk, as well as monitoring their attendance through a geolocating check-in system to ensure they are in the vicinity of the venue at the correct time. Members will also be able to apply to be a speaker or recommend other people for Bristech to approach to invite to be a speaker. Managers on the app will have the ability to create events, view user data analytics, and remind non-attendees of the frustration when they register for a popular talk and then fail to attend.

2 Requirements

2.1 Stakeholders

2.1.1 End Users

Anyone who is going to use the app, who we have categorised into attendees, speakers and managers.

2.1.2 Attendees

These are people who attend the Bristech talks and have the ability to request to be a speaker. They may come every month, have only attended a single event or somewhere in between. They currently sign up for the events via meetup.com. They will be able to view events (upcoming and previous) and their associated feedback on the app and will be prompted to provide feedback when attending a talk.

2.1.3 Speakers

A subcategory of attendees, who have the exact same app functionality as attendees. They are the most likely to look at feedback on events, and ideally, we would like them to leave the event very satisfied with their experience.

2.1.4 System Managers

These are the managers and administrators of the system. They would likely be the Bristech founders, such as Nic Hemley, or others that are brought in in the future. They will have privileged access on the app after logging in, which will include the ability to create events, view data analysis on users and remind non-attendees of the frustration when users register for a popular talk and then fail to attend.

2.1.5 System Owners

47 Degrees: the client would like us to make our system white label and with the potential to be applicable to other meetup style events.

2.1.6 Engine Shed

They are the hosts of the events currently. They want calm and manageable events with minimal interaction so that they do not have to hire extra staff.

2.1.7 Engine Shed Staff

Staff who are working at the Engine Shed while the event is taking place. Mainly security guards, who want the event to go off without a great deal of intervention required so that they can have a calm night.

2.1.8 App Store

Host the app, providing the ability for end users to be able to download it to phones and tablets. They require the app to fulfil their standards for distribution.

2.1.9 Sponsors of Bristech

These provide the event with money that is needed for it to run, in return for advertising and promotion.

2.1.10 Server Host

Needs to interact with the app correctly, and in return requires payment for cloud usage.

2.2 App user use-case diagram

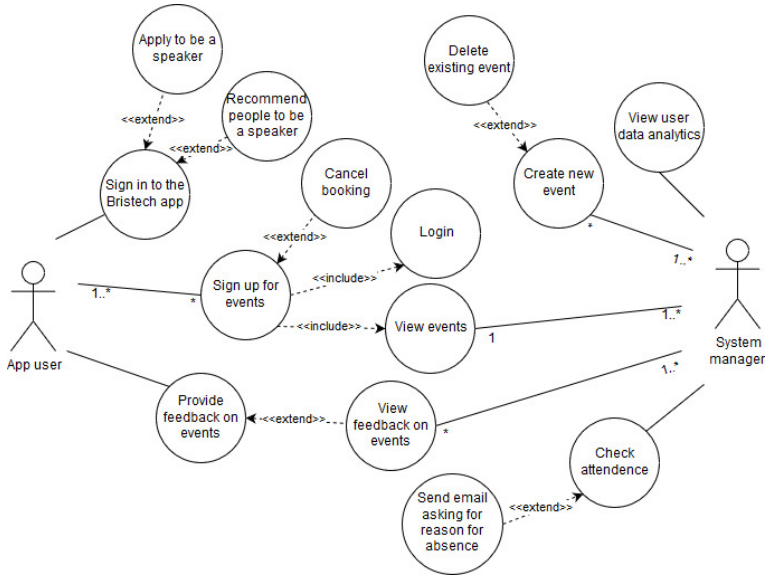


Figure 1: Use Case Diagram

2.3 Core use-case goals

2.3.1 Basic flow for Signing up for events

1. User launches the application and goes to the homepage
2. User views upcoming events listed on the homepage
3. User decides which event to attend
4. User opens the Event Detail screen by pressing on the selected event's card
5. User registers for the event by pressing "Register for event" button
6. User logs into their account in order to register for the event

2.3.2 Basic flow for Creating an account

1. User launches the application and goes to the homepage
2. User toggles the sidebar by swiping right on the home-screen
3. User opens the User Settings screen by pressing "User Settings" button on the sidebar
4. User opens the Create account screen by pressing "Create account" button
5. User inputs their user data and presses the "Create account" button
6. Account is created if the information is valid and unique
7. User returns to the homepage

2.3.3 Basic flow for Leaving Feedback

1. User physically attends an event
2. User receives a notification on their phone asking them to leave feedback
3. If user decides to leave feedback, they press the notification
4. User enters the Provide feedback screen
5. User enters their feedback
6. User presses the "Submit" button

2.3.4 Alternative flow for Leaving feedback

1. User launches the application and goes to the homepage
2. User toggles the sidebar by swiping right on the home-screen
3. User opens the View past events screen by pressing "View past events" button on the sidebar
4. User is triggered to log in
5. User logs in
6. User views events that they have attended
7. User opens Event Detail screen by pressing on the selected event's card
8. User enters the Provide feedback screen by pressing "Provide feedback" button
9. User enters their feedback
10. User presses the "Submit" button
11. User returns to the View past events screen

2.3.5 Basic flow for Creating events

1. User launches the application and goes to the homepage
2. User toggles the sidebar by swiping right on the home-screen
3. User opens the User Settings screen by pressing “User Settings” button on the sidebar
4. User opens the Login screen by pressing “Login” button
5. User logs in with a manager privileged account
6. Manager enters the Management screen
7. Manager enters the Create event screen by pressing the “Create event” button
8. Manager inputs event information
9. Manager presses the “Create event” button
10. Manager returns to the Management screen

2.4 Atomic requirements for registering for an event

2.4.1 Functional requirements

1. User shall be able to view the home-screen on application launch, given there’s a stable connection to the Internet
2. User shall be able to browse through all available upcoming events listed on the home-screen. User shall be able to view event title, location, time, speakers and short description
3. User shall be taken to the Event Detail screen on event card press
4. User shall be able to scroll through the event’s details
5. User should be able to view event duration, long description and number of registered attendees
6. User should be able to see event images
7. User shall be able to register for an event by pressing a button
8. User should be able to unregister for an event by pressing a button
9. User should be able to favorite an event by pressing a button
10. User shall be able to go back to the homescreen by pressing a button
11. User shall be returned to the homescreen after registering to the event

2.4.2 Non-functional requirements

1. Application shall fetch the upcoming event list within 2 seconds after starting the application, provided there’s a stable Internet connection
2. Application shall cache the upcoming event list for later use to avoid fetching the list again
3. Application shall start a background running service to connect and write to the database on event registration
4. Application should cache any images for later use to avoid re-downloading them again

3 Architecture

3.1 Key architecture features

Our client's overall requirement is for us to build a system to manage the interaction between the speakers and members of their talks. As an answer to this, we could take either the web application or mobile application route. Our chosen platform is an Android mobile application and the key architectural drivers that will shape our architecture design are:

- **Design Purpose**
  We've tried to keep the design of our system as simplistic as possible to enable us all to maintain a solid understanding of how our system and its various sections function and interact with each other and to provide an enjoyable experience for the end-users.

- **Quality Attributes**
  This includes any measurable or testable properties of a system used to indicate how well the system satisfies the needs of stakeholders. With Bristech holding monthly events we have the luxury of being able to test out whatever functionality we have with our system at these monthly talks and receive feedback on how well the users think our system works. Our client has expressed to us what they believe to be the most important attributes and these have been outlined clearly in the requirements section of our portfolio. Therefore from this, we can see what aspects of functionality we need to be working toward first.

- **Primary Functionality**
  These aspects of functionality are those which are critical to achieving the end business goal. For our system, this will be the creation of events for the managers and also the logging in and sign up for events for the users. However, we will ensure that our architecture is always open to change and improvement.

- **Architectural Concerns**
  We may encounter various architectural concerns to do with supporting updates to our system and authentication and authorization for our members and managers when they are logging into the platform.

- **Constraints**
  Technical: Our system will have to integrate with Oracle Cloud services who will be hosting and storing our various databases.
  Non-technical: We will have to ensure that we aren't breaching any standardized data protection laws regarding the personal information of our users. Time and the technical abilities of our developers are also constraints for us to have to bear in mind.

3.2 Architecture design diagram

Our architecture diagram (figure 2) shows concisely how our system will be structured and also how the various different parts of our system interact with each other, mainly our Android application and the server side of the system where we're using Oracle Cloud services for this. The holding of our databases for members and all their information will be managed by implementing MySQL.

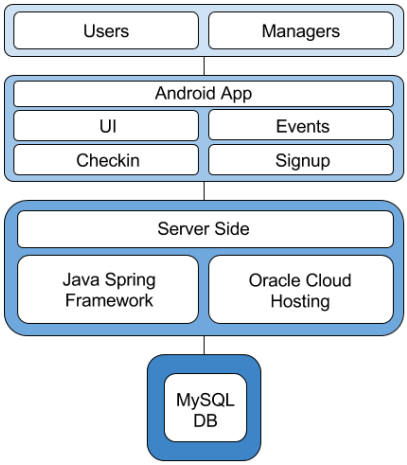


Figure 2: High Level Architecture Diagram

4 Development Testing

4.1 Strategy

Our approach to testing during the implementation phase takes the form of automated unit testing using the Espresso framework, which is a superset of JUnit made specifically for Android development. Using this framework we can test state expectations, interactions and assertions clearly and in an optimal manner. We will use a black box approach to test our system against our overall specification (eg: creating an event, registering to attend an event, etc) and utilise white box path testing on critical components such as logging in and checking in at an event to ensure these parts function precisely as we intend.

4.2 Core Goal Tests

Below we will outline the tests needed to ensure a core component of our system works. We will use 'Checkin at Event' as it is a system critical component for both operating functionality and meeting the outline of the specification. As such, we can break it down into many steps that we must perform tests on.

Functionality	Test	Desired Outcome
Geofence setup	Does the geofence correctly deploy in target location	Geofence deploys according to our coordinates
User enter geofence	Does the system detect when a user enters the geofence	System detects user entering and changes state
User leave geofence	Does the system detect when a user leaves the geofence	System detects user leaving and changes state
Timer countdown begin	Does the timer begin once the user enters the geofence	Timer begins once system detects user state
Auto checkin after 5 mins	Does the system register a user as attended once they have remained in the area for 5 mins	System changes state and alerts server once time limit has passed and user is registered as attended
Timer countdown ends upon time up	Does the system stop the timer once it reaches 5 mins	Timer stops once timer reaches 300 seconds
Timer countdown ends when user leaves geofence	Does the timer stop when the user leaves the fence	Timer stops if user leaves the geofence while running
Geofence available at time	Does the geofence activate at the time for the event	Geofence active on time/date of choice and not otherwise
Timer restarts once user reenters the fence	Does the timer restart once the user enters the fence again	Timer restarts and system changes state once user reenters fence

Figure 3: Development Testing Core Function Tests

5 Release Testing

5.1 Strategy

Our strategy for testing during the integration and deployment phases take establishing a weekly regular release with a framework called Extensive testing, which it is an open and generic automation framework designed to make integration, validation, regression, deployment and end-to-end testing with a collaborative workspace environment. During the cycle, we suggest a sequence of tasks to release the software, then we would do a dry run, using dummy code for each element. One of the core use-cases of our system is that users can sign up for an even and they could cancel it when they regret. This is the main reason that to develop a solution to keeping track of member attendance at the events, and by tracking the attendance in an efficient way, we could decrease the percentage of people who say they are coming to an event but never show up.

5.2 Core Goal Tests

In order to ensure the software runs as intended, we will do these release testing every week as a regular cycle with a limited number of beta users for validation and find issues with current use cases. After that, this will then be followed by an alpha release to a larger user group. Some core function tests are as follows.

Functionality	Test	Desired Outcome
Installation	Make use of developers' local machines and will be performed on all supported architectures (android)	Recreate the users' installation experiences to ensure that users will not have problems during installation
Interoperability	Interoperability testing tested how release sends and receives messages	When all tests are executed, the test results are captured and the final results are generated manually
Regression	Automated test if previously existing functionality works in current release	Ensure that previously existing functionality still works in the current release.

Figure 4: Release Testing Core Function Tests

6 OO design and UML

6.1 Static app activity design

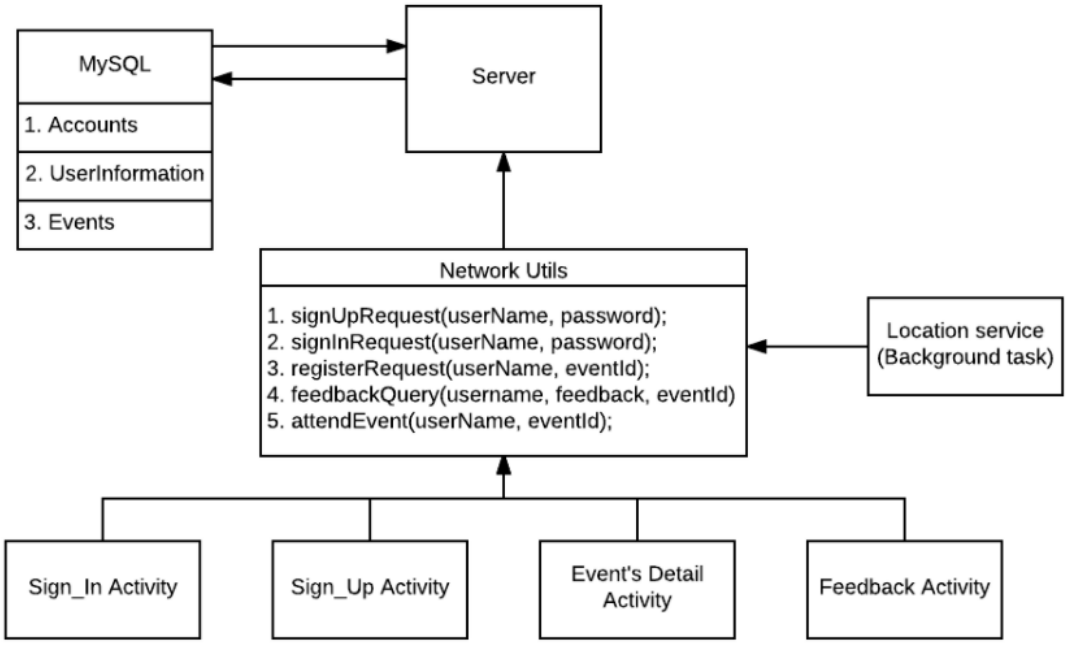


Figure 5: Static app activity design

The above diagram shows how the different parts of the system fit together and interact with each other. It demonstrates how the different parts of the Android app fit together and interact, and then how these as a whole interact with the other parts of the system as defined in the architecture section.

6.2 Dynamic app activity design for generic most frequent case use flow interaction

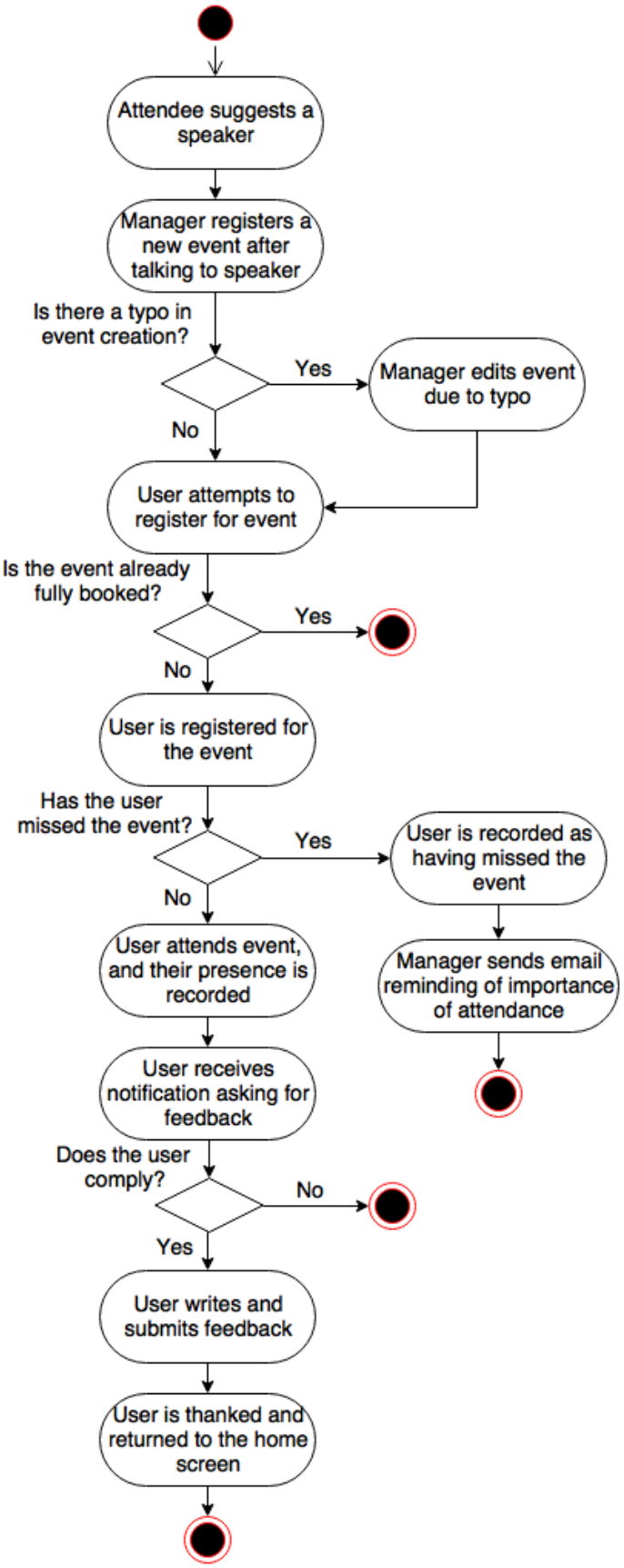


Figure 6: Dynamic app usage integration design for a user attending an event

The activity diagram above models the overall control flow of the system. The activity diagram is chosen to visualise the dynamic nature of our system because it models activities which are business requirements. Furthermore, the activity diagram also gives a high-level view of our system which is easy for a non-technical person to understand the flow of control in our system. In our applications, the activities modelled are sequential.

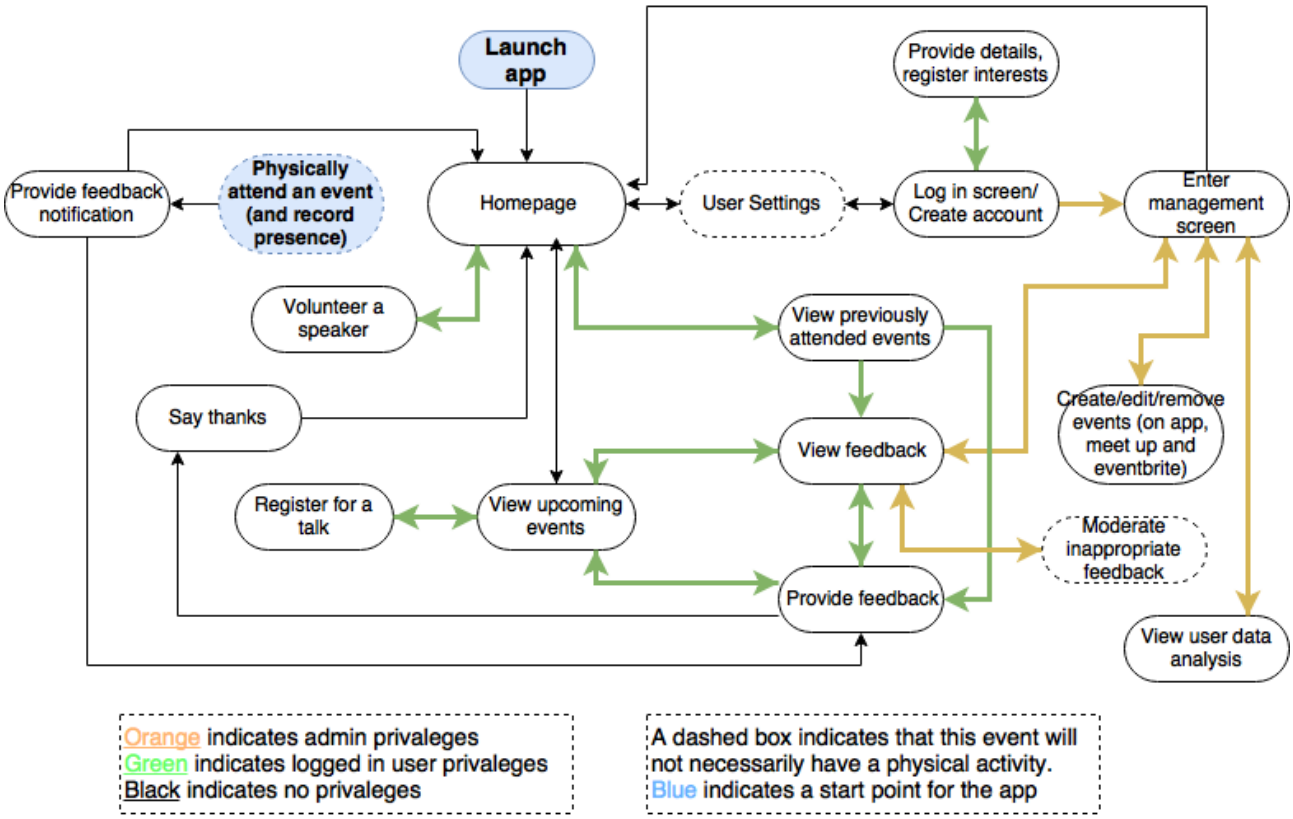
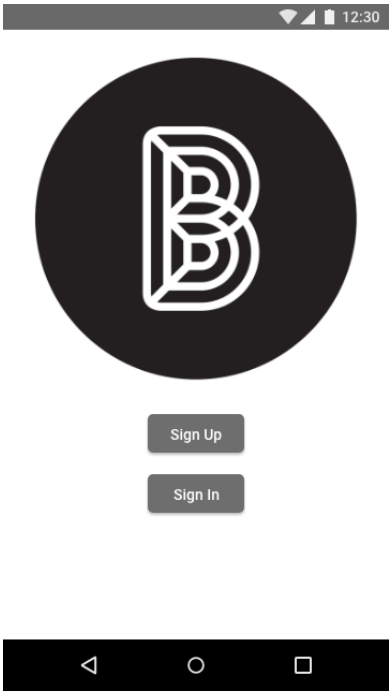


Figure 7: Basic app activity design

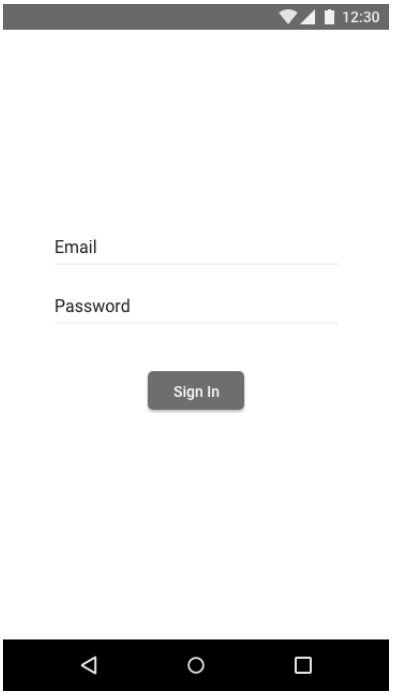
This diagram illustrates key aspects of the app, that will provide all end-users with the desired functionality. Physically attending a talk is a particularly unique event in this design, as it directly triggers a notification based off of the user's location at a specific time.

7 Interface Design

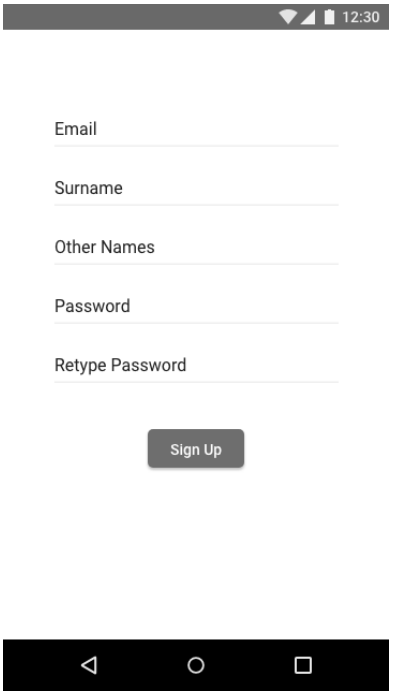
7.1 Images



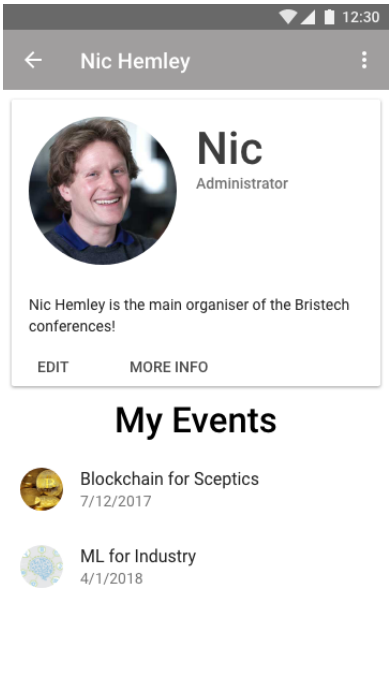
(a) Welcome Page



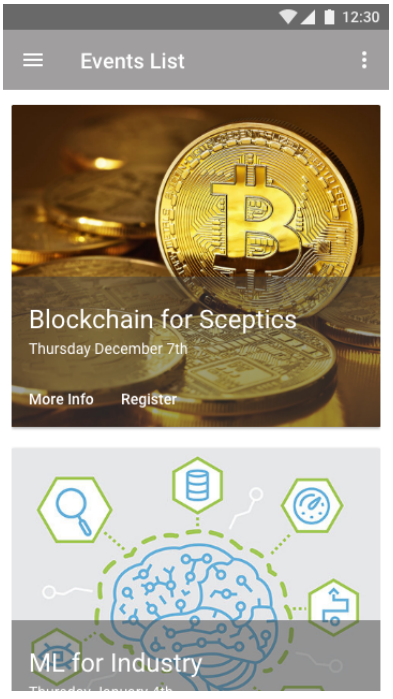
(b) Sign In



(c) Sign Up



(d) User Profile



(e) Events List

(f) Event's Details