# Bristech Speaker Relationship Management System

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

20th April 2018

## 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 which comes primarily from local technology companies. There are not many upfront costs of hosting the events, other than the hiring of the venue and supplying free drinks and refreshments 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. Every year, Bristech host a conference day for which they sell tickets and which brings up to 6 speakers in one day, this being their premium event.

## 1.3 Project

The aim of 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.

Part of the current problem is that Meetup has ownership of all the user account data. An independent platform gives the control over the data 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 and providing feedback functionality for users.

These problems stem from the fact that the events have a high drop out and no-show rate and an ideal solution would therefore 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 means by which system managers can increase, expand and retain high levels of member participation.

## 1.4 Our solution

Our solution is a dynamic mobile application partially integrated with meetup.com, to begin the transition to a holistic application while not losing the benefits of meetup.com. Members of the event will be able to view past and upcoming events, register for events, provide feedback on events, and be prompted to provide feedback when attending a talk, as well as monitoring their attendance through a geofencing 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. Bristech managers will be able to create events and view user analytics. The application integrates with the Meetup.com API in order to maintain consistency between the app and the website, thus giving the app a high degree of interoperability with the current system in use.

Users will be able to create an account with an email or password, or login using their Facebook or Google accounts. This will be available due to OAuth integration.

# 2 Requirements

## 2.1 Stakeholders

### 2.1.1 Attendees

This is the largest stakeholder group and also the main user group. These are people who attend the Bristech talks and have the ability to request to be a speaker. They may come every month, occasionally or have only attended a single event. 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.

#### 2.1.2 Speakers

A far smaller subcategory of users, who have the exact same app functionality as regular attendees. They are able to volunteer other speakers who they may work with or know personally. These stakeholders are important as they are more directly connected to the system managers, and as such they will be more discerning of whether nor not they use the app.

#### 2.1.3 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. This is a user group of small size and so therefore need an efficient way to process lots of data about their events.

#### 2.1.4 System Owners

A design choice is to make the system white label and with the potential to be applicable to other meetups. Thus the system owner, who is initially 47 Degrees, can duplicate and share the functionality in order to expand the systems mandate and make it more widely usable.

#### 2.1.5 Engine Shed

Hosts of the events currently who want a calm and manageable event with minimal interaction so that they do not have to hire extra staff. They would not directly use the app, but may be affected by how it works: less crowding at the door to tick off who is at the event. Also, they may wish to suggest usage of the application to other monthly meetups that they host.

#### 2.1.6 Engine Shed Staff

Staff who are working at the Engine Shed while the event is taking place. Mainly security guard, who wants the event to go off without a great deal of intervention required so that they can have a calm night. This stakeholder could use the app to monitor who should be allowed in and who shouldn't.

#### 2.1.7 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. They will only care about the application in as far as it passes their quality assurance tests.

#### 2.1.8 Sponsors of Bristech

These provide the event with money that is needed for it to run, in return for advertising and promotion. They can get more value for their sponsorship money by being able to be shown on the app. They will also be very influential, and therefore it is essential that the app be a quality product such that it shows them in a good light to be associated with it.

#### 2.1.9 Server Host

Needs to interact with the app correctly, and in return requires payment for cloud usage. This is a small group of stakeholders who do not interface directly with the app functionality. Instead they are simply a key part of the app infrastructure.

## 2.2 App user use-case diagram

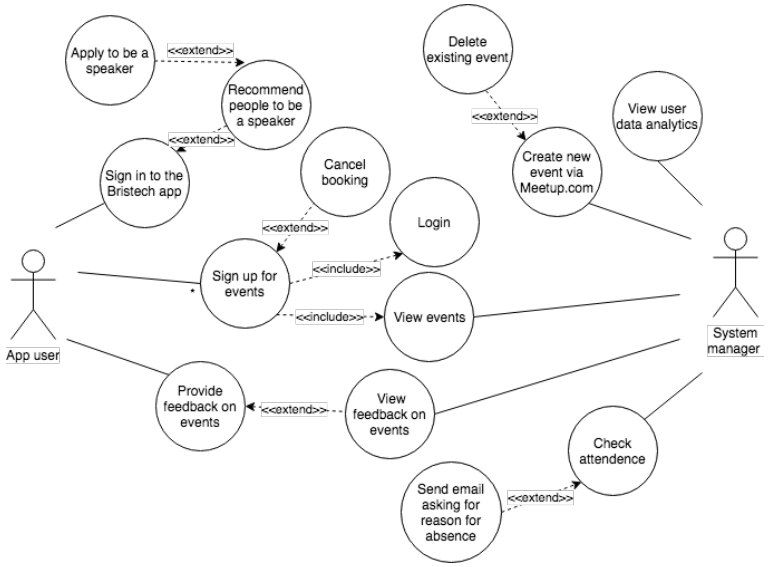



Figure 1: Use Case Diagram

## 2.3 Core use-case goals

### 2.3.1 Basic flow for Signing up for events

1. User launches the application and is greeted by the log in screen.
2. User logs in and is presented with the upcoming events screen.
3. User scrolls up and down the cards, deciding 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 the "REGISTER" button

### 2.3.2 Basic flow for Creating an A

1. User launches the application and is greeted by the log in screen
2. User presses the "Create account" button and is transferred to a form for creating an account
3. User inputs their user data and presses the "CREATE ACCOUNT" button
4. Account is created if the information is valid and unique
5. User returns to the homepage by pressing a back button

### 2.3.3 Basic flow for Volunteering a Speaker

1. User launches the application and either has to log in or is already logged in and so taken to the upcoming events page by default.
2. User toggles the sidebar by swiping right on the home-screen
3. User enters the Give feedback screen by pressing "Volunteer a speaker" button on the sidebar
4. User clicks the link to the google poll, which opens in browser
5. User provides information on the speaker they would like to suggest
6. User may return to the app and press the back button to return to their previous page

#### 2.3.4 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 Give feedback screen
5. User clicks the link to the google poll, which opens in browser
6. User provides feedback
7. User may return to the app and press the back button to return to their previous page

#### 2.3.5 Alternative flow for Leaving feedback

1. User launches the application and either has to log in or is already logged in and so taken to the upcoming events page by default.
2. User toggles the sidebar by swiping right on the home-screen
3. User enters the Give feedback screen by pressing "Give feedback" button on the sidebar
4. User clicks the link to the google poll, which opens in browser
5. User provides feedback
6. User may return to the app and press the back button to return to their previous page

#### 2.3.6 Exceptional 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 Give feedback screen
5. User clicks the link to the google poll, which opens in browser
6. User has no internet and so google poll does not load
7. User uses the alternative flow for leaving feedback once they have access to internet

#### 2.3.7 Basic flow for Creating events

1. Manager logs in to meetup.com and creates an event
2. Event is now displayed in an Event card in the Upcoming Events screen on the app

### 2.4 Atomic requirements for viewing and registering for events

#### 2.4.1 Functional requirements

1. User shall be able to open sidebar to navigate to other screens. Sidebar will contain:
    1.1. Upcoming events - Scrollable view of all upcoming events
    1.2. Past Events - Scrollable view of all past events
    1.3. User Settings - Returns to log in screen
    1.4. Volunteer a speaker - Screen allowing users to volunteer other people/themselves as a speaker through a google poll link
    1.5. Give feedback - Screen allowing users to provide feedback through a google poll link
2. User shall be greeted by a User Settings page upon launching the app.
    2.1. If the user is already logged in, they will be greeted by the default Upcoming events page

3. User shall be able to browse through all available upcoming events listed on the Upcoming Events screen.
    3.1. In order to load upcoming events correctly an Internet connection is required
    3.2. Users will be notified that the app requires an internet connection when installing.
    3.3. Events page features a series of scrollable cards
    3.4. Requires integration with Meetup API to load events properly
4. User shall be able to view past events
    4.1. Same requirements as upcoming events, but with past events displayed on the cards instead
5. User shall be able to view event title, time and short description
    5.1. Details taken from event listing on meetup.com
        i. Changes propagate from meetup.com to the app due to frequent server updates
6. User shall be taken to the Event Detail screen on event card press
    6.1. User can tap anywhere on the event card to be taken to the detail screen
        i. This enables easy usage for less dextrous people
7. User shall be able to scroll through the event's details
    7.1. Details shown from information on meetup.com
    7.2. User can scroll up and down to see all the details of the event
8. User shall be able to register for an event by pressing the register button
    8.1. The button is at the bottom of the description
        i. This ensures registrees have read what the event is about before signing up
    8.2. Button changes colour when pressed to symbolise state change in registration
9. User should be able to unregister for an event by pressing a button
    9.1. Button should change colour when pressed to notify users of a return to an unregistered state
10. User shall be able to go back to the the previous screen by pressing a back button
    10.1. This button should be at the top left
        i. This will be consistent with the whole of the app design, including the side navigation bar

#### 2.4.2 Non-functional requirements

1. Application shall fetch the upcoming event list within 2 seconds after starting the application
    1.1. This will fail if there is no Internet connection
2. Application shall start a background running service to connect and write to the database on event registration
    2.1. User attendance is stored in the database
        i. This can be examined by system administrators
    2.2. In app connections to this data are used to display who attended and who failed to

# 3 Architecture

## 3.1 Key architecture features

The overall requirement was for us to build a system to manage the interaction between the managers and attendees of their talks. As an answer to this, we could've taken either the web application or mobile application route. Our chosen platform was an Android mobile application and the key architectural drivers that have shaped 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've had 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've been able see what aspects of functionality need to be implemented first.
- **Primary Functionality**
  These aspects of functionality are those which are critical to achieving the end business goal. For our system, this has been the creation of events for the managers and also the logging in and sign up for events for the users. However, we've ensured that our architecture has always been open to change and improvement.
- **Architectural Concerns**
  We could have encountered 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. In order to simplify providing feedback, we are using google poll, as it makes it easier for the manager to update at his leisure.
- **Constraints**
  Technical: Our system will have to integrate with Heroku services who will be hosting and storing our various databases.
  Non-technical: We have had to ensure that we aren't breaching any standardised data protection laws regarding the personal information of our users. The allotted time and technical abilities of our developers have also been 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 Heroku services for this. As well as this we are integrating with the Meetup.com API to ensure consistency with the application and website. The integration with meetup means that there has been a degree of communication between the two platforms. The holding of our databases for members and all their information will be managed by implementing MySQL.

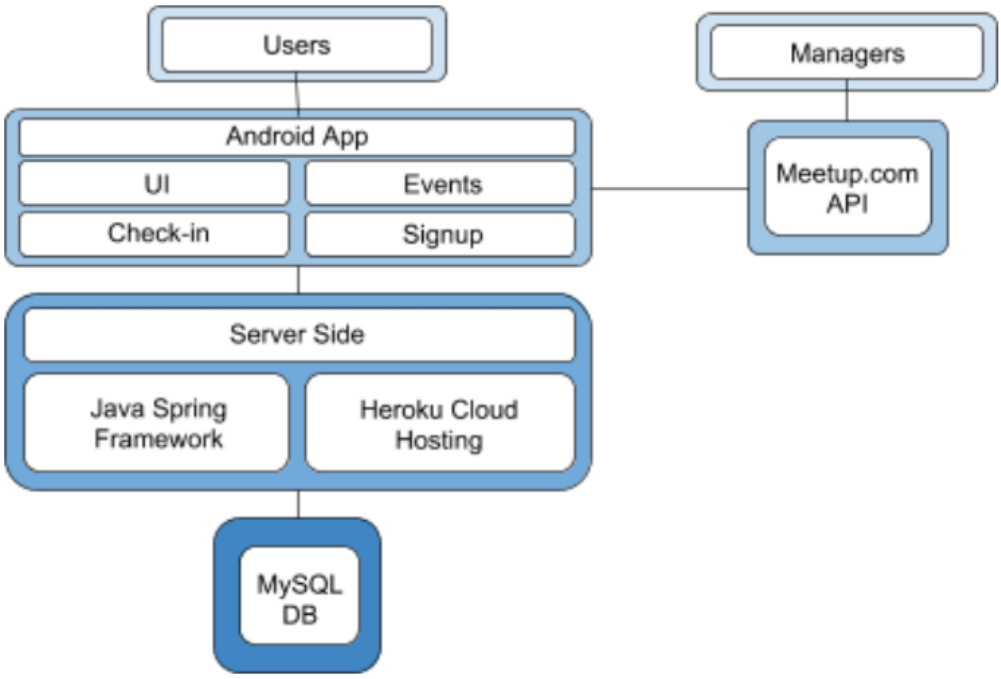



Figure 2: High Level Architecture Diagram

# 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

The core functionality we will be exemplifying here is the create account function. This is an essential core function as without it users will be unable to access any further interactions with the app. We therefore test that correct email inputs are given and that the data for username and password is both in the correct format and correct in correspondence to other data.

| Functionality | Data Input | Outcome |
|---|---|---|
| Email | 1234@my.bristol.ac.uk | Pass: valid email input |
| Email | fdlghdakb | Fail: invalid email input |
| Username | Bristech47 | Pass: valid string for username |
| Username | memes™ | Fail: contains invalid character |
| First Name | Richard | Pass: valid string for first name |
| First Name | 爱chicken | Fail: contains invalid character |
| Surname | Davies | Pass: valid string for surname |
| Surname | 死亡ofme | Fail: contains invalid character |
| Password | 123456789 | Pass: valid string for password |
| Password | ab87jiKlm | Pass: valid string for password |
| Password | oJ9m©jiK | Fail: contains invalid character |
| Password | a | Fail: password too short |
| Password | kjdhflksjdhflksjdhflskjdhflskhdf | Fail: password too long |
| Retype Password | 123456789 | Pass: correct string equal to password |
| Retype Password | abcde | Fail: string not equal to password string |

Figure 3: Development Testing Core Function Tests

# 5 Release Testing

## 5.1 Strategy

Our strategy for release testing involves establishing a monthly release schedule with an assigned team to perform a sequence of tasks, as detailed in the flow steps, to simulate using the application during normal usage. We will test key areas such as, account setup and login, events interaction and leaving feedback.

We will utilise dummy variables and code to simulate events and accounts needed to warrant interaction in the initial stage, however in the later releases we will be able to directly interface with real data as we will have API connections.

These system tests will be performed in team by the developers to ensure consistent quality and that the whole system is functioning as desired before it is presented to real users.

## 5.2 Core Goal Tests

Below we will outline the tests needed to ensure a core component of our system works. We will use 'Check-in 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.

| Test | Description of Correct Function |
|---|---|
| 1 | User clicks button to REGISTER to an event and the button changes colour in acknowledgement |
| 2 | Server registers user signing up and adds them to list of attendees |
| 3 | A notification alerts user to allow location services |
| 4 | On the day of the event, users entering the engine-shed geofence are detected. |
| 5 | When an attendees phone is detected within the geofence at the correct time a notification is sent to alert user that they have been checked in |
| 6 | User is added to the list of attendees who successfully turned up |
| 7 | If the user does not enter the geofence then they are sent a notification that they failed to attend |
| 8 | After the event the system manager can see who attended and who didn't as their names will be highlighted according to their status |

Figure 4: Release Testing Core Function Tests

# 6 OO design and UML

## 6.1 Static app activity design

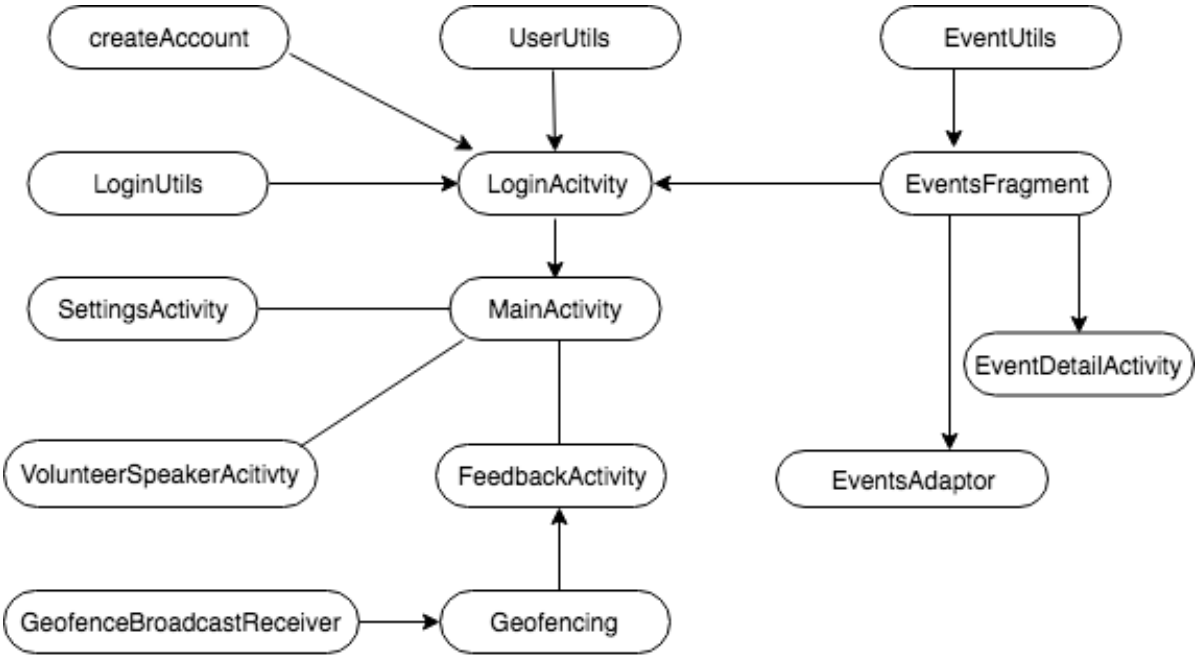



Figure 5: Front end application UML diagram

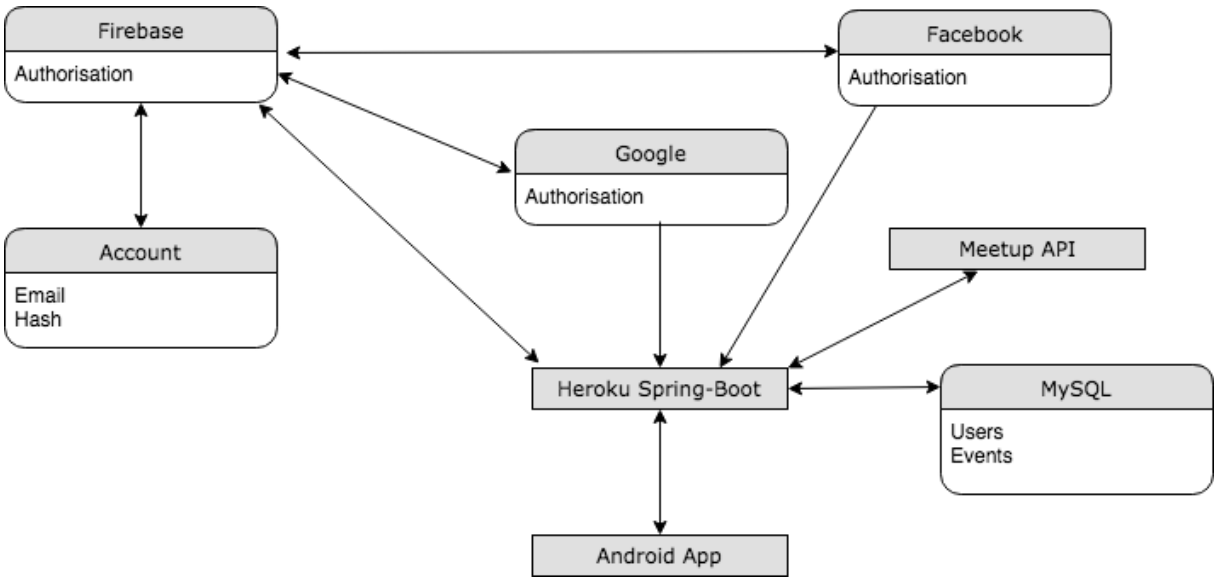



Figure 6: Back end UML diagram

The above diagrams show how the different parts of the system fit together and interact with each other. They demonstrate respectively how the app and the back-end systems are laid out and how the various aspects interact with each other.

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

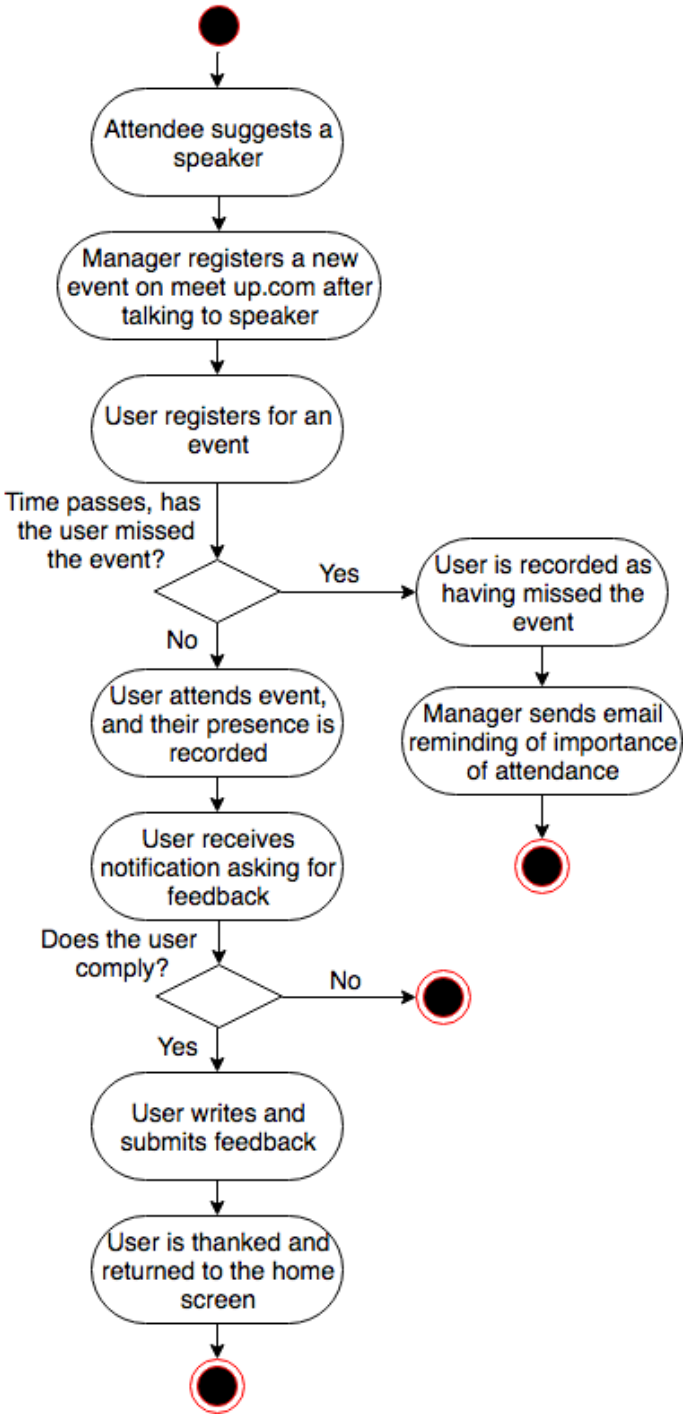



Figure 7: Dynamic app usage integration design for a user attending an event

The activity diagram above models the overall control flow of the system. The activity diagram is chosen to visualize 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 modeled are sequential.

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.

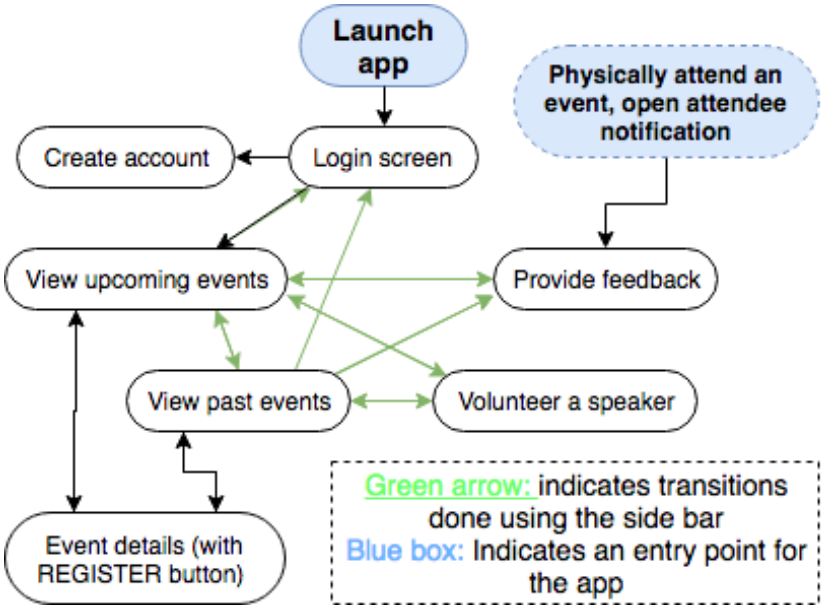
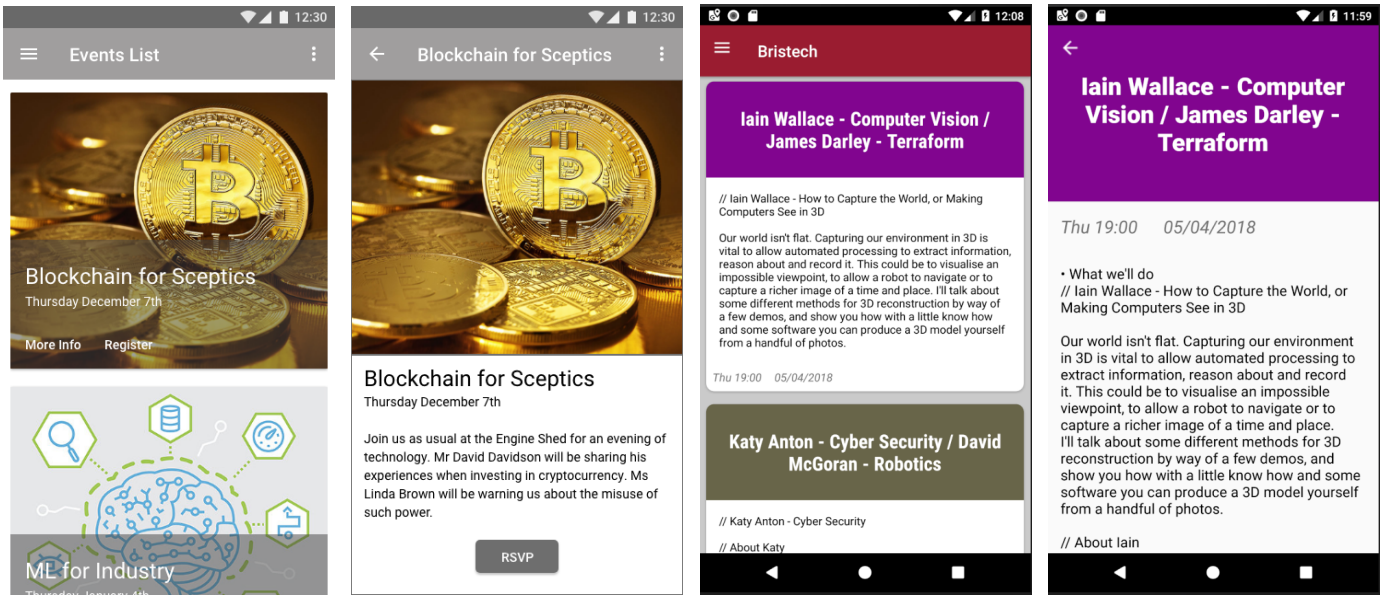



Figure 8: Basic app activity design

# 7 User Interface Design

## 7.1 Design Examples




(a) Events List - Prototype (b) Event Details - Prototype (c) Events List - System (d) Event Details - System

## 7.2 Discussion

The images above show two screenshots each from both our User Interface prototype and our current working version. The two left hand images are taken from our initial prototype for the application, the two on the right are from our current working version. They show the events page and the event details page. The events page is the central interface of the application via which you can access all other functionality, it is therefore to have it be both useful and aesthetically pleasing. The event details page is a more detailed description of the event that has been chosen from the events page, listing date and topic, and going into more information about the content of the talks. It is also through the event details page that you can sign up to attend events, via the REGISTER button.

As you can see they are very similar: both implement a card system on the events page as a means to display multiple events at once. This page is scrollable and fills up with as many events as there are currently listed. Also, both feature a full explanation of what the event will be on the event details page as well as a REGISTER button. Both feature the ability to pull out a sidebar menu from the events page, using the 'hamburger' in the top left

hand corner of the navigation bar.

Differences between our prototype and our working version include not having a picture for each event. This is because we sync the data directly to the event listing on Meetup.com which does not have a picture associated with it. As such, there is no image to display, however if the system manager were to specify it then they could have a picture displayed. Also, in our working version, in order to fill up space and normalise the size of the cards, we have a short description of the event visible in the events page. This is to give a basic understanding of what the event is about without taking up too much space. If the user wishes to view more they can simply click the card and be taken to the event details page.

We have also employed a more colourful colour scheme in our working version as we viewed the monotone greys and blacks in our prototype version to be too drab. Instead we opted for maroon as our primary colour choice, symbolising Bristol and its university.

# 8 Acceptance Testing

## 8.1 Strategy

Our approach to acceptance testing is a combination of both heuristic evaluation and both the quantitative and qualitative aspects of user evaluation. Our approach to heuristic evaluation is to utilise Nielsen Heuristics to gauge the degree to which our software product appears usable, analysing using a 1-5 scale to determine the degree to which the product satisfies the heuristic.

In regard to user evaluation, we will take the software product to one of the events for which the product was designed and test with users. These tests will take the form of timing how long users take to complete tasks they are given using the app, alongside observing specifics of their usage: how many clicks, how many noticeable pauses, etc. After completing the tasks, users will be given a questionnaire to using Likert scale to determine more qualitative aspects of their time using the product, such as their attitude towards the User Interface, the ease with which they can use the app, etc.

## 8.2 Core Goal Tests

The use case we will demonstrate is that users can register for events they are interested in and can cancel their registration if they no longer wish to attend. This is an essential use case as it is the fundamental purpose for which this app is built, and as such needs to be tested thoroughly to ensure that it is easily understandable to users and functions optimally.

### 8.2.1 Heuristic Evaluation

1. **Visibility of System Status**: e.g. 4/5, when users press the RSVP button it changes colour to indicate the change in the status of the user's foreseen attendance
2. **Aesthetic and Minimalist Design**: e.g. 5/5, there is only one button for the user to click, meaning that they will find it difficult to get confused and push something incorrectly. This streamlines usage

### 8.2.2 User - Quantitative Evaluation

Tasks include:

1. Find specific event from events page
2. Open the event description and RSVP as going
3. Change RSVP to no longer attending

Whilst user performing tasks watch for:

1. How many clicks to achieve task
2. How many noticeable pauses
3. Number of times back button is used

#### 8.2.3 User - Qualitative Evaluation

Questions include:

1. How simple was it it to use the app to perform the tasks assigned?
2. What is your opinion on the presentation of the user interface?
3. What is the likelihood you would use this app?
4. What is the likelihood you would use the app if it offered you special features, such as early signup for events etc?

# 9 Project Summary

Overall the project has been a success and fulfilled the majority of its aims. A piece of software has been built which satisfies the requirements to provide a platform through which Bristech managers can capture data about their attendance numbers for each event. There is also significant scope for Bristech to use our foundational code to expand upon in their own way, which was a key design choice we decided upon. Given that one of the requirements was for the application to be white label, we have made the code repositories available for any person to fork if they desire to make a similar application for managing their own events.

Having performed our acceptance testing we can state that 83% of users tested said that they found performing the assigned tasks, such as finding an event and RSVPing for it, to be simple or very simple. These simple tasks were performed in an average of 7.8 seconds and with average number of clicks being 2. Thus, we can say that the app is well received by users and therefore has great potential for the future.

We at 47 Degrees wish to express our thanks to Bristech for allowing us the opportunity to tackle this project. We hope that you are pleased with our final output and that you will take our foundational product forward with any future expansions.

# 10 Documentation

## 10.1 Google Drive

All materials used to compile portfolio and video can be found at: https://bit.ly/2F2sfXi

## 10.2 Code Repositories

Code repository for application can be found at: https://github.com/TheWalkingFridge/Bristech-App
Deployment instructions for the application can be found in the accompanying Readme file in the repository.

Code repository for server can be found at: https://github.com/TheWalkingFridge/Bristech-Server
Deployment instructions for the server can be found in the accompanying Readme file in the repository.

## 10.3 Promotional Video

Promotional video can be found at: https://youtu.be/kJiZLTGYyPU