Alexander Ford 18361148
Andre Tertzakian 17682148
Neema Boutorabi 18694142
Omar Dzick 36579143
Emre Pekel 48488134
Konrad Izykowski 26506148
Luigi Sacco 21139143

Beacon - Requirements, Vision, and Scope

Motivation/Opportunity

People have a limited ability to stay connected to social events going on around them. There is no seamless way to see what events are currently going on near your current location. Additionally, there is no way for small event planners to advertise to potential attendees in their area while their event is occurring. Beacon solves these problems by allowing users to pin their event location and information to a shared map that all users can view in real-time.

This app will separate itself from other platforms, such as the event planning system offered by Facebook, by not focusing as much on the planning aspect of social events. Instead, Beacon will try and encourage on-the-fly decision making, by focusing on nearby events that are occurring at the moment. Beacon will also have a live comment page for every event, that will allow event attendees to share their experiences at an event as it's happening.

Problem Statement

The problem of	Spontaneously finding/planning events
affects	1) People who have no plans and are looking for fun, nearby activities/events to attend. 2) People who are hosting an event and want people to attend it.
the impact of which is	1) Missing nearby events that you are unaware of 2) Having few/no people attend your event
a successful solution would be	An application that allows user to search for nearby events/create events for others to find.

Product Position Statement

For	The public
Who	1) Want to find a nearby event 2) Want to broadcast an event they are hosting
Our System	Is an android app that communicates with a server

That	Shows events around the user's location in real-time, and allows users to create their own events.
Unlike	Other applications where you plan events ahead of time.
Our Product	Allows for spontaneity and focuses on proximity.

User Demographics

Searching Users: Users who are looking for something to do will be able to easily search for nearby events by name, tags, etc. The app will be designed to be immediately easy to use and will not require any complex configuration. Users who search for events will want the app to operate quickly and show a map of nearby events.

Hosting Users: Users who want to share events they are hosting will be able to create their events easily by providing basic information such as event type, location, duration, etc. The interface for creating events will be simple and user-friendly. Administrator will be able to broadcast their official events. Users who are hosting events will want the app to be a secure and responsive platform for sharing information about their public/private events.

Feature List

1. Real-time mapping of events and hangouts occurring within a user's vicinity
    a. Beacons overlayed on google map
    b. Shows user's current location
2. An option for the user to plan events and hangouts for future dates and make them private/public
    a. Shareable event information
        i. Title
        ii. Event Description
        iii. Location (address)
        iv. Start and end time
        v. Media related to event (photos/comments)
        vi. Tags for the event'+' button for easy event creation
    b. Public events can be seen by any nearby user
    c. Private events can only be seen by those with a direct link to the event page
    d. Event are automatically deleted 24 hours after the event's end time
3. Searching and sorting through events
    a. keyphrase
    b. distance
    c. tags

    d. date
4. A system for administrators to add persistent Beacons with public posts
5. Ability for users to post comments/photos to an event listed through the app

Constraints

The application will be majorly constrained by the Firebase platform. The price associated with using this service will be charged based on a “as-you-go” basis. So, as Beacon grows, so will its costs.

Furthermore, if upgrades want to be made that are outside of the capabilities supported by Firebase (for whatever reason), it would be very difficult. The vital features of the app, as well as the majority of the backend functionality, are supported by Firebase, and replacing these would require much time and effort.

Scope and Limitations

The following features will not be implemented in our product

1. User “friend lists” within the Beacon application
2. The ability to broadcast an event more than 48 hours ahead of time
3. Direct messaging between users

Assumptions and Dependencies

1. Beacon will depend heavily on Google services, especially the services offered by Firebase, to function properly. This dependency assumes that Google will not cease to provide these services.

2. Beacon will require a minimum amount of events and users. If there isn’t a critical mass of users who actively utilize Beacon, the primary goal of the app won’t be achieved.

Use Cases

Actor	Goal
All actors	- Sign in to application
Searcher	- Search for an event - Upload a comment/photo to an event - Change personal settings
Host	- Create an event - Edit/delete their event page
Administrator	- Create a persistent Beacon

Identification	1 - Search for an event
Primary Actor	Searcher
Stakeholders and Interests	- Hosts
Preconditions	- If Searcher has just opened the app, they have a stable internet connection and there are existing events within the Searcher's search radius
Postconditions	- User has found an event to attend
Main Success Scenario	1 - Searcher opens the Beacon app 2 - A list of nearby Beacons is displayed 3 - The Searcher selects an event that suits them 4 - Event page, where photos and event information is displayed to searcher
Extensions/Alternative Flows	2.1 - Searcher enters a query to filter events 2.2 - An interactive map with pins tagged to Beacon locations is displayed 4.1 - Searcher attends the event 4.2 - Searcher favourites the event 4.3 - Searcher returns to list of events
Error Scenarios	2a - There are no nearby events 2a.1 - Searcher is prompted to create their own event 2.1a - Searcher's query returns no results 2.1a.1 - Searcher is asked to submit a new query

Open Issues	- ~~Should a Google login be a precondition for viewing events?~~

Identification	2 - Upload a comment to an event
Primary Actor	Searcher
Stakeholders and Interests	- Hosts - Administrators - Other searchers
Preconditions	- Searcher has logged in with a Google account - Searcher has found an event they wish to upload a photo to
Postconditions	- Other searchers who view the Beacon page can view the comment
Main Success Scenario	1 - Searcher selects an event page 2 - Event information and existing comments are displayed 3 - Searcher selects option to post a comment 4 - Searcher types their comment 5 - Searcher uploads content to event page 6 - Comment appears on event page for all other users
Extensions/Alternative Flows	4.1 - Searcher attaches a picture to the comment
Error Scenarios	3a - Searcher is not within 50km of the event 3a.1 - no option to view event page or upload a comment will be displayed 5a - Comment fails to upload 5a.1 - Searcher is prompted to upload their comment again
Open Issues	- ~~Does a searcher need to be near an event to upload a comment?~~

Identification	3 - Change user settings
Primary Actor	Searcher
Stakeholders and Interests	- Searcher

Preconditions	- Searcher has logged into the app with a Google account
Postconditions	- User has changed personal settings
Main Success Scenario	1 - Searcher opens the Beacon app 2 - Searcher navigates to settings tab 3 - A list of personal settings the user can change are displayed 4 - Searcher alters desired settings 5 - Searcher's settings have been changed
Extensions/Alternative Flows	4.1 - Searcher changes desired search radius to a value in the range from 0 km to 100 km 4.2 - Searcher enables or disables notifications for favourited events 4.3 - Searcher signs out of app 4.3.1 - User is returned to the sign in page
Error Scenarios	4.3a - Searcher accidently hit sign out button 4.3a.1 - Additional alert dialog is presented to confirm sign out.
Open Issues	~~- What additional settings should be implemented?~~

Identification	4 - Create an event
Primary Actor	Host
Stakeholders and Interests	- Searchers
Preconditions	- Host has logged in with a Google account
Postconditions	- A Beacon for the host's public event is created, and can be viewed by nearby users - A Beacon for the host's private event is created, and can be viewed only by users who have the shareable link
Main Success Scenario	1 - Host opens app and chooses to create an event 2 - Empty fields to enter in event information are displayed to host. 3 - Host enters information about their event -name -start time/end time -description -picture(s)

	-tags (for queries) -private or public 5 - Host adds pin on map for location of event 6 - Beacon is created
Extensions/Alternative Flows	5.1 - Beacon is created at host's current location 5.2 - Host specifies a location, somewhere near their location for the Beacon to be created 6.3 - If event is private, host receives a sharable link to the event page 6.3.1 - Host shares the link for their friends to view 6.4 - If event is public, Beacon becomes viewable to all nearby searchers
Error Scenarios	5.2a - Host attempts to create Beacon at a location outside of the allowable range 5.2a.1 - Google maps will be limited within 100km of a user's location. Locations outside that range will not be viewable or searchable. 6a - name field, start time, or location field are empty or invalid (ie start time is before the current time) 6a.1 - searcher will be prompted to enter in a valid value for the fields violated.
Open Issues	~~- How far away from the Host's current location can a Beacon be created?~~ ~~- Should a user not be able to create an event with offensive words in the title or description?~~ ~~- Should a Host be able to set the visible radius of their Beacon? If not, how large should the view radius of a standard Beacon be?~~

Identification	5 - Edit an event page
Primary Actor	Host
Stakeholders and Interests	- Searchers
Preconditions	- Host has logged into the app through a Google account. - Host has created a Beacon for their event.
Postconditions	- Host's event information has been altered
Main Success Scenario	1 - Host opens app and navigates to their event page 2 - Event page is displayed with additional option to edit page

	3 - Host selects the edit page option 4 - Event information fields on the page now become editable 5 - Host changes the event information 6 - Host confirms changes and Beacon is updated
Extensions/Alternative Flows	5.1 - Host changes name, location, time, description, picture(s), and/or tags 5.2 - Host deletes comment(s) on the event page 5.3 - Host cancels the event (ie deletes event page) 5.4 - Host changes event page from public to private or vice versa
Error Scenarios	2a - The host is not the creator of the event in question 2a.1 - No option for editing the event will be available to them 5.1a - Host attempts to change the location of the event to somewhere outside of the allowable range, or change name, or time to empty or invalid (ie a time before the current time) 5.1a.1 - No location further than 100km from the user will be visible or searchable. 5.1a.2 - Changes are not applied, host is prompted to enter valid entries for all fields
Open Issues	~~- Should the host be able to change their event from public to private, or vice versa?~~

Identification	6 - Create a persistent Beacon
Primary Actor	Administrator
Stakeholders and Interests	- Searchers
Preconditions	- Administrator has logged in with a Google account, and is the creator of the event page in question - Administrator has received payment for advertising
Postconditions	- Administrator can broadcast their event to a wider radius - Persistent Beacon has become distinguishable from other Beacons
Main Success Scenario	1 - Administrator opens app and goes to the event page 2 - Event page and additional administrator options are displayed 3 - Administrator specifies broadcast radius

	4 - Administrator broadcasts the event 5 - Beacon is updated
Extensions/Alternative Flows	5.1 - Beacon pin for the event gets distinguishable by color as specified by the Administrator 5.2 - Administrator pins are persistent and do not automatically get deleted
Error Scenarios	3a - Administrator attempts to broadcast to a radius larger than allowed 3a.1 - Administrator is prompted to specify a valid broadcast radius
Open Issues	~~- For advertised events, should the app send push notifications to searchers within the radius?~~

Identification	7 - Sign in to application
Primary Actor	All users -Searcher -Host -Administrator
Stakeholders and Interests	- Searchers
Preconditions	- User has downloaded the application and is running it on their phone
Postconditions	- User's credentials have been saved and user has access to all features of the application
Main Success Scenario	1 - User starts application 2 - The sign in page is displayed, with a google sign in button 3 - User chooses to sign in, and is prompted with an option to select their google account. 4 - User is authenticated 5 - Main map view of the application is displayed
Extensions/Alternative Flows	2.1 - User has already authenticated themselves previously then no sign in page is displayed and the main map view is displayed.
Error Scenarios	3a - User does not have a google account 3a.1 - User will be prompted to create a google account. User will not have access to the

	application until they create a google account
Open Issues	~~Should other methods of authenticating (ie facebook account) be implemented?~~

Non-Functional Requirements

Performance & Scalability Requirements:

- The app should be usable when offline and should notify the user if certain features require a stable internet connection. If the user wants to upload event data, the app should store that data locally and push it to the server automatically when a connection is restored.

- As the app grows in size, the functionality of the app should not be impacted. That is, the user should not see longer load times for any aspect of the application. This should be achievable through smart UI design and data caching.

- Data usage should be kept to a minimum and should not grow proportionally with number of users and events. In other words, even when our real time database gets to a point where it contains large volumes of data, only the most relevant data should be fetched and displayed on the UI. This can be accomplished by “data flattening” the JSON tree that stores our data in firebase and also by using smarter design patterns to transfer data between modules in our application (such as using intent extras provided by the android framework)

Security Requirements:

- Users that want to host a private event should be able to get a secure, shareable link that they can share with their friends without worrying about the event’s exposure to the public. This means that the people without the Beacon link cannot see the event on the app even if they are nearby.

- Pulling and pushing data to Firebase’s real time database must be moderated by certain database rules. This is to restrict all users from being able to retrieve or change sensitive information regarding events, users and media.

Software Quality Attributes:

- We anticipate that as we build our application, we will frequently need to revise our database schema. For this reason, the code base should not depend heavily on the structure of the firebase database, this way as we change our database the amount of code that needs to be refactored can be kept to a minimum.

- All methods that deal strictly with data manipulation or UI tweaking should exist in static utility classes that can be accessed from anywhere within the application. This is to improve modularity and code re-use since such functions are completely independent of the android framework.

- Packages should be used to organize and separate classes that rely heavily on the android framework. For instance a package should exist for each of: (Activities, Fragments, Adapters, Services, Views, Resources, etc..)

- All Java classes should contain no more than 1000 lines of code and xml files should contain no more than 400 lines of code. We believe these are reasonable bounds as beyonds this the code becomes nearly unreadable. Accomplishing this can be done through the use of better modularity/ helper classes/ resources (such as styles.xml).

- All src files should be formatted according to a standard code formatter, the default android studio code formatter should suffice.

- Every class/method that exceeds 50 lines of code should be provided with adequate documentation.

Changes and Rationale:

We made several revisions to our requirements document in response to the feedback provided and due to changing circumstances regarding the progress of our application. We decided that the following modifications were appropriate:

- A new use case for all actors regarding initially signing in to the application was added.

- A new use case for changing a user's personal settings.

- All open issues have been resolved. Use cases have been modified to reflect those changes.

- Additional steps were added to the main success scenario in use cases 4, 5, and 6 to better illustrate the interaction between the actor and application.

- We removed the Safety/Censorship portion of our non-functional requirements. Adding server side moderation seems like a tall order given the time we have left to complete the application. Instead, we believe that since admin users can manually delete content from the database, this will serve as an adequate solution for now.

- We got rid of all requirements regarding payments and in-app purchases as we believe it is no longer feasible to implement these features within our application.

- Overall data usage was given more attention in the revised document. We added a point about how we have attempted to minimize data usage across the app and how we plan on restricting database accesses to ensure that our data is secure.

- Several additions were made to the software quality portion of the non-functional requirements. These were included as they provide tighter restrictions on how our code base is structured. This will be useful going forward since keeping the code base readable and logically separated will help immensely when implementing additional features or when refactoring.