Use Cases and Logical Architecture

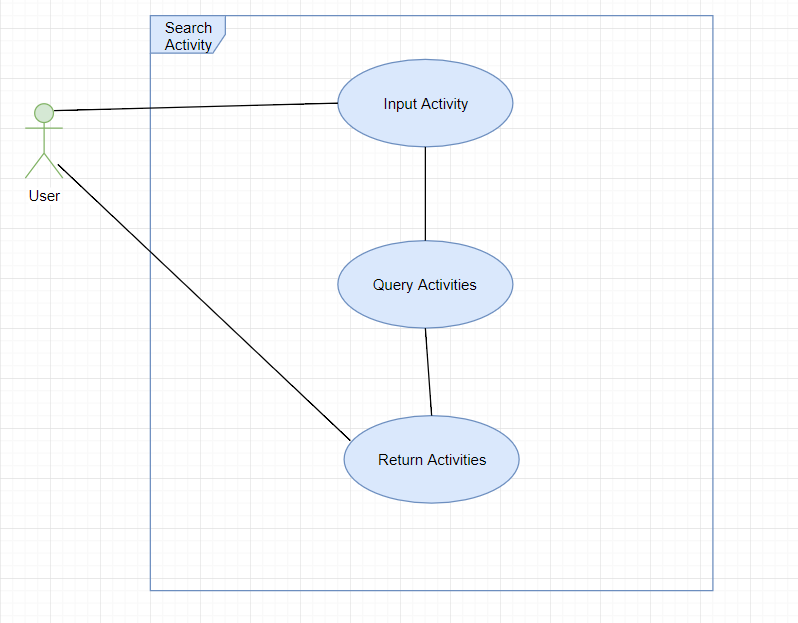
* XID: X00119025
* Name: Martin Casey
* Project Title: Community Application

Section 1: Use Cases:

**Use Case 1: Search for Activity**

|  |  |
| --- | --- |
| Use Case Element | Captures the processes undertaken when a user searches for an activity in the application. |
| Use Case Number | 1 |
| Application | The function relates to the search feature of the application. |
| Use Case Name | Search for Activity |
| Use Case Description | The user of the application is deciding to look into activities that are available on the community application, regardless of their current location. |
| Primary Actor | The user who is trying to search for an activity in the community to take part in. |
| Precondition | 1. The user must have access to an internet service. |
| Trigger | The user accessing the search bar in the Community application. |
| Basic Flow | 1. The user inputs an activity they are interested in into the application search bar. 2. The application sends a query request to the database. 3. The database queries its tables to find a for activities that match the inputted parameter by the user. 4. The database returns a collection of activities and organisations that host them to the application. 5. The application displays the list of activities to the user. |
| Alternate Flows | 1. The user enters their area of interest with incorrectly spelling. The application analyses the text the user entered and calculates what it believed the user meant to input. It sends a query request to the database using the calculated text and the database returns a collection of activities based on the calculated text to the application. The application displays this list to the user prefixed with the message “Is this what you meant?”. 2. The activity the user inputted does not exist or could not be found hosted by any of the organisations in the applications database. The application returns a message to the user stating that it could not find the activity they were looking for. |

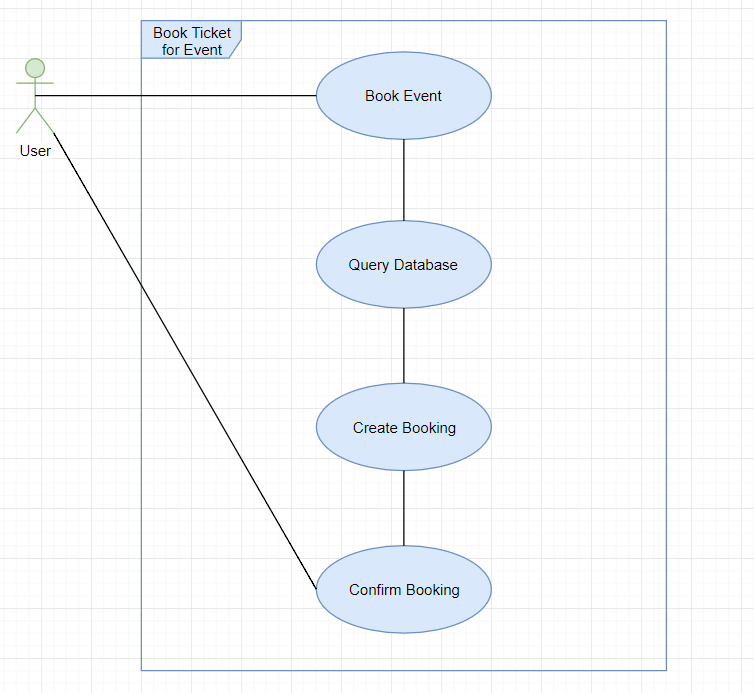
**Use Case 1 Diagram**



**Use Case 2: Book Ticket for Event**

|  |  |
| --- | --- |
| Use Case Element | Captures the processes undertaken when a user books a ticket for an event. |
| Use Case Number | 2 |
| Application | The function relates to the booking feature in the application. |
| Use Case Name | Book Ticket for Event |
| Use Case Description | The user of the application is looking to book themselves a ticket for an event/activity they are interested to part take in. |
| Primary Actor | The user who is trying to book a ticket for an event/activity. |
| Precondition | 1. The user must have clicked on an activity or event link. 2. There must be an activity or event created and hosted by an organisation. |
| Trigger | The user clicking on the “book ticket” button. |
| Basic Flow | 1. The user clicks on the ‘Book Ticket’ but on an activity, they want to partake in. 2. The user selects the time and day they want to partake in the activity. 3. The application sends a query request to the database. 4. The database queries its tables. 5. The database finds a free space associated with the organisation that is hosting the activity the user wants to book. The database enters the users enters the users details into the booking slot. 6. The database returns the newly created booking record to the application. 7. The application displays a ‘Booking Confirmed’ message to the user. |
| Alternate Flows | 1. No free slot related to the organisation hosing the activity could be found. The application returns a message to the user stating that the booking they wanted was not possible. |

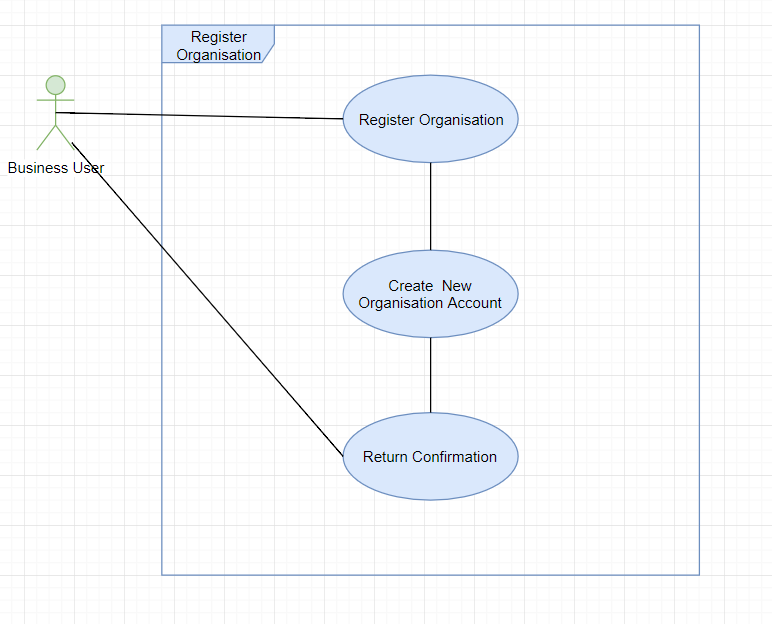
**Use Case 2 Diagram**



**Use Case 3: Register Organisation**

|  |  |
| --- | --- |
| Use Case Element | Captures the process of how a business user registers their organisation on the application. |
| Use Case Number | 3 |
| Application | The function relates to the registration feature of the application. |
| Use Case Name | Register Organisation |
| Use Case Description | A business user is made aware of the application and wishes to register their organisation on the application with a business account, so that users of the application can be informed of events the organisation hosts in the community. |
| Primary Actor | The business user trying to register their organisation on the application. |
| Precondition | 1. The business user must have access to the internet. |
| Trigger | The business when on the registration page clicks the create business account link. |
| Basic Flow | 1. The business user, when on the registration page clicks the link to create a business account. 2. The business user fills out a form on the application, detailing information about their organisation, such as name, location, and activities hosted. 3. The application sends a query request to the database using the information provided in the registration form. 4. The database creates a new record in its organisations table. 5. The database returns the new record to the application. 6. The application displays a ‘Confirmed Registration’ message to the business user. |
| Alternate Flows | 1. The application has reached the specified limit of the number of servers it can expand to, so the database is unable to create a new organisation record due to storage issues. |

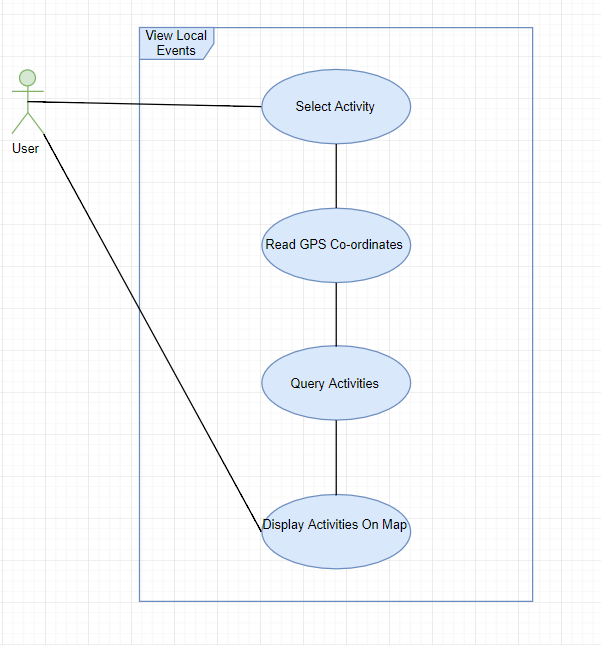
**Use Case 3 Diagram**



**Use Case 4: View Local Events**

|  |  |
| --- | --- |
| Use Case Element | Captures the processes undertaken when a user requests to examine what areas in the local community are prone to certain types of activities. |
| Use Case Number | 4 |
| Application | The function relates to the map functionality of the application. |
| Use Case Name | View Local Events |
| Use Case Description | A user wants to examine the local area around them and see what areas are provide or are prone to what activities, based on a parameter inputted by the user. |
| Primary Actor | The user trying to use the application to find out what activities happen in what areas. |
| Precondition | 1. The user must have access to the internet. 2. The area the user is operating in has access to satellite or cell tower transmissions. 3. The user must have the GPS receiver functionality of their mobile device enabled. |
| Trigger | The user clicks the activities tab and inputs what activities they wish to see displayed on the area map. |
| Basic Flow | 1. The user clicks on the activates tab and selects an activity/s they wish to see displayed on the map. 2. The application reads the current GPS location of the user. 3. The application sends a query request to the database, based on the activity/s the user select and their current location. 4. The database queries its tables for a match between the activity/s the user is interested in, and the area the user is currently located in. 5. The database returns a collection to the application containing the activities the user is interested in and the locations they are located in. 6. The application loads up the map of the area around the user and displays the locations of the activity/s the user is interested in. |
| Alternate Flows | 1. The user is currently located in an area where due to technical reasons they are unable to receive GPS transmissions. The application displays a message to the user saying it cannot get a reading of the current area. 2. The database was unable to find any activities in the user’s current local area. The application displays a ‘No Activities Found’ message to the user. |

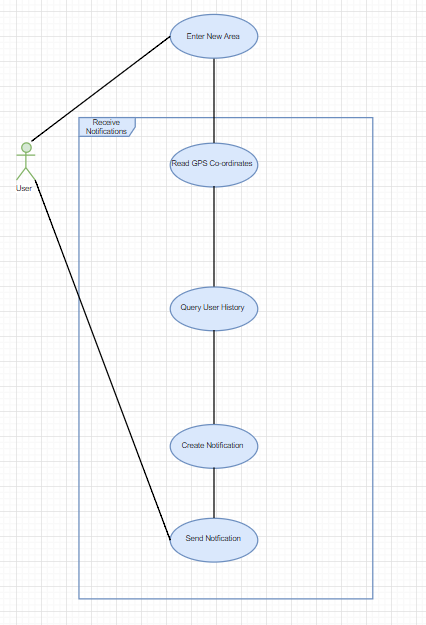
**Use Case 4 Diagram**



**Use Case 5: Receive Notifications**

|  |  |
| --- | --- |
| Use Case Element | Captures the process that occur when the user of the application receives a notification. |
| Use Case Number | 5 |
| Application | The function relates to the ability to receive notifications in the application. |
| Use Case Name | Receive Notifications. |
| Use Case Description | A user engaged on the app will receive notifications about services that are available to them via the application. The services that the user received will be optimised based on the user’s past activity with the application. |
| Primary Actor | The user of the application who will receive notifications. |
| Precondition | 1. The user must have access to the internet. |
| Trigger | The user access the application. |
| Basic Flow | 1. The user interacts with the application. 2. The application reads the current GPS co-ordinates of the user. 3. The application sends a query request to the database. 4. The database queries its tables for the user’s stored data on the application. 5. The database returns a collection of the user’s data to the application. 6. The application sends the user’s data to the notification hub. 7. The notification hub creates an abstract push notification using the user’s data. 8. The notification hub sends the push notification to the mobile devices respected notification service. 9. The respected notification service sends the notification to the user. |
| Alternate Flows | 1. The application after receiving the data from the database can notice that the user has chosen to ‘opt-out’ of certain notifications. The application only sends the notification hub data on activities/events that the user has not chosen to ‘opt-out’ of. The notification hub performs its usual operation and the user receives notifications only related to the activities they have not ‘opted-out’ of. 2. The notifications hub is unavailable due to technical issues, therefore, no notifications are created and sent to the user. |

**Use Case 5 Diagram**

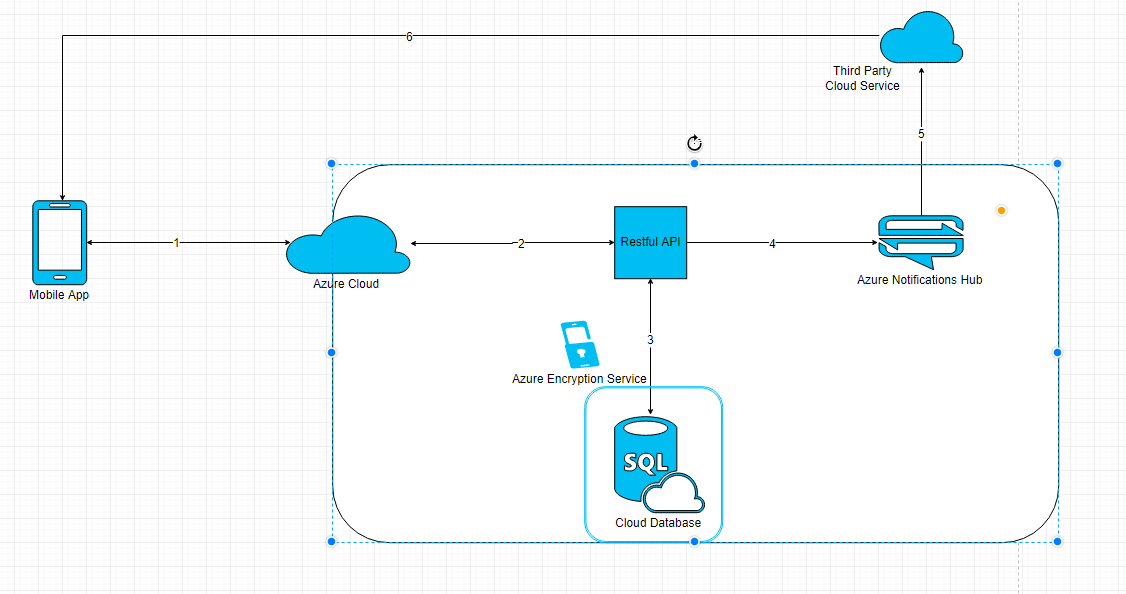


## Section 2: Prototype Schedule, Winter Semester 2017

|  |  |
| --- | --- |
| Iteration #1, Complete 25/10/2017 | Use Case 1, Use Case 2 |
| Iteration #2 Complete 15/11/2017 | Use Case 4 |
| Iteration #3 Complete 13/12/2017 | Use Case 3, Use Case 5 |

## Section 3: Logical Architecture

**Logical Architecture Diagram**



Details on Components

**Mobile App**

The community application is an ASP.NET MVC based application that will be available on the android mobile platform. The component of the application will act as the View compartment of the applications framework, which is how the user will interact with and experience the services of the application. The View will be consist of HTML, CSS, and JavaScript code. The application will communicate with the Azure Cloud service via Hyper Text Transfer Protocol Secure (HTTPS), when it needs to make request to the function requests to the back end services. The Cloud will return any functional service response to the application in the same HTTPS format.

**Azure Cloud**

The Azure cloud is on online cloud service that is used to store data, host applications and provide server features to any applications or web-services that use service. The majority of services the application will use will be located in this service. When the cloud receives a transmission from the application, it will send that transmission via HTTP to the Restful API backend component of the application to be processed. After wards when it receives a function response it sends that response back to the View compartment of the application via HTTPS.

**Restful API**

This component is located inside the Azure Applications Service. It is here where the Controller and Model aspects of the application will function. The Controller will perform the majority of the backend functionality for the application while it communicates with the other two factors of the framework. The Controller will consisted of functions and values built primarily using the C# coding language, but will have some functions that will be intended to understand JSON and SQL code used for the application. The Model of the application will also be built using C# and will also make use of SQL and JSON code, as this aspect of the framework will need to commutate with the database that application uses. Along with the Restful API, the application will also make use of the Google Maps JavaScript API to provide map and location functionality to the application. An algorithm that creates a radius around the user on the applications map, to help pin-point the location area of the user, will also be managed in the backend services.

**Azure Encryption Service**

The Azure Encryption Service, is a programing service available in the Azure Cloud that will allow for the encryption and decryption of data with the use of certificates, in order to protect important data from malicious hacker attacks. JSON formatted data will be passed through this service and encrypted to add an extra layer of protection as import user, organisation, and other is passed on to the database. When data is return from the database through this service, it is decrypted before it reaches the Azure Application Service.

**Cloud Database**

The Cloud Database is an relation database that will store the data that the application receives. Data will be both accessed and returned from this location in the cloud, via the Model aspect of the applications framework. All queries performed in the data base will be done through the use of SQL.

**Azure Notification Hub**

The Azure Notifications hub is a service provided by Azure to allow for nonfictions to be sent to mobile or other devices that are hosting applications built using the Azure cloud service. When the application needs to send a push notification to the user it will send a JavaScript formatted code message to the Notification Hub, which will go on to create an abstract push notification which it sends off to the respected notification service of the user’s mobile device.

**Google Cloud Messaging**

The Google Cloud Messaging service is the notification service that must be used when an application needs to send a push notification an android device. The messaging service will receive an abstract push notification from the Azure Notification Hub, where it will go on to modify and then send that push notification to the user’s mobile phone in a HTTP format.

Flow of Data

* **Dataflow 1:**
  + The application sends HTTPS formatted function request to the Azure Cloud. The Azure Cloud returns any response to those requests in the same HTTPS format.
* **Dataflow 2:**
  + The Azure Cloud sends any received HTTPS function requests to the Azure Restful API in a HTTP format. The Restful API returns any responses to the could in the same HTTP format.
* **Dataflow 3:**
  + The Azure Application Service sends any data the application must store through the Azure Encryption Service in a JSON format, to the Cloud Database. The Cloud Database returns the results of any query request in the same JSON format through the Azure Encryption Service, to the Restful API.
* **Dataflow 4:**
  + The Restful API sends JavaScript formatted code to the Azure Notification Hub, to be used to create a new notification.
* **Dataflow 5:**
  + The Azure Notification Hub sends an abstract notification to the Google Cloud Messaging Service in a HTTP format.
* **Dataflow 6:**
  + The Google Cloud Messaging Service sends the newly modified push notification to the user’s applications View on the user’s mobile device in an HTTP format.