National College of Ireland

# Group 1- Requirements Specification




18th February 2024

2023/2024

# Contents

# 1.0 Introduction

## 1.1. Purpose

The purpose of this document is to set out the requirements for the development of the application called WhatsOn. Included in this document is the purpose outline, project scope explanation, the definitions, acronyms, and abbreviations used in the document are explained. Then the user requirements are defined. After that it defines the functional requirements through use case requirements and then the non-functional requirements. Then it showcases the GUI. The system architecture is shown through the context view and components view. Finally, the document describes how the system may evolve.

The intended customers are adults aged 21-40, living in Ireland with an interest in events and socialising. Foreign nationals and tourists will be a significant part of this segment.

WhatsOn will encourage users to open to the opportunities for connection all around them. We have identified barriers such as decision fatigue, planning and the anxiety associated with trying new things or the FOMO (Fear of Missing Out) associated with missing things. As such we have consciously chosen to limit events to ones that are current and remove them afterwards. We want this app to be a welcoming user experience instead of another source of digital anxiety.

A user need only open the application, allow location permissions and, at a glance, they will be able to see all the current events going on around them without the need to sign up or search. They can click through to the source site if they are interested. If a user can't decide on what to pick, the random button will present an event for them. Pins on the map will indicate the type of event it is and the distance from the user. If a user wants to see only a certain type of event, they can click on the icon event to see only events that meet that description. Users will have the option to sign up for an account if they would like to follow recurring events or specific organisers. Those events will appear with a star above them.

## 1.2. Project Scope

The scope of the project is to develop a web app named “WhatsOn” which functions on desktop and mobile and allows users to view events in their area on a live map with informative data.
The system shall have a live map, registration, following, legend, random event suggestion, and the ability to contact the developers.
The functions that are outside the scope of the project are anything monetary, there will be no direct access for event organisers to change their events in the app, we will only pull the information from the API. The app will not analyse the users’ frequented events automatically. A user will have to select the favourites themselves.

The app will be coded on Microsoft Visual Studio Code. The languages are CSS, JavaScript, HTML(EGS), NodeJS.

## 1.3. Definitions, Acronyms, and Abbreviations

**API**
Application programming interface. A set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

**Legend**
A guide as to what symbols on a map mean.

**CSS**
Cascading style sheets. A language used to change the website's appearance.

**NodeJS**
JavaScript runtime environment.

**EGS**
Embedded JavaScript templates. A language system used in making websites.

**End User**
A customer or anyone who uses the final product. They are not involved in development.

**GUI**
Graphical user interface. What the user sees when using the application.

**UI**
User Interface

**USP**
Unique Selling Proposition

## 2.0 User Requirements Definition

Clients are looking for:

**Discoverability**: The client would like to be able to discover new events that are on in their area, they would like to be able to discover events that are both like what they know they enjoy and something completely new.

**Reliability**: The client would like the map to be reliable when it comes to navigating towards their area. Also, when pressing on events they would like the pop-up to be consistent.

**Ease of Use:** The client would like the app to be easy to use, with symbols being intuitive and those which are not, they are described somewhere accessible.

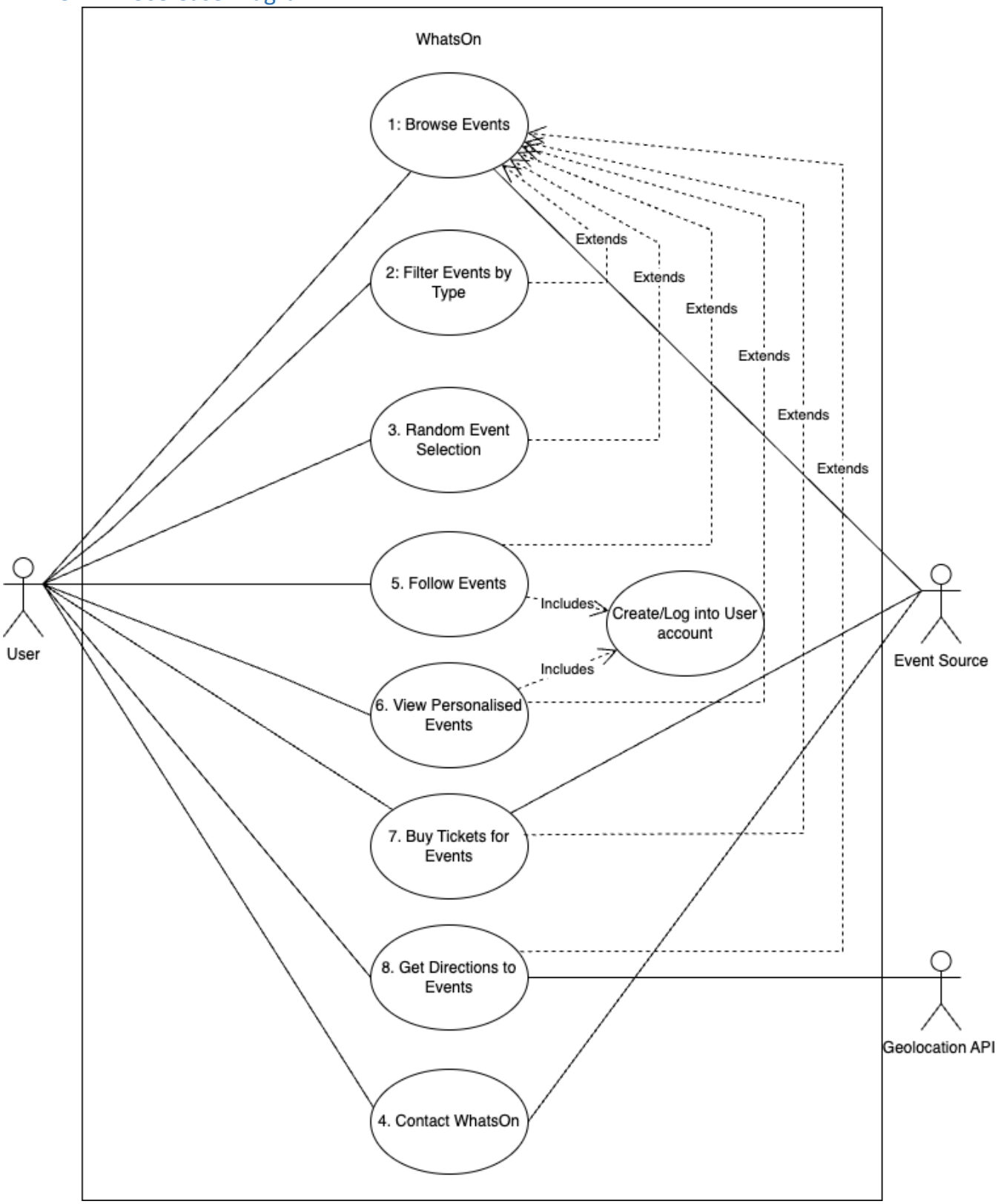
**Information**: The client would like to know more about events before going to them so they do not arrive at something completely in the dark. They want to know who runs the event, where it is, and how long it will last.

# 3.0 Requirements Specification

## 3.1 Functional Requirements

1. **Browse Events:** Enable users to browse events on a map-based user interface.
2. **Filter Events By Type:** Allow users to filter events by type.
3. **Random Event Selection:** Provide users with the ability to randomly select events.
4. **Contact WhatsOn:** Allow users to contact WhatsOn for inquiries or support.
5. **Follow Events:** Enable users to follow recurring events.
6. **View Personalised Events:** Implement personalized event viewing based on user preferences.
7. **Buy Tickets for Events:** Facilitate ticket purchasing for events.
8. **Get Directions to Events:** Provide users with directions to events.

### 3.1.1 Use Case Diagram



### 3.1.2 Requirement 1: Browse Events

#### Description & Priority

Browsing events on a map-based UI is crucial for the app's USP and differentiation from competitors. It forms the core functionality of the app and is essential for user engagement.

**Priority**: Highest.

#### Use Case

##### Scope

The scope of this use case is for a user to browse events within the WhatsOn app.

##### Description

This use case describes a user browsing events within the WhatsOn app.

##### Flow Description

**Precondition**

1. The user has location services enabled on their device.
2. The user has granted location permission to the WhatsOn App.

**Activation**

This use case starts when the user launches the WhatsOn app.

**Main flow**

1. The user launches the WhatsOn app on their device.
2. The system retrieves the user's location using GPS/IP geolocation.
3. The system displays a map user interface centred on the user's current location.
4. The system retrieves current events from external API relative to the user's location.
5. The system represents events as pins with an icon type.
6. The user can interact with the map by dragging across the map or zooming in/out with a cursor/finger.
7. The system displays events dynamically based on visible area.
8. The user clicks/taps the event icon.
9. The system displays a pop-up window with event information such as:
   - Event name
   - Event type
   - Event venue/location
   - Event date/time
   - Event description
   - Event host
   - Event accessibility information
   - Event price
10. The user can exit pop-up window by clicking/tapping outside the window.
11. The system closes pop-up window.

**Alternate flow**

A1: User's location cannot be determined.

1. The system prompts user to enable location services and grant permission to the WhatsOn app.
2. The user enables location services and grants permission to the WhatsOn app.
3. The use case continues at position 2 of the main flow.

A2: User's location cannot be determined persists.

1. The system displays a default view centred around a predefined location.
2. The use case continues at position 3 of the main flow.

**Exceptional flow**

E1: Error retrieving event data

1. The system notifies the user and prompts them to retry fetching the data.
2. The user clicks/taps the retry button.
3. The use case continues at position 4 of the main flow.

E2: Error retrieving event data persists.

1. The system notifies the user that data is currently unavailable.
2. The system displays a default map user interface.
3. Once the issue is resolved, the use case continues at position 4 of the main flow.

**Termination**

The user exits the "Browse Events" use case by navigating to a different screen or closing the app.

**Post condition**

The system defaults to position 2 of the main flow.

### 3.1.3 Requirement 2: Filter Events by Type

#### Description & Priority

Filtering events based on predefined types enhances user experience by allowing users to narrow down their preferences without the need to search or type. This requirement further differentiates WhatsOn by creating an uncluttered and seamless user experience.

**Priority:** High.

#### Use Case

##### Scope

The scope of this use case is for a user to filter events to display only a defined type within the WhatsOn app.

##### Description

This use case describes a user filtering events by type within the WhatsOn app.

##### Flow Description

**Precondition**

1. The user has clicked/tapped an event icon.
2. The system displays a pop-up window with event information.
3. Or the user has clicked/tapped the “Legend” Button.
4. The system displays a pop-up window with icon information.

**Activation**

This use case starts when the user clicks/taps the “Type” button within the event information pop-up window.

**Main flow**

1. The user clicks/taps the “Type” button within the event information pop-up window or the “Legend” pop-up window.
2. The system exits the pop-up window.
3. The system displays events that match the selected type within a defined range of the user.
4. The user can exit the filter by clicking/tapping on an event to open the event information pop-up window or the “Legend” button to open the “Legend” pop-up window.
5. The user clicks/taps the “Type” button within the event information/legend pop-up window.
6. The system terminates the filter and displays all events within a defined range of the user.

**Alternate flow**

A1: No events match the selected type from the “Legend” pop-up window.

1. The system notifies the user that no events match the selected type.
2. The system displays all events within a defined range of the user.

**Exceptional flow**

E1: Error retrieving event type data.

1. The system notifies the user and prompts them to retry fetching the data.
2. The user clicks/taps the retry button.
3. The use case continues at position 4 of the main flow.

E2: Error retrieving event data type persists.

1. The system notifies the user that event data is currently unavailable.
2. The use case continues at position 6 of the main flow.
3. Once the issue is resolved, the use case continues at position 3 of the main flow.

**Termination**

The user clicks/taps the “Type” button within the event information/legend pop-up window. The system terminates the filter and displays all events within a defined range of the user.

**Post condition**

The system defaults to the “Browsing Events” requirement interface, displaying all events within a defined range of the user.

### 3.1.4 Requirement 3: Random Event Selection

#### Description & Priority

Random event selection is crucial for creating an addictive novelty factor while addressing barriers to spontaneous socialising like decision fatigue and avoidance.

**Priority:** High.

#### Use Case

##### Scope

The scope of this use case is for a user to randomly select an event within the WhatsOn app.

##### Description

This use case describes a user randomly selecting an event within the WhatsOn app.

#### Flow Description

**Precondition**

1. The user has location services enabled on their device.
2. The user has granted location permission to the WhatsOn App.

**Activation**

This use case starts when the user clicks/taps the "Random" button within the WhatsOn app.

**Main flow**

1. The user clicks/taps the "Random" button within the WhatsOn app.
2. The system randomly selects from the available events within a defined range of the user.
3. The system displays a pop-up window with event information such as:

- Event name
- Event type
- Event venue/location
- Event date/time
- Event description
- Event host
- Event accessibility information
- Event price

4. The user can click/tap the "Random" button again.
5. The system randomly selects another available event within a defined range of the user.
6. The user can exit the pop-up window by clicking/tapping outside the window.
7. The system closes the pop-up window.
8. The system displays all events within a defined range of the user.

**Alternate flow**

A1: No events available for random selection.

1. The system notifies the user.

**Exceptional flow**

E1: Error retrieving event-type data.

1. The system notifies the user and prompts them to retry fetching the data.
2. The user clicks/taps the retry button.
3. The use case continues at position 2 of the main flow.

E2: Error retrieving event data type persists.

1. The system notifies the user that event data is currently unavailable.
2. The use case continues at position 8 of the main flow.
3. Once the issue is resolved, the use case continues at position 2 of the main flow.

**Termination**

The user exits the “Random Event” use case by clicking/tapping outside the event pop-up window.

**Post condition**

The system defaults to the “Browsing Events” requirement interface, displaying all events within a defined range of the user.

### 3.1.5 Requirement 4: Contact WhatsOn

#### Description & Priority

Enabling users to contact WhatsOn directly within the app is essential for gathering user feedback, addressing concerns promptly, and fostering partnerships with events for business opportunities.

**Priority:** High.

#### Use Case

##### Scope

The scope of this use case is for a user/event to contact WhatsOn for inquiries or support within the WhatsOn app.

##### Description

This use case describes how a user or event interacts with the contact feature within the WhatsOn app to send inquiries to WhatsOn. We will refer to both actors as user for simplicity.

##### Flow Description

**Precondition**

1. The user clicks/taps the "Contact" button on the main UI.
2. The system displays a pop-up window with a form for entering an email address, a text field for the message, and a "Send" button.

**Activation**

This use case starts when the user/event clicks/taps the "Contact" button on the main UI.

**Main flow**

1. The user clicks/taps the "Contact" button on the main UI.
2. The system displays a pop-up window with a form for entering an email address, a text field for the message, and a "Send" button.
3. The user fills out the email address input form and the message text field.
4. The user clicks/taps the "Send" button.
5. The system sends the inquiry/message to the WhatsOn support team.
6. The system displays a confirmation message to the user indicating that the message has been sent successfully.

**Alternate flow**

A1: User decides not to send message.

1. The user clicks/taps the "Contact" button on the main UI.
2. The system displays a pop-up window with a form for entering an email address, a text field for the message, and a "Send" button.

3. The user closes the contact pop-up window use case by clicking/tapping outside of it to return to the default UI.

**Exceptional flow**

E1: Error sending message.

1. The system notifies the user and prompts them to retry sending the message.
2. The user clicks/taps the “Send” button.
3. The use case continues at position 5 of the main flow.

E2: Error retrieving event data persists.

1. The system notifies the user that the contact feature is unavailable.
2. The system displays a default map user interface.
3. Once the issue is resolved, the use case continues at position 5 of the main flow.

**Termination**

The user exits the "Contact WhatsOn" use case by clicking/tapping outside the contact pop-up window to return to the default UI.

**Post condition**

The system defaults to position 4 of the main flow.

### 3.1.6 Requirement 5: Follow Events

#### Description & Priority

While useful for users interested in specific recurring events, following events is not essential for all users and can be considered as supplementary functionality.

**Priority:** Medium.

#### Use Case

##### Scope

The scope of this use case is for a user to follow recurring events within the WhatsOn app.

##### Description

This use case describes how a user interacts with the "Follow" feature in the event pop-up window to follow recurring events. If the user is logged in to their account the event will be added to their “Following” list. If the user is not logged/ registered they will be prompted to log in/create an account.

##### Flow Description

**Precondition**

1. The user may or may not be logged into their WhatsOn account.
2. The user has clicked/tapped an event icon.
3. The system displays a pop-up window with event information.

**Activation**

This use case starts when a user clicks/taps the "Follow" button within the event information pop-up window for a recurring event.

**Main flow**

1. If the user is logged in:
   - The system adds the event to the user’s followed events list.
   - The user can view/edit their following list by accessing by clicking/tapping the “Following” button.
   - The system displays a heart icon above the event when it has been followed.
   - The user can click/tap a heart icon to see only followed events.
   - The system will display only followed events on the map.
   - The user can unfollow an event by clicking/tapping a previously clicked event.
   - The system will remove that event from the following list.
2. If the user is not logged in:
   - The system prompts the user to log in/create an account to follow the event.
   - The user clicks/taps on the prompt.
   - The system adds the event to the user’s followed events list.

**Alternate flow**

A1: Account creation Process.

1. The user selects the option to create an account.
2. The system prompts the user to enter an email address and password.
3. The user enters an email address and password.

4. The system validates the entered information.
5. If the email address is already associated with an existing account, the system prompts the user to log in instead.
6. If the email address is not associated with an existing account, the system creates a new account for the user.
7. The use case continues at position 1 of the main flow.

A2: User forgets password for account.
1. The user clicks/taps the "Forgot Password" button.
2. The system prompts the user to enter their email address.
3. The user enters their email address.
4. If the email address is already associated with an existing account:
    - The system sends a password reset link to the user's email address.
    - The user checks their email for the password reset link.
    - The user clicks/taps on the password reset link.
    - The system prompts the user to enter a new password.
    - The user enters a new password.
    - The system updates the user's password.
    - The use case continues at position 2 of the main flow.
5. If the email address is not associated with an existing account:
    - The system prompts the user to create an account instead.
    - The use case continues at position 1 of the A1 alternative flow.

A3: No events are recurring.
1. The system will not display the “Follow” event button within the event information pop-up window.

**Exceptional flow**

E1: Error processing the user's request to log in or create an account.
1. The system notifies the user and prompts them to retry log-in/account creation.
2. The use case continues at position 2 of the alternate flow.

E2: Error processing the user's request to follow the event.
1. The system notifies the user and prompts them to retry following the event.
2. The user clicks/taps the retry button.
3. The use case continues at position 1 of the main flow.

E2: Error processing the user's request to follow the event persists.
1. The system notifies the user that the follow event feature is currently unavailable.
2. The system defaults to event info pop-up window interface.
3. Once the issue is resolved, the use case continues at position 1 of the main flow.

**Termination**

The user exits the “Follow Events” use case by tapping either clicking/tapping the “Follow” button to unfollow an event or by removing the event from the “Following” pop-up window.

**Post condition**
The system defaults to the event pop-up window, displaying the event information. If the user has just followed the event, the system displays a heart icon above the event to indicate it has been followed.

### 3.1.7 Requirement 6: View Personalized Events

#### Description & Priority

While personalized events contribute to user satisfaction and engagement, they are not essential for all users and can be considered as additional, optional functionality.

**Priority:** Medium.

#### Use Case

##### Scope

The scope of this use case is for a user to view events based on their personalised event profile within the WhatsOn app.

##### Description

This use case describes how a user creates their personalised event profile and views personalised events within the WhatsOn app.

##### Flow Description

**Precondition**

1. The user does not have a user account on WhatsOn.

**Activation**

This use case starts when a user clicks/taps the “Following” button from the main UI without an account.

**Main flow**

1. The user clicks/taps the “Following” button from the main UI.
2. The system prompts the user to “Create account” by entering an email address and password.
3. The user enters an email address and password.
4. The system validates the entered information.
5. If the email address is already associated with an existing account, the system prompts the user to log in instead.
6. If the email address is not associated with an existing account, the system creates a new account for the user.
7. The system prompts the user to create their personalised event profile by selecting their preferred event types.
8. The user selects their preferred events.
9. The system saves the user’s preferences in their user account.
10. The system retrieves the user’s personalised event data.
11. The system displays a star above personalised events on the map.
12. The user can tap/click a star to see only personalised events on the map.

**Alternate flow**

A1: The user has an account and is not logged in.

1. The user clicks/taps the “Following” button from the main UI.
2. The system prompts the user to enter an email address and password.
3. The user enters an email address and password.

4. The system validates the log in by comparing email address and password to existing accounts
5. The use case continues at position 10 of main flow.

A2: The user forgets the password to the user account.
1. The system has a “Forgot my password” button.
2. The user clicks/taps the button.
3. The system sends a code to the user’s email address to reset their password.
4. The system prompts the user to enter the code.
5. The user enters the code.
6. The user is prompted to create a new password.
7. The use case continues at position 10 of the main flow.

**Exceptional flow**

E1: Error creating an account.
1. The system notifies the user and prompts them to retry the account creation process.
2. The user retries the account creation process with a valid email address and password.
3. The use case continues at position 4 of the main flow.

E2: Error creating account persists
1. The system notifies the user that the create account feature is currently unavailable.
2. The system defaults to the default UI.
3. Once the issue is resolved, the use case continues at position 2 of the main flow.

E3: Error retrieving personalized event data.
1. The system notifies the user that personalized event data is currently unavailable.
2. The system defaults to the default UI displaying all available events.
3. Once the issue is resolved the use case continues at position 10 of the main flow.

**Termination**

The user exits the “View Personalized Events” use case by logging out of their account.

**Post condition**

The system defaults to the main UI interface, displaying all available events. If there are personalized events they will appear with a star icon above them.

### 3.1.8 Requirement 7: Buy Tickets for Events

#### Description & Priority

WhatsOn does not feature ticketing or payment processing. However facilitating ticket purchases contributes to user convenience and satisfaction, making it a valuable feature without being as essential as core functionalities.

**Priority:** Medium.

#### Use Case

##### Scope

The scope of this use case is for a user to purchase tickets for events displayed on the WhatsOn app.

##### Description

This use case describes a user viewing an event within the WhatsOn app and being redirected to the event source site to buy tickets.

##### Flow Description

**Precondition**

1. The user clicks/taps an event icon from the main UI.
2. The system displays a pop-up window with event information and a source link to the event source site

**Activation**

This use case starts when the user clicks the “Source” button within the event information pop-up window.

**Main flow**

1. The user clicks/taps the “Source” button within an event information pop-up window.
2. The system redirects the user to the ticket purchase page on the external event source website.
3. The user can navigate the external website to view event details and purchase tickets for the event.

**Alternate flow**

A1: User does not buy tickets.

1. The user clicks/taps the “Source” button within an event information pop-up window.
2. The system redirects the user to the ticket purchase page on the external event source website.
3. The user can navigate the external website to view event details and decide not to purchase tickets.

**Exceptional flow**

E1: Error redirecting to event source site.

1. The system notifies the user and prompts them to retry fetching the data.
2. The user clicks/taps the retry button.
3. The use case continues at position 2 of the main flow.

E2: Error redirecting to event source site persists.

1. The system notifies the user that the external website is currently unavailable.
2. The system defaults to the event info pop-up window.

E3: Error retrieving external website address with event info.

1. The system does not display “Source” button.

**Termination**

The user exits the "Buy Tickets for Events" use case by closing the external website window and/or returning to the WhatsOn app.

**Post condition**

The system defaults to the event pop-up window, displaying the event information. If the user clicks/taps the "Source" button, the system returns to the event pop-up window after closing the external website browser window.

### 3.1.9 Requirement 8: Get Directions To Events

#### Description & Priority

WhatsOn does not feature native route planning. Linking to a reliable and accurate route planning site such as Google Maps enhances convenience and facilitates event attendance.

**Priority:** Medium.

#### Use Case

##### Scope

The scope of this use case is for a user to obtain directions to events displayed on the WhatsOn app.

##### Description

This use case describes a user viewing an event within the WhatsOn app and being redirected to an event source site a reliable mapping service (Google, Apple Maps) for accurate route planning in different modes of transport such as Walking, Driving, Cycling, Public Transport, etc.

##### Flow Description

**Precondition**

1. The user clicks/taps an event icon from the main UI.
2. The system displays a pop-up window with event information and a source link to an external mapping service site.
3. External Geolocation API tracks the event accurately.

**Activation**

This use case starts when the user clicks the "Directions" button within the event information pop-up window.

**Main flow**

1. The user clicks/taps the "Directions" button within an event information pop-up window.
2. The system retrieves the coordinates of the event venue from the event data.
3. The system launches the selected mapping service (Google Maps, Apple Maps) on the user's device.
4. The system displays the route and directions from the user's current location to the event venue, allowing the user to choose different modes of transport (Walking, Driving, Cycling, Public Transport, etc).

**Alternate flow**

A1: No directions available for event.

1. The system does not display “Directions” button.

**Exceptional flow**

E1: Error redirecting to mapping service.

1. The system notifies the user and prompts them to retry fetching the data.
2. The user clicks/taps the retry button.
3. The use case continues at position 2 of the main flow.

E2: Error redirecting to mapping service persists.

1. The system notifies the user that the external website is currently unavailable.
2. The system defaults to the event info pop-up window.

E3: Error retrieving event venue coordinates.

4. The system does not display “Directions” button.

**Termination**

The user exits the "Get Directions to Events" use case by closing the maps application or returning to the WhatsOn app.

**Post condition**

The system defaults to the event pop-up window, displaying the event information. If the user clicks/taps the "Directions" button, the system returns to the event pop-up window after closing the maps application.

## 3.2 Non-Functional Requirements

### 3.2.1 Performance/Response Time Requirement

**Requirement:** The WhatsOn app must have fast response times and performance to ensure smooth browsing, filtering, and interaction with events on the map interface.

**Rationale:** Fast response times are essential for providing a simple and seamless user experience to remove the friction from socialising and event attendance.

### 3.2.2 Availability Requirement

**Requirement:** The WhatsOn app must have high availability, with minimal downtime and reliable access to event data and features.

**Rationale:** Our USP is based on spontaneity and availability is crucial for ensuring users can access event information and features whenever they need it.

### 3.2.3 Recover Requirement

**Requirement:** The WhatsOn app can have robust mechanisms for data recovery and system restoration in the event of failures or disruptions.

**Rationale:** Recoverability ensures that users can recover their data and resume using the app seamlessly after unexpected events such as crashes or data loss. This is not crucial for Whatson However as we do not handle sensitive data.

### 3.2.4 Robustness Requirement

**Requirement:** The WhatsOn app must be robust and resilient to handle unexpected errors or disruptions without compromising user experience.

**Rationale:** Robustness is essential for maintaining user trust and preventing data loss or corruption in the event of system failures.

### 3.2.5 Security Requirement

**Requirement:** The WhatsOn app should ensure the security and confidentiality of user data.

**Rationale:** Security is paramount to protect user privacy, prevent unauthorized access, and maintain trust in the app. However again WhatsOn doe not handling sensitive data so this is not a crucial requirement.

### 3.2.6 Reliability Requirement

**Requirement:** The WhatsOn app must be reliable, with minimal errors, crashes, or disruptions during user interactions.

**Rationale:** Reliability is essential for providing a consistent and predictable user experience, fostering user trust and satisfaction.

### 3.2.7 Maintainability Requirement

**Requirement:** The WhatsOn app must be maintainable, with well-structured code, clear documentation, and easy-to-use development tools.

**Rationale:** Maintainability is crucial for enabling efficient ongoing development, bug fixes, and enhancements to the app over time.

### 3.2.8 Portability Requirement

**Requirement:** The WhatsOn app must have robust mechanisms for data recovery and system restoration in the event of failures or disruptions.

**Rationale:** Recoverability ensures that users can recover their data and resume using the app seamlessly after unexpected events such as crashes or data loss.

### 3.2.9 Extendibility Requirement

**Requirement:** The WhatsOn app must be designed to be easily extendable, allowing for future feature additions and integrations.

**Rationale:** Extendibility enables the app to evolve and adapt to changing user needs, market trends, and technological advancements.

### 3.2.10 Reusability Requirement

**Requirement:** The WhatsOn app must promote code reuse, leveraging existing components, libraries, and frameworks wherever possible.

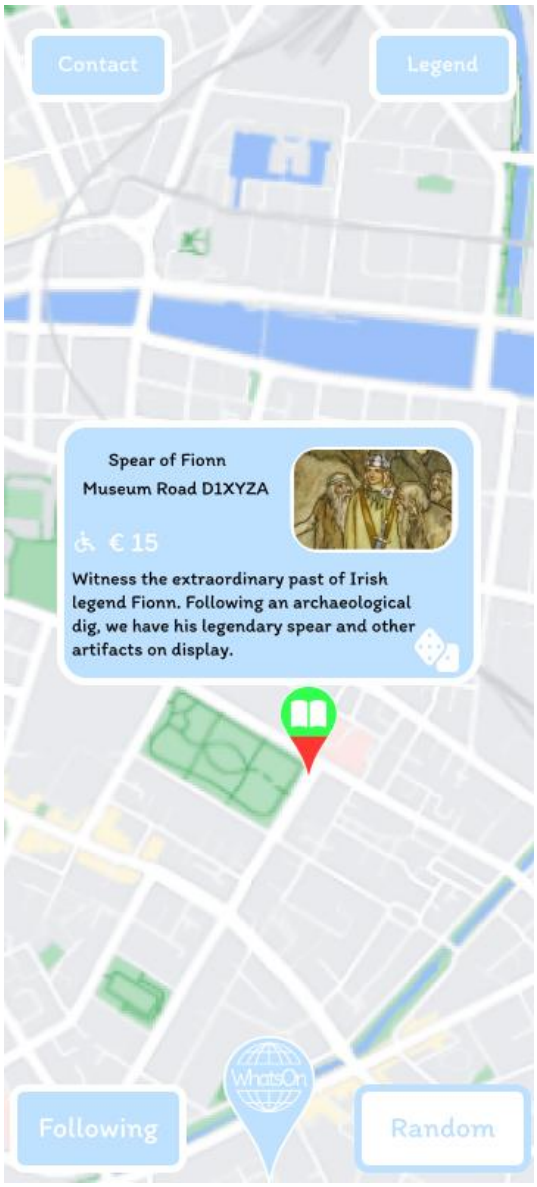
**Rationale:** Reusability reduces development time and effort, improves code maintainability, and ensures consistency across the app.

### 3.2.11 Resource Utilization Requirement

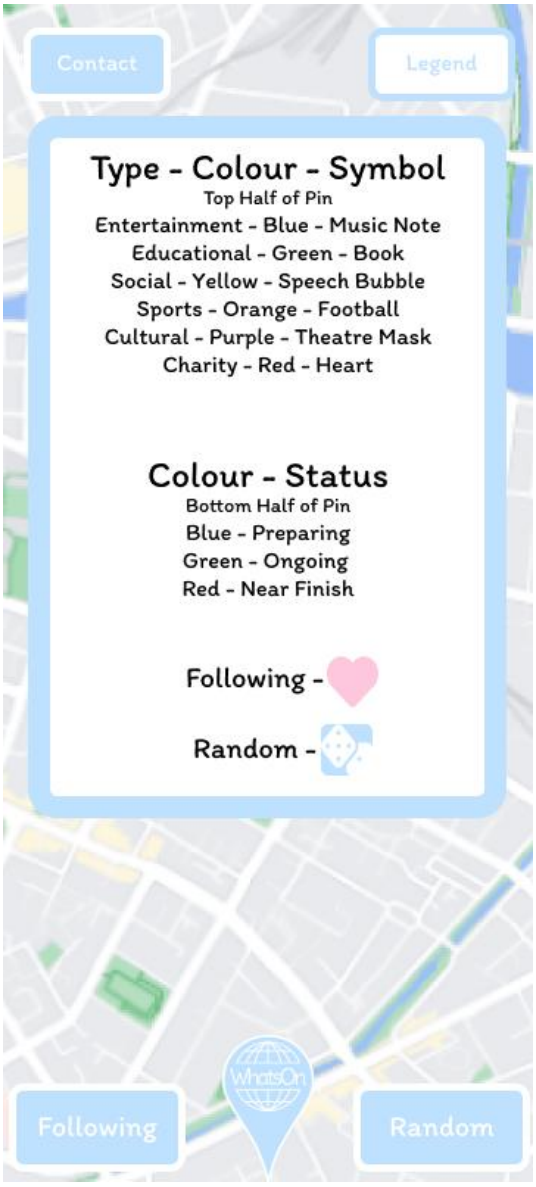
**Requirement:** The WhatsOn app must optimize resource utilization, including memory, CPU, and network bandwidth, to ensure efficient operation on various devices and network conditions.

**Rationale:** Efficient resource utilization improves app performance, battery life, and user experience, particularly on resource-constrained devices.

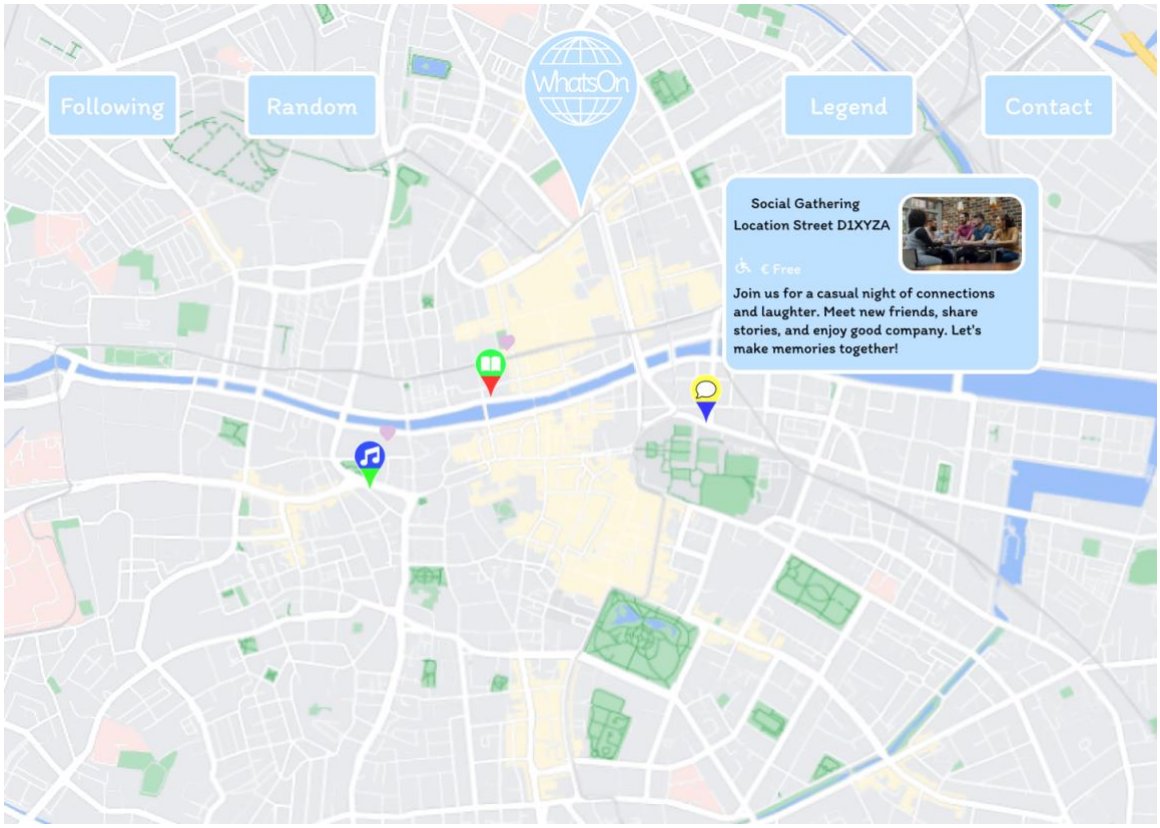
## 4.0 GUI



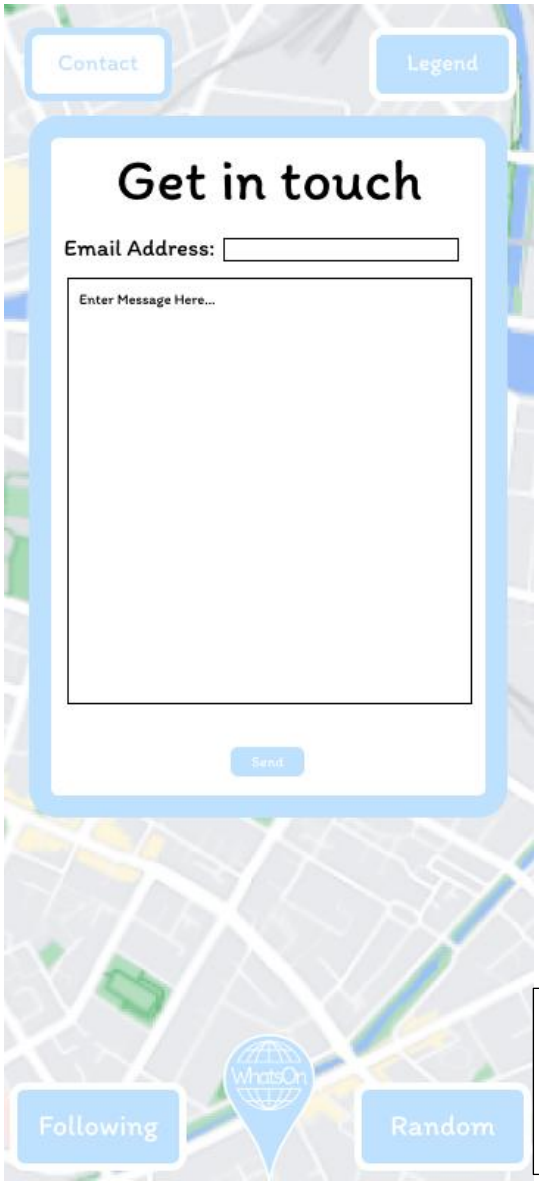

A random suggestion from the app, with a colour coded pin.




The legend of the application. The pin colouring system is explained here.



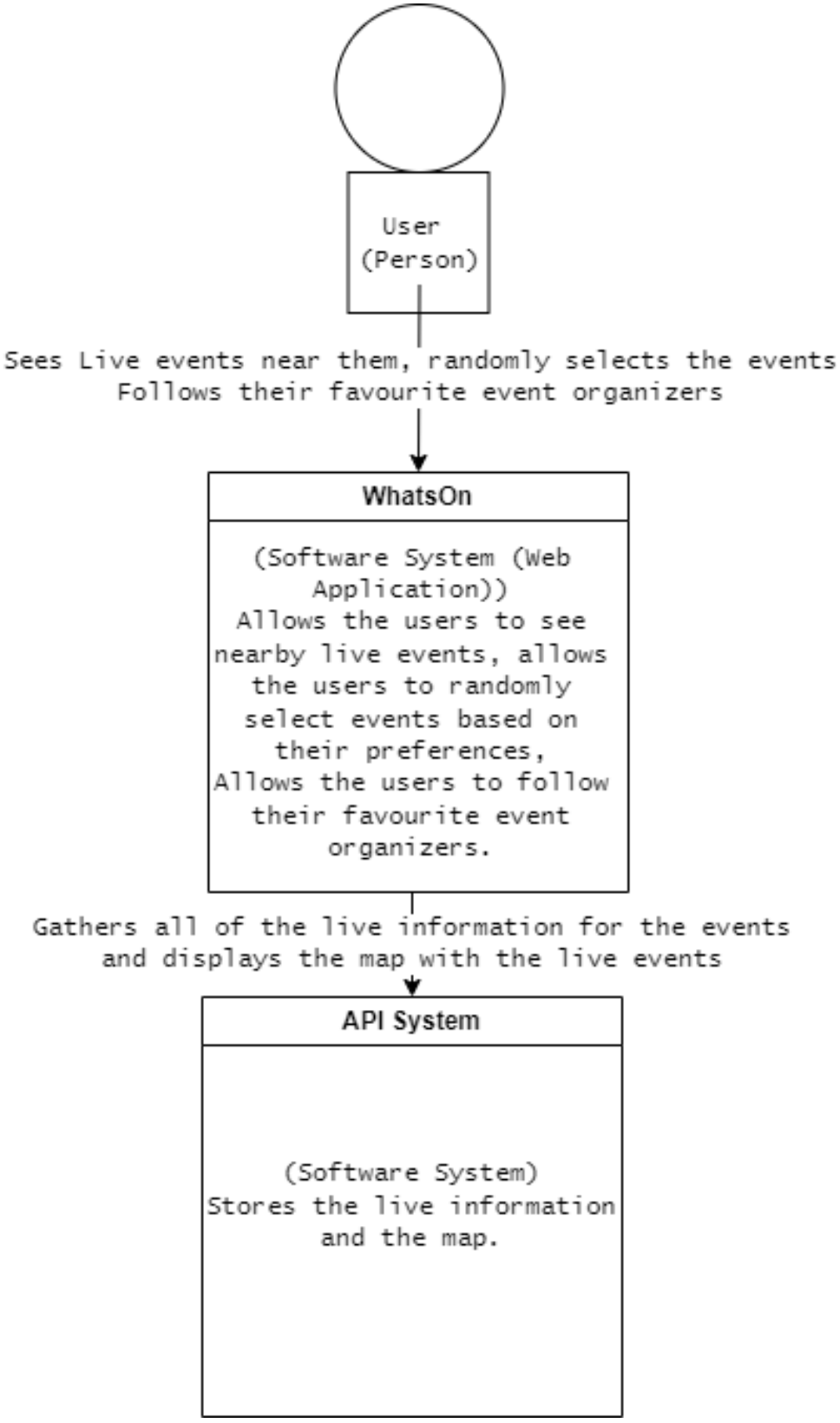

The standard view of the application with multiple icons of different types, the icon at the top depicts the type as well as the upper colour and the lower colour indicates the time status of the event. The little hearts next to the pins indicate you are following the event organiser, when you press the following button, it hides all the events you are not following.



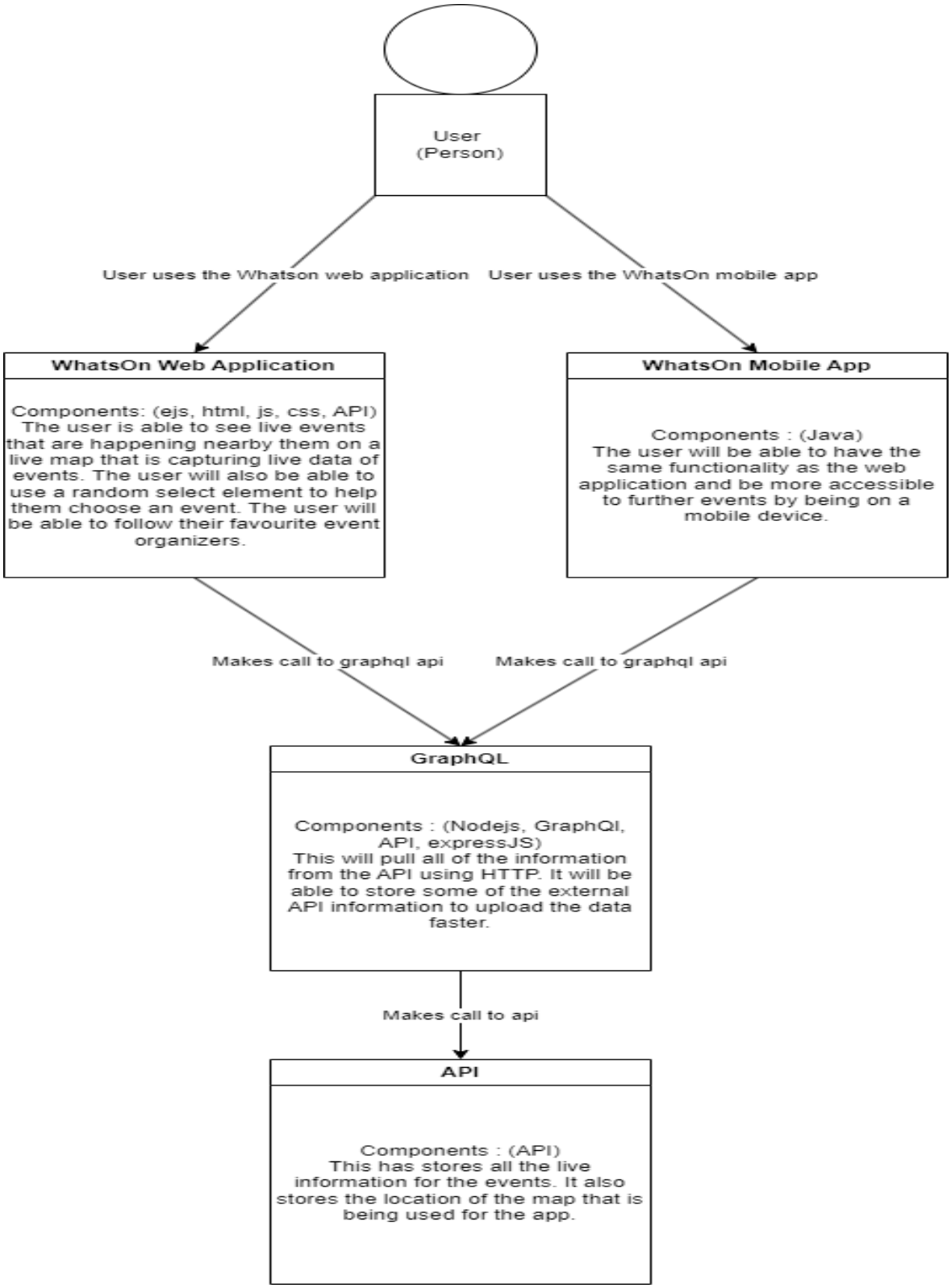

How the end user will contact us if they have difficulties or praise.

# 5.0 System Architecture

## 5.1 Context View



## 5.2 Components View



## 6.0 System Evolution

There are many ways on how this system could evolve, here is a list of a few:

- Adding databases to store user's details
- Adding algorithms
- Creating a native mobile app
- Making a coupon system
- Adding more APIs to enhance the functionality of the app.

Adding Database:
This would evolve our system greatly because we would be able to permanently store our user's detail to the system meaning they will not miss out on their followed events at all times.
This would help us store algorithms made for the users to show them events more relevant to their interests. This would make the app more interactive with the customers using the application.

Adding Algorithms:
The algorithms would evolve our system by gathering the data by the user to make events like their interests from other events to appear more regularly. This links in with the database that it would store all of its gathered data on the user. The data collected on this user will be purely based on what events they attend to on this application. As a protection of privacy for the user.
(Desai, 2023)

Native mobile App:
We intend in fully creating a mobile app for the website, but it would not be up to its full potential because the mobile app won't be made natively. When we do make a native app, it would evolve our app massively being more accurate and better optimization.
(Shevchenko, 2018)

Coupon feature:
We intend to add this to evolve our app to encourage users to use our application more to win awards of coupons. We could partner with local companies who would provide us coupons so more customers can attend their events/venues.

Adding more API’s:
This will evolve our app massively depending on the API’s we decide to use. In the case of adding a traveling system showing how to get to the venue from the app with the possibility of what transport nearby the user can take them there. These API’s can extend the amount of the events that are available on our application providing a wider range of events.