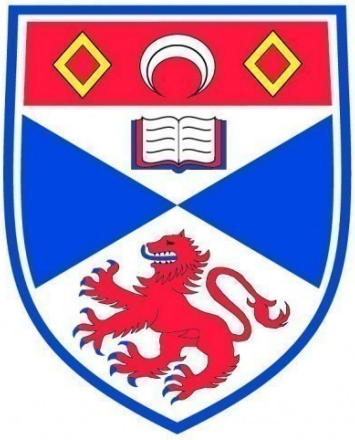

University
of
St Andrews

StAR-Wiki: An Augmented Reality Wiki

Samuel Koch

January 18, 2014

StAR-Wiki: An Augmented Reality Wiki

Samuel Koch
Student
sk524@st-andrews.ac.uk

ABSTRACT

I present a novel Augmented Reality (AR) wiki system called StAR-Wiki. Keeping up-to-date with the ever changing world of information is extremely difficult, especially when users desire different subsets of all the information available. My StAR-Wiki system shows information from user-selected subscription layers and provides users with a customisable experience. Giving end-users the functionality to add and edit the information directly in the app creates an extensible AR system. The system can then be adapted to any environment to build a wealth of information gathered from multiple users, thus creating an AR wiki. Validation tests carried out show that all features of the system work as expected.

DECLARATION

I declare that the material submitted for assessment is my/our own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is 10,841 words long, including project specification and plan. In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

INTRODUCTION

Augmented Reality (AR) aims to create the sensation that makes virtual objects appear present in the real world. This is done by combining Virtual Reality (VR) elements with the real world, in real time. AR in its simplest form can be overlaying a 2D image onto a video stream, but it is also possible to render 3D objects that can appear to belong to a scene containing 3D objects.

Though VR and AR are related, both lie towards opposite ends of the Reality-Virtuality (RV) Continuum [19], a continuum going from the real environment to a virtual environment. VR can be fictional, consisting of virtual objects, and therefore is not always restricted by laws of physics. In contrast, AR provides augmentation of real world objects, being observed directly in person, through some kind of a window or a video display.

An example of an AR environment can be seen in Figure 1, where a user is viewing information about a coffee machine in a work office whilst using the StAR-Wiki system. They are in a real world environment, experiencing a real world object through a mobile device that is streaming a live camera feed. Information about the coffee machine, such as its working state, is overlaid onto the screen that the user can interact with to find out more information or update themselves.

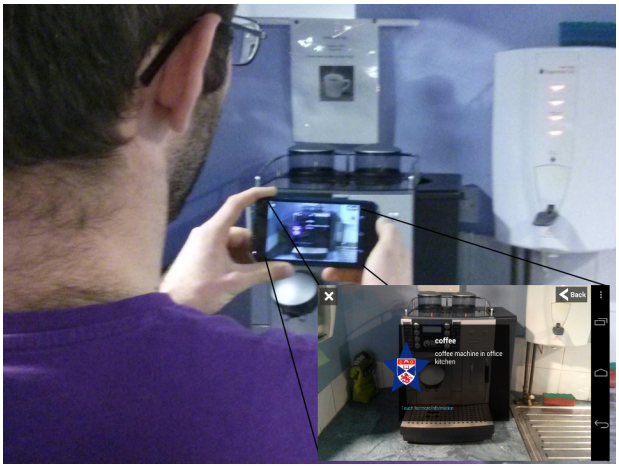

Figure 1. The StAR-Wiki system being used in an office to display information about a coffee machine recognised by the system.

Augmented Reality

There are three basic strategies [17] when approaching AR. Firstly by augmenting the user, making them wear or carry a device normally on their head or in their hands, so they can obtain information about physical objects. This began with the earliest head-mounted display by Sutherland in 1968 [26].

At Harvard University in 1968, Ivan Sutherland created a working prototype of what is generally considered to be the first VR and AR system. The user wore a head-mounted display (HMD), and although the system only used simple wireframe graphics, the project became the genesis of AR.

The second strategy involves augmenting physical objects, by embedding input or output devices on or within the objects. In the 1970s, Papert [24] created a floor turtle, a small robot, that could be controlled by a programming language called Logo. LEGO/Logo was built from Logo, allowing Logo to control constructions made from LEGO bricks, motors and gears. The system takes the computer code and augments it into the physical robot object.

The last strategy augments users or objects in their surrounding environment. Independent devices collect information from the environment and display it onto objects and track users interaction with them. This project will focus on this

AR strategy, although there are many applications of all three strategies.

Doctors may use AR systems to train for certain surgeries, or even use it during surgeries to detect features that current technology, such as MRI or CT scans cannot. Assembly, maintenance and repair of complex machinery is another main application of AR. Instructions superimposed upon equipment, rather than in manuals with text and pictures, can show step-by-step the tasks required.

Other applications include robot path planning, using a local virtual robot to test and determine the best path which is sent to the real robot to execute. AR is also used in entertainment, where actors stand in front of a blue screen and a computer-controlled motion camera records the scene. A 3D virtual background can then be applied to the scene.

Military aircraft and helicopters use Head-Up Displays and Helmet-Mounted Sights to superimpose vector graphics on the pilots view of the real world [4]. This provides basic navigation and flight information and sometimes registers targets in the environment, giving pilots a way to aim the aircrafts weapons.

Once computers increased in power and decreased in size, mobile devices became rapidly feasible, providing flexibility for new kinds of applications exploiting the users surrounding context [11]. AR can now be used away from conditioned environments in research laboratories and special-purpose work areas. Combining global tracking, wireless communication and location-based services provides the base for a mobile AR experience.

Combining real and virtual objects enhances a users perception of and interaction with the real world. The objects displayed may provide information to users that they may not be able to discover using their own senses and can even help them perform real-world tasks. There are many real-world applications that greatly benefit from AR.

Some modern AR applications include wearable glasses, such as Google Glass [9], that provide users with navigation, a camera and voice control. Mobile AR applications, such as Layar and Wikitude, provide users with information, from one or more sources, about the environment around them, wherever they are.

In principle, AR systems have very few limits, mainly the information available to overlay. However, in practice, some of the core technologies, used to retrieve locations and calculate the direction the device is facing, are not well suited for use indoors. AR has many uses indoors, such as navigating a user through a large building or displaying additional information about art objects in a museum.

Inside buildings with reinforced concrete, satellite signals used for the Global Positioning System (GPS) get disrupted, causing inaccurate locations, making GPS unusable inside a building. Compasses rely on the magnetic field of the Earth, however, they too get disrupted inside a building [27]. Without these two essential technologies unusable indoors, there is a necessity for the aid of other means of measuring direction and location inside a building.

There have been many studies to determine the feasibility of other methods to get indoor location, using triangulation of wireless signals, motion tracking with accelerometers or gyroscopes, specific visual recognition or a combination of the methods.

Li et al. [14] created a meter-level indoor positioning system, that was infrastructure free, device position independant, user adaptive and easy to deploy. It relied solely on the device sensors, tracking the steps taken by the user and used algorithms to estimate the stride length for each user and determine the direction of the user at each step. However, the system struggled to correctly determine the heading, due to the disruption of the magnetic fields whilst indoors.

The solution many systems choose to overcome this problem is to implement a specific visual recognition, normally using Quick Response (QR) codes.

QR codes [33] are two-dimensional codes and an ISO standard which can encode specific information, such as text, a Uniform Resource Locator (URL) or other data. They can handle 7,089 numeric characters, 4,296 alphanumeric characters and 2,953 binary. Widely used in e-commerce, advertising and product tracking, QR codes have provided better integration with more realistic and appealing contents.

There are three main uses for QR codes in AR systems. They may contain information about the QR code, information about content to be augmented or information used to track the code. However, a system using QR codes, to provide AR information indoors, must rely on QR codes to have been already been created and placed inside a building before users are able to discover the information. An improved system would attempt to use other indoor technologies, such as image recognition rather than QR codes to provide information indoors to users. However, image recognition still requires images of the objects to be taken before users can discover them.

StAR-Wiki

Current AR systems have content provided by developers or businesses, who subscribe to systems to create content about their business. This content is limited to the imagination of the developers and the businesses that subscribe, which may not be what users are interested in. The users of these systems tend to set the trends for the information that gets created, but that creates a delay for users before they can explore the information. The solution to this would be to provide a system with very little or no content and provide users with the ability to create the content themselves, that is directly relevant to them and others.

The StAR-Wiki system I present in this dissertation will provide an AR system, using GPS and network location for outdoor use, and image recognition for indoor use, rather than relying on QR codes. No content will be provided on the developer side, but will allow users to create all the content themselves on their own devices, by tagging locations and ob-

jects with their own information. All the information will be stored in a central Wiki, which all users of the system will have access to. This information can then be kept up-to-date by users to create a Wiki containing only relevant information to users.

OBJECTIVES

This section lists the objectives of the project in order of importance, with the primary objectives providing the main features to implement. The secondary and tertiary objectives provide extensions to the project that will be completed if there is time.

Primary Objectives

- An Android app capable of displaying a central store of wiki information as a graphical display over a continuous camera stream.
- A central store of structured wiki information that can be added to or retrieved by any user.
- An in-app user interface allowing users to add or edit information stored in the wiki.
- An investigation of how to best leverage multiple sensors for the purposes of this project.

Secondary Objectives

- Using additional sensors and AR features to enhance the user experience and improve critical features, such as object and location detection.
- A user study investigating the efficacy of the system.

Tertiary Objectives

- An investigation of the feasibility of not having to rely on relaying information over the network for object and location detection.

CONTEXT SURVEY

A review of AR and Wiki projects, similar to StAR-Wiki will be discussed in this section. The key features of the systems and how they work will be summarised, along with any limitations.

FourSquare

Foursquare [6] is an app used to share and save places users visit. It provides recommendations and deals for restaurants, shopping and other entertainment based around information it gathers from the users and their friends activity in the app. There are over 45 million users worldwide, across the web, Android, iOS and many other platforms, and there have been over 5 billion check-ins to places, as of January 2014. While Foursquare is not an AR system, it is a very successful system with an easy to use service that allows users to discover the information they want by searching in categories they select.

Augment

Augmented [3] provides an AR mobile app to allow users to visualise 3D models in real time, in their actual size and environment. Developers and businesses can design 3D models on the Augment website, which can then be downloaded to devices for offline use. They provide a universal tracking image, or provide instructions on creating custom designs, that can be printed and tracked by the device to display the 3D graphics over. Alternatively, the graphics can be displayed in a trackerless mode, where the device uses the its gyroscope and accelerometer to keep models in their position whilst the device moves.

Augment provides a powerful indoor AR experience, without relying on indoor location and simply rendering graphics in user defined locations in their environment which is tracked by the device.

Onvert

Onvert [22] is an system that combines QR codes and AR. Users and developers create AR graphic overlays, on the Onvert website, along with a website link and a soundtrack. The QR code tag is then automatically generated which the user or developer adds to a base image to display the overlay on. If the tag is scanned with a normal QR code scanner, the website link is opened. If the tag is opened using the Onvert app, the overlay is displayed on top of the base image that is tracked by the app, and the soundtrack is played.

Outdoors Augmented Reality

Takacs et al. [28] produced an outdoor AR on mobile phones that matches camera-phone images against a large database of location-tagged images. They avoid network latency by implementing the algorithm on the end device and use a state-of-the-art image retrieval algorithm based on robust local descriptors. By pruning irrelevant features, based on device location, and compressing and incrementally updating features stored on the device, they ensured the system was still responsive to low-bandwidth wireless connections.

The system provides a very powerful outdoor AR image recognition, but does not provide any content to the images recognised, other than their name.

ARviewer

ARviewer [16] is an AR browser and editor which developers can integrate into Android apps. It works outside using latitude and longitude and indoors with QR-codes. It provides basic information for locations from developers, but can also be used to upload multimedia content associated with each location. The content is displayed in the AR view overlaid onto a live camera feed or in a separate list view. This system provides a full AR experience for users, but relies on content to be provided by developers and QR codes to already be placed indoors.

Layar

The Layar [13] app can scan printed material, such as magazine articles and books, and display interactive content in the app allowing users to connect with links to web content, share the items on social media or purchase items with direct mobile shopping. It has enabled over 64,000 publishers to create content for the AR app, which has been downloaded over 35

million times across both iOS and Android. Layar is primarily used in publishing, education, automotive and real estate areas.

The system does not rely on QR codes, but rather image recognition, of magazines and books, which it uses to track and overlay graphics on. Layar provides an AR system that does not relying on QR codes or indoor locations.

Wikitude

Wikitude [32] is an AR app that allows users to explore and find new places, events and activities around where they are. Users can also play games and scan objects for recognition. The content is provided through a Software Development Kit (SDK) and an online Studio which allows businesses and users to create the content themselves. The app displays the content in an AR view, in a map or on a list.

This system provides an all round solution to creating an AR system, whilst avoiding QR codes and indoor locations. However, content is only provided by developers, using the SDK, or by users, using their online Studio. Users cannot keep information up-to-date whilst exploring content in the app.

REQUIREMENTS SPECIFICATION

In this dissertation, I refer to Markers in the Wiki, as the objects that store a given latitude and longitude and other information. In the app, a Marker is the Java object used to store the information from the Markers in the Wiki.

User Requirements

Functional User Requirements

1. The user shall be able to select subscriptions of Markers they want to see.
2. The user shall be able to select a radius to search within around them.
3. The user shall be able to view all Markers within the radius they select.
4. The user shall be able to create new Markers.
5. The user shall be able to edit Markers.
6. The user shall be able to use object recognition to scan for Marker images and see the Marker information displayed on top.
7. The user shall be able to turn on/off flash on the camera.
8. The user shall be able to turn on/off notifications.

Non-Functional User Requirements

1. The user must be notified of new Markers around where they are.
2. The user must be able to understand how to view a Markers in full detail.
3. The user must be able to understand how to add and edit Markers.

Software Requirements

Functional System Requirements

1. The system shall provide an interactive AR view to display Markers around the user.
2. The system shall provide a way to change the radius of Markers to search for.
3. The system shall provide a checklist of Subscriptions that can be turned on or off which is saved on each device.
4. The system shall provide the options to toggle on/off flash and notifications.
5. The system shall provide all forms required to create and edit Markers.
6. The system shall provide a central storage of Markers for the Wiki.
7. The system shall provide the ability to retrieve and send Markers to and from the Wiki.
8. The system shall provide the ability to scan for Marker images and display information over those it can recognise.
9. The system shall provide web based tools to monitor the Wiki usage and user activity in the app for analysis.

Non-Functional System Requirements

1. The system shall be executable on any Android device, running Android version 2.3.3 or higher.
2. The system shall maintain all Markers in the Wiki.
3. The system shall provide a scheme to ensure when no Network connection is found that new Markers and edits to Markers are applied to the server when a Network connection is found.
4. The system, whilst accessing a users location, will keep it private.

SOFTWARE ENGINEERING PROCESS

This project is relatively small and has a seven day cycle and is by built a single programmer. Most Software Engineering process are designed for larger projects, with longer life cycles, built by larger teams, however, some of the values are relevant to this project.

Agile Development

An Agile development, iterative and incremental, was used to develop this project, providing flexibility where requirements could be altered and changed from one iteration to the next.

Extreme Programming

A selection of Extreme Programming features, relevant to the size of this project, were applied to the system development.

Planning was carried out incrementally with weekly meetings. In these meetings the features from the previous week were presented to the project supervisor to ensure the correctness with the objectives and new features to add in the following week were identified.

Building features incrementally allowed for a solid foundation to be created and built upon in subsequent weeks, ensuring the system could adapt to any technical restrictions or limitations discovered and whilst avoiding any problems or alterations that arose at the meetings.

Version Control (Git)

Revision controls, such as Mercurial and Git, provide tools to permit commitments of software in small increments. This allows for easier discovery of errors from previous versions and more controlled experimenting with new features, where if new changes are unsuccessful they can be reverted back to a stable revision.

Git is a distributed revision control and source code management which is used in GitHub, a web-based hosting service for software development. Git promotes itself with being quick and GitHub provides web-based tools, to easily monitoring repositories, and were both therefore used with this project to provide the best version control for managing the software.

Testing

Due to this project creating an interactive system, it was critical to check that all features of the system behaved as expected, taking into account small edge cases. A large number of test cases were produced in the form of validation tests. Each test contained steps to carry out with expected outcomes from the system. The tests were continuously run throughout the development of the system and any differences from the expected outcomes to actual outcomes, highlighted features that were broken and required fixing.

A summary of the test cases with results can be found in the *Validation Tests* section in the Appendix.

A Heuristic Evaluation was also carried out on this project to test the UI of the app to ensure it met acceptable standards. The established usability principles of interactive design [20] used were:

1. **Visibility of system status** - informing users on what is going on in the app.
2. **Match between system and the real world** - presenting words in a natural and logical order in concepts familiar to the user.
3. **User control and freedom** - allowing the user to move to and from forms in the app without restrictions.
4. **Consistency and standards** - not confusing users by presenting actions with the same meanings for different situations or actions.
5. **Error prevention** - presenting users with confirmation before committing actions after checking for possible problem conditions.
6. **Recognition rather than recall** - presenting instructions before use of features in the app.
7. **Flexibility and efficiency of use** - ability for experienced users to hide instructions once they become custom to the system.
8. **Aesthetic and minimalist design** - instructions and dialogues should contain only the minimal and relevant information.
9. **Help users recognise, diagnose, and recover from errors** - error messages should be displayed in only plain English, directly indicating the error and suggesting a fix to the problem.
10. **Help and documentation** - documentation provided should provide the necessary steps required for users to complete specific tasks.

The heuristic evaluation test consisted of observing four evaluators who were exploring the app. It aimed to highlight any features or actions users might struggle with, any violations of the usability principles of interactive design and the overall experience of the app.

Four evaluators were chosen because Figure 3, from 'A Mathematical Model of the Finding of Usability Problems [21], shows it is more beneficial when using three or more evaluators. With more evaluators, more issues can be identified. Four evaluators, the optimal number, provided the highest cost-benefit ratio, after which there is diminish returns.

A summary of observed results can be found in the *Heuristic Evaluation* section in the testing summary, in the Appendix. The *Evaluation and Critical Appraisal* section will discuss the results with regards to the project objectives and achievements.

ETHICS

The development of this project has provided almost no cause for worry over any ethical issues. The app gets the users location to set the initial location for new Markers, which can be changed for the Markers in the app, and uses the location to search for Markers around the user. The ID of the Android device the user has is stored when Markers are added and edited to the Database, however, this is only used to track each device activity within the app and is not made available to any other user on the system.

DESIGN

App Overview

Users of the app will be able to explore the world around them, using their device to display information stored in the central Wiki. This information will be Markers that all users of the system can add and edit themselves, all from their devices. There will be three ways in the app to discover these Markers.

The primary method will be using the AR View. This will provide a live feed from the camera with an overlay of icons displaying Markers in the display window of what they can see. This display window will be calculated by using the location of the device, gathered from the device Network or

GPS, the radius around the device to search in and the pitch, roll and azimuth, using the device accelerometer.

The second method will be in the Map View, which takes the Markers in the AR View and puts them on a map the user can explore, showing the locations of the Markers in their exact locations.

The third method will be using the Explore Mode. Every Marker in the Wiki will be uploaded with an image, and providing this image contains enough detail, the live feed from the camera will be scanned to try and recognise any of these images. If an image is recognised, a display will be shown on top of the live feed.

Operating System

I made a choice between Apple's iOS or Google's Android mobile and tablet Operating System (OS) to ensure all system features were possible to implement. Even though Apple offer a high-end collection of development tools, they restrict developers access to the OS. On the other hand, Android uses an open-source nature which allows for much more control over the OS and therefore access to all required devices features.

Android provides a wider user base, covering more devices, with 51.8% of the top smartphone users, compared to 40.6% who use iOS on Apple iPhones and iPads [5], in the U.S.

iOS apps are developed in Objective C [2], whereas Android apps are developed in the Java programming language, using the Android Software Development Kit (SDK). Both OSs provide C and C++ compilers for use with native code, allowing the reuse C and C++ libraries. [8]. The ability to use multiple programming languages allows for a more extensive code base which can utilised to get more features into the app.

Memory management is taken care of in Android Java, with Android's Dalvik virtual machine that performs routine garbage collection. [10] However, Objective C requires developers to manage the memory themselves, which can make programming more complex.

Extreme programming principles explains that taking the easiest option promotes faster development. Therefore, the selection of using Android provides better access to the OS and easier programming, making it the most suitable choice for this project.

Wiki

Having reviewed the apps explored in the Context Survey, all appear to store the same core information for a Marker. Therefore, the Wiki needs to store the core information for Markers:

- Name - display what the Marker is.
- Description - more detail about the Marker or add notes about it.
- Location - latitude, longitude and altitude.
- Image - display the Marker from any other location.

Users will be able subscribe to different categories of Markers, to ensure they only explore Markers they are interested in. Wikipedia contains a wealth of information about the world, where each entry is allocated one or more categories. Though the Wiki required for this project does not need to be as in depth, the subscriptions for Markers will follow some of the main categories Wikipedia [31] uses.

A subset of these subscriptions will be stored with the Markers in the Wiki:

- Arts
- Books
- Culture
- Education
- Entertainment
- Food and Drink
- Games
- Health and Fitness
- Nature
- People
- Places
- Politics
- Religion
- Science
- Shopping
- Sport
- Technology
- Toys

Marker Hierarchy

There are many situations where the location of one Marker may house many more possible Markers. For example, if an Art gallery was added as a Marker, users can discover it when walking around or on a map, but only see a summary of information about the gallery. However, the gallery itself may contain many paintings and other art pieces which can have their own Markers storing more information about the pieces. When walking around the gallery it would be useful to be able to explore just the Markers relating to the gallery.

However, if all Markers are added as individuals to the Wiki, when exploring in the AR View or Map View, the screen will be too clustered with Markers. To provide a less clustered screen, and an easy way to view only those relating to one main Marker, a simple two level hierarchy will be used.

The user will be able to make a Marker a Master, the top level, meaning it either denotes a place, like an Art gallery or museum, which can have a set of second level Markers added

to it. Keeping the hierarchy simple ensures users do not get confused when creating Markers.

This two level hierarchy provides a way for users to explore a place, and use object recognition to scan for any images inside that place. For example, art pieces can be recognised and information about them can be displayed as an overlay.

Marker Information

The type of information associated to Markers can extend just a name, description and image. Take for example a. It has opening times and items within it that have prices or objects with a functioning state. To enable this system to be live, up-to-date and interactive, requires the information to match. Markers need to be dynamic so they can store an object's current functioning state, that may suddenly break or be fixed or a place's opening times, that may change due to other related events. Therefore, Markers will be stored with the options for users to add a schedule and status.

In situations where the Wiki cannot store all the useful information for a Marker, the user will be able to set a link to a web page, containing more information, with the Marker.

Database

Every user will have access to one central Wiki that stores Marker records. Therefore, a DB will be used to provide this service. Figure 2 shows the simple DB relationship diagram required to store such a Wiki as described in the previous section.

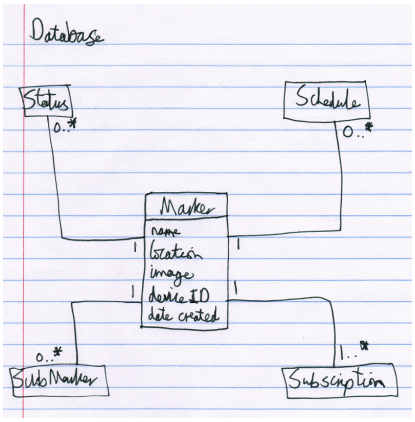

Figure 2. Wiki DB

Server

A simple server is required to send new Markers and updates to Markers, and to query for Markers around a user.

User Interface

There are three main principles or User Interface (UI) Design [18]:

- **Organise** - produce a simple and persistent conceptual structure.

  Following existing platform conventions across the UI, using spatial layouts with grid structure, standardised screen layouts and grouping common UI elements.

- **Economise** - maximise the effectiveness of the smallest set of cues needed.

  Using only essential controls produces a less clustered screen, which allows users to find relevant controls easily, whilst also not overly attracting the user's attention.

- **Communicate** - match the presentation to the capabilities of the user.

  Designing individual characters or symbols allowing users to easily notice and distinguish different UI elements. Keep the number of typefaces to a minimum when displaying different types of information. Using colour to group related UI elements and a consistent colour for screen displays.

These principles, along with the practices of Extreme Programming, focus on keeping the UI as simple as possible. Therefore, there will be a simple colour scheme that will remain constant throughout the app. Each type of UI element, such as buttons, text boxes, check buttons and labels will appear with the constant shapes, colours and fonts. All elements will be large enough to ensure no inadvertent touches on other elements.

App Flow

Figure 3 shows a simple overview of the app activity flow. The AR View will be the center of the app with access to all other activities, to add or edit Markers and switch to the Map View and Explore Mode.

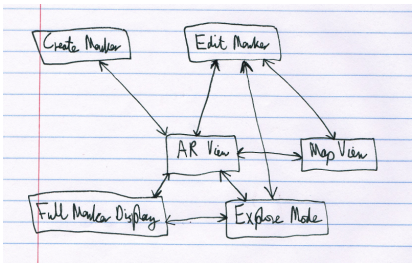

Figure 3. An overview of the app flow

AR View

Figure 4 shows a simple sketch of the main AR View. It will contain a live feed from the camera, along with a radar in the corner of the view, with dots representing Markers around the user, and icons overlaying the camera feed showing Markers in the current view window. The menu will provide options to change subscriptions, the radius to search for Markers within and links to other views.

This view will provide users the ability to view Markers in various levels of detail. The basic icon will display roughly where the Marker is, without any information of what it is. The icon can be expanded to show the name, distance away and a description of the Marker, whilst still in the AR View. This description can then be expanded to show the full details of the Marker in a full screen view. Providing multiple levels of detail will allow users to spend more time discovering Markers they have more interest in. The user will also be able to open the Marker for editing from this view.

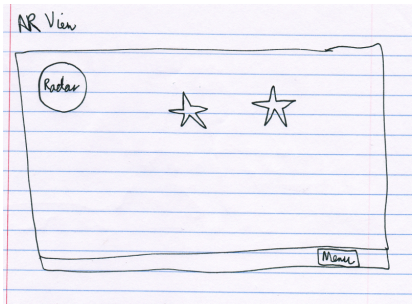

Figure 4. AR View Design

Map View

This will display the map, initially centered to the location of the user, with icons overlayed of Markers at their exact locations. Similarly to the AR View, the icons show no information other than their location, however, when expanded will show a full screen display with all the information of the Marker. Just like the AR View, the user will also be able to open the Marker for editing from this view.

Explore Mode

The live camera feed will be scanned to recognise images of Markers. Once an image has been recognised, it will be tracked by the device that will display an overlay with some information about the Marker. This can be expanded to show the full display with all the information about the Marker and also provide users the ability to open the Marker for editing.

Add/Edit Form

To keep the app simple, the creation and editing of Markers will take place in the same form but with subtle, yet noticeable, differences. The layout will take a grid formation, as seen in Figure 17. When editing a Marker, the title will change to display "Edit StAR and the create button will change to say "Apply. This will ensure the user is always aware of their action on the content being manipulated.

To select a Marker to add to, a separate form will be made. It will contain two views that can be accessed via tabs. One tab will display the same map from the Map View, with all Markers around the user. The other tab will display a list of the Markers, with their name, distance away and a thumbnail of the Markers image. Figure 18 shows a rough drawing with the layout of the list tab.

Full Marker Display

The full marker display will take a similar form to the Add/Edit form, but will display the information instead of displaying buttons or checkboxes used to add or edit information. Also included in this, for Master Markers, will be a tabbed view, with a map and list, of all the Markers that have been added to it. These differences will be enough to ensure users do not confuse this form with the Add/Edit forms.

IMPLEMENTATION

Development Software

The open-source software, Eclipse [30], is used to build applications using a large set of comprehensive frameworks and tools, that manage software for the duration of its lifecycle. The Android Development Tools (ADT) [7] is a plugin for Eclipse, designed to provide a robust environment to develop Android applications. Eclipse, with the ADT plugin was used to develop the app for this project.

AR Frameworks

There were many frameworks that provide AR support, such as AndAR [1], but these relied on QR codes. There were fewer frameworks providing an easy base for Marker data structures and gathering Marker information from a server.

ARviewer [15], while not relying on QR codes, only provided a service to invoke their own app with a layer of Markers. Whereas, the android-augmented-reality-framework (AARF) [12] provided an extensive Marker data structure along with the ability to send requests to multiple servers, to retrieve Markers, all in one application. The code it provided could be heavily altered and extended to allow for complete customisability and was therefore chosen as the base for this app.

Qualcomms Vuforia platform, supported on iOS, Android and Unity 3D, provides an SDK with technical resources for computer vision-based image recognition along with a large set of features and capabilities. Along with offering the use of a Cloud DB to store up to one million images, it can recognise any of those images on the phone at run-time, even in low light or when the image is partially covered. [25] This platform was used in the app to store images of the Marker to be recognised whilst exploring.

App Framework

At the base of the framework was the Marker object structure, storing the name, description, image and all the other main attributes. Extending the Marker object was the IconMarker, which added an icon to be displayed in the AR View. For this app, IconMarker was extended further to create the StARMarker object, to store all the other information required, including the schedule, status history and the Marker added to or a list of Sub-Markers. The schedule contained a list of schedule events, with a name, description, start and end times. The status history contained a list of previous states: working, not working, known fault, unknown fault, being fixed, unassigned.

App Flow

Figure 5 shows the app flow between all forms in the app. The AR View was the central form, providing access to all other forms. The menu provided access to the Map View and Explore Mode, the tag button opened the create Marker form. Long holding Markers in the AR View allowed users to open the Marker for editing, and tapping on the Markers, expanding them through the different levels of information eventually took users to their full display.

AR View

AARF provided a main AR View screen with a radar, showing dots where the Markers around the user were, a slider, to easily change the radius to search within, and a tag button, to

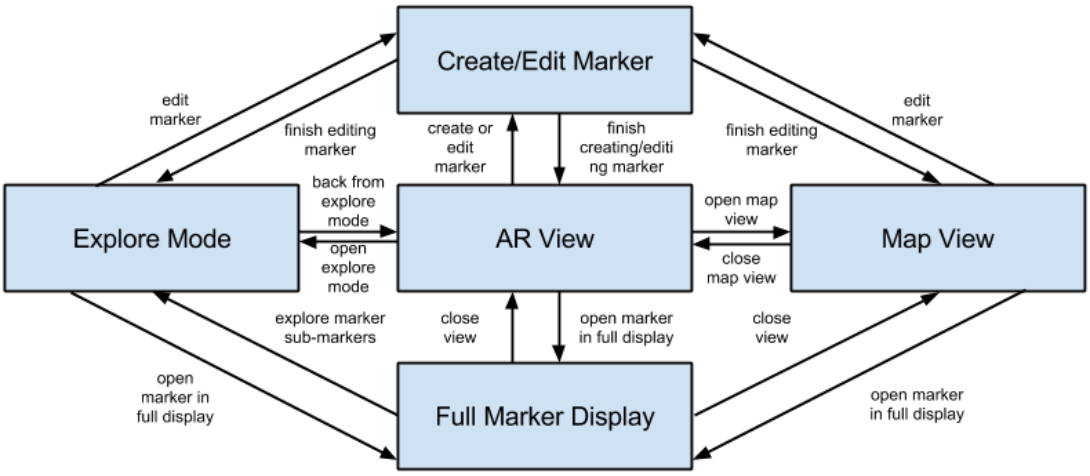

Figure 5. The app form flow diagram

take a photo and take the user to the add Marker form. Only Markers that were Masters were displayed and the icon used was a star to represent the StAR-Wiki. Figure 6 shows how this view appeared to the user.

Figure 6. The AR View with a Marker icon displayed

When the user clicked on a Marker, it opened a simple description for the Marker, seen in Figure 7.

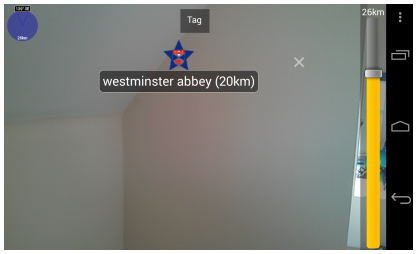

Figure 7. The AR View with a Markers basic description displayed

Clicking on this description opened the Marker to its Full Marker display, described in the *Full Marker View* section.

This view worked very well when only a few Markers appeared, that were spread out around the user. However, when Markers were located very close together, the screen displayed icons that overlapped, causing problems when a user wanted to pick out an individual one.

To solve this problem, I created two different schemes. The AARF continuously updated and drew the Markers on a short time loop, meaning the solution to this problem required a scheme that was quick to run. The first was a collision detection and the second a more complex cluster algorithm.

The first scheme detected when any two Marker displays collided and moved them away from each other. It used a simple rectangle collision detection and updated the Markers screen locations accordingly.

The second scheme created a cluster object, making groups of all Markers that overlapped and displayed a different icon, to show there were multiple icons grouped together. Clicking the icon would expand the icons within it around the center of the cluster. However, it required a higher level of objects, encapsulating the Markers, to be created and tracked whenever the device was moved, subscriptions changed and the radius to search for updated. Some Markers within the groups were removed, or new ones added, creating a longer delay in updating, which caused the display to suffer.

Therefore, the app only applied the collision detection scheme, which provided the most simple and quick solution, without delaying the display update speed.

Users could also long touch a Marker to bring up a dialog with options to edit the Marker, taking the user to the Add/Edit Form, where the Marker information was loaded into the form ready for editing. The other option opened Google Maps with directions from the users current location to the location of the Marker.

Map View

Figure 8 shows the Map View with a Marker icon displayed.

All maps used in the app were provided by the Google Maps Android API V2, which allowed the app to display maps, initially set to a set location, with custom icons for Markers that could be displayed at their geographical location and process clicks on those Markers.

A button to change the type of map displayed, between a normal roads only, a satellite or a hybrid, was overlaid onto each map in the app. This improved the searching ability for users to find the exact location of a desired Marker.

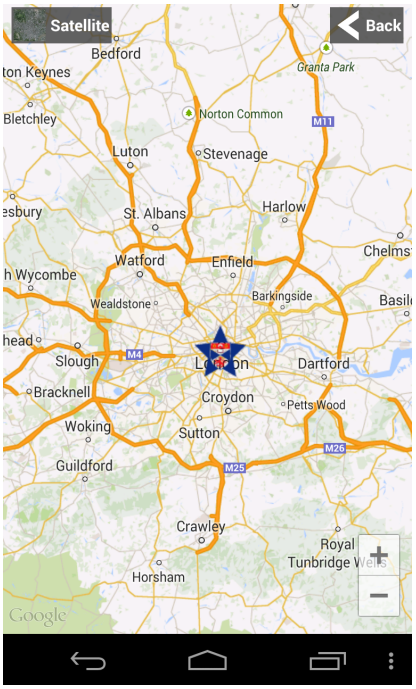

Figure 8. The Map View with a Marker

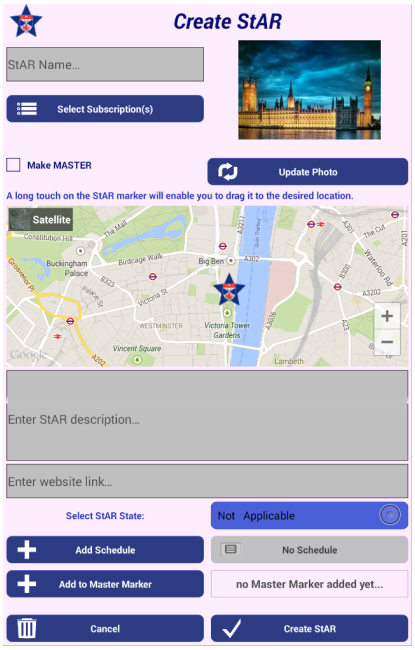

Figure 9. The Create Form

Add/Edit Form

This form allowed users to create and edit Markers. It had to provide users the ability to set the name, description, web page link, schedule, initial functioning state (if applicable) and if the Marker was to be a Master or added to a Master. Figure 9 shows the form, after a photo was taken with the tag button from the AR View.

When the user confirmed the creation or application of changes, a progress dialog was shown whilst the app sent the appropriate request to the server. If no network connection was found or the task took longer than 30 seconds, the user was notified and the information was written to a file on the device. A background task continuously attempted to upload the information to the server.

This solved the problem scenario when a user was exploring Master Markers, viewing their content from the server whilst outside with a strong mobile wireless connection, but then deciding to explore a Master Marker inside a building. The connection strength become too weak or stopped entirely. The app still allowed them to make changes, which were uploaded to the server once they regained a strong enough connection.

Schedule Form

To add a schedule to a new Marker a separate form was used. It allowed users to create events, typing a name and description and selecting a day, from a list containing days of the week and specials day, such as Christmas and New Years Day. There were lists to select the start and end times. Figure 10 shows the form with some events added to the list. Users could long touch an event to edit or delete it.

Add to Master Form

Adding the Marker to a Master used another form that provided users the ability to search for Markers around them on a map or in a list, enabling users to find the Marker they desired very efficiently. Figure 11 shows the form where the Marker selected appears with a highlighted icon.

Full Marker View

This form, rather than providing text boxes and buttons to add or edit information, provided a simple form where the information of a Marker was displayed. If the Marker had a web page link, the link would appear and be clickable, taking the user to the web page. The current status for the Marker was shown, colour coded depending on the state. For example, if it was working, the text appeared green, whereas if it was not working it appeared red. For Master Markers, a tabbed view, with a map and list, showed of its Sub-Markers. A list of schedules for the Marker was also show, for those it contained.

Figure 12 shows the full Marker view displaying information for a coffee machine. The Marker had no Markers added to it, so the explore button was hidden.

The buttons at the bottom of the form either closed the form or took the user to the Explore Mode, to scan for Sub-Markers added to the Master, without relying on indoor location to find them.

Explore Mode

Using the Cloud DB provided by Qualcomm Vuforia and the image recognition native code, the Explore Mode was able to scan the live camera feed to detect any image stored in the DB. If this form was accessed from the full Marker display, a filter was added to only scan for a list of Markers, that were added to a Master Marker. If it was accessed from the AR

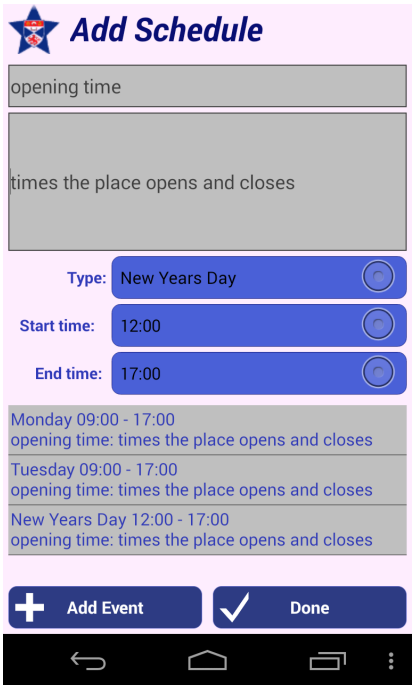

Figure 10. The Schedule Form

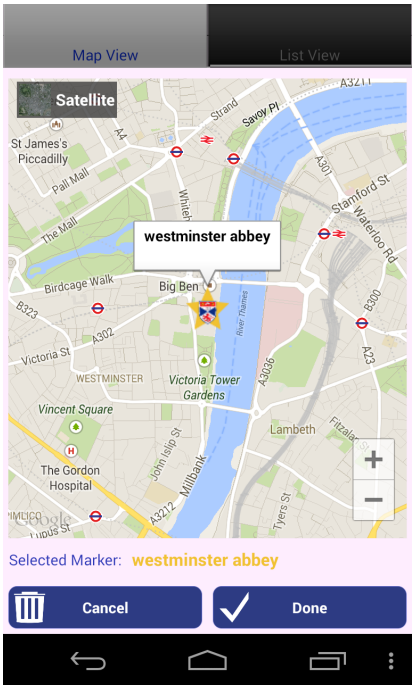

Figure 11. The Add to Master Form in its map tab

View, no filter was used, and it scanned for any image in the DB within the radius set in the AR View.

One issue this project tried to explore was how to better AR systems whilst indoors, where precise locations are difficult to attain. Using the full Marker display, with the list and map of Sub-Markers, and the Explore Mode, scanning the environment around the user, there was no need to rely on indoor location.

The camera feed was scanned, with points of detail in the feed being displayed, as seen in Figure 13. Provided the images used for the Markers had enough detail, they could be recognised.

Once an image was recognised, a bitmap overlay, made from a summary of the Marker information, was displayed over the object recognised, as seen in Figure 14, which was then tracked by the device as it moved. If the object became untrackable, out of the camera view or from a difficult angle, the overlay was then viewed directly to the middle of the screen, as seen in Figure 15.

Notifications

Another way to ensure users discovered Markers around them was to continuously search for Markers they might be interested in as they walked around not using the app.

This was done by creating a background Android Service, that tracked the users location (keeping it private), and queried the Wiki for Markers within a 500 metre radius that had a subscription that matched one of the users selected subscriptions. Only Markers not added to the Wiki by the user and Markers not previously discovered using the Service were searched for.

Once a new Marker was found, a notification was sent to the device with the name and distance away. When the user clicked on the notification it opened the Marker in the full Marker display, so it could then be explored.

Notifications were only displayed once every 30 minutes to ensure they did not receive too many.

Instructions

Instructions were added using the MrTips 2-Class [29] library, making sure users could quickly and easily understand how to use the app. The library provided a quick way to add instructions to be displayed when forms opened. The user had to check a box in order not to see them again.

Figure 16 shows an example instruction in the AR View, with notes on how to use the form.

Database

MySQL [23] provided a well supported DB to store the Wiki information. Figure 19 in the Appendix shows the entity-relations diagram for the DB used.

Each Marker had a core set of attributes, a name, location, image, description and web link. The status history, schedule events and subscriptions were stored in separate tables. This made it easy to add extra information to the Marker.

The edits and logging tables will be discussed in the *System Analysis* section.

Server

To connect the app to the Wiki DB, PHP files were created. The app made HTTP POST and GET requests to these PHP files to send and retrieve Wiki information. SQL injection

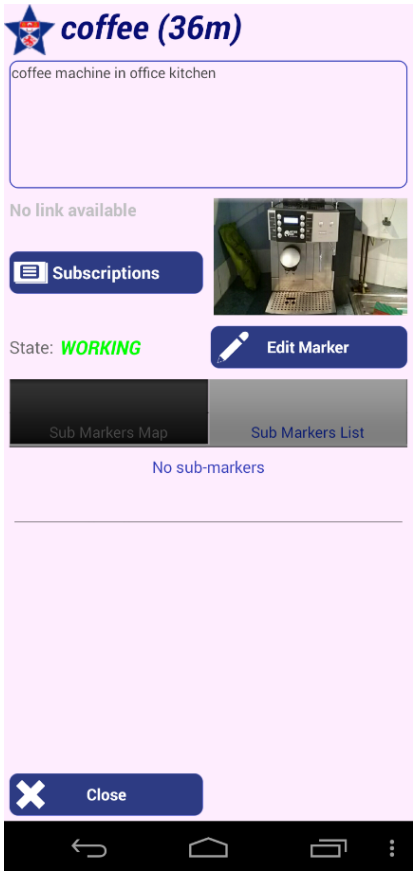

Figure 12. The full Marker display, showing the information for a coffee machine in an office.

was used to ensure the system remained secure throughout its usage.

System Analysis

To analyse the system usage, Markers were stored with timestamps of when they were added and edited. When a Marker was edited, a record with the Marker name, timestamp and device ID was added to the Edits table in the DB. These records allowed the system to track the number of times Markers from different subscriptions were created and edited. Device IDs were also stored with the Marker records to track the number of Markers created and edited from each user.

Along with this, to fully analyse the app, user activity was logged in the app. For example, the time spent in the AR

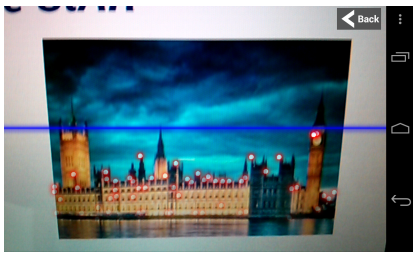

Figure 13. Scanning an image for points of detail

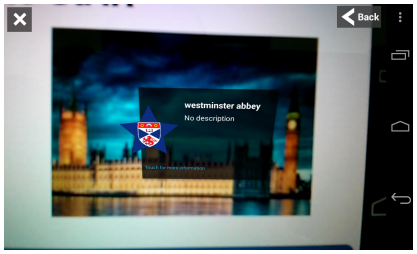

Figure 14. Overlay of Marker information

Figure 15. Scanning an image for points of detail

View, Map View and Explore Mode were logged, to find out which view users spent most of their time in. Time spent adding and editing specific information of Markers was also logged to discover what type of information users tended to provide and keep up-to-date.

Together with the time spent in the different areas of the app, counts of many actions were stored to ensure the average time spent in these different areas could be calculated. Every time the app was started, a session ID was created, representing the number of times the users had used the app. The session ID was assigned to each record in the logging tables allowing for a comparison between different user sessions. An example of the counts tracked by the system included:

- Number of times changing the subscriptions.
- Number of clicks on Markers in the AR View.
- Number of clicks on the web link from a Marker.
- Number of Markers created and edited.

When there was no network connection, the system wrote the log events to a local file and continuously ran a background task to try to upload them. This ensured no logging information was lost.

Analysis Tools

A web page was made to track the usage of the system. The home page displayed the total number of users, the number of Markers created and edited.

Another page showed a chart, shown in Figure 20 in the Appendix, of the number of Markers created and edited for each type of subscription. This enabled an easy comparison of exactly what types of Markers users tended to create and edit.

On another page, using the Google Maps API, a map of the entire Wiki, displaying icons for all Markers was used to give an overall usage of the system. Figure 21, in the Appendix,

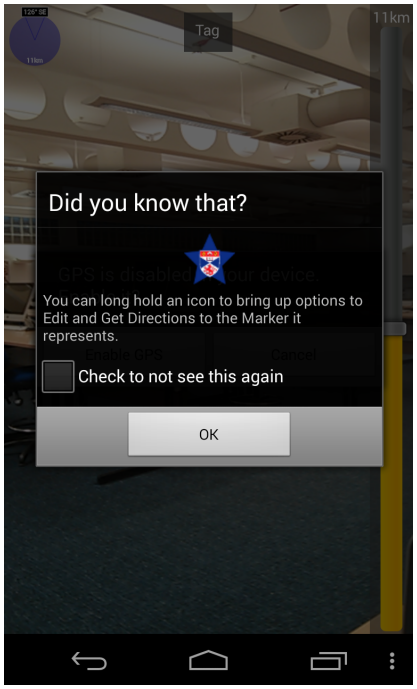

Figure 16. Instructions being displayed in the AR View.

shows a screenshot of this page. When clicking an icon, it displayed the name, description, image, date added, and web link, if the Marker had one.

The final page provided a list of device IDs to select from which showed the activity of the device in the app.

EVALUATION AND CRITICAL APPRAISAL

Objectives

- **Primary**: All completed, though with only a short investigation of how to best leverage multiple sensors. Whilst outdoors, locations could only be attained by GPS or networks, either Wifi or mobile Internet, therefore creating very limited leverage between them. Indoors, the project attempted to not rely on location tracking, but instead investigate object recognition to provide an AR experience.

- **Secondary**: All completed. The system used image recognition, provided by Qualcomms Vuforia, in the Explore Mode. Due to the time limit of the project, there was no time for a full user study, though the system does provide the logging of user activity so a full user test can be carried out at a later date. Instead, within the time limits of the project, validation and heuristic evaluation tests were carried out to ensure the system worked as expected.

- **Tertiary**: All completed. Using the Qualcomm Vuforia SDK allowed image recognition from images in a Cloud DB. The DB, that can store over one million images, stored all images for Markers in the Wiki and produced very quick recognition results over a network. The SDK provided functionality for image recognition from images stored locally on the device, however, it is designed for use with a small number of images. In order to integrate local image recognition with this system, all Marker images in the Wiki would have to be stored locally on the device, which would require a vast amount of memory.

Testing

Validation

To assess the correctness of the system, I conducted tests using a script with tasks to carry out. Each test had desired results which could be compared against the actual results seen when running the tests. The tests evaluated all the main features, including smaller edge cases, of the system and showed that all functional and nonfunctional requirements were met. A sample of the tests can be seen in the *Validation Tests* section in the Appendix.

Heuristic Evaluation

The heuristic evaluation was carried out on four evaluators, providing the highest cost-benefit ratio. With more evaluators, more issues can be identified. However, after four evaluators, there are diminishing returns, as seen in Figure 3, from 'A Mathematical Model of the Finding of Usability Problems [21].

The test involved observing the evaluators using the app. The tests aimed to discover any violations of the usability principles of interactive design [20] and the overall experience users had with the app.

The results from the tests can be seen in the *Heuristic Evaluation* section in the Appendix. The results produced a list of violations of the usability principles of interactive design.

One violation occurred when users were adding schedule events. They were able to press the done button whilst in the middle of creating or editing an event, without a confirmation to cancel the creation, taking them out of the form and losing the event. To solve this, a confirmation dialog was put in whenever the done button is pressed, to check if the user is done adding/editing events.

Some of the evaluators noticed that the tips displayed in the app were not present in some areas of the app which they initially struggled to use. Noticeably when creating or adding schedule events or when they can long press a Marker icon. The solution was to add extra instructions and tips to cover the smaller features of the app.

When selecting a Marker to add to in the app, some of the evaluators noticed that once a Marker had been selected, all icons remained the same in the map. This meant they did not know which had just been selected. The solution to this was to highlight the selected Marker icon in the map by changing its colour.

In the Explore Mode, when it could not initialise the image recognition, most likely from a weak Internet connection, it displayed an error message stating, "Failed to INIT Qualcomm RECO". The evaluators were confused, to fix this problem, the message changed to explain the problem, why it most likely happened and to try using the service later.

The main problem evaluators had came when they wanted to edit a Marker. Once viewing a Marker in its full display,

users had to back out of the display and apply another action to open the Marker for editing. The solution to this problem was to add an edit button in the full Marker display, taking users directly to the edit form.

User Study

A user study would enable a full investigation of the system. However, as there is no content in the system, with users providing it all themselves, it is very difficult to run a short study. The system would need to be used regularly, by a large number of users, who would provide the content. When the Wiki contains a lot of information, users will be able to discover more information and data about their usage with the information can be gathered through logging their activity in the app.

Although I did not carry out a full study, the app is prepared for the study, as it logs all activity from each user, storing it in a central DB for analysis. The web tools provided the interface to select a device ID to view an anonymous users app activity. It displayed live information from the device, such as the amount of time spent in various areas of the app and how many Markers were created, edited and visited.

By storing the amount of time spent in certain areas of the app, an evaluation on what information in Markers provide the most useful to users and which AR delivery method works best can be carried out.

Comparison to Related Work

This project created an AR system, where content was stored in a central Wiki, available to all users, and overlaid onto a live camera stream of the real world. Wikitude and Layar provide very similar systems, where Wikitude focuses on the Wiki of information provided to users, and Layar focusing on overlaying graphics onto real world objects.

However, content in Wikitude is created from developers using the Wikitude SDK and content for Layar is provided by developers or businesses using web based tools. This project relied on all content to be created by users on the end devices.

AndAR relies on QR codes to track target objects when overlaying graphics. This project overlays graphics in a similar manner, but by tracking real world objects that have had images taken and stored in a Cloud DB.

CONCLUSIONS

Achievements

All objectives were completed, producing a fully functioning mobile AR system, with image recognition for use indoors and outdoors, where all information is provided by users entirely on end devices. It allows users to tag objects or places, stored in a central Wiki, which all users of the system have access to, who can then update information, ensuring the Wiki contains up-to-date information.

The main goal of the project changed from investigating location service technologies, to investigating whether an AR system where all information is provided by users entirely from their devices and an alternative method for indoor AR, using image recognition rather than QR codes, provided a usable and effective AR experience for users.

The new goal enabled this project to deliver a different AR system to those seen previously. However, due to the time limitations, it is not known if this system provides users with an improved experience.

Future Work

To build the Wiki information, to the point where a full analysis of the system can be carried out, will require the app to be released on the Google Play Store. Then users all around the world can experience and build the content for the system. Receiving comments from these users about what information they find useful, and what they would like to add, may provide a better understanding of what users find more interesting and useful.

The information given to Markers can greatly extend a web link and schedule. For example, Markers could represent items users could purchase, attaching a price to them. Markers can also represent places, such as restaurants and cafes, which could be associated with a menu. Multimedia is often attached to objects in the real world, as seen in many current AR systems, like Layar, that provides videos to magazine articles and books.

Users want to discover new information in the environment around them, with ability to access specific information about the places or objects they find. To expand the Wiki, I could add these attributes to Markers. Alternatively, users could be given a base Marker framework, with a location and name, and be given the ability to form their own type of Marker, all whilst using the app. This would provide users with the freedom to make Markers with the information they want.

REFERENCES

1. AndAR. Andar - android augmented reality. **https://code.google.com/p/andar/.**
2. Apple Inc. Develop apps for ios. **https://developer.apple.com/technologies/ios/.**
3. Augment. Augment is an augmented reality app. **http://augmentedev.com/.**
4. Azuma, R. T. A Survey of Augmented Reality. *Presence 6* (1997), 355–385.
5. ComScore. Comscore reports september 2013 U.S. smartphone subscriber market share. **http://www.comscore.com/Insights/Press_Releases/2013/11/comScore_Reports_September_2013_U.S._Smartphone_Subscriber_Market_Share.**
6. Foursquare. About foursquare. **https://foursquare.com/about/.**
7. Google. Adt plugin. **http://developer.android.com/tools/sdk/eclipse-adt.html/.**
8. Google. Android ndk. **http://developer.android.com/tools/sdk/ndk/index.html.**

9. Google. Google glass. http://www.google.co.uk/glass/start/.

10. Google. Managing your app's memory. http://developer.android.com/training/articles/memory.html.

11. Höllerer, T., and Feiner, S. Mobile augmented reality. *Telegeoinformatics: Location-Based Computing and Services. Taylor and Francis Books Ltd., London, UK* (2004).

12. Justin Wetherell. android-augmented-reality-framework. https://code.google.com/p/android-augment-reality-framework/.

13. Layar. Layar - augmented reality. http://www.wikitude.com/.

14. Li, F., Zhao, C., Ding, G., Gong, J., Liu, C., and Zhao, F. A reliable and accurate indoor localization method using phone inertial sensors.

15. LibreGeoSocial. Arviewer sdk. http://www.libregeosocial.org/node/24.

16. LibreGeoSocial. Libregeosocial: Augmented reality floss. http://www.libregeosocial.org/.

17. Mackay, W. E. Augmented reality: Linking real and virtual worlds a new paradigm for interacting with computers, 1998.

18. Marcus, A. Human-computer interaction. In *Readings in Human-Computer Interaction: Toward the Year 2000*, R. M. Baecker, J. Grudin, W. A. S. Buxton, and S. Greenberg, Eds. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, ch. Principles of Effective Visual Communication for Graphical User Interface Design, 425–441.

19. Milgram, P., Takemura, H., Utsumi, A., and Kishino, F. Augmented reality: A class of displays on the reality-virtuality continuum (1994). 282–292.

20. Nielsen, J. Heuristic evaluation. http://www.nngroup.com/articles/ten-usability-heuristics/, 1995.

21. Nielsen, J., and Landauer, T. K. A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, ACM (New York, NY, USA, 1993), 206–213.

22. Onvert. Overt - more than you see. http://onvert.com/.

23. Oracle. Mysql. http://www.mysql.com/.

24. Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA, 1980.

25. Qualcomm. Qualcomm vuforia. http://www.qualcomm.com/solutions/augmented-reality.

26. Stephen Cawood, Mark Fiala. Augmented reality - a practical guide. Programatic Bookshelf (2007), 1–3.

27. Study Tour Pixel - University of Twente. Toward the indoor use of location-based augmented reality.

28. Takacs, G., Xiong, Y., Grzeszczuk, R., Chandrasekhar, V., chao Chen, W., Pulli, K., Gelfand, N., Bismpigiannis, T., and Girod, B. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *In Proceeding of ACM international conference on Multimedia Information Retrieval* (2008), 427–434.

29. Tezan Enssat. Mrtips library. https://github.com/lethargicpanda/mrtips.

30. The Eclipse Foundation. Eclipse home. http://www.eclipse.org/.

31. Wikipedia. Portal:contents/catergories. http://en.wikipedia.org/wiki/Portal:Contents/Categories.

32. Wikitude. Wikitude - the app. http://www.wikitude.com/.

33. Yoon, H., Park, N., Lee, W., Jang, Y., and Woo, W. QR code data representation for mobile augmented reality. *The International AR Standards Meeting* (2011), 17–19.

APPENDIX

TESTING SUMMARY

This section presents the test results carried out in this project.

Validation Tests

A script containing a list of system validation tests was carried out during the development of this project. The system was interactive, with the main features providing visual feedback on the device, which is difficult to write unit testing for. Instead, the script had interactive test cases, including many edge cases, checking the features of the app against desired results. Only a sample of the tests are given below, due to the large number of test cases required for the whole system.

Add a Marker
Steps:

- Connect the device to the internet.
- Press the Tag button to take a photo of the Marker.
- Fill in the Create StAR form with a name and subscriptions
- Make the Marker a Master or add it to a Master

Desired Result: A progress dialog should show, whilst the Marker is sent to the server. Once complete, a dialog is displayed showing the Marker has been successfully added. The Marker should be visible in the AR View and Map View within 30 seconds and in Explore Mode in about 5 minutes.
Actual Result: Progress display shown, after which a message about the Marker being created was shown. The Marker became discoverable in the app within 30 seconds.
Pass/Fail: *Pass*

Change Subscriptions
Steps:

- Open the app into the AR View
- Turn on all subscriptions
- Add a Marker with a single subscription
- Once the Marker has loaded in the AR View
- Uncheck the subscription for the Marker
- Recheck the subscription for the Marker

Desired Result: The Marker should become invisible in the app when the subscription is unchecked and reappear when rechecked.
Actual Result: Marker disappeared and reappeared as expected.
Pass/Fail: *Pass*

Full Marker Display
Steps:

- Open the app to the AR View
- Find a Marker in the form
- Click the Marker to bring up its description still in the AR View
- Click the description to bring up the full Marker display

Desired Result: All the Marker information should be displayed in the full Marker display form.
Actual Result: All information was displayed in the form correctly.
Pass/Fail: *Pass*

Edit Marker
Steps:

- Open the add to the AR View or Map View
- Find a Marker in the form

- Long hold the Marker to bring up an options dialog
- Select the edit option
- Edit some information with the Marker
- Apply the changes
- Wait for the updated Marker to be sent to the server
- View the Marker in full detail

Desired Result: The Marker detail should have changed from the original information, seen in the edit form, to the new information seen in the full Marker display.
Actual Result: All information is seen to change, including the name, description, link, schedule, location.
Pass/Fail: *Pass*

Sub Markers
Steps:

- Create a Master Marker and add another Marker to that Master
- Open the Master Marker in the full display

Desired Result: The Marker added to the Master Marker being viewed in full should display a map and list containing the sub Marker.
Actual Result: The Marker is in the list and on the map.
Pass/Fail: *Pass*

Explore Mode
Steps:

- Create a Marker and wait for 5 minutes (for the image to be uploaded to the Cloud DB)
- Scan the original object(s) used for the Marker image

Desired Result: An overlay with the Marker name and description should be displayed over the object(s) and tracked by the app. When moving completely away from the object(s) the overlay should move and stay in the centre of the screen.
Actual Result: The overlay is displayed and tracked by the app.
Pass/Fail: *Pass*

Heuristic Evaluation

This section will summarise the results from observing the four evaluators whilst they used the app, referring to the usability principles of interactive design.

Principle: **Error Prevention**

When adding a schedule event for a Marker, pressing the done button took the app back to the create/edit form without checking if the user wanted to add the event being created. This lost the event.

Principle: **Recognition rather than recall**

Instructions were not present in all areas of the app. This meant some users to struggled to understand how to perform certain actions, for example when adding or creating schedule events and when to long press a Marker icon in the Map View.

Principle: **Visibility of system status**

When selecting a Marker to add to when creating or editing a Marker, all Marker icons in the map appeared the same colour, even after one had been selected.

Principle: **Help users recognise, diagnose, and recover from errors**

While in Explore Mode, when the app could not initialise the image recognition, error messages were not explained in plain English, confusing users. The solution was to change these messages to directly indicate the error and suggest a fix for the problem.

Principle: **User control and freedom**

Users were frustrated that when they viewed a Marker in its full display, they had to back out of the full display and apply another action in order to open the Marker for editing. This restricted users freedom within the app.

ADD/EDIT MARKER DESIGN

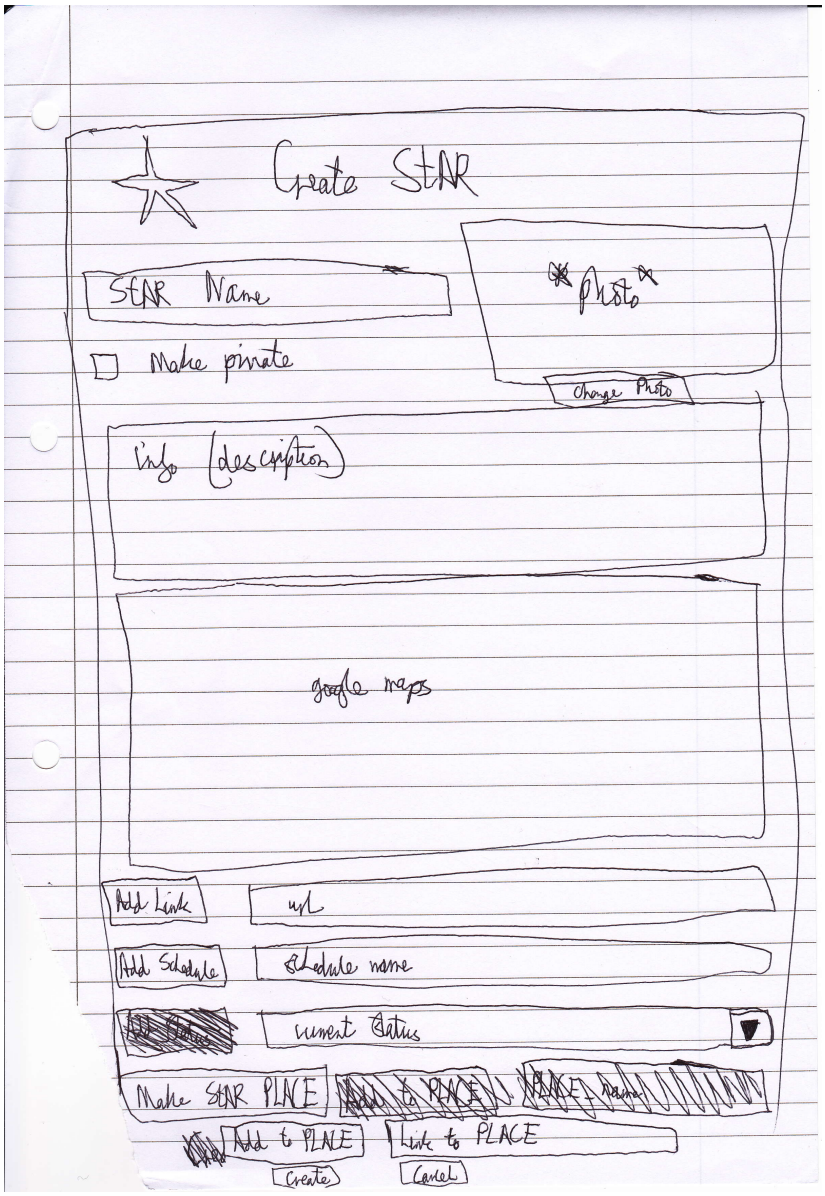

Figure 17. Design for the Add/Edit form

ADD TO MASTER DESIGN

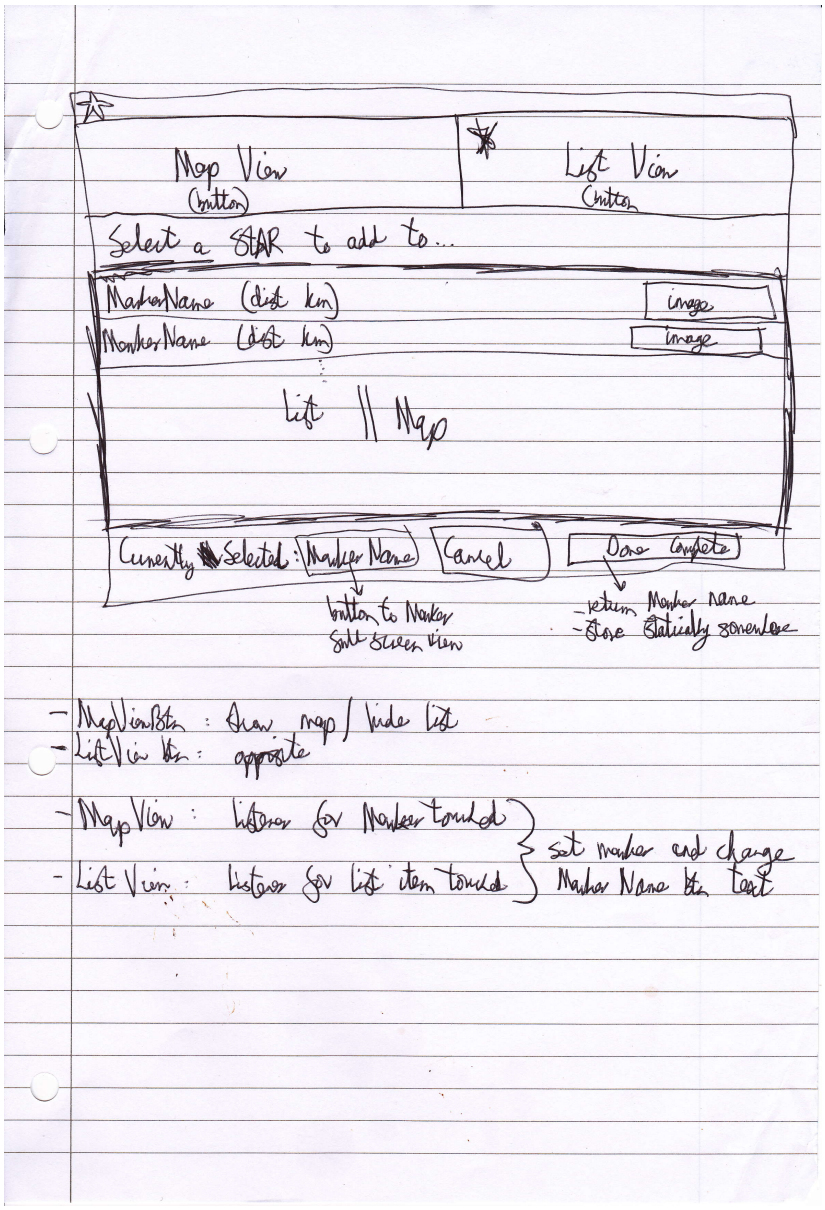

Figure 18. Design for the Add to Master form

DATABASE ENTITY-RELATIONS DIAGRAM

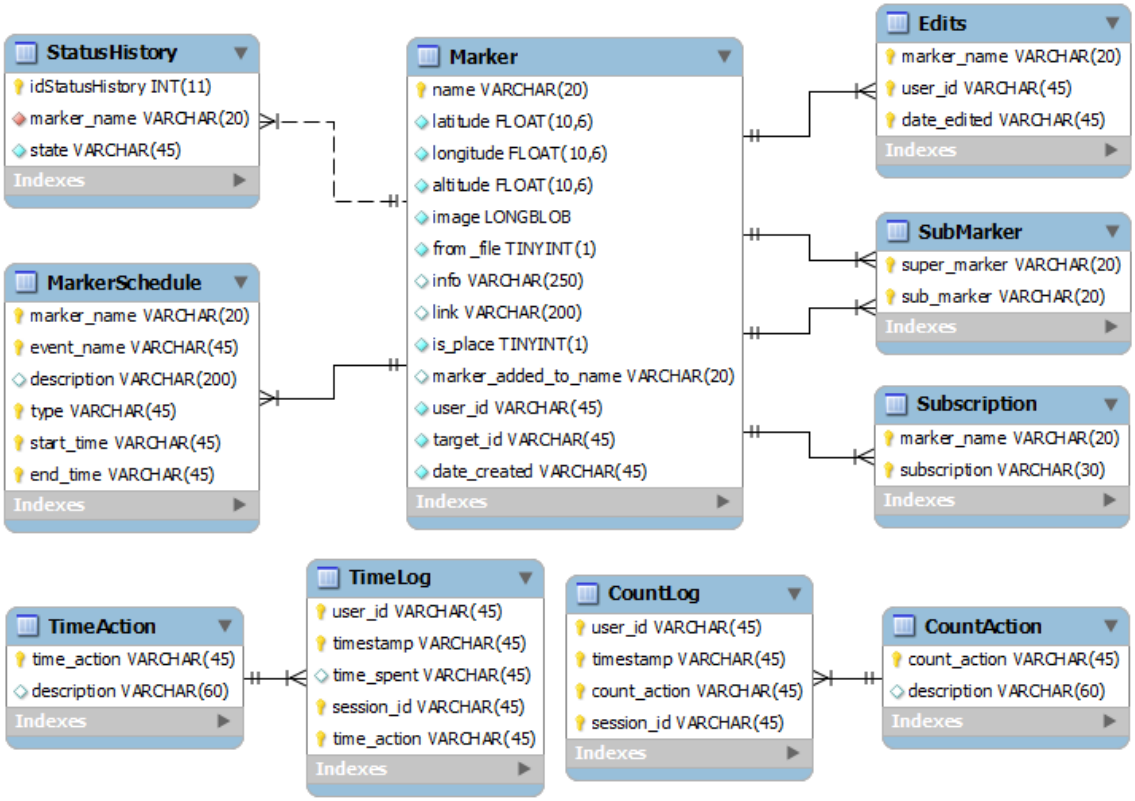

Figure 19. The entity-relationship (ER) diagram used to store the Wiki Markers and log user activity.

SYSTEM TOOL SCREENSHOTS

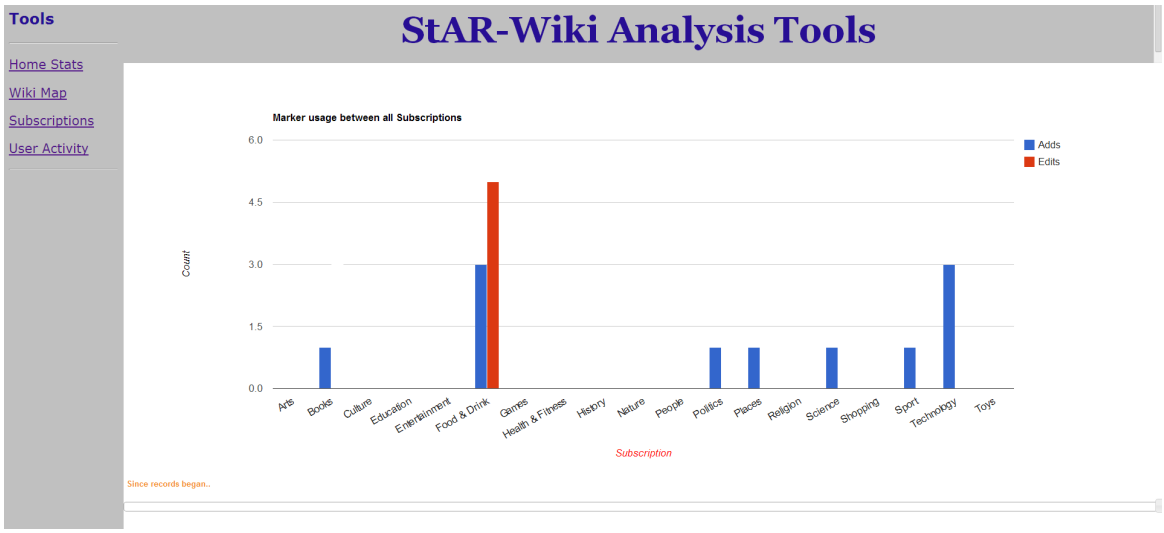

Figure 20. The web tools, showing the number of Markers added and edited for each subscription.

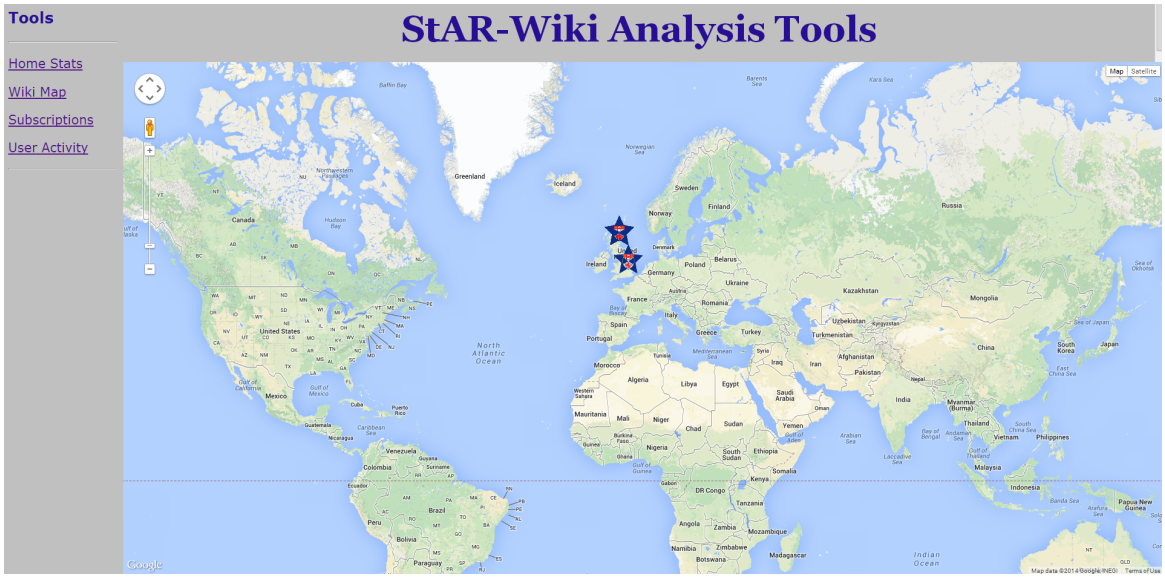

Figure 21. The web tools, showing the entire Wiki of Markers in the world.

SOFTWARE LISTING

- The app code can be found in the ”StAR-Wiki” directory
- The Android application package file (APK) file, used to distribute and install the app can be found in the ”APK” directory
- The server code can be found in the ”Server” directory
- The web analysis tools can be found in the ”Analysis Tools” directory