# WeKnowIt

**Emerging, Collective Intelligence for Personal, Organisational and Social Use**

**FP7-215453**

# D7.3.1

# Initial Emergency Response Case Study Implementation

| | |
|---|---|
| **Dissemination level** | Public |
| **Contractual date of delivery** | Month 20, 30-11-09 |
| **Actual date of delivery** | 18-12-09 |
| **Workpackage** | WP7, Case studies |
| **Task** | T7.1.3 |
| **Type** | Prototype |
| **Approval Status** | Final |
| **Version** | 1.0 |
| **Number of pages** | 72 |
| **Filename** | D7.3.1.doc |

**Abstract:**

This deliverable describes the implementation of the Emergency Response Case Study Demonstrator. WeKnowIt is exploring the concept of Collective Intelligence – a form of intelligence derived from the cooperation of multiple layers of intelligence using a shared knowledge base. The emergency response demonstrator shows how Collective Intelligence can support individuals and organisations involved in a serious emergency in a typical city.



co-funded by the European Union

# History

| Version | Date | Reason | Revised by |
|---|---|---|---|
| 0.1 | 22/10/2009 | TOC and Timeplan | USFD |
| 0.2 | 10/11/2009 | First Contributions | SMIND |
| 0.3 | 12/11/2009 | First Integrated Version | USFD |
| 0.4 | 25/11/2009 | Second Contributions | SMIND |
| 0.5 | 25/11/2009 | Second Contributions | USFD |
| 0.6 | 28/11/2009 | Second Integrated Version | USFD |
| 0.7 | 03/12/2009 | Reviewed Version | TID |
| 0.8 | 06/12/2009 | Addressed Reviewer Comments | USFD |

# Author list

| Organization | Name | Contact Information | Reason |
|---|---|---|---|
| USFD | Simon Tucker | s.tucker@dcs.shef.ac.uk | Author |
| USFD | Gregoire Burel | g.burel@dcs.shef.ac.uk | Interfaces |
| USFD | Alfonso Sosa | alfonso.sosa@gmail.com | Interface |
| SMIND | Tomek Kaczanowski | tomasz.kaczanowski@softwaremind.pl | Architecture |
| TID | Manuel Escriche | mev@tid.es | Reviewer |

# Timeplan before delivery

| Next Action | Deadline | Care of |
|---|---|---|
| Agreement on the TOC; statement of partners' interest and allocation of work | 26/10/2009 | USFD |
| 1st contributions received | 06/11/2009 | All |
| 1st integrated version | 08/11/2009 | USFD |
| 2nd contributions received | 22/11/2009 | All |
| 2nd integrated version | 23/11/2009 | USFD |
| Deliverable ready for being internal reviewed. | 24/11/2009 | USFD |
| Reviewer contributions received | 28/11/2009 | TID |
| Revision to address reviewers comments | 29/11/2009 | ALL |
| Final deliverable ready for being reviewed and submitted. | 30/11/2009 | USFD |

# Executive Summary

This document describes the implementation of the first prototype of the emergency response demonstrator for the WeKnowIt project. WeKnowIt is exploring the concept of Collective Intelligence – a form of intelligence derived from the cooperation of multiple layers of intelligence using a shared knowledge base. The emergency response demonstrator shows how Collective Intelligence can support individuals and organisations involved in a serious emergency in a typical city.

The development of the demonstrator began by interviewing target users of the system and defining a set of requirements as to how the system should function. Following this, a *scenario* was defined which described how the eventual system would be used in the context of a flooding event in the city of Sheffield. Using this scenario a set of mock-up interface designs was made which illustrated what the resulting system may look like.

To define the requirements for the first prototype of the demonstrator a *mini-scenario* was defined which described a subset of the functionality defined in the main scenario. This mini-scenario defines the key interaction, which the first prototype should implement:

- The general public can upload information (primarily images) to the WeKnowIt Knowledge Base.
- This upload process should make use of the intelligence layers to enrich this information.
- This information can then be visualised by citizens or by members of the emergency response team.
- The emergency response team can choose to make key information available to the general public.

From this mini-scenario, a set of workflows was produced which illustrate how the user interacts with the system and how the system should combine and collate information internally. In parallel to this, the core system was implemented which consists of a number of services provided by the intelligence layers that are combined in a single composition layer. The first prototype demonstrator makes use of services from the Personal, Mass, Social and Media Intelligence layers, implementing intelligent functionalities such as a multimodal interface (WP1), information enrichment via tag normalization (WP2), tag suggestions through local tag detection (WP3), personalised access and support for multiple users and permissions management (WP4), and an interaction model for event and sub-event representation (WP1). Mobile and Web applications were then

implemented which interact with the core system via this composition layer.

Abbreviations and Acronyms

| | |
|---|---|
| **API** | Application Programming Interface |
| **CAP** | Community Administration Platform |
| **CDL** | Community Design Language |
| **CSG** | Consumer Study Group use case |
| **DTO** | Data Transfer Object |
| **ER** | Emergency Response use case |
| **FLO** | Forward Liason Officer |
| **FOAF** | Friend-of-a-Friend |
| **HTTP** | Hypertext Transfer Protocol |
| **JSON** | JavaScript Object Notation |
| **M3O** | Multimedia Metadata Ontology |
| **MPEG-7** | Multimedia Content Description Interface |
| **OSGi** | Open Services Gateway initiative |
| **POI** | Point of Interest |
| **POJO** | Plain Old Java Object |
| **RDF** | Resource Description Framework |
| **RDFa** | Resource Description Framework - in - attributes |
| **REST** | Representational State Transfer |
| **RSS** | Really Simple Syndication |
| **SIOC** | Semantically-Interlinked Online Communities Project |
| **SPARQL** | SPARQL Protocol and RDF Query Language |
| **SSL** | Secure Sockets Layer |
| **UI** | User Interface |
| **URI** | Unique Resource Identifier |
| **URL** | Unique Resource Locator |
| **W3C** | World Wide Web Consortium |
| **WKI** | The WeKnowIt project |
| **WKI DS** | The WeKnowIt Data Storage |
| **WP** | Work package |
| **WS** | Web Services |
| **XML** | Extensible Markup Language |

# Table of Contents

# List of Figures

# 1. Introduction

This deliverable describes the implementation of the first prototype of the Emergency Response (ER) demonstrator, according to section B.1.3.6 Work package descriptions of WeKnowIt Annex I – "Description of Work". Deliverable D7.3.1 – "Initial emergency response case study implementation" is meant to report the natural outcome from Task T7.2.2 – "Development" where the ER prototype is implemented. A second version of this deliverable is planned for the future - Deliverable D7.3.2 – "Emergency Response case study implementation", where the final implementation will be reported.

The first prototype demonstrator for the WKI Emergency Response Scenario has four primary aims. It should enable individuals to upload information to the WKI system about the emergency incident; this information should additionally be enriched through the action of the intelligence layers present in WKI. This information should then be presented to citizens and ER personnel allowing them to understand the incident and make improved decisions on the basis of this information. Additionally it should allow the ER personnel to make selected information available to the general public. All above functionalities should be supported by the WKI platform and architecture developed within WP6.

The process of developing the WKI ER first prototype demonstrator is shown below:

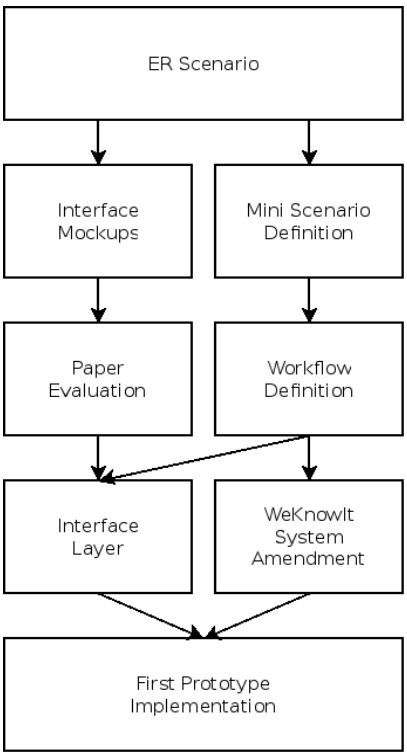


**Figure 1: Prototype 1 Development Process**

The development processed started from the definition of the main ER scenario described in D7.1. From this main scenario, interface mock-up designs were designed and these were evaluated with a small number of target users. In coordination with this process, a subset of the ER scenario was chosen as a *mini-scenario* that defined the functionality of the first prototype of the ER demonstrator. Following the specification of the mini scenario workflows were defined which illustrated how the users of the system interacted with the various services that are present in the core WKI system, and have been described in D6.1.2 [3.8] and D6.4.1 [3.9]. Finally, a web application and an example mobile application were developed to allow users to access and interact with the system.

## 1.1. Focus and Aims of the First Prototype

The first prototype of the ER demonstrator is therefore focused around the *mini-scenario* generated by WKI as a subset of the larger ER scenario described in deliverable D7.1.

The first prototype demonstrates how the intelligence layers present within the WKI system contribute to Collective Intelligence processing which allows the ER system to function effectively. The aims of the first prototype are, therefore, threefold:

1. To validate the system architecture developed for WKI within WP6 and its effectiveness at providing a framework for integrating the multiple layers of intelligence.
2. To provide a framework for evaluating whether the Collective Intelligence processing implemented by WKI improves the quality of information passed from members of the general public to ER staff and vice versa
3. To allow the generation and specification of new requirements for the ER demonstrator as part of the user-centred design cycle.

The user-centred design lifecycle is a standard cyclical methodology [3] for developing systems with the requirements of the user at the centre of the process. It is shown below:

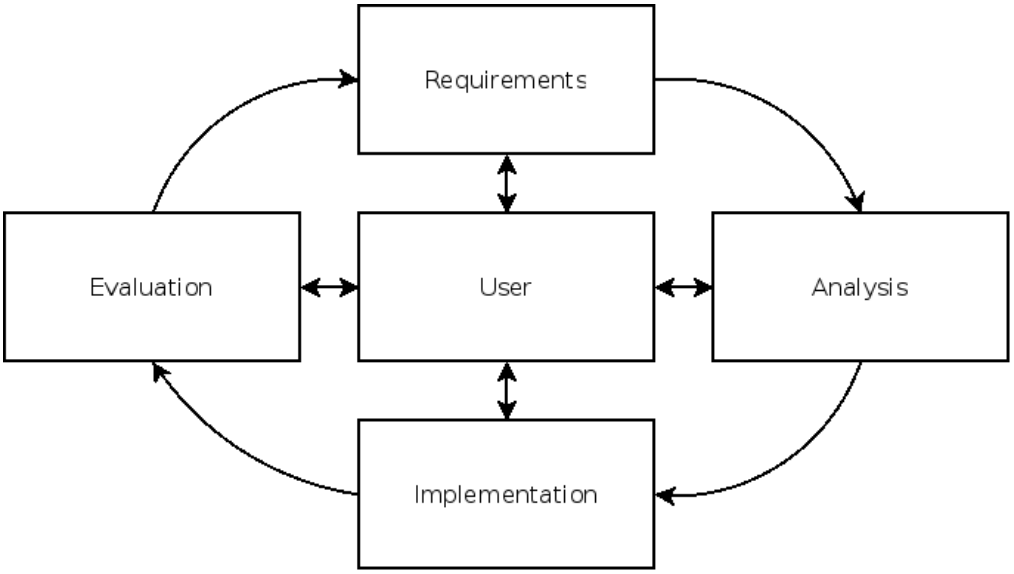


**Figure 2: User Centred Design Methodology**

Thus, each stage of the user-centred design process involves both progression of development and validation with target users. Previous work has defined the requirements for the ER demonstrator. Briefly stated the requirements for the WKI system are:

- The system should have a multimodal interface.
- The system should support the upload of information.
- The system should enrich any information uploaded by exploiting the Intelligent Layers and Collective Intelligence services.
- The system should expose an interface which allows the user to search and browse for information.
- The system should support personalised access to the information contained within it.
- The system should provide support for multiple users and visualisations.
- The system should prioritise the communication flow within the system.
- The system should support recommendations making them available to citizens and ER personnel.
- The system should support task management.
- The system should allow users to rate information.

These requirements and the processes that defined them were described in detail in D7.2. These requirements can be considered to be generic requirements for the WKI system and as such the first prototype is designed to meet a subset of these requirements. The implementation of the prototype and subsequent evaluations will be used to focus and refine these initial requirements prior to the development of the second prototype.

This document describes the development and implementation of the first prototype of the ER demonstrator for the WKI project. It begins by

outlining how the mini-scenario was defined and gives an overview of the mini-scenario. Following this, the overall architecture used to implement the system is described. The next section details the mock-up interface designs that were produced for the overall system, which is then followed by UML descriptions of the workflows that were developed for the first prototype demonstrator. The implemented interfaces are then described and the data that defines each interface is given. Finally the requirements described in D7.2 are revisited in order to assess how the first prototype has met these requirements.

# 2. Scenario for the first ER demonstrator

The full ER scenario was presented in Deliverable D7.1. This scenario describes how the full WKI system should function in a typical emergency situation. To enable the development of the first prototype a mini-scenario was developed. This mini-scenario defines a subset of the functionality and ability of the overall ER demonstrator, by enabling functionalities from different layers of Intelligence, thus building towards Collective Intelligence.

## *2.1. Mini-Scenario Definition*

The mini scenario defines a characteristic series of events that involves the use of the first prototype of the ER demonstrator. Although the mini-scenario is centred on a flooding event, the types of actions and use of the demonstrator could easily be applied to any other type of incident and of any level of severity.

The mini-scenario describes the key use of the ER demonstrator. The ER demonstrator enables citizens and ER staff to upload information using an intelligent interface based on a Personal Intelligence interaction model. The ER personnel are then able to view the uploaded information, which should assist them in making effective decisions as to how to deal with the emergency. In uploading and viewing the information, the users are assisted by intelligent media analysis techniques that perform tag normalization and enable provide tag suggesting mechanisms through Mass Intelligence dynamic tag analysis. They are also able to make selected information available to the general public, by making use of the capabilities of the Social Intelligence Community Administration Platform. The scenario describes how these events transpire and how the different actions of the individuals involved interrelate. Here the mini-scenario is split into four stages describing each individual's interaction with WKI as a flooding incident occurs.

### 2.1.1. Stage 1: The initial phase of the incident

*A call from the Fire Service is received by the Emergency Response office in the Sheffield City Council, warning about a potential flood emergency. John, an experienced member of the ER staff, is taking the call and, as he thinks it may evolve to be a major emergency, he puts resources on standby. After a few minutes John receives two more calls, one by the Police Chief and one by the Ambulance Services, both describing the floods. At this point, John decides to activate resources, as it is a major emergency. He decides to activate the team and sets up a Gold Team to strategically manage the emergency:*

- *Establish a gold strategy for managing the event*
- *Communication with the media*

- *Communication with other agencies*
- *Prioritise requests*
- *Prepare a long-term recovery strategy*

*Other command structures are set up. A Bronze command centre is the operational command structure that will act hands on for managing events A tactical level (Silver) may be set up to manage the event from an incident control point established in the vicinity of an event. John will be part of the Silver strategy team. Lucy will be responsible for the Gold team. Andrea is sent by Lucy to the disaster scene as part of the Bronze Team.*

This step of the mini-scenario is largely concerned with the management of the ER staff. This type of group management will be handled by Organizational Intelligence services and will be implemented in the second prototype of the ER demonstrator. This step illustrates the background to the mini-scenario.

### 2.1.2. Uploading information to WKI

*Mark, a Sheffield citizen, is still at work and getting ready to go home. When exiting the office, he realises the road is flooded and it may be very hard to reach the parking.*

*Mark calls the emergency service, he is asked to describe the situation. The emergency service asks Mark, whether his mobile phone is equipped with a digital still camera and whether he can take a picture of the scene and upload it to WKI. While Mark is talking to the emergency service, inputs his name into the "New user" form in the system. A message is sent to Mark with the invitation to WKI containing login details.*

*After Mark has taken a picture of the road, he uploads the photo using the web interface. Mark also quickly tags the picture by entering the word "flood". Immediately the image is analysed by the intelligent system and is categorised as "flooding". As someone else has already added a picture tagged as flooding from a nearby geographic location the system automatically clusters them together.*

This stage of the scenario describes the process of uploading information to WKI from the perspective of a citizen. Initially, the emergency services call handler creates an account for Mark within WKI – this process then sends Mark his credentials for accessing WKI over his mobile device.

Mark then takes a picture of the flooding and uploads the photo. The contents of the photo is analysed and suggested tags are derived from similar images using Media Intelligence services. Mark is then able to tag the image. The tags are dynamically analysed by Mass Intelligence services in order to suggest new tags that could be applied to the information.

Finally, the information is uploaded to WKI and Mass Intelligence services could be used to cluster and categorise the image.

### 2.1.3. Information Processing Stage

*When John logs in (from his Desktop PC) he gets a screen with an overview of the available information. When opening the main page John views the information using a geographical visualisation. John can see the location of the uploaded photos. This helps John in having a better overview of the situation.*

*John sends Andrea, A Forward Liaison Officer (FLO), to the scene of incident to take pictures When Andrea arrives on the scene of the incident she starts taking pictures and uploads them to E-WeKnowIt.*

This stage of the mini-scenario illustrates how WKI is used by ER staff to assist in making resource decisions. John is able to visualise the information geographically and this information enables him to make the decision to send an FLO to the scene of the incident.

The upload process for the FLO is the same as that for the citizen except that since the system knows that the FLO is a member of the emergency services their information is treated differently internally. However, the actual process of uploading and tagging remains the same.

### 2.1.4. Informing the General Public Stage

*John starts thinking about making some of the information public so normal users of the Sheffield City Council website can see the current situation and know which the most affected areas are. He then approves some of the images for public visibility. John asks for an analysis and an overview of what was happening in a specified period of the incident. The metadata on the phone calls received by the city council as well as their content are taken into account.*

This stage of the mini-scenario illustrates how WKI is used to pass information between the Emergency Response team and the general public. John, being a member of the ER team, has sufficient access rights to alter the dissemination level of information and is able to make selected pieces of information available to the general public.

### 2.1.5. Recovery Phase

*Mark is at home and wants to review all the information that he captured during the emergency and submit new additional photos that he did not submit yesterday. Mark has also been sent by friends some pictures about the emergency and wants to add them. Mark logs into E-WeKnowIt using his Desktop PC. He has already a login to the site so the system immediately presents him his home page. Mark chooses to upload the new content. When the content is uploaded, Mark decides to tag all his photos with the words "flood, river, road, water". He also adds a short comment of how he got involved in the floods and what he saw.*

*In his office, John is dealing with the recovery phase. He still has to monitor the incidents but also start thinking to how the whole process can*

*be managed. John receives a notification via mail that new content has been added to WKI so he logs in and browses the new content.*

This phase of the mini-scenario occurs after the incident and illustrates how information can be recovered from WKI post-hoc. The processing steps involved in the recovery stage are, however, the same as those described previously.

# 3. Prototype Architecture

The first prototype architecture is consistent with the WKI architecture defined in terms of the Integration workpackage (WP6) and is thoroughly described in D6.1.2 [3.8]. In general, it comprises four different logical layers. This approach was chosen in order to separate the intelligence layers and knowledge store present in the WKI system from the user interface components. Because there is no direct connection between the user interface and the underlying services both are free to change their implementation without having an effect on the other.

The WKI core system consists of a number of OSGi services running over a Fuse ESB. This contains the services from each intelligence layer in addition to the data storage and knowledge base modules. The composition layer is then used to mediate between the core system and the user interface layer. This allows the user interface to make single requests to the composition layer that are then translated by this layer into multiple service requests. On top of this layer is the translation layer, which converts the POJOs that are generated by the core services into a format, which is more amenable to the user interface layer. Finally, the user interface layer presents information to the user and allows the user to interact with the system via the different architectural layers. The architecture is described in detail below.

## *3.1. WKI System Architecture*

The architecture of the WKI System was prepared by WP6 and described in D6.1.2 [3.8]. This section provides information that is necessary to understand the integration of the use case application with the WKI System.

The WKI System has a layered architecture, as presented on Figure 3. The communication paths are strictly defined, which means that the communication is allowed only between the direct neighbours.

The colours of components on Figure 3 mark the division between the *use-case applications* (green colour), and the *WKI System* (blue colour). Yellow vertical arrows represent messages exchanged between the WKI System and the UI components.

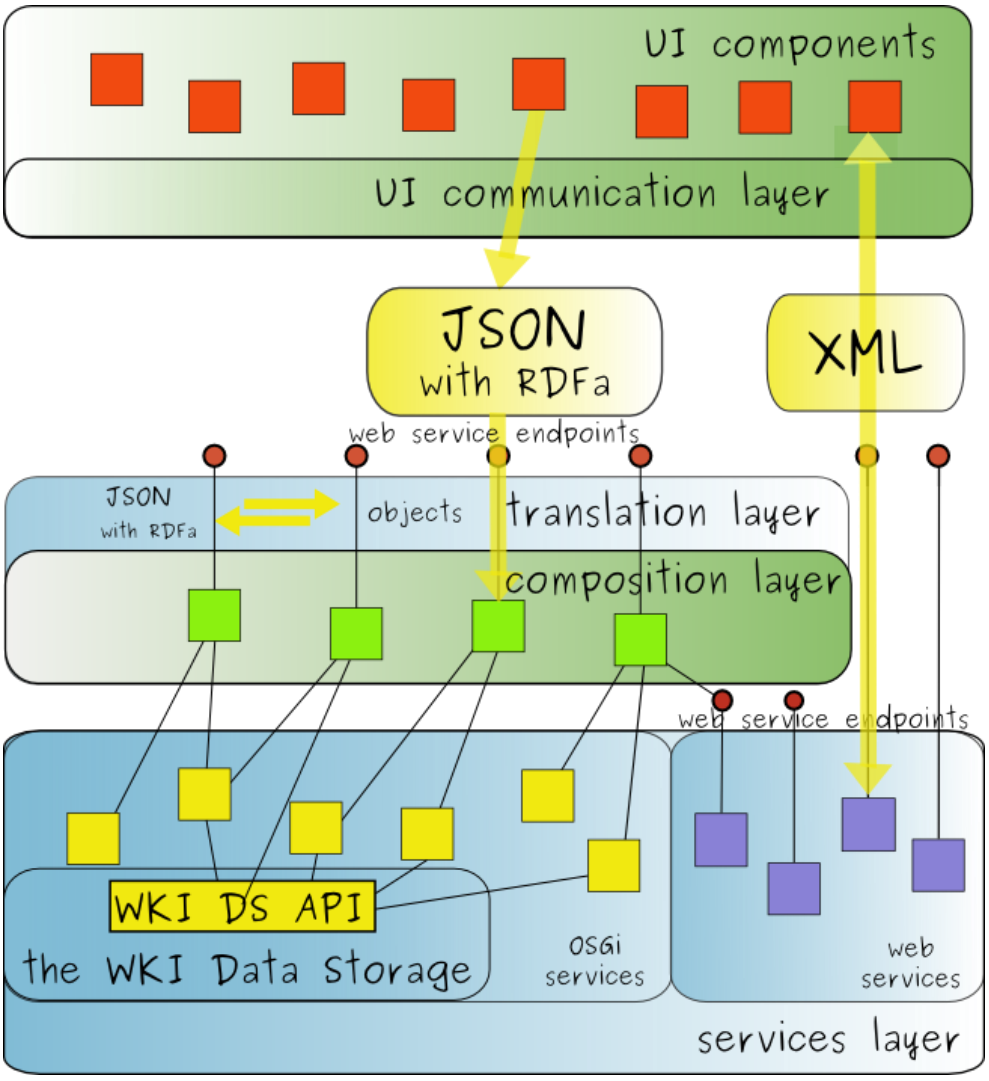


**Figure 3 UI components integration - an overview**

### 3.1.1. The use-case applications

The use case applications consist of:

1. UI components (represented as red rectangles on Figure 3), which are responsible for presentation of data. They can also include some business logic (even thought the real logic - especially combining of data from many services – is handled by the composition layer).
2. UI communication layer, which is responsible for making calls and receiving responses from the layers underneath (i.e. composition layer).
3. The composition layer, which acts as an intermediary between the UI components and the services of the WKI System. The composition layer is described in detail below.

It is worth noticing, that UI components and UI communication layer live outside the OSGi environment provided by the WKI System, while the composition layer lives within this environment.

### 3.1.2. Messages

Messages between the WKI System and the UI components are exchanged via the HTTP protocol.

UI components of the ER use case rely on the messages in RDFa/JSON[1] format. UI components of the CSG use case use different types of messages. Some services of CSG use case return XML, which is consumed directly, without any transformation, by the UI components. Such messages do not go through nor the translation layer, neither the composition layer. The rest of the CSG use case services, as for example the login mechanism, require the mediation of the translation and composition layers.

The translation layer (part of the WKI System) is responsible for the translation of messages format.

### 3.1.3. The WKI System

The WKI System consists of:

- WS endpoints (represented as red dots on Figure 3),
- The Translation layer,
- The Services layer.

### 3.1.4. WS endpoints

WS endpoints can expose:

- Compositions of services,
- Single services.

The first option is preferred, as it takes the burden of combining outputs derived from many services from the UI components.

### 3.1.5. The translation layer

The translation layer is responsible for translating messages between formats used by the WKI System and by the UI components of ER use case. This layer is capable of translation of Java POJOs to JSON + RDFa (and vice versa). This layer is still under development and it is not yet fully decided, if it is possible to be completely use-case agnostic, or if it has to be somehow configured to fit the requirements of a specific use case. In Figure 3 this layer is placed within the WKI System, as ideally it should be independent of the use case application.

[1] http://json.org/

### 3.1.6. The WKI Services

The services (yellow and violet squares) on the bottom of Figure 3 represent the low-level, atomic services provided by WPs, as described in D6.4.1, section 2. Some of the services live in the OSGi environment, while others are accessible via Web Services and are outside the OSGi environment of the WKI System.

Among the services, one is singled out - it is the WKI DS service. Technically, it is just one of many services, but it plays a central role in the system. The use-case application can contact with the WKI Data Storage only by using other services - the interface of the WKI Data Storage is not exposed via Web Services.

## *3.2. Composition Layer*

Technically, the Composition Layer consists of three OSGi bundles that are deployed to the OSGi environment provided by the WKI System:

- wp7-compositionlayer-commons – contains common functionalities used by both use-case applications,
- wp7-compositionlayer-csg – contains CSG-specific functionalities; not used by the ER first prototype,
- wp7-compositionlayer-er – contains ER-specific functionalities; this bundle is the most important for the ER use-case, as it contains the business logic (workflows) required by the ER use-case application.

The Composition Layer is accessible via Web Services. This is realised by the Jetty component[2] available in the WKI System. The Composition Layer returns data that is later translated by Translation Layer so the components of the User Interface Layer can use them without further transformations.

The Composition Layer passes calls received from the UI layer to the services of the WeKnowIt System. At the moment, direct method calls are used via the OSGi registry. This is possible since all the services (of the Service Layer and of the Composition Layer) reside in the same OSGi environment provided by the WKI System.

### 3.2.1. wp7-compositionlayer-commons

The idea behind this module is to group common functionalities required by both use-case applications. At the moment, as the development of both applications is in the early stage and happens very rapidly, no such functionality has been isolated yet[3]. After the first prototype is finished, the functionalities of both use-case applications will be reviewed and this

[2] Jetty WebServer, http://www.mortbay.org/jetty/

[3] However, some common functionality – e.g. login – has been identified.

module will be extended with common functionality in order to reduce the maintenance cost of duplicated code.

Actually, the wp7-compositionlayer-common module contains the following classes:

- CLUtils - utility class,
- CLException – common exception thrown by methods of the Composition layer,
- InputStreamProvider – implementation of the javax.ws.rs.ext.MessageBodyReader, that is necessary to return input streams via the REST services.

### 3.2.2. wp7-compositionlayer-er

This module contains the following packages:

| Package | Description |
|---|---|
| eu.weknowit.composition .layer.er | API of the ER composition layer - interfaces of services. |
| eu.weknowit.composition .layer.er.dto | DTOs of composition layer. |
| eu.weknowit.composition .layer.er.impl | Implementation of ER composition layer services. |
| eu.weknowit.composition .layer.er.impl.mocks | Mocks for not-yet available functionalities. |
| eu.weknowit.composition .layer.er.providers | Translation layer - contains providers for RDFa+JSON/POJO translation. |

**Table 1 Packages of the Composition Layer**

The API of the Composition layer is included in the eu.weknowit.composition.layer.er package, and the core functionality is grouped in two packages – eu.weknowit.composition.layer.er.dto and eu.weknowit.composition.layer.er.impl.

### 3.2.3. Services of the Composition Layer

The following services are available and exposed as the API of the Composition Layer via Web Services. Most of them are aimed at processing of a particular DTO. ICLSearchService differs in this aspect, as it provides an "entry" to the WKI DS allowing for execution of various search queries.

| Service | Description |
|---|---|
| ICLDocumentService | This interface declares all methods required to add/update/remove/retrieve documents (CLDocument). |

| | |
|---|---|
| ICLEventService | This interface declares all methods required to add/update/retrieve event (CLEvent) including functionalities like "return all events that are related to this one". |
| ICLResourceService | Service provides methods to operate on resources (CLResource). |
| ICLSearchService | This service provides specialized methods for search in the WKI Data Storage. It employs the service WP6_IWKIDataStorage which was implemented within WP6 and is described in D6.4.1 [9] |
| ICLTagService | Interface contains method for processing tags (CLTag). It employs intelligent services that were developed in WP2 and WP3, i.e. WP2_ITagProcessingService, WP2_ITagNormalization and WP3_LocalTagCommunityDetector. |
| ICLUserService | Service provides operations connected with users (CLUser), like addition of friends, profile update etc. It employs intelligent services that were developed in WP1, i.e. WP1_ILoginService, and WP1_AccountManager, and in WP4, i.e. WP4_AuthCheck |

**Table 2: Services of the Composition Layer**

The methods of the services are prepared in such way, that the UI components can make direct calls to receive the data required by a particular element of the user interface. For example, the home page of the WKI System displays latest public events. Such events can be retrieved from the WKI DataStorage with one request - via getLatestPublicEvents method of ICLEventService. Thus, one of the requirements of the Composition Layer (to make UI layer free of the business logic related to manipulation of data retrieved from different services) is fulfilled.

Services of the Composition Layer are annotated with JAX-RS annotations provided by Apache CXF project[4]. The annotations are used to make the services available under specified URLs, and specify types (e.g. POST or GET) and formats of requests and response.

The intelligent services that are employed by the above Composition Layer services are presented in the table below.

[4] Apache CXF, http://cxf.apache.org

| Service name | Description |
|---|---|
| WP1_ILoginService | The login service is the main entry point to the rest of the services in WKI, provided a user has already created an account. It allows users to authenticate via a username and password or, preferrably, through the use of the OpenId standard. |
| WP1_AccountManager | This service allows a new user to register in E-WKI. Once the user has an account, he is able to access the other services provided by WKI. This is centred on the OpenId standard and the concept of a federated identity. A federated identity is one which spans multiple information systems. |
| WP2_TagProcessingService | The list of suggested tags for the specific image are paired with known landmarks close by the location the image was taken (as listed in the Geonames list), or discarded. The list of tags that persist through the pairing process, are replaced with a more formal form (capitalized where appropriate, etc) and returned to the system/user. |
| WP2_TagNormalization | The list of tags that have been assigned to each resource are matched to one or more domain ontology concepts. Each matching is accompanied by a weight coefficient that shows the degree of the tag-concept relation. |
| WP3_LocalTagCommunityDetector | Given some input tag, the goal of this service is to identify a collection of tags that form a community around it. |

| | |
|---|---|
| WP4_AuthCheck | Calls the Community Administration Platform (CAP) via the structured language Community Design Language (CDL). Via CDL expressions the following commands can be executed: a. Define policies , b. Define access rights, c. Check access rights, d. Report about policies and access rights, e. Get help on CDL |
| WP6_IWKIDataStorage | This service provides API for storing/retrieving data from the WKI Data Storage. This API covers all components of the WKI Data Storage - triple store(s) and other databases, and file storages. |

**Table 3: Intelligent Services employed by the Composition Layer**

### 3.2.4. DTOs of the Composition Layer

The DTOs of the Composition Layer are strictly related to the set of ontologies used in the WeKnowIt project. This is a requirement of the UI components of ER use-case, which work with RDFa data format. Composition Layer, in cooperation with Translation Layer, delivers the data in the requested format.

| Types | Description |
|---|---|
| ICLComponent, CLAbstractResource, CLResource, CLSiocItem, CLSiocPost | Interfaces and abstract classes for all DTOs. |
| CLTag, CLTaggedResource | DTOs related to Tags. |
| CLDocument, CLFoafDocument | DTOs related to Documents. |
| CLEvent, CLEventNews, CLFullEvent, CLParticipation | DTOs related to Events. |
| CLSimpleUser, CLUser, CLUserDetails, CLUserPermission, CLAddress, CLOnlinePresence | DTOs related to Users. |
| CLGmlGeometry, CLGmlPoint, GeoCoordinates | DTOs related to geo localization. |

**Table 4 DTOs of the Composition Layer**

The DTOs of the Composition Layer are annotated with JAXB annotations. This results in automatic serialization (marshalling and unmarshalling) of

DTOs to/from XML during sending requests and receiving responses by Web Services.

## 3.3. Further work on the Composition Layer

The composition Layer is a work in progress and will be maintained and enhanced along with the WeKnowIt project. In the case of ER use-case application, the first prototype version offers some functionality although it will be expanded on by later development. The development of the Composition Layer is strictly connected to new functionalities required by UI layer, and at the same time to new services that will be added to the Service Layer.

It is expected, that as the fulfilment of the requirements of UI layer will gradually require CL to invoke methods of more WKI services, and combine the data they return, than it is now. In such case, the direct calls made by the CL layer via OSGi registry, will be replaced by Camel[5] flows, which are easier to maintain and offers some built-in additional functionalities (e.g. asynchronous calls).

Apart from addition of new functionality, some bookkeeping tasks will also be executed. The most important is to avoid duplication of services in ER and CSG modules of the Composition Layer. In order to achieve this, both modules will be frequently scanned in search of duplicated functions.

## 3.4. Translation Layer

The translation layer mediates between the composition layer and the user interface layer. The purpose of this layer is to transform the standard java objects that are used in the composition layer into the JSON+RDFa data that is shown at the user interface level.

The translation layer is responsible for translating messages between formats used by the WKI System and by the UI components of ER use case. This layer is capable of translation of Java POJOs to JSON + RDFa (and vice versa). This layer is under development and subject to the requirements of a specific use case.

## 3.5. User Interface Layer

The User Interface layer has been implemented using the Ruby on Rails[6] framework using the JRuby[7] interpreter. This framework enables the rapid development of the user interface framework and has built in support for simple access to REST [1] services. The foundation of the user interface is implemented as HTML and CSS pages, with each page corresponding to a

[5] Apache Camel, http://camel.apache.org/

[6] http://rubyonrails.org/

[7] http://jruby.org/

fundamental action within the user interface workflow (see below). On top of the basic framework, Javascript is used to enhance the user experience and, additionally, AJAX support allows the interface to connect to the core WKI services. JRuby was chosen as a language since it will allow for future versions of the interface to support more component based widgets, such as those developed within the JMaki[8] framework.

[8] https://ajax.dev.java.net/

# 4. Mock-up ER Interface Design

As with the workflows, the interface design used the mini-scenario as a foundation and for identifying the screens required for the first prototype. The interface design went through three phases. Firstly, an initial outline was developed early on for the purposes of demonstrating WP1 functionality. These initial designs were presented in D1.1.

Following from the initial designs and the specification of the Emergency Response scenario, these initial designs were developed into more detailed interface mock-ups. These mock-ups (presented below) were designed to display the full range of functionality that is intended to be present in the overall WKI ER prototype, so the mock-ups reflect a greater functionality than that which is described in the mini-scenario. This process is in accordance with the user-centred design methodology described above. This approach is that any problems with the overall use of terminology and interface design will be caught at this early stage. Thus the current and future interface design can be tailored to the user perception of how the interface should function before the full interface is implemented.

## *4.1. Mobile Interfaces*

### 4.1.1. WKI ER Home Page

The following screen is shown when the user accesses the ER website from their mobile phone and they have not previously logged in.

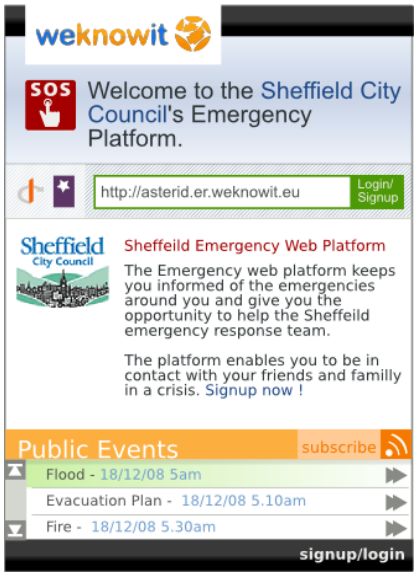


**Figure 4: Mobile WKI Home Page**

As the screen shows, the initial home page details the main purpose behind WKI and lists the most recent public events. This interface populates the event information from the WKI Knowledge Base.

Where public information about events is available they are shown in the event list and the user can select one of these events and see any public information about the event. The views will be the same as those detailed below except that the user will not be logged in and so will be unable to access any WKI functionality or see any information that they may have provided to this event.

The login option given is via AsterID, which is a semantic federation service implemented on top of OpenID. OpenID is a standard login protocol and allows a trusted website (here the AsterID site is used) to authorize and validate the user on behalf of WKI. The trusted website then vouches that the user has logged in correctly and a corresponding login occurs for WeKnowIt.

### 4.1.2. Login

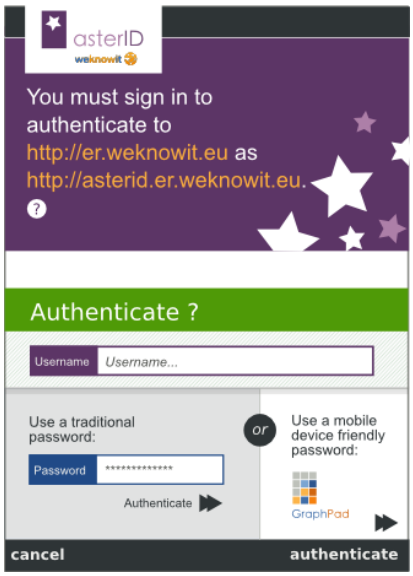


**Figure 5: AsterID login screen**

The AsterID login screen allows the citizen to specify their username in the standard way but gives the user two options for specifying their password details. They can use a standard password entry or select the GraphPad authentication option to use the graphical login screen.

Note that the AsterID server exists outside of the WKI architecture and is just used to provide authentication services for WeKnowIt applications. The system is similar to an OpenID service but includes the ability to store and process federated semantic identities where necessary. This functionality is not used directly within the first prototype implementation so AsterID acts as an OpenID authentication server.

The standard password can coexist with the graphical password which allows the user to have the same password but use either the entry box (which is more appropriate when logging in via a Desktop machine) or the GraphPad entry system (which is more appropriate when logging in via a Mobile device).

### 4.1.3. GraphPad Login


**Figure 6: GraphPad login screen**

The GraphPad login screen allows the user to specify their password in the form of a sequence of graphical images. The process uses a two-stage image selection mechanism that assists the user in recalling their password without comprising the security requirements of logging in.

The first stage contains high-level concepts, which can jog the users memory with regard to their specific image. The low-level images are shown once a high-level concept has been selected and the combination of the two images corresponds to a single alphanumeric character of the standard password. Thus, the user is able to have a single password for both interfaces and use them interchangeably.

### 4.1.4. Personalized Homepage

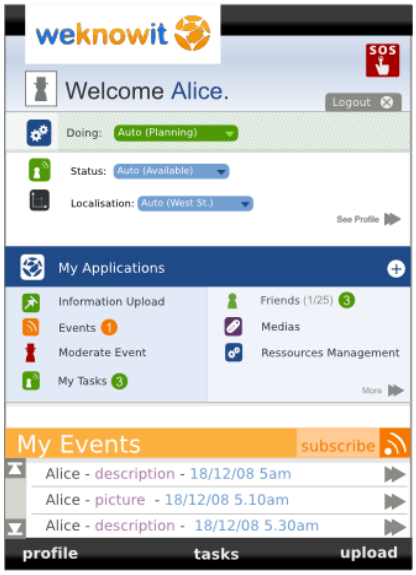


**Figure 7: Personalized Homepage Screens**

The personalized homepage screen is shown to the user upon logging in to WKI. Shown here are all the options available to all users – some of the ER professional specific options will not be shown when citizens access WKI.

The screen allows the user to perform the primary functions required by the mobile interface – uploading information, seeing the status of their friends by employing Social Intelligence tools and a history of their interaction with WKI. The initial interface design also allows the user to specify their current task, status and location. In addition, the interface allows the user to have more direct interaction with WKI through viewing events that the user has subscribed to and to have some level of control over the media that the user has uploaded to WKI.

### 4.1.5. Profile Data


**Figure 8: Profile Edit Screen**

The profile screen is accessed from the profile option on the homepage screen. This screen allows the user to amend or define their profile.

The user can define their address, postcode etc. This information is then stored in their profile and could make up supplemental data for supporting user profiling at a later date. For example, this information could be used to identify users that are potentially at risk from emergency incidents on the basis of the location information that has been provided to WKI.

### 4.1.6. Networks and Friends

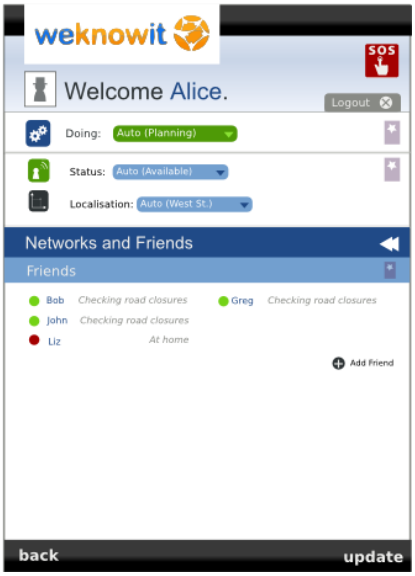


**Figure 9: Networks and Friends**

This screen has differing functions depending on whether it is accessed by citizens or ER professionals. For citizens it allows them to see which of their friends are online and get a short status update about each. For ER professionals it allows users to track what their colleagues are currently doing and allow them to open up dialogue with other members of the emergency services. This allows the coordination of actions amongst members of different emergency services. Such functionalities are supported by the Social Intelligence Community Administration Platform.

### 4.1.7. Upload Data

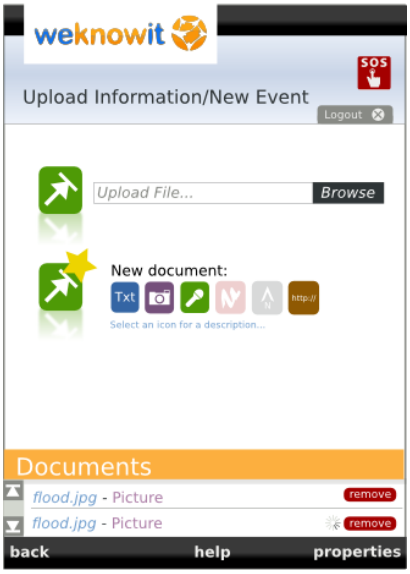


**Figure 10: Upload Information**

This screen is the first point in the process of uploading information to WKI. It allows the user to browse to a file stored on their device and upload that file to WKI. Thus, the upload process is the same whether the information that the user wishes to upload is an image, a piece of audio or a video recording.

The interface allows the user to explicitly specify what kind of information they wish to upload to WKI – specifically whether they want to upload some text, an image, a piece of audio, GPS routes, GPS coordinates or web links. In addition to this, it shows details about any information the user has previously uploaded.

### 4.1.8. Tag Upload Data

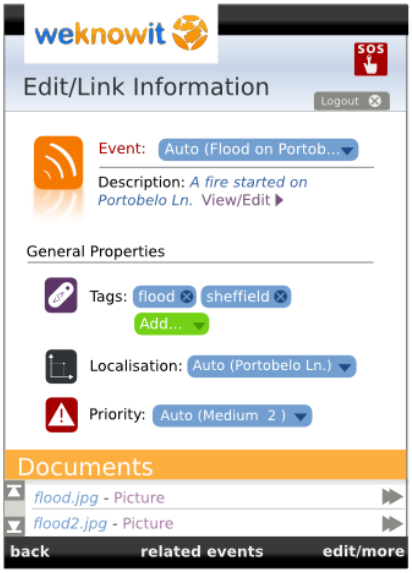


**Figure 11: Tagging Uploaded Data**

Once the user has provided their initial data, it can be linked to an event within the WKI Knowledge Store and the user is able to enrich the information associated with the initial upload. Specifically the user is able to add tags to the information and specify further information about the nature of the information.

The addition and specification of tags is an interactive process driven by the services provided by WP2 and WP3 that analyse any tags that have been applied to the data. These services generate tag suggestions on the basis of any tags that have already been applied to the data. Thus, the process seeks to maximise the amount of textual information that can be applied to the data before the full upload process is complete.

This screen also reports the localisation that has been registered for the information and allows the user to correct it if an incorrect location has been identified. The option for priority is applicable to ER personnel and allows them to register how information they upload should be treated.

### 4.1.9. Publish Upload Data

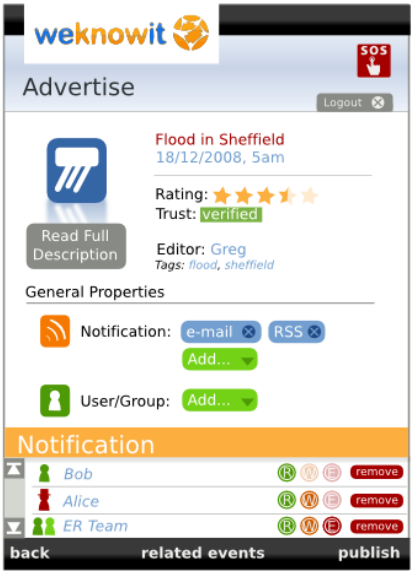


**Figure 12: Publishing Uploaded Data**

This screen is used to publish the upload data to WKI. This acts as an overview of all the information the user has provided to WKI and allows them to monitor how that information is used. For example, they are able to monitor this information by receiving emails or through and RSS feed.

This screen also acts as a point where the user is able to specify if they would like to receive notifications about information relating to the event. They can specify that they would like to receive updates about the event via email or using a standard RSS reader. In addition, the user is able to specify which other users or groups they wish to notify about the event using the User/Group options.

Furthermore, aided by the WP4 Community Administration Platform, the screen details which users will be notified about the information upload and what permissions are appropriate to each of these users. This allows the user to specify how their information can and should be processed upon being published to WKI.

### 4.1.10. Event – Geographical View

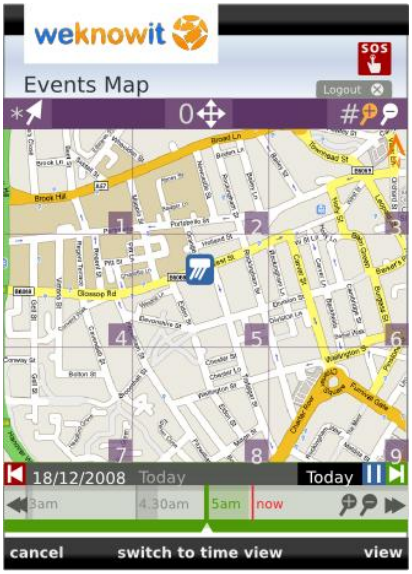


**Figure 13: Geographical Event View**

The geographical view shows event information using a geographical projection. The map is overlaid with keys corresponding to a phones numerical keypad that allows users to navigate and zoom the map if their device is not equipped with a touch screen display. In addition to the map display, the interface shows the temporal extent of the event on the timeline.

The Event map also indicates the location of information that has been provided to WKI in the form of icons on the map. The user is then able to see the uploaded to WKI by selecting the icon from the map. This allows the user to get a rapid overview of the location of the event and the location where the information is arriving from and then drill down to get more specific information.

### 4.1.11. Event – Detailed View

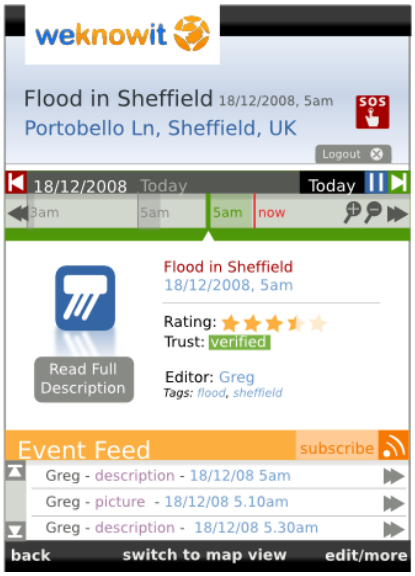


**Figure 14: Detailed event view**

The detailed view shows what the user sees when they select a specific piece of information from the map display. The detailed view gives them the full image display (here shown as an icon), information about the event to which this piece of information is attached and specific details about the uploaded information.

The user is able to view the tags that have been applied to the information along with the name of the user that uploaded the information. Additionally, for ER professionals, they are able to view the trust level applied to the information and the rating of the information. This functionality is provided by the WP4 Community Administration Platform. The trust level describes to what degree the ER professional should rely on the information. Thus information that has been provided by a reputable source (such as an FLO or other member of the emergency services) will be associated with a high level of trust whereas information provided by the general public may be associated with a lower level of trust until the nature of the incident can be verified.

In addition to the specific information, this screen also displays the feed of relevant information that the user has subscribed to. This allows the user to monitor their events from the mobile interface and keeps them up to date with any extra information that has been added to the event whilst they are in the field.

### 4.1.12. Event – Full Description


**Figure 15: Full Event Description**

This shows the screen that the user sees once they have selected to view the full event information from the previous display. The interface allows the ER staff to provide a lengthy description of the incident that can then be shown to citizens to inform them of the status of the incident and any ramifications that the incident my cause for the user.

In addition to the overall event view, the user is able to access any related sub-events and view any relevant comments from other users.

## 4.2. Desktop Interfaces

### 4.2.1. WKI Home Page

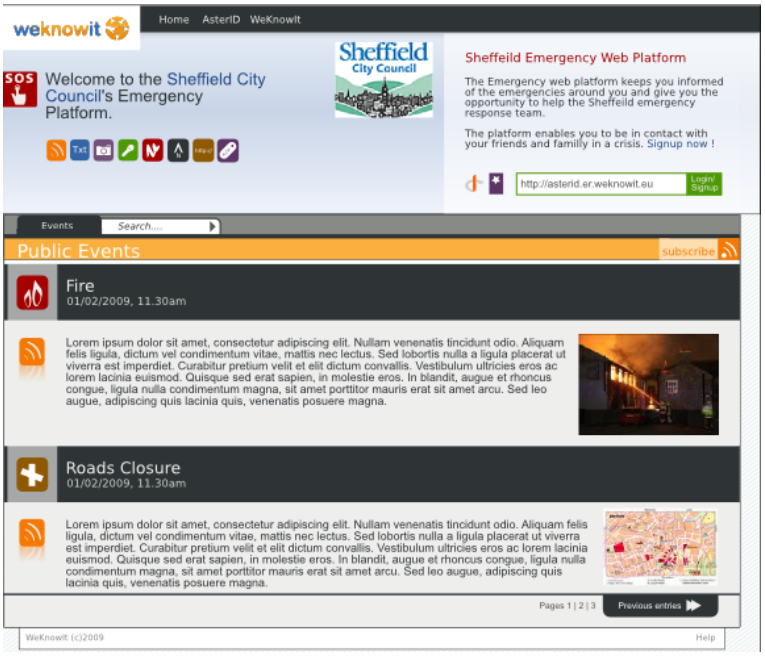


**Figure 16: WKI Home Page**

This screen shows the home page that the user sees when initially accessing the site using the desktop interface. This display is similar to the mobile interface in that it shows relevant information about incidents occurring within the city. It also allows the user to login to the main WKI site using the AsterID login procedure. Additionally the user can subscribe to the feeds associated with the public events in order to be informed of any new information that has been made available to the public.

### 4.2.2. Login

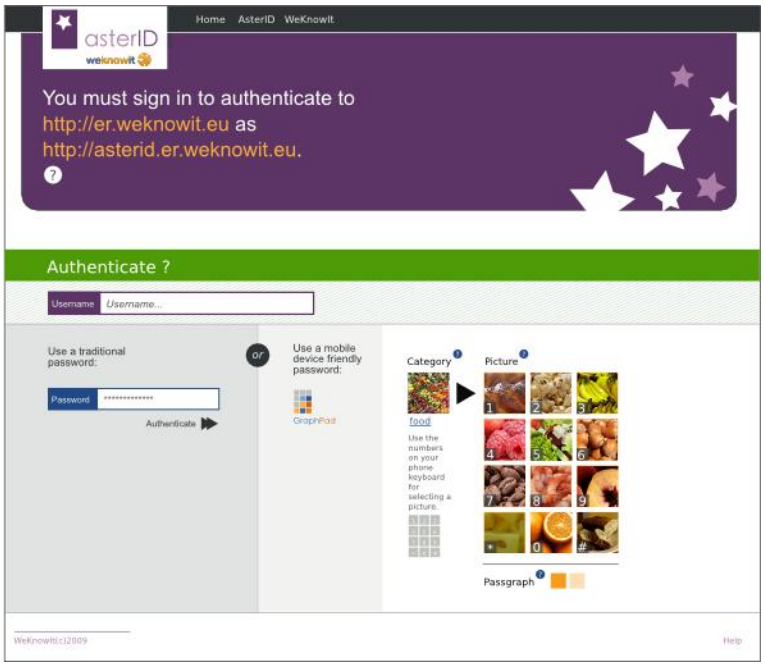


**Figure 17: Desktop Login Screen**

The AsterID login screen for the desktop modality is similar to the login screen shown for the mobile device. Again, it allows the user to log in using a standard username and password or by using the graphical log in system. This gives the user the flexibility of logging in using the keyboard or solely using the mouse.

### 4.2.3. Personalized Homepage

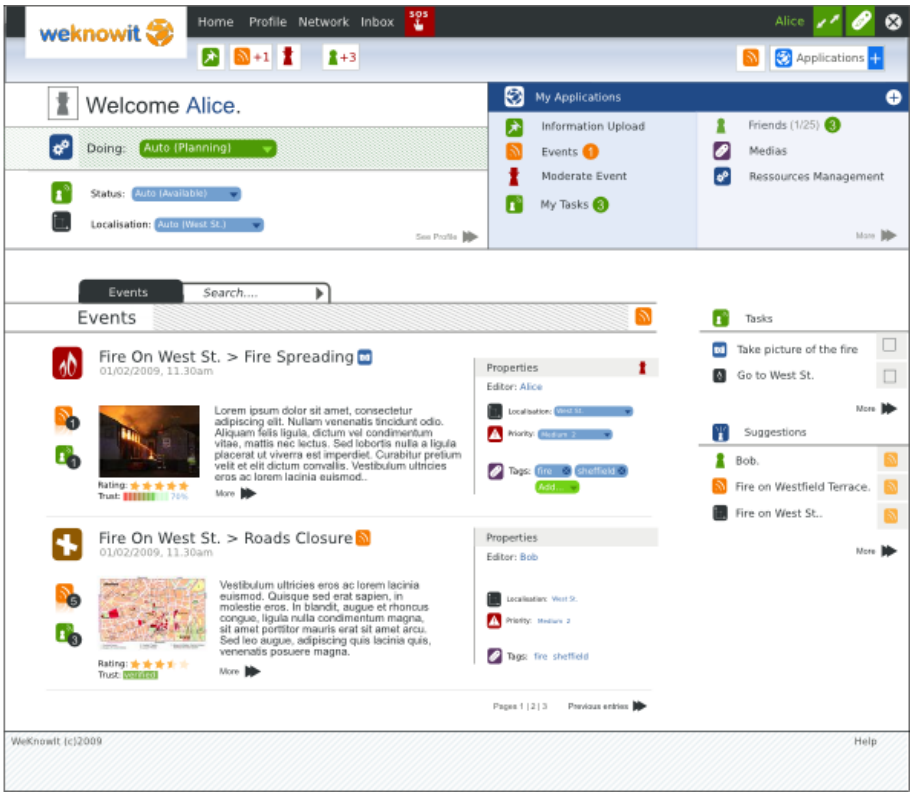


**Figure 18: Personalized Homepage**

The personalized homepage for the Desktop, again, is similar to the same interface provided for the mobile device. It gives the user access to typical tasks they might undertake with WeKnowIt. The user can access their applications (for uploading information, viewing their event subscriptions etc.) as well as set their current status and location.

The user is also able to track events that they have subscribed to and see any new information associated with those events.

As with the mobile interface, the homepage has extra affordances if the user is a member of the emergency services. Such users can update the tags, priority and location information attached to each event.

### 4.2.4. Profile Data

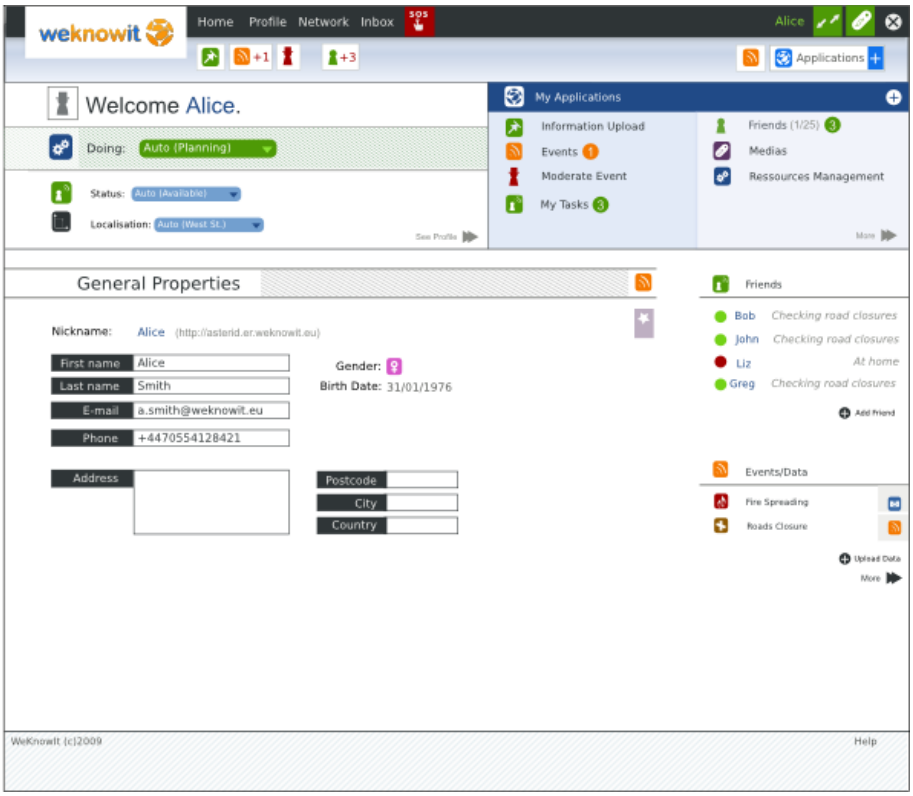


**Figure 19: Profile Screen**

The overall profile screen is, again, similar to the one seen for the mobile interface and allows the user to set or edit their personal details.

### 4.2.5. Upload Data

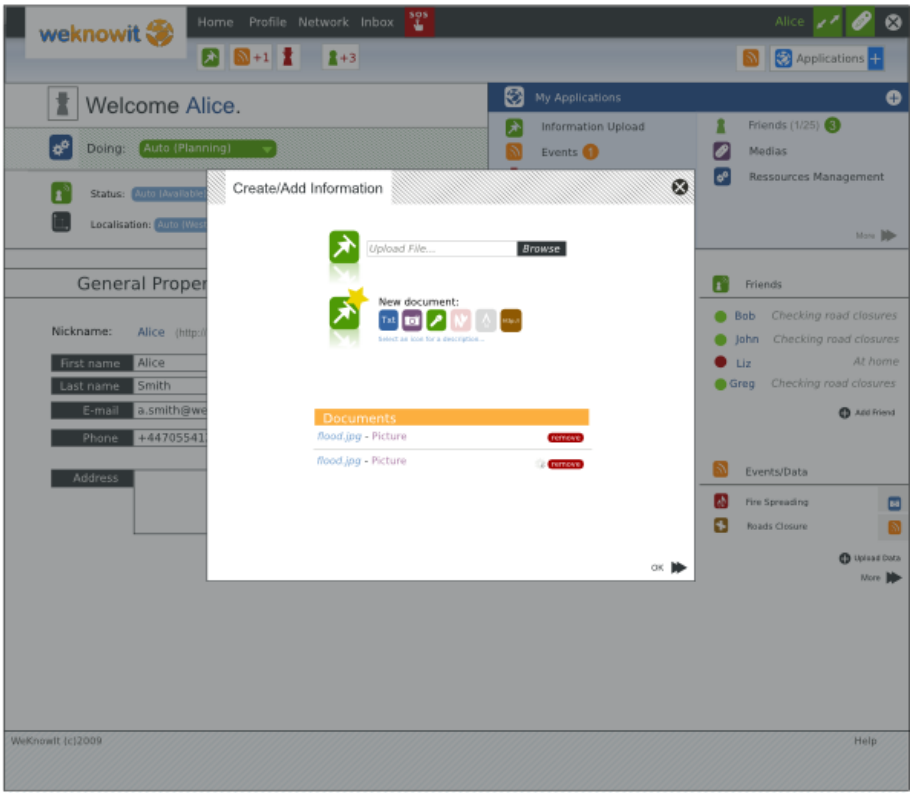


**Figure 20: Desktop upload interface**

This screen allows the user to upload information from their desktop. The procedure and interface used are the same as that for the mobile case – the mobile screens are shown as an overlay over the current desktop screen. This ensures that the upload process is consistent for both desktop and mobile users and makes the implementation of the upload process more efficient.

### 4.2.6. Event – Geographical View

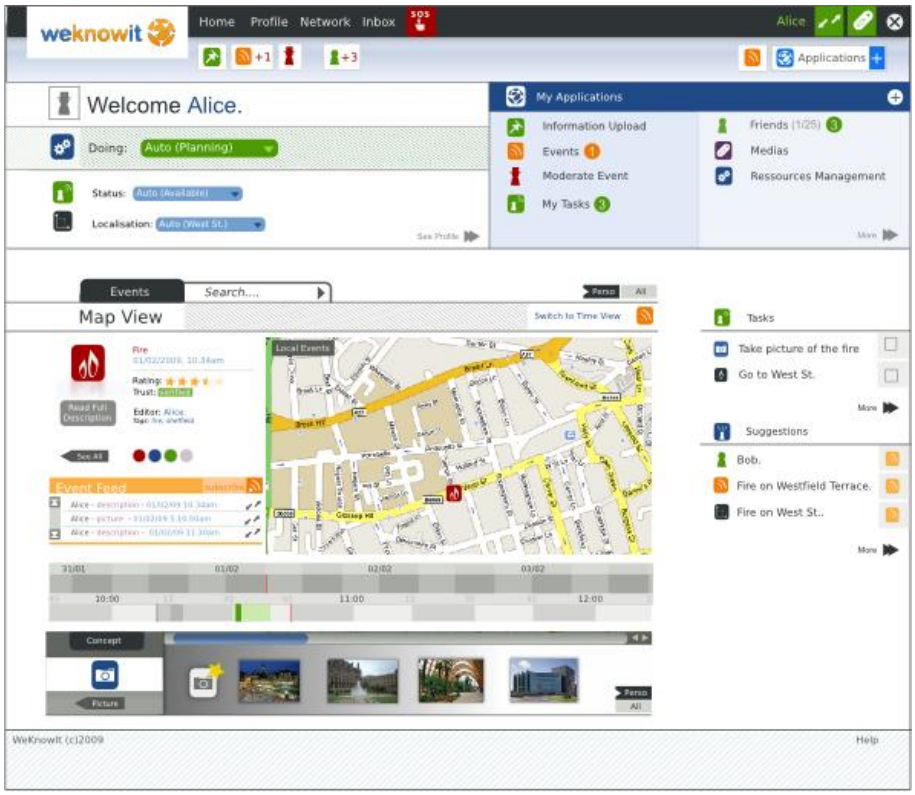


**Figure 21: Geographical event overview**

The geographical event overview screen shows the information relating to an incident using a geographical projection. The screen retains the layout and overall structure presented to the user in the home page ensuring that the user has access to their applications and can alter their status or location information from a single interface.

The information relating to the incident is shown directly on the map and the system also projects the same information onto a timeline. The interface also shows images in the carrousel at the bottom of the screen. This carousel is used to show images that have been uploaded to the system but have not been localised. The user is able to drag these images onto the map in order to associate them with the current event and provide basic localisation information.

The user is free to use the standard map interaction techniques to explore the map display using the typical interaction styles including panning and zooming etc.

### 4.2.7. Event – Detailed Incident View

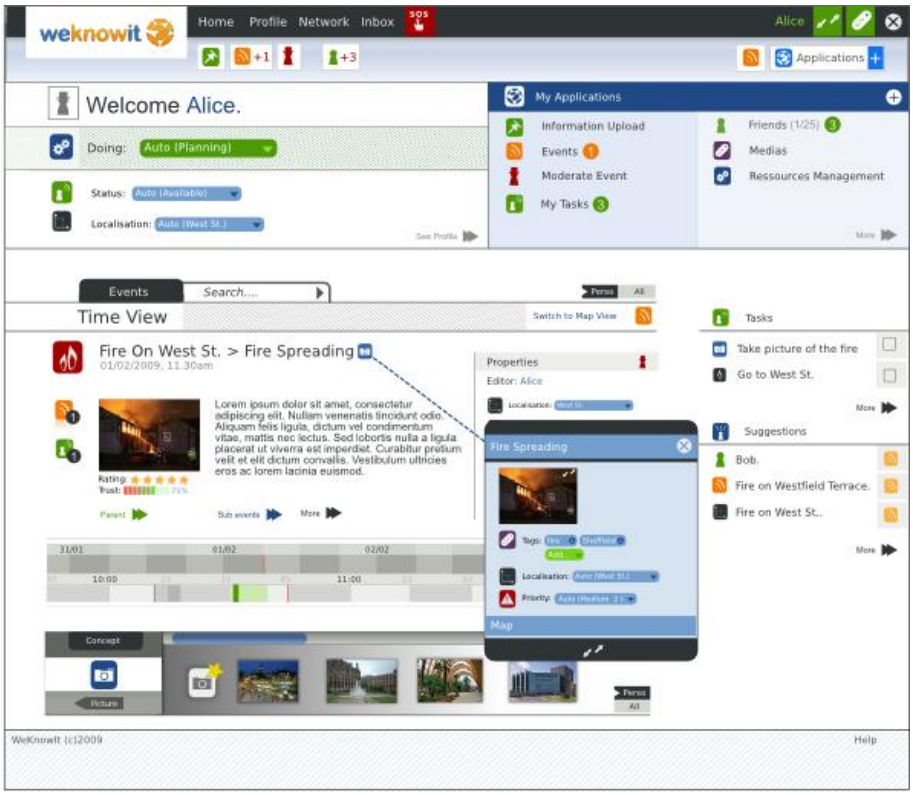


**Figure 22: Temporal incident view**

The temporal view of the incident shows more information about the event. Similarly to the mobile interface, the desktop interface gives detailed textual information about the status of the incident and can be used to provide citizens with relevant information. It also enables the WP2 and WP3 intelligent tagging and tag recommendation services.

# 5. Mini-Scenario Workflows

The purpose of the workflows defined for the first prototype are to explain how data is passed between the core WKI services and the various users that interact with the system in the Emergency Response Scenario. The workflows were defined by examining each step of the Mini-Scenario and identifying for each where interaction with WKI took place and what the informational requirements for each step were. To complete the specification of the workflow, the interactions between the services and the users as well as the temporal ordering of these interactions were defined.

The workflows are illustrated with reference to the mini-scenario outlined before. Whilst the definition of these workflows has been driven by the Emergency Response scenario, they are largely a generic composition of services and can apply to any WKI application. For example, the process of logging in to WKI will be similar across WKI applications.

## *5.1. Creating a New User Workflow*

*The emergency service asks Mark, whether his mobile phone is equipped with a digital still camera and whether he can take a picture of the scene and upload it to WKI. While Mark is talking the emergency service inputs his name into the "New user" form in the system.*

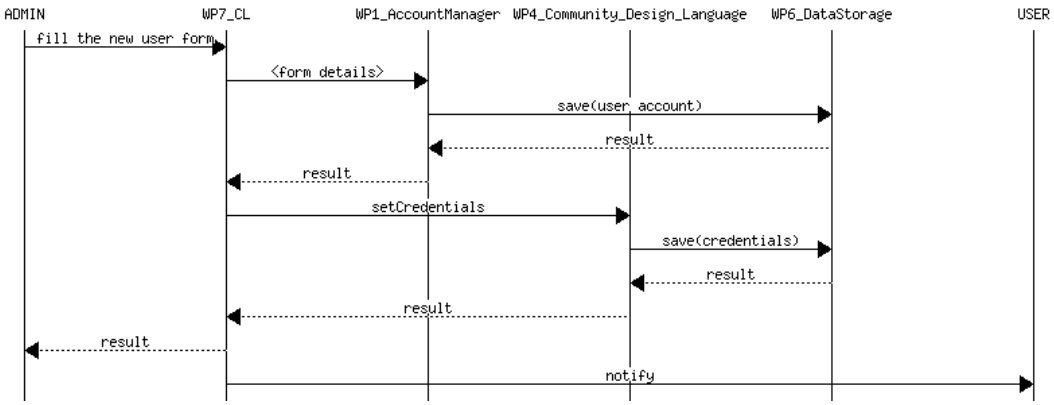


**Figure 23: Admin Creating New User Flow**

Thus, the admin fills out the user details that are then sent to the WP1 AccountManager service to create the user account. The WP4 CDL services from the WP4 Community Administration Platform are used to set the permissions of the user according to their citizen status. On creating the account, the admin application confirms that the account has been successfully created; notification is then sent to the user that their account has been created and provides confirmation of their OpenID, which is used to access the WKI system.

## 5.2. Logging In Workflow

*A message is sent to Mark with the invitation to WKI containing login details. Mark opens the link on his mobile phone, which gives him immediate access to the WKI webpage and mobile application.*

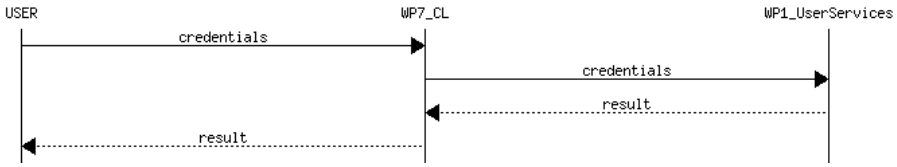


**Figure 24: User Login Workflow**

The login process uses the WP1 LoginManager service to register that the user has logged in and verifies their credentials. The login process for the first prototype is handled through the OpenID system developed by WP1, and so the workflow here does not need to authorise the user, merely ensure that the system is aware of their credentials.

## 5.3. Upload Image Workflow

*After Mark takes a picture of the road, he uploads the photo using the web interface.*

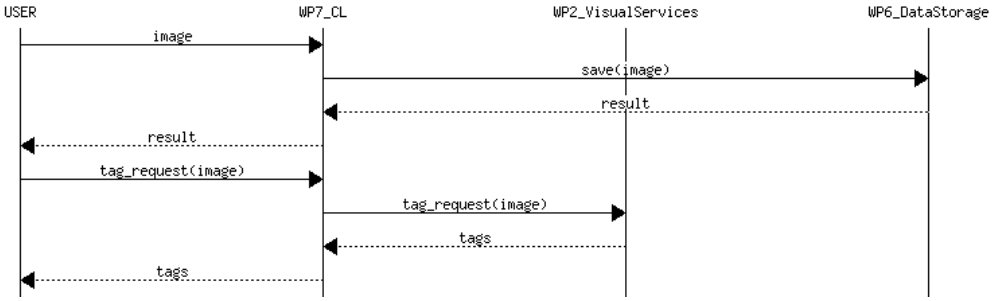


**Figure 25: Upload Image Workflow**

This workflow illustrates the process of uploading an image to WKI. The image is stored the Knowledge Store, the user then requests tags using the WP2 Tag Processing Service. These tags are then returned to the user for evaluation.

## 5.4. Tag Image Workflow

*Mark also quickly tags the picture by entering the word "flood".*

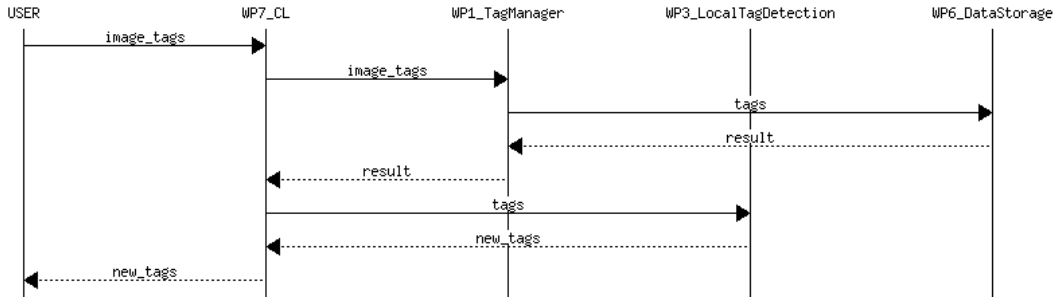


**Figure 26: Image Tagging**

Here the user is adding tags to the image – these tags are then stored to the Knowledge Base. On the basis of these tags, the WP3 Local Tag Detection service is used to suggest new tags that the user could apply to the image. Thus, the process of tagging the image is cyclical with new tags leading to new tag suggestions, until the user is happy with the tags they have entered.

## 5.5. ER Log In

*When John logs in (from his Desktop PC) he gets a screen with an overview of the available information.*

The login process is supported by the *Logging In* workflow detailed above in section 5.2.

## 5.6. ER Uploading Image

*When Andrea arrives on the scene of the emergency she starts taking pictures and uploads them to E-WeKnowIt.*

This process is handled by the Image Upload workflow detailed above, though since Andrea is a member of the ER team the image credentials will be handled differently.

## 5.7. ER Making Images Public

*John starts thinking about making some of the information public so normal users of the Sheffield City Council website can see the current situation and know which the most affected areas are. He then approves some of the images for public visibility.*

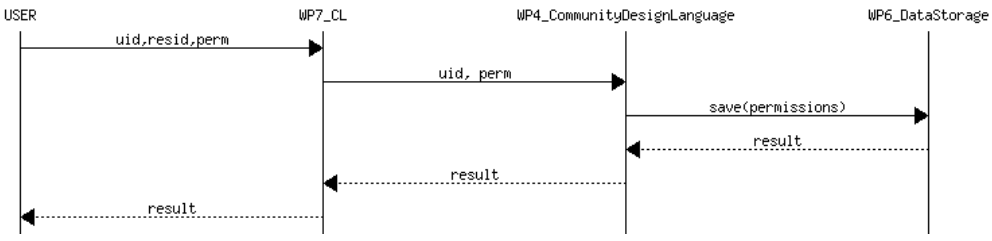


**Figure 27: Image Permissions Workflow**

As the workflow shows the process of making an image available to the public simply involves altering the permissions of the image via the WP4 CDL service.

## 5.8. Post Incident Login and Upload

*Mark has also been sent by friends some pictures about the emergency and wants to add them. Mark logs into E-WeKnowIt using his Desktop PC. He has already a login to the site so the system immediately presents him his home page. Mark chooses to upload the new content. When the content is uploaded Mark decides to tag all his photos with the words "flood, river, road, water".*

The login process is supported by the Login workflow detailed above; again OpenID handles the authentication process and the process retrieves the credentials of the user. The remaining steps follow the upload workflow outlined above.

## 5.9. Commenting Workflow

*He also adds a short comment of how he got involved in the floods and what he saw.*

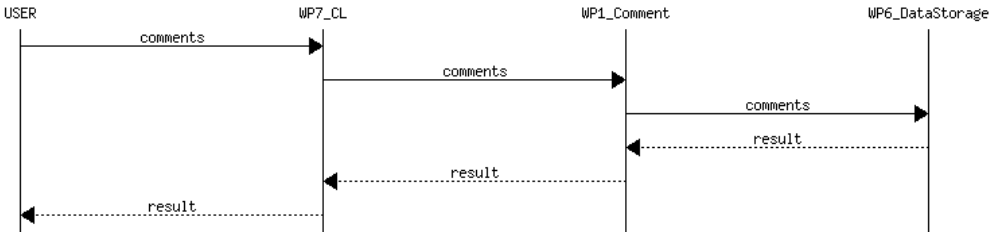


**Figure 28: Adding Comments Workflow**

The process of adding comments is handled using the WP1 Comment services – the comments are stored in the WKI KS and associated with the image.

# 6. Implemented Interface Designs

In this section, the implementation of the interface is described as this evolved from the interface mock-up designs and the paper-based evaluation of the designs.

The implemented interfaces differ from the mock-ups given above as they account for the first stage interface evaluation that will be described in D7.5.1. The primary findings of the first stage interface evaluation were that the upload interface was too complex and left the users unclear as to which action was required at each stage of the upload process. The desktop interface was clearer and the primary recommendation was for the terminology to better reflect the practices of the Emergency Response personnel.

Additionally, because of the current limitation of mobile web browsers it is not possible to use standard web techniques to support the uploading of information – this can only be achieved through the use of a native application for whichever operating system the phone uses. To overcome this limitation simple uploading applications can be developed to handle the upload in isolation. The user is then returned to the web interface to complete the upload process. This limitation of mobile browsers is likely to be resolved in the near future so this process represents a temporary solution for the upload process. Since the native application simply needs to just handle the process of logging in and uploading information it can be rapidly developed for multiple platforms. In addition since the tagging and publishing processes are maintained by the web application these processes can be developed and deployed centrally.

To illustrate the development of an uploading application for a typical mobile device, an Android application was written which supported the logging in of users and the subsequent upload of images that had been taken by the phones internal camera. This upload process was designed to mirror the mock up designs detailed above.

## 6.1. Desktop User Account Creation

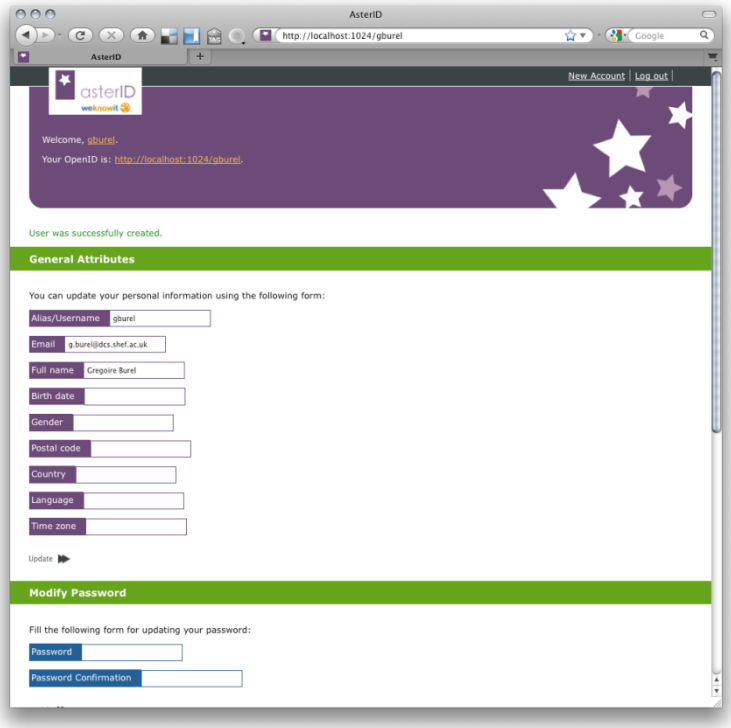


**Figure 29: Interface for creating user accounts**

This screen allows the emergency services call handler to create an account for members of the general public, thereby allowing them to upload information to WKI. To access the WKI system the user must be registered on the AsterID service which stores all the properties of the user and their ID must also be registered with the WKI system. This interface allows both registrations to occur.

This process corresponds with the Creating a New User Workflow described in the workflow section. The AsterID server stores the user details and registers the OpenID with the WKI system, thus meaning that the user is able to access the main ER application.

## 6.2. Mobile Login Screen

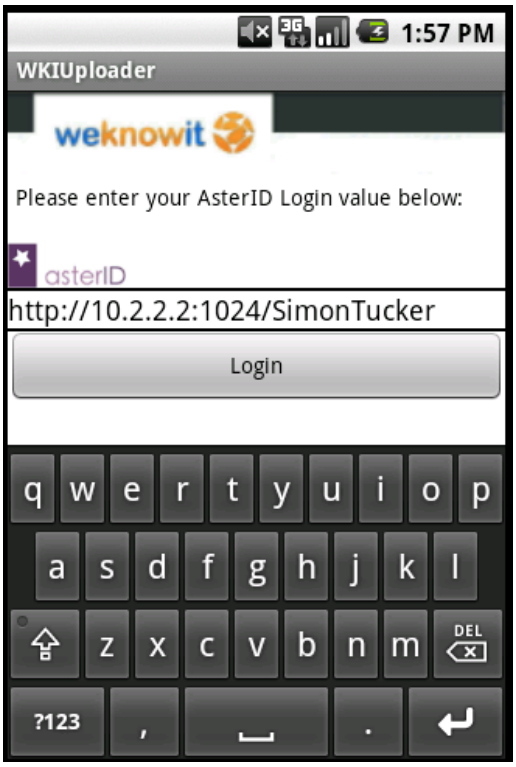


**Figure 30: Mobile Login Screen**

Once the user has installed the Mobile Application, they are able to login to WKI using their AsterID and subsequently upload information. The login screen allows the user to enter the AsterID in order to get access to the main upload section. If the user enters an ID that is not registered with WKI they are blocked from uploading any information.

In line with the user evaluation, the upload process has been simplified and designed to guide the user through the upload process quickly and with minimal distraction. Thus the login screen guides the user towards entering their credentials and then moving directly on to the image selection and upload process.

The login process uses the Logging In Workflow that is described above. The workflow returns the user id, which is then used to upload the image in the following step.

## 6.3. Mobile Upload Screen

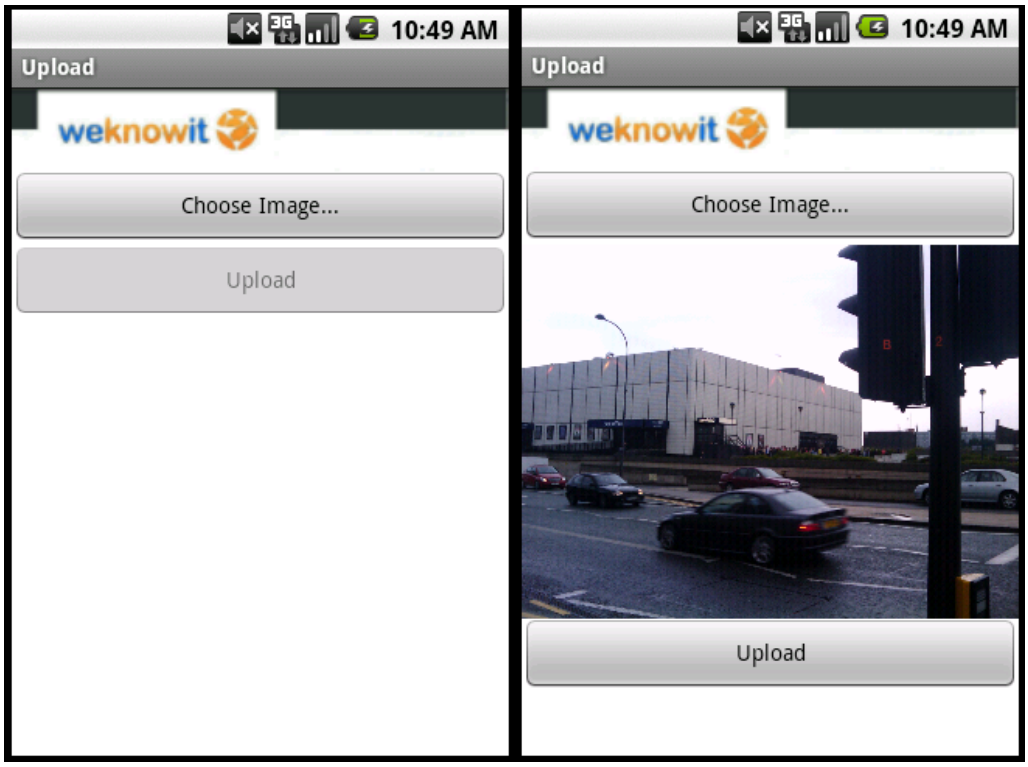


**Figure 31: Mobile Upload Screen**

Following on to the next stage of the mini-scenario, the user is then able to select an image from those available on their mobile device. They are given a reminder of the image that they have selected and can then choose upload to transfer the image to the WKI system. Upon doing so they see a progress bar that shows the state of the image transfer. Once the image has been transferred correctly the mobile application has finished and the user is transferred to the browser in order to complete the upload process.

As with the login screens this process has been highly simplified following the feedback from the mock-up evaluation. Consequently the mobile application purely focuses the user on choosing a suitable image and then uploading this image to WKI. Once the image has been uploaded control of the process passes to the standard WKI web interface.

This upload process uses the first part of the Upload Image Workflow described above. Thus the image is stored using WP1 services, while the following step requests tags for the image.

## 6.4. Mobile Tagging Page


**Figure 32: Mobile Tagging Screen**

The user can then add tags to the image. The image has been processed by WKI and so already has some seed tags (which the user can remove using the cross next to the tag name), using the WP2 Tag Processing service. This service uses the ER domain ontology in order to normalize existing tags to the ER domain ontology concepts. In addition, on the basis of the tags that have been applied to the image so far, further tags are suggested which the user can apply to the image by simply selecting them. These tags are discovered using the WP3 Local Tag Community Detector service, which identifies a collection of tags that form a community around the initially present tags. Since the mock-up evaluations suggested that the concept of tagging was unclear to some of the users, a brief piece of text describes how and why the user should apply tags to the image they are uploading to the system.

This process uses the Upload Image Workflow and Tag Image Workflow to firstly retrieve the initial tag suggestions and then update the user with more tag suggestions as the image is tagged. The suggestions are derived from WP3 services and the WP1 services are used to store the specified tags in the Knowledge Store.

## 6.5. Desktop Home Page

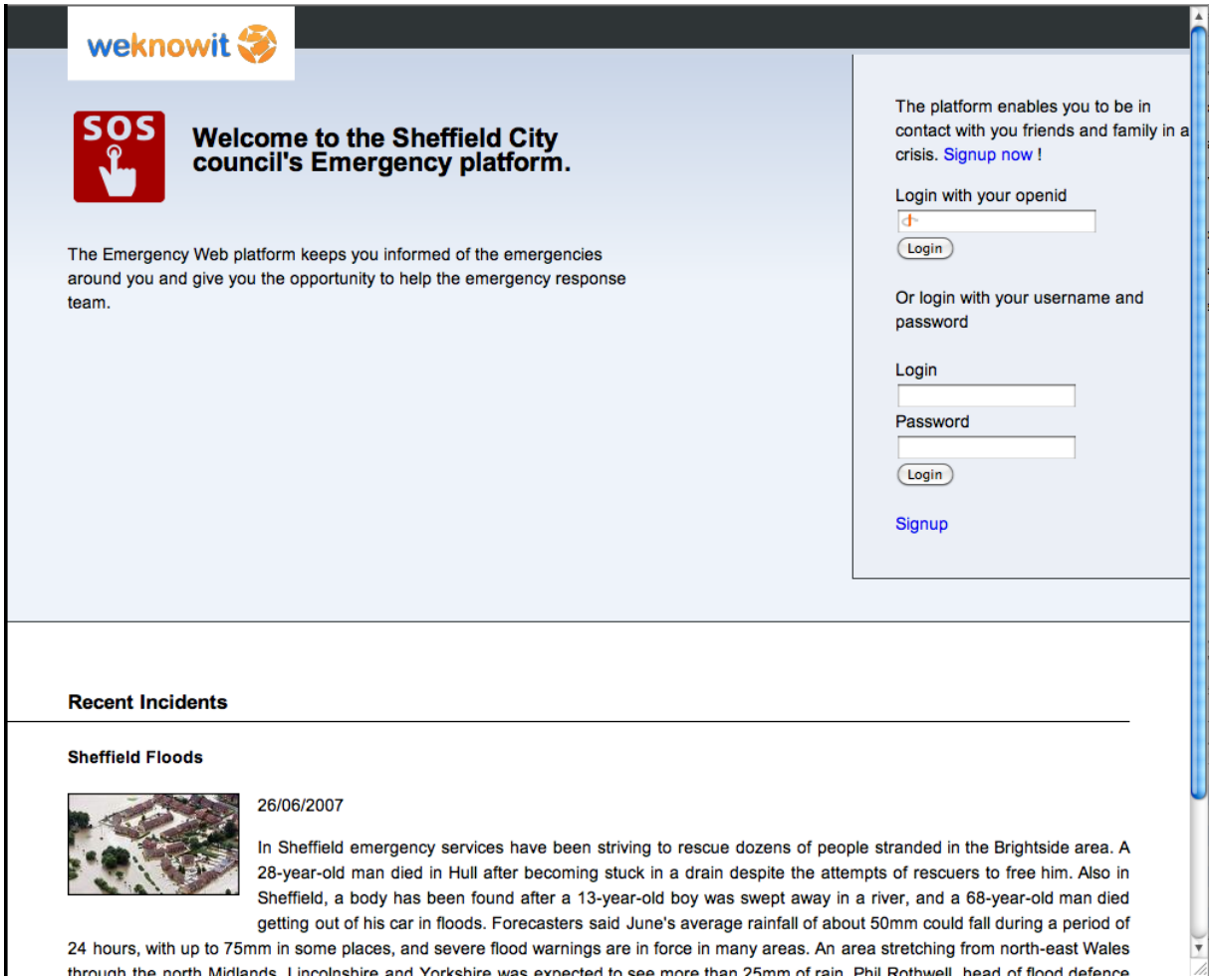


**Figure 33: Desktop Home Page**

The desktop home page corresponds to the home page shown in the mock-up designs. The display shows any public incidents that are occurring and allows an unregistered user to examine the public information represented by this incident. The screen allows users to either sign up to WKI or login via their OpenID in order to get full access to the system.

## 6.1. AsterID Login Page

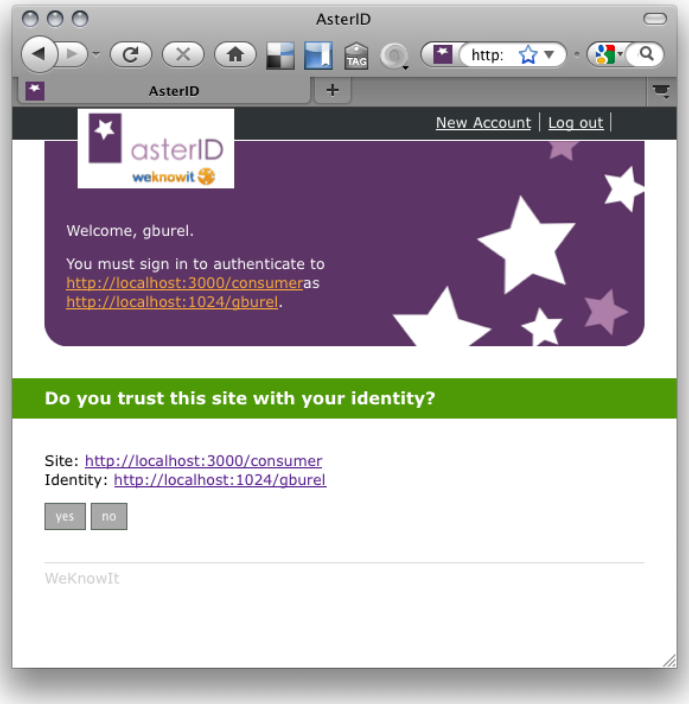


**Figure 34: AsterID Login Page**

These screens show the process of authentication using AsterID. Since the user is already logged into the AsterID server, all they need to do to access WKI is to confirm that they trust the WKI site with their identity information.

Once the users' credentials are confirmed, the WKI application registers the user using the Logging In Workflow. This allows the system to retrieve the user properties from the AsterID server and the ID of the user from the WKI server. The combination of this information allows the user to access the system.

## 6.2. Desktop Logged In Home Page

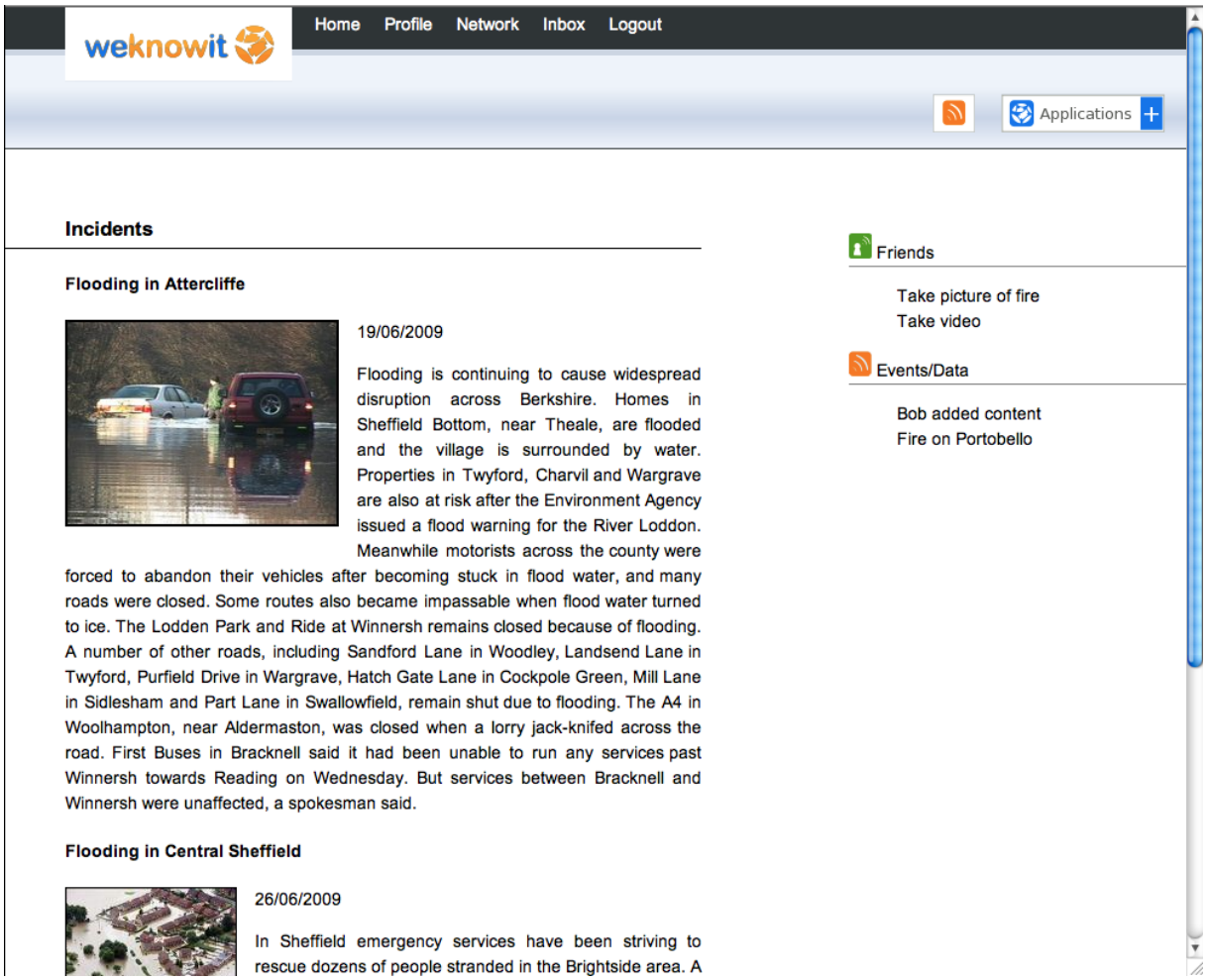


**Figure 35: Logged In Home Page**

This is the screen that the user sees when they have logged in to the system. As well as the public event that was visible from the previous home screen, the user can additionally see an event to which they have provided images. By selecting the image associated with an event, they can examine the information connected to the event using a geographical overview.

## 6.3. Desktop Geographical Overview

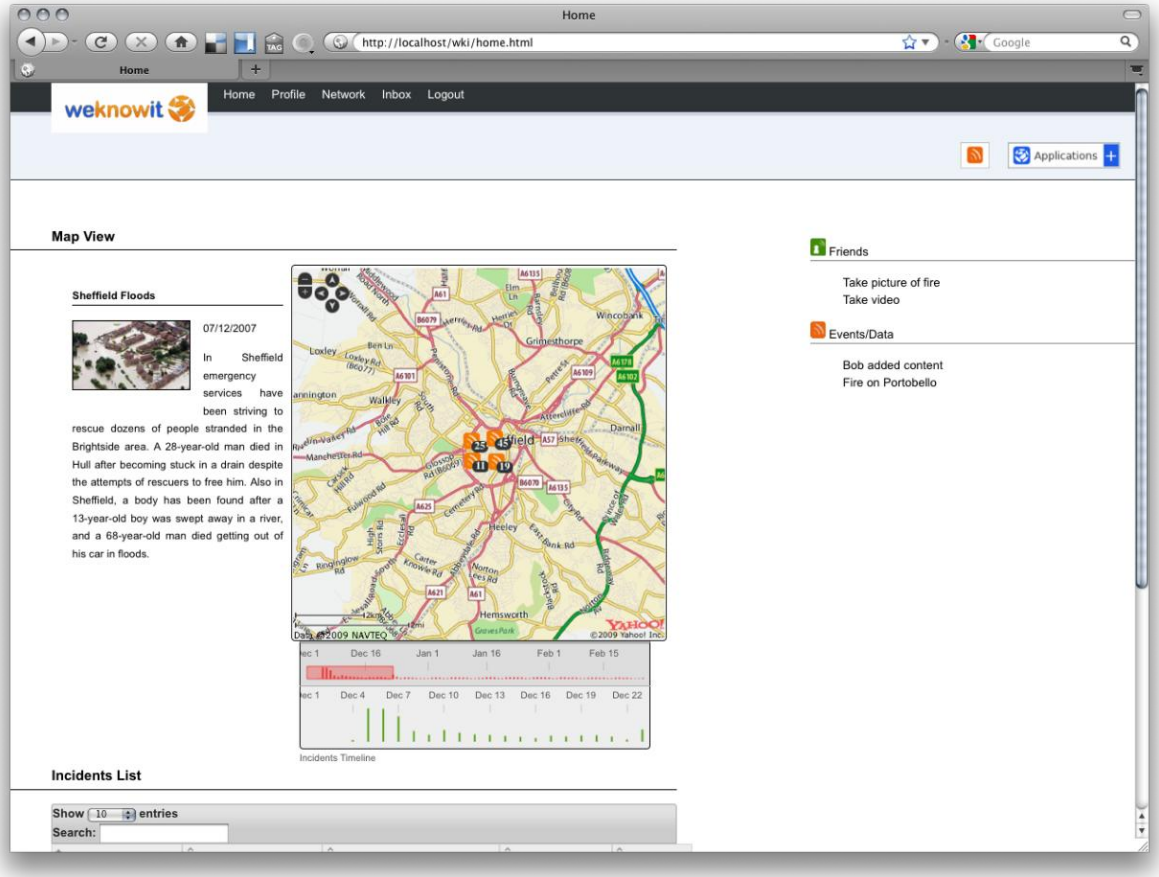


**Figure 36: Geographical Overview**

The user (whether they are citizens or ER personnel) is then able to view the information that they have permission to see using a geographical projection. The user is able to interact with the map with the standard actions of panning and zooming. The map attempts to assist the user with dealing with information overload by clustering images that are geographically close together into single entities. By zooming in to these clusters, the user is able to get more detail. This allows the user to see information relating to a single incident at multiple levels of detail. In addition, the user can see the temporal range of the event information and can filter the information by time, using the time display shown below the map.

## 6.4. Desktop Upload Process

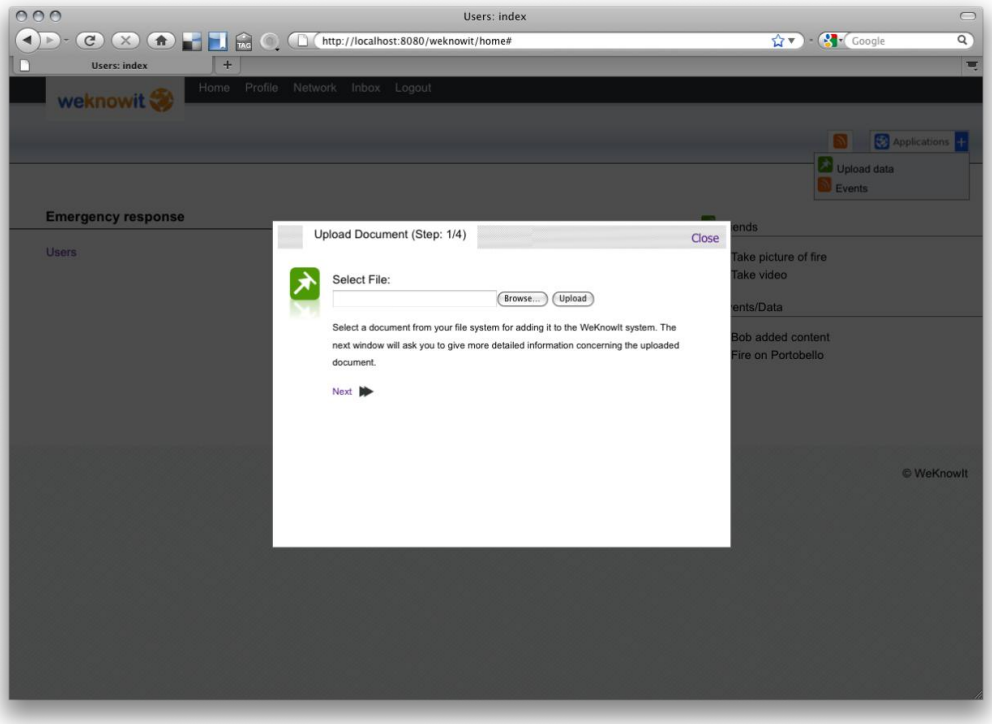


**Figure 37: Beginning the upload process from the Desktop**

There are four stages to the desktop upload process, which has been designed to mirror the upload process seen for the mobile device. Thus, the first stage allows the user to select a file from the file space to upload to the WKI system.

The workflows and services used in this process mirror the upload process described above unless noted otherwise.

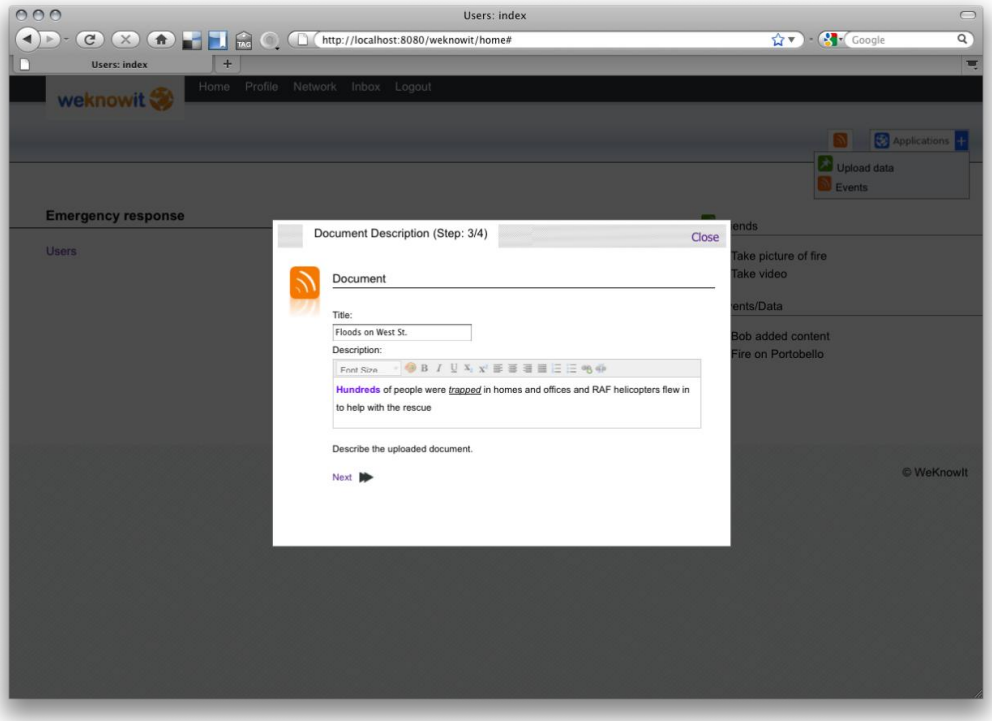


**Figure 38: Desktop Upload (2): Describing the Information**

When uploading information from a Desktop computer the user is likely to have more time to provide further information. Thus, this screen shows that the user can add a more descriptive title to the image and can additionally add a comment to the data. This information is then stored in the knowledge store and can be accessed from the Desktop interface.

The process of commenting on images is handled by the Commenting Workflow given above.

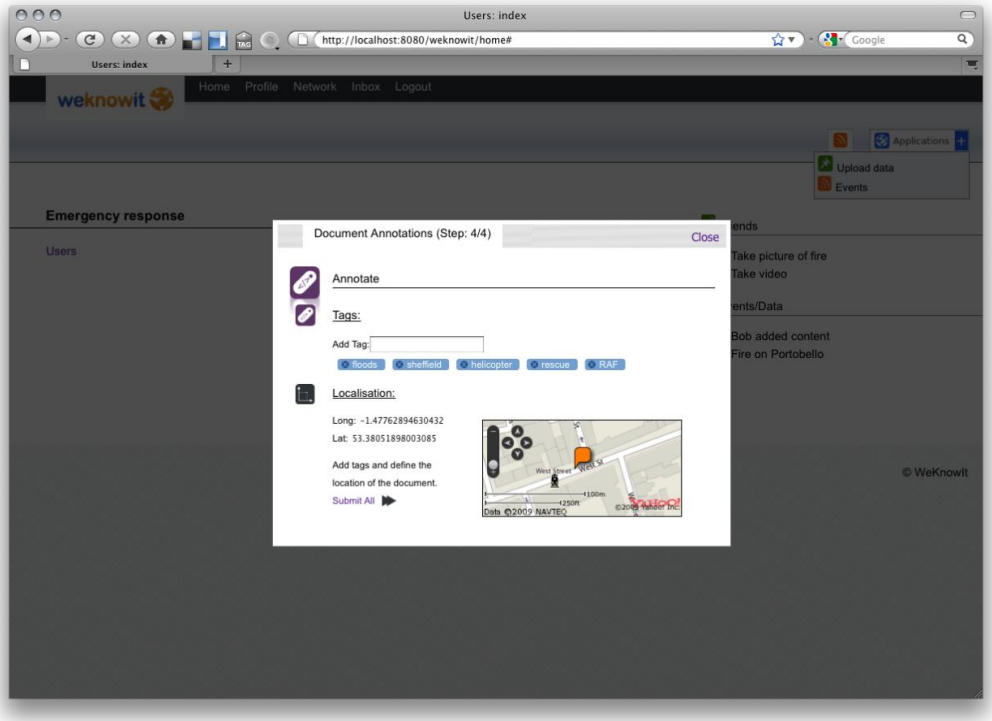


**Figure 39: Desktop Upload(3): Tagging the Information**

Finally, the user is able to enrich the information they have uploaded to WKI by adding tags, either from the list of suggested tags or by manually entering tags in the box displayed above. The desktop upload process also allows the user to directly localise the image using a map display. Since this process requires a rich interactive interface this process was not available in the mobile upload interface. Additionally, information uploaded from a mobile device may have location information already set by the device sensors so this step may be unnecessary.

# 7. Emergency Response Requirements

Deliverable D7.2 outlined the requirements of the first prototype for the emergency response use case. These highlighted both functional and non-functional requirements that the first prototype demonstrator must meet. This section describes how the implemented first prototype meets the functional requirements described in that document. Whilst the first prototype has been designed to additionally meet the non-functional requirements the success of the first prototype in meeting these requirements can only be determined through the evaluation process and so this discussion is left for the evaluation report.

## *7.1. Functional Requirements*

The functional requirements outline the general technical requirements for the first prototype. These functional requirements focus on the generic requirements for WKI applications whose focus is the ER use case and as such not all of these requirements will be met by the first prototype implementation. This section describes where the requirements have been addressed by the first prototype implementation.

### 7.1.1. Multimodal Interface

The first prototype of the ER demonstrator allows the user to access WKI through multiple interfaces as it enables access to the required functionality through both a mobile device such as a web-enabled phone, as well as through a desktop interface. These interfaces are designed to support the typical user actions that would be undertaken with each class of device. Thus, the mobile interface is designed to allow the user to upload images and the desktop interface is used to support access to the intelligence stored by WKI.

### 7.1.2. Content Upload

The first prototype implementation is geared towards the upload and processing of images, although other forms of information (for example comments) can be uploaded to the system. The prototype focuses on images to reflect the citizens' feedback.

### 7.1.3. Information Enrichment

The upload process incorporates information enrichment as part of the intelligence tagging process and also via the image analysis stage. Both these processes happen as part of the natural course of uploading images and so they are not explicitly invoked by the user. Thus, the information upload process naturally enriches the knowledge provided to the user.

### 7.1.4. Search and Browse functionality

The first prototype of the ER demonstrators supports the process of search and browsing in the desktop interface. The user is able to use the facets of the display (the timeline and the map view) in order to both search and browse and the information available to them. They can also use the high-level browse functionality to examine specific events in more detail.

### 7.1.5. Personalised Access and Support for Multiple Users

Personalised access to the ER is enabled through the process of logging in to the demonstrator system. The logging in process identifies the user and ensures that the information they are able to view is appropriate for their level of access. Thus, a citizen is only able to see images they or their friends have uploaded to WKI but the ER personnel are able to view all the images available to them. In addition to this, the user identification also ensures that actions that are only appropriate for ER personnel (for example, setting an image to be publically available).

### 7.1.6. Information Control

The requirement of information control is supported by the necessity for ER personnel to explicitly make information available to the public. This allows the personnel to ensure that sensitive information is not seen by the general public but also allows them to make relevant information (for example images of escape routes or of blocked roads) available to the general public if necessary.

### 7.1.7. Feedback/Rating

The ER demonstrator supports the process of feedback and rating – these values taking on different meanings depending on whether the rating originates from a citizen or a member of the ER personnel. For citizens the rating can be used to identify the quality of the information and this measure can be used to determine which information should be shown to the user. For ER personnel the rating can be used to determine the reliability of the information.

# 8. Conclusion

This document has described the implementation of the first prototype of the demonstrator for the Emergency Response scenario. The Emergency Response scenario is exploring how Collective Intelligence can support individuals at the scene of emergency incidents and the organisations that are dealing with the incident.

The goal of the first prototype demonstrator for the Emergency Response scenario is to allow citizens to upload information to WKI using an intelligent upload process. Thus, as part of this process, the information is enriched through the addition and refinement of tags. The emergency response personnel are then able to view this information with a geographical projection. On the basis of this information, the ER personnel are able to make more informed decisions as to how to deal with the emergency situation. The ER personnel are also able to make some information available to the general public.

The development of the first prototype demonstrator followed a standard user-centred development process. Prior work used interviews and questionnaires to identify the requirements for the overall demonstrator. From these requirements a scenario was defined which detailed how WKI technologies could improve the actions of individuals and organisations in an emergency scenario. Overall interface mock-ups were then designed and trialled with target users.

To define the functionality of the first prototype of the ER demonstrator, a mini-scenario was developed using a subset of the functionality defined in the overall scenario. From this mini-scenario, the interaction between the user and system was identified in the form of workflow diagrams. Following this, the interface was implemented and a composition layer was developed to bind the interface to the underlying WKI system.

The development of the first prototype demonstrator has shown that the WKI system developed within WP6 combined with the Composition Layer provides a suitable framework for building WKI applications. Further work (to be described in D7.5.1) will evaluate the system with target user groups in order to identify how the users perceive the benefit of Collective Intelligence processing in emergency situations. Moreover, the ER application will be continuously evolving with the development and incorporation of more intelligent services from the workpackages WP1-5.

# 9. References

1. Roy T. Fielding. Architectural styles and the design of network-based software architectures. PhD Thesis, University of California, Irvine, 2000.
2. Nielsen, J., and Landauer, T. K. (1993). *A mathematical model of the finding of usability problems*. Proc. ACM INTERCHI'93 Conf. (Amsterdam, the Netherlands, 24-29 April), 206-213
3. Preece, J., Rogers, Y. and Sharp, H. (2002). Interaction Design: Beyond Human-Computer Interaction. New York:Wiley
4. Schneiderman, B. (1998), Designing the User Interface. Addison Wesley.
5. Snyder, C. (2003). Paper Prototyping. Morgan-Kaufman
6. WeKnowIt – D1.1 – Report on Interaction Model
7. WeKnowIt – D1.2 – Personal Intelligence Management Technologies
8. WeKnowIt – D6.1.2 - Identification of architecture elements and relations, version 2
9. WeKnowIt – D6.4.1 - Integration of WP 1-5 tools and common components version 1
10. WeKnowIt – D7.1 - Consumer and Emergency Response Use Case Initial Requirements
11. WeKnowIt – D7.2 - Emergency response and consumers' social group case study design and specification

# Installation Instructions

## *Requirements*

Installing the first prototype of the ER demonstrator requires the following software to be have been installed:

- JRuby v1.4.0
- WKI System v0.9

The system can run on any typical unix based operating system and has been tested under linux and Mac OS X.

**Configuration**

The following jruby packages should be installed via the jruby gems installer:

- actionmailer (2.3.4)
- actionpack (2.3.4)
- activerecord (2.3.4)
- activerecord-jdbc-adapter (0.9.2)
- activerecord-jdbcmysql-adapter (0.9.2)
- activeresource (2.3.4)
- activesupport (2.3.4)
- crack (0.1.4)
- httparty (0.4.5)
- jdbc-mysql (5.0.4)
- jruby-openssl (0.5.2)
- json-jruby (1.1.7)
- rack (1.0.1)
- rails (2.3.4)
- rake (0.8.7)
- rspec (1.2.9)
- sources (0.0.1)

Note that some of these may already be installed as part of the JRuby installation. Assuming that the WKI system has been installed and is running correctly on the local machine, the web application can then be run from the command line:

./startserver

The application can be packaged inside a suitable web server using the standard approach for that server. Using the above method the webapp will be hosted at the following location:

**http://localhost:3000/weknowit/**

The web application will assume that the WKI system and composition layer are running on the same machine and can be accessed through the default port settings.

### Image Upload Mobile Application

The android application is supplied as an application file and can be installed on a suitable phone using the standard method.

# Ancillary Interface Screens

This part of the appendix details interface screens that do not form an integral part of the mini-scenario events but provide support to some of the steps. Figure 40 to Figure 43 concern screens that may be shown during the authentication process for the AsterID server.

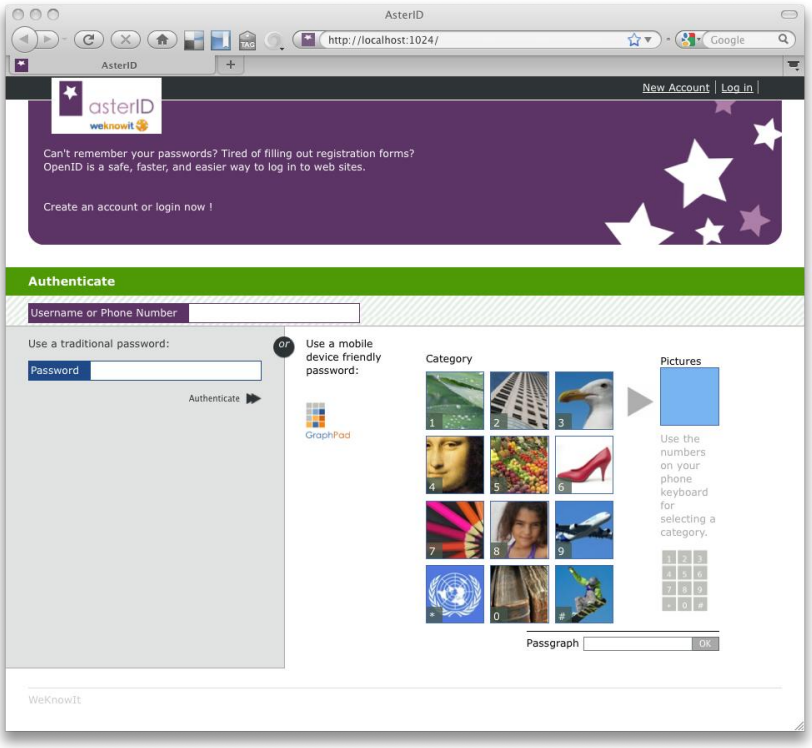


**Figure 40: AsterID Home**

This is the home screen for an unlogged user for the AsterID server.

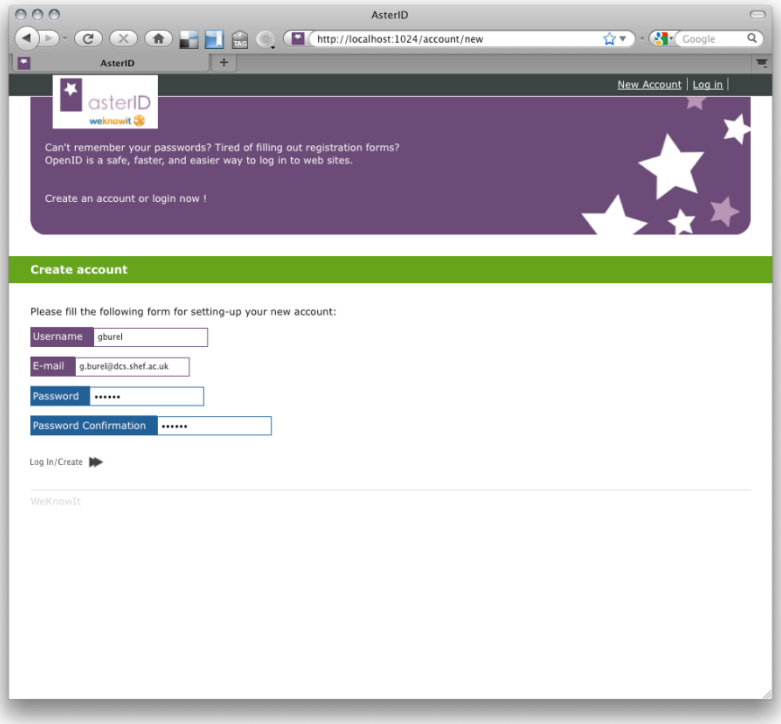


**Figure 41: Initial Account Creation Screen**

This screen shows the initial account creation screen.

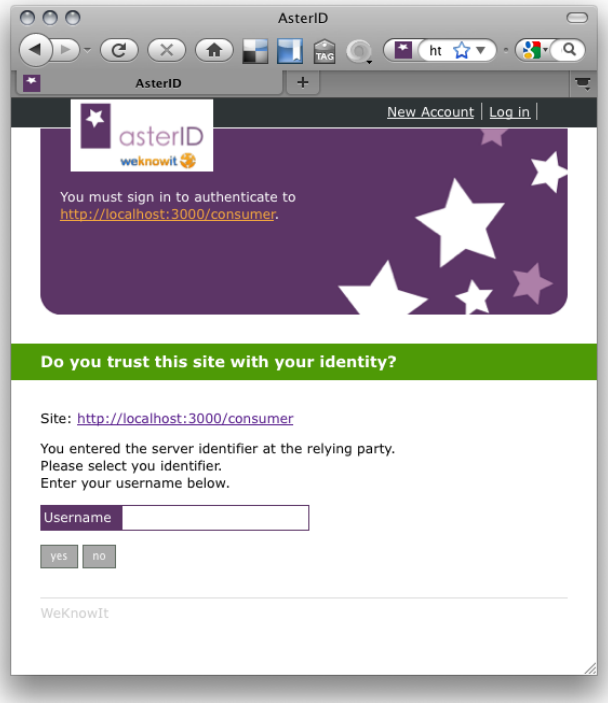


**Figure 42: Sign In Screen**

The user sees this screen when logging in via AsterID.

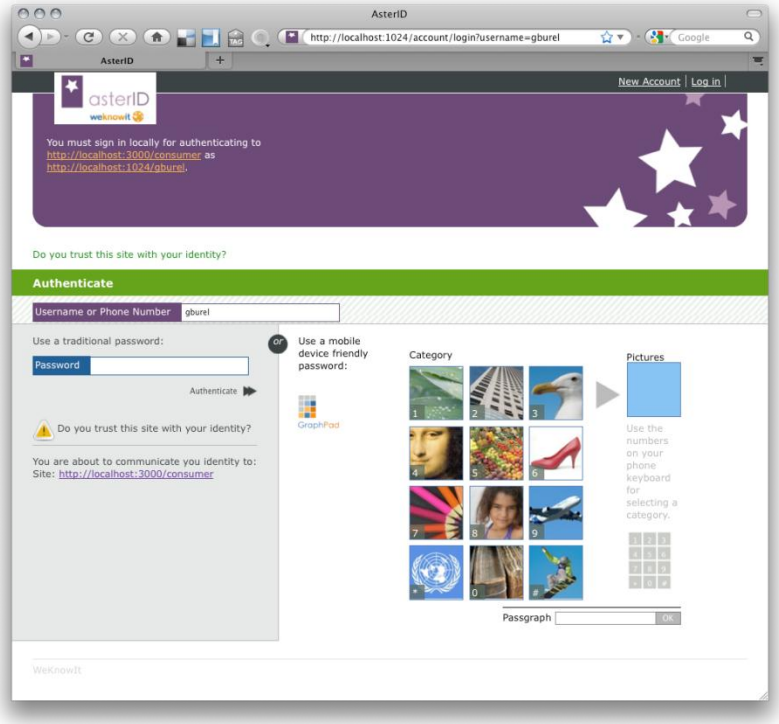


**Figure 43: Authentication Screen**

This screen shows the authentication process for the AsterID server.