An Agent-based mTest and mAssessment Service Delivery Platform

Ivan Ganchev¹, Damien Meere¹, Zhanlin Ji¹ 2, Máirtín O’Droma¹

¹Telecommunications Research Centre, University of Limerick, Ireland.
²Beijing University of Science and Technology and Hebei United University, P.R. China.

Abstract

Within this paper, an InfoStation-based multi-agent architecture is presented, which provides the infrastructure to support the delivery of contextualised mobile assessments in the form of mTests, which could be utilised to enhance more traditional localised educational practices or indeed to provide additional tools to support distance learning. The utilisation of a multi-agent system architecture, incorporating the personal mobile devices of learners, is described, building on the work presented in [1-3]. The main architectural components within the system are discussed, detailing the functionality required in delivering effective assessments and feedback to learners. As well as this, the continuing client-side developments, including the current research geared towards the incorporation of HTML5 is detailed. Finally, the system functionality required for effective service management by educators is detailed.

1. Introduction

Up until recently, for the most part, educational practices have generally remained unchanged for a great many years, with educators delivering information face to face with learners. However, the technological innovations of the last 50 years have precipitated a change in the way these practices are undertaken, just as they have changed the so many aspects of everyday life. In particular, the advent of the Internet has made information more readily available to people than ever before. The major gains associated with the utilisation of eLearning within a blended learning environment is well suited to meeting the varying needs of such diverse cohorts, aiming at replacing old-fashioned time/place/content predetermined learning with a just-in-time/at-workplace/customised/on-demand process of learning [5]. The blended learning approach can offer these learners (both digital natives and more mature learners) a more holistic experience of education (particularly tertiary education). The major gains associated with the utilisation of blended learning are the opportunities it offers for learner collaboration and formative assessment, as well as enabling the educators to more actively monitor the progression and attendance of learners, without the need to expend a great deal of time on these tasks [6, 7].

However, while studies on the use of blended-and online learning advocate technology adoption,
they have also highlighted a number of possible pitfalls. For example, in order for the learners to fully engage with the learning experience, it is vital that the focus be kept on the material being delivered, rather than the attention being drawn on the novelty of the technological medium used to convey the learning material. Therefore, the technology must, in essence, act as a transparent medium through which the educational materials may be delivered. The key to an enhanced learner experience is a tool that is capable of meeting both the learners’ and the educators’ needs and one that is reliable and easy to use [8]. This calls for the utilisation of technology which can be seamlessly integrated in the learners’ lives and the lessons. A possible solution, it seems, is already in the pockets of many learners.

The system presented in this paper seeks to harness the potential of mobile technologies to augmenting educational practices, enhancing the experiences of learners, as well as providing educators with newer and more effective teaching tools. As mobile technologies become the dominant mode of access to the Internet [9], what has commonly been known as eLearning, is being enhanced by the new communicative potential, becoming mobile eLearning (mLearning). The utilisation of mobile technologies will enable higher education institutions to extend their reach, no longer constrained by location or time, but offering true ‘anywhere-anytime-anyhow’ learning [10]. This utilisation of mobile technologies represents a huge resource that, for the most part, has been left untapped as regards the delivery of learning content within the educational domain. The system presented in this paper, offers educators the opportunity to dynamically create assessment materials to be accessed by learners via their own mobile devices. In doing so, an educator can better interact with larger groups of learners, both local and remote, and better appreciate learners’ progress with presented materials.

The rest of the paper is organised as follows. Section 2 describes the InfoStation-based multi-agent architecture underpinning this system, detailing, in particular, the main components involved in facilitating the user access to services within lecture theatres, and throughout the university campus. Section 3 presents a detailed view of the interactions which take place between the various system components in effective delivery of the mTest service to users. Also detailed within this section are the two main variations of the typical mTest/mAssessment service within this system. Section 4 details the functionality of the system from the perspective of both the educator and the learner, highlighting how each may access and interact with the service to effectively enhance traditional practices. Finally, section 5 concludes the paper.

2. InfoStation-based multi-agent platform

The InfoStation-based multi-agent system detailed in this paper provides access to mLearning services, for users equipped with mobile devices, via a set of intelligent wireless access points, called Information Stations (InfoStations), deployed in key points across a university campus (e.g. in lecture theatres, labs, library, cafeterias etc) [11-13]. The InfoStation paradigm was originally proposed to provide “many-time, many-where” wireless data services [14]. As opposed to facilitating ubiquitous coverage, within geographically intermittent pockets, InfoStations would offer high-bit-rate connectivity to broadband data- and network services [15]. The system presented in this paper involves various entities operating within a 3-tier structure as depicted in Figure 1: user mobile devices, InfoStations, and an InfoStation Centre [2, 3, 11].

![Figure 1. The 3-tier InfoStation-based network architecture](image)

The presented architecture harnesses the communicative potential of modern mobile devices in order to present learners with a more pervasive learning experience which can be dynamically altered and tailored to suit them and their mobile devices. Whilst within range of an InfoStation, users may gain access to various contextualised and personalised mLearning services and resources distributed throughout the system architecture. Due to the inherent mobility support of the system, a means of facilitating sufficient flexibility and adaptability, in overcoming issues related to the delivery of services within heterogeneous wireless deployment environments, was required. For this reason an agent-oriented approach was adopted for the development of the system. Following this approach, intelligent agents operate throughout each tier of the system. Indeed the InfoStations and the InfoStation Centre themselves take on the appearance of multi-agent systems (MAS). [11, 16] describe the implementation of MAS, facilitated through the utilisation of the Java Agent DEvelopment (JADE) framework [17, 18], developed by the Telecom Italia Lab (TILAB). This software framework simplifies MAS implementation through the provision of a predefined set of services.
and management tools in addition to the runtime library and the agent programming library.

The JADE architecture itself is completely modular. By utilising specific modules (or add-ons), JADE can be configured to adapt to the requirements of a number of different deployment environments. This alteration of the platform is especially important within mobile environments, where the availability of resources such as processing power, memory and connectivity can have a much greater bearing on the effectiveness of JADE execution. One of the main add-ons associated with JADE is the Lightweight Extensible Agent Platform (LEAP) module [19-21], which replaces various parts of the JADE kernel providing a modified lightweight run-time environment and enabling Foundation for Intelligent Physical Agents (FIPA)-compliant agents to execute on a wide range of Java-enabled mobile devices. These various agents take responsibility for selecting the optimal operation mode and establishing a client-server cross-platform connection, conveying context information, and delivering adapted and personalised service content, etc. Figure 2 illustrates the main components within the system, including the main agents involved in the service delivery, and the data persistence mechanisms incorporated into the system.

The Connection Advisor Agent (CAAgent) performs the role of identifying the context of the user making the request for services from an InfoStation. The main task of this agent is to facilitate users with a list of services directly applicable to them. When the user selects a particular service, they wish to access, the CAAgent is then tasked with supplying the location of the relevant service content to the user’s Personal Assistant Agent (PAA), installed on the user’s mobile device. The list of applicable services is harnessed from the service database. The two main mLearning services relate to the delivery of contextualised lecture content and testing materials. With this in mind, two agents are tasked with delivering these services to requesting users – the WALL Next Generation agent (WNGAgent) and the mTest delivery agent (mTDAgent). The primary task assigned to the WNGAgent is to facilitate the delivery of the requested service content to the user’s PAA. This process involves discerning the capabilities of the requesting device, in particular the file-types and mark-up supported, and then adapting the requested content to suit the operating environment onboard the requesting device. The mTDAgent is charged with the delivery of mTests to users. The mTest service [22] provides a means to evaluate the learners acquired knowledge and receive valuable feedback, making the learners aware of their progress in the assimilation of the presented course content, and highlighting the knowledge or skills they lack or may need to strengthen. This is achieved through the delivery of various forms of test content to the user’s PAA, which will be detailed over the coming sections.

Figure 2 illustrates the main components within the system, including the main agents involved in the service delivery, and the data persistence mechanisms incorporated into the system. The Connection Advisor Agent (CAAgent) performs the role of identifying the context of the user making the request for services from an InfoStation. The main task of this agent is to facilitate users with a list of services directly applicable to them. When the user selects a particular service, they wish to access, the CAAgent is then tasked with supplying the location of the relevant service content to the user’s Personal Assistant Agent (PAA), installed on the user’s mobile device. The list of applicable services is harnessed from the service database. The two main mLearning services relate to the delivery of contextualised lecture content and testing materials. With this in mind, two agents are tasked with delivering these services to requesting users – the WALL Next Generation agent (WNGAgent) and the mTest delivery agent (mTDAgent). The primary task assigned to the WNGAgent is to facilitate the delivery of the requested service content to the user’s PAA. This process involves discerning the capabilities of the requesting device, in particular the file-types and mark-up supported, and then adapting the requested content to suit the operating environment onboard the requesting device. The mTDAgent is charged with the delivery of mTests to users. The mTest service [22] provides a means to evaluate the learners acquired knowledge and receive valuable feedback, making the learners aware of their progress in the assimilation of the presented course content, and highlighting the knowledge or skills they lack or may need to strengthen. This is achieved through the delivery of various forms of test content to the user’s PAA, which will be detailed over the coming sections.

The mTests are represented by Java-based mTest objects, which are stored within a Database for Objects (DB4O) [23]. DB4O is an Object-Oriented Database Management System (OODBMS) which facilitates “pure object storage”. It’s use within this system was discussed previously in [24]. On receipt of a request from a PAA for a particular mTest object, the mTDAgent queries the mTest DB4O database in order to harness a particular mTest instance (Figure 2). Each mTest object is an instance of the mTest class which takes the form illustrated in Figure 3. Each instance of an mTest object contains a String value identifying the name of the mTest, as
well as an ArrayList of mTestQuestion objects. These mTestQuestion objects, instances of the mTestQuestion class, contain a number of basic string members relating to various aspects of that particular question. Once the mTest object has been retrieved from the database, the mTDAgent harnesses the various associated questions, and passes these on to the requesting client. The client then harvests the inputted answers from the user (via the PAA) and passes these back to the mTDAgent. At this point the agent sets about checking the received answers against those specified within the answer string of the relevant mTestQuestion object. Once the agent aggregates the user’s score, the user’s associated profile (within the UserDB4O database) is updated, to incorporate the new test score. In order to interact with the various elements within any mTest instance, the mTest class contains a number of ‘Get’ and ‘Set’ method pairs, associated with each particular member within the mTest class (e.g. GetMtestName & SetMtestName). The nested mTestQuestion objects are composed in much the same way, using ‘Get’ and ‘Set’ methods to access member variables. The utilisation of these methods to facilitate interaction to the mTest materials, from both the perspective of the learners and the educators, is detailed within Section 4.

![Figure 3. The mTest class structure](image)

### 3. Enhanced mTest/mAssessment service

Within educational institutions, it is not enough to simply present information to learners. Educators require a means to assess the effectiveness of new educational tools and practices. However, while very often more traditional approaches to assessment take precedence over the utilisation of a technology enhanced technique, the provision of a means for evaluation is crucial to the success of any educational system. This is no different with this InfoStation-based system. The mTest service provides a means to evaluate the learners acquired knowledge and provide valuable feedback, making the learner aware of their progress in the assimilation of the presented course content, and highlighting the knowledge or skills they lack or may need to strengthen. Indeed the main benefit of using quizzes is the motivation of the learners engagement in the material, without the stress associated with traditional exams [2]. Educators may also benefit from such information. By monitoring the progression of a group of learners, the educator may actively modify their approach to conveying the course content and as such, optimise the performance of the group, and enhance the overall learning experience. The mTest service can be a very useful tool beyond the confines of the lecture theatre, in a more out-of-band approach, with learners accessing mTest whilst on the move. However, it is not limited to this type of usage. While there has been a stigma around the utilisation of mobile devices within educational practices for many years, this negative view is slowly changing.

Under the proper circumstances, mobile technologies can even be utilised to enhance the participatory nature of education, as interactivity has very often been highlighted as a means for promoting more active and engaged learning, building a greater sense of community and motivation for learners, as well as providing educators with a means to gain feedback [25]. The mTest service does not operate explicitly as a mechanism for either testing or as a tool to aid educator-learner communications. Instead it attempts to build on the aspects of both. With this in mind, there are two very distinct approaches taken to the deployment of the enhanced mTest/mAssessment service in this system. The first approach, type A, involves the utilisation of the mTest service within the classroom environment, designed to enhance the participatory nature of education, just as was undertaken by Kinsella and Markett [26, 27]. Within the lecture hall, the educator may periodically deploy mTests to be executed during the lecture to obtain the current level of understanding of learning material by the class. The educator can also use this service to harvest the opinions of learners, which can often prove quite difficult when dealing with large classes. The second approach, type B, involves the deployment of mTests (mAssessments) outside the environs of the lecture hall. With this approach, the mTest is utilised for general testing and assessment of knowledge, e.g. as part of midterm tests which contribute towards the final module assessment. With this second approach, the educator can utilise a much more diverse range of media and question types. This service, utilised in conjunction with the mLecture service, provides the requisite tools for educators to truly facilitate learners with an effective technologically enhanced educational environment. Another aspect of this enhanced service, which is paramount to its success, is the synchronisation of the off-line service interactions with the on-line mLearning process. This synchronisation is especially important in an InfoStation scenario due to this paradigm’s geographically intermittent connection where mobile users will invariably pass in and out of range/contact with the InfoStations. When a mobile user enters within the range of an InfoStation, s/he goes through the normal authentication, authorisation and accounting (AAA) procedure, and selects the relevant service. Let’s assume that the mTest/mAssessment
service is chosen. Once the user/learner gains access to the service, s/he may choose a particular assessment. The PAA on the user’s mobile device forwards a service request on to the InfoStation, specifying the user’s choices. The InfoStation in turn, having analysed the capabilities of the target device, discerns the optimal format in which to present the assessment. This service content is then made available to the PAA.

In the current iteration of the system, the mTest of type A is a small text-based test, with the various questions being conveyed from the InfoStation to the PAA within Agent Communication Language (ACL) messages. On the other hand, the mTest of type B is contained within a Java Archive (JAR) file and may itself be its own standalone application, containing various media types. In the case of the latter approach, on selection of this service via the PAA, the JAR file is transmitted via the available wireless connection. With both approaches, upon completion of the mTest, the PAA passes on the answers inputted by the user on to the InfoStation. Within the InfoStation, the mTDAgent, on receipt of the submitted answers, queries these against those stored in the mTestDatabase.db4o file, correlating the user’s score for that particular mTest. This scoring information is then stored to the user profile within the UserDatabase.db4o file, and the scoring information is passed back to the PAA, which in turn displays this feedback to the user. The educator can then access the scoring information by querying the UserDatabase.db4o file. This process is illustrated in detail in the sequence diagram on Figure 4.

However moving forward, the client-side of the mTest design will expand to incorporate newer technologies to increase the system's effectiveness. To enable mobile devices to run mTests independently of hardware/software constraints, we are developing a mTest presentation layer based on Sencha Touch [28]. Sencha is the first HTML5 framework for mobile devices. The graphical user interface (GUI) application, based on Sencha, can provide effective rendering on initialization of the system. Through the built-in functionality, the application can implement different interface renderings, depending on the device type, which is essential in a mobile environment, as it enables an application to be hardware independent. The application needs only a WebKit-based browser. Sencha, combined with CSS3, makes it easy to develop effective Web applications. This is similar to local applications and allows for the realization of rapid development of cross different platforms. These strengths make Sencha a good tool for development of the mobile device’s mTest GUI. In addition, Sencha Touch provides a flexible layout system, which makes it easy to determine the size and location of the layout for different devices, which provides basic support for mTest customisation. The framework of Sencha Touch can determine the device style when the application is starting and can ensure the following:

1. Quick start of the application and interface rendering.
2. Quick update of the interface after rotation.
3. As the layout engine is very small, the running speed is increased.

Figure 4. The mTest/mAssessment service provision

It is reasonable to use Sencha to design mTest mobile GUI. The style of the display interface looks somewhat like Swing in Java but is more dynamic in terms of the appearance, particularly through the use of widgets and effective layouts for display. Through the creation of widgets and their desired combination within a layout, we can design effective and attractive interfaces.

3.1. Overall architecture of mTest

The MVC framework is a popular design pattern within the presentation layer, which is employed within the mTest mobile GUI design. A HTML5/Sencha application usually includes Models, Views, Controllers, Stores, Profiles, Icons, etc. Figure 5 shows the mTest application structure, described below.
• The Model represents the various types of objects in the mTest application, such as Question objects, Item objects, Image objects, etc.
• The View represents graphical interfaces to be displayed on the screen. It is a description of the data object in the application. In mTest, it includes Login, TestPaper and other interfaces.
• The Controller is in charge of capturing the interaction of the user and the program event. Through monitoring the touch and slide events on screen, the program can make responds and render the GUI.
• The Store is a storage container for loading data. It is very useful in mTest as it could show the data in List or DataView or within other widgets.
• The Profile is a protocol for judgment in the environment of different handheld devices. This allows for the reuse of code across multiple platforms.

3.2. Design and implementation of mTest

To achieve the MVC design of mTest in Sencha, related modules should be stored in the corresponding application folder, such as View, Model, Controller, and Store. The mTest entrance function will automatically load these modules. A mTest consists of a number of views, such as the login screen, question selection screen, etc. The controller monitors the interaction between user’s actions. When an event is triggered, the controller will capture it automatically and will display the corresponding view. In mTest, different parts are connected together through the controller. The program runs by monitoring the screen UI events triggered by the user, e.g., clicking and sliding. The controller separates the display logic and control logic to ensure loss coupling of mTest. When the user logs into the mTest, s/he will first touch the login screen (Figure 6). The controller running in the background is responsible for monitoring the user operations. Once the user touches the submit button, the controller receives a trigger event notification and verifies the username and the password.

As the mTest mobile GUI is a hardware-independent application, all the datasets, i.e., personalized information, exams, pass score, etc, are stored on the server side. Communication between the mTest GUI and the mTest server utilises the JavaScript Object Notation (JSON). The mTest defines a set of models, which are objects themselves, with various attributes. For instance, the user model includes requisite user parameters. At the same time, an exam object is defined in mTest to represent an online exam that includes questions and answers, etc. The Store can load the model objects and persistence data from one of two proxies – the Client Proxy or the Server Proxy. With the Client Proxy, data can be loaded in the memory space of the browser directly; it could also load the local-storage data which is supported by HTML5. With the Server Proxy, data is loaded from the server side. The server proxy is predominantly used in the mTest mobile

![Figure 6. The login operation of mTest](image)

![Figure 7. The mobile device’s mTest GUI](image)
device’s GUI design. Figure 7 presents a number of views of the developed mobile device’s mTest GUI.

4. mTest/mAssessment utilisation and administration

From the client’s perspective, the mTest/mAssessment service is based within a standalone MIDlet application. In delivering the service in a basic textual format, this MIDlet facilitates the initiation of PAA, the harvesting of user service requests and interaction with the InfoStation agents, and the retrieval of service content. When the user submits their credentials to the system via PAA, the CAAgent within the InfoStation discerns a list of mTests available, which are applicable to that particular user, based on their own personal context [22]. Upon selection of the relevant mTest (as illustrated in Figure 8a), a request is again sent to the CAAgent in order to harness the location of the mTest content (c.f. Figure 2). The request to the mTDAgent simply consists of an ACCEPT_PROPOSAL ACL message, specifying the name of the requested mTest. On receipt of this user request, the mTDAgent locates the requisite mTest instance within the mTest DB4O database, and harvests the various ‘question’ strings from its constituent mTestQuestion objects. These strings are then passed back to the requesting client. Upon receipt of these question strings, the ClientMTest agent must set about presenting them to the user. The strings are initially broken up into a String array and displayed on the screen on the user’s device, as illustrated in Figure 8b. The user cycles through the various questions and fills in their responses. Once the user completes the mTest, the ClientMTest MIDlet passes the inputted answers to the mTDAgent for correction. An INFORM ACL message bearing the user’s answers in string form is sent on to the mTDAgent, which compares the answers given by the user against those stored within the mTest DB4O database. Once the user’s score has been collated, the User DB4O database is updated to reflect the scores achieved by the user, and the scoring information is passed back to the client. On receipt of this information, the client MIDlet displays the attained mTest score to the user within the result screen (Figure 8c).

As detailed previously in [29], the InfoStation system described here is built upon a custom HTTP server [30], which also facilitates the initialisation of the JADE environment. This approach enables the delivery of service content to the users through one of two access mechanisms – the agent-based system or though a device’s browser application. As presented in [31, 32], upon initiation of the server, the JADE MAS is started up, and a number of HTTP handlers are associated with the server for processing the requests and responses. Within this system, the educator interacts with the server in creating new mTest instances and harvesting the results of learners via a web interface, supported by these HTTP handlers. The web interface enables the educator to interact with the system, inputting and retrieving information from the various DB4O databases. Figure 9 illustrates the interface presented to the educator when creating a mTest instance. Within this form, the various pieces of information necessary to create a mTest service instance can be seen. The information harnessed within this form is stored to both the mTest- and service databases. The reason for this is that there must be an association between both databases, as the service database is utilised to advertise the content of the mTest database. The service database must also be aware of the location of files stored within the underlying file directory, which are to be delivered to the user’s device. Upon completion of this form by the educator, the information is stored to the mTest- and service DB4O databases. At this point, the learners may retrieve the newly created assessment material through the system.

As detailed previously in [29], the InfoStation system described here is built upon a custom HTTP server [30], which also facilitates the initialisation of the JADE environment. This approach enables the delivery of service content to the users through one of two access mechanisms – the agent-based system or through a device’s browser application. As presented in [31, 32], upon initiation of the server, the JADE MAS is started up, and a number of HTTP handlers are associated with the server for processing the requests and responses. Within this system, the educator interacts with the server in creating new mTest instances and harvesting the results of learners via a web interface, supported by these HTTP handlers. The web interface enables the educator to interact with the system, inputting and retrieving information from the various DB4O databases. Figure 9 illustrates the interface presented to the educator when creating a mTest instance. Within this form, the various pieces of information necessary to create a mTest service instance can be seen. The information harnessed within this form is stored to both the mTest- and service databases. The reason for this is that there must be an association between both databases, as the service database is utilised to advertise the content of the mTest database. The service database must also be aware of the location of files stored within the underlying file directory, which are to be delivered to the user’s device. Upon completion of this form by the educator, the information is stored to the mTest- and service DB4O databases. At this point, the learners may retrieve the newly created assessment material through the system.
To facilitate the educators' interactions with the InfoStation server, two separate handlers’ instances are invoked by the HTTP server – the WNGHandler and the WebIntHandler.

The WNGHandler builds on the underlying WURFL installation, examining the UserAgent of the incoming HTTP request and responding with content adapted to the correct format to suit the requesting environment. The first step in the process undertaken by the WNGHandler is to discern what file is being requested. This is a relatively simple step, except in the case where the requested web content requires adaptation procedures to render the content suitable for the requesting device. Should the WNGHandler detect that the requested file does not exist, it changes the extension on the requested file to either .xml or .wng, to check if the file is contained within another format. To facilitate the identification of the file being requested, the requested path is harnessed from the HttpExchange object utilising the getPath() method. Upon identifying that the requisite file exists within the server’s file system, the handler obtains the extension of the file. The extension is utilised to identify the file to the requesting client browser. If the file has the extension .mp3, .mp4 or .pdf, the handler must specify the media type. The browser can then utilise the requisite applications to open the received file on the client’s local machine. The file is read from the server’s file system and written to the HTTPExchange’s Response body. The content type is specified within the response header sent to the client.

The WebIntHandler is used to facilitate the management interface for educators, as illustrated in Figure 9. This handler operates much like a servlet, i.e. given a specific input in the web interface the handler will perform some operations and responds with dynamically created web content. On receipt of an incoming HTTP request, this handler first determines the type of the incoming request. If the handler receives a simple GET request, then the procedure of locating, adapting and delivering the requested content is quite similar to the process detailed in the previous section. However, the WebIntHandler deviates from the WNGHandler in its handling of POST requests. While the GET requests are simply a request to access a particular page on the server, POST requests involve the passage of some information from the client to the server, along with a request to perform some specific operation. Figure 10 illustrates the index page, identifying the various functions available to the educator, i.e. Get, Store or Delete information from the User, Course, Service, and mTest databases. The WebIntHandler parses the received POST request, harnessing the various elements of information, and discerning the requested function. For example, a request for a particular user may contain the following information:

```
username=Damien.Meere&action=getuser
```

This information can then be broken up by the handler into the following constituent parts:

<table>
<thead>
<tr>
<th>Token</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Damien.Meere</td>
</tr>
<tr>
<td>action</td>
<td>getuser</td>
</tr>
</tbody>
</table>

It is this action which defines how the response is to be formulated. Based on the specified action, one of a number of methods is invoked in order to create the requisite output. For each action, i.e. Getting, Storing or Deleting information from the User, Course, Service or mTest databases, there is an associated Action method. Once the requisite output has been created, the handler sets about placing this content within the HTTP response for the Client, according to the same procedure undertaken by the WNGHandler.

```
Welcome to the TRC InfoStation Web Interface
Please select the function you wish to use:

User Function:
- Get User
- Store User
- Delete User

Course Function:
- Create Course
- Store Course
- Delete Course

Service Function:
- Create Service
- Store Service
- Delete Service

MTest Function:
- Create MTest
- Store MTest
- Delete MTest
```

Figure 10. The educator web interface

5. Conclusions

Within this paper, an InfoStation-based multi-agent architecture, providing an infrastructure to support the delivery of contextualised assessment in the form of mTests was presented. These mTests can be utilised to enhance more traditional localised educational practices, or indeed to provide additional tools to support distance learning. The utilisation of a multi-agent system (MAS) architecture, incorporating the personal mobile devices of learners, was detailed. The main architectural components within the system were discussed, detailing the functionality required in delivering effective assessments to learners, as well as providing educators with a means to dynamically create new mTest material. Some of the future directions in the development of the client-side application environment were detailed, highlighting the incorporation of the HTML5-based Sencha framework. In all, this system seeks to facilitate educators with new tools for interaction with an ever-diversifying cohort of learners, including both the ‘digital-native’ and more mature learners.
6. Acknowledgements

The authors wish to acknowledge the financial support of the Irish Research Council for Science, Engineering and Technology (IRCSET).

7. References


