Published in final edited form as:

Proc SPIE Int Soc Opt Eng. 2008 February 20; 6919: nihpa162285. doi:10.1117/12.770312.

Implementation and use of a web-based interface for confidential communication of data between the clinical and research environments

Peyton H. Bland^a, Gary E. Laderach, and Charles R. Meyer

Department of Radiology, University of Michigan Health System BSRB, Room AC51, 109 Zina Pitcher Place, Ann Arbor, MI, USA 48109-2200

Abstract

Policies and regulations in the current health care environment have impacted the manner in which patient data - especially protected health information (PHI) - are handled in the clinical and research settings. Specifically, it is now more challenging to obtain de-identified PHI from the clinic for use in research while still adhering to the requirements dictated by the new policies and regulations. To meet this challenge, we have designed and implemented a novel web-based interface that uses a workflow model to manage the communication of data (for example, biopsy results) between the clinic and research environments without revealing PHI to the research team or associated research identifiers to the clinical collaborators. At the heart of the scheme is a web application that coordinates message passing between researchers and clinical collaborators by use of a protocol that protects confidentiality. We describe the design requirements of the messaging/communication protocol, as well as implementation details of the web application and its associated database. We conclude that this scheme provides a useful communication mechanism that facilitates clinical research while maintaining confidentiality of patient data.

Keywords

PHI; protected health information; confidentiality; healthcare information management; data security; HIPAA; DICOM; anonymization; de-identification

1. INTRODUCTION

Recent years have seen significant changes in the management of patient data as spelled-out by the rather broad mandates of the Department of Health and Human Services. The most notable example is Health Insurance Portability and Accountability Act (HIPAA)^{1,2} which has affected the manner in which medical staff and medical centers operate with respect to clinical practice and clinical research. In addition, clinical research facilities are influenced by local institutional review boards (IRB). Both are concerned with the proper handling of protected health information (PHI) which may be defined as any information about health and health care that can be linked to an individual. Confidentiality of PHI must be

^abland@umich.edu.

maintained for both legal and ethical reasons, and the adherence to appropriate standards is mandatory for facilities that receive federal funding. Clearly, a new set of procedures is needed to facilitate research while protecting PHI in this new environment. The subject of this paper is a web-based application that aids in this task. The conceptual design of this application is thoroughly described in a companion paper³ and is summarized here as background for a more thorough presentation of implementation of the application and early experience in its use.

Results obtained from a radiologic examination typically include a set of DICOM images (Digital Imaging and Communications in Medicine⁴). Such images contain meta-data relating to the patient, the imaging equipment, the imaging exam, etc. along with the image data. The meta-data (the DICOM attributes, often referred to collectively as the “DICOM header”) are integral with the image data and thus cannot be lost or separated from it, a clear advantage to researchers. However, this comes with a price: In the current environment of confidentiality practices, the information stored in the meta-data - much of it PHI - must be dealt with properly. As a result, to comply with confidentiality regulations, the DICOM images must be de-identified. (This process is sometimes referred to as “anonymization”). In our lab, de-identification is performed automatically by use of a custom-designed DICOM receiver that processes DICOM image files as they are transferred into the lab via the network.

The primary focus of this paper, however, is the communication of PHI that exists *independently* of images. This PHI includes clinical data from the patient interview, physical examination, reports from pathology and surgery, etc. Since radiologic research may involve the use of this information along with image data, the question arises how PHI can be accessed and handled while maintaining confidentiality. In some cases, access to PHI results from direct communication between the researcher and a clinical collaborator. Such communication (which may include verbal and written communication and email) does not guarantee protection of PHI nor can it reliably conceal the identity of patients from the research team.

One problem in dealing with de-identified DICOM images or any other de-identified data is the increased difficulty in matching de-identified clinical PHI with research results that are related to the de-identified data. This separation, however, works in the favor of increasing the integrity of the research since it prevents the researcher from being influenced by knowledge of patient information, clinical data, diagnostic outcomes, etc. If de-identified PHI and research results are matched in a blinded fashion, integrity can be maintained.

As a solution to the issues described above, we have designed a novel scheme built around a web application that we call “Web-CAP”, Web-based Communications Application for PHI. Web-CAP is designed as a tool to assist in the process of communicating de-identified PHI between the hospital and research environments by implementing a secure, well-defined, controlled protocol that improves upon *ad hoc* procedures sometimes used to manage PHI. Proper use of Web-CAP and the systems that surround it not only facilitate the conduct of research but also facilitate the adherence to federal and local regulations. As such, it can

potentially assist in obtaining local IRB approvals by demonstrating to an IRB that proper safeguards for handling PHI are in place.

2. DE-IDENTIFYING DICOM IMAGES

Since DICOM images are at the center of our research, as well as being obvious carriers of PHI, we designed Web-CAP to coordinate with and be consistent with the de-identification process that we routinely perform on DICOM images. Our data typically originate in the PACS maintained by the Department of Radiology in our hospital. Each DICOM image coming into our lab is modified by a custom-designed, de-identifying DICOM receiver³ as illustrated in Figure 1. DICOM images arrive in our lab via a restricted/private network connection and are processed by the de-identifying DICOM receiver. (We implemented this receiver by modifying the application known as `simple_storage` furnished as a part of the DICOM Central Test Node (CTN) software suite developed at the Mallinckrodt Institute of Radiology⁵.) This modified, de-identifying DICOM receiver interfaces with a database maintained in our lab (implemented using MySQL⁶). Most importantly for the purpose of this discussion, it stores the correspondence between various pieces of PHI removed from the incoming DICOM files and the clear-text values substituted in their place by the de-identifying receiver. This is indicated by the correspondence table shown in the figure. PHI stored in the database is stored in encrypted form. The de-identified DICOM file is written to a disk array on the fileserver in our lab.

Of particular interest to the work described here is de-identification of the patient's hospital ID which is stored in the "Patient ID" DICOM attribute as shown in Figure 1. The value of this attribute is replaced with a "research ID", a unique identifier that is assigned by our customized DICOM receiver. The original value of the Patient ID attribute is encrypted as denoted by "encrypt(7654321)" and is stored with its corresponding research ID in the correspondence table of the database.

3. WEB-CAP SYSTEM DESIGN

Using this de-identification process as a starting point, we can now specify a general mechanism to manage the communication of non-image PHI between the hospital and the research lab; refer to our companion paper³ for a more complete description. Implementation and usage details are given later in Sections 4 and 5.

In the text that follows, we consider PHI to exist in one of two forms. When PHI exists in its original form in the hospital, it is typically a piece of information associated with a particular patient via an identifier: for example, "The patient having a hospital ID of 7654321 has negative biopsy results." However, in order to comply with confidentiality requirements, this information can only exist in the research setting as de-identified PHI: for example, "The patient having a research ID of 1234 has negative biopsy results."

Consider our environment to consist of two parts: On the hospital side, the patient ID identifies a patient, and on the research side, the research ID identifies a patient. Web-CAP acts as an interface between the two sides, translating between the patient ID and the research ID using the encrypted correspondence table that is managed by the DICOM de-

identification process as shown in Figure 1. These IDs are used by Web-CAP in the messages passed between users on the clinical and research sides. The researcher sees and has knowledge of only the research ID, and the clinical collaborator sees and has knowledge of only the patient ID. With these identifiers, users of the system interact via Web-CAP over a secure, controlled protocol.

Figure 2 illustrates the essential parts of the process. We continue with the Figure 1 example for the patient identified by research ID “1234”. Suppose that this patient’s images (which have already been received in the lab and de-identified) are being used in a research project, and the researcher needs to obtain biopsy results for this patient.

Following login/authentication, the message exchange or transaction begins with the creation of a request message by the researcher (Figure 2a). (For clarity, only simplified versions of the messages are shown. In addition, due to space constraints, the presentation format of the forms and messages shown in Figure 2 is much more compact than the actual presentations shown by the screenshots in Section 5.) The steps for creating a request message are as follows:

- Step 1: The user enters information (the patient’s research ID and the request) into the request form and submits it.
- Step 2: The information contained in the form is stored in the database.

(Note that the box labeled “Database” represents both the DICOM de-identification database and Web-CAP’s database. Whether it refers to de-identification or Web-CAP or both depends on the context.)

At some later time, a clinical collaborator for the research project sees the message sent by the researcher as an entry in his/her worklist - a table generated by Web-CAP (Figure 2b) containing all outstanding requests addressed to a particular collaborator. The collaborator responds to the request as follows:

- Step 3: Web-CAP creates the worklist for the collaborator. Note that creating the worklist involves accessing the database to retrieve request message content as well as accessing the encrypted correspondence table to translate the research ID to the patient ID.
- Step 4: Web-CAP displays the worklist to the collaborator.
- Step 5: The collaborator selects an entry from the worklist.
- Step 6: Web-CAP creates a response form and loads the selected worklist entry into the patient ID and request fields.
- Step 7: To fulfill the request, the collaborator obtains the requested information - most likely, PHI - by accessing the appropriate clinical information system and adds the information to the form.

The response process continues as shown in Figure 2c:

- Step 8: The clinical collaborator submits the response form.

- Step 9: Submitting the response triggers Web-CAP to translate the patient ID to the research ID and to store the translated message in the database.

At some later time, the researcher checks the status of requests that s/he has made; see Figure 2c:

- Step 10: The researcher requests that his/her status information be displayed.
- Step 11: The researcher views the response message (with the ID in de-identified form) in the status table display. This completes the transaction.

Note that the translation between the patient ID and the research ID performed by Web-CAP is automatic and is hidden from both the researcher and the clinical collaborator throughout the process. Further, the identifier field name and content presented to the user are context-sensitive: “Patient ID” is used on the clinical side, and “Research ID” is used on the research side.

4. WEB-CAP IMPLEMENTATION

4.1 Ruby and Rails

Web-CAP is written in Ruby using the Ruby on Rails framework. Ruby, available as open-source software, is a dynamic, object-oriented language⁷ known for its clean syntax and ease of programming. It shares features with the languages Perl, Smalltalk, Python, and Lisp. The intent is that an entire application can be written in this one language rather than multiple languages as is often the case for web application development. Ruby on Rails - or Rails for short - is an open-source, free web application that operates under the Model-View-Controller (MVC) framework. It was developed to ease the creation of web-backed applications⁸ and is thus well-suited for Web-CAP. Rails relies on two fundamental principles: “Convention over Configuration” and “Don’t repeat yourself”. The first means that the developer need only explicitly specify aspects of the application that deviate from a set of default conventions. For example, Web-CAP contains a class called **User** in its model; therefore, by default, its corresponding database table is called **users** without the need for explicit configuration. Note the Rails convention of upper-/lowercase conversion of the leading characters and the use of singular/plural forms. Naming the table **user_info** would require an explicit configuration specification on the part of the developer. The second principle means that information used in the model is located at a single location. For example, rather than specifying database table column names explicitly in class definitions, Rails obtains them from the database itself (dynamically at runtime). Rails provides scaffolding to ease the writing of models and views by automatically providing initial code which is replaced with permanent code as development continues.

4.2 Database

As noted earlier, databases are used for two purposes and thus consist of two independent sets of tables. The first set is used by the de-identifying DICOM receiver (in read/write mode) (Figure 1), as well as by Web-CAP (in read-only mode) for translating between the hospital and research IDs. This table is accessed as indicated by Step 3 in Figure 2b and Step 9 in Figure 2c. The second set of tables (used only by Web-CAP) holds user information, as

well as information that comprises the message exchanges or transactions themselves as indicated by Step 2 in Figure 2a and Step 10 in Figure 2c. Each transaction stored in the transaction table is updated as the transaction progresses (that is, as the messages comprising the transaction are created and viewed). Thus, the state of a transaction can be constructed at any time by a query to the transaction table. Each transaction record contains the content of the various messages along with message status and other “bookkeeping” information. These tables furnish the information from which the worklist and status tables and the response form are constructed. Note that PHI is not stored in the second set of database tables; instead it contains only de-identified PHI and contains research IDs but not patient IDs along with other information used by Web-CAP.

The database used by Web-CAP was designed following the guidelines in Rails. One such feature in Rails is that of database migrations, a feature that eases database-driven application development. Migrations are independent of database: The database description is written once in Ruby and is translated to any common SQL database - MySQL in our case. Migrations also allow different versions of a database to be used, thus allowing a web-based application to be more easily modified under Rails. Further, migrations allow database schema to be changed easily.

The database that is associated with the de-identifying DICOM receiver was not designed under Rails guidelines since the receiver is a pre-existing application that could not be modified. Thus Rails’ automatic naming conventions could not be used for the receiver’s legacy database. This necessitated the extra step of creating a model in Rails that effectively aliases the legacy database table and field names with new names used by Rails.

Reports for monitoring and logging purposes can be easily generated from the transaction table. Any PHI in these reports would, of course, be presented in de-identified form. In addition, a log of all SQL database commands is maintained by Rails from which an audit trail could be constructed if needed. Reports of virtually any kind are can be generated by parsing the log or designing the appropriate database queries based on local requirements.

4.3 Web server

Ruby on Rails includes a number of options for setting up the web server to deploy a Rails application. We chose to use Apache as a proxy to Mongrel9, an open-source HTTP library and web server for use with Ruby.

The software described herein (the de-identifying DICOM receiver, the MySQL database management system, and the Rails Web-Cap application) run on a typical server in our lab. No part of this data communication scheme places undue demands on the server, and it is anticipated that a reasonably active set of research projects could be easily handled by a single server.

Web-CAP uses HTTPS for communicating any information containing passwords or hospital/patient IDs. Further, we operate Web-CAP only on a private subnet maintained by our medical center’s information technology group. Note, however, that this additional layer

of protection would not be available if Web-CAP were used in a multi-institution project with off-campus users.

4.4 User interface

The overall appearance of Web-CAP's pages is defined by the use of an open-source CSS template from Rayk Web Design¹⁰.

Users furnish input to Web-CAP via simple forms generated by Rails in the process of request and response message creation. Widgets on the forms include type-in boxes, drop-down lists, checkboxes, and buttons; hyperlinks are also used for some functions. These types of user interfaces are employed by virtually all applications that are routinely encountered in common use of the web and are thus intuitive to most users. Results are presented to users in the form of tables (worklist and status tables) and confirmation screens (confirmation of requests and responses) generated by Rails. Error messages to the user generally appear in a box at a standard location on each page.

A comment is in order concerning the text input widgets used to form the messages. A simple, single-line type-in box is appropriate for the research ID as shown in Figure 2a. However, the request and the response type-in boxes, in general, need to accommodate larger amounts of text data. Perhaps the most obvious solution is simply to furnish multi-line type-in widgets for free-form text entry by the user; this is what we have implemented in the current version of Web-CAP. This has the advantage of flexibility in terms of information content. However, there is a disadvantage: For example, due to user error, the text might contain PHI or other information that is not appropriate for transmission in clear text between the clinical and research sides. (This type of user error is, of course, also possible with conventional methods of PHI transmittal such as verbal and hand-written communication and email.) It may be possible to design input validation schemes to detect certain types of erroneous or inappropriate data using an approach similar to that described in Gupta, et al.¹¹ As an alternative to free-form text input for the request and the response, forms could be designed on a project-specific basis to contain more specific and restrictive input widgets such as checkboxes, drop-down lists, and radio buttons pre-programmed with acceptable choices.

4.5 DICOM file de-identification

Inasmuch as the DICOM file de-identifier is a component used in the overall process of communicating PHI, any one of a number of de-identification applications could be used in this process; it is not mandatory to use the particular de-identifier that we developed from the Mallinckrodt DICOM CTN software suite. There are, however, two requirements: First, the de-identifier must perform within the stipulations of the local IRB with regard to how it processes PHI that is present in the incoming DICOM images. Second, the de-identification scheme must include some sort of correspondence table (or a similar structure or mechanism) that allows bi-directional mapping between the patient's hospital ID (stored in encrypted form as described earlier) and the research ID as illustrated in Figure 1, Step 3 in Figure 2b, and Step 9 in Figure 2c.

4.6 Porting to other sites

Porting Web-CAP to other sites requires only minimal site-specific configuration. This is due primarily to the standard directory structure employed by Rails. The bulk of the configuration involves the **users** database table: It must be created and populated with user names, roles, and credentials. In addition to a relational database, Web-CAP requires that the Ruby language along with Ruby on Rails be installed on a server. As described above, some sort of DICOM file de-identification system is also needed.

5. USE

The figures in this section are screenshots from simulated sessions with Web-CAP. Each screen presented to the user has the same general appearance as defined by the “header” and “footer” portions of the CSS layout and the main menu bar as shown by Figure 3; however, the screenshots that appear in Figure 4 and onward will be presented in cropped and zoomed form showing only the essential parts of each page. Fictitious user names and patient IDs are used in the simulated sessions.

Figure 3 shows the first step in the process: login/authentication - a required step for all users of Web-CAP. Users are divided into two groups that define their “role”: researchers and clinical collaborators, each having different privileges. An administrative user is also defined.

The following screenshots will illustrate the use of Web-CAP for the scenario illustrated by Figure 2 and Steps 1 through 11 that accompany Figure 2. The researcher begins the message creation process by selecting the desired project, task (create the request), and recipient (a clinical collaborator), the result of which is shown in Figure 4. Valid projects, roles, and tasks for each user are defined in a database table that controls the choices presented to the user in the project, task, and recipient drop-down lists. Next, the research ID and the text of the request to the clinical collaborator are entered in the form via type-in boxes. The researcher presses the “Create New Request” button to store the request in the transaction table of the database (Steps 1 and 2 in Figure 2a). A confirmation screen of the newly created request message is then displayed to the researcher.

Later, perhaps in response to an email notification of the request (if this option was selected by the researcher when the request was created), the clinical collaborator logs into Web-CAP and checks his/her worklist as shown in Figure 5 (corresponding to Steps 3 and 4 in Figure 2b). Note that Step 3 represents the database query that translates the research ID to the patient’s hospital ID that appears in the clinician’s worklist. The clinician sees that there is an outstanding request and clicks on the Transaction ID number (Step 5) which retrieves information from the database from which the response form is created (Step 6) as shown by Figure 6. The request is now ready to be serviced.

The clinician obtains the information needed to fulfill the request (Step 7), enters it in the form, and presses “Create Response” (Step 8 in Figure 2c) to activate processing of the response. A confirmation of the response is displayed to the user. Recall that when the

response is stored in Web-CAP's database, the hospital ID is translated to the research ID (Step 9).

Researcher "r3" logs into Web-CAP later and selects the task "view status" which causes Web-CAP to generate a status table for the researcher's requests (Figure 7; Steps 10 and 11 in Figure 2c). In this simple case, we see that r3's single request has been serviced, and thus the transaction is complete. In general, there could be any number of requests - some completed, some in progress.

As described up to this point, communication using Web-CAP is initiated by the researcher. However, there are situations where the clinical collaborator would initiate a message. For example, as a part of a research protocol, the clinical collaborator is requested to send the research team lung CT exams for a certain class of patient as the cases are read in the department. In addition to having the DICOM images of these exams, the researcher needs to know the smoking history of each subject in the study. Thus, after pushing a set CT images for a particular patient to the researcher's lab, the clinical collaborator would send a message called a "note" via Web-CAP. This message would contain the patient ID (which would be translated to the research ID by Web-CAP by the time it is received by the researcher) along with the smoking history. Thus, the clinical collaborator's message serves three purposes: the researcher (1) knows that a case has been pushed, (2) knows its research ID (which can be used to locate the de-identified images on the lab's disk array), and (3) knows the smoking history of the subject.

Early usage experiences with Web-CAP validate both its conceptual design and its implementation. User interactions with it are intuitive, and each transaction can be completed quickly. Overall, it is anticipated that using Web-CAP to communicate de-identified PHI will incur little or no additional overhead for users compared with conventional methods such as composing email messages or filling-out hardcopy forms. The added value of using Web-CAP is that the communication of de-identified PHI is virtually guaranteed to comply with confidentiality requirements. A further advantage of Web-CAP is that all interactions with the system are stored in the database (in addition to being logged by Rails) so that no additional record keeping on the part of the users is needed to track the status of a request for data.

6. SUMMARY AND CONCLUSION

Regulatory requirements that define the handling of patient data and the concomitant changes in clinical and research practices mandate a new way of carrying-out research that uses clinical data. We have described Web-CAP, a web-based system that facilitates adherence to these regulations by providing a communication protocol to be used by researchers and their clinical collaborators. The protocol allows patient-related data to be communicated without revealing the identity of the patient to researchers and without revealing research identifiers to the clinical collaborators, thus ensuring the confidentiality of patient data, as well as enhancing the integrity of the research. Early usage experiences have demonstrated that Web-CAP is easy to use and imposes little or no additional overhead compared with existing practices. It replaces *ad hoc* methods of exchanging data with a

secure, well-defined, auditable data communication protocol. It is implemented using Ruby on Rails and conventional database technology. We conclude that this scheme provides a useful communication mechanism that facilitates clinical research while maintaining confidentiality of patient data.

ACKNOWLEDGEMENTS

This work was supported in part by NIH/NCI IP01 CA87634, Automated 3D registration for enhanced cancer management; and NIH/NCI IU01 CA91099, Lung image database consortium.

REFERENCES

- [1]. Department of Health and Human Services. 45 CFR Parts 160 through 164, Standards for Privacy of Individually Identifiable Health Information. May 16.2006 United States <http://www.dhhs.gov/ocr/hipaa/guidelines/finalreg.html> (retrieved January 4, 2008)
- [2]. Department of Health and Human Services; National Institutes of Health. HIPAA Privacy Rule and Its Impacts on Research. Feb 2.2007 United States <http://privacyruleandresearch.nih.gov/healthservicesprivacy.asp> (retrieved January 4, 2008)
- [3]. Bland PH, Laderach GE, Meyer CR. A web-based interface for communication of data between the clinical and research environments without revealing identifying information. Acad. Rad. 2007; 14(6):757–764.
- [4]. National Electrical Manufacturers Association. Digital Imaging and Communications in Medicine. Jan 4.2008 DICOM Homepage, <http://medical.nema.org/dicom/> (retrieved Jan 4.2008)
- [5]. Mallinckrodt Institute of Radiology; MIR DICOM Central Test Node Software. Dec 13.2002 <http://wuerlim.wustl.edu/DICOM/ctn.html> (retrieved January 4, 2008)
- [6]. MySQL AB. Jan 4.2008 <http://www.mysql.com/> (retrieved Jan 4.2008)
- [7]. Thomas, D. Programming Ruby. The Pragmatic Bookshelf; Raleigh & Dallas: 2005.
- [8]. Thomas, D.; Hansson, DH. Agile Web Development with Rails. The Pragmatic Bookshelf; Raleigh & Dallas: 2006.
- [9]. Krause R. Rayk Web Design. Jan 17.2008 <http://design.raykonline.com/> (retrieved Jan 17.2008)
- [10]. Shaw ZA. Mongrel: Home. Jan 17.2008 <http://mongrel.rubyforge.org/> (retrieved Jan 17.2008)
- [11]. Gupta D, Saul M, Gilbertson J. Evaluation of a deidentification (De-Id) software engine to share pathology reports and clinical documents for research. Am. J. Clin. Pathol. 2004; 121(2):176–186. [PubMed: 14983930]

Figure 1.

De-identifying DICOM images. This flowchart illustrates how the “Patient ID” attribute is processed when a DICOM image is received by our lab. This attribute holds the patient’s hospital ID when the image file leaves the PACS in the hospital. However, by the time the image file is written to the disk array in our lab, the value of this attribute has been changed to the so-called “research ID”. Note that the database table that stores the correspondence between the hospital and research IDs stores the patient ID in encrypted form.

Figure 2.

Flowchart of Web-CAP showing the life cycle of a request for information as initiated by the researcher. Type-in boxes indicate information that is entered in the forms presented to the users. For simplicity, the messages are shown in abbreviated form; in practice, the messages include additional identifying information. The bi-directional arrow pairs represent queries and responses to/from the database. Part 2a depicts the creation of the request message via a form by the researcher and its storage in the database. Part 2b shows the retrieval of the request by the clinical collaborator and the creation of a response message - again via a form. Part 2c shows the storage of the response message in the database and the subsequent retrieval and viewing of it by the researcher. The Web-CAP application and the database act in tandem as an interface between the clinical and research sides. Note how the identifier field name and content on the request and response forms change depending on the context of the message: "Patient ID" on the clinical side and "Research ID" on the research side.

Figure 3.

The login page for a researcher having a login name of “r3”. (As shown in later screens, the corresponding “real” name is “Researcher Three”.) User messages are displayed in the narrow, gray box at the right edge of the page. While this screenshot shows the entire login page, the screenshots that appear in the figures that follow will be presented in cropped and zoomed form.

The screenshot displays the 'Web-CAP' web application interface. At the top, there is a header with a computer icon and the title 'Web-CAP'. Below the header is a navigation bar with links for 'Main', 'My Account', and 'Logout'. The main content area is divided into two columns. The left column contains a welcome message: 'Welcome : Researcher Three (researcher)'. The right column contains a 'New Request' form. The form includes the following fields: 'Project Name: Liver Ca', 'Recipient Name: doc2' (with a dropdown arrow), 'Research Number: 1234', and a large text area for 'Request Text' containing the text 'Please send the BX results.'. Below the text area are two checkboxes: 'Request Text:' and 'Send an email to the recipient:'. At the bottom of the form is a button labeled 'Create New Request'.

Figure 4.

Creating a request. At this point, the user has already used three drop-down lists on this and previous screens to specify the project (“Liver Ca”), the task (create a request as denoted by the phrase “New Request” on the screen), and the recipient (“doc2”). The user then enters the research ID of the patient for whom information is requested and the text of the request via type-in boxes. Pressing “Create New Request” causes the request to be processed.

Figure 5.

Checking the worklist. The clinician sees that the status of this transaction is “in progress”, and therefore knows that the request is ready to be serviced. Clicking on the Transaction ID number activates this transaction for servicing. Although only one transaction is shown, a worklist could, in general, contain a number of requests.

The screenshot displays the Web-CAP web application interface. At the top, there is a header with a laptop icon and the title 'Web-CAP'. Below the header is a navigation bar with links for 'Main', 'My Account', and 'Logout'. The main content area is divided into two sections. The left section, titled 'Welcome : Doctor Two (clinician)', is empty. The right section, titled 'New Response', contains a form with the following fields: 'Project Name: Liver Ca', 'Transaction Number: 6', 'Request Text: Please send the BX results.', and 'Hospital ID: 7654321'. Below these fields is a large text input area where 'biopsy: negative' has been entered. At the bottom of the form, there is a 'Response Text:' label, a checkbox for 'Send a confirmation email:', and a 'Create Response' button.

Figure 6.

Creating a response. A number of fields are filled-in based on information stored in the transaction table of the database. The user only needs to enter the requested information and press "Create Response".

The image shows a web application interface titled "Web-CAP". At the top, there is a navigation bar with links for "Main", "My Account", and "Logout". Below this, a welcome message reads "Welcome : Researcher Three (researcher)". To the right of the welcome message is a table titled "Status" that displays transaction details. The table has columns for Transaction Number, Type/Status, Project Name, Research ID, Date, Text, Recipient, and Originator. Two rows are shown: a "REQUEST" from Researcher Three to Doctor Two, and a corresponding "RESPONSE" from Doctor Two back to Researcher Three, indicating the transaction is complete.

Status							
Transaction Number	Type/Status	Project Name	Research ID	Date	Text	Recipient	Originator
6	REQUEST	Liver Ca	1234	Friday 18 January 2008 - 14:55 PM	Please send the BX results.	Doctor Two	Researcher Three
6	RESPONSE	Liver Ca	1234	Friday 18 January 2008 - 14:55 PM	biopsy: negative		

Figure 7. Viewing the status of a request. The request has a corresponding response; thus the transaction is complete.