CloudStudy: A web-based system for supporting multi-centre studies

CloudStudy: A web-based system for supporting multi-centre studies

Amalia Tsafara

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master in Computer Science and Technology

Committee

Assistant Professor Christos Tryfonopoulos, Supervisor
Associate Professor Spiros Skiadopoulos, Supervisor
Assistant Professor Nikolaos Tselikas, Member

University of Peloponnese
2014

Abstract

Among the basic research tools for (bio)medical science are epidemiological studies that typically involve a number of hospitals, clinics, and research centres scattered around the world, and are often referred to as multi-centre studies. Clearly, the effectiveness and importance of a multi-centre study increases with the number of participating centres and enrolled patients, but at the same time this natural distribution in the production of research data requires sophisticated data management infrastructures to support the participating units. This kind of infrastructure is not only expensive to build and maintain, but also cannot be reused as it is often tailored to a specific study. In this work, we present a cloud-based system, coined CloudStudy, that allows users without any computer science background to design, deploy, and administer platforms aimed for managing, sharing, and analysing clinical data from multi-centre studies. The CloudStudy system provides a zero-administration, zero-cost online data management tool that *(i)* enhances re-usability by introducing study templates, *(ii)* supports (bio)medical needs through specialised data types, and *(iii)* emphasises data filtering/export through an expressive yet simple graphical query engine. CloudStudysystem has already been used by 20 Greek hospitals and clinics for supporting and managing two multi-centre studies.

Περίληψη

Οι επιδημιολογικές μελέτες είναι ένα απο τα σημαντικοτερα εργαλεία που χρησιμοποιούνται σήμερα στην έρευνα της (βιο)ιατρικής επιστήμης. Οι μεγάλης έκτασης επιδημιολογικές μελέτες που περιλαμβάνουν ένα μεγάλο αριθμό νοσοκομείων, κλινικών, και ερευνητικών κέντρων, τα οποία βρίσκονται διάσπαρτα σε όλο τον κόσμο, ονομάζονται πολυκεντρικές μελέτες. Προφανώς, η αποτελεσματικότητα και η σημασία μιας πολυκεντρικής μελέτης αυξάνεται καθώς αυξάνεται και ο αριθμός των συμμετεχόντων κέντρων και ασθενών. Ο μεγάλος όμως αριθμός ασθενών, συλλεχθέντων δεδομένων καθώς και η μεγάλη φυσική απόσταση μεταξύ των νοσοκομείων, αυξάνουν την πολυπλοκότητα διεξαγωγής μίας τέτοιας έρευνας, με αποτέλεσμα να απαιτεί εξελιγμένες υποδομές για να πετύχουν οι μελετητές τη διαχείριση των δεδομένων, και την υποστήριξη των συμμετεχόντων μονάδων. Το κόστος ανάπτυξης και συντήρησης μίας τέτοιας υποδομής είναι αρκετά υψηλό, και οι προτεινόμενες λύσεις είναι συνήθως προσαρμοσμένες πάνω σε συγκεκριμένες μελέτες, με αποτέλεσμα να μην μπορουν να επαναχρησιμοποιηθούν. Στην παρούσα διπλωματική εργασία, παρουσιάζουμε μια διαδικτυακή εφαρμογή, που ονομάζεται CloudStudy, και επιτρέπει σε χρήστες χωρίς υπόβαθρο στην πληροφορική, να σχεδιάζουν, να δημιουργούν, και να διαχειρίζονται πλατφόρμες που τους παρέχουν τη διαχείριση, την ανταλλαγή και την ανάλυση κλινικών δεδομένων που έχουν συλλεχθεί μέσω πολυκεντρικών μελετών. Το CloudStudy είναι η πρώτη δωρεάν υπηρεσία διεξαγωγής πολυκεντρικών μελετών η οποία είναι αποκλειστικά web-based, δεν απαιτεί διαχείριση, έχει μηδενικό κόστος για τους συμμετέχοντες, υποστηρίζει τη δημιουργία και επαναχρησιμοποίηση μερών ή ολόκληρων ερωτηματολογίων, υποστηρίζει εξειδικευμένους τύπους δεδομένων και παρέχει ένα ευέλικτο και φιλικό στον χρήστη εργαλείο φιλτραρίσματος και εξαγωγής πληροφορίας μέσα από τη βάση δεδομένων. Το CloudStudy χρησιμοποιήται ήδη από 20 νοσομεία και κλινικές στην Ελλάδα για την υποστήριξη πολυκεντρικών μελετών.

Acknowledgements

At this point I would like to thank Assistant Professor Christos Tryfonopoulos for his guidance, expert advice, and encouragement during the elaboration of this master thesis. I would also like to thank the members of my committee, Assosiate Proffesor Spiros Skiadopoulos and Assistant Professor Nikos Tselikas for their advices, comments and suggestions on improvements of this work.

I would like to thank my family for their love, help, encouragement and support during all these years.

Also, I would like to thank the Software and Database Systems Lab at the University of Peloponnese for the resources supplied in order to complete my research. Finaly, I would like to thank University Hospital Attikon and Greek Chemotherapy Group who instructed me on the creation of a platform for supporting a multi-centre study. Special thanks go to Dr. Garifalia Poulakou who helped me understand the needs of the researchers and clinicians when carrying out a multi-centre study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis presents CloudStudy, a web-based system for supporting multi-centre studies. CloudStudy solves problems that are caused by the management and analysis of medical data, while carrying out large scale multi-centre studies. In this chapter we define the problem, outline the existing solutions, and briefly summarize our solution and contributions.

1.1 Problem Statement

In this section we define the problem, present the basic points in multi-centre studies and point at the difficulties of carrying out these studies.

Epidemiology is the study (or the science of the study) of the patterns, causes, and effects of health and disease conditions in defined populations. It is the cornerstone of public health, and is used to determine policy decisions and evidence-based practice by identifying risk factors for disease and targets for preventive healthcare. Epidemiologists are responsible (i) for the study design, (ii) collection and statistical analysis of data, and (iii) interpretation and dissemination of results. Epidemiology has helped the development of a methodology that is used in clinical research, public health studies and, finally basic research in the biological sciences.

Major areas of epidemiological studies include disease etiology, outbreak investigation, disease surveillance and screening, biomonitoring, and comparisons of treatment effects such as in clinical trials. Epidemiologists rely on other scientific disciplines like biology to better understand disease processes, statistics to make efficient use of the data and draw appropriate conclusions, social sciences to better understand proximate and distal causes, and engineering for exposure assessment.

Epidemiological studies become more effective as the stakeholders and the target population grows. The large-scale epidemiological studies are called *multi-centre studies* and they typically involve a number of different stakeholders, including hospitals, clinics, and research centers, physically distributed around the world. The effectiveness of multi-centre studies increases with the number of participating research centers and patients. Multi-centre studies are invaluable as they collect large amounts of data from different regions, correlate them, and draw useful conclusions on important research questions. However, the physical distribution of the participants and the asynchronous nature of data acquisition pose a number of issues including the collection, organisation, and processing of data.

As we mentioned above a multi-centre study involves a number of physically distributed stakeholders where each involved member (that could be a doctor, a nurse, a researcher, etc.) has to gather the needed information for each participating patient. So the first issue posed by carrying out multi-centre studies is how all these involved members can be tuned to collect and organise this large amount of data in a common way, overcoming the distance between them, that has as a result for the collected patient data to be naturally distributed and produced asynchronously. So the organisation, the processing and the analysis of the collected data becomes a complex and time consuming task. The next issue arising from multi-centre studies is how this large amount of data can be stored and organised in such way to achieve the most efficient data processing, analysis and querying.

1.2 Existing solutions

To tackle data fragmentation and address the issues arising from the coordination of geographically distributed participants, a number of platforms, that focus on the storage and management of (bio)medical data, have been previously proposed [1–5]. However, all these platforms are either designed for a *specific task or study* [1, 3, 6–8] (and are thus unusable in any other study), or require significant *computing infrastructure* and an *expert* in Information Technology (IT) for setup and tuning [4].

Thus, all the existing approaches require either time-consuming meetings between scientists of different principles trying to understand each other's needs, or resource-consuming IT infrastructure, that requires outsourcing to IT specialists and regular maintenance/upgrades to keep up with technological requirements. Due to these issues, a great number of multi-centre studies that lack the resources are still performed by resorting to *manual procedures*, such as collecting data on paper, exchanging data by post, or emailing enormous spreadsheet files with patient data. Therefore, concerns like coordination among participants, data freshness/integrity, control of participants' involvement, and timeliness of results are lost between versioning in exchanged spreadsheets, hard copies of patient data, and requests for participation.

1.3 Contributions

A major part of the systems implemented to support multi-centre studies, are developed for supporting single or specific studies and cannot be reused by other studies in an efficient, timely and cost-effective way. In this work, we concentrate on how we can create electronic health systems for *managing, sharing, organising, analysing and queering* clinical and patient data from *multi-centre studies.* The proposed system, coined CLOUDSTUDY [9, 10] covers all the functional requirements posed by multi-centre studies, and enables researchers to easily organise and share data and knowledge generated by the research activity. We propose an innovative, integrated framework for creating platforms for multi-centre studies that enables users with *no prior IT knowledge* to:

- Design and launch, in an easy and transparent way, platforms tailored to the specific needs of their studies.
- Perform basic and advanced user management tasks (manage users, assign user privileges and permissions, perform access control on data).
- Record, organise and manage clinical/patient data by resorting to a number of built-in and customisable data entry forms.
- Search and filter information by using a powerful yet simple point-and-click mechanism that poses restrictions on the stored data and extracts the re-

quested information in a number of formats/outputs including raw data, pie/-column charts, and ready-to-process spreadsheets.

- Reuse older data collections to new studies.
- Update platforms even if they are in use.
- Update a platform and adapt it to a new study without any cost.
- Update platforms already in use, and adapt to the needs of new studies, while maintaining the initial study operational.

In the light of the above, the contributions of this thesis are the following:

- We propose a *web-based, zero-cost, zero-administration* tool that offers both fundamental and advanced user and data management functionality for multi-centre studies. To the best of our knowledge, this is the first web-based system that focuses on multi-centre studies and allows users to deploy their own data management platforms within minutes, alleviating the need to rely on expensive custom-made solutions that require IT infrastructure and skills to maintain.
- We present the architectural considerations and solutions behind the proposed tool, and describe a number of *novel services* that allow users without any prior IT knowledge to create, administer, launch, and use personalised data management platforms.

1.4 Application scenarios

In this section we present some application scenarios to provide a better understanding of the underlying idea and novelty that is brought by CloudStudy and highlight its key differences from existing approaches. For the needs of the application scenarios, let us consider three research groups (namely ABC, Bio, and Med) located in three geographically distributed hospitals. These hospitals have decided to carry out an epidemiological multi-centre study for Infectious Endocarditis. The Group ABC is the coordinator group, i.e. the group that has decided to lead the multi-centre study, and is responsible to set up the data management platform that will facilitate the collection and processing of the data.

Scenario A:

In the first possible scenario Mary (working for ABC), who is the person in charge of the setup of the data management platform, gets in touch with an IT company, and explains to an IT manager the needs and specificities of the multi-centre study. After a long period of time spent between financial offers from different parties and long user requirements meetings, the IT company will deliver the developed software. The ABC group will then need to buy and maintain a costly IT infrastructure on-site to host the developed solution or outsource the hosting in some appropriate third-party company. Also the group will possibly need to hire an IT professional to keep both the system and the infrastructure up-to-date. It is clear that such an approach would require both a lot of time and plenty of resources that will be used only for the specific study. Any subsequent study coordinated by ABC will result in a new process to acquire the customised software. Moreover groups without so many resources are left out from the option to lead such studies.

Scenario B:

In the second scenario Mary decides to resort on one of the free systems (e.g., REDCap [4]) offering data management services. At the beginning Mary will be asked to take a license for the REDCap software, she will design a platform for supporting the study, and she will submit it. After the platform's submission, the REDCap administrator will review and approve the platform, and he will send an e-mail confirmation to Mary, confirm that her group is legible and meets all requirements for the acquiring of the platform. The same procedure has to be followed if Mary decides to make any platform changes while it is in use. While seems more time and cost efficient than the previous scenario, still Mary needs to cooperate with a third party to develop the needed database. This process is sensitive to delays and is not easily customisable in terms of the supported questions.

Scenario C:

In the next scenario Mary and her group will use CloudStudy with which they would benefit from accessing a web-based service that would allow them to create and deploy such a platform in a fast, free, and effortless way. This system would be a valuable tool beyond anything supported in current solutions, that would allow Mary and her group to save time, effort, and resources. After the platform creation, Mary will be able to create and manage the users of the deployed platform, as well as define access control policies. These users will then be able to login and input data, filter and export them and create charts for optimizing the collected patient data through the platform from any location or device. This is the first system that is able to support so specialised and complex needs without the need of an IT expert.

1.5 Thesis Organisation

This thesis is organised as follows. In Chapter 2 we introduce Health Information Technology and the usage of electronic health systems in multi-centre studies and discuss related works in the field. In Chapter 3 we describe the system architecture and functionality, and in Chapter 4 we provide details on the implementation of CloudStudy. Finally in Chapter 5 we present our conclusions and propose directions for research.

Chapter 2

Related Work

A few years ago (in some cases even today), doctors and nurses used to keep hand-written notes about patients during their visits. These notes would be added later to the patient's medical file. Today, the world is full of digital devices - like smart-phones, tablets, laptops, desktops, and this makes storage and retrieval of patient information a relatively fast and easy procedure. With the development of technology, new methods created (software, digital devices) for paper-less storage of patient information were created. Today, Health Care Providers are making significant efforts to adopt the Health Electronic Systems to hospitals, clinics and medical centers in order to use Electronic Health Systems to store their medical data.

2.1 Health Information Technology

Health information technology (HIT) provides a framework to describe the management of health information systems. Health information technology (HIT) is a very promising tool for improving the quality, safety and efficiency of the health delivery system. The advantages HIT usage are a lot, such as the health care quality and effectiveness improvement, and the improvement of health care productivity and efficiency. Also, with the usage of HIT clinicians achieve the prevention of medical errors, and the increasement of health care accuracy and correctness. Finally, it is very hopeful that HIT decreases paperwork and unproductive work time, reduces health the care costs, as well as it makes the access to health care more affordable.

2.1.1 Electronic Health Records

An electronic health record (EHR) is a collection of electronic health information about an individual patient or a group of patients [11]. It is a digital record that is able to be shared across different health care stakeholders. In some cases this sharing can occur by means of network-connected, enterprise-wide information systems or other information networks. An EHR system usually contains functions like demographics, medical history of patients, diagnoses, medications, treatment plans, immunization statuses, allergies, radiology images, laboratory and test results, patient demographic like age and weight or maybe billing information.

EHR systems are designed to store patient data during the time. With EHRs doctors and nurses achieve fast and easy access to patient history without the time-consuming searches for patient records in archives, and ensures data accuracy and legibility. Their usefulness lies in the fact that users modify the digital patient record and do not need to keep old-versions of files, this has as result to eliminate issues such as lost forms or paperwork. Due to the fact that all the information are gathered in a single patient record, it makes the medical data extraction much more effective for the examination of possible trends and long term changes in the patient. EHRs are designed to reach out beyond the health organization that originally collects and compiles the information. They are built to share information with other health care providers, such as laboratories and specialists, so they contain information from all the clinicians involved in the patient's care.

During the last years, a big number of EHRs have been developed and exist in the health care market. Some of them [12–15] aim at helping users to collect, sort, archive, explore, analyse, export, and organise raw data like medical images scanned documents, laboratory data, and clinical ratings. Apart from data storage the offered functionality may also extend to quality control, and data analysis on the stored information to support clinical decision making [16] or promote medication adherence [17].

2.1.2 Electronic Medical Records

Another type of electronic health systems are Electronic Medical Records (EMRs) that are digital versions of the paper charts that are used in hospitals. An EMR may keep the medical and treatment history of a patient. EMRs have obvious advantages over paper records. One of the advantages is that they are able to track patient data over time, something very useful for patients with chronicle diseases. Another advantage is that these systems may keep patient examination, medical and treatment history, this make them able to remind clinicians/patients for pending checkups, compare the new examinations with the old ones, and alert the clinician for any undesirable changes. Also a system like this can monitor and improve the overall quality of care within the practice.

The main differences between EHR and EMR systems is that using EMRs the information moves with the patient to the specialist, the hospital, the next state or even across the country. The EHR represents the ability to easily share medical information among stakeholders and to have a patient's information follow him or her through the various modalities of care engaged by that individual. EHRs are designed to be accessed by all people involved in the patients care. And that makes all the difference, because when information is shared in a secure way, it becomes more powerful. Health care is a team effort, and shared information supports that effort. After all, much of the value derived from the health care delivery system results from the effective communication of information from one party to another, and the ability of multiple parties to engage in interactive communication of information. Some of the most popular EMRs are eClinicalWorks[1], McKesson[2], Cerner[3], Allscripts[4], Athenahealth[5], EG Healthcare[6], Epic[7], Practicefusion[8].

[1]www.eclinicalworks.com/
[2]www.mckesson.com/
[3]https://www.cerner.com/
[4]www.allscripts.com/
[5]www.athenahealth.com/
[6]www.philippinecompanies.com/
[7]www.epic.com/
[8]www.practicefusion.com/

2.1.3 Personal Health Records

Another big category of health systems is Personal health records (PHRs). PHRs are software systems that allow patients to store, edit and organize their own health and medical information with or without their doctor's participation. Some PHRs are part of an EHR, while others are independent.

During our research we found several PHR systems and the most of them refer to patients with chronic diseases. These systems help patients to manage their medical data. COPD[18] (designed for people with obstructive pulmonary disease) is an example of a PHR system that predicts and detects exacerbations and, through this, help patients self-manage their disease, and prevent hospitalization. Other popular systems include HealthDesign [9] that focus on patients with heart failure, as well as PHA[19] that focuses on diabetic patients and helps them manage their own medical data. Another notable PHR that is worth mentioning is MyMediHealth[20] that aims at improving the safety and quality of medication delivery in child population. ChronoMedIt[21] is another example of PHR system that can reliably execute a wide range of clinically useful queries to identify patients whose chronic condition management can be improved. Finally Colorado care tablet [22] that assists older adults in managing and sharing medication regimen across care transitions.

2.1.4 Electronic Prescribing

Another important and useful category of health systems are Electronic Prescribing (e-Prescribing) systems like MedicsRx[10], MicroMD[11], ScriptSure eRx[12], Eprescribing[13] that give the ability to doctors to electronically prescribe medications and automatically order them at the pharmacy on behalf of the patient. This simplifies the drug ordering procedure, avoids over-the-counter mistakes, and ensures availability at the pharmacy. E-Prescribing may be implemented as a stand alone system[14]

[9] `www.healthdesigns.com`
[10] `http://www.adsc.com`
[11] `http://www.micromd.com/`
[12] `http://www.dawsystems.com/`
[13] `http://www.eprescribin.com/`
[14] `http://mdtoolbox.com/`

or as a separate module of an EHR[15].

2.1.5 Health prediction systems

Finally we have to mention that during our research we found a big category of health care systems (EHRs and PHRs) that except from storing and organizing the medical data, are also capable to make predictions. COPD[18] is one prominent example of an EHR systems that is able to detect/predict exacerbations and, thus help patients self-manage their disease in order to prevent hospitalization.

that designs technology to fit people's characteristics. Also there are many systems [23–25] that specialize to make predictions in order to help physicians decide which medicine care is the most affective for each unique situation. [23] apply algorithms in a group of patients and tries to identify which patients are most likely to benefit from situations such as rehabilitation, manage a rehabilitation treatment plan, and monitor progress both during rehabilitation and after return to a community residence. Another prediction system is [24] that refers to patients with diabetes and makes predictions for which organizational units the patient has to move to meet the right agents, in order to receive the health services she/he needs and to communicate information about her/his health status and results of tests and visits. Another notable prediction system is[25] that is able to make detailed predictions, about patient survival, disease progression, the effect of a treatment, as well as the development of complications. Finally Health-e-Child [26], is a health system that aims to develop an integrated healthcare platform for helping European paediatrics on taking decisions for each child health care.

2.1.6 Benefits of using Health Information Technology

To sum up, the usage of HIT in health care allows easy access of all healthcare stakeholders to the latest patient information enabling a coordinated patient-centered care. With the usage of HIT in public health: (i) the doctor can be informed about each detail of the patient health even if the patient is unconscious, (ii) a patient may access to his own record and see the history of the lab results over the last year,

[15]`http://www.bcbsnc.com/`

which may help motivate him to take his medications or keep up with the lifestyle changes that have improved the numbers, (iii) the lab results are readily available to the specialist to examine them, (iv) the clinician's notes from the patient's previous hospital stays may help formulate the discharge instructions and follow-up care and enable the patient to move from one care setting to another in a smooth and transparent way, (v) allows for the early detection of medical conditions, (vi) the clinicians can easily track, manage and understand better a chronic disease, (vii) patient data collected by health care systems can be evaluated.

2.2 Electronic Health Systems and Multi Centre Studies

The aforementioned systems were developed with purpose to improve the patient health care. In this section we focus on EHRs that were developed in order to support specific multi-centre studies and draw conclusions.

2.2.1 Systems designed for a specific task or study

The most of the proposed health systems that are used in multi-centre studies, focus on a *specific study*, and they put forward architectures and services tailored to the problem at hand. For example, [27] is an information system for multicenter epidemiological studies on cancer with capabilities of reusing the data collected during the achievement of epidemiological studies: The system stores the data gathered during interviews to patients and their relatives in a centralized database, and uses the defined mappings to answer remote queries written in the terminology of the ontology. In addition, the information system is able to use the ontology as a reference to query remote data sources in an integrated manner.

Similarly, MSBase [28] is a powerful tool to assess the long-term outcome in chronic diseases. The web platform MSBase has been designed to collect prospective data on patients with multi sclerosis (MS). MSBase facilitates collaborative research by allowing the online creation of investigator-initiated regional, national and international substudies. The registry aims to answer epidemiological questions that can only be addressed by prospective assessments of large patient cohorts.

Another worth to be mentioned project is COBRED[8] aims at discovering colon cancer and breast cancer biomarkers for patient follow-up by exploiting the capacity of three state-of-the-art high-throughput technologies in an integrated systems biology approach. The goal is to identify biomarker candidates (metabolites, proteins, PBL derived mRNAs) capable to detect and assess the status of minimal residual disease, metastases and recurrence after surgery and chemotherapy.

HIV/AIDS has become the world's leading infectious cause of adult deaths and takes its greatest toll in remote, resource poor areas. Dramatic improvements in survival have been seen with use of antiretroviral drugs in developed countries and in Brazil. Since 2001, substantial resources have been pledged to treat HIV infected patients in developing countries, but concerns have been expressed that many such countries lack the infrastructure to support the complex treatment regimen for this chronic disease. [3] describes approaches to improving important infrastructure components for HIV treatment in very impoverished areas. The proposed system supports clinical communications, data analysis, and drug supply management. CASCADE[16] is a project that is a collaboration between the investigators of 29 cohorts of people with well-estimated dates of HIV seroconversion, involving a network of epidemiologists, statisticians, virologists, and clinicians from lead HIV institutions in 14 European countries, as well as Australia, Canada, and Africa. Seroconverters are enrolled into the individual cohorts locally and nationally and are typically followed up life-long. CASCADE's main aim is to monitor newly-infected individuals and those already enrolled in studies, covering the entire duration of HIV infection. Another project that aims to help researchers to study about HIV/AIDS prevention and treatment is is HICDEP [7]. DREAM[1] is another project that operates within the framework of the national health systems of several sub-Saharan African countries and aims to introduce the essential components of an integrated strategy for the prevention and treatment of HIV/AIDS. Specific software for the management of the patients' EMR has been created within the DREAM program in order to deal with the challenges deriving from the context in which DREAM operates. Over the years this EMR has been used in 10 countries in sub-Saharan Africa by thousands of professionals and by now it has reached its fourth version. The medical files of

[16]`http://www.ctu.mrc.ac.uk/cascade/`

over 73,000 assisted patients are managed by this software and the data collected with it have become essential for the epidemiological research that is carried out to improve the effectiveness of the therapy.

XML is a more desirable format for modeling and storing clinical data in EMR (Electronic medical record) applications for its extendibility; however, existing EMR systems either are built on top of RDBMS or file systems or lack of support for complex and large scale healthcare applications, such as treatment effectiveness analysis and procedure optimization. Xbase[29] system is built on top of Hadoop, and it is the first XML-based information appliance designed specifically for large scale and complex healthcare applications. XML presents a different set of challenges for query processing, indexing, parallelism, and distributed computing using existing Hadoop's APIs as well as its HDFS storage infrastructure and MapReduce framework.

Simulation provides a powerful framework for understanding biological form and function, and can be used by biologists to study macromolecular assemblies and by clinicians to design treatments for diseases. Simulations help biomedical researchers understand the physical constraints on biological systems as they engineer novel drugs, synthetic tissues, medical devices, and surgical interventions. Simbios[30] is a Simulation of Biological Structures (Simbios) to help integrate the field and accelerate biomedical research. Simbios focuses on problems at both the molecular scale and the organismal level, with a long-term goal of uniting these in accurate multiscale simulations.

To investigate molecular-biological causes, effects of diseases, and their therapies it becomes increasingly important to combine data from clinical trials with high volumes of experimental genetic data and annotations. [31] presents an approach to integrate such data for two large collaborative cancer research studies in Germany. The proposed platform interconnects a commercial study management system (eRN) with a data warehouse-based gene expression analysis system (GeWare). It utilizes a generic approach to import different anonymized pathological and patient-related annotations into the warehouse. The platform also integrates different forms of experimental data and public molecular-biological annotation data and thus supports a wide range of genetic analyses for both clinical and non-clinical parameters.

In [32] are presented two components that allow non-technical users that are domain experts to create and reuse complex data integration processes. The GUAVA component enables data analysts to make informed data integration decisions based on detailed accounts of the user interface that was used to generate the data. The MultiClass component allows analysts to revisit decisions made for prior studies and reuse them or not each time the data is used.

Concluding, as we see the above systems are designed for a *specific task or study*, and are thus unusable in any other study.

2.2.2 Systems that create platforms for supporting multi-centre studies

The large number of existing specialised systems and the needs dictated by each different multi-centre study led researchers to the design of systems that are able to support *classes* of functionalities needed in multi-centre studies. In this subsection we present the most prominent paradigms in this line of work: the REDCap project [4] and the Qure system [5]. These two systems are designed to create platforms for multi-centre studies and are the most related to CloudStudy system.

The REDCap project

In this paragraph we present REDCap system and compare it with CloudStudy. REDCap (Research Electronic Data Capture), is a secure web-based application for building and managing online research surveys and databases. The goal of REDCap is to achieve collaboration and easy data management. REDCap can be used to rapidly develop projects via a point-and-click interface or by creating a *data dictionary* template file in Microsoft Excel that can easily be uploaded into the application. With REDCap a researcher may build and manage data entry forms, online surveys and associated databases. REDCap provides the *Online Designer* that is an online, interactive tool for designing the survey platforms through a web-browser. The Online Designer supports question reorganization through drag-and-drop actions, branching logic, standard question datatypes (text box, note box, radio-buttons, multiple-choice, check-boxes, titles and file upload buttons) like CloudStudy, but

CloudStudy provides a more flexible questionnaire designer as it also offers custom drop-down lists and complex data types. Also REDCap supports an offline, bulk method by constructing a template file in Microsoft Excel and uploading into REDCap. With REDCap users may create, modify and deploy the platform content without having to perform any programming or database development, but the difference with CloudStudy is that in REDCap the changes have to be reviewed and accepted by REDCap administrator, while in CloudStudy the changes are applied immediately. Some other applications that REDCap offers are: (i) a calendar that keeps important events, (ii) an export tool where the user may check the questions to export, while CloudStudy provides a flexible filtering tool that allows advanced export and filtering, and (iii) a visualization tool that presents a chart for each question.

The Qure System

Qure Data Management is a tool similar to CloudStudy, that allows users to design platforms for supporting research studies. In this paragraph we present Qure system and we compare it with CloudStudy. The Qure system provides a tool for the questionnaire using input controls, controlled vocabularies and ontologies, dynamic behavior, and scripting. The designed projects are uploaded to the Qure Server and distributed to Qure Desktop and Qure Browser applications. Qure Desktop is a tool provided by Qure and supports on-line and off-line data entry and management. Each system is able to support many studies at the same time. Desktop tool offers has simple user interface and helps to record data when the computer is off-line. Qure Server is a a server installation for the platform. It includes user friendly modules for users, projects, data, software management, etc. Qure Browser is a module that provides data entry, management, and browsing as the stand-alone Desktop tool. Qure DataView is a server module for querying and reporting on the database content. It is a graphical interface for queries with filtering and aggregation support and end-result specification with multiple data output options (HTML, CSV, PDF). Finally, Qure Reports are specially designed reports that presents statistical overviews, graphical elements, etc. The main advantages of CloudStudywhen compared to Qure, is that it supports complex data types, recurring questions and drop-down lists in the questionnaire design, questionnaires that are created from CloudStudy more flexible, effective, and accurate when compared to those of Qure system. Also another part that Qure is disadvantaged compared is that Qure Desktop module runs on dedicated software with no adaptation policy (e.g., elasticity of resources) while CloudStudy provides elasticity as it runs on a dedicated cloud software(provided by owncloud).

Chapter 3

The CloudStudy System

In this chapter we present CloudStudy, an innovative, integrated web-based framework that we developed aiming at creating platforms for multi-centre studies. The main idea of CloudStudy is to allow users design and build data management platforms with which they will be able to collect, organise and analyse (bio)medical and patient data. These platforms will be created through a series of simple and adaptive processes. This can be done transparently through simple-to-follow wizards from users without any IT training, while the web-based architecture automatically makes CloudStudy accessible anytime from any device.

Initially, we give a description for the concepts *question, platform* and *template* that will be used in the description of the system. *Questions* are the backbone of epidemiological multi-centre studies. To support the set of questions of a particular multi-centre study, CloudStudy introduces *platforms*.

Technically, a platform is a framework that consists of a custom-made set of questions (questionnaire), and the necessary user administration and data management components of a study. Platforms and templates are handled through Platform Manager. Platforms are typically created and administered by Platform Administrators. Very often, specific parts of a study may be used (as they are or with minor modifications) in other studies. For example demographic data are a typical reusable part of many biomedical multi-centre studies since they are not expected to change among different studies. To support this reusability, CloudStudy offers Platform Administrators the possibility to create *templates* that can be used across different platforms.

3.1 System Architecture

In this section we present the architecture of CloudStudy and we describe in detail each component. For the development of the system we used the Linux/Apache/ PostgreSQL/PHP (LAPP) framework as the backend database infrastructure, while the rest of the modules have been developed using HTML, Javascript, PHP, and jQuery tools. CloudStudy is a web application, and it is installed and setup over a medium sized web server available at the University of Peloponnese.

The CloudStudy architecture consists of six main modules. As we see in Figure 3.1, the first module is the User Interface, which is responsible for creating the application interface, identifying the hardware that is used to connect to CloudStudy (PC/ tablet/ smartphone) and adjust the viewing components accordingly. The tools used for this module's development are the CSS, HTML, Javascript, PHP and jQuery tools. The User Interface module communicates with the Platform and Platform Manager modules to create the interface of each component.

The Platform module is responsible for the storing, updating, and management of patient data, and consists of a number of modules utilised to *(i)* manage the Participants and the stored data associated with a study and *(ii)* filter data with the filtering tool requested by a user. The most important Platform's components are the Record Manager, the Filtering module and the Chart module. The Record Manager is responsible for the creation, updating and deleting of the patient records. The Filtering module provides users with the Filtering Tool, a flexible tool with which a non-IT user may extract complex patient information from the database through a user-friendly interface with simple point-and-click interactions. Finally, the platform module provides users with the Chart Tool which is responsible for creating complex charts of the stored patient data. The Platform module communicates with the DB Manager to retrieve and store the patient data, with the User Manager to manage the Participants users, with the User Interface module to create and adjust the platform interface, and with the Security and Control module to ensure the proper operation of the system.

The next module of CloudStudy is the Platform Manager. The Platform Man-

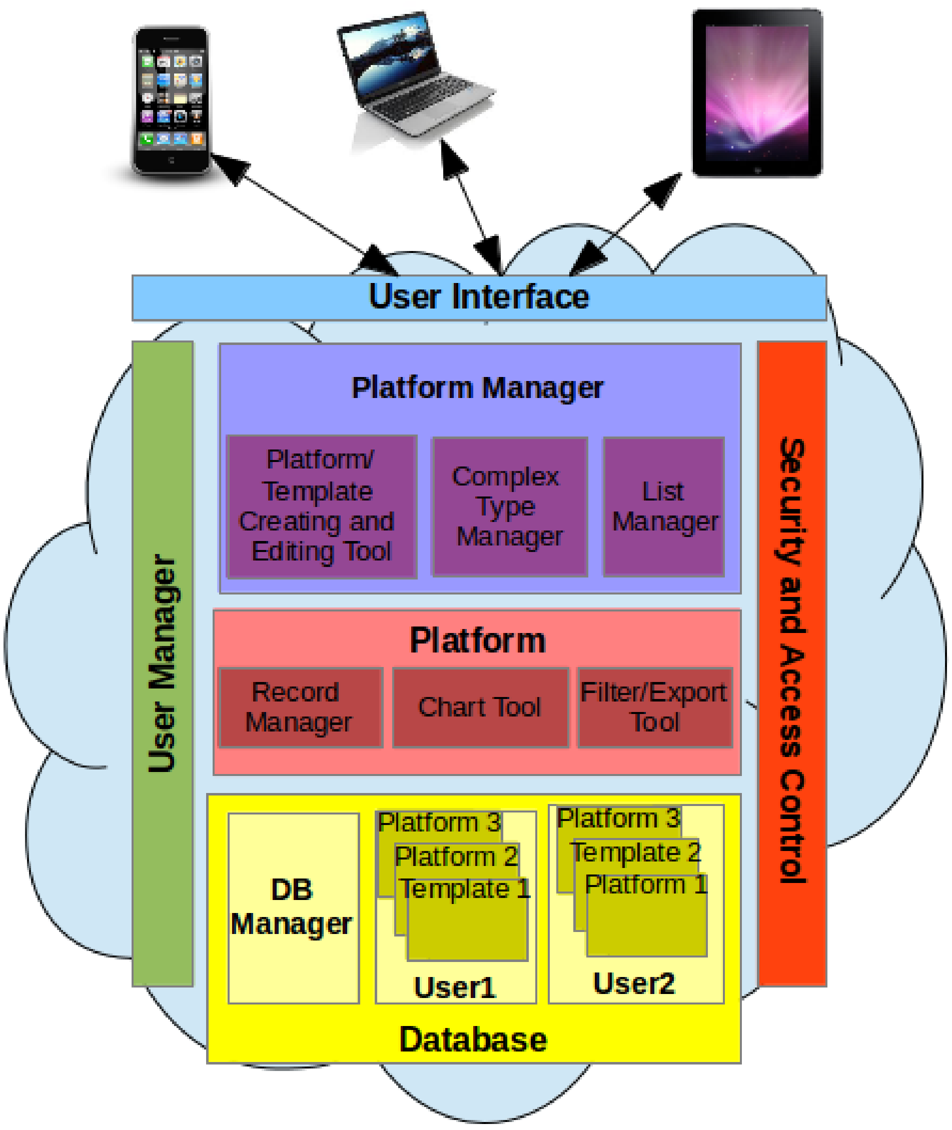


Figure 3.1: A high-level view of CloudStudy architecture.

ager is used to manage platforms and templates utilised by different studies. This module consists of four components. The Questionnaire component, that involves functions for creating and updating the platform questionnaire, the Complex-type Manager that allows PAs to create, update and delete the system's complex-types, and also the List Manager that allows PAs to create, update and delete the system's lists of elements. The Platform Manager module communicates with the DB Manager for retrieving and storing the platform data, as well as with the User Manager to manage the PAs. Finally, Platform Manager communicates with the User Interface and Security and Control modules to ensure the smooth functioning operation of the system. Platform and Platform Manager modules were developed wit PHP, Javascript and jQuery tools.

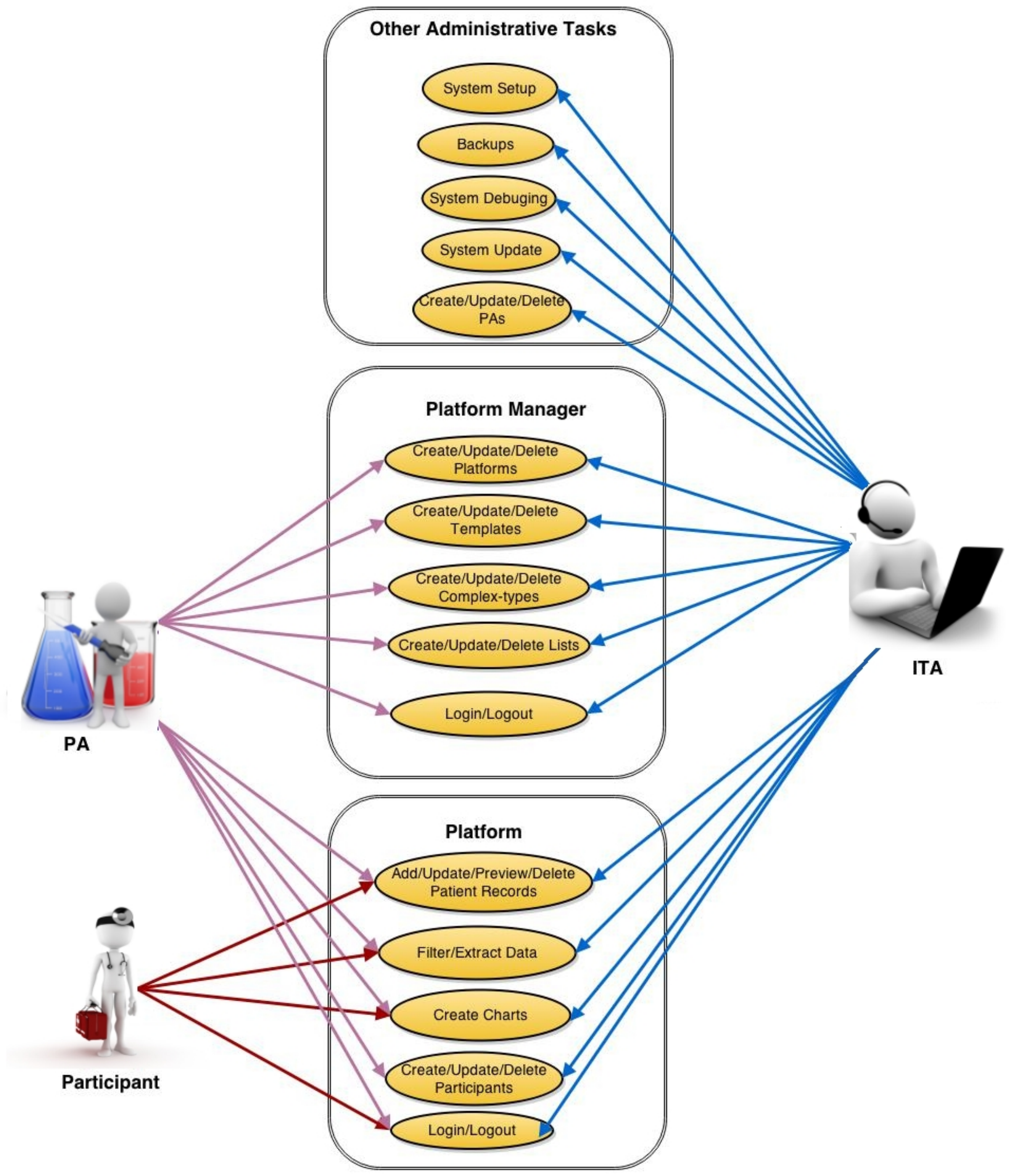
DB Manager, is the module that performs all the necessary storage and retrieval operations to the database (roles, databases, tables) backend. The DB Manager communicates with the Platform, Platform Manager, and User Manager modules for retrieving and storing the data by using PHP and PostgreSQL tools.

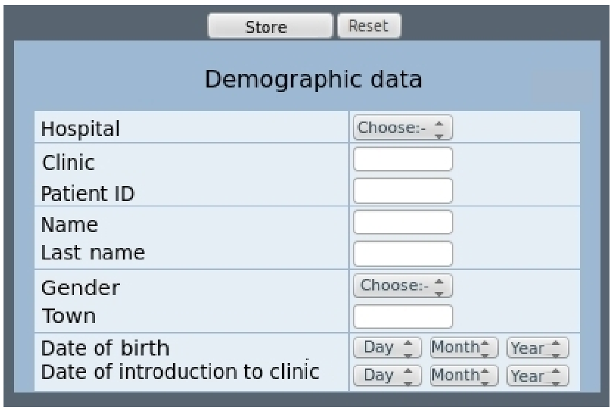
The User Manager module is utilised by the ITA to manage the PAs, as well as by the PAs to manage the Participants (and their roles) in a specific study. The User Manager communicates with all the other modules. The User Manager tool was developed with PHP, Javascript and jQuery tools.

The Security and Access Control module enforces the security policies for the system and controls access privileges over the stored data. Security features include certificate and password-based authentication, single sign-on policy, error handling, data validation, and role-based user management. The Security and Control module communicates with all the other modules to ensure that the accessed at each of the modules is performed in a secure way. The Security and Access Control module was developed using PHP.

3.2 System Overview

CloudStudy has three types of users, and two subsystems Platform and Platform Manager (see Figure 3.2). In this section we describe Platform and Platform_Manager subsystems.


Other Administrative Tasks
System Setup
Backups
System Debuging
System Update
Create/Update/Delete PAs
Platform Manager
Create/Update/Delete Platforms
Create/Update/Delete Templates
Create/Update/Delete Complex-types
Create/Update/Delete Lists
Login/Logout
Platform
Add/Update/Preview/Delete Patient Records
Filter/Extract Data
Create Charts
Create/Update/Delete Participants
Login/Logout
PA
ITA
Participant



Demographic data

Hospital	Choose:-
Clinic	
Patient ID	
Name	
Last name	
Gender	Choose:-
Town	
Date of birth	Day Month Year
Date of introduction to clinic	Day Month Year

Figure 3.3: Example of a platform questionnaire.

3.2.1 The Platform Subsystem

The platform is a custom-made set of questions (questionnaire), together with all necessary user administration and data management components of a study. Figure 3.3 shows the questionnaire of a platform meant for demographic data. Platforms are typically created and administered by PAs and can be used by both PAs and Participants. The only difference between PAs and Participants is that the PAs have access and privileges (preview, update, delete, filter or export) over all platform patient data, while the Participants have access only over their data.

In this subsection we assume that there is a platform for supporting a multicentre study, that has been created with CloudStudy (the platform creating procedure will be described in a later subsection). The Platform module consists of the components Questionnaire, Record Manager (update, preview and delete), Filtering Tool, Chart Tool and User Manager(create,update and delete). In this subsection we describe the functionality of each one of these components. In Figure3.4 we present a flowchart that describes the process of creation, updating, deletion, filtering and exporting files and charts for platform system.

Questionnaire Component

Questionnaires are frequently used in epidemiological studies, as they are one of the most efficient data collection method. They support large size studies and give great statistical power. The filling of the platform questionnaire is a very simple

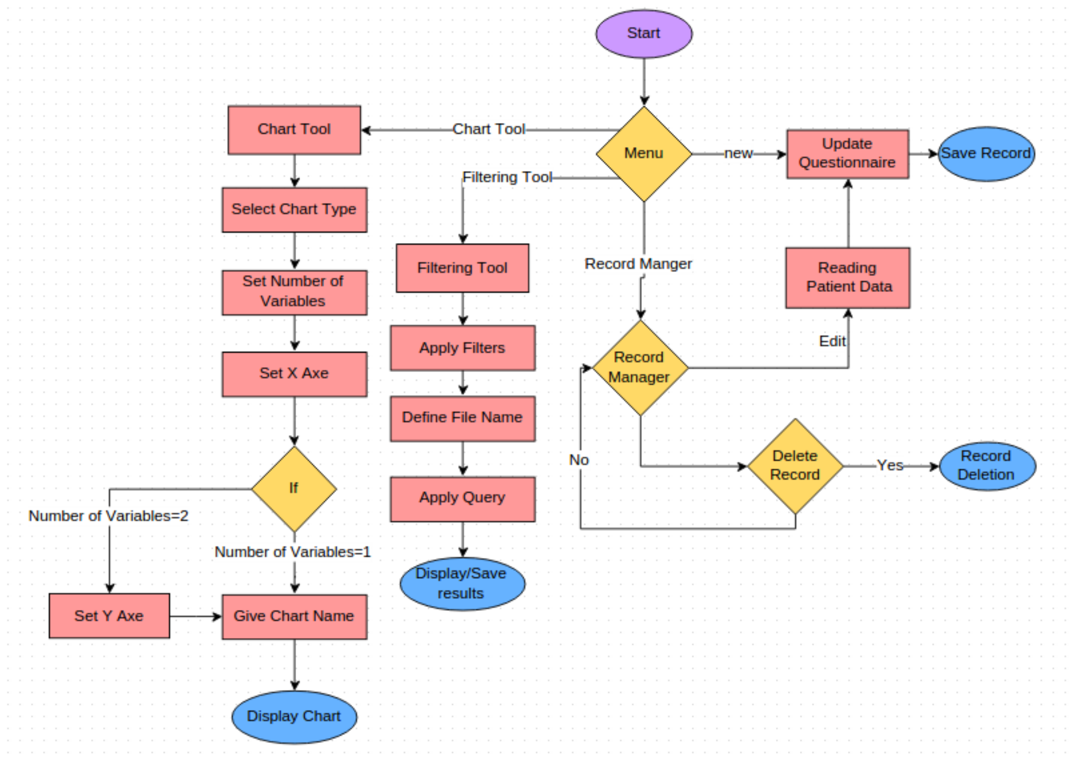


Figure 3.4: Flowchart for the Platform/Template.

procedure. The questions of the questionnaire can be either open-ended, or specific depending the target outcome. Open-ended questions are questions without restrictions on the answers and are generally used to record simple factual information such as name, weight, age, etc. (e.g., number, text, and complex multiple choice questions). While restricted questions have restrictions in the form of a limited number of possible answers (e.g., multiple choice, date, and list questions). Also the questionnaire supports recurring questions. A recurring question is a question that can be repeated more than one times. For example, a patient may receive more than one therapy cycles in one clinical episode. The clinician can dynamically choose how many therapy cycles the patient received, and create the necessary number of therapy questions. Recurring questions help us achieve better data modeling, and subsequently allow us to support richer database queries. A user may save the data at any time while filling in the questionnaire and continue its task in a later time. Also the questionnaire supports branching logic, that means that the questionnaire changes according to the responses to specific questions. In a questionnaire with branches, questions appear only if they apply to someone's situation. If the ques-

Open-ended Questions	
Complex multiple-choice	The user may select the question value either from a drop-down list with the suggested value or type an alternative answer.
Text	The user has to type a string
Number	The user has to set any number value.
Date	The user has to choose a date from a date field.
Restricted Questions	
Multiple-choice	The user has a drop-down list with the possible question values and he has to choose one.
List	The user has to choose a value from the list.

Table 3.1: Types of questions that the questionnaire supports.

tions don't apply, the person can answer a different set of questions or skip that set of questions.

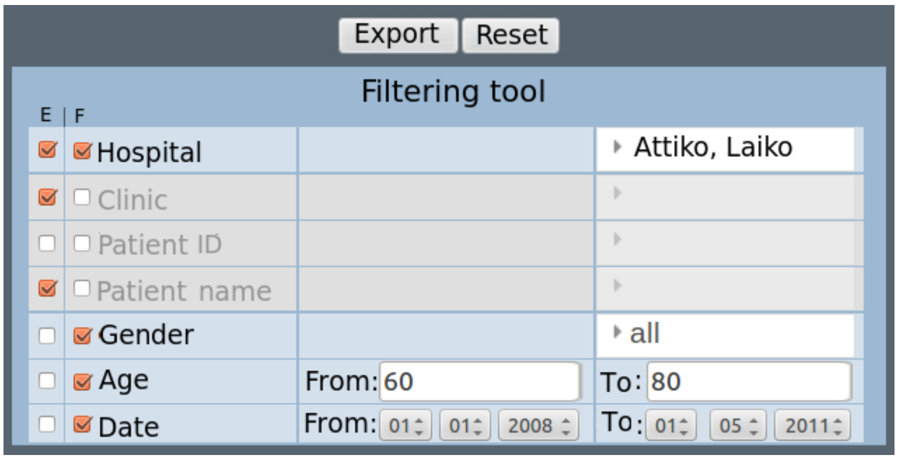
The questions can take:

Record Manager Component

The Record Manager is the component of the Platform subsystem that allows users to manage their patient records. Through the Record Manager users may Add, Update and Delete patient records. For adding a patient record the user has to fill the platform questionnaire. Also, through the Record Manager the users may preview, update and delete any stored patient record.

Filtering and Chart Components

The Filtering and the Chart tools are also components of the platform. With these tools PAs and Participants are able to extract information from the database. The difference between PAs and Participants is that the first ones may filter and extract information from the whole study, while the second ones may extract information only on their own data due to access restrictions. This happens because PA is the study coordinator and he has access over the whole study data, while Participants have access only on their data for anonymity reasons. Such information may be retrieved and presented as *(i)* a set of tuples by resorting to the Filtering Tool, and



Filtering tool

E	F			
✓	✓	Hospital		▸ Attiko, Laiko
✓	☐	Clinic		▸
☐	☐	Patient ID		▸
✓	☐	Patient name		▸
☐	✓	Gender		▸ all
☐	✓	Age	From: 60	To: 80
☐	✓	Date	From: 01 01 2008	To: 01 05 2011

Figure 3.5: The filtering tool of a platform.

(ii) a graph by resorting to the Chart Tool.

The Filtering Tool (Figure 3.5) uses a powerful, yet easy-to-use, query issuing mechanism that allows users to *(i)* filter and retrieve and *(ii)* export to a third-party application stored records by applying constraints with simple point-and-click interactions. Data filter and export involves a *two-step process*. In the first step the user defines the query output by checking the questions that will be used for *data projection* (i.e., data to be exported). In the second step, the user may apply one or more *filtering conditions* on the data. There exist two types of filtering conditions (see Figure 3.5): the questions that take numeric values (e.g., date, integer, decimal) have start-end restrictions, i.e., the user may set the *start* value, or the *end* value, or both of them. While the questions that accept string values (e.g., multiple-choice, text, lists) are introduced by presenting *all distinct values* stored for the specific question and allowing the user to define the ones that satisfy the filtering criteria. In this way, the user has the possibility to query database with complex queries that involve *conjunctions* and *disjunctions* while the user has no prior IT knowledge. The *conjunctions* are allowed between the recurring questions, while the *disjunctions* can be applied on questions that take string values.

The query result may then be exported in a spreadsheet, or presented as a list of tuples. In Figure 3.5 the PA has selected to export the results for Hospitals, Clinics, and Patient names (notice the checked boxes in the first column) for all records that were input between 01/01/2008 and 01/05/2011 in Attiko or Laiko hospital and involve patients between 40 and 60 years of age (notice the specified constraints).

The Filtering Tool is able to capture the complex data types described earlier, allowing the user to *(i)* set more than one filters for every complex data type and *(ii)* include constructs like concurrent episodes of a diagnosis or treatment. Examples of supported queries include:

- Show patients older than 40 years of age with temperature greater than 38, who were diagnosed with microorganisms (Pseudomonas and Candida). Notice that the names of the microorganisms are values already stored in the database.
- Show patients between 40 and 60 years of age, who live in (England or France or Greece), have been diagnosed with microorganisms ((Pseudomonas and Candida) or (Pseudomonas and Salmonella) or (Citrobacter)), have undertaken therapy with (Ampicillin or Cefotaxime), and had progress as the eventual outcome of their condition. Notice the application of Boolean operators both on the questions, and on the data stored for each question.
- Show patients who suffer from Massive hemoptysis, had an initial episode of Bacteremia and accepted a dose between 2 and 4mg of antibiotic Amikacin, and subsequently had a second episode of Bacteremia. Notice that the query is applied to a complex data type (recurring episodes/treatments as described above).

The Chart Tool (Figure 3.6) may be used to extract and present information from stored data directly as a *pie, column, bar, or donut chart.* The Chart Tool supports the creation of single and multiple variable graphs depending on the constraints defined by the user. The users may extract a chart by specifying the following options.

1. The *chart type* that can take values (2D Pie, 3D Pie, Bar, Column, Donut charts),
2. The *number* of the chart variables that can be one or two.
3. The values of X and Y *axes* (Y is used only if the variables are two). The X and Y parameters are lists that include all the questions of the questionnaire.

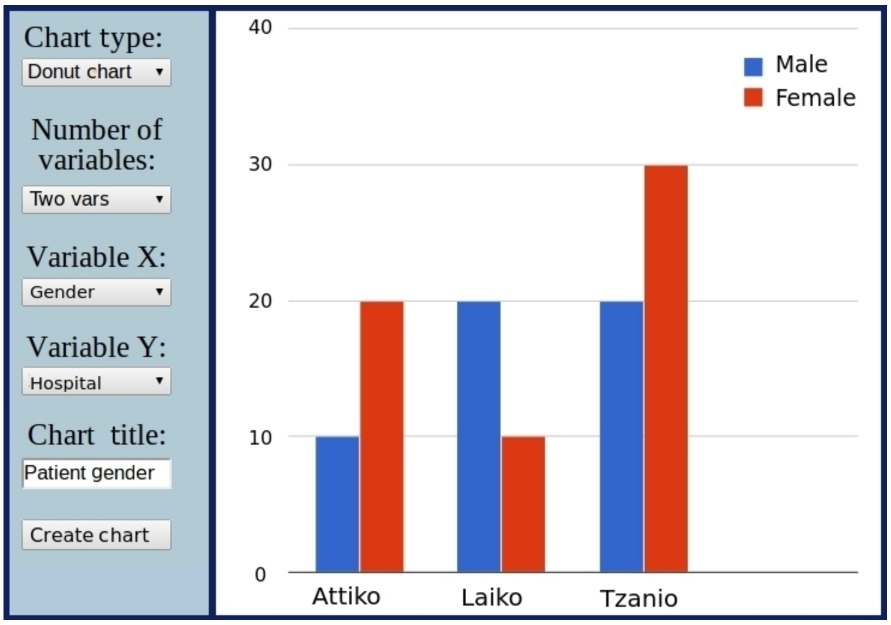


Chart type:
Donut chart
Number of variables:
Two vars
Variable X:
Gender
Variable Y:
Hospital
Chart title:
Patient gender

Figure 3.6: The Chart Tool of a platform.

4. The chart *name*.

In this way the user may dynamically create graphs for queries with a single variable like “how many patients in the database are male/female", but also for queries with multiple variables like “how many patients are male/female in each hospital that participates in the study" (shown in Figure 3.6).

3.2.2 Platform Manager Subsystem

The next subsystem of CloudStudy is the Platform Manager. The Platform Manager consists of the Platform/Template Create tool, the Platform/Template Update tool, the Platform Delete tool, the List Manager, the Complex-type Manager and the User Manager. In this subsection we describe each one of these components, and present the offered what the user can do with these tools. In Figure3.7 we present a flowchart diagram that describes the creation, updating and deletion of the platform Manager subsystem a platform/template.

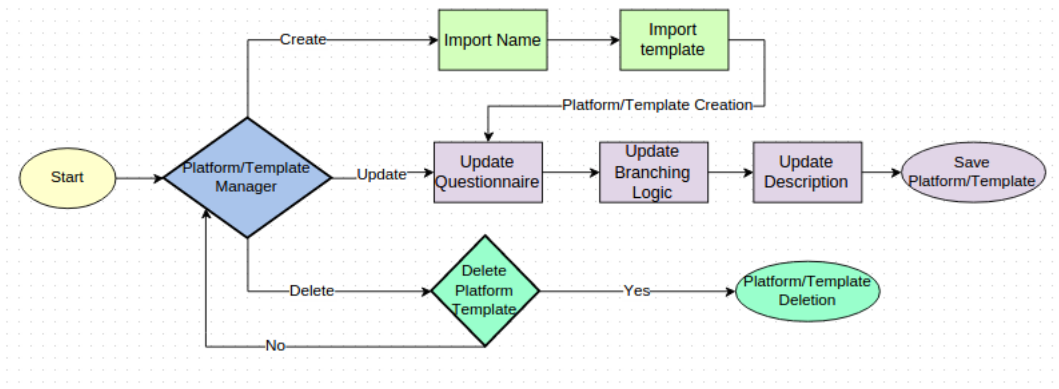


Figure 3.7: Flowchart for the platform/template creation, updating and deletion.

Platform Creating Component

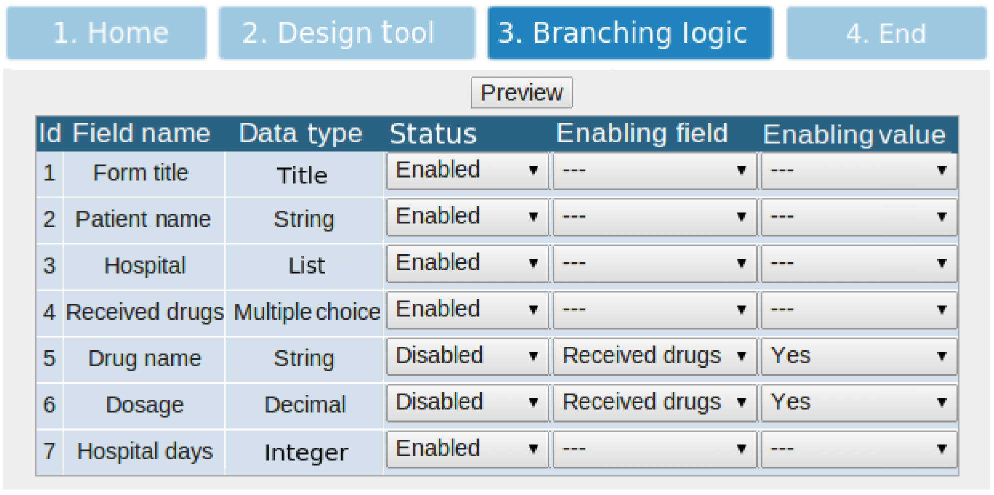
Technically, a platform is a custom-made set of questions, together with all necessary user administration and data management components of a study. Platforms are typically created and administered by PAs. Very often, specific parts of a study may be used (as they are or with minor modifications) in other studies. For example, demographic data are a typical reusable part of many biomedical multi-centre studies since they often remain unchanged among different studies. To support this reusability, CloudStudyoffers the PA the possibility to create templates that may be used across different studies. Figure 3.3 shows a template meant for demographic data. Platforms and templates are handled through the same tool that is the Platform Manager.

The procedure of creating a platform and a template is exactly the same. To create a new platform/template the system has to know the platform's name, the questionnaire structure, the questionnaire branching logic, provide a brief description of the study, and define whether it will be used as a platform or a template (see Figure 3.7).

Designing the platform/template questionnaire is the most important part of the platform/template creation, as it affects the quality of results of the final study results. The creation tool is flexible and easy-to-use, while at the same time offers to PAs the possibility to design complex questionnaires that satisfy each specific study. The *questionnaire structure* has to be defined during the platform creation. As we described earlier, the questionnaire consists of a number of questions. For each question the PA has to specify three key elements: the question *data type*, the *question text*, and the *question values*. Question values are enabled depending on each question's data type. When the question is a multiple choice or complex multiple choice type the question value takes the set of multiple choice values. Similarly when the data type is a list then the *question value* field becomes a drop-down list with all the available lists. The questionnaire design includes functions for Adding, Deleting, or Rearranging the questions. The function Add allows PA to add as many questions as he needs, while the function Delete allows PAs to delete any question. The function Rearrange allows the PA to rearrange the questions of the questionnaire through drag-and-drop actions. CloudStudy supports all standard data types such as *titles*, *strings*, *integers*, *dates*, *decimal*, together with more advanced types such as *multiple choices*, *Complex Multiple choices*, *lists* and *complex data types*. These data types are explained below in detail.

- *Titles* are strings that are used to introduce new questionnaire sections.
- *Multiple choices* are drop-down lists with a number of possible answers, that force the user to select one of the available values.
- *Complex Multiple choices* are multiple-choice fields where the user is not necessary to choose between the available values, but he may input a new value.
- *Lists* are drop-down lists that the PA dynamically creates, stores, and updates during the platform design or maintenance. Lists are described in detail in Section 3.2.2.
- *Complex datatypes* support complex medical operations (see Section 3.2.2).

The *Branching Logic* definition is the part of the design where the PA specifies the values in questions that are required to enable or disable following ones. For example in Figure 3.8, the PA specified that if the answer to the question Received drugs is yes, the questions Drug name and Dosage should be enabled. In this example the question Received drugs is the *master* question while the questions *Drug name* and *Dosage* are the follow-up questions that are by default disabled. The *master* question has to be before the follow-up question and there is always one value of the *master* question that enables the disabled follow-up question. Finally the fourth



Id	Field name	Data type	Status	Enabling field	Enabling value
1	Form title	Title	Enabled	---	---
2	Patient name	String	Enabled	---	---
3	Hospital	List	Enabled	---	---
4	Received drugs	Multiple choice	Enabled	---	---
5	Drug name	String	Disabled	Received drugs	Yes
6	Dosage	Decimal	Disabled	Received drugs	Yes
7	Hospital days	Integer	Enabled	---	---

Figure 3.8: The Branching Logic definition of a questionnaire.

part of the creation tool is to give the title and the description of the study. The branching logic is a three-step operation: in the first step the user defines if the question is a follow-up or not, if the question is follow-up the user has to set it to disabled, the user selects from a drop down menu the question that can enable it, and finally the third step is to select the value that enables the follow-up question.

Platform/Template Updating Component

The Platform Manager subsystem supports the Platform/Template updating even if the platform is in use. Updating an existing platform/template in CloudStudy involves two different scenarios: The (i) *simple mode* allows the PA to make changes on platform that cannot affect the stored data of the platform. For example, *simple mode* allows users to make changes that involve rephrasing and addition of questions, or changing the branching logic of specific questions. The (ii) *structural mode* is a module that allows PAs to make any change on the platform, such as question reorganization, removal or modification of question datatypes. The later scenario may affect data consistency or may cause compatibility problems between the stored patient records and the new questionnaire. In the following we present an example of a case that would cause compatibility problems between the stored patient records and the new questionnaire. We suppose that a PA updates the questionnaire using the *structural mode* and changes the datatype of question Patient name from string

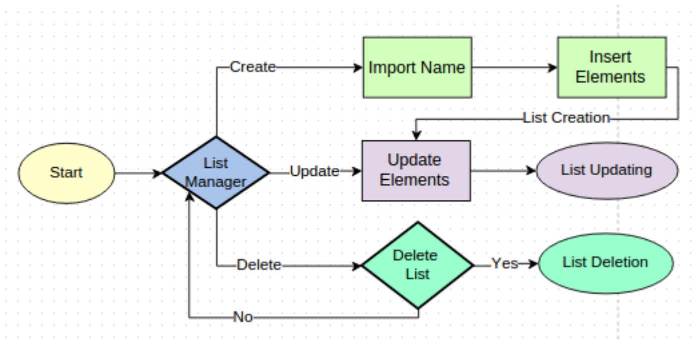


Figure 3.9: Flowchart for the List Manager.

to integer, then saves the changes to the existing platform/template. Subsequently, the platform user would open the platform manager, where the system would try to retrieve the stored patient records. When the system would arrive to question Patient name it would try to read an integer value but the database would return a string value. So, to avoid problems like these the system formats the database and clears all the stored data when the *structural mode* ends.

Platform/Template Deleting Component

CloudStudy provides PAs with the deletion tool with which a PA may delete a platform. This operation removes the platform including the Participants and the associated data.

List Manager Component

In many multi-centre studies researchers use lists of elements such as lists of hospitals, microorganisms, antibiotics, etc. Platform Manager provides PAs with the possibility to create lists of elements for making platform creation and usage more usable and flexible. A list has a unique name and contains a set of elements. These lists may be shared across different studies of the same PA and are used to ensure data consistency, enhance data integrity, and enforce validation of input. The List Manager module provides users with the appropriate tools to create, update and delete lists during the questionnaire design. The create list function allows PAs to create a new list and define the list elements, while the Update function allows PAs to edit/reorder or add list elements. Finally the function Delete allows PAs to delete

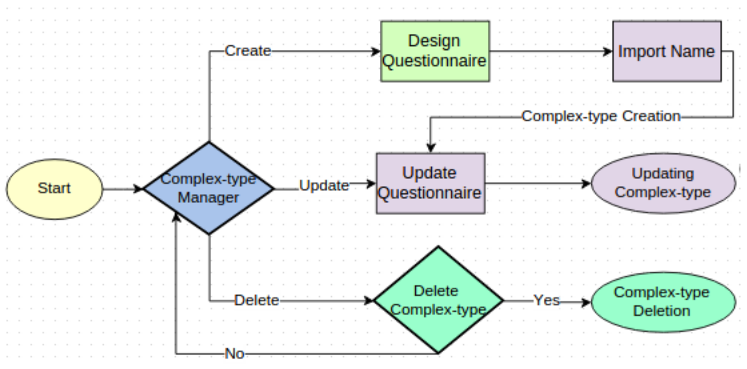


Figure 3.10: Flowchart for the complex-type Manager.

any lists that are in use by a platform/template/complex-type. To use a list during the questionnaire design the PA sets the question's data type to list and selects one of the stored drop-down lists. In Figure3.9 we present a flowchart diagram that describes the creation, updating and deletion of a list functions through the Platform Manager subsystem.

Complex-type Manager

CloudStudy supports complex medical operations such as treatment plans, therapeutic protocols, antibiograms, etc. These complex operations are groups of recurring questions that usually consist of a number of questions (e.g., medicine name, start/end dates of therapy, outcome of therapy, etc). The benefits of creating complex-types include better data modeling, and thus richer query possibilities, and also offers flexibility in the design of studies that need to record complex/recurring medical processes. For example, a questionnaire may include a question about therapies where the user has to choose how many therapy cycles have been received by the patient. A therapy cycle might consists of three questions namely antibiotic name, start and end date, and therapy result. In this subsection we describe the Complex-type Manager and discuss its usefulness to the PAs through the addition/ updating/ deletion of complex-types.

The procedure of Inserting/Creating a new complex-type is very simple and the platform Creation process. The complex-type creation consists of two parts. The first part includes the definition of the complex-type name, while the second part is the questionnaire structure design of the complex-type. For each question, the PA specifies three key elements: the *question type*, the *question text*, and the *question values*. In this step the user may add, rearrange, or delete the questions through drag-and-drop actions. CloudStudy supports all standard data types such as *Strings*, *Integers*, *Dates*, *Decimal*, *Multiple Choices*, *Complex Multiple choices* and *Lists*.

Except from creating complex-types, PAs can also update them. Possible updates include rephrasing of existing questions or addition of new ones. If the complex-type is not in use by a platform, the users may also delete or change the datatypes of the questions. Finally the PAs may delete the complex-types given that they are not any platform or template. In Figure3.10 we present a flowchart diagram that describes the steps that a PA has to follow for the complex-type creation, updating and deletion through the Platform Manager subsystem.

3.2.3 User Manager

Another component of CloudStudy is the User Manager, that is accessible both through Platform and Platform Manager subsystems. This component allows ITAs to manage the PAs, and PAs to manage the Participants of a study. The ITAs have access to the User Manager through the Platform Manager, while PAs may access the User Manager through the Platform. The User Manager provides users with the ability to create new users, or update/delete existing ones. When an ITA deletes a PA all of his platforms and data are deleted, and when a PA deletes a platform data ownership rights are automatically transfered to him.

3.3 Database

In this section we describe the database schema and we give information about each database of the system.

3.3.1 Database Schema

In Figure 3.11 we see the layers of the system's databases. The system has three layers of databases. The First Layer contains one database called `logindb` used by the platform manager to authenticate the identity of the PA during the login procedure

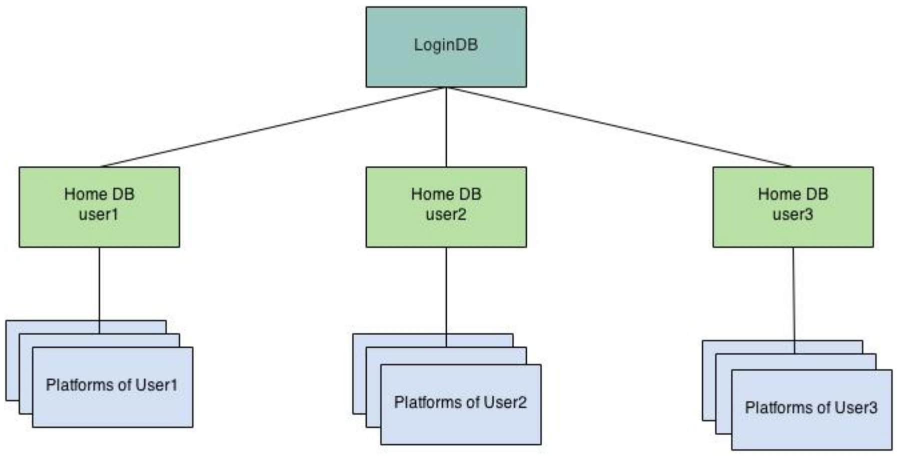


Figure 3.11: The Layers of CloudStudy Databases.

and contains two tables: `cloudusers` and `db_info`. In the second layer, for each user of table `cloudusers`, the system creates a `home` database named with the user's name. This layer has as many databases as the number of the users. A `home` database contains one table for each *List of elements* of the user. The reason that these tables are in the home database of a user is to be visible from all the user's platforms/templates. Finally, the third Layer contains one database for each platform/template of the system. So as user account contains a database for each platform/template owned by that user. Each platform/template database consists of at least four tables (`records, questions, users, ct_questions, complextype_x`). Table `records` contains all the data of a patient record (e.g., each column corresponds to a question of the questionnaire). Table `questions` contains information about all the columns (e.g., each row of table questions corresponds to a column of table records) of table `records`, table `users` includes information for the platform users. Table `complextype_x`, where x can be any name, represents a complex-type of the platform. These tables are as many as the number of the complex-types of the specific platform. Table `ct_questions` contains information about all the questions of all `complextype_x` tables (e.g., each row of table `ct_questions` corresponds to a column of a complex-type table).

In Figure 3.11 we see the layers of the system's databases. The system has three layers of databases. The First Layer contains one database called `logindb` used by the platform manager to authenticate the identity of the PA during the login procedure. Database `logindb` contains two tables: table `cloudusers` that lists all the Platform Manager users, and table `db_info` that includes details about each platform database (`owner, name, platform info`, etc). In the second layer, for each user of table `cloudusers` (described above), the system creates a `home` database named with the user's name. This layer has as many databases as the number of the users. A `home` database contains one table for each list of elements of the user. The reason that these tables are in the home database of a user is to be visible from all the user's platforms/templates. Finally, the third Layer contains one database for each platform/template of the system. So as user account contains a database for each platform/template owned by that user. Each platform/template database consists of at least four tables (`records, questions, users, ct_questions, complextype_x`). Table `records` contains all the data of a patient record (e.g., each column corresponds to a question of the questionnaire). Table `questions` contains information about all the columns (e.g., each row of table `questions` corresponds to a column) of table `records`, table `users` includes information for the platform users. Table `complextype_x`, where x can be any name, represents a complex-type of the platform. These tables are as many as the number of the complex-types of the specific platform. Table `ct_questions` contains information about all the questions of all `complextype_x` tables (e.g., each row of table `ct_questions` corresponds to a column of a complex-type table).

3.3.2 Databases and Tables

First Layer

The first layer consists of the `logindb` database, which is the default database that the platform manager connects to identify if the given login name and password are valid. `Logindb` has two tables the `cloudusers` table (see Table 3.2) that contains information `id, username, password, fullname, email, hospital, database name, role name)` for each user of the platform manager, and `db_info` table (see Table 3.3) that contains information (database name, owner of the database, title of the platform, description of the platform) about each database. A user may have more than one databases but a database belongs only to one user.

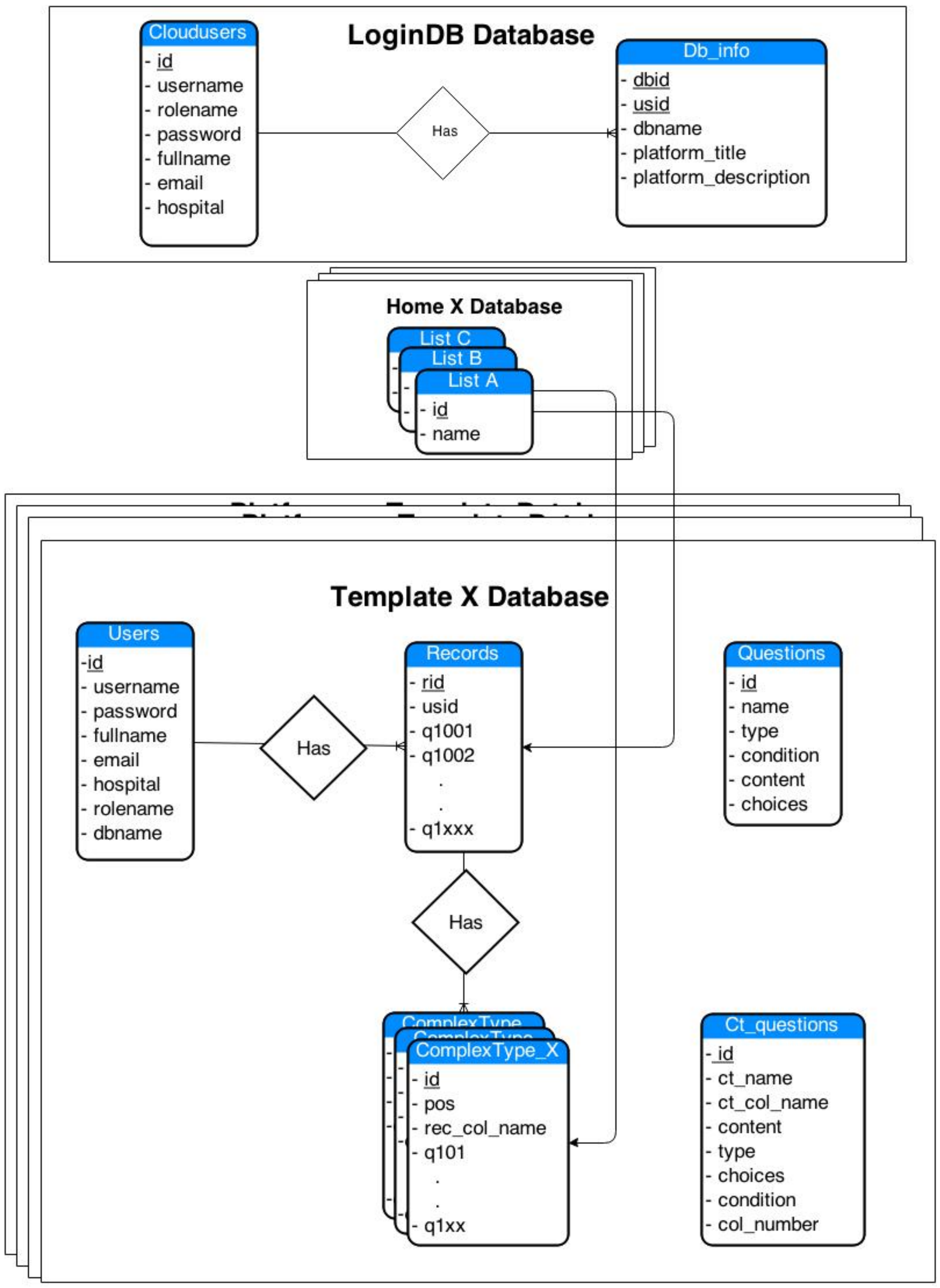


Figure 3.12: CloudStudy Database Schema.

Data Item	Type	Description
id	Integer	Platform administrator id.
username	Varchar	The PA username.
password	Varchar	The PA password.
fullname	Varchar	The PA fullname.
date	Date	The PA registeration date to the system.
email	Varchar	The PA email.
hospital	Varchar	The PA working hospital.
dbname	Varchar	The database name that the PA belongs.
rolename	Varchar	The database role name.

Table 3.2: Database table cloudusers.

Data Item	Type	Description
name	Varchar	The database name.
ownerid	Integer	The PA id.
title	Varchar	The platform/template title.
description	Varchar	The platform/template description.

Table 3.3: Database table db_info.

Second Layer

The second layer consists of the set of `Home` Databases. When a Platform Administrator is created by the system, also is created a `role` and a `database` with the PA's name. This `database` is the `Home` Database. For example, when the system creates the PA Bio, subsequently creates a role and a database named `Bio`. In `Bio` database are storing the list tables. A list table has a name that begins with prefix *dbtable_*, for example a list of hospitals has the name `dbtable_hospitals`(see Figure 3.4).

Data Item	Type	Description
id	Integer	The element id.
name	Varchar	The element name.

Table 3.4: Database table dbtable_hospitals.

Data Item	Type	Description
qid	Integer	The element id.
condition	Varchar	A string for defining question branching logic. If it is null the question is enabled.
type	Varchar	The question type.
content	Varchar	The question text.
choices	Varchar	This field takes a set of values if the data type of the question is multiple choice/complex multiple choice, a number of position if the data type is complex-type, a name of a list table if data type is list.

Table 3.5: Database table questions.

Third Layer

The third layer of the databases is the set of the platforms and templates databases. As we described earlier, for each platform or template the system creates a database. All platform databases have names that begin with the prefix *platform_* and all the template databases have names that begin with the prefix *template_*. For example: The PA Bio owns a platform called `Bacter`, subsequently he owns the database `platform_Bacter`. The database `platform_Bacter` consists of the following tables (`questions, records, ct_questions, complextype_Therapy, users(see Table 3.9)`). The table `questions` that contains information about the questions of the platform questionnaire. The table `records` (see Table3.6) that stores the patient data. Each row in this table corresponds to a patient. Table `ct_questions` (see Table 3.5) contains information for the set of all the complex types questions. As we mentioned before, for each complex-type the database keeps a complex-type table. When a new complex type *Therapy* is created, the system performs two actions. The first is to add to `ct_questions` (see Table3.7) the information for the Therapy questions, and the second is to create a new table `complextype_Therapy` (see Table3.8) in the `platform_Bacter` database. Finally the table `users` includes information about all the platform users that are necessary for the platform login authentication. The tables are described as following:

Data Item	Type	Description
usid	Integer	The user id.
q1001	Any Type	The first question of questionnaire.
q1002	Any Type	The second question of questionnaire.
.	.	.
q1xxx	Any Type	The xxx question of the questionnaire.

Table 3.6: Database table records.

Data Item	Type	Description
id	Integer	The question id.
ctname	Varchar	The complex-type name.
column_name	Varchar	The column name.
content	Varchar	The question text.
type	Varchar	The question data type.
choices	Varchar	A set of values if the datatype is multiple choice or complex multiple choice or a name of the list table if the datatype is list.
condition	Varchar	The condition that enables the question when it is disabled, if the value is null the question is enabled.
column_number	Integer	The position of the question in the complex type.

Table 3.7: Database table ct_questions.

Data Item	Type	Description
id	Integer	The id of user.
pos	Integer	The complex data types have the affiliation of being recurring, for example a patient may receives three recurring cycles of therapies, these three therapies create a group of questions, the position shows the position of the group and is unique for the group in the questionnaire.
column	Varchar	The column name of the table questions that refer to this complex type (e.g., column q1001).
recordid	Integer	The id of the patient record.
q101	Any Type	The first question of complex type therapy.
q102	Any Type	The second question of complex type therapy.
.	.	.
q1xx	Any Type	The xx question of complex type therapy.

Table 3.8: Database table complexype_x.

Data Item	Type	Description
usid	Integer	The platform user id.
logname	Varchar	The name with which the platform user logs in the system.
password	Varchar	The password with which the platform user logs in the system.
type	Varchar	The type of the platform user that correspond to the access permission of the user (superuser or simple user).
fullname	Date	The fullname of the platform user.
date	Varchar	The date that the platform user registered to the system.
email	Varchar	The email of the platform user.
hospital	Varchar	The hospital that the platform user works.

Table 3.9: Database table Users.

Chapter 4

Set-up and Usage of CloudStudy

In this chapter we provide more details on the installation and requirements of CloudStudy, and discuss the most important functions of the system. We also present two actual multi-centre studies at using CloudStudy, and finally provide a comprehensive step-by-step user guide for the system.

4.1 Installation and Requirements

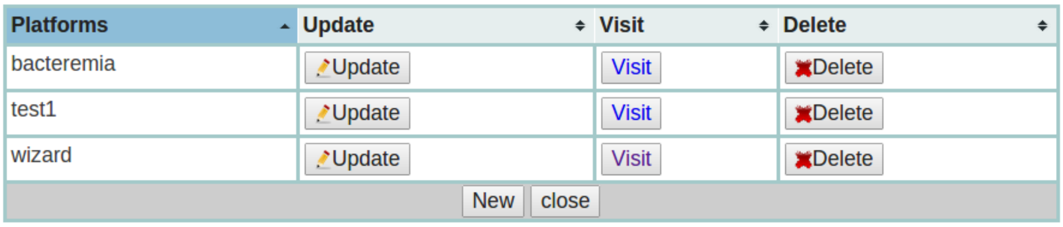
CloudStudy is a web application implemented by resting on technologies such as HTML, PHP, Javascript, jQuery while the database backend utilises PostgreSQL. In this section we give all necessary information on how to setup CloudStudy.

Firstly, the IT Administrator has to setup the system in a server. The server has to include the PHP, PostgreSQL and Apache software packages. An alternative solution for non-IT users is to install to their computer the Bitnamy LAPP-stack (`https://bitnami.com/stack/lapp`) that completely automates the process of installing and configuring all the above software, with just a few clicks. The user has to create a *public_html* folder and set permissions to allow read/write/execute access for everybody. The *public_html* folder is the web root for the primary domain name. This means that *public_html* is the folder where the user has to put the CloudStudy source files. After this procedure, the user has to to run the `init.sh` script, that is presented in the following, to initialize the CloudStudy database.

After the setup, any user may access CloudStudy through an internet browser. This makes access possible via any device that has internet connection and a standard browser. The compatible internet browsers are the: IE 8+, Chrome 30+, Firefox 30+, Safari 3+ and Opera 11+. CloudStudy will likely run problem-free in older browser versions, but it has not been systematically tested on older browser versions.

init.sh script

```
*init.sh script is responsible to initialize CloudStudy database.

*Creates the role logindb and the database logindb.
Create role logindb createrole createuser login password 'dbpass';

Create database logindb;

*connects to database logindb.
\connect logindb;

*creates the table cloudusers that consists all the Platform
*Administrators.

Create table cloudusers(
id serial,
username varchar(20) not null unique,
password varchar(15) not null,
fullname varchar(40),
date date,
email varchar(20),
hosppostgresl varchar(30),
dbname varchar(25) not null,rolename varchar(20) not null);

*insert the first user into table cloudusers.
Insert into cloudusers (username,password,dbname,rolename)
values ('username','homedb','databasename','rolename');
```



Platforms	Update	Visit	Delete
bacteremia	Update	Visit	Delete
test1	Update	Visit	Delete
wizard	Update	Visit	Delete

New close

Figure 4.1: TableSorter plugin example.

4.2 Libraries and Plugins

In this section we describe the libraries and plugins used to develop the CLOUD-STUDY system.

4.2.1 jQuery Library

jQuery[1] is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. jQuery(1.8.3) is one if the libraries that was used for CLOUDSTUDY development and is compatible with all browsers. In the following we describe the most important jQuery plugins that were used for CLOUDSTUDY development.

jQuery Tablesorter Plugin

Tablesorter(2.0.5)[2] is a jQuery plugin that turns a standard HTML table into a sortable table (see Figure 4.1) without page refreshes. Tablesorter can successfully parse and sort many types of data, including linked data in a cell. Some of the most useful features that provides are Multi-column sorting, as well as some Parsers for sorting text. TableSorter is compatible with browsers: IE 6.0+, FF 2+, Safari 2.0+ and Opera 9.0+.

[1] `http://jquery.com/`
[2] `http://tablesorter.com/docs/`

jQuery-UI

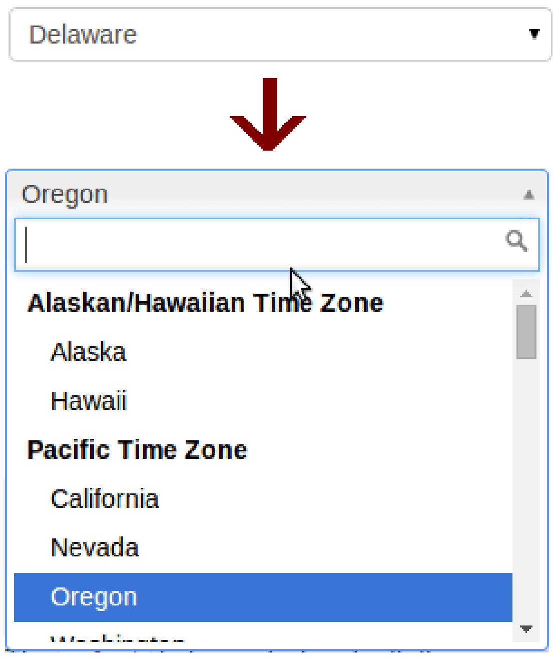
jQuery-UI(1.9.1)[3] is a popular jQuery library that is based on the jQuery library. The jQuery-UI provides a set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library, and it is used mostly for building web applications. In the following we describe the most important UI plugins that were used on CloudStudy development.

- The *Draggable* plugin allows users to drag-and-drop elements using a mouse, and it was used for the implementation of the question rearrangement in the questionnaire design.
- The *Sortable plugin* enables a group of DOM elements to be sortable. With the usage of this plugin the user may click on and drag an element to a new spot within the list, and the other items will adjust to fit. By default, sortable items share draggable properties.
- *Dialog* is another useful plugin which is an overlay positioned within the view-point, and is protected from page content (like select elements) shining through with an iframe. It has a title bar and a content area, and may be moved, resized and closed with the 'x' icon by default.
- The *Hide/Show* plugins were used by the programmer to allow users hide and show elements without page refreshes.

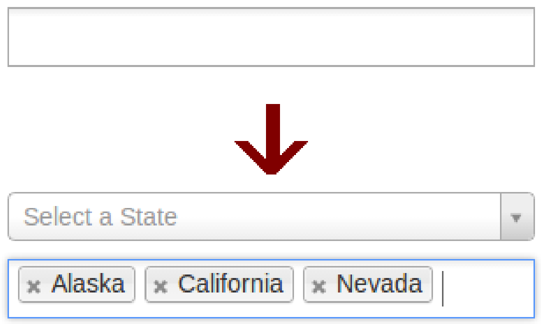
jQuery Plugin Select2

A useful plugin that was used is the Select2(3.3.0)[4] plugin, which provides the *Select2* boxes(see Figure 4.2) that replace the standard html select boxes. *Select2* boxes support searching, remote data sets, as well as infinite scrolling of results. CloudStudy supports both types of Select2 boxes: the one-value select2 boxes (see Figure 4.2(a)) that supports quick option filtering via a search box, and the multi-values (see Figure 4.2(b)) select2 boxes, that except from the filtering via a

[3] `http://api.jqueryui.com/`
[4] `http://ivaynberg.github.io/select2/`

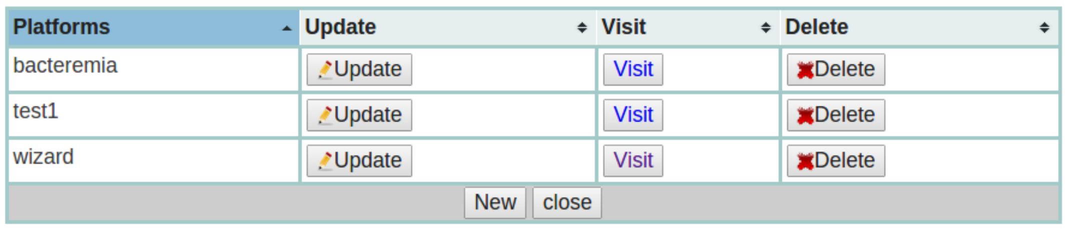


(a) One-value Select2 boxes.



(b) Multi-value Select2 boxes.

Figure 4.2: Select2 Boxes.



Platforms	Update	Visit	Delete
bacteremia	Update	Visit	Delete
test1	Update	Visit	Delete
wizard	Update	Visit	Delete

New close

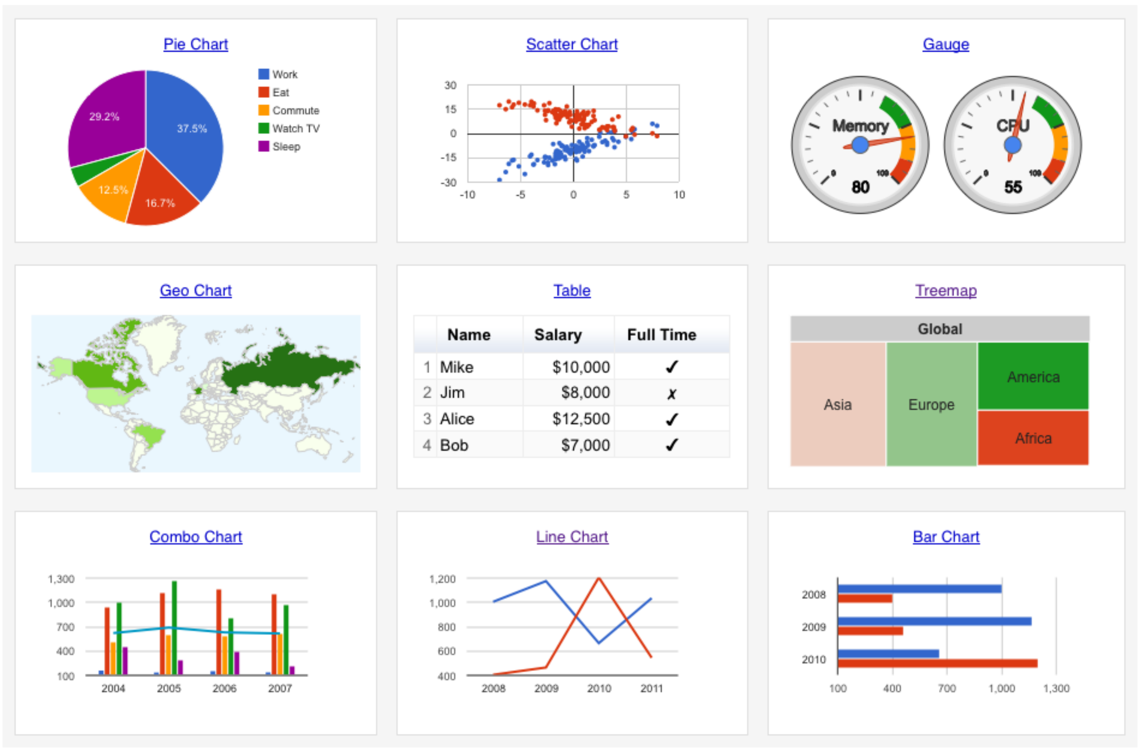
Figure 4.3: TableSorter plugin example.

search box, also support multi values. The Select2 plugin is compatible with IE 8+, Chrome 8+, Firefox 10+, Safari 3+ and Opera 10.6+.

jQuery Steps

jQuery Steps[5] is another plugin that allows the programmer to easily create wizard-like interfaces (see Figure 4.3). This plugin groups content into sections for a more structured and orderly page view.

[5]`http://www.jquery-steps.com/GettingStarted`



	Name	Salary	Full Time
1	Mike	$10,000	✓
2	Jim	$8,000	✗
3	Alice	$12,500	✓
4	Bob	$7,000	✓

Figure 4.4: Google Charts example.

4.2.2 Google Charts

Google Charts[6] is a JavaScript library that provides a perfect way to visualize data on websites. From simple line charts to complex hierarchical tree maps, the chart gallery (see Figure 4.4) provides a large number of ready-to-use chart types that can be easily embed to a site/application.

4.3 Implementation

In this section we describe the implementation of the most important functions of Plarform and Platform Manager.

4.3.1 Platform

In this subsection we describe some of the most important functions of the platform.

[6]`https://developers.google.com/chart/interactive/docs/index`

Login Function

The `login(username, password, platform)` function is called during the user login to the platform. The function takes three parameters the `username`, the `password` and the `platform name`, that the user typed in the login form. The `login` function creates a database connection with the given `platform` database, and checks if the combination of the `username` and `password` exists in the database table `users`. If the combination exists the system allows the user to enter the platform.

Database Connections

During the platform usage, the system has to be connected with the databases of the system, so that to retrieve and store the data. When the user logs in the platform the system opens two connections, the first connection with the user's `home` database in order to have access to the list tables, and the second connection with the platform database in order to have access to the platform data.

Record_Manager Function

The function `record_manager(type)` is responsible for presenting a list of the user patient records. The function may be called by users through the main menu. The records are organized in an HTML Table, and for each record the user may call the `update`, `view` or `delete` functions.

Questionnaire Function

The `Questionnaire()` function is responsible for creating the interface of the platform questionnaire. The function may be called by the system through the `create`, `update` and `view` functions. To create the interface of the tool, the function retrieves from the platform database table `questions` the attributes `text`, `type`, `values` and `condition`. The tool designs an HTML table called `tableQ` for the platform questionnaire where each row consists of two fields, the `text` and the `field` of the question. For each row (question) of the table, the platform tool creates the question `field` according to the question `type`. Also for each question, if the attribute condition is not `null`, the platform tool sets the current question to *disabled.* The question fields are defined as follows. For each:

- Multiple-choice question, the questionnaire creates a drop-down list with all the possible values, that are retrieved from the database table `questions`.
- Complex multiple-choice question, the questionnaire creates a *combo box* (see Section 4.2) with all the possible values, that are retrieved from the database table `questions`.
- List question, the function creates a drop-down list with all the list elements, that are retrieved from the database tables `dbtable_x` of home database.
- Date question, the function creates a date field. A date field consists of three drop-down lists (Day/Month/Year), where the user has to set the date.
- String question, the function creates a `text` field.
- Integer and decimal question, the function creates a `number` field.
- Complex-type question, the function creates a table of questions. A complex-type question consists of many questions. So, the complex question is a row in the HTML table that involves a smaller HTML table with the questions of the complex-type.
- Recurring question, the function creates a field where the user has to select the `recurrence` number. The recurring questions are groups of complex questions that are invisible from the user and when he selects the number of the complex questions that he needs, the system makes *visible* the selected number of the complex questions.

Create Function

The `create()` function is responsible for creating a new patient record, and may be called by users through the main menu. In the page loading, the `create` function calls the `questionnaire` function to design the interface of the tool. When the user ends the questionnaire filling, the system stores all the questions in the database

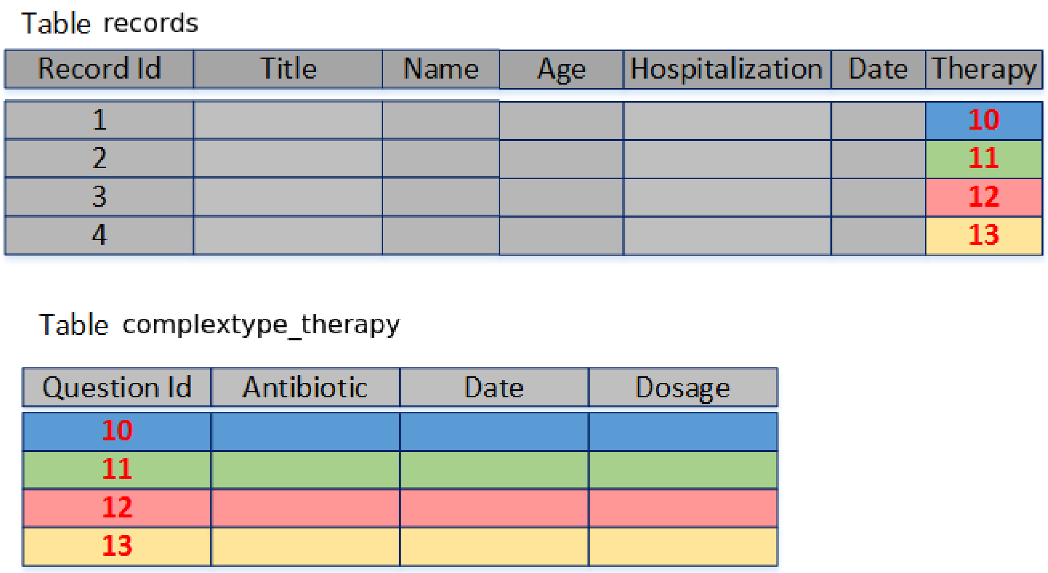


Table records

Record Id	Title	Name	Age	Hospitalization	Date	Therapy
1						10
2						11
3						12
4						13

Table complextype_therapy

Question Id	Antibiotic	Date	Dosage
10			
11			
12			
13			

Figure 4.5: Database tables records and complextype_therapy.

table `records`, except from the complex questions. Each complex question is stored to the corresponding complextype database table (see Figure 4.5). A row in a database table `complextype_x` stores the complex question data, and the `id` of this row is stored in the `records` table in the corresponding complex question.

Retrieve Function

The `retrieve(id)` function, takes as parameter the record `id`, and is responsible to retrieve the data of this patient from the platform database. It is called by `update` and `view` functions. The system retrieves the data from the platform database tables `records`, `questions`, `ct_questions` and `complexType_x`, and set the retrieved values in the HTML table `tableQ` of the questionnaire.

Update Function

The `update(id)` function is responsible for updating the patient records, and takes as parameter the record `id`. This function is called by users through the `record_manager` function. In the page loading, the function calls the `questionnaire` function to create the questionnaire interface, then calls the `retrieve` function to fill the questionnaire. When the user ends the update, the function updates the database table `records`, as well as the database tables `complexType_x`.

Delete Function

The `delete(id)` function takes as parameter the record `id`, and may be called by users through the `record_manager` function to delete a patient record. The function deletes the patient data from the database table `records`, as well as from the `complexType` tables.

Filtering_Tool Function

The `filtering_tool()` function is responsible for creating the filtering tool with which users may extract patient information from the database. The function may be called by users through the main menu. In the page loading, the function creates an HTML table where each row of the table has two *check-boxes* and a disabled *question filter*. The first *check-box* shows if the question will be exported, and the second *check-box* shows if there is a restriction over the specific question. If the second *check-box* turns to checked the question filter is enabled. The are two types of filters (see Figure 3.5). The questions that take number values (e.g., date, integer, decimal) have from-to filters, i.e., the user may set the *from* value, or the *to* value, or both of them. While the filters for the questions that take string values (e.g., multiple-choice, text, lists) are introduced by presenting *all distinct values* stored for the specific question and allowing the user to define the ones that satisfy the filtering criteria. For the last filter type we use a drop-down *checklist* (see Section 4.2), where the user may check multi values. When the user set all the restrictions over the questions, the system calls the Query function to create the SQL query, and finally submits it to the database.

Query Function

The `Query(array)` function is responsible for creating a large SQL query that is composed of the restrictions that the user defined in the filtering tool. Also, the function submits the SQL query to the database. Query function takes as parameters a multi-dimensional `array` with all the data that the user set to the filtering tool.

The procedure to create the SQL Query is described in the following. The format of this query is:

```
Select
   ExportQuestions
from
   FromTables
where
   FilterQuestions;
```

The ExportQuestions is the set of the questions that correspond to the checked check-boxes 1. The ExportQuestions has the format:

```
ExportQuestions=(Question1, Question2, ... , QuestionX);
```

The FromTables is the set of the database tables that the data will be extracted from, and its format is:

```
FromTables=(Records, Complextype1,Complextype2, ... , ComplextypeX);
```

And the FilterConditions is the set of the questions that correspond to the checked check-boxes2 and the restrictions over the questions. The format of the FilterConditions is:

```
FilterConditions= (Condition1 AND Condition2 AND ... AND ConditionX);
```

Each one of these conditions corresponds to one question, and consists of smaller conditions according to the filter type. For example a *start-end* filter creates a condition like

```
(Question1 > 5 AND Question1 < 65)
```

while a check-box-list filter creates a condition of format:

```
(Question1=A  OR Question2=B OR Question3=C)
```

The recurring questions have a different way for quering the data. The difference is that the user may add more than one restrictions over the recurring question. The format of a query over a recurring question has the format:

```
Condition1=(complexconditionA AND complexconditionB AND
complexconditionC);
```

A complexcondition is a restriction over a complex question, and as a complex question consists of many questions, the complexcondition involves one restriction for each one of the questions of the specific complex type. So, a complexcondition may be like this:

```
complexconditionA =((Question1=A OR Question1=B) AND
(Question2=C) AND (Question4<3 AND Question4 >1));
```

The HTML table of the filtering tool corresponds to the questionnaire. Each row of the HTML table is a column of database table `records`. So, if the user sets a restriction *Name equals to A*, the system looks in column `name` of table `records` for the value A (see Figure 4.5).

For the recurring questions of the questionnaire the search works in a different way. The platform, with the complex and recurring questions achieve better data modeling and more complex data retrieval. For example, we have a complex-type `therapy(antibiotic, dosage)`, the patient received five therapy cycles. When the clinician looks if the patient received the `antibiotic X`, the tool correlates the therapy questions, and understands that are similar objects. That means that the system understands that the `antibiotic` of question1 is the same information with the `antibiotic` of question2, question3, e.t.c. So, the filtering tool may find that patient received the `antibiotic X` in any of the therapy cycles (see Figure 4.5). This makes the tool very flexible and may achieve complex queries. The reason that the system can find the antibiotic in any of the therapy cycles is because the therapy questions are stored in a database table Therapy with attributes Antibiotic and Dosage, (i.e all the antibiotics belong to the same database table column). The system correlates that the question1 of the complex type is an antibiotic and search for A. Finally the system submits the query to the database.

Chart Function

The chart() function is responsible for visualizing charts of the stored patient data. It may be called by users through the main menu. The function takes as input the `number of variables`, the `variables` of the chart, the `type` of the chart and the `title` of the chart. The number of variables can take the values *one* or *two*. The variables can take as value one of the questions. The chart `type` can take one of the values *2D pie, 3D pie or donut* for the *one-variable* charts, and the values *Bar* or *column* for both *one* and *two* variable charts. For *one-variable* charts the system counts the `number` of patients that have each value of the selected question, while for two variables charts, the system groups `question1` by `question2`, and takes as result a *two-dimensional* array. Then the function creates a *google.visualization.DataTable* (see Section 4.2) called texttttData , where it stores the results array. Finally, the tool calls the `chart.draw(data, chart type)` function (see Section 4.2) with parameters the `DataTable Data` and the `chart type`, and finally draws the chart.

4.3.2 Platform Manager

In this subsection we describe some of the most important functions of the platform manager. The Platform Manager can be used only by ITAs and PAs. So, in this subsection when we talk about creating, updating and deleting users we refer to PAs.

Login Function

`Login(username, password)` function takes two parameters, the `username` and the `password`s that the user typed in the login screen. The function opens a database connection with `logindb` database and checks if the given `username/password` combination exists in database table `cloudusers`.

Database Connections

During the Platform Manager usage, the tool has to be connected with the databases of the system to retrieve and store data. When the user logs in the Platform Manager, the system opens two connections, one with the user's `home` database in order to have access to the `list tables`, and the other with the `logindb` in order to have access with the `db_info` table. These two connections are opened until the user logs out the platform manager. Also, when a user connects to the updating tool, the system also connects to the specific `platform/template` database.

Manager Function

The `Manager(type)` function is responsible for displaying the system elements in an HTML table, and takes as parameter the type of the system elements (`lists, platforms, users, templates` or `complex-types`). This HTML table supports a sort function which the user may call to sort the elements of the table alphabetically. The HTML table is developed using the TableSorter jQuery plugin (see Section 4.2). Also, through the function manager the users may call the functions:

- `Creating_tool` where the type parameter takes one of the values `platform, template, list, user` or `complex-type`.
- `Updating_tool` where the type parameter takes one of the values `platform, template, list, user` or `complex-type`.
- `DeleteDB` where the type parameter takes one of the values `platform, template, list, complex-type or user`.

Creating Tool Function

The function `Creating_tool(type)` is called through the `manager`. The function is responsible to collect the necessary information about the system element that will be created. The parameter `type` takes one of the values `platforms, templates, lists, complex-types` and `users`.

The `creating_tool` is the same for platform and template types. Firstly, the tool asks for the platform `name`, and calls the `check` function to certify if the name is unique. If the name is unique, the tool checks if the user wants to import a template in the platform, and ends the procedure. When the procedure ends, the system calls the `createDB(template)` function or the `createDB(platform)` function to create and initialize the database of the platform/template. Here we have to mention, that the system has created a platform with an empty questionnaire, and the user has to call the `update_tool` in order to add questions into the questionnaire.

When the `creating_tool` takes as parameter the value `complex-type` or `list`, asks from the user to type a unique `name` for the element. Then, the tool calls the `createDB` function to create the `complex-type`. Finally, when the creating tool is called for the `user` parameter, asks as input the user information, and calls the `createDB` function to create the user.

Updating_tool Function

The `updating_tool(type, id, mode)` function is used by PAs for updating the system elements, it takes three parameters the element `type` (platform, template, complex type, list or user), the element `id` , and the update `mode`. The `updating_tool` may be called by users through the `manager` function.

When the parameter `type` takes one of the values *Platform* or *Template*, the function creates an HTML `div` node called `Quest`, which includes three HTML `section` childNodes. Each HTML `section` corresponds to a different step of the `updating_tool`. Then, the system calls the `steps` function which is provided by jQuery Library Steps(see Section 4.2). The `steps` function converts the `div` to a wizard-like environment and each `section` is converted to a page/step in the wizard. In this wizard the users are navigated between the pages with *Next* and *Previous* buttons.

The first HTML `section` of this `div` includes an HTML `table` called `table1`. `Table1` keeps information that users give for the questionnaire structure. Also, this section corresponds to the questionnaire `design tool`. `Table1` consists of rows, each row corresponds to a *question*, and each *question* consists of three columns (i) question `text`, (ii) question `type`, (iii) question `values`. The question `value` field is adjusted in accordance with the question `type`. When the field datatype is changed to *multiple-choice/complex multiple-choice* the question `value` field turns into a multi-value `select2` box (see Section 4.2), where the user defines the multiple-choice possible values. If the datatype is changed to *list*, the `value` field turns into a `select2` drop-down list with the available lists as values. In any other case the value field is disabled. If the datatype changes to Complex-type, a pop-up window appears and asks from the user to select the complex-type name from a drop-down list and the maximum recurrence number of the question. Also, `table1` keeps a hidden variable that defines the position of the complex question. When the procedure ends, the system creates an HTML `list` of elements (hidden from the user) in the row that has as many child-nodes as the maximum number of the question. The user may easily add, remove and rearrange the `table1` rows by using the jquery functions `addrow`, `removerow` and `rearrange`. The function rearrange allows users to reorganize the table rows with drag-and-drop actions. For the implementation of the `rearrange` function we use the `draggable` and `sortable` plugins of jQuery UI Library (see Section 4.2). When the *Next* button of the HTML `section1` is clicked the second section appears.

The second HTML section corresponds to the `Branching_Logic tool`, and it is used by PAs to define the questionnaire's branching logic. The second HTML section includes an HTML table called `table2`. Each row in `table2` (see Figure 4.24) corresponds to a question Q and consists of six fields `id, text, type`, and `state` (i.e., if the question is follow-up question or not), the *master question* and the *enabling value* of a question. The `id, text, type` fields are passed to `table2` from `table1`. The *master question* is the fifth field of a row, and corresponds to the master question of the follow-up question Q. Also, the *enabling/disabling* value corresponds to the value of the *master question* that enables/disables the question Q. The enabling question is a drop-down list that contains all the previous multiple-choice and complex multiple-choice questions (see Figure 4.24), while the *enabling value* is a drop-down list that contains the values of the selected *master question* (see Figure 4.25). The system stores a string with format `(master question=enabling value)` in the database table `questions` for each specific question. When `table1` is updated by a PA, the system dynamically updates `table2`. The third HTML `section` includes some text fields for the study title and the study description.

The `updating_tool` provide PAs the `simple` and the `structural` updating modes. In the `simple` mode the datatypes are disabled, the user may only update the text and the values of the existing questions, so changes related to datatypes or question order maynot be performed in this mode. Also in `simple` mode, the PA may add new questions in the end of the questionnaire, and also he may update the branching logic. On the other hand, in `structural` mode the user may make any change that he wants to the platform questionnaire. When the update ends the system calls the `updateDB` function to update the database.

If the element type is *Complex-type*, the updating tool creates an HTML `div` node `Quest` that consists of the `section1` and corresponds to the questionnaire `design_tool`. The structure and the functions of this section are the same with the platform case that we described earlier with the difference that does not support the complex-type and title datatypes. The `updating_tool` supports two modes, the simple and the structural update that we described earlier. The structural update is available only if the platform tool is not in use by the platform. When the update ends the system calls the `UpdateDB` function to update the database.

When the element type is List, the function creates an HTML `table` called `ListTable`, that consists of rows, and each row is an element of the specific list. Each row consists of two fields, the `id` (that has to be unique) and the `name` of the element. The tool includes the appropriate functions for adding, deleting and saving the elements. The lists are stored in the home database of the user, so that they are visible from all platforms and templates of the user. When the update ends the system calls the `updateDB()` function to update the database.

Finally, when the element type is `user`, the `updating_tool` (i) creates an HTML `table`, (ii) retrieves from the database the information for the specific user, (iii) and sets the retrieved values to the HTML `table` fields. The ITA updates the values of the fields, and the system calls the `updateDB` function to update the database.

CreateDB Function

The createDB(type) function may be called through the `creating_tool`, and it is responsible for creating the elements of the system in the database. This function takes as parameter the element type (`platform, template, complex-type, user` or `list`).

When the element `type` is `platform`, and there is a `template` import, the system clones the `template` database and defines the clone as the `platform` database. Otherwise, the system creates and initializes the `platform` database. The initializations of the `platform` database includes the creation of tables `questions, records, users and ct_questions`. The database table `records` has as attributes the questions of the questionnaire, while the database table `questions` keeps information (`type, id, name,` e.t.c) for each question of the questionnaire (see Figure 4.6). When the element type is a `template`, the procedure is the same with the difference

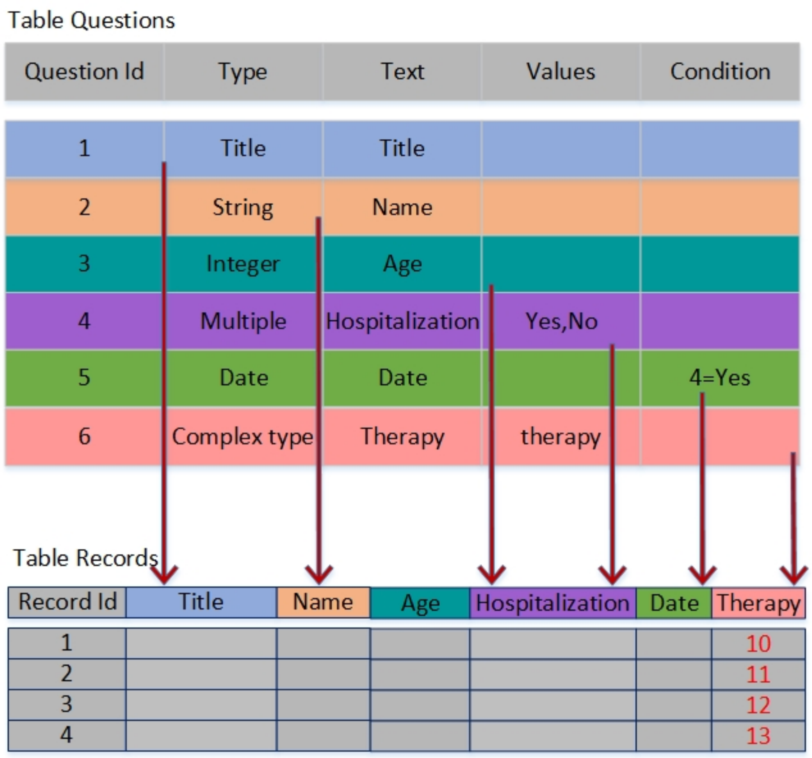


Table Questions

Question Id	Type	Text	Values	Condition
1	Title	Title		
2	String	Name		
3	Integer	Age		
4	Multiple	Hospitalization	Yes,No	
5	Date	Date		4=Yes
6	Complex type	Therapy	therapy	

Table Records

Record Id	Title	Name	Age	Hospitalization	Date	Therapy
1						10
2						11
3						12
4						13

Figure 4.6: Database tables Questions and Records.

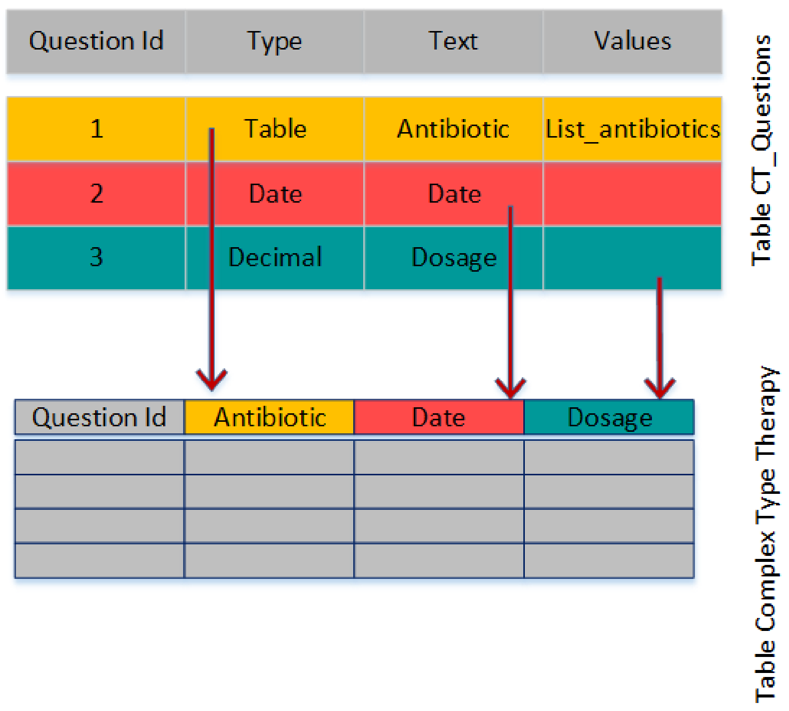


Table CT_Questions

Question Id	Type	Text	Values
1	Table	Antibiotic	List_antibiotics
2	Date	Date	
3	Decimal	Dosage	

Table Complex Type Therapy

Question Id	Antibiotic	Date	Dosage

Figure 4.7: Database tables ct_questions and complextype_therapy.

that the function defines the template as a `template` to the database, in order to be reusable. When the element type is `list`, the system creates a new table `dbtable_x` (x is the name of the list) in the home database of the user. When the element type is `complex-type`, the system creates a database table `complextype_x`, where x is the name that the user chose, and adds to table `ct_questions` information about each question of the complex-type. The database table `complextype_x` has as attributes the questions of the type, while the database table `questions` keeps information (`type, id, name,` e.t.c) for each question (see Figure 4.7). Finally, when the element type is `user`, the function asks as input the user's information, creates a new database role and a `home` database for the new user, and adds the new user in database table `cloudusers`.

UpdateDB Function

The function `updateDB(type, id, mode, array)` is responsible for updating the database with the changes that happened in the `updating_tool`. The `updatedb` function is called through the `updating_tool` and takes four parameters, the `type` of the system element (list, platform, template, user or complex-type), the `id` of the element, the update `mode` and a PHP array with all the changes that happened in the updating tool. The update mode takes one of the values *simple* and *structural* for the element types platforms, templates and complex types. For the element type list and user the update mode takes the *null* value.

If the type of the element is *platform* or *template*, (i) if the update `mode` is *simple*, the system updates the old questions of database table `questions`, adds the new questions (as rows) in the database table `questions` and finally adds the new questions (as columns) in database table `records`. (ii) If the update mode is *structural*, the system recreates the platform/template database, so the stored data are deleted. If the element type is complex-type, (i) if the update mode is *simple* the system updates the questions of database table `ct_questions`. Also, if there are new questions the function adds the new questions to `ct_questions` (as rows) and `complextype_x` (as columns) in the platform database tables. (ii) If the mode is *structural*, the function recreates the database `complextype_x`, and updates the `ct_questions` table. If the element type is user, the system updates the database table `cloudusers`. Finally, if the element type is list, the system updates the database table `dbtable_x`.

DeleteDB Function

The `deleteDB(type, id)` function takes as parameters the `type` (list, platform, template, user and complex-type) and the `id` of the element. The `deleteDB` function may be called through the `manager` function to delete a specific system element.

When the `type` parameter takes the value a *template* or *plarform*, the function drops the specific platform/template database and deletes the specific platform/template database information from the database table `db_info`. If the parameter type takes the value *complex-type*, the system checks if the type is in use by any platform/template, and if it is not in use, the function drops the database table `complextype_x`, and deletes the questions from the database table `ct_questions`. If the element `type` takes the value *list*, the system checks if the specific list is in use by any platform, template or complex-type, and if it is not, the function deletes the table `dbtable_x`. Finally, if the element type is *user*, (i) the system deletes the database role, including the databases of the role, and finally removes the user from the database table cloudusers.

4.4 User Guide

In this section we present a user guide for CloudStudy. In this user guide we describe the available tasks of each tool of the system, as well as the procedure that a user has to follow to complete a task, step-by-step. The user guide is organized in two parts, the platform and the platform manager.

4.4.1 Platform

In this subsection we describe the available tasks of each tool of the platform. To access in any platform task, the user has to login first to the Platform. At home page the user may has access to the main menu (see Figure 4.8), that consists of six tabs:

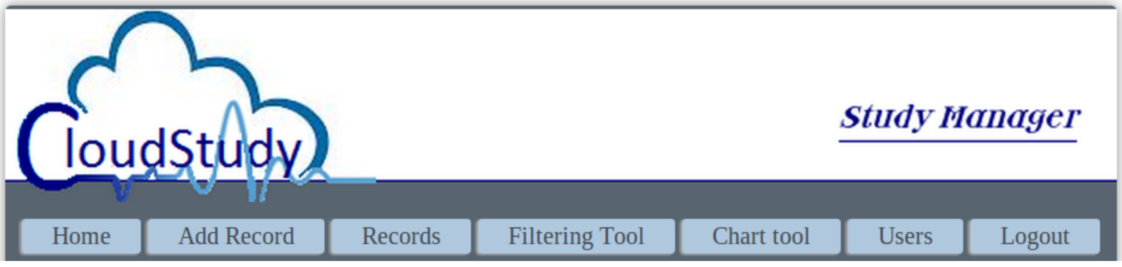


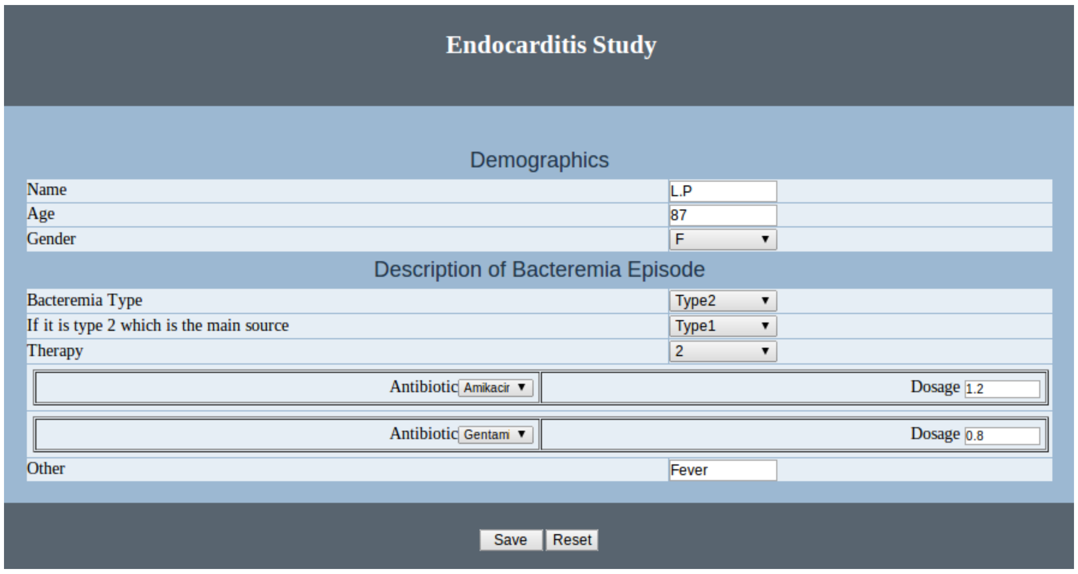
Figure 4.8: The main menu of a Platform.

1. The *Add Record* tab, where the user may create a new patient record (see Add Records).
2. The *Records* tab presents the record manager to the user. The record manager (see Update, View and Delete patient records) lists all the records of a user and allows him to update, view and delete his records.
3. The *Filtering Tool* tab presents the filtering tool to the user (see Filter and Export Data).
4. The *Chart Tool* tab presents the chart tool to the user (see Create Charts).
5. The *Users* tab is visible only to Plarform Administrators and allows them to access user manager (Create, Update, and Delete Users).
6. The *Logout* button disconnects the user from the system.

Add Records

To add a new record the user has to click the *Add Record* tab in the main menu. The tool for adding new records includes the platform questionnaire (see Figure 4.9), which the user has to fill with the patient data. To add a new record the user has to:

- Type/select the question values of the questionnaire.
- Click on a recurrence button to select how many recurring you need. ➪ The forms appear below the recurrence button.
- Click the *Save* button to save the record.



Endocarditis Study

Demographics

Name	L.P
Age	87
Gender	F

Description of Bacteremia Episode

Bacteremia Type	Type2
If it is type 2 which is the main source	Type1
Therapy	2
Antibiotic Amikacir	Dosage 1.2
Antibiotic Gentam	Dosage 0.8
Other	Fever

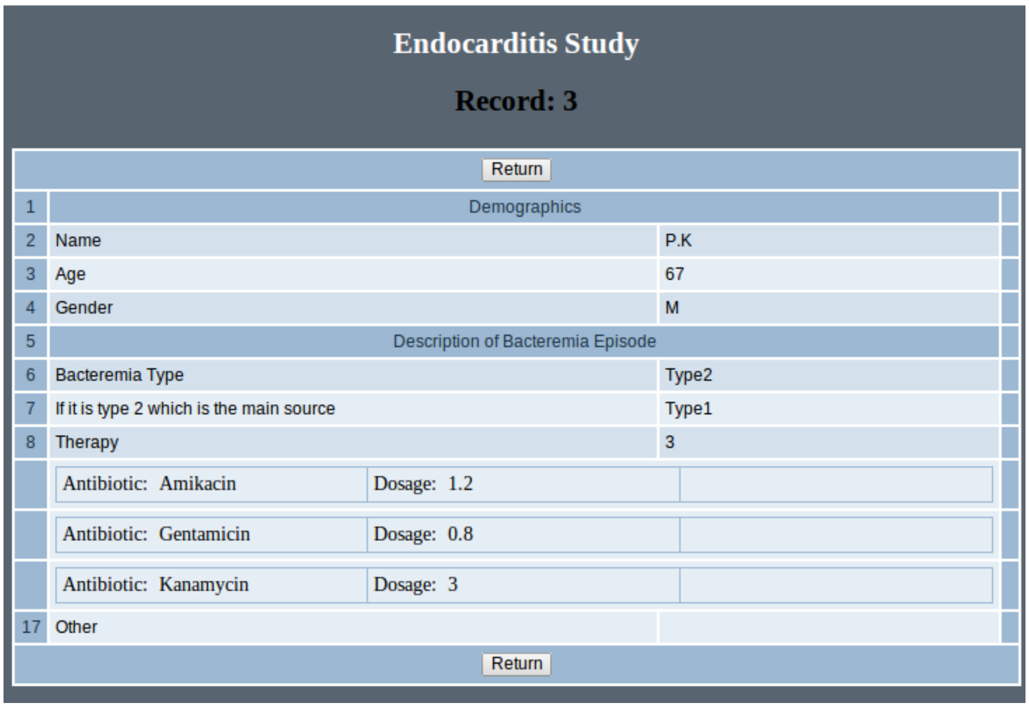
Figure 4.9: Add a patient record.

- Click the *reset* button to clear the record.

Update, View and Delete Patient Records

The platform users may access the record manager (see Figure 4.11) trough the tab *Records* of main menu. The users through the record manager may view, update or delete their records. The procedure of view, update or delete records is described in the following:

- To display the records, the user has to choose a record and click the *View* button (see Figure 4.10). ➪ The data cannot be modified in this mode.
- To update a record the user has to choose a record, and click the *Update* button. ➪ The data may be modified in this mode.
    1. The user makes the changes in the record, and clicks the *Save* button to save the changes, or the *Cancel* button to cancel the record update.
- To delete a record the user has to choose a record, and click the *Delete* button (see Figure 4.11).

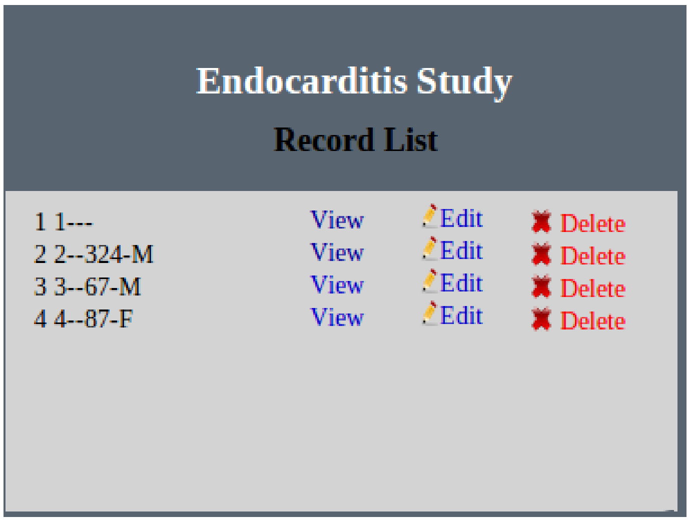


Endocarditis Study

Record: 3

1	Demographics	
2	Name	P.K
3	Age	67
4	Gender	M
5	Description of Bacteremia Episode	
6	Bacteremia Type	Type2
7	If it is type 2 which is the main source	Type1
8	Therapy	3
	Antibiotic: Amikacin	Dosage: 1.2
	Antibiotic: Gentamicin	Dosage: 0.8
	Antibiotic: Kanamycin	Dosage: 3
17	Other	

Figure 4.10: Record Preview.

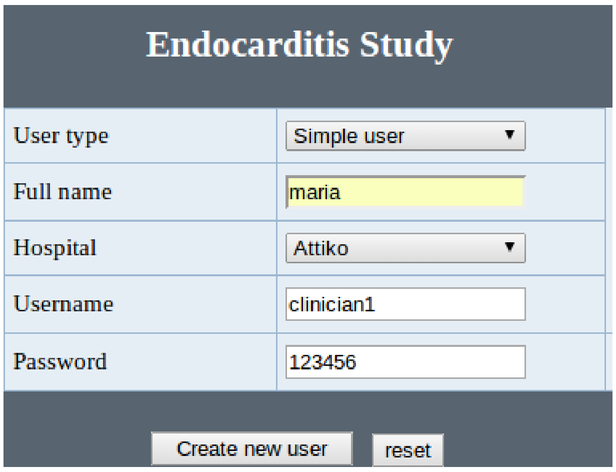


Endocarditis Study

Record List

1 1---	View	Edit	Delete
2 2--324-M	View	Edit	Delete
3 3--67-M	View	Edit	Delete
4 4--87-F	View	Edit	Delete

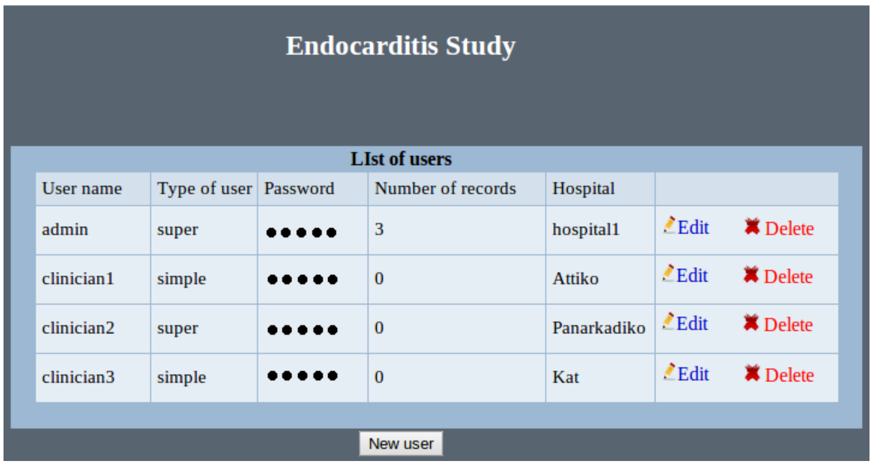
Figure 4.11: Record Manager.



Endocarditis Study

User type	Simple user
Full name	maria
Hospital	Attiko
Username	clinician1
Password	123456

Figure 4.12: Create a new platform user.



Endocarditis Study

LIst of users

User name	Type of user	Password	Number of records	Hospital		
admin	super	●●●●●	3	hospital1	Edit	Delete
clinician1	simple	●●●●●	0	Attiko	Edit	Delete
clinician2	super	●●●●●	0	Panarkadiko	Edit	Delete
clinician3	simple	●●●●●	0	Kat	Edit	Delete

Figure 4.13: User manager.

Create, Update and Delete Users

The PAs may access the User Manager (see Figure 4.13) through the *Users* tab of main menu. The User Manager is a tool that allows PAs to create, update or delete a platform user. The procedure for creating, updating or deleting the platform users is described in the following:

- To create a new platform user the PA has to click the *New User* button. ➪ The user creation form appears (see Figure 4.12).

    1. In the *user type* field, the PA selects the user type.

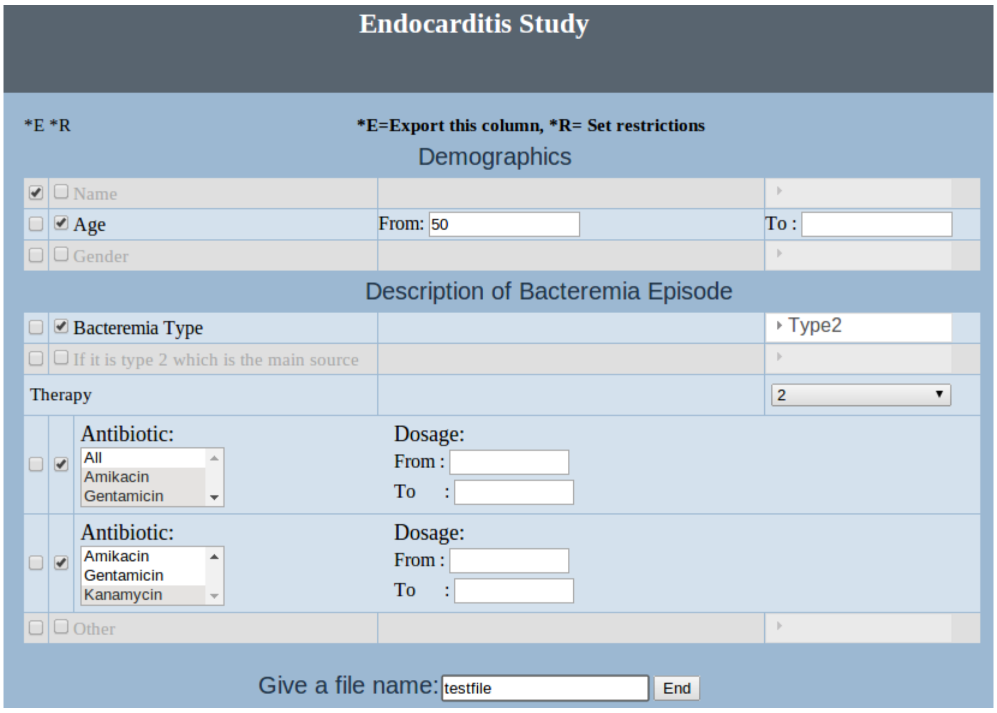
2. In the *fullname* field, the PA types the user full name.
3. In the *Hospital* field, the PA selects the hospital that the platform user works.
4. In the *Username* field, the PA types a unique username.
5. In the *Password* field, the PA types a user account password.
6. The user clicks the *Create* button. ➪ The user manager appears.

- To update a user, the PA has to choose a user, and click the *Update* button (see Figure 4.13).
  1. The PA makes any changes to the *user type, fullname, hospital, username or password* fields.
  2. The PA clicks the *Save* button to save the changes.
- To delete a platform user, the PA has to choose a user and click the *Delete* button (see Figure 4.13). ➪ His records now belongs to the PA that deleted him.

Filter and Export data

Through the tab *Filtering Tool* of main menu, the user has access to the filtering tool (see Figure 4.14). With the filtering tool the user may filter and export the filtered patient data. To filter the patient data the platform user has to:

1. Check in the first column, the check-boxes that he wants to export to the file.
2. Check in the second column, the check-boxes for the questions that he wants to add a restriction. ➪ The checked row will be enabled.
3. Add the restrictions over the data.
   - For From - To filters, he has to type the range of the values.
   - For Drop-down-checklists, he has to check the values of the questions that he wants to filter.


Endocarditis Study

*E *R *E=Export this column, *R= Set restrictions

Demographics

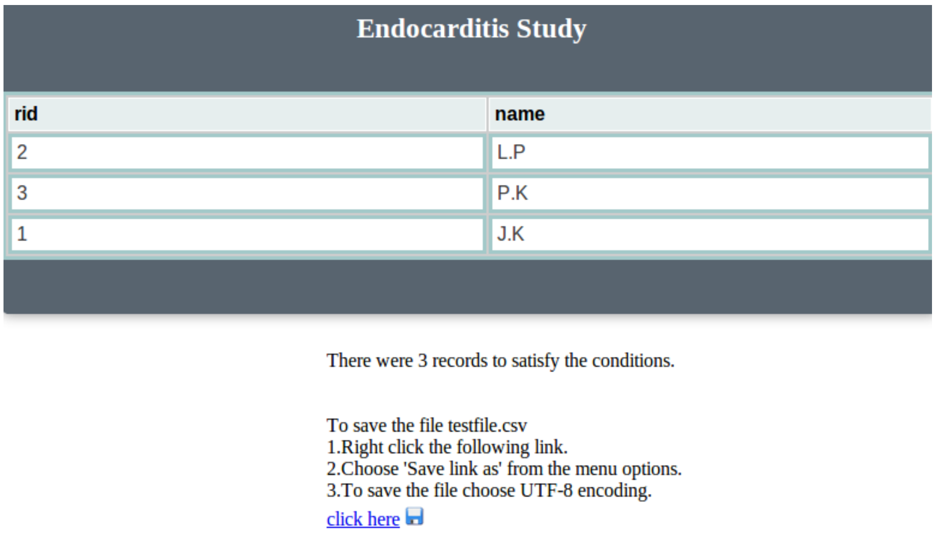
*E	*R			
✓	☐	Name		
☐	✓	Age	From: 50	To :
☐	☐	Gender		

Description of Bacteremia Episode

*E	*R			
☐	✓	Bacteremia Type		Type2
☐	☐	If it is type 2 which is the main source		
		Therapy		2
☐	✓	Antibiotic: All, Amikacin, Gentamicin	Dosage: From : To :	
☐	✓	Antibiotic: Amikacin, Gentamicin, Kanamycin	Dosage: From : To :	
☐	☐	Other		

Give a file name: testfile End

Figure 4.14: Filtering tool.


Endocarditis Study

rid	name
2	L.P
3	P.K
1	J.K

There were 3 records to satisfy the conditions.

To save the file testfile.csv
1.Right click the following link.
2.Choose 'Save link as' from the menu options.
3.To save the file choose UTF-8 encoding.
click here

Figure 4.15: An example of filtering tool results.

- For Drop-down lists, he has to check the number of the restrictions that he wants to submit for a recurring question.

4. Type a name for the exported file, in the *File Name* field. ➪ A table with the results appears.

5. Right click the link with the file name.

6. Choose 'Save link as' from the menu options (see Figure 4.15).

Open the exported files

The files that are exported from the CloudStudy platforms are Tab Separated Values (TSV). TSV is a file extension for a tab-delimiter file used with spreadsheet software (SPSS, Microsoft Office EXCEL, Open Office Calc, Google Docs, e.t.c.). To open the TSV file the user has to:

1. Open the file with a spreadsheet application.

2. Set the encoding to *UTF-8*.

3. Set the *Tab* character as column delimiter.

Create Charts

To create a chart the user has to access the Chart tool through the main menu. To create a new chart the user has to:

1. Select the number of variables from the *Number of Variables* drop-down list in the chart tool menu (see Figure 4.16).

2. Select the type the of the chart from the *Chart Type* drop-down list.

3. Select the first variable from the *Variable X* drop-down list.

4. If the two variables chart is chosen, the user has to select the second variable from the *Variable Y* drop-down list.

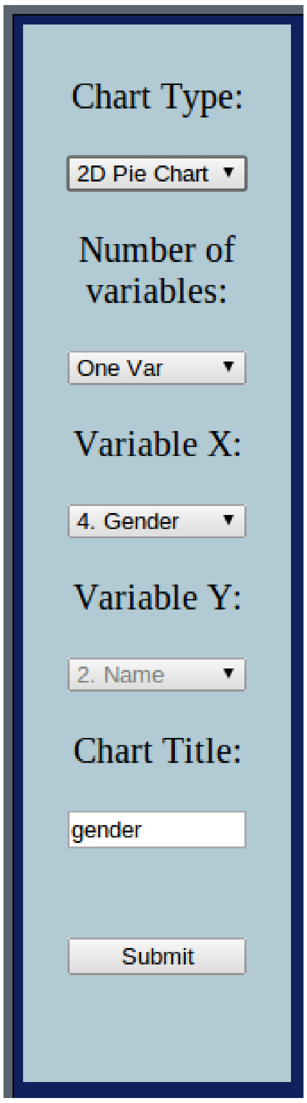


Chart Type: 2D Pie Chart

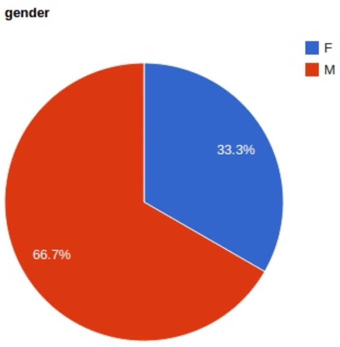
Number of variables: One Var

Variable X: 4. Gender

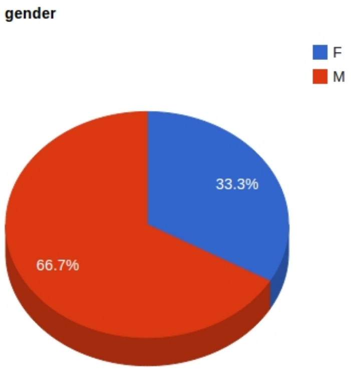
Variable Y: 2. Name

Chart Title: gender

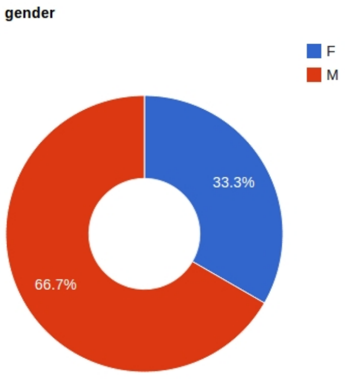
Chart Menu



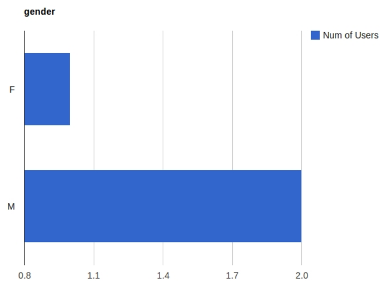
2D Pie Chart.



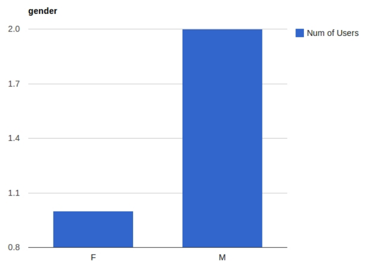
3D Pie Chart.



Donut Chart.



Bar Chart.



Column Chart.

Figure 4.16: One variable charts for the question Gender.

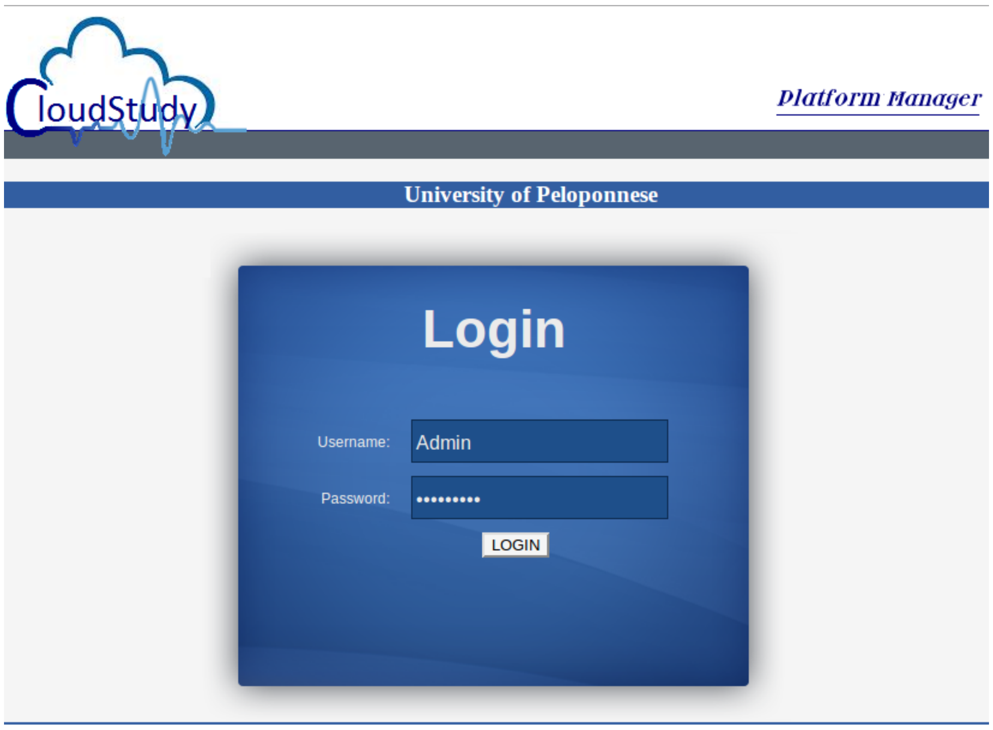


Figure 4.17: Plarform manager login screen.

5. Type the chart name in the *Chart Title* field.

6. Click the *Submit* button to draw the chart (see Figure 4.16).

4.4.2 Platform Manager

In this subsection, we describe the platform manager tools, and the procedure step-by-step that a user has to follow to complete different tasks using these tools. To use the platform manager the user has to login (see Figure 4.17) to the system. In the

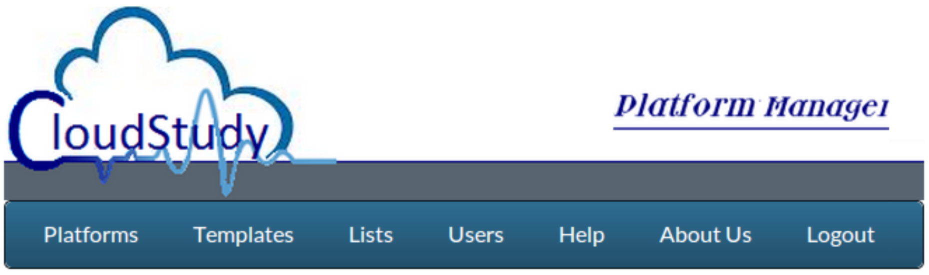
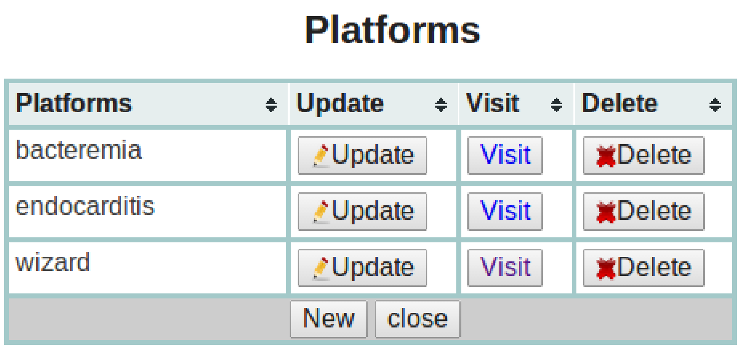


Figure 4.18: Platform Manager main menu.



Platforms

Platforms	Update	Visit	Delete
bacteremia	Update	Visit	Delete
endocarditis	Update	Visit	Delete
wizard	Update	Visit	Delete

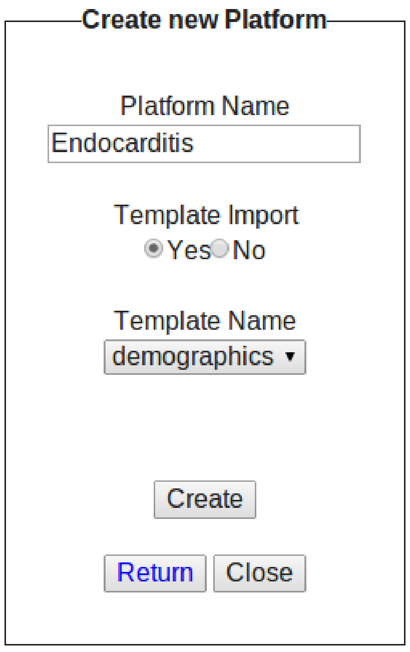
Figure 4.19: Platform Manager.

home page of the tool, the user see the main menu (see Figure 4.18) that consists of six tabs:

1. The *Plarform* tab presents the platform manager to the user. The platform manager lists all the platforms of a user and allows him to create (see Create Platforms), update (see Update Platforms), visit and delete (see Delete Platforms) his platforms.
2. The *Templates* tab presents the template manager to the user. The template manager lists all the templates of a user and allows him to create (see Create Templates), update (see Update Templates) and delete (see Delete Templates) his templates.
3. The *Lists* tab presents the list manager to the user. The list manager lists all the lists of a user and allows him to create (see Create Lists), update (see Update Platforms) and delete (see Delete Platforms) his platforms.
4. The *Help* tab presents a manual of the system to the user.
5. The *About Us* tab presents the CloudStudy site.
6. The *Logout* tab disconnects the user from the system.

Create Platforms/Templates

To create a new platform or a new template the user has access the platform/template manager (see Figures 4.19). The platform/template manager is a tool for


Create new Platform

Platform Name: Endocarditis

Template Import: [✓ Yes] No

Template Name: demographics

Figure 4.20: Platform/template creation screen.

creating, updating and deleting the platforms/templates of a user. To access the platform/template manager the user has to click the *Platform* or the *Template* tab of main menu. To create a new platform template the user has to:

1. Click the *New* button. ➪ The platform/template creation form appears (see Figure 4.20).
2. Type a unique name for the platform/template in the *Platform/Template Name* field.
3. Choose an option in the radio button *Import Template.* ➪ If the chosen option is *yes* the user has to select one template from the drop-down list.
4. Click the *Create* button. ➪ The platform/template manager appears.

Update Plarforms/Templates

To update a platform/template the user has to access the platform/template manager (see Figures 4.19). The platform/template manager is a tool for creating,

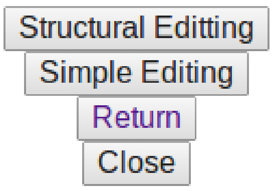
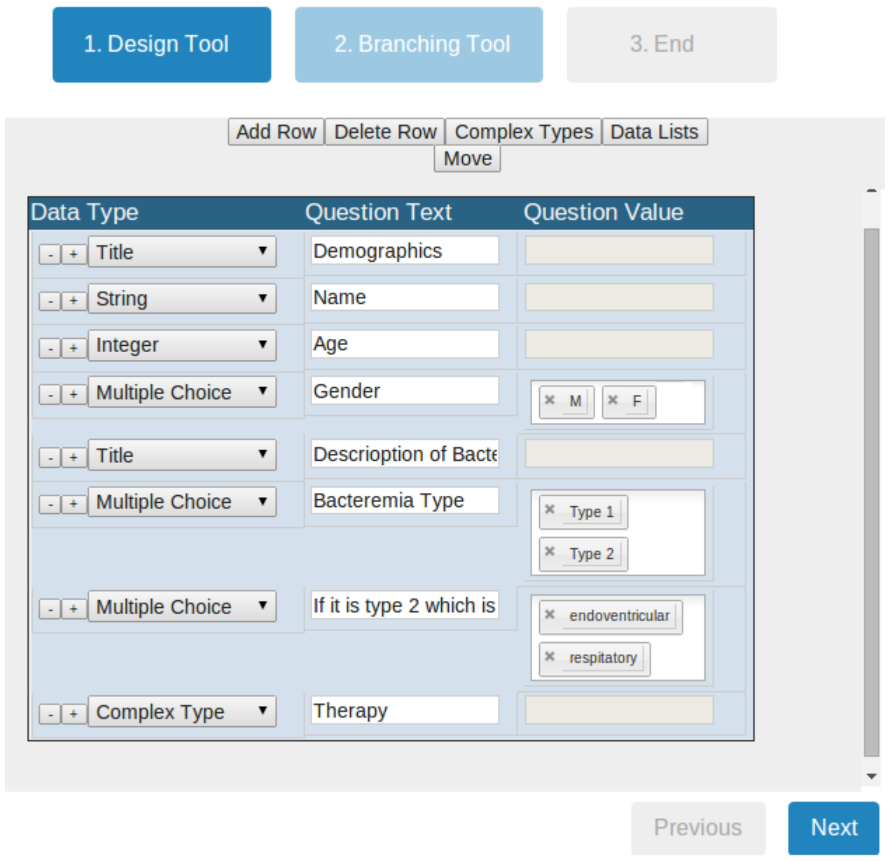
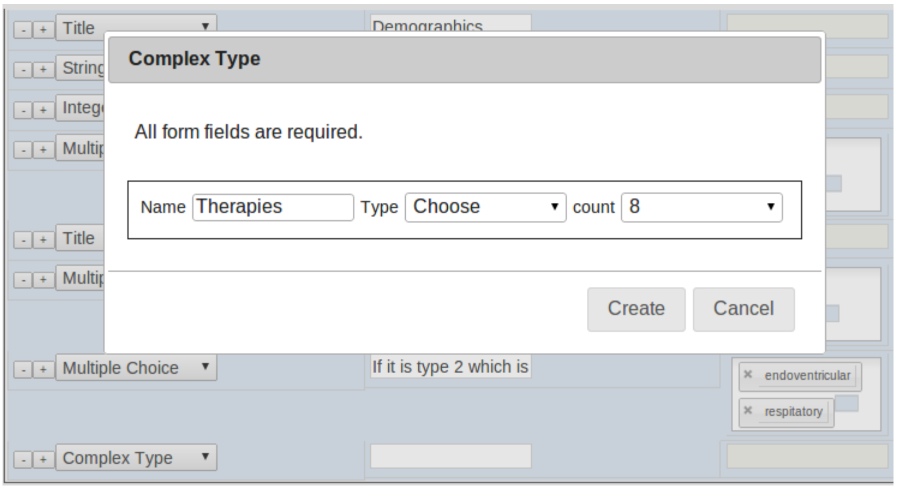


Figure 4.21: Choose the update mode.



Data Type	Question Text	Question Value
Title	Demographics	
String	Name	
Integer	Age	
Multiple Choice	Gender	M, F
Title	Descrioption of Bacte	
Multiple Choice	Bacteremia Type	Type 1, Type 2
Multiple Choice	If it is type 2 which is	endoventricular, respitatory
Complex Type	Therapy	

Figure 4.22: Platform questionnaire design tool.



- + Title	Demographics	
- + String		
- + Integ		
- + Multip		
- + Title		
- + Multip		
- + Multiple Choice	If it is type 2 which is	endoventricular, respitatory
- + Complex Type		

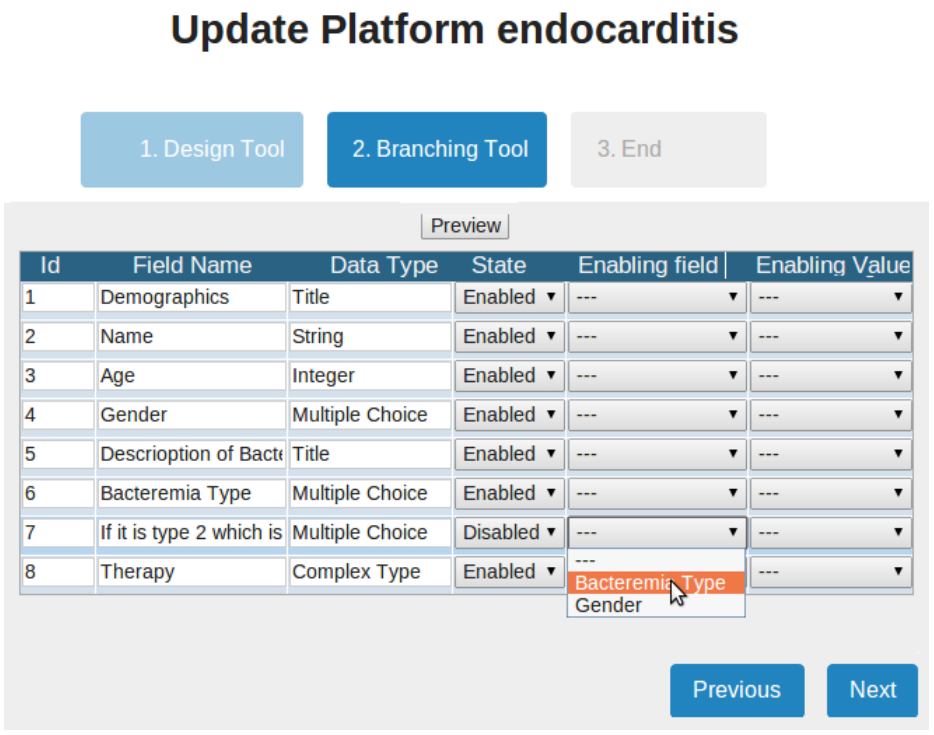
Complex Type

All form fields are required.

Name Therapies Type Choose count 8

Create Cancel

Figure 4.23: Add a complex type question in the questionnaire.



Update Platform endocarditis

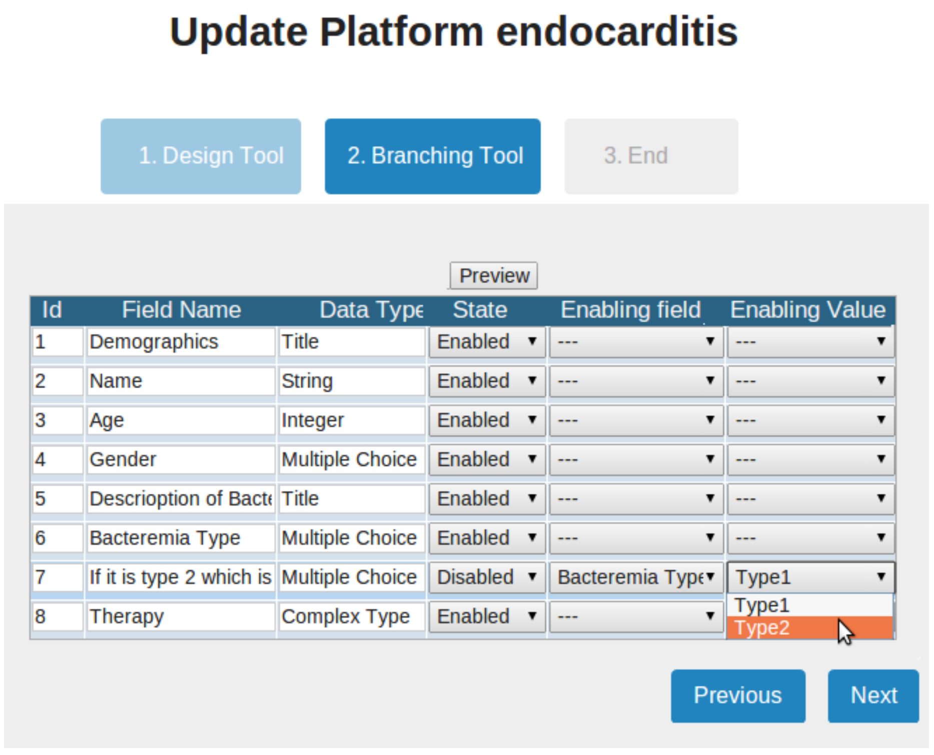
1. Design Tool 2. Branching Tool 3. End

Preview

Id	Field Name	Data Type	State	Enabling field	Enabling Value
1	Demographics	Title	Enabled	---	---
2	Name	String	Enabled	---	---
3	Age	Integer	Enabled	---	---
4	Gender	Multiple Choice	Enabled	---	---
5	Descrioption of Bacte	Title	Enabled	---	---
6	Bacteremia Type	Multiple Choice	Enabled	---	---
7	If it is type 2 which is	Multiple Choice	Disabled	---	---
8	Therapy	Complex Type	Enabled	--- Bacteremia Type Gender	---

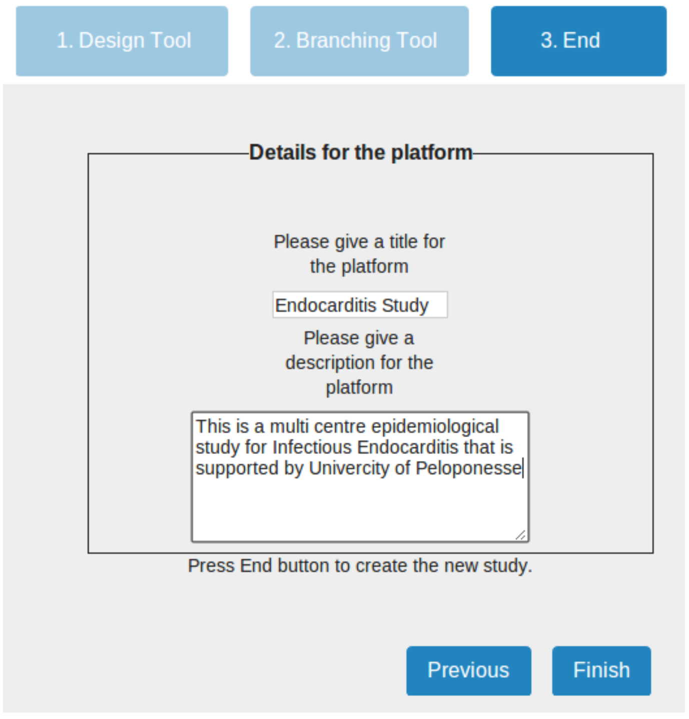
Previous Next

Figure 4.24: Branching logic tool: definition of enabling question.


Update Platform endocarditis

Id	Field Name	Data Type	State	Enabling field	Enabling Value
1	Demographics	Title	Enabled	---	---
2	Name	String	Enabled	---	---
3	Age	Integer	Enabled	---	---
4	Gender	Multiple Choice	Enabled	---	---
5	Descrioption of Bact	Title	Enabled	---	---
6	Bacteremia Type	Multiple Choice	Enabled	---	---
7	If it is type 2 which is	Multiple Choice	Disabled	Bacteremia Type	Type1
8	Therapy	Complex Type	Enabled	---	Type1 / Type2

Figure 4.25: Branching logic tool: definition of enabling value.



Details for the platform

Please give a title for the platform

Endocarditis Study

Please give a description for the platform

This is a multi centre epidemiological study for Infectious Endocarditis that is supported by Univercity of Peloponesse

Press End button to create the new study.

Figure 4.26: Update a platform Step3.

updating and deleting the platforms/templates of a user. To access the platform/template manager the user has to click the *Platform* or the *Template* tab of main menu. The procedure of updating a platform or a template is exactly the same for both. To update a platform or a template the user has to:

1. Click the *Update* button. ➪ The platform/template updating tool appears, it consists of three steps.

2. Select the *Simple* or the *Structural* mode button (see Figure 4.21). ➪ The user has to be careful in the *Structural mode*, as the stored patient data will be deleted.

3. Design the questionnaire (see Figure 4.22):

   - To add a question in the bottom of the questionnaire the user has to click the *Add question*, or click the (+) button of a question to add a new one below this question.
   - To remove a question the user has to click the (-) button of the specific question, or he has to click the *Remove* button to delete the last question of the questionnaire.
   - To reorganize the questions the user has to click the *Move* button, and drag-and-drop the questions to reorganize them. ➪ To terminate the reorganizing he has to click the *End Move* button.
   - To access the complex type manager, the user has to click the *Complex Type* button (see Figure 4.31).
   - To access the list manager the user has to click the *Lists* button (see Figure 4.27).
   - To set the question type the user has to select one type from the datatype drop-down menu of the question.
     - If the question type is Multiple Choice/Complex Multiple Choice, the user has to set the values of the question to the values field.
     - If the question type is List, the user has to choose one of the lists from the drop-down menu *values*.

  - If the question type is Complex Type, a pop-up window (see Figure 4.23). In name field the user has to type the name of the question. In the Type drop-down menu he has to select one of the complex types. In the Recurrence drop-down menu the user has to set the maximum recurrence of the question. Click *Create* button. ➪ The pop-up menu closes.

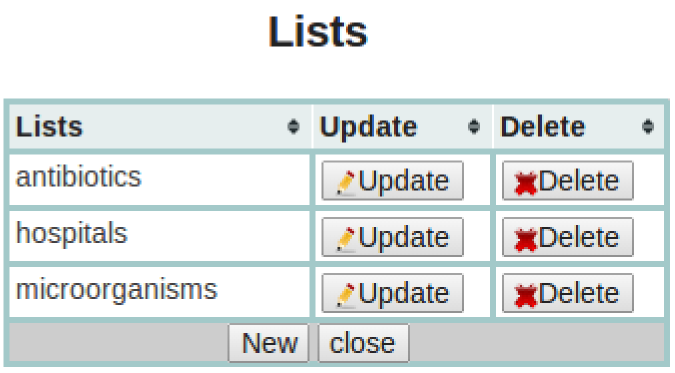
4. Click the *Next* button. ➪ The Branching tool appears (see Figure 4.24).
5. Click the *State* drop-down menu and set a question to *Disabled* for disabling it.➪ The *Enabling Question* field of this question will be enabled.
6. Click the *Enabling Question* field of the question and select from the drop-down menu which question will enable this question.
7. Click the *Enabling Value* field of the question and select from the drop-down which value of the question enables this question.
8. Click the *Next* button. ➪ The third step appears (see Figure 4.26).
9. Type a title for the platform's questionnaire, and in the description field he has to type a description for the study.
10. Click *Finish* button. ➪ The platform/template manager appears.

Delete Platforms/Templates

To delete a platform/template a user has to access the platform/template manager. The user has to click the *Delete* button to delete a platform/template (see Figure 4.19).

Create Lists

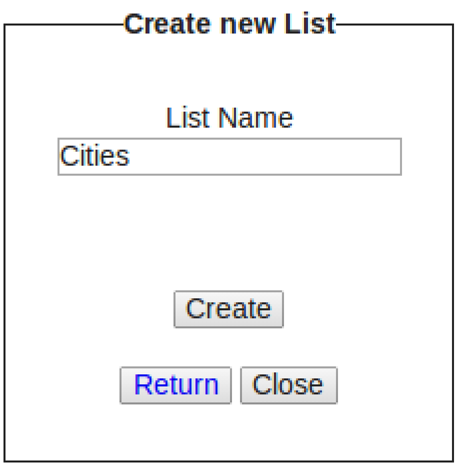
To create a new list of elements the user has to access the list manager. The list manager fig:pmlists is a tool for managing the system lists. Through the list manager the user may create, update or delete a list. The list manager in four different ways. It may be accessed from the *Lists* tab of main menu, or from the



Lists

Lists	Update	Delete
antibiotics	Update	Delete
hospitals	Update	Delete
microorganisms	Update	Delete

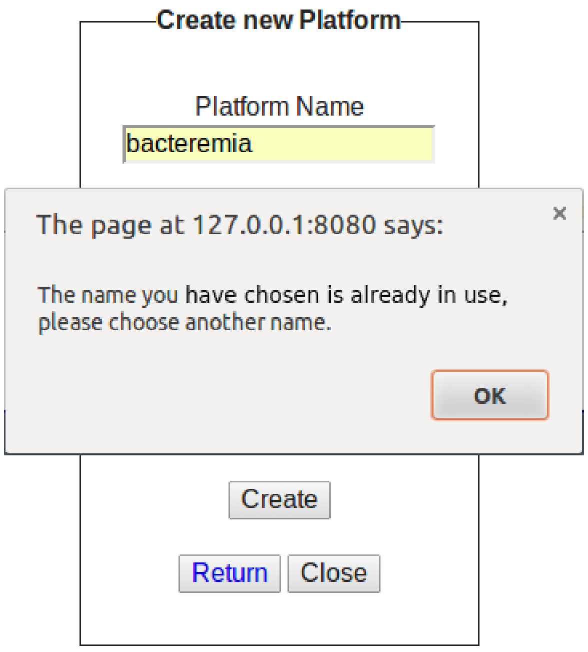
Figure 4.27: List manager.



Create new List

List Name: Cities

Figure 4.28: Create a new list.



bacteremia

The page at 127.0.0.1:8080 says:

The name you have chosen is already in use, please choose another name.

Figure 4.29: Warning that the given name is not unique.

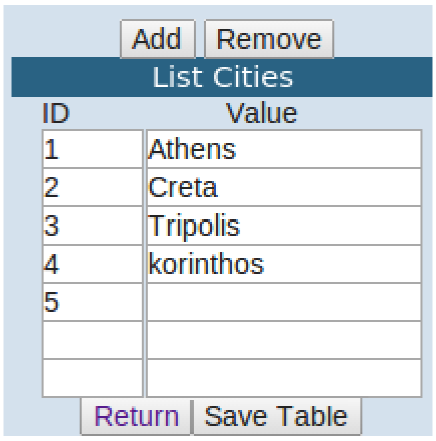
platform/template/complex type updating tool. To create a new list the user has to:

1. Click the *New* button. ➪ The List creating form appears (see Figure 4.28).
2. Type a unique list name in the *Name* field (see Figure 4.29).
3. Click the *Create* button. ➪ The list manager appears.

Update Lists

To update a list of elements the user has to access the list manager. The list manager is a tool for managing the lists. Through the list manager the user may create, update or delete a list. The list manager in four different ways. It may be accessed from the *Lists* tab of main menu, or from the platform/template/complex type updating tool. To update a list the user has to:

1. Click the *Update* button of the list that that he wants to update. ➪ A table with the list elements appears (see Figure4.30).



List Cities	
ID	Value
1	Athens
2	Creta
3	Tripolis
4	korinthos
5	

Figure 4.30: Update a list.

- Click the *Add* button to add a new element at the bottom of the table.
- Click the *Delete* button to delete the last element.
- Rephrase any old list elements (Not the ids).

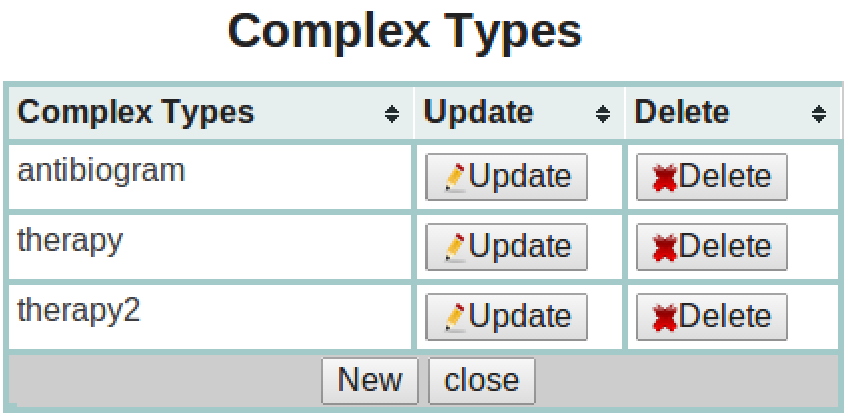
2. Click *Save* button to store the changes. ➯ The list manager appears.

Delete Lists

To delete a list of elements the user has to access the list manager. The list manager is a tool for managing the lists. Through the list manager the user may create, update or delete a list. The list manager may be accessed in four different ways. It may be accessed from the *Lists* tab of main menu, or from the platform/template/-complex type updating tool. Here we describe the procedure of deleting a list. To delete a list the user has to be click the *Delete* button of the list that he wants to delete.

Create/Update/Delete Complex Types

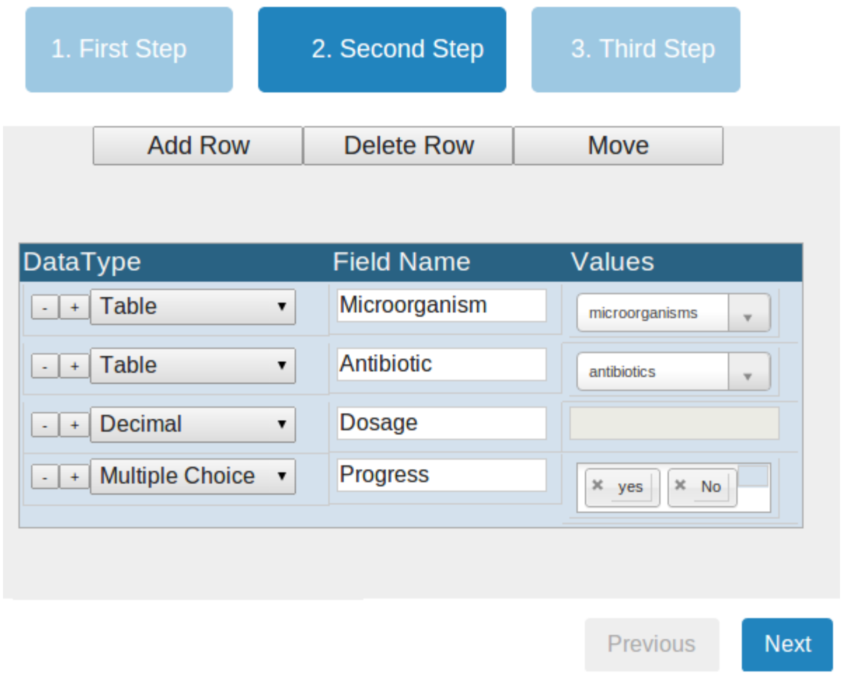
The complex type manager (see Figure 4.31) is a tool for managing the complex types of a platform/template. Through the complex type manager the user may create, update or delete a complex type. The complex type manager may be accessed through the platform/template updating tool. Here we describe the procedure of creating, updating and deleting a complex type.



Complex Types

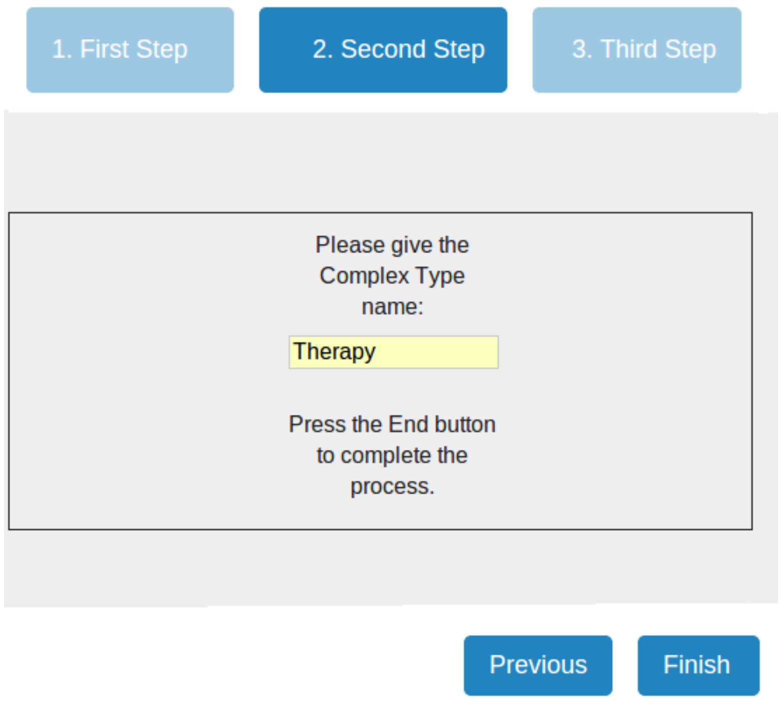
Complex Types	Update	Delete
antibiogram	Update	Delete
therapy	Update	Delete
therapy2	Update	Delete

Figure 4.31: Complex type manager.



DataType	Field Name	Values
Table	Microorganism	microorganisms
Table	Antibiotic	antibiotics
Decimal	Dosage	
Multiple Choice	Progress	yes, No

Figure 4.32: Create a complex type: step1.



Please give the Complex Type name:

Therapy

Press the End button to complete the process.

Figure 4.33: Create a complex type: step2.

1. The user has to click the platform or template tab according to what he wishes to update.

2. The user has to choose the *update* button of the specific platform/template. ➪ The questionnaire design tool for the platform appears.

3. The user has to click the *Complex Types* button. ➪ The pop-up window of the complex manager appears (see Figure 4.31).

   - To create a new complex type the user has to click the *Create* button (see Figure 4.32). ➪ The questionnaire design tool for the complex type appears. In the questionnaire design tool the user:

     – To add a new question in the bottom of the questionnaire the user may click the *Add question*, or he may click the (+) button of a question to add a new question below it.

     – To remove a question the user may click the (-) button, or he may click the *Remove* button to delete the last question.

     – To reorganize the questions, the user may click the *Move* button and drag-and-drop the questions. ➯ To terminate the question reorganizing, the user has to click the *End Move* button.

  - To add a new list, he may access the list manager through the the *Lists* button.
  - To set a question datatype, he has to select a type from the datatype drop-down menu of a question.
    * If the question type is Multiple Choice/Complex Multiple Choice, he has to set the values of the question to the values field.
    * If the question type is List, he has to choose one of the lists from the drop-down menu *values*.
  - To appear the next step the user has to click the *Next* button (see Figure 4.33).
  - In *Name* field, the user has to type a unique name.
  - To finish the creation, the user has to click the *Finish* button. ➯ The platform/template manager appears.

- To update the complex type, the user has to click the *Update* button of the complex type. ➯ The procedure is the same with the creation.
- To delete a complex type the user has to select the complex type and click the *Delete* button.

4.5 Case Studies and Results

CloudStudy has been used for creating two platforms for two multi-centre epidemiological studies for Hellenic Chemotherapy Society. The first study is a case study for Infectious Endocarditis, and the second is a case study for Bacteremia/Fungemia in Intensive Care Units. In this section we present the case study for the Infectious Endocarditis, and some of the results drawn out of the usage of such a system.

The infectious endocarditis (IE), is a disease with significant morbidity and mortality that vary internationally. The Hellenic Society of Chemotherapy recognized this fact, and organized a multicentre epidemiology study. The multicentre study was carried out using a platform created by CloudStudy. The

Staphylococcus aureus	n = 40, MRSA n = 7
Viridans group streptococci	n=20
Enterococci	n=16
Fungi	n = 3
Coxiella burnetii	n=3
Gram(-)	n=5
Group D Streptococci	n = 2
CoNs	n = 11
Other	n = 6
Negative	n = 24 (19%)

Table 4.1: Most Common pathogens for Infectious Endocarditis

project was supported by the Hellenic Society for Chemotherapy, University Hospital Attikon, and University of Peloponnese. While in the multi-centre study participated more than 20 clinics/hospitals of Attica, Thessaloniki, Patras, Heraklion, Larissa, Alexandroupolis.

The study[33] concerns patients with natural and prosthetic valves and implantable cardiac devices since 2011. Epidemiological data (demographics, clinical presentation, diagnosis, microbiological documentation, cardiac treatment and antimicrobial therapy and outcome) was recorded and systematically analyzed. The clinicians collected 130 patient records, the 127 of them satisfied the criteria of the infectious endocarditis. Some of the results that the clinicians came by using the platform are described in Tables (4.2,4.1).

The clinicians, analyzed the collected patient records using the platform filtering and chart tools, and concluded that: (i) The most incidents of IE where from the community and less were iatrogenic. (ii) A significant percentage of the infected users had use substances. (iii) The most common pathogen is the Staphylococcus aureus. Finally, (iv) high percentage IE that had successful cardiosurgery treatment.

Men	76,4%
Average age	56.2 years (1-85)
Diabetes	n = 22
Arterial hypertension	n = 47
Heart failure	n = 22
Rheumatic fever	n = 8
Chronic renal failure	n = 15
Hepatitis C	n = 15
HIV	n = 5
Use substances	n = 21
Predisposition of physical valve for IE	n = 27
Prosthetic valve	n = 23
Duration of symptoms IE less than 1 month	n = 40
Antimicrobial treatment before diagnosing IE	50%
Transthoracic ultrasound	n = 39
IE concerned left cavities	70%
Blood cultures during the treating	95%
Iatrogenic endokarditis	n= 20 (15.7%)
Mortality	n = 24

Table 4.2: Statistics for Infectious Endocarditis

Chapter 5

Conclusions and Future Work

In this last chapter of the thesis, we summarize our main achievements, discuss our contributions, and we highlight important conclusions drawn by the design and usage of such a system. Subsequently we present our main goals and future plans for the CloudStudy system.

5.1 Summary and Conclusions

In this work, we created an Electronic Health System for managing, sharing, organising, analysing and querying clinical and patient data collected from epidemiological multi-centre studies. The system that we developed, coined CloudStudy, covers all the functional requirements posed by multi-centre studies, and enables researchers to easily organise and share data and knowledge generated by the research activity. CloudStudy is an innovative, integrated framework for creating platforms for multi-centre studies, that enables users with no prior IT knowledge to design and launch, in an easy and transparent way, platforms tailored to the specific needs of their studies. The generated platforms allow users to record, organise and manage clinical/patient data by resorting to a number of built-in and customisable data entry forms. Also, CloudStudy allows users to search and filter information by using a powerful yet simple point-and-click mechanism that poses restrictions on the stored data and extracts the requested information in a number of formats/outputs including raw data, pie/column charts, and ready-to-process spreadsheets.

There are many benefits of using Electronic Health Systems as they help clinicians achieve better organisation, faster and easier access to patient data, easier sharing of clinical and patient data, and improve the quality of health care by reducing the medical errors. CloudStudy goes beyond that by offering a zero-cost, zero-administration web application that supports both fundamental and advanced user and data management functionality for multicentre studies. To the best of our knowledge, this is the first web-based system that focuses on multi-centre studies and allows users to deploy their own data management platforms within minutes, alleviating the need to rely on expensive custom-made solutions that require IT infrastructure and skills to maintain.

In this thesis we presented the architectural considerations and solutions behind the proposed tool, and we described a number of novel services that allow users without any prior IT knowledge to create, administer, launch, and use personalised data management platforms. To validate CloudStudy usefulness, we presented two case studies that have resorted on CloudStudy, to manage the distribution of partners and data, and highlighted the keyfindings of these studies, produced by resorting to the analytical tools provided by CloudStudy.

5.2 Future Work

In this section we discuss the possible extensions of CloudStudy, and the directions that we plan to focus in our future work.

Our future plans involve extra flexibility on the data collection and storage by (i) supporting more question types like radio buttons, checklists, tables, e.t.c., (ii) supporting calculated forms (e.g., SOFA, SAPS, APACHE scores) that the user will create dynamically during the platform design, and (iii) to support file and image storage (e.g., X-ray radiographies, magnetic images, ultrasonographies, endoscopies, e.t.c.). Our future work on the data filtering and optimization includes (i) improving the filtering tool which now supports conjunctions and disjunctions, so as to support also negation filters for the patient data, and (ii) improving the chart tool by supporting more chart types, more than two chart variables, and more interactive and effective ways to visualise the collected data. Another part of the system that we would like to improve on is the statistical part by supporting more statistical calculations. Finally, other plans include implementing communication services (e.g. instant messages) between the platform users as we believe that would improve the knowledge and data sharing, supporting more export formats and data types, dealing with legacy issues by supporting heterogeneous data representations, and improving the security by performing encryption algorithms on patient data.

CloudStudy is currently under alpha testing for multi-centre studies led by the Hellenic Society for Chemotherapy and the University Hospital Attikon, and has already been used by more than 20 public hospitals in Greece. Our goal is to expand the usage of the system to more clinicians and hospitals and aid them at performing more large-scale multicenter studies. This will help us collect feedback on possible improvements and shortcomings of the system, and target us towards improving both system usability and functionality.

Bibliography

[1] A. N. et al., "A global approach to the management of emr (electronic medical records) of patients with hiv/aids in sub-saharan africa: the experience of dream software," *BMC Medical Informatics and Decision Making*, 2009.

[2] J. Lee, "Emr management system for patient pulse data," *Med. Syst.*, 2012.

[3] H. F. et al., "An information system and medical record to support hiv treatment in rural haiti," *British Medical Journal*, 2004.

[4] P. Harris, R. Taylor, R. Thielke, J. Payne, N. Gonzalez, and J. Conde, "Research electronic data capture (redcap) – a metadata-driven methodology and workflow process for providing translational research informatics," *Biomedical Informatics*, 2009.

[5] M. Jager, L. Kamm, D. Krushevskaja, H. Talvik, J. Veldemann, A. Vilgota, and J. Vilo, "Flexible database platform for biomedical research with multiple user interfaces and a universal query engine," in *DB&IS*, 2008.

[6] "Cascade: Concerted action on seroconversion to aids and death in europe." Accessible at: http://www.ctu.mrc.ac.uk/cascade/.

[7] "Hicdep: Hiv cohorts data exchange protocol." Accessible at: http://www.hicdep.org/.

[8] "Cobred: Colon and breast cancer diagnostics." Accessible at: http://www.cobred.eu/.

[9] A. Tsafara, C. Tryfonopoulos, and S. Skiadopoulos, "Cloudstudy: A cloud-based system for supporting multi-centre studies.," in *In Proceed-*

ings of the 13th IEEE International Conference on BioInformatics and BioEngineering (BIBE), 2013.

[10] A. Tsafara, C. Tryfonopoulos, and S. Skiadopoulos, “Cloudbased data management for multicentre biomedical studies,” 2014.

[11] “Electronic health systems.” Accessible at: http://en.wikipedia.org/wiki/Electronic_health_record.

[12] C. Adamson and A. Wood, “Dfbidb: a software package for neuroimaging data management,” *Neuroinformatics*, 2010.

[13] J. V. Horn and A. Toga, “Is it time to re-prioritize neuroimaging databases and digital repositories?,” *Neuroimage*, 2009.

[14] A. Pozamantir, H. Lee, J. Chapman, and I. Prohovnik, “Web-based multicenter data management system for clinical neuroscience research,” *Med. Syst.*, 2010.

[15] M. Arya, W. Cody, C. Faloutsos, J. Richardson, and A. Toga, “Qbism: A prototype 3-d medical image database system,” *IEEE Data Eng. Bull.*, 1993.

[16] S. Zasadaa, T. Wangb, A. Haidara, E. Liub, N. Grafc, G. Clapworthyb, S. Manosa, and P. Coveneya, “Imense: An e-infrastructure environment for patient specific multiscale data integration, modelling and clinical treatment,” *Computational Science*, 2012.

[17] M. Lopez-Nores, Y. Blanco-Fernandez, J. Pazos-Arias, and J. Garcia-Duque, “The icabinet system: harnessing electronic health record standards from domestic and mobile devices to support better medication adherence,” *Computer Standards and Interfaces archive*, 2012.

[18] L. Sørensen, P. Lo, H. Ashraf, J. Sporring, M. Nielsen, and M. Bruijne, “Learning copd sensitive filters in pulmonary ct,” in *MICCAI (1)*, pp. 699–706, 2009.

[19] S. J. Fonda, R. J. Kedziora, R. A. Vigersky, and S.-E. Bursell, “Evolution of a web-based, prototype personal health application for diabetes self-management,” *Journal of Biomedical Informatics*, vol. 43, no. 5-Supplement-1, pp. S17–S21, 2010.

[20] S. J. Fonda, R. J. Kedziora, R. A. Vigersky, and S.-E. Bursell, "Evolution of a web-based, prototype personal health application for diabetes self-management," *Journal of Biomedical Informatics*, vol. 43, no. 5-Supplement-1, pp. S17–S21, 2010.

[21] T. D. S. Mabotuwana and J. Warren, "Chronomedit - a computational quality audit framework for better management of patients with chronic conditions," *Journal of Biomedical Informatics*, vol. 43, no. 1, pp. 144–158, 2010.

[22] K. A. Siek, S. E. Ross, D. U. Khan, L. M. Haverhals, S. R. Cali, and J. Meyers, "Colorado care tablet: The design of an interoperable personal health application to help older adults with multimorbidity manage their medications," *Journal of Biomedical Informatics*, vol. 43, no. 5-Supplement-1, pp. S22–S26, 2010.

[23] "Evidence-based careflow management systems: the case of post-stroke rehabilitation,"

[24] "Interacting agents through a web-based health serviceflow management system,"

[25] "Dynamic bayesian networks as prognostic models for clinical patient management,"

[26] "The management and integration of biomedical knowledge: Application in the health-e-child project,"

[27] M. Martinez, J. Vazquez, M. Lopez, F. Arnal, B. Gonzalez-Conde, J. Pereira, and A. Pazos, "Semantic integration of data in an information system for multicenter epidemiological studies on cancer," in *MIE2008*, 2008.

[28] H. B. et al., "Msbase: an international, online registry and platform for collaborative outcomes research in multiple sclerosis," *Multiple Sclerosis*, 2006.

[29] W.-S. L., J. Y., Y. Y., and J. Z., "Xbase: cloud-enabled information appliance for healthcare," in *EDBT*, 2010.

[30] S. L. Delp, J. P. Ku, V. S. Pande, M. A. Sherman, and R. B. Altman, "Simbios: an nih national center for physics-based simulation of biological

structures," *JAMIA*, vol. 19, no. 2, pp. 186–189, 2012.

[31] T. Kirsten, J. Lange, and E. Rahm, "An integrated platform for analyzing molecular-biological data within clinical studies," in *EDBT Workshops*, 2006.

[32] J. Terwilliger, L. Delcambre, and J. Logan, "Context-sensitive clinical data integration," in *EDBT Workshops*, 2006.

[33] G. W. G. for Infective Endocarditis, S. Skiadopoulos, and C. Tryfonopoulos, "Greek multi-centre epidemiological study on infective endocarditis: Initial results," in *In Infections*, 2013.