

**NAME: ARTWELL MAMVURA**

**TITLE: National Health Practitioner Informatics System**

**YEAR: 2013**

### **Abstract**

ICT tools are continuously changing business processes while the Zimbabwe's public health sector still lags behind. The commercial industry has completely evolved because of e-Commerce business models but an e-Health strategy has not yet been implemented in the health sector. The need for a tool to track the areas in the country with critical shortages of public health care professionals and ease manipulation of health practitioner's data stirred the development of the National Health Practitioner Informatics System (NHPIS). The NHPIS system is a health informatics system that allows ease querying, updating, deleting, inserting of health practitioners' data. The term current system was used to refer to the existing system.

The primary purpose of this study is to document the problems of the current system, objectives of NHPIS application, the development procedure and steps, and maintenance of the NHPIS system. This documentation is phased in five main chapters, which include Introduction, Planning, Analysis, Design and Implementation Phases. The introduction phase is an introduction of the research study and a historical background of the organization. The planning phase details the scheduling of the study and a feasibility study which was conducted to evaluate viability of the system. From the feasibility study, the project proved to be feasible.

During the analysis phase, the researcher analyzed the weaknesses and strengths of the current or existing system and evaluated whether to develop, improve or outsource the software. The researcher had the relevant skills needed to develop the system. The researcher gathered user

requirements using questionnaires, interviews and observations. After analysis phase, the design phase followed, in this phase the researcher designed the database using MySQL Database management system, input forms, menus and output forms using diagrams to illustrate the major functionalities of the NHPIS. Security designs were done at this stage

The last phase was implementation, during this phase the researcher converted the pseudo code into Java source code using Netbeans IDE. The researcher validated, verified and tested the system. The system was deployed using the parallel system conversion method. System maintenance is an ongoing process. Better versions of the NHPIS will be developed continually.

**Declaration**

I Mamvura Artwell do hereby declare that I am the sole author of this dissertation; I authorize the Midlands State University to lend this dissertation to other institutions or individuals for the purpose of scholarly research.

**Signature**.....

**Date** .....

## **Approval**

This dissertation entitled National Health Practitioner Informatics System by Mamvura Artwell meets the regulations governing the award of the degree of Computer Science Honours of the Midlands State University, and was approved for its contribution to knowledge and literary presentation.

**Supervisor**..... **Date** .....

## **Acknowledgements**

Firstly, I would like to thank the almighty God for the source of strength that he has always given me throughout my research. I would like to express my profound acknowledgement to the following people: The project supervisor Mr T G Rebanowako who contributed immensely throughout the entire project development process, the management and staff at Hitrac and MoHCW for allowing me to carry out my research activities. I would also like to extend my sincere gratitude to my friends Regis Dzumbunu, Faith Mazuzu, Mornisher Mihloro and Simbarashe Makwangudze for their support during the research study.

Lastly, I would like to thank my father Mr Mamvura and my sisters Alice and Aleta for their love and financial support. I could have never done it without them. I can safely say, "May God continue to bless you".

## **Dedication**

“This dissertation is dedicated to my late mother, Irene Sithole who is my major source of inspiration. Her loving memories always inspire me to be the best and to bring out the best in me. The greatest lesson that she taught me is that burdens becomes light when ideas to bear them are lighter and brighter.”

## **Table of Contents**

<b>Abstract</b>	<b>I</b>
<b>Declaration</b>	<b>II</b>
<b>Approval</b>	<b>III</b>
<b>Acknowledgements</b>	<b>IV</b>
<b>Dedication</b>	<b>V</b>
<b>Table of contents</b>	<b>VI</b>
<b>List of acronyms</b>	<b>X</b>
<b>List of tables</b>	<b>XI</b>
<b>List of figures</b>	<b>XII</b>
<b>List of appendices</b>	<b>XIV</b>
<b>CHAPTER ONE: INTRODUCTION</b>	<b>1</b>
<b>1.1 Introduction</b>	<b>1</b>
<b>1.2 Background information</b>	<b>1</b>
1.2.1 Background of the study	1
1.2.2 Background of UZ College of Health Sciences	1
1.2.3 Background of HITRAC	2

1.2.4 Organizational Structure of HITRAC	2
<b>1.3 Problem definition</b>	<b>4</b>
1.3.1 Description of the current system	4
1.3.2 Problems of the current system	4
<b>1.4 Objectives</b>	<b>5</b>
<b>1.5 Aim of the study</b>	<b>5</b>
<b>1.6 Hypothesis</b>	<b>5</b>
<b>1.7 Justification</b>	<b>7</b>
<b>1.8 Conclusion</b>	<b>7</b>
<b>CHAPTER TWO: PLANNING PHASE</b>	<b>8</b>
<b>2.1 Introduction</b>	<b>8</b>
<b>2.2 Reasons for building the system</b>	<b>9</b>
<b>2.3 Business value</b>	<b>10</b>
<b>2.4 Feasibility study</b>	<b>10</b>
2.4.1 Purpose of the feasibility study	11
2.4.2 The benefits of completing a feasibility study	11
2.4.3 Social feasibility	11
2.4.4 Technical feasibility	12
2.4.5 Economic Feasibility	14
2.4.6 Operational Feasibility	19
<b>2.5 Risk Analysis</b>	<b>19</b>
2.5.1 Technical risks	19
2.5.2 Risks related to Software Engineering Process	20
2.5.3 Tight Schedule	20
2.5.4 Risks associated with development & Testing Tools	21
2.5.5 Risks related to the developmental Environment	21
<b>2.6 Work Plan</b>	<b>21</b>
2.6.1 Project Activities	22
2.6.2 Project Time Allocation or Schedule	22
2.6.3 Work Breakdown Structure	22
2.6.4 Gantt Chart	23
<b>2.7 Conclusion</b>	<b>24</b>
<b>CHAPTER THREE: ANALYSIS PHASE</b>	<b>25</b>

<b>3.1 Introduction</b>	<b>25</b>
<b>3.2 Information gathering methodologies</b>	<b>26</b>
3.2.1 Interviews	27
3.2.2 Questionnaire	29
3.2.3 Observation	30
<b>3.3 Analysis of existing system</b>	<b>31</b>
<b>3.4 Process analysis</b>	<b>31</b>
3.4.1 Activity Diagram of current system	32
<b>3.5 Data Analysis</b>	<b>32</b>
3.5.1 Context Diagram of the current system	33
<b>3.6 Weaknesses of current system</b>	<b>35</b>
<b>3.7 Evaluation of Alternatives</b>	<b>35</b>
3.7.1 Outsourcing	35
3.7.2 Improvement	36
3.7.3 In House Development	36
<b>3.8 Requirements Analysis</b>	<b>37</b>
3.8.1 Functional Requirements	37
3.8.2 Non-functional requirements	40
<b>3.9 Conclusion</b>	<b>41</b>
<b>CHAPTER FOUR: DESIGN PHASE</b>	<b>42</b>
<b>4.1 Introduction</b>	<b>42</b>
<b>4.2 System Design</b>	<b>42</b>
4.2.1 The new system	43
4.2.2 Context Diagram of the Proposed System	45
4.2.3 DFD of the new system	46
4.2.4 Logic Flow Chart of the new system	47
<b>4.3 Architectural design</b>	<b>48</b>
4.3.1 Architecture Design of the NHPIS Software	48
<b>4.4 Physical design</b>	<b>51</b>
<b>4.5 Database design</b>	<b>52</b>
4.5.1 Conceptual database design	53
4.5.2 Logical database design	56
4.5.3 Physical Database Design	62



<b>4.6 Program design</b>	<b>63</b>
4.6.1 Package Diagram	63
<b>4.6.2 Class Diagram of the new system</b>	<b>64</b>
<b>4.6.3 Sequence Diagram</b>	<b>66</b>
<b>4.7 Interface design</b>	<b>67</b>
4.7.1 Menu Design	67
4.7.2 Input Design	69
4.7.3 Output User Interface	74
<b>4.8 Security Design</b>	<b>75</b>
4.8.1 Web application design issues	75
<b>4.9 Conclusion</b>	<b>76</b>
<b>CHAPTER FIVE: IMPLEMENTATION PHASE</b>	<b>77</b>
<b>5.1 Introduction</b>	<b>77</b>
<b>5.2 Coding</b>	<b>78</b>
5.2.1 Pseudo Code	78
<b>5.3 Testing</b>	<b>80</b>
5.3.1 Methods of Testing	80
5.3.2 Levels in Testing	81
5.3.5 Verification and Validation	88
<b>5.4 Installation</b>	<b>91</b>
5.4.1 Steps in software installation process	91
5.4.2 User training	92
5.4.3 System Conversion	94
<b>5.5 Maintenance</b>	<b>95</b>
5.5.1. Corrective maintenance	96
5.5.2 Adaptive maintenance	97
5.5.3 Perfective maintenance	97
5.5.4 Preventive maintenance	97
5.5.5 Managing maintenance	98
5.5.6 The system maintenance life cycle	98
5.5.7 Prioritization	99
5.5.8 Fire fighting	99
5.5.9 Standards and quality assurance	99
<b>5.6 Conclusion</b>	<b>100</b>

<b>REFERENCES</b>	<b>101</b>
<b>Appendix A: User Manual</b>	<b>104</b>
<b>Appendix B: Interview Guide</b>	<b>117</b>
<b>Appendix C: Questionnaire Checklist</b>	<b>118</b>
<b>Appendix D: Observation Score Sheet</b>	<b>122</b>
<b>Appendix E: Snippet of Code</b>	<b>123</b>

### List of Acronyms

<b>ANSI-SPARC</b>	The American National Standards Institute Standards Planning and Requirements Committee
<b>CBA</b>	Cost-Benefit Analysis
<b>CDC</b>	Centre for Disease Control and Prevention.
<b>CEPHI</b>	Center for Evaluation of Public Health Interventions.
<b>CSS</b>	Cascading style sheets
<b>DBMS</b>	Database Management System
<b>DCM</b>	Department of Community Medicine.
<b>DNS</b>	Directorate of Nursing Science.
<b>GUI</b>	Graphical User Interface
<b>HIO</b>	Health Information Officer
<b>HITRAC</b>	Health Informatics Training and Research Advancement Centre.
<b>HIU</b>	Health Informatics Unit.
<b>ICT</b>	Information and Communication Technology.
<b>JSP</b>	Java Server Pages
<b>MEPI</b>	Minority Engineering Program of Indianapolis
<b>MoHCW</b>	Ministry of Health and Child Welfare.
<b>MVC</b>	Model View and Control
<b>NHPIS</b>	National Health Practitioner Informatics System
<b>NPV</b>	Net present value

<b>PEPFAR</b>	The United of States President's Emergency Preparedness For Aids Relief.
<b>RDBMS</b>	Relational Database Management System
<b>SDLC</b>	System Development Life Cycle
<b>SQL</b>	Structured Query Language
<b>UZCHS</b>	University of Zimbabwe College of Health Science.

## List of Tables

Table 2.1 Database Server	13
Table 2.2 Client Computer Specifications	13
Table 2.3 Network Specifications	13
Table 2.4 Software specifications	14
Table 2.5 Development costs	17
Table 2.6 Operational costs	17
Table 2.7 Tangible benefits	17
Table 2.8 Cost benefit analysis	18
Table 2.9 Work Breakdown Structure	23
Table 4.1 Hardware and software specified	51
Table 4.2 Country Table	57
Table 4.3 User Table	57
Table 4.4 User_role Table	57
Table 4.5 Privilege Table	58
Table 4.6 Role Table	58
Table 4.7 Qualification Table	59
Table 4.8 Title Table	59
Table 4.9 Address Type Table	60
Table 4.10 Practitioner Table	60
Table 4.11 Person Table	61

Table 5.1 HITRAC Administrators training	93
Table 5.2 Provincial Administrators	93

## List of Figures

Fig 1.1 Organizational structure for HITRAC	3
Fig 2.1 Project plan Diagram	9
Fig 2.2 Waterfall diagram	23
Fig 2.3 Gantt chart	29
Fig 3.1 Activity Diagram of the current System	32
Fig 3.2 Context Diagram of current System	33
Fig 3.3 Data Flow Diagram of current System	34
Fig 3.4 Use case diagram of the current system	39
Fig 4.1 Context Diagram for the Proposed NHPIS system	45
Fig 4.2 Data flow diagram of the proposed System	46
Fig 4.3 Logical Flow chart for the proposed NHPIS	47
Fig 4.4 Three-Tier Architecture	48
Fig 4.5 N-tier	49
Fig 4.6 Java EE platform Architecture	49
Fig 4.7 Model View and Controller	50
Fig 4.8 Layered multi-tier Java EE application architecture based on MVC	50
Fig 4.9: Physical design of the proposed system	52
Fig 4.10 Entity Relationship Diagram	54
Fig 4.11 Enhanced Entity Relationship Diagram	55
Fig 4.12 ANSI-SPARC Database design	62

Fig 4.13 Package Diagram	64
Fig 4.14 Class Diagram of the proposed system	65
Fig 4.15 Sequence diagram of the proposed system	66
Fig 4.16 Main Menu Form	67
Fig 4.17 Static Data download Form	68
Fig 4.18 Data Export Form	68
Fig 4.19 Data Import Form	68
Fig 4.20 Login Form	69
Fig 4.21 Add Static Data form	69
Fig 4.22 Search Practitioner Form	69
Fig 4.23 Add or Edit Static data Form	70
Fig 4.24 Create Practitioner step 1 of 5 Form	70
Fig 4.25 Create Practitioner step 2 of 5 Form	71
Fig 4.26 Create Practitioner step 3 of 5 Form	72
Fig 4.27 Create Practitioner step 4 of 5 Form	72
Fig 4.28 Confirm Details Create Practitioner step 5 of 5 Form	73
Fig 4.29 Advanced Practitioner Search Form	73
Fig 4.30 Reports generated	75
Fig 4.31 Security Design Issues	75
Fig 5.1 Audit trail on database	87
Fig 5.2 Add Practitioner qualification	87
Fig 5.3 Checks for duplication	87
Fig 5.4 Validated Login form	89
Fig 5.5 Null Field Validation	89
Fig 5.6 National ID and Passport Number Validation	90
Fig 5.7 EC number validation	90
Fig 5.8 Parallel Conversion	95
Fig 5.9 System Maintenance Life Cycle	99

## **List of Appendices**

Appendix A	User Manual
Appendix B	Interview Guide
Appendix C	Questionnaire checklist
Appendix D	Observation Score Sheet
Appendix E	Snippet of code



# **CHAPTER ONE: INTRODUCTION**

## **1.1 Introduction**

This chapter documents all the objectives and tools the researcher used to successfully develop and implement the National Health Practitioner Informatics System (NHPIS). NHPIS system was developed for the budget preparation division of the Ministry of Health and Child Welfare (MoHCW) and HITRAC's administration. The NHPIS supports the delivery of health care by providing the information required for measuring the performance of health practitioners at each health facility in the country.

## **1.2 Background information**

This is the background information of the NHPIS system, the background information of the organization and the background of the study.

### **1.2.1 Background of the study**

Health care is a field in which accurate record keeping and communication are critical, yet the use of computing and networking technologies lags behind as compared to other fields for example the commercial industry where there is e-Commerce. E-health involves the use of information and communication technology (ICT) in the health sector to store, manipulate, and disseminate patient and practitioner data. Before the implementation of the NHPIS application health practitioners' information was transmitted through paper files. Health informatics or e-Health is a new phenomenon in the health sector.

### **1.2.2 Background of UZ College of Health Sciences**

The UZ College of Health Sciences (UZCHS) was established in 1963 under the auspices of the University of Birmingham, U.K. The College currently offers degree programmes in Medicine, Dentistry, Pharmacy, Nursing Science, Medical Laboratory Sciences, Rehabilitation, Radiology and Health Education & Promotion. UZCHS is committed to the training of Health Care Delivery personnel to meet the health needs of the people of Zimbabwe. Training is geared



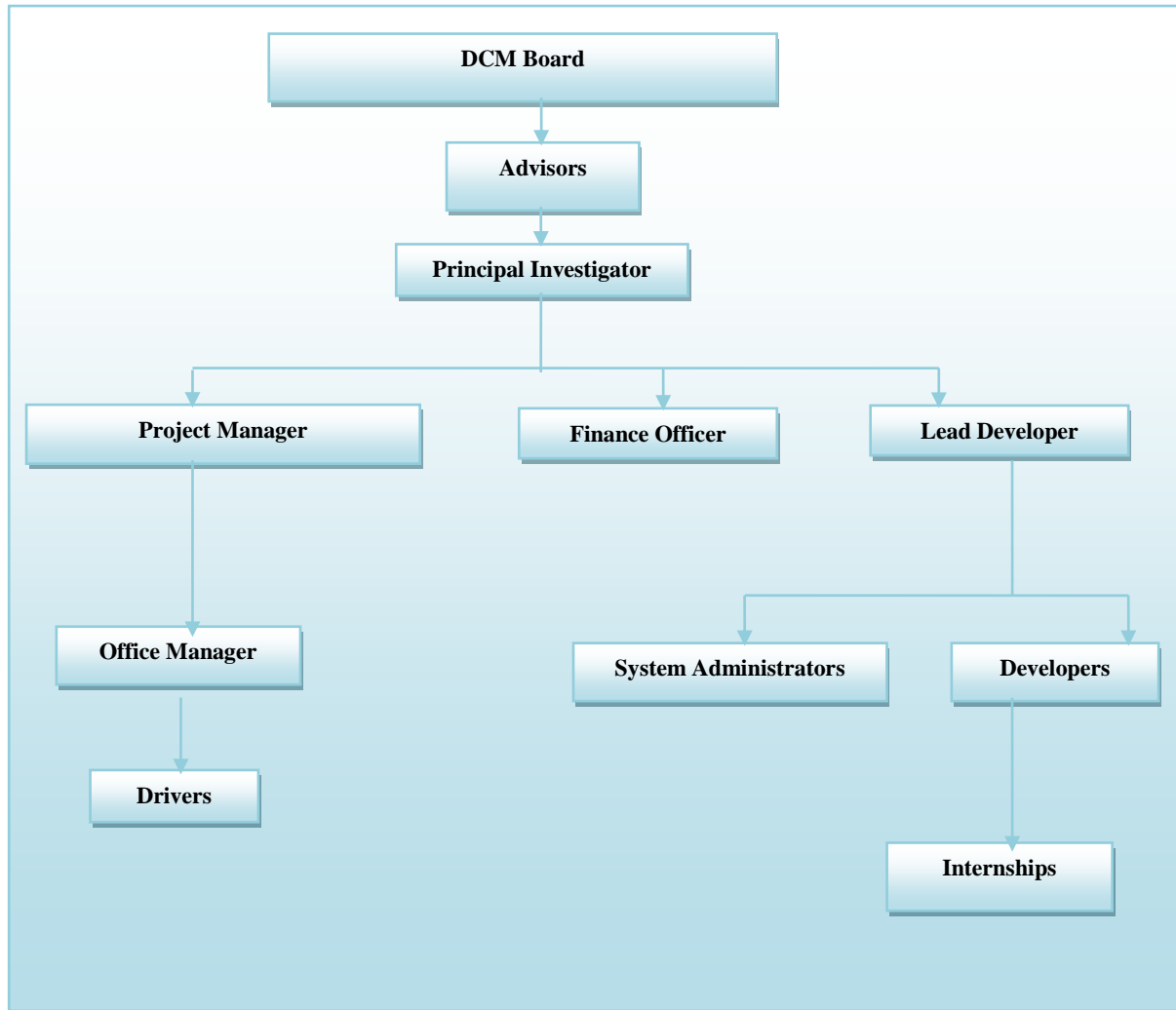
towards primary health care with emphasis on a preventive rather than curative approach. As an institute of higher learning, it takes academic excellence as the hallmark of its existence. UZCHS is at Parirenyatwa Group of Hospitals' teaching wing.

### **1.2.3 Background of HITRAC**

HITRAC organization was established in 2000 as a health informatics unit under the Centre for Evaluation of Public Health Interventions (CEPHI) a division of the UZCHS Department of Community Medicine (DCM). It is a leading research centre, and health software development organization in Zimbabwe. HITRAC is funded by the United States President's Emergency Plan for Aids Relief (PEPFAR) through CDC (Centre for Disease Control and Prevention) in collaboration with Minority Engineering Program of Indianapolis (MEPI). The organization develops the software applications in Java programming language and uses MySQL database management system.

### **1.2.4 Organizational Structure of HITRAC**

The department of community medicine (DCM) board directly controls HITRAC. The board is mainly responsible for defining of the organizational strategic goals. The Advisor is responsible for the sourcing out of funds and reports to the DCM board. The Principal Investigator reports to the Advisor. The Principal Investigator who is also the Domain Expert is responsible for the day-to-day operations of the organization. The project manager, finance officer and lead researcher all report to the principal investigator. The project manager is responsible for managing the project scope, deadlines, quality and quantity control. The finance officer is responsible for the purchasing of office equipment, staff and project team needs. The Lead researcher is the key personnel in projects undertaken by the organization. He or she is the one responsible for directing the project team. The Lead researcher is responsible for outlining software development tools and policies. The office manager reports to the project manager and assigns drivers duty. The project team consists of systems administrator who manages the networks, hardware and software infrastructure and the developers who writes the source code files. Student interns reports to the developers.



**Fig 1.1 Organizational structure for HITRAC**

#### **1.2.4.1 Mission Statement**

Developing competent, self motivated and innovative staffs that are always abreast with the vast changing information and communication technologies

#### **1.2.4.2 Vision**

HITRAC aims to be an internationally recognized centre of excellence in health informatics.

### **1.3 Problem definition**

A problem is defined as the deviation of the existing state from the desired state, or a deviation from a norm, standard, or status quo (Bennatan, 1995). A problem definition is the process of identifying the perceived gap between the existing state and desired state. The researcher thoroughly identified the right problems from the beginning as it was important for the overall success of the project.

#### **1.3.1 Description of the current system**

The data capturers manually registers a practitioner at each station in all the provinces, the practitioner data files are then stored in cabinets, and in some cases a deceased practitioner continue being included in payroll. Anyone with physical access to the files can add, update, delete or modify data and still be unaccountable to his or her actions. Each province has different naming standards. A lot of paper work was involved when re-grading, adding practitioner disciplinary case and appraising a practitioner.

#### **1.3.2 Problems of the current system**

The current system involves a lot of paper work and time-consuming processes. The other problems of the current system are:

- Lack of immediate retrieval of a practitioner's data - retrieving practitioner's data involves sifting through large volumes of files. The process of retrieving practitioners' details is tiresome and cumbersome.
- Lack of prompt updating of practitioners' details - changes in one file is manually updated in all files. This promotes loss of data integrity
- Unintentional duplication of data – There is need for additional storage space
- Limited data sharing - A requested report requires data from several incompatible files in separate files, and inaccurate reports are frequently prepared.
- Data insecurity – there was unauthorized access to the files containing practitioner data.

The problem definition was successfully completed and the next step was formulation of the objectives of the system. NHPIS is a solution to the identified problems.

## **1.4 Objectives**

An objective is a specific, measurable result that a system aims to achieve within a period and with the available resources (Stewart, 1987). Objectives are basic tools that underlie all planning and strategic activities. They serve as the basis for the creating of policies and evaluation of performance. The development NHPIS system allows the collection of practitioner data within the country. The data repository supports different data manipulations for national health budgets. The main objectives of the informatics application are:

- To enable synchronization of practitioner data manipulation processes at the provincial and national level.
- To promote accountability and responsibility on every act, event, use or modification made to the data in the system.
- To enable flexible and accurate report generation.
- To develop a system that uniquely identifies a practitioner.
- To track and monitor continuous professional development of the health practitioners.
- To develop a system that enforces same naming standards for reporting consistency.
- To establish connectivity, functional integration and interoperability of the databases at provincial level with the national instance.
- To enable automatic synchronization of a provincial data manipulation with the national instance database state.

## **1.5 Aim of the study**

To develop an innovative tool and approach for allowing capturing and manipulation of health care professionals' data and dissemination of such information to the right people in right time in this electronic era.

## **1.6 Hypothesis**

Hypothesis is a justification of the tools used for development or stating why those tools are appropriate for development of a system. Usually for software projects, using the tools a researcher is familiar with reduces the risk of project failure. There is no time for the researcher

to start learning new technologies. The listed software tools were used to develop, write the source code, deploy and to host of the NHPIS system:

- Netbeans 7.2.1 IDE - Netbeans 7.2.1 Integrated Development Environment will be used for writing source code and JUnit testing of the source code or software
- Maven - A build and project management tool with automated integration unit tests and allows loose coupling and modular programming.
- Apache Tomcat Server 7.0.27 - Apache Tomcat Server is a web server for hosting Java Web application or standalone deployment. It is open source software.
- MySQL - MySQL is one of the top databases available in the market. MySQL is a relational database with many advanced features. MySQL is an Open Source RDBMS and anyone can use it freely. Developers can customize the source code to suit their requirements. It has a faster development time, can be installed easily and is operable on different platforms including. MySQL is secure since all access passwords are stored in an encrypted format restricting any unauthorized access to the system and MySQL clients can access this relational database through standard TCP/IP sockets, named pipes, UNIX sockets and many more.
- Java is a write once run many times anywhere and object oriented programming language with fast development time, hence java programming language is suitable for development of the NHPIS software.
- Hibernate Framework - an object relationship mapping tool for MySQL database queries
- JavaScript for client side validation and graphical user interface enhancements,
- Velocity is a front-end templating tool
- Hyper text mark-up language, Java Server Pages (JSP) for end user views i.e. Graphical User Interface (GUI) and Cascading style sheets (CSS) is a language that is used for display and representation manipulations on the of the NHPIS
- Spring MVC Framework for dependency injection and inversion of control,
- Spring Security for privileges and roles based login.
- Firebug for CSS manipulation
- IReport for report generation
- RestEasy Webservices and Client UI for system integration

## **1.7 Justification**

The development of the practitioner data repository enhances the country's health care professional planning, that is the government can determine the health workforce needs within the country and be able to strategically plan towards strengthening of the work force to better the nation's growing health needs. In as much as practitioners needs their patients' demographic data, health information and treatment history the patients also may need to query the data of their specialists to see if they meet the government and state regulatory quality measure before they can be treated. If the proposed NHPIS is deployed the MoHCW easily queries the ever changing practitioner data with the country.

The rationale for the study is:

- With the NHPIS, there is support for provincial strategic planning and quarterly monitoring.
- Ongoing feedback and follow-up to provinces on data quality
- Ongoing training and skills development (of data collectors, information officers, managers) at national, provincial, district and hospital levels
- Can be used as an standalone application with the forms and reports
- Easy to learn and deploy
- Requires reasonably few resources to operate
- Compatible with RDBMS
- It is easy to maintain

The researcher included audit trails in the system hence any change in the database records is traced back to the modifier or creator.

## **1.8 Conclusion**

National Health Practitioner Informatics System is a vital component in giving the MoHCW practitioner data or information and tools to help them with their decision-making processes. The introduction of NHPIS improves the quality of health practitioners' data captured at all levels. Having completed the preliminary stage the researcher proceeded to the planning phase. The Introduction phase was one week in duration.

## CHAPTER TWO: PLANNING PHASE

### 2.1 Introduction

The planning phase highlights the preliminary analysis; it brings out the general overview of the NHPIS. The planning phase is the second phase in the software project life cycle. Planning phase is the time when the project team translates the initial scope from the envisioning phase into practical plans on how to achieve it, (Davis C and Alan M, 1995). Often project planning is ignored in favour of getting on with the work. However, project managers fail to realize the value of a project plan in saving time, money and many risks. The planning phase was carried out in a systematic way using the stepwise approach to project approach (Bennatan, 1995).

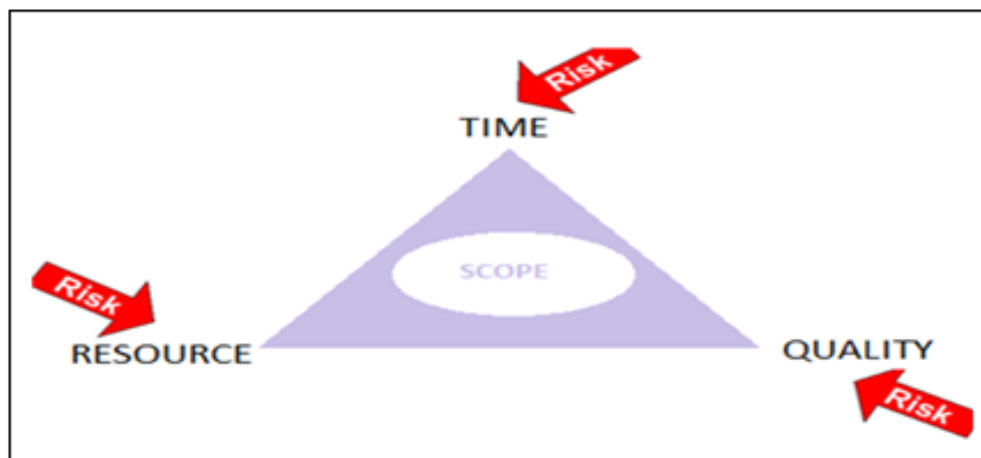
The planning approach involved the following:

- The establishment of project objectives and the analysis of characteristics of a project
- The establishment of an infrastructure consisting of an appropriate organization and set of standards, methods and tools
- The identification of the products of the project and the activities needed to generate those products
- The allocation of resources to activities and the establishment of quality controls

A project plan has five key characteristics, and these include:

- Scope: defines what the project covers.
- Resource: human and financial resources and instruments used to develop the system.
- Time: what project tasks are undertaken and when.
- Quality: the spread or deviation allowed from a desired standard.
- Risk: defines in advance, what may happen to drive the plan off course, and how to recover from the situation.

The diagram below illustrates the five key characteristics of project plan



**Fig 2.1 Project plan Diagram**

## **2.2 Reasons for building the system**

NHPIS system is a means by which the nation can possibly manage the exponential growth of practitioner health data, information, and knowledge. The following factors are reasons why the system was built:

- Administrative information used in the health sector changes on a daily basis hence the need for a tool that allows easy and multiple data manipulations
- Health care organizations track regulations and guidelines from different data sources.
- Access to current reliable knowledge is a key determinant in the behavior and performance of health care professionals.
- The NHPIS enables sharing of data, information, and knowledge across domains on a right-to-know and need-to-know basis.
- There is need for a basic technological platform to ensure that the benefits of ICT diffuse to all health care organizations.



### **2.3 Business value**

According to McGraw (2004), business value refers to the success of the organization in using information to achieve its strategic objectives. This is the part of the corporate strategy and comprises the development and declaration of shared view of business' direction and the benefits of the NHPIS from a business perspective. The system is of great value to the country in the following ways:

- Improved support for provincial strategic planning and quarterly reporting.
- Ongoing feedback and follow-up to provinces on health care professionals' data quality
- Continuous training and skills development for users at national, provincial, district and health care service delivery facilities.
- Increased authorized access to data hence ensuring data confidentiality and control as well as keeping up-to-date information.
- Reduced transaction costs and time: The automation of the manual processes involved reduces the transaction time and cost involved.
- Improved data quality and compatible formats of the data from different external sources

### **2.4 Feasibility study**

According to Clifton D.S and Fyffe (1977), the aims of a feasibility study are to find out whether the system is worth implementing given the existing budget and schedule. A feasibility study is an evaluation and analysis of the proposed project, based on extensive investigation and research to give full comfort to the decisions made. The inputs of a feasibility study are a set of business requirements, a description of the system and an outline of how the system supports the business processes. The results of the feasibility study should be a report that recommends whether or not it is worth carrying on with the requirements engineering and system development process.

Carrying out a feasibility study involves information assessment, information collection and report writing. Some examples of possible questions that might be asked include:

- How would the organization cope if the new system was not implemented?
- What are the problems of the current system and how is the new system supposed to alleviate these problems?

### **2.4.1 Purpose of the feasibility study**

- Establishing whether the project is operationally, technically, socially and economically feasible.
- To establish whether the benefits outweigh the costs.
- To determine if there is schedule feasibility.
- To identify the strengths and the drawbacks of the existing system.
- Gathering the requirements specifications.

### **2.4.2 The benefits of completing a feasibility study**

- Informed decision-making - To manage stakeholder expectations about how much the project costs and how long it takes to implement the system.
- Reduces costs - Concluding that a project is not feasible is not a bad outcome, as it avoids wasting resources on a project that would later fail.
- It records what is known about the project: During the feasibility study, report the analyst gathered significant amounts of valuable information on the project.
- Increases chances of the project being a success - Feasibility study identifies the hard parts of the project. It help in identification of dependencies in the planning phase i.e. trials that need to be done or questions that need to be answered before key decisions on eradication design can be made. This gives more time for the analyst to address all of the issues before the operation starts.

### **2.4.3 Social feasibility**

Social feasibility entails how the proposed system affects the stakeholders in the organization and whether it receives the full support of the management (Clifton D.S, Fyffe, 1977). It also investigates whether the system is going to risk the jobs of the employees or increase the labour turnover. Generally, the questions asked are:

- What changes will the proposed system bring?
- What organizational structures are disturbed?
- Do the existing staff members have these skills, if not; can they be trained?

The NHPIS system positively affects the duties of management and improved quality and flow of information since the success of management largely depends on the disposal of information. The successful deployment of the NHPIS promises to boost the morale of staff members and funders. System training was conducted and the users understood how the system works. The responses obtained from the stakeholders were:

- MoHCW: The MoHCW supported the development of the NHPIS.
- Sponsors (CDC): Ascertained to constantly fund the development of the system.
- Users (Administrator at Hitrac): The system proved to be a solution for the administrators as a tool for consistent naming and data sets definition within the whole system
- Users (HIO): Support the development of the software; they want a system with reduced effort processes in communication and transport.
- HITRAC management: supported the development of the system and promised to allow their subordinates to stress test the system before it is deployed.

#### **2.4.3.1 Social feasibility conclusion**

Analysis of the responses from stakeholders shows that the application is socially feasible.

#### **2.4.4 Technical feasibility**

This is concerned with specifying equipment and software that satisfy the user requirements (Clifton D.S, Fyffe, 1977). The technical needs of a system vary considerably, but might include:

- Facility to produce outputs in a given time.
- Response time under certain conditions.
- Ability to process a certain volume of transaction at a particular speed.
- Facility to communicate data to distant locations.

In examining technical feasibility, configuration of the system is more important than the actual make of the hardware. The configuration should give the complete picture of the system's requirements, for example, how many workstations are required, how these units are to be interconnected so that they could operate and communicate smoothly. Technical assessment of a

proposed system consists of evaluating the required functionality against the hardware, software and technical skills (human resources) available.

#### 2.4.4.1 Technical Expertise

According to McGraw and Hoglund (2004), technical feasibility answers the question as to whether the system can be built given the apparent constraints in terms of resources and time. The researcher of the system possesses the relevant skills required to develop the NHPIS.

#### 2.4.4.2 Hardware and Software Specifications

For the successful implementation of the system, the following hardware and software specifications was added.

**Table 2.1 Database Server**

Item	Minimum	Recommended	Available
Hard disk drive	500GB	3T	80GB
Printer	Inkjet	Laser	Laser

**Table 2.2 Client Computer Specifications**

Processor	800MHz	1.8GHz Intel P4	800 MHz
Memory	1G	2G	512MB
HDD	50 GIG	400 GIG	80 GIG
Network Cards	10/100	10/100	None

**Table 2.3 Network Specifications**

HUB	32 Port	32 Port	16 Port
Patch Panel	32 Port	32 port	6 port
Connecting Cables	UTP CAT 35 Fly leads patch codes	UTP CAT 35 Fly leads patch codes	UTP CAT 35 Fly leads patch codes
UPS	3u	3u	3u

### 2.4.4.3 Software Specification

Most of the software required to develop this project are open source packages that is there is no cost of licensing incurred. To effectively implement the software product the softwares in the table below are needed:

**Table 2.4 Software specifications**

Microsoft Windows/ Linux	7
Adobe PDF Reader	Latest Version
Firefox Browser	Latest Version
Antivirus	Norton or Avira
Microsoft Excel and Word	From 2003

### 2.4.4.4 Conclusion on Technical Feasibility

The project was technically feasible taking into account the availability and affordability of the additional hardware, software and expertise required to develop the project. This study also looked at the availability of the technical expertise needed for developing, supporting and maintaining the system and it was proved that the researcher had the technical skills needed to undertake the project.

### 2.4.5 Economic Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system (Clifton D.S, Fyffe, 1977). The objective of economic feasibility study was to determine the benefits and savings expected from the NHPIS system and compare them with costs. If benefits outweigh costs, a decision to design and implement the system is considered. Otherwise, further justification or alternative in the proposed system has to be made if it is to have a chance of being approved. This outgoing effort improves in accuracy at each phase of the system life cycle.

#### **2.4.5.1 Cost Benefit Analysis (CBA)**

Cost-benefit analysis (CBA) is an analytical tool used to assess the benefits and costs of regulatory proposals, (McGraw, 2004). The Costs and benefits were examined from the management's perspective to identify the system proposal had the highest net benefit. The standard way of evaluating the economic benefits of any project is to carry out a Cost-Benefit Analysis, which consists of two steps:

- Identifying and estimating all of the costs and benefits of carrying out a project - This includes development costs of the system, the operating costs and the benefits that are expected to accrue from the operation of the system.
- Expressing these costs and benefits in common units - Evaluating the net benefit i.e. the difference between the total benefits and the total cost.

The analyst expressed each cost and benefit in monetary terms to establish whether the project was feasible. The researcher used ROI to determine the economical feasibility of NHPIS.

#### **2.4.5.2 Costs**

These are costs related to the system and include both development operational costs incurred during the development process as well as costs of running and maintaining the proposed system, (McGraw, 2004). Most costs are relatively easy to identify and quantify in approximate monetary terms. Costs are classified according to where they originate in the life cycle of the project. Embarking on the project should not strain the company's budget and should yield benefits that can justify its costs and implementation. Costs are categorized as either development or operational costs. Development costs: these are the costs incurred during the development process only, are estimated at the onset of a project, and should be refined at the end of each project phase. The development costs include:

- Human Resource Costs
- User training Costs
- Equipment and setup costs

Operational Costs: These are costs incurred during the development of the NHPIS. Operational costs are classified either as variable or fixed.

- Fixed Costs - occur at regular intervals, at relatively fixed rates.
- Variable costs - occur in proportional to a usage factor. For example supplies such as that of CDs, DVDs, and USB flash drives e.t.c.

### **2.4.5.3 Benefits**

The benefits that come along with the introduction of the system were categorized as either strategic or tactical benefits

- Strategic Benefits: are those benefits that help the MoHCW to perform better usually at a lower cost. They are usually long-term benefits and are regarded mainly as intangible since they are not clear for anyone to see.
- Tactical Benefits: these are long-term benefits realized by the top management, which help the organization to improve usually socially or the working environment henceforth affecting the production of the organization.

These benefits are either tangible or intangible. Tangible and Intangible benefits - according to Jones (1998), Tangibility refers to the ease with which costs or benefits can be measured. An outlay of cash for any specific item or activity is referred to as a tangible cost. These costs are known and can be estimated quite accurately. Benefits are often more difficult to specify exactly than cost. Tangible benefits such as completing jobs in fewer hours or producing error free reports are quantifiable. Intangible benefits such as more satisfied customers or an improved corporate image because of using new system are not easily quantified. Both tangible and intangible costs and benefits should be taken into consideration in the evaluation process.

Some benefits will be realized by the company after implementation of the new system and these include the following:

#### **2.4.5.3.1 Tangible benefits of NHPIS**

- Operations efficiency
- Increased automation of processes
- Reduced operations

#### 2.4.5.3.2 Intangible Benefits of NHPIS

- Improved asset utilization
- Time saving
- Increased practitioner morale

**Table 2.5 Development costs**

Description	Amount (USD)
DELL Power Edge R410	2500
HP Compaq 500B x4	2500
Total	5000

**Table 2.6 Operational costs**

Description	Amount (USD)
System maintenance/year	2000
User training	2500
Stationary and other computer consumables	500
Total	5000

**Table 2.7 Tangible benefits**

Description	Amount (USD)
Reduced operation costs	1000
Improved Data Quality	1000
Consistent Reports and Time saving	1000
Increased Practitioner Morale	1000
Better Health Outcomes	1000
Total	5000



**Table 2.8 Cost benefit analysis**

Cost and Benefits	Estimated Value (USD)	Total (USD)
Tangible Benefits	5000	
Intangible benefits	7000	
Total benefits		12000
Development costs	5000	
Operational costs	5000	
Total costs		(10000)
Net Benefits		2000

#### **2.4.5.4 Return on Investment (ROI)**

According to Kendell K.E and Kendell J.E, (2002), return on investment technique is used to compare the net profit against the investment required. ROI was used to calculate the viability of the project. It is the widely used cost-benefit analysis technique and is calculated using the percentage of profitability. ROI is a performance measure used to evaluate the efficiency of an investment or to compare the efficiency of a number of different investments. To calculate ROI, the benefit (return) of an investment is divided by the cost of the investment; the result is expressed as a percentage or a ratio. ROI was expressed as a percentage and was calculated as follows:

$$\begin{aligned} \text{Returns on Investments (ROI)} &= \frac{\text{Net Benefits X 100}}{\text{Total Costs}} \\ &= \frac{2000 \times 100\%}{10000} \\ &= 20\% \end{aligned}$$

#### **2.4.5.5 Conclusion on economic feasibility**

Taking into accounting the net benefits as well as the favorable return on investment (20%) the system is economically feasible hence; the company can embark on the project.

## **2.4.6 Operational Feasibility**

Operational feasibility is a measure of how well a new system solves the problems, and takes advantage of the opportunities identified during scope definition. The operational feasibility assessment focuses on the degree to which the proposed development projects fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture, and existing business processes

### **2.4.5.5 Conclusion on operational feasibility**

The NHPIS system is operationally feasible, it produces consistent reports, allows automatic synchronization of practitioners' details. There is ease manipulation of practitioner details

## **2.5 Risk Analysis**

Risk analysis is a technique employed to identify and assess various factors, which may jeopardize the success of a project or achieving a goal (Bennatan, 1995). These factors can pose some sort of threat to the project. Thus, risk analysis covers the process of scientific assessment of such threats vulnerable to the attainment of the organizational goals. Risk analysis technique is helpful in defining preventive measures to reduce the probability occurrence of such threatening factors. It includes identification of various countermeasures to successfully deal with such constraints with an objective to avoid devastating effects on the organization's competitiveness in the trade.

### **2.5.1 Technical risks**

A product coming out of the hands of personnel of lower skill levels shall be certainly a cause of risk to the organization. Following checklist shall be helpful in bridging the gaps in this area.

- Deployment of personnel having best possible skills appropriate to the project
- When in a team, proper combination of various personnel with different temperament and skill levels is important.

- Availability of the nominated personnel during the complete duration of the project is of key importance. The project would be affected if the researcher who is key personnel had left or fallen ill during system development.

The researcher has competent skills for developing a more secure web application.

### **2.5.2 Risks related to Software Engineering Process**

A clear-cut definition of the entire process of software engineering is of paramount importance for the success of the product. A roughly planned process results in a software product that poses great threats to itself as well as to the organization. The following guidelines or checklist can be helpful in identifying the software engineering related threats & planning their counter measures.

- The researcher ensured the availability of a documented process planned for the development of the software product.
- Ensure that all the participants of the product development team (whether in-house or third party peoples) is religiously following the documented process
- Ensure the availability of a mechanism for monitoring the activities and performance of third party developers and testers.
- Ensure the proper documentation of outcome of the technical reviews detailing the resources deployed to unearth what type of software bugs.
- Ensure the availability of a configuration management mechanism for ensuring adequate consistency in design, development and testing of the product in line with the basic requirements already defined.

### **2.5.3 Tight Schedule**

A period of two months to implement the NHPIS was scheduled. This could have resulted in an undesired pressure on the researcher making him produce a sub-standard product, and end-up not implementing all the desired functionalities in an effort to meet the deadline. To avoid this risk, the researcher remained focused on the work at hand to avoid deadline pressure. The researcher also worked overtime to be able to deliver a quality product within the specified period. The researcher modularized the development of the system to allow loose coupling.

#### **2.5.4 Risks associated with development & Testing Tools**

Different types of development and testing tools can also be a cause of concern many a times during the SDLC. To prevent such risks the researcher used fast development languages, which include:

- Java - This is a write once and run in any platform.
- Maven - This is both a build tool and project management tool and performs automatics JUnit testing on each module of the system.

The researcher is familiar with Netbeans IDE.

#### **2.5.5 Risks related to the developmental Environment**

Environment provided for development of the product also plays a key role in the success of the product. Some of the factors or situations described below can pose certain amount of risk.

- Availability of suitable testing tools compatible with the product being developed.
- Compatibility of the databases with the environment under which they are deployed.
- Compatibility or proper integration of all software tools with each other

#### **2.6 Work Plan**

A work plan is an outline of all tasks that need to be complete in order to finish an entire project. A work plan includes management's layout for each member in the team and the tasks that each individual will be performing. It includes:

- A list of personnel participating in the project
- A list of all equipment and facilities to be used in the project
- A breakdown of the project into specific tasks, with indications of which tasks are dependent upon the completion of others
- A schedule indicating when each activity or task start and its activity duration, and who perform it; this information may be represented as an annotated bar chart

### **2.6.1 Project Activities**

Project activity is any kind of work that is focused on accomplishing a project. The less resources (time, budget, labour) are spent to support these activities without defecting from quality and project scope, the more effective are the project plan and management. There are two essential tools used to manage project activities: project activity list and project activity schedule. Project activity list is an instrument that is used to describe sequence of tasks, successive operations or steps that are necessary to complete the project.

It can be represented as a plain to do list or as a task tree where all work components are grouped and shown in interrelated manner. Project activity reports are usually created as a task list showing which tasks are completed; in progress, etc. This corresponds to a brief description of activities to be performed during project implementation. The project was broken down into work break down structures to make planning for the researcher easier. The Waterfall Model was used in the development of the NHPIS. Project activity schedule is a type of work plan that shows tasks and project phases with respect to their time parameters. Such a schedule can be represented on a calendar-like project activity template with different scale of the time grid.

### **2.6.2 Project Time Allocation or Schedule**

Project time management is important in any project. Project time allocation involves the allocation of duration to activities or tasks of the project. The whole project was subdivided into Waterfall phases and each step was an activity of the NHPIS system. Time was allocated to individual activities of the project.

### **2.6.3 Work Breakdown Structure**

A work breakdown structure in project management and systems engineering, is a deliverable oriented decomposition of a project into smaller components. It defines and groups a project's discrete work elements in a way that helps organize and define the total work scope of the project.

**Table 2.9 Work Breakdown Structure**

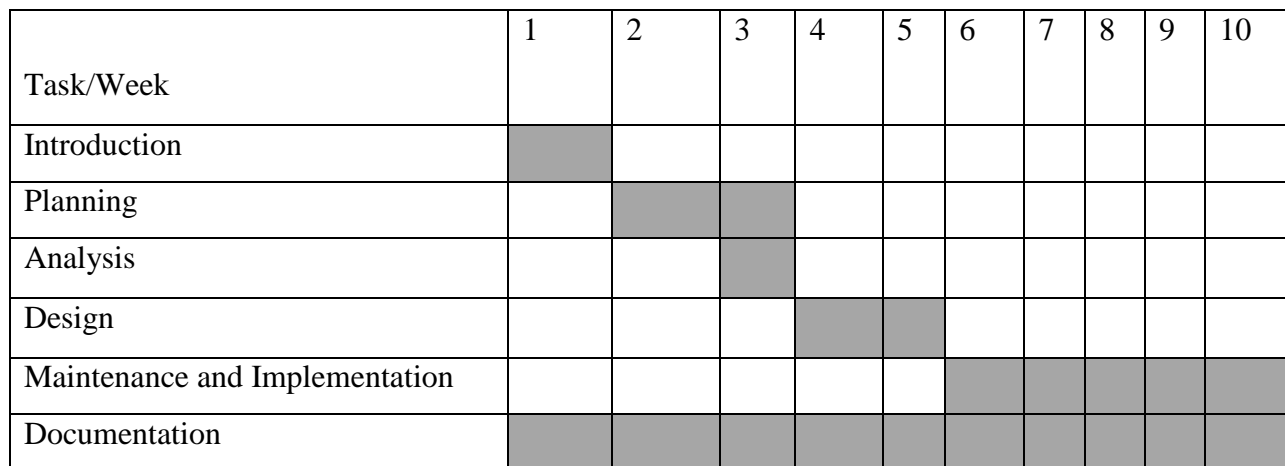
Activity	Start Date	End Date	Duration (Weeks)
Introduction	18/02/13	25/02/13	1
Planning	25/02/13	11/03/13	2
Analysis	04/03/13	11/03/13	1
Design	11/03/13	25/03/13	2
Maintenance	25/03/13	29/04/13	5
Documentation	18/02/13	29/04/13	10

**2.6.4 Gantt Chart**

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. A bar represents each activity; the position and length of the bar reflects the start date, duration and end date of the activity.

The Gantt chart below helped the researcher to identify:

- The activities and activity duration, that is when each activity begins and ends
- How long each activity is scheduled to last
- Where activities overlap with other activities, and by how much
- The start and end date of the whole project



**Fig 2.3 Gantt chart**

## **2.7 Conclusion**

The project was approved, individual tasks established, scheduled, a work plan laid down, and the researcher did the analysis of the current system and all other activities. During the planning phase, a resource allocation schedule was developed that stated the allocation of financial and human resources to individual activities. The planning phase was two weeks in duration; the analysis phase started in the beginning of the second week of planning phase. The design phase started after the completion of the planning and analysis phase. The Maintenance phase started after the completion of the design phase. At each phase, the researcher properly documented the entire project task he was carrying out.

## **CHAPTER THREE: ANALYSIS PHASE**

### **3.1 Introduction**

According to Rajaraman (2004) systems analysis is a process of collecting factual data, understanding the processes involved, identifying problems and recommending feasible suggestions for improving the functioning and performance of the system. It involves studying the business processes, gathering operational data, understand the information flow, finding out bottlenecks and evolving solutions for overcoming the weaknesses of the system to achieve the organizational goals. System analysis also includes subdivision of complex process involving the entire system, identification of data store and manual processes.

Systems analysis is an iterative process that continues until a preferred and acceptable solution emerges. There are always gaps between the user and the systems analyst. The user understands the current system, but the analyst does not. The analyst understands the new system, but the user does not. How well the analyst and user work together to bridge these gaps, determined the success the new system. During the information gathering, the analyst communicated with the users frequently to understand the current system. The researcher or systems analyst came up with functional and non-functional requirements. Requirement analysis is discussed in several sections of this documentation through the discussion of data-flow diagrams, data dictionary, entity-relationship diagrams, and process description.

In the analysis phase, our objective is to:

- Determine and document how the current system works.
- Determine how the system could work better.
- Develop a logical or business model of the new system.

The major output of the systems analysis phase is a systems proposal that describes the findings of the analysis. The researcher came up with a logical model of the new system from the analysis stage.



### **3.2 Information gathering methodologies**

According to Tait and Vessey (1988), system analysis involves gathering information about the present system. The proper use of tools for gathering information is the key for successful analysis. The feasibility report in the systems planning phase did not contain details of the systems requirement. Thus, a full-blown study for the requirement of the system was necessary in order to understand the detail operations of the business. To determine system requirements, the analyst sought information of the current system and the opportunities for improvements in the following areas:

#### **System Objectives**

- Identify the objectives of the new system
- Evaluate these objectives

#### **System Inputs and Outputs:**

- Identify the inputs and outputs of the new system
- The origin of the inputs and the destinations of the outputs

#### **System Functions:**

- Define the functions of the new system
- Identify the components of the systems: manual procedures, user interfaces, computer programs, files and databases
- Identify timings of input, output and processing
- Identify controls on data entry, security, and processing

The analyst used the information-gathering techniques listed below:

- Interviews
- Questionnaires
- Onsite Visits or Observation

The system analyst used the interviews, questionnaire, and observations, as information gathering techniques to collect the required information.

### **3.2.1 Interviews**

The interview is the most common method of information gathering. There are several basic steps to the interview process:

- Preparation for the interview
- Design of interview questions
- Conduct the interview
- Document the Interview and perform a follow-up interview.

#### **3.2.1.1 Preparation for the Interview**

To plan for the interview, the systems analyst reviewed the documents from existing system. The systems analyst examined:

- The company's goals and objectives
- Forms, reports, and business models of the current system
- Organization chart and user roles.

Referring to the organization chart the analyst prepared a list of interviewees who would provide various levels of information for the current system and future needs of the system. Higher levels of management normally provide an overview of the current system and its future needs. The lower-level users provided the detailed operations of the system. Thus, the researcher interviewed provincial data capturers, as they were the user of the current system and an employee of the MoHCW.

#### **3.2.1.2 Designing Interview Questions**

There are three types of interview questions and these are:

Closed-ended questions: Closed-ended questions enable analysts to control the interview and obtain information they need. Answers to the questions require only a short answer of one or two words (true or false, multiple choice, rating on a scale, or ranking).

Open-ended questions: Open-ended questions are those that leave rooms for further elaboration by the interviewee. These questions provide additional information or problems that a user normally does not like to talk about.

Probes: These are follow-up questions in response to one of the above questions, when the analyst is unclear about the answer.

### **Refer to Appendix B**

#### **3.2.1.3 Conducting the Interview**

There are certain guidelines for a successful interview. Some of them are:

- Listen very carefully to the interviewee and give opportunity to answer and ask questions.
- Take a second person to take notes, if permitted by the organization, record the interview.
- Ask questions, even if they sound “dumb”. Not asking questions may result in wrong conclusion causing further potential problems
- Separate facts from individual user opinions. Facts are important but opinions are not.
- Do not make any premature promise on any part of the delivery of the system.
- Thank the interviewee at the end and mention that a follow-up interview may occur to clarify questions that may arise.

#### **3.2.1.4 Document the Interview**

After the interview, the analyst organized and typed interview notes in the form of a report. The researcher listed unclear or additional questions that arose during the interview. The report was sent to the interviewee with a request to read, make comments, and his or her approval.

#### **3.2.1.5 Follow-Up Interview**

Follow-up interview arises due to the questions that may arise in writing the interview report and obtaining a response of the report. Depending on the number and type of questions, the follow-up interview can be performed in person or over the telephone or any other suitable mechanism.

### **3.2.1.5 Findings from the Interviews**

From the interview, the researcher was able to find out that the users of the current system were not comfortable with the current system. The users spent a lot of time when performing a task in the current system. The users were ready for the new NHPIS software. Interviews are a good tool to collect detailed information. It allows exploration and follow-up questions. Interviews build rapport between the users and the systems analyst however, interviews are time-extensive and expensive.

### **3.2.2 Questionnaire**

If a project requires input from a large number of people, the questionnaire can be a valuable tool to determine system requirements. A questionnaire is a document containing a number of standard questions. The researcher used questionnaires to obtain information about workloads, reports received and volumes of transactions handled

#### **3.2.2.1 Choosing Respondents to Questionnaire**

It is important to select the right group of people to send the questionnaire. The group should be a representative sample of all users of the system. The researcher selected randomly users to give the questionnaires.

#### **3.2.2.2 Designing a Questionnaire**

When designing a questionnaire, the most important rule is to make sure that the questions collect the right data. Questions asked should be brief and user-friendly and there should be clear instructions on how to answer the questions. The questions asked should relate to the requirements of the system and should be in logical order. Some of the points to take into consideration when designing questionnaires include:

- Use of simple wording to avoid misunderstanding
- Avoiding leading questions
- avoiding open-ending questions are difficult to tabulate
- limit questions raising concern or negative issues

**Refer to Appendix C**

### **3.2.2.3 Findings from the questionnaire**

The researcher noted that questionnaires were most useful when used for specific purposes rather than for more general information gathering. The researcher discovered that questionnaires can be given to many people at a time, whereas interviews can be performed on a single person at a time. Questionnaires are most useful for closed-end questions, although some open-ended questions can be included for information gathering, are less expensive and less time-consuming and can be performed on paper, over the telephone and electronically. However, questionnaires are a rigidly structured means to obtain answers to pre-selected inquiries. From the answers of the questionnaires, the users were ready for the new system. The current system had tiresome processes. However, the researcher noted that when using questionnaires respondents could leave some questions unanswered and in some cases, users gave inappropriate answers. Questionnaires are ideal as they promote anonymous answers

### **3.2.3 Observation**

The observation of current operating procedures of the system is another fact-finding technique. A system can be understood better and faster through observation. The main objective of an onsite visit is to get as close to the real system as possible. The researcher is a keen observer knew the processes in the current system and the normal activities that occur within the system.

#### **Refer to Appendix D**

The researcher considered the **Hawthorne Effect**: During the observation day, people may work more efficiently than the normal day. Operations may also run less smoothly because people might be nervous during observation.

### **3.2.3 Findings from the Observation**

During an onsite visit at MoHCW for observation the analyst would ask questions to obtain a good understanding of the system operation procedure. The analyst observed all steps in the processing cycle and noted the output from each procedural step, that is examine each file, record, and report. The analyst successfully determined the purpose of each item on the documents. The analyst was able to observe a complete a process of the current system, and the

MoHCW employee who receive reports to see whether the reports were complete, timely, accurate, and in a useful form. The processes in the current system were time consuming and often times the workers often left the processes hanging. The users were not comfortable with the current system. The data capturers often made cancellations and dirty records difficult to read.

### **3.3 Analysis of existing system**

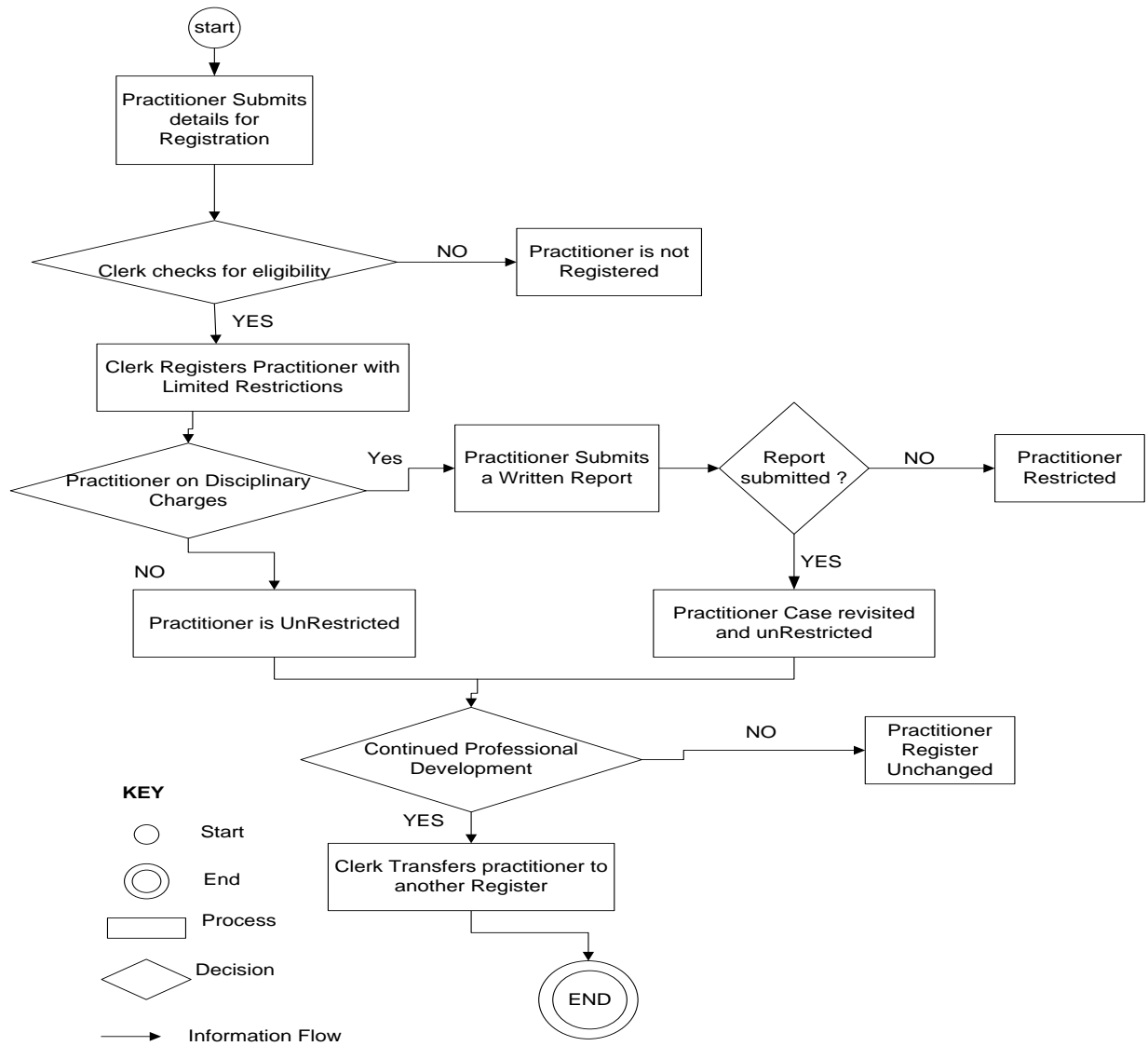
With the current system operating, a practitioner's data existed in two different operating systems in different file format. There was duplication of health practitioner's data. There was need for integration of the operating systems capturing or recording the practitioners' data in order to facilitate easy national report creation. The total number of health practitioners in the public health sectors was unrecorded.

- The data capturers manually registers a practitioner at each station in all the provinces, the practitioner data files are then stored in cabinets, and in some cases a deceased practitioner continue being included in payroll.
- Anyone with physical access to the files can add, update, delete or modify data and still be unaccountable to his or her actions.
- Each province has different naming standards.
- A lot of paper work was involved when re-grading, adding practitioner disciplinary case and appraising a practitioner.

### **3.4 Process analysis**

A step-by-step breakdown of the phases of a process, used to convey the inputs, outputs, and operations that take place during each phase. A process analysis can be used to improve understanding of how the process operates and to determine potential targets for process improvement through removing waste and increasing efficiency. An activity diagram illustrates how the processes of existing system are executed. It starts from the time a process is initiated up to the time a process is complete and all the involved decisions during the execution of a process. A clear process analysis of the existing process helps the researcher understand the existing system more.

### 3.4.1 Activity Diagram of current system

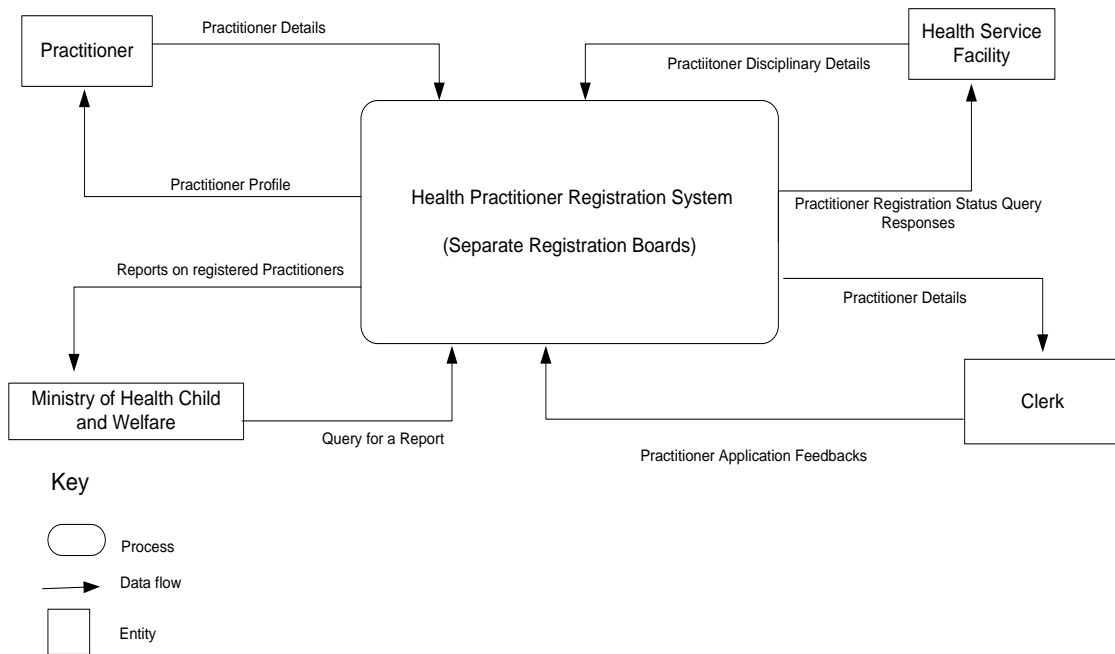


**Fig 3.1 Activity diagram of the current system**

### 3.5 Data Analysis

The process model is a formal way of representing how a business system operates. It illustrates the processes and activities that are performed and how data moves among them

### 3.5.1 Context Diagram of the current system



**Fig 3.2 Context Diagram of the current System**

### 3.5.2 Data Flow Diagram

According to Wiegers and Karl E (2003) a data flow diagram is a pictorial representation of the flow of information from the beginning of a transaction to the end. A transaction in system development can be for example in the current system, the registration of a practitioner. A data flow diagram helps the analyst understand the processes and flow of information within the current system. Clarification of such processes aids the analyst when developing the functions or modules of the proposed system. As each process is analyzed, loopholes of the current system are identified and rectified.



### Data flow diagram of the current system

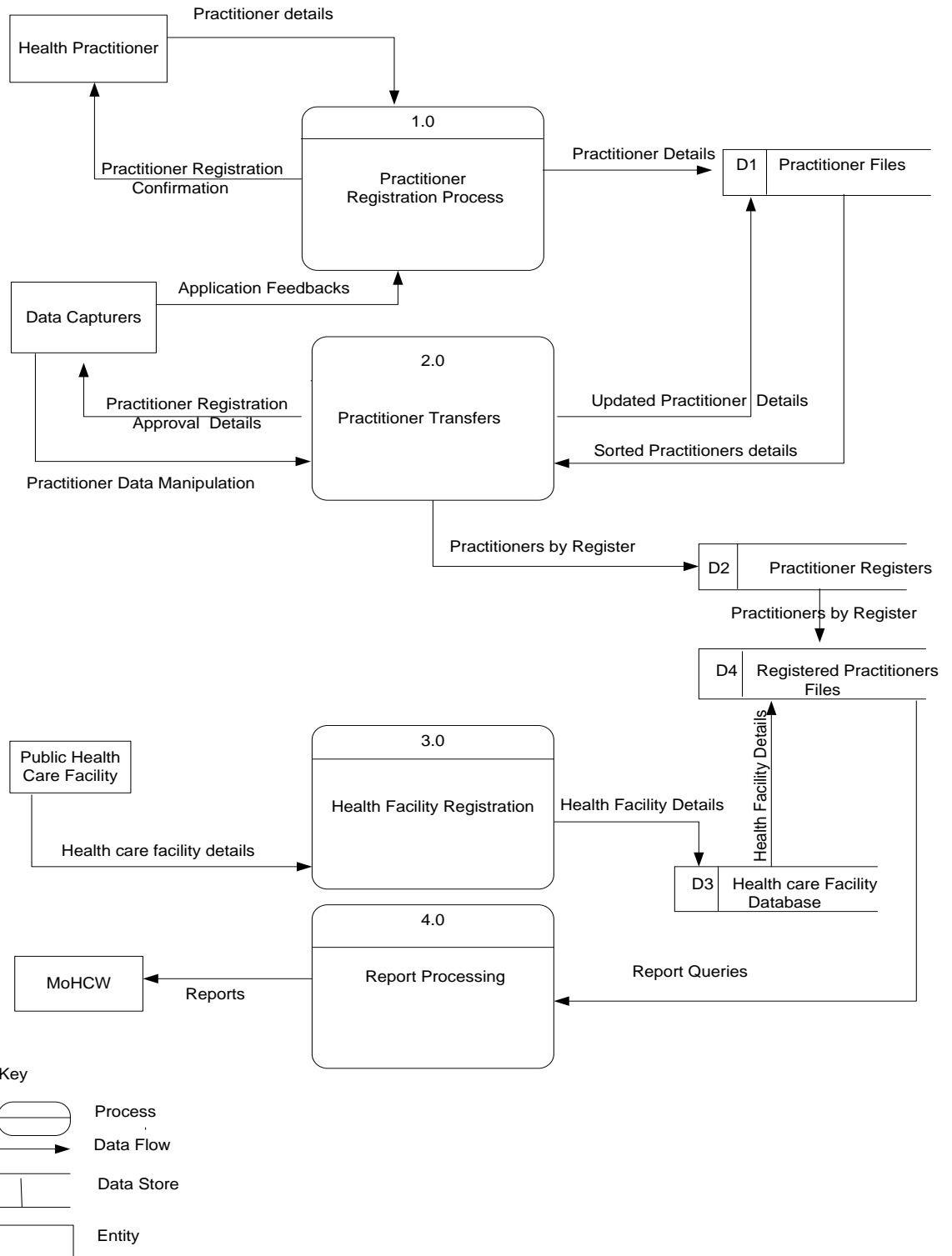


Fig 3.3 Data Flow Diagram of current System

### **3.6 Weaknesses of current system**

- Data duplication: the same data gets repeated over and over since the data capturers find it hard to keep track of the documents, information and transactions.
- Lack of security: Since data is stored in filing cabinets, it is freely available to anyone. If information falls into the wrong hands it can be used against the practitioners
- Common errors: when entering data customers might have accidentally switched details and data since it is hand written.
- Inconsistency of data: Current data is prone to unavailability for future use, since data might get misplaced during manual filing.
- Repetition of work: if there are any changes to be made, the data have to be entered again. At times the data capturer would forget to make the changes or forget that they had already altered it and might redo it again hence time consuming.
- Too much paper work: since everything and every detail is written in papers and filed
- Slow retrieval of data: the information of practitioners and details were stored in different sites and so it takes a long time to retrieve the data. It takes a long time to find the information about a relevant person

### **3.7 Evaluation of Alternatives**

With reference from the feasibility study that was undertaken in the planning phase, it was concluded that the project is viable. Evaluation of the alternatives assists us to choose the best alternative that yields optimal results. Various alternatives exist which are outlined below:

- Outsourcing.
- Improvement of the current system.
- In-House Development.

#### **3.7.1 Outsourcing**

The outsourcing alternative entails buying a software package off the shelf, or having the software package developed by an external service provider (Software Development Houses). From the feasibility study the company had the required technical resources needed for in house development. The hardware and software required was available. Though outsourcing has got it

own merits, for example it reduces the development and implementation time; the following are the reasons as to why HITRAC should not outsource:

- There is need for external support of hardware and software.
- There is need for specialists to install and operate.
- May not be easy to integrate with existing manual system and the package is expensive.
- Off-the-shelf software package is most likely not to meet the organization's standards.
- The software may not leave room for customizing the system to suit the needs of the organization.

### **3.7.2 Improvement**

From the feasibility, study carried out improving the current system is a relatively cheaper alternative in term of development and operational costs; it has the following demerits:

- Does not meet global standards therefore the firm may lose some of their customers.
- Time is wasted redesigning the new system as it involves frequent interviews with the stakeholders.
- High operating costs due to extra hired labor because of redesign.

Based on the outlined it is not the best alternative.

### **3.7.3 In House Development**

The firm can create a system that is specifically for HITRAC. The system to be developed should be able to curb all the problems currently being faced by the current system. The researcher has the technical expertise required for development of the system. The following outlines the reasons why the NHPIS system was developed in-house:

- It meets the organization's long-term goals and is less costly to implement.
- The system has room for expansion in respect with the changing technology and small tasks can be integrated in the system thereby reducing work force.
- The system meets international technology standards and efficient execution of tasks.

- Maximizes utilization of resources and it is easier for users to produce reports, so delivery of tasks becomes more efficient.
- Users of the system will assume ownership of the system since their requirements are incorporated during system development.
- There will be continual testing of the system, as the development of the system is now in house.

### **3.8 Requirements Analysis**

According to Hoffer, George and Valacich (2006) requirements analysis is critical to the success of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design. Conceptually, requirements analysis includes three types of activities:

- Eliciting requirements - the task of identifying the various types of requirements from various sources including project documentation
- Analyzing requirements - determine whether the stated requirements are clear, complete, consistent and unambiguous, and resolve any apparent conflicts.
- Recording requirements: requirements are documented in various forms, usually including a summary list and may include natural-language documents, use cases, user stories, or process specifications.

Requirements analysis can be a long process during which delicate psychological skills are involved. New systems change the environment and relationships between people, so it is important to identify all the stakeholders, take into account all their needs and ensure they understand the implications of the new systems.

#### **3.8.1 Functional Requirements**

According to Mohammad, R. (2006) functional requirements explain what has to be done by identifying the necessary task, action or activity that must be accomplished. Functional requirements analysis was used as top-level functions for functional analysis. Functional

requirements define a set of features i.e. inputs, processes, outputs and stored data needed to satisfy the system objectives. In this case, the system's functional requirements are:

**Online Registration of Practitioner:** The HIO registers a practitioner and there is an automatic update of the database at national. When a practitioner is registered, his or her details are automatically be pushed to the national database.

**Automatic downloads of static data:** to ensure data consistency within the whole system, static data e.g. the administrator at the national instance should capture Registration status. The administrator at provincial level should then be able to download that static data.

**Uploading and downloading of practitioner details:** if an instance is offline from the national server, practitioner information is downloaded into USB and CDs, forwarded to the national instance, and uploaded.

**Independent processing:** Each different station or instance should operate or register practitioners independently.

**Integration of the System:** The NHPIS system is integrated and allows no duplication of data.

**Use of UUID:** The system should incorporate use of UUID to uniquely identify a practitioner.

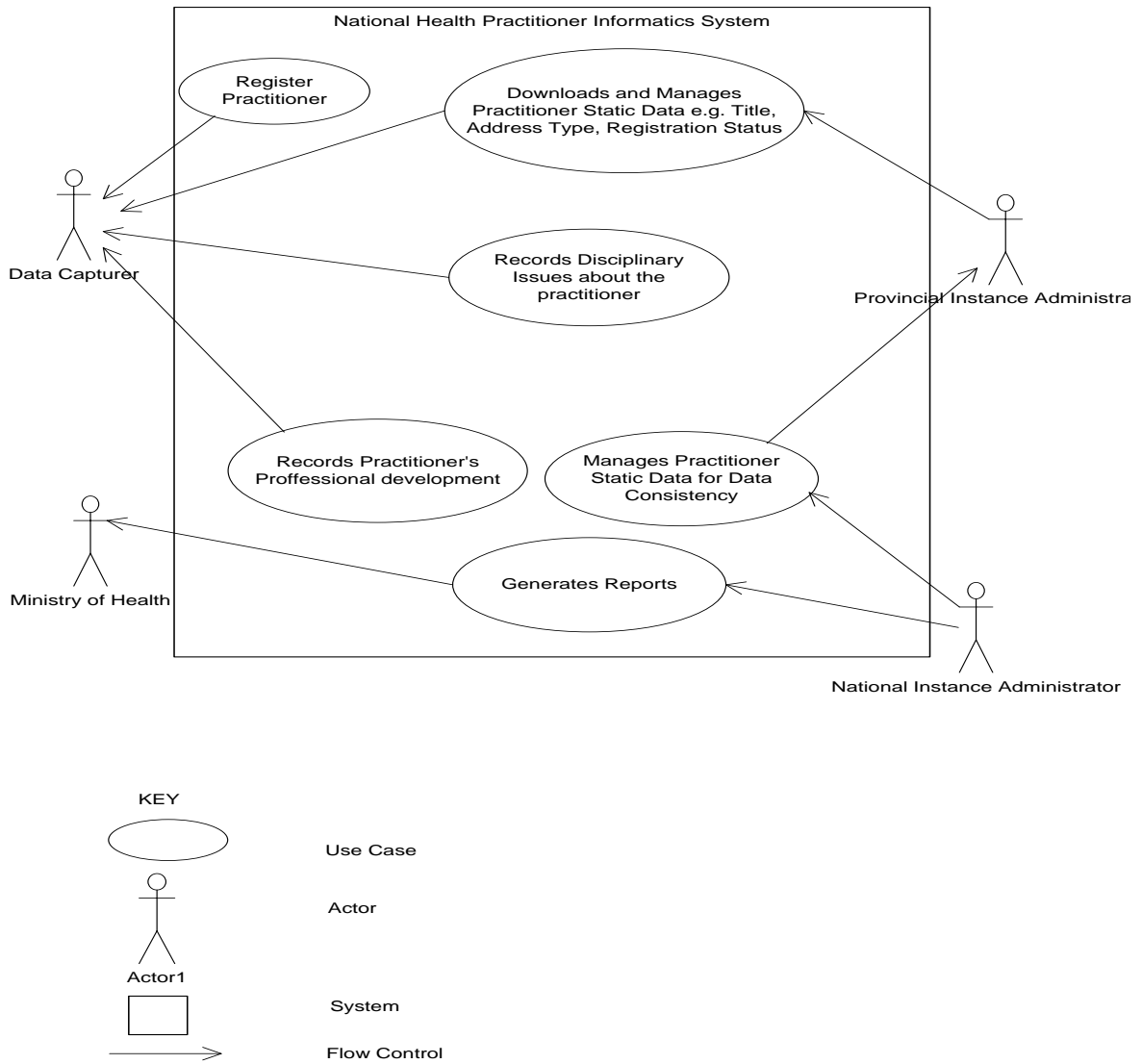
**Report Generation:** Several reports that are helpful to the Ministry of Health and Child Welfare are produced for easy budgeting, recruiting and allocation of practitioners to health facilities and managerial decision-making.

## Use Case Diagrams

A use case is a technique for documenting the potential requirements of a new system or software change. Each use case provides one or more scenarios that convey how the system should interact with the end user or another system to achieve a specific business goal. Use cases typically avoid technical jargon, preferring instead the language of the end user or domain expert. Use cases are deceptively simple tools for describing the behavior of software or systems. A use case contains a textual description of all of the ways, which the intended users could work with the software or system. Use cases do not describe any internal workings of the system, nor do they explain how that system will be implemented. They simply show the steps that a user

follows to perform a task. All the ways that users interact with a system can be described in this manner.

### 3.8.1.1 Use Case Diagram of proposed system



**Fig 3.4 Use case diagram of the NHPIS system**

### 3.8.2 Non-functional requirements

Non-functional requirements are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional system requirements include:

- **Availability:** A system's availability, or "uptime," is the amount of time that it is operational and available for use. This is specified because the system will be designed with expected downtime for activities like database upgrades and backups.
- **Efficiency:** Specifies how well the software utilizes scarce resources: CPU cycles, disk space, memory, bandwidth, etc.
- **Flexibility:** If the organization intends to increase or extend the functionality of the software after it is deployed, that should be planned from the beginning; it influences choices made during the design, development, testing, and deployment of the system.
- **Portability:** Portability specifies the ease with which the software can be installed on all necessary platforms, and the platforms on which it is expected to run.
- **Integrity:** Integrity requirements define the security attributes of the system, restricting access to features or data to certain users and protecting the privacy of data entered into the software.
- **Performance:** The performance constraints specify the timing characteristics of the software. Certain tasks or features are more time-sensitive than others are; the nonfunctional requirements should identify those software functions that have constraints on their performance.
- **Reliability:** Reliability specifies the capability of the software to maintain its performance over time. Unreliable software fails frequently, and certain tasks are more sensitive to failure (for example, because they cannot be restarted, or because they must be run at a certain time).
- **Reusability:** Many systems are developed with the ability to leverage common components across multiple products. Reusability indicates the extent to which software components is designed in such a way that they can be used in applications other than the ones for which they were initially developed.

- **Robustness:** A robust system is able to handle error conditions gracefully, without failure. This includes a tolerance of invalid data, software defects, and unexpected operating conditions.
- **Scalability:** Software that is scalable has the ability to handle a wide variety of system configuration sizes. The nonfunctional requirements should specify the ways in which the system is expected to scale up (by increasing hardware capacity, adding machines, etc.).
- **Usability:** Ease-of-use requirements address the factors that constitute the capacity of the software to be understood, learned, and used by its intended users.

Generally, non-functional requirements impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

### **3.9 Conclusion**

The analysis of the current system was done thoroughly and all the alternatives were considered. It was concluded that developing a unique software package is the ideal solution for the problem and as a result, all the functional and non-functional requirements have been identified. There is therefore the need to move over to the design stage of the NHPIS system that is going to be covered in the next chapter.

The researcher realized that the development of the system was easier as the user requirements were clearly defined.



## **CHAPTER FOUR: DESIGN PHASE**

### **4.1 Introduction**

According to Conger (1994) design phase is the art of designing system components and interrelationships between those components in the best possible way to solve some well-specified problem. The aim of the design phase is to map the functional requirements of the application to the software and hardware environment. The results of the design phase are programming specifications and plans for testing, conversion, training and installation of the system.

The design stage transformed the detailed requirements of the analysis stage into a complete, detailed specification of the system. The analyses of this stage are performed within the framework of the system concept, converting the functional and data requirements of the definition stage into a complete system design, which will guide the work of the development stage. During the design stage, a plan is build, of how the project was developed through the rest of the SDLC process from implementation, to verification, to the time when the system was implemented. During the design phase best practices to follow were established that is functional and design specifications, and risk analysis was performed to identify threats and vulnerabilities in the software.

### **4.2 System Design**

According to Schach (1999), system design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. The purpose of system design is to create a technical solution that satisfies the functional requirements for the system. The functional specification produced during system requirements analysis is transformed into a physical architecture. System components are distributed across the physical architecture, usable interfaces are designed and prototyped and technical specifications are created for the application researcher thus enabling him to build and test the system.

The design phase consists of the following processes:

- Preparation for system design, where the existing project repositories are expanded to accommodate the design work products, the technical environment and tools needed to support System Design are established, and training needs of the team members involved in system design are addressed.
- Define technical architecture, where the foundation and structure of the system are identified in terms of system hardware, system software, and supporting tools, and the strategy is developed for distribution of the various system components across the architecture.
- Define system standards, where common processes, techniques, tools, and conventions that were used throughout the project are identified in an attempt to maximize efficiencies and introduce uniformity throughout the system.
- Create physical database, where the actual database to be used by the system is defined, validated, and optimized to ensure the completeness, accuracy, and reliability of the data.
- Prototype system components, where various components of the solution may be developed or demonstrated in an attempt to validate preliminary functionality, to better illustrate and confirm the proposed solution, or to demonstrate “proof-of-concept.”
- Produce technical specifications, where the operational requirements of the system are translated into a series of technical design specifications for all components of the system, setting the stage for system construction.

#### **4.2.1 The new system**

In line with continued efforts to strengthen the NHPIS, posts for data capturers or Health Information Officers (HIO) were created at provincial, district and station levels. The HIO will be using the system to update and record every public health medical practitioners’ details. The system should allow instant updating of the national practitioner repository as soon as the HIO or data capturers registers or updates a practitioner’s data. In case of a station, being offline from the national server daily-encrypted registrations transactions the system should support exportation of the encrypted practitioner details. The exported files are transmitted in USBs or DVDs to the administrator at HITRAC who in turn will import the encrypted practitioner details to the database.

There is consistency in the data captured at various stations, districts or provinces. The administrator at the national level captures the static data and the data capturers or HIO will only be downloading the static data from the national repository hence consistent naming standards.

Reports for the MoHCW are generated from the system.

**Login:** The system authenticates and authorizes only the users in the system, hence the need of server side validation. The system should also expire sessions if the user becomes idle and require re-logging of the user.

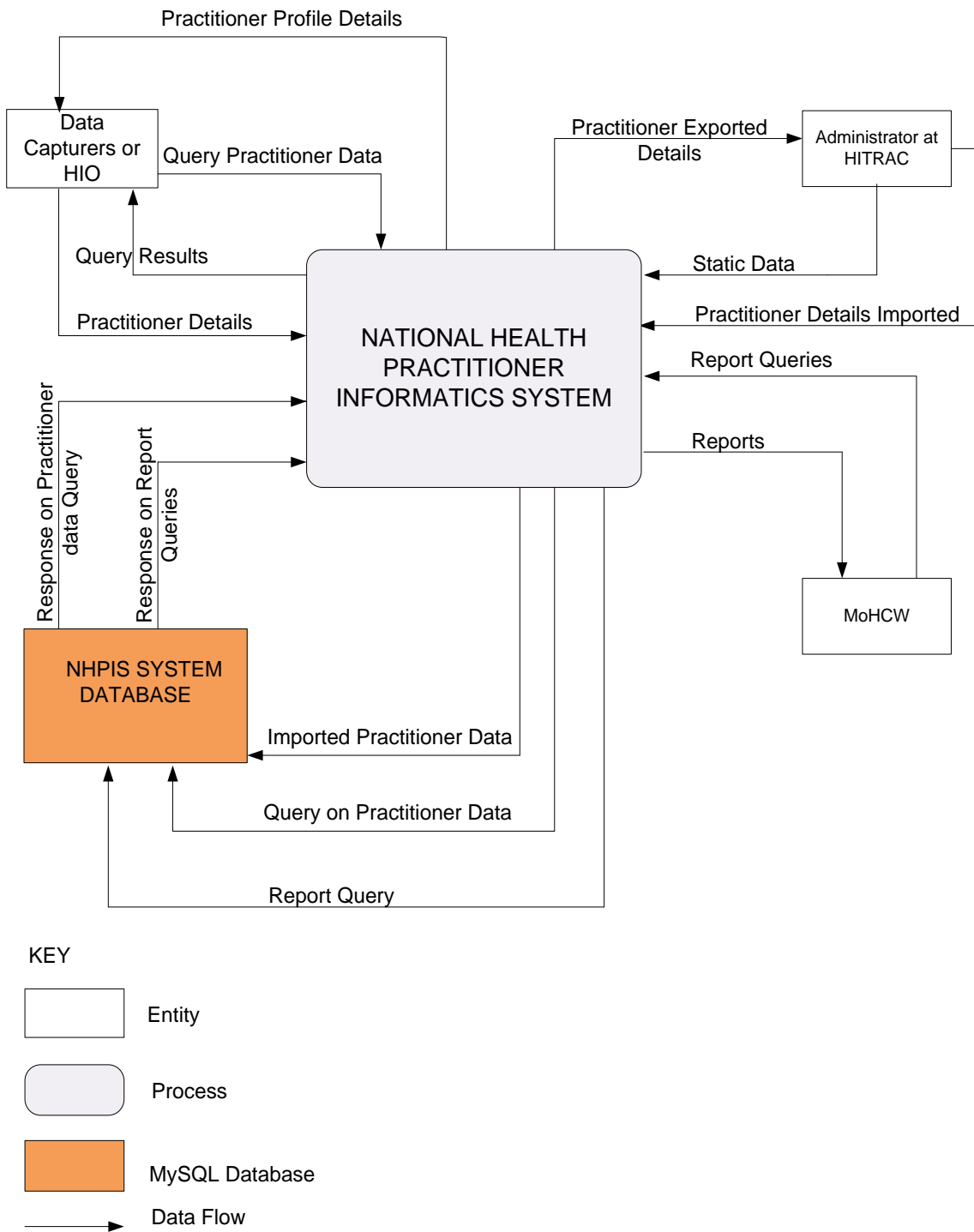
**Registration of Practitioner:** The system allows the HIO or data capturers to capture or update the practitioner's profile. The use of UUID in the system allows unique identification and eliminates duplication of the practitioners' data. The system validates the sensitive datasets like EC numbers, national ID numbers or passport numbers. The system does not register identical EC numbers for different practitioners

**Advanced Searching:** The system allows the users to navigate throughout the practitioner records in the database that is search by EC number, first name, and last name. The system should also allow a user to search a practitioner by province, district, station, post, employment status, employment category, employment type, and appointment term.

**Exporting and Importing:** The system allows the data capturers to export the registration details of a practitioner and allow the administrator to import the practitioner details.

**Downloading static data:** To ensure consistent naming standards and consistency of practitioners' datasets, the system allows only the national instance administrator to add, retire and edit static data sets, and allow the data capturers or HIO at provincial instances to download the static data.

## 4.2.2 Context Diagram of the Proposed System



**Fig 4.1 Context Diagram for the Proposed NHPIS system**

### 4.2.3 DFD of the new system

According to Ulrich et al (2000), a data flow diagram (DFD) is a graphical representation of the flow of data through an information system. It starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. It uses a top-down approach to show all the levels of the processes of a system. It shows the system's primary processes, data stores, sources, and destinations linked by data flows. The components of a DFD lead directly into physical design, with processes highlighting the programs and procedures, and data stores suggesting data entities, files, and databases.

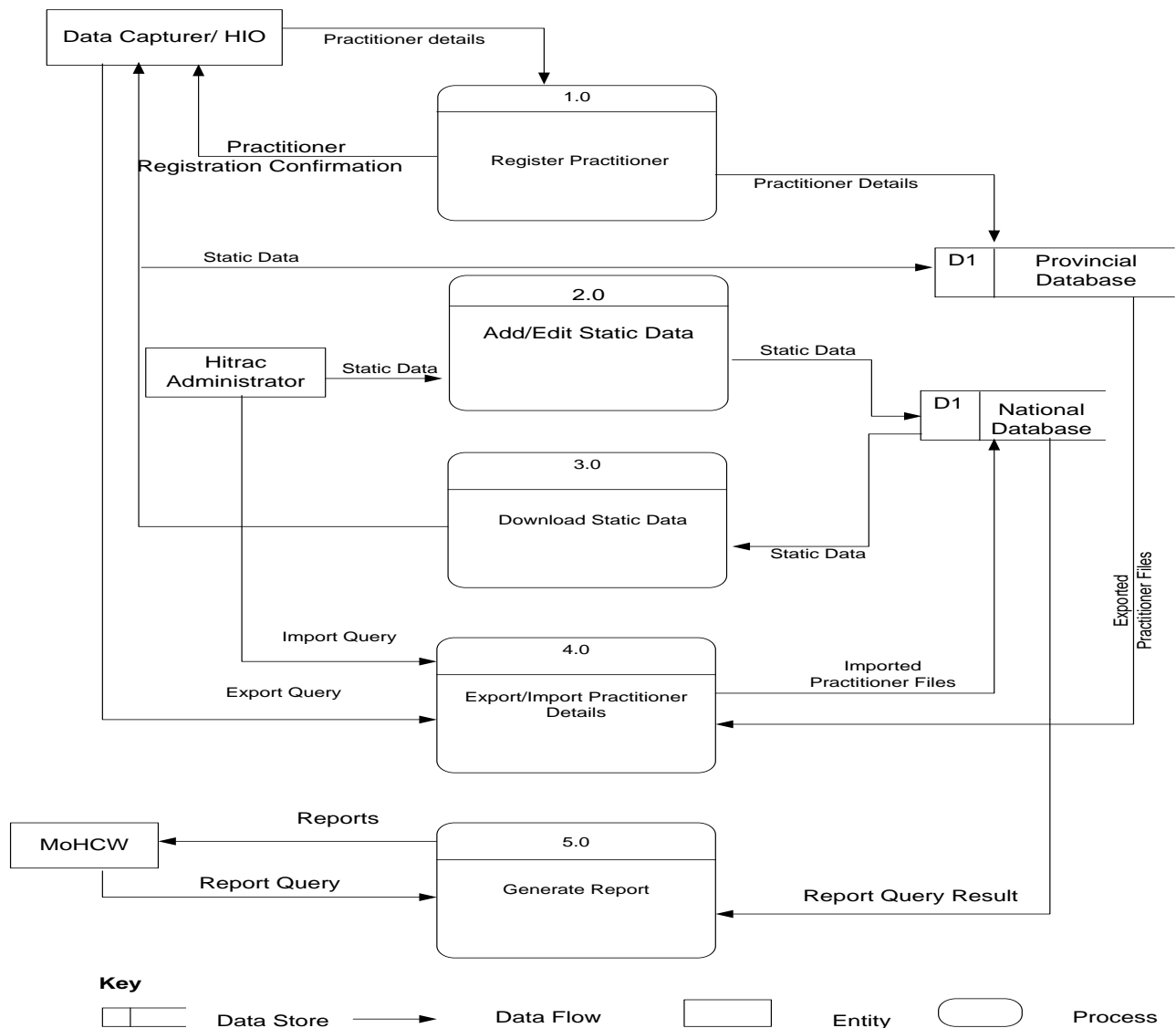


Fig 4.2 Data flow diagram of the proposed System

#### 4.2.4 Logic Flow Chart of the new system

A flow chart is a graphical representation of the sequence of transactions or processes of the proposed system. Logic flow charts are used to provide procedure manuals, which can be followed when carrying out certain tasks. A logical flow chart provides a pictorial representation of a series of processes in the NHPIS system. The flowchart specifies and documents the order in which tasks are performed. Flowcharts are used for documentation and for planning.

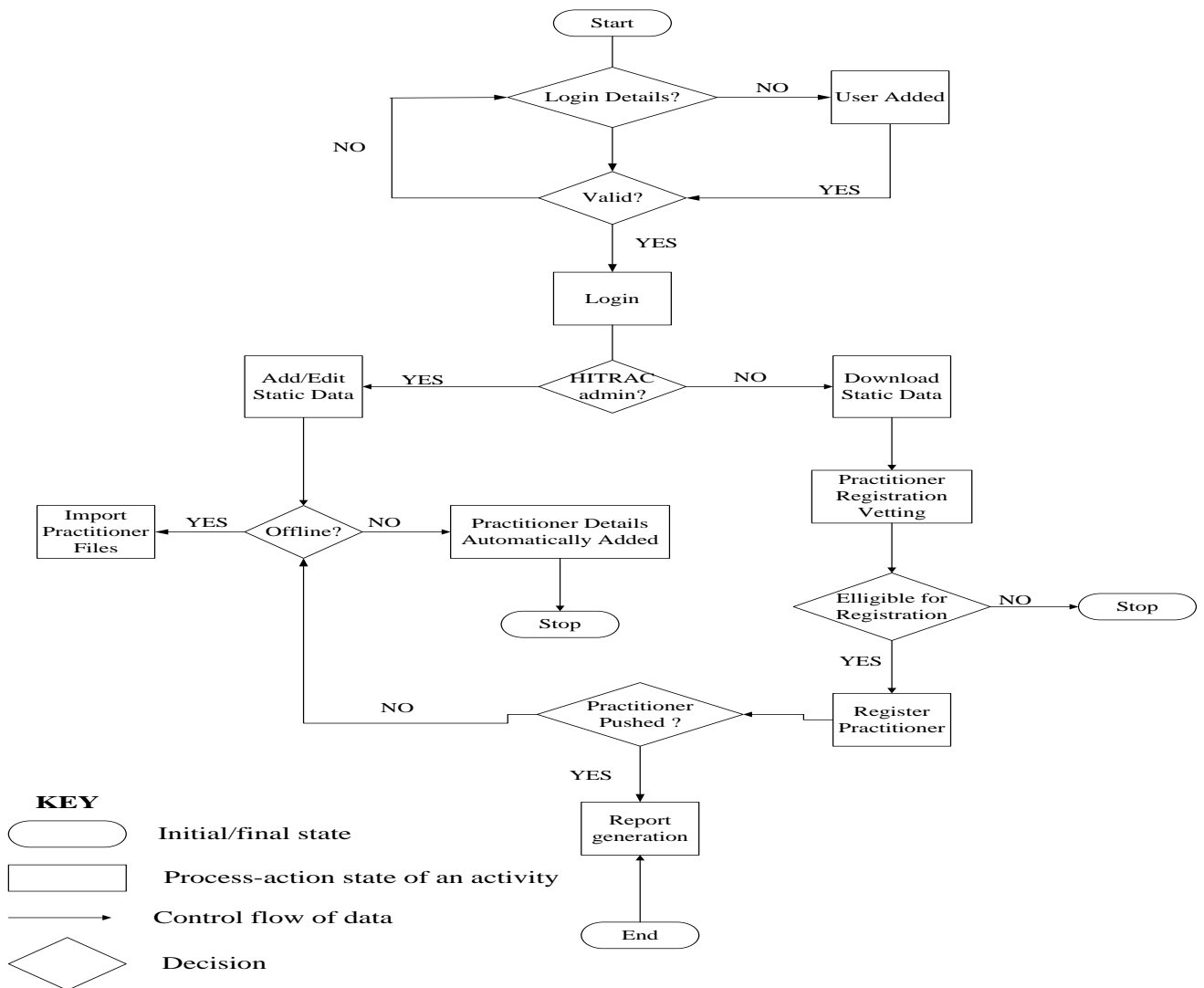


Fig 4.3 Logical Flow chart for the proposed NHPIS

### 4.3 Architectural design

An early stage of the system design process represents the link between specification and design processes. It is often carried out in parallel along with specified activities. It involves identifying major system components and their communications.

The advantages of explicit architecture are:

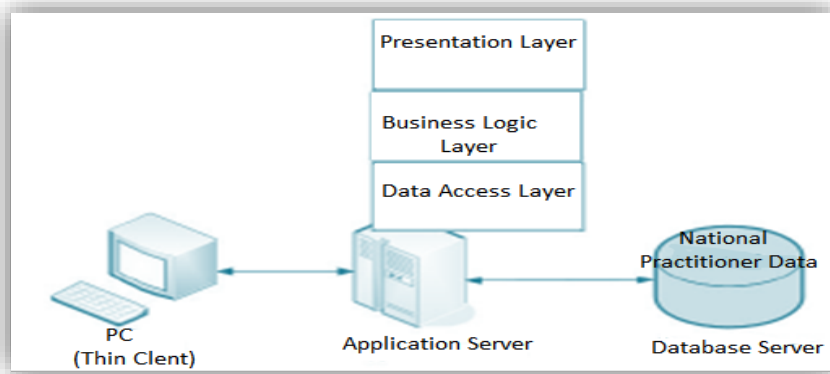
- System analysis: Means that analysis of whether the system can meet its non-functional requirements is possible.
- Large-scale reuse: The architecture may be reusable across a range of systems.

The Architecture and system characteristics include:

- Performance: Localize critical operations and minimize communications. Use large rather than fine-grain components.
- Security: Use a layered architecture with critical assets in the inner layers.
- Availability: Include redundant components and mechanisms for fault tolerance.
- Maintainability: Use fine-grain, replaceable components.

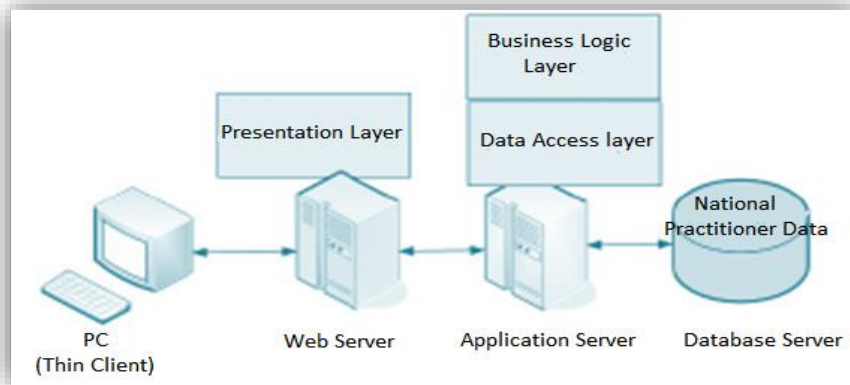
#### 4.3.1 Architecture Design of the NHPIS Software

Two-tier client applications are easy to develop but any changes in the user interface or business logic has to be rolled out for all clients in order to upgrade software hence the need for three-tier architecture. The client PC needs client software such as a browser to display the presentation content from the server. The server hosts the presentation, business and the data access logic.



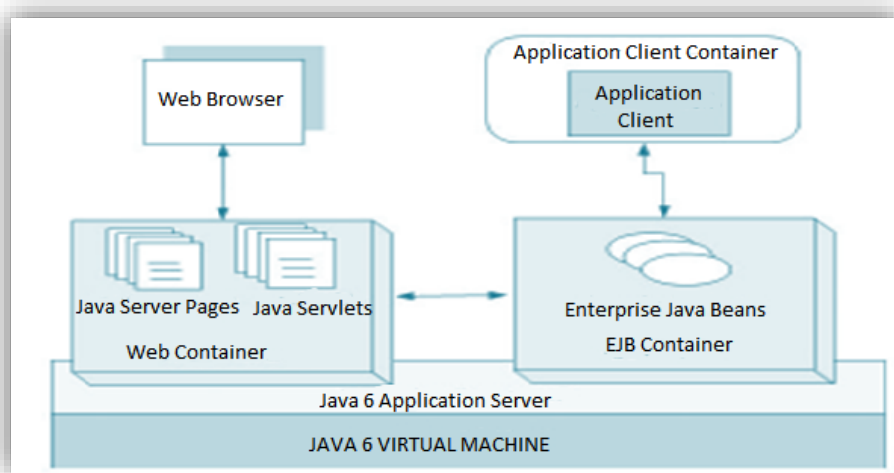
**Fig 4.4 Three-Tier Architecture**

With widespread growth of internet bandwidth, it is convenient to consider web-enabling services. As a result, the application server is no longer used for presentation layer, business logic and data accessing. Apache Tomcat, JBoss and Glassfish are used to generate presentation content and hence the presentation transferred to the browser on the client tier. However keeping these tiers as isolated silos serves no useful purpose.



**Fig 4.5 N-tier**

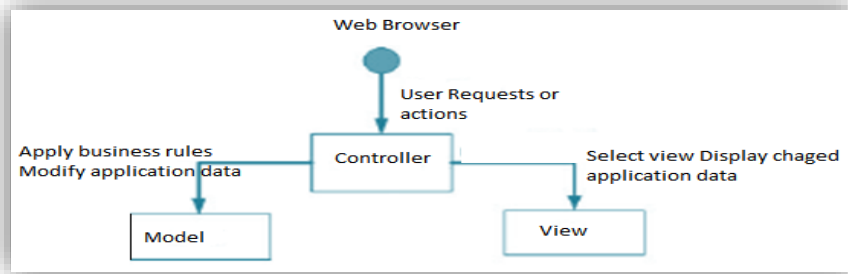
The n-tier is a complex architecture. For Java EE applications, the Java EE container architecture is suitable. The Java EE platform provides the essential system services through a container based architecture. The container provides the runtime environment for the object oriented application programs written in Java. It provides low-level services such as security, transaction, life cycle management, object lookup and caching, persistence and network communication.



**Fig 4.6 Java EE platform Architecture**

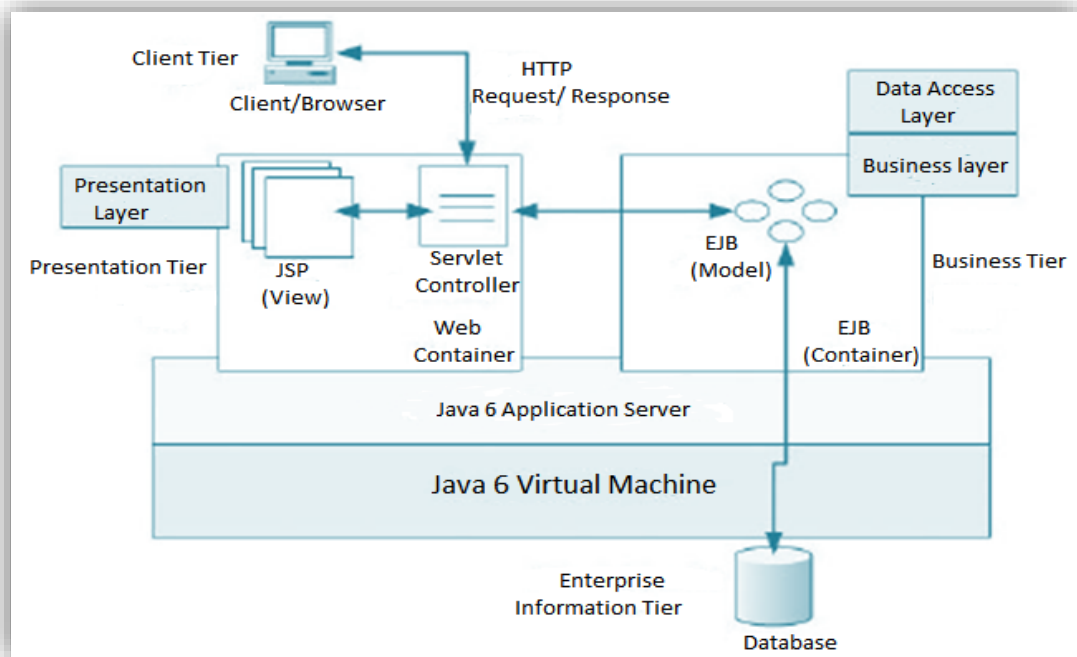


The proposed NHPIS system uses the Java EE architecture with Model, View and Control (MVC). The model manages the data of the application by applying business rules. The view is responsible for displaying the application data and presenting the control that allows the users to interact with the system. The controller takes of the mediation between the model and the view.



**Fig 4.7 Model View and Controller**

Any browser request from the user of the new NHPIS system is transferred via HTTP to a Servlet. The Servlet container invokes EJB model components, which encapsulate business rules and also retrieve and modify the application data. The retrieved and/or altered data can be displayed using JSP.



**Fig 4.8 Layered multi-tier Java EE application architecture based on MVC**

#### 4.4 Physical design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified or authenticated, how it is processed, and how it is displayed as output. In physical design, following requirements about the system are decided.

- Input requirement,
- Output requirements,
- Storage requirements,
- Processing Requirements,
- System control and backup or recovery.

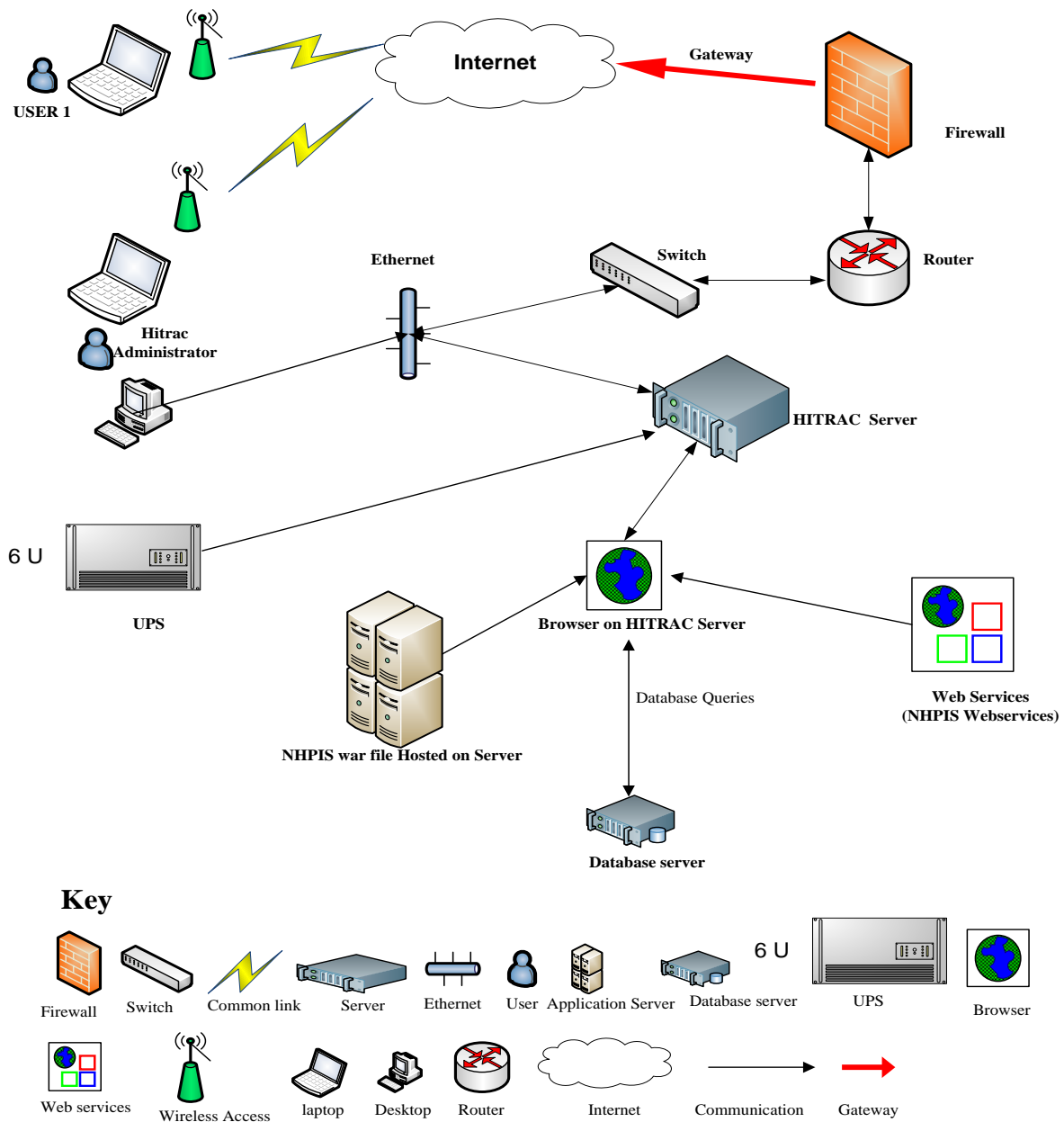
The physical part of systems design can generally be broken down into three sub-tasks:

- User Interface Design - how users add information to the system and with how the system presents information back to them.
- Data Design - how the data is represented and stored within the system
- Process Design - how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system

At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

**Table 4.1: Hardware and software specified**

Hardware	Software
Database server (Dell Power Edge R410)	Apache Tomcat Server
Client Computers (HP Compaq 500B)	Mozilla Firefox 13 or Update
Router	



**Fig 4.9: Physical design of the proposed system**

#### 4.5 Database design

A database is a collection of interrelated data designed to meet the varied information needs of an organization. A tool used to store information, or data. Information is something that we all use on a daily basis for a variety of reasons. With a database, users should be able to store data in

an organized manner. Designing a database requires an understanding of the business functions you want to model. A well-designed database also performs better.

Structured approach uses procedures, techniques, tools, and documentation aids to support and facilitate the process of design. Database design methodology has three main phases:

- Conceptual database design
- Logical database design
- Physical database design

#### **4.5.1 Conceptual database design**

Conceptual database design is the process of constructing a model of information used in an enterprise, independent of all physical considerations. This step involves:

- Constructing the ER Model
- Check the model for redundancy
- Validating the model against user transactions to ensure all the scenarios are supported

##### **4.5.1.1 ER Modeling**

An entity relation model is a pictorial Representation of the real world problem in terms of entities (which have attributes) and relations between the entities is referred as ER diagram.

- Entities: An entity is a class of distinct identifiable objects or concepts
- Relations: associations among entities.
- Attributes: attributes are properties or characteristics of entities.

### 4.5.1.2 Entity Relationship Diagram of the new system

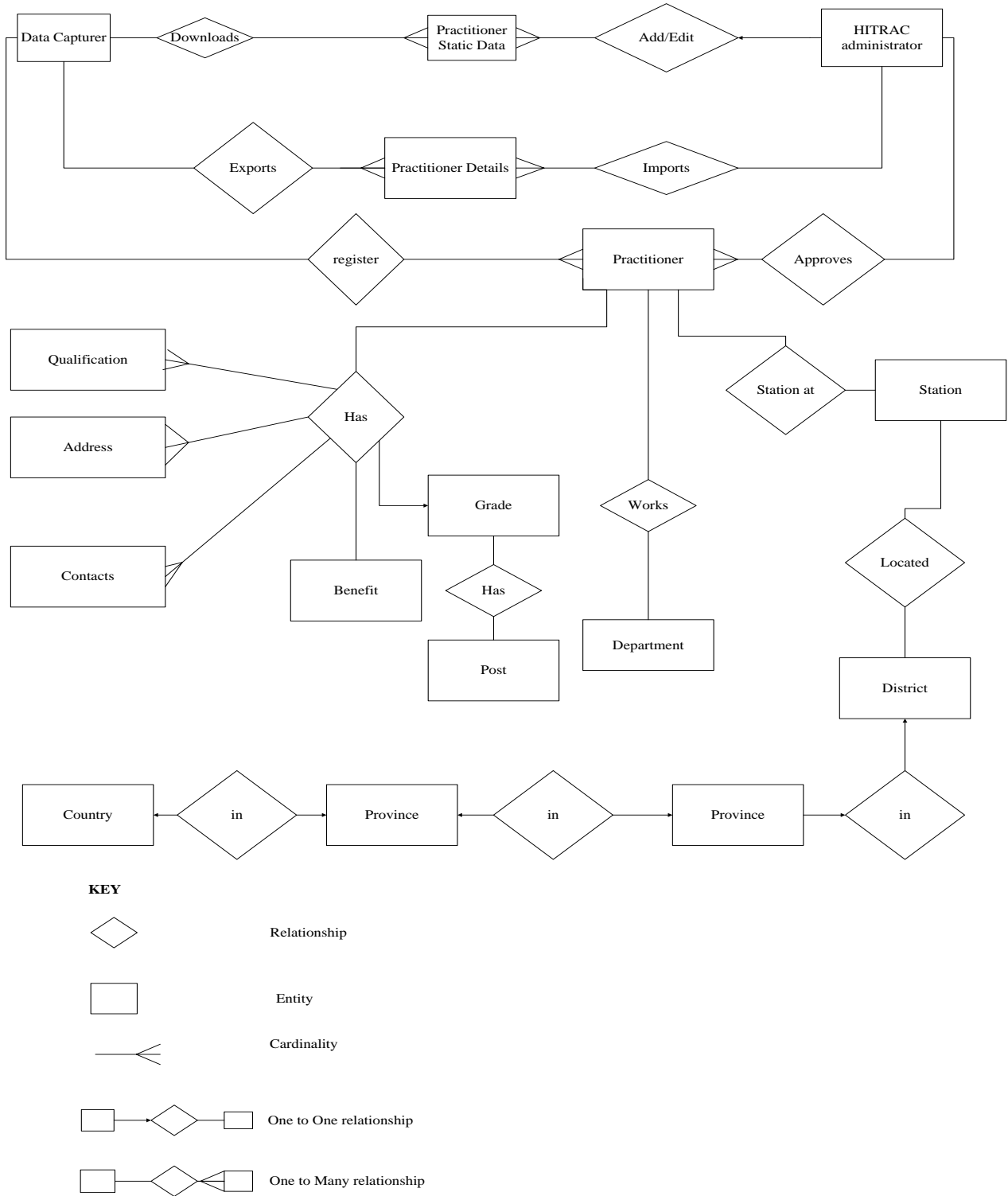


Fig 4.10 Entity Relationship Diagram

### 4.5.1.3 Enhanced Entity Relationship Diagram

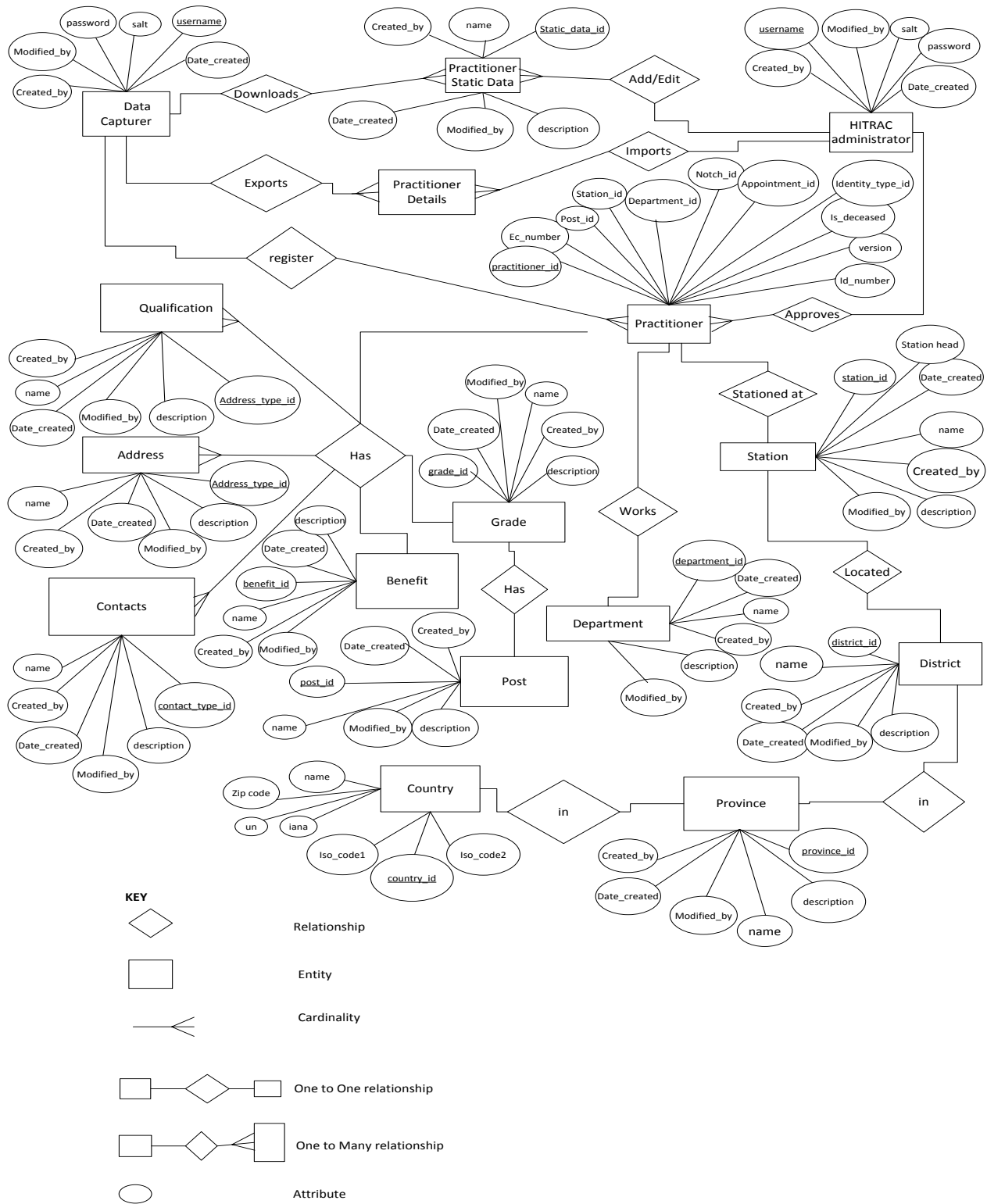


Fig 4.11 Enhanced Entity Relationship Diagram

## 4.5.2 Logical database design

Process of constructing a model of information used in an enterprise based on a specific data model (e.g. relational), but independent of a particular Database Management System (DBMS) and other physical considerations. This step involves:

- Table Generation from ER Model
- Normalization of Tables

### Table Generation from ER Model

When deriving the tables from ER model the cardinality of relationships is divided into the following categories:

**One-to-one:** One-to-one relationships results in a single entity. A table models each remaining entity with a primary key and attributes, some of which may be foreign keys

**One-to-many:** For one-to-many relationships, a foreign key attribute in another table references a primary key in the second table. This foreign key would refer to another table that would contain the “many” side of the relation.

**Many-to-many:** Many-to-many relationships between two entities by a third table that contains foreign keys referring to both the entities, (Buffer table).

**Normalization of Tables:** Normalization is a process of eliminating redundancy and other anomalies in the system. In most cases in the enterprise world, normalization up to Third Normal form would suffice. In certain cases or some transactions it is desirable that certain table be demoralized for efficiency in querying the database tables. In those cases, tables can be in demoralized form.

### 4.5.2.1 Database Tables

A table is a primary unit of physical storage for data in a database. Whenever a user accesses the database, the desired data is queried from a database table. Multiple tables might comprise a database; therefore, a relationship might exist between tables. Because tables store data, a table requires physical storage on the host computer for the database.

**Table 4.2 Country Table**

Table Name	Description	Field	Data Type	Foreign Keys
country	Countries	name	VARCHAR(255)	
		iso_code1	VARCHAR(45)	
		iso_code2	VARCHAR(45)	
		iana	VARCHAR(45)	
		un	VARCHAR(45)	
		ioc	VARCHAR(45)	
		iso	VARCHAR(45)	
		itu	VARCHAR(45)	

**Table 4.3 User Table**

Table Name	Description	Field	Data Type	Foreign Keys
user	user details	Created_by		
		username	VARCHAR(45)	
		password	VARCHAR(255)	
		salt	VARCHAR(128)	
		retired	SMALLINT(6)	
		retired_by	CHAR (36)	
		date_retired	DATETIME	
		date created	DATETIME	
		date_modified	DATETIME	
		modified by	CHAR (36)	

**Table 4.4 User\_role Table**

Table Name	Description	Field	Data Type	Foreign Keys
user_role	A user's roles	user_id	CHAR (36)	user_id in table user
		role	VARCHAR(50)	role in the table role
		username	CHAR(36)	user_id in table user



**Table 4.5 Privilege Table**

Table Name	Description	Field	Data Type	Foreign Keys
privilege	users' permission	retire_reason	VARCHAR(255)	
		retired	SMALLINT(6)	
		retired_by	CHAR (36)	user_id in user table
		modified_by	CHAR (36)	user_id in user table
		date_modified	DATETIME	
		date_created	DATETIME	
		date_retired	DATETIME	
		created_by	CHAR (36)	user_id in user table
		description		
		privilege		

**Table 4.6 Role Table**

Table Name	Description	Field	Data Type	Foreign Keys
role	users' permission	retire_reason	VARCHAR(255)	
		retired	SMALLINT(6)	
		retired_by	CHAR (36)	user_id in user table
		modified_by	CHAR (36)	user_id in user table
		date_modified	DATETIME	
		date_created	DATETIME	
		date_retired	DATETIME	
		created_by	CHAR (36)	user_id in user table
		description	VARCHAR(255)	
		role	VARCHAR(50)	

**Table 4.7 Qualification Table**

Table Name	Description	Field	Data Type	Foreign Keys
qualification	practitioner's qualifications	creator	CHAR (36)	
		qualification_id	CHAR (36)	
		practitioner_id	CHAR (36)	
		created by	CHAR (36)	
		date created	DATE	
		date_modified	DATETIME	
		date_voided	DATETIME	
		void_reason	VARCHAR(255)	
		voided	SMALLINT(6)	
		modified by	CHAR (36)	
		voided_by	CHAR (36)	
		institution_id	VARCHAR(36)	

**Table 4.8 Title Table**

Table Name	Description	Field	Data Type	Foreign Keys
title	person's title	title_id	CHAR (36)	
		name	VARCHAR(255)	
		description	VARCHAR(45)	
		date created	DATETIME	
		created by	CHAR (36)	user_id in the table user
		retired_by	CHAR (36)	user_id in the table user
		retire_reason	VARCHAR(255)	
		retired	SMALLINT(6)	
		date_retired	DATETIME	
		modified_by	CHAR (36)	user_id in the table user
		date_modified	DATETIME	

**Table 4.9 Address Type Table**

Table Name	Description	Field	Data Type	Foreign Keys
address_type	Type of addresses	address_type_id	CHAR(36)- PK	
		name	VARCHAR(50)	
		description	VARCHAR(255)	
		date_created	DATETIME	
		date_modified	DATETIME	
		description	VARCHAR(255)	
		retired	SMALLINT(6)	

**Table 4.10 Practitioner Table**

Table Name	Description	Field	Data Type	Foreign Keys
practitioner	Practitioners data	practitioner_id	CHAR (36)	PK in table person
		ec_number	VARCHAR(45)	
		department_id	CHAR (36)	PK in table department
		created_by	CHAR (36)	PK in table user
		date_created	DATE	
		date_modified	DATETIME	
		modified_by	CHAR (36)	PK in table user
		post_id	SMALLINT(6)	PK in table post
		station_id	CHAR (36)	PK in table station
		appointment_date	DATE	
		notch_id	CHAR (36)	notch_id in notch table
		appointment_term_id	CHAR (36)	PK in appointment_term

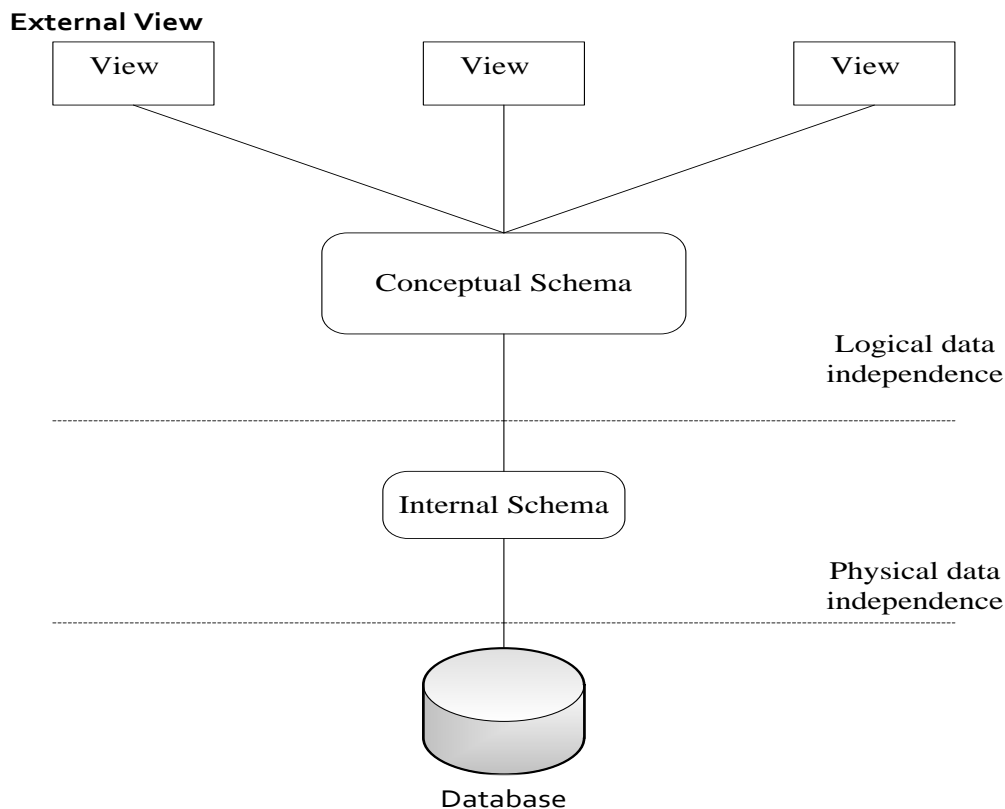
**Table 4.11 Person Table**

Table Name	Description	Field	Data Type	Foreign Keys
person	demography data	person_id	CHAR(36)	
		date_of_birth	DATE	
		gender	VARCHAR(45)	
		marital_status_id	CHAR(36)	PK in marital_status table
		title_id	CHAR(36)	title_id in table title
		created_by	CHAR(36)	user_id in user_table
		date_created	DATETIME	
		date_modified	DATETIME	
		modified_by	CHAR(36)	user_id in user_table
		voided	SMALLINT(6)	
		voided_by	CHAR(36)	user_id in user table
		date_voided	DATETIME	
		void_reason	VARCHAR(45)	
		nationality	VARCHAR(250)	
		dead	SMALLINT(6)	
		date_of_death	DATE	
		firstname	VARCHAR(45)	
		lastname	VARCHAR(45)	
		middlename	VARCHAR(45)	
		identity_no	VARCHAR(45)	
		identity_type_id	CHAR(36)	PK in identity_type table

### 4.5.3 Physical Database Design

Process of producing a description of the implementation of the database on secondary storage, it describes the physical configuration of the database on the storage media. This step involves describing the base relations, file organizations, and indexes design used to achieve efficient access to the data, and any associated integrity constraints and security measures.

The American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC), ANSI-SPARC database architecture uses three levels of abstraction: external, conceptual, and internal.



**Fig 4.12 ANSI-SPARC Database design**

#### 4.5.3.1 External Level

The external level represents the user's view of the database. It describes the part of the database that is relevant to a particular user.

#### **4.5.3.2 Conceptual Level**

This level describes what data is actually stored in the database and the relationships that exist amongst the data. The data is stored in tables, the table attributes specific feature's (integer, string and the exact size, format) (Viega and McGraw, 2001).

#### **4.5.3.3 Internal Level**

This level shows the highest level of abstraction and it is the physical representation of the database on the computer. This level describes how the data is stored in the database. The database manipulation is done using Structured Query Language (SQL). Below the internal level there is a physical level that may be managed by the operating system under the direction of the DBMS. A DBMS is a software system that enables users to define, create, and maintain the database and which provides controlled access to this database.

### **4.6 Program design**

The function of the program design phase is to produce the actual working software modules specified in the System design phase. The program specifications from the system design phase are used to design and code the individual program modules. Each module must conform to the design documents produced in earlier phases and must be thoroughly tested in readiness for integration and testing. The program design can be illustrated in three diagrams namely;

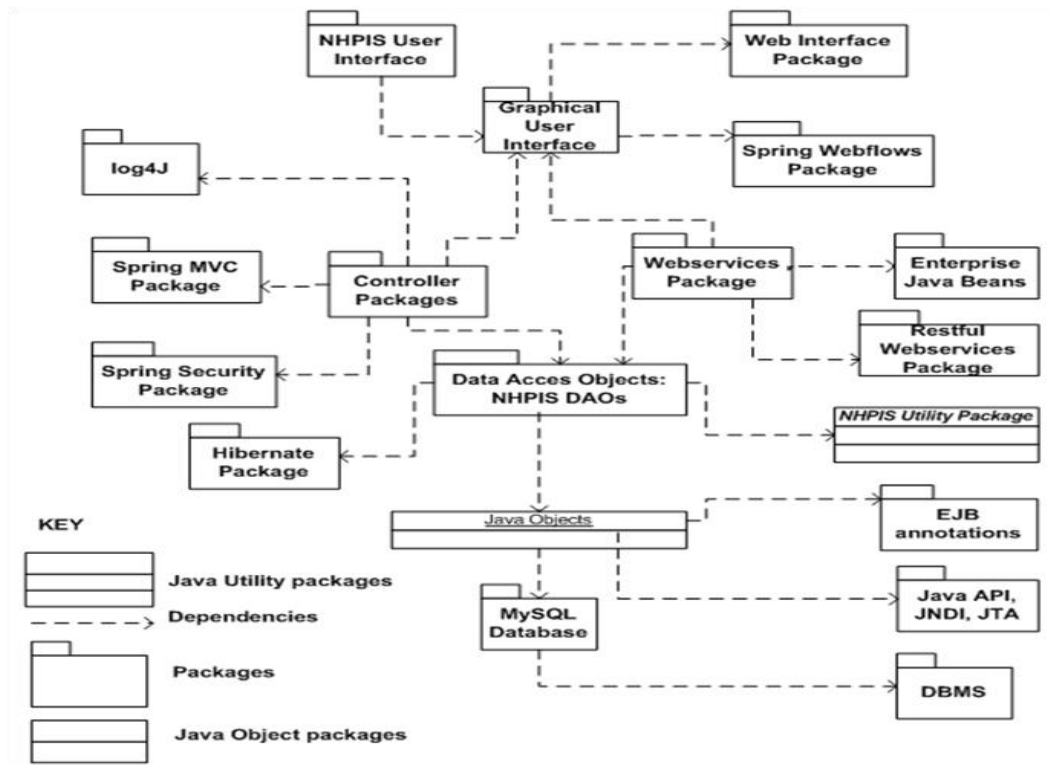
- Class Diagram.
- Package Diagram.
- Sequence Diagram.

#### **4.6.1 Package Diagram**

A package diagram in the Unified Modeling Language (UML) depicts the dependencies between the packages that make up a model. In addition to the standard UML Dependency relationship, there are two special types of dependencies defined between packages:

Package import - a package import is a relationship between an importing namespace and a package, indicating that the importing namespace adds the names of the members of the package to its own namespace.

Package merge - a package merge is a directed relationship between two packages that indicates that the contents of the two packages are to be combined.



**Fig 4.13 Package Diagram**

#### 4.6.2 Class Diagram of the new system

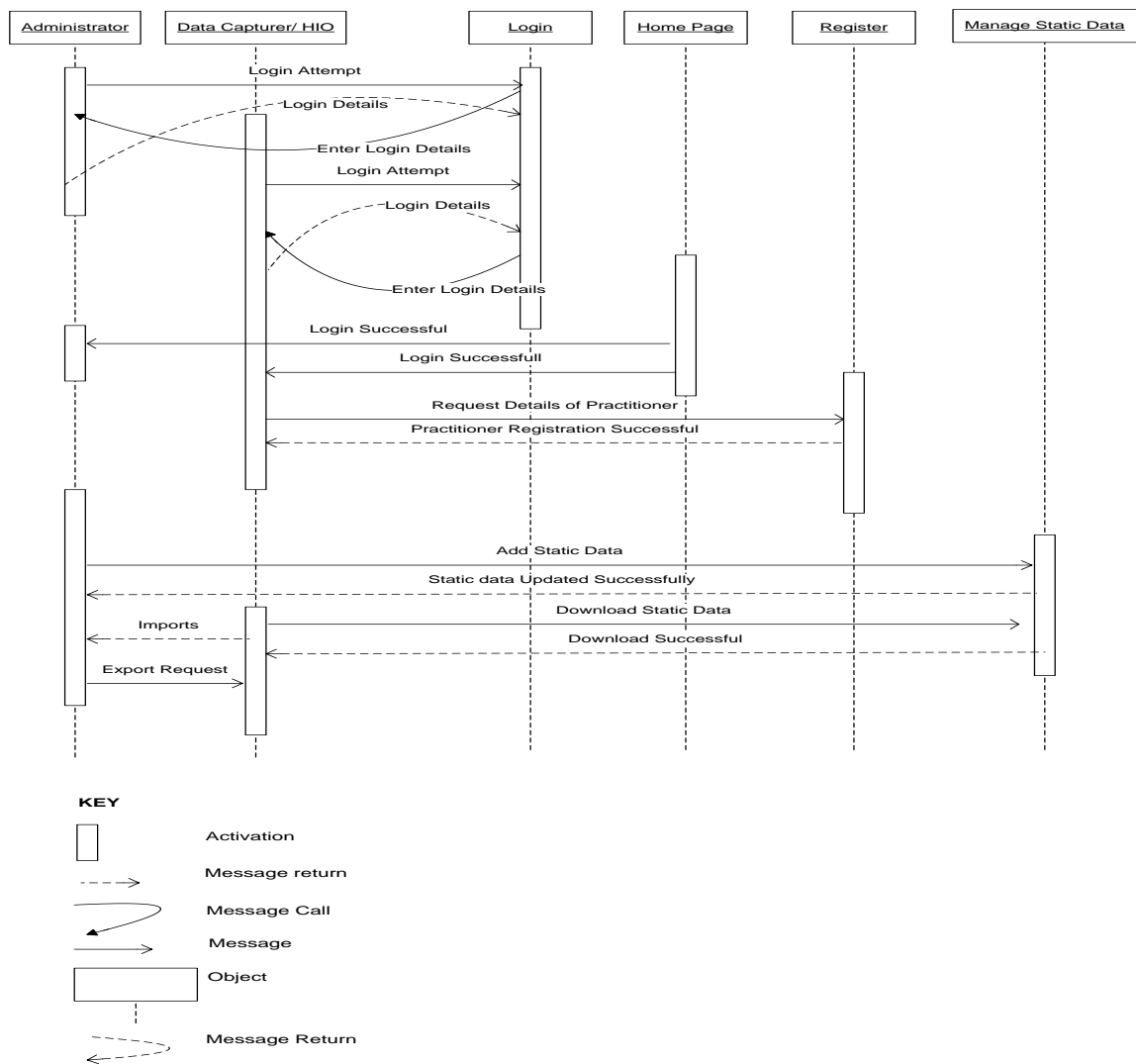
A class diagram is a static structure diagram that describes the structure of a system by showing the system's classes, their attributes and the relationships among the classes. Static models of a system describe the structural relationships that hold between the pieces of data manipulated by the system. Two major relationships exist between classes and these are inheritance and association. Inheritance, also known as generalization, describes a super class/subclass relationship. An empty arrow that points from the subclass to the super class represents an inheritance relationship.





### 4.6.3 Sequence Diagram

A sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically (but not always), are associated with use case realizations in the Logical View of the system under development.



**Fig 4.15 Sequence diagram of the proposed system**

## 4.7 Interface design

User interface design or user interface engineering is the design of computers, appliances, machines, mobile communication devices, software applications, and websites with the focus on the user's experience and interaction. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals—what is often called user-centered design. Good user interface design facilitates finishing the task without drawing unnecessary attention to its self. Graphic design may be utilized to support its usability. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

### 4.7.1 Menu Design

This is the design of pages that a user sees when he or she logs into the system. This usually includes the main menu and the sub menus

#### 4.7.1.1 Main Menu

When a user logs into the system the main menu form below should appear.

Name of the System	
Home Link   Manage Practitioner   Create Practitioner   Administrator   Reports   Current User(Login name)   Logout Link	
Registration	<input type="text" value="Search Practitioner"/>  <input type="text" value="Search Practitioner by EC Number, firstname, lastname, or both"/>
Reports	
Data Export/Import	
System Properties	
Name of Province	
Logout Link	

**Fig 4.16 Main Menu Form**

#### 4.7.1.2 Sub Menus

After logging into the System, a user can register, download, upload, or manage practitioners. If a user is a provincial admin, he or she downloads all the static data used in registering a practitioner using the form below

<a href="#">Manage Allowance Type</a>   <a href="#">Download Allowance Type</a>		
<u>Allowance Type</u>	<u>Description</u>	<u>Action Command</u>
Housing Allowance	Housing Allowance	<a href="#">View</a>

**Fig 4.17 Static Data download Form**

If a user clicks the Data export link the below is displayed

<b>Practitioner File Export</b>	
Minimum Update Date	<input type="text"/>
Maximum Update Date	<input type="text"/>
<a href="#">Download Button</a>	

**Fig 4.18 Data export Form**

If a user clicks the Data import link the below is displayed

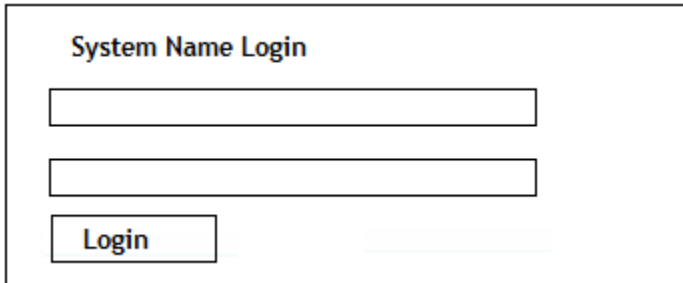
<b>Practitioner File Export</b>		
Name	<input type="text"/>	
File	<input type="text"/>	<a href="#">Browse</a>
	<a href="#">Import</a>	

**Fig 4.19 Data import Form**

### 4.7.2 Input Design

It is the process of converting a user-oriented description of the input into a computer-based system. The design of input focuses on controlling the amount of input required, controlling the errors and keeping the process simple. The input design is a sketch of the input forms in the system.

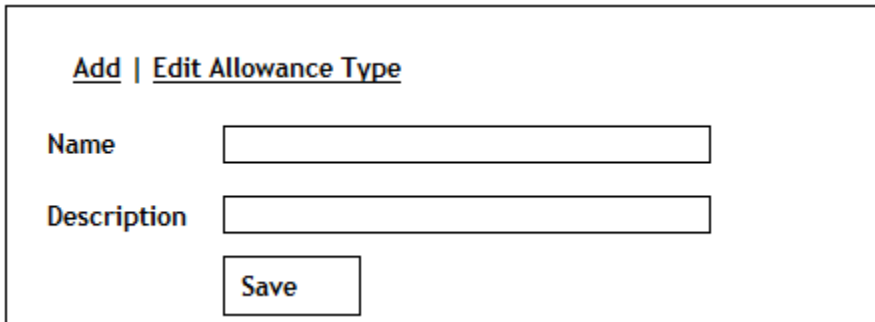
Login user interface of the proposed NHPIS System



The form is titled "System Name Login". It contains two horizontal input fields stacked vertically. Below the second input field is a rectangular button labeled "Login".

**Fig 4.20 Login Form**

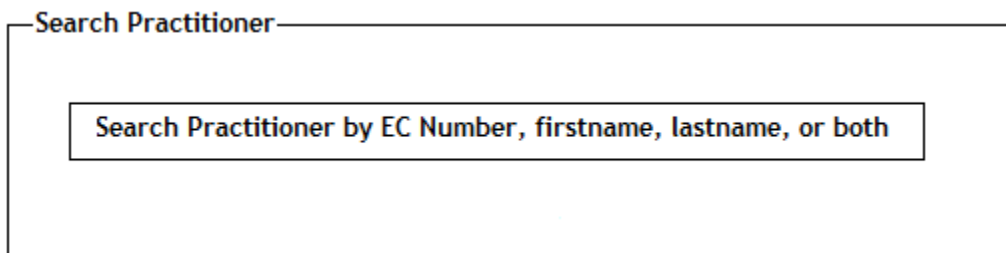
The after logging in the administrator at HITRAC can add static data using the form below.



The form is titled "Add | Edit Allowance Type". It features two input fields: "Name" and "Description". Below the "Description" field is a rectangular button labeled "Save".

**Fig 4.21 Add Static data Form**

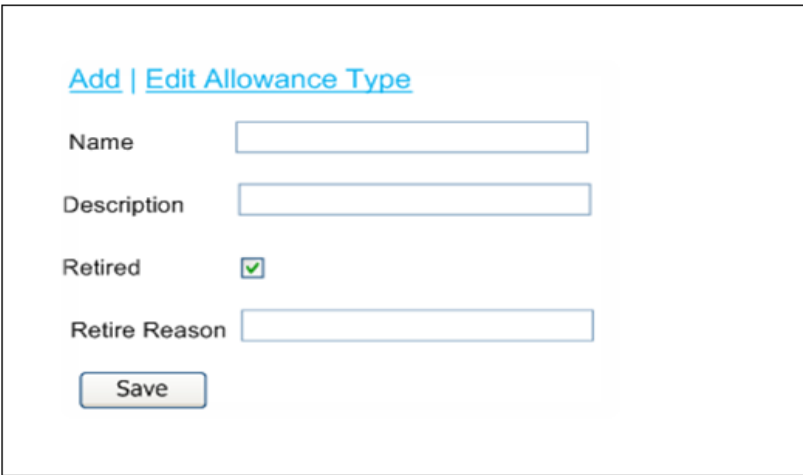
A user of the system can search for a practitioner using the Search form.



The form is titled "Search Practitioner". It contains a single search input field with the placeholder text "Search Practitioner by EC Number, firstname, lastname, or both".

**Fig 4.22 Search Practitioner Form**

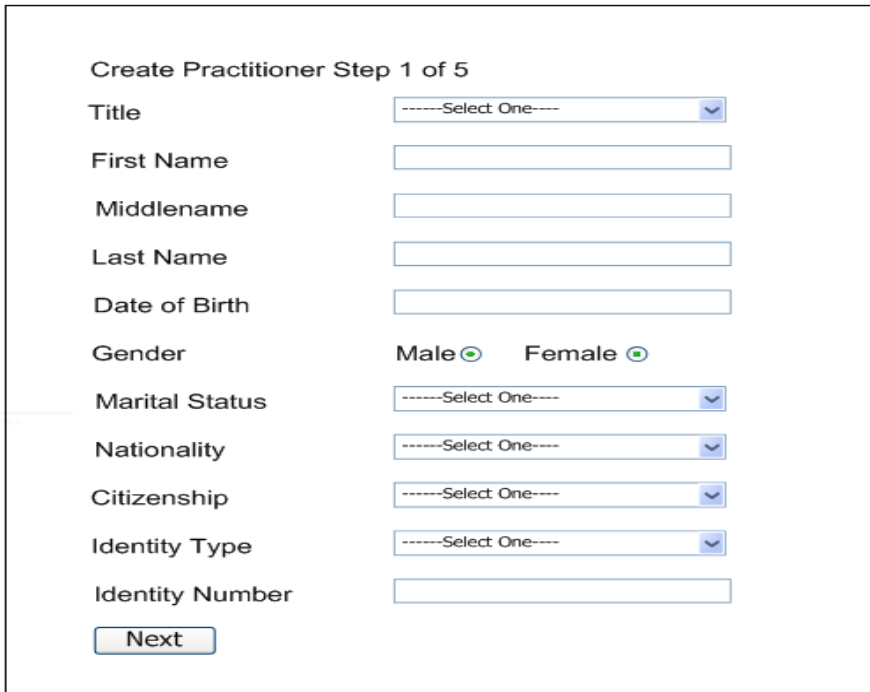
The administrator can also edit, retire a static data using the form below.



The form is titled "Add | Edit Allowance Type" in blue text. It contains the following fields: "Name" (text input), "Description" (text input), "Retired" (checkbox, checked), and "Retire Reason" (text input). A "Save" button is located at the bottom left.

**Fig 4.23 Add or Edit Static data Form**

The HIO registers a practitioner using the create practitioner form  
Step one of creating a practitioner



The form is titled "Create Practitioner Step 1 of 5". It contains the following fields: "Title" (dropdown menu, "-----Select One-----"), "First Name" (text input), "Middlename" (text input), "Last Name" (text input), "Date of Birth" (text input), "Gender" (radio buttons, "Male" selected, "Female"), "Marital Status" (dropdown menu, "-----Select One-----"), "Nationality" (dropdown menu, "-----Select One-----"), "Citizenship" (dropdown menu, "-----Select One-----"), "Identity Type" (dropdown menu, "-----Select One-----"), and "Identity Number" (text input). A "Next" button is located at the bottom left.

**Fig 4.24 Create Practitioner step 1 of 5 Form.**

Step two of creating a practitioner

**Create Practitioner Step 2 of 5**

*\* Can not Continue without adding one Address, Contact or Qualification*

**Add Practitioner Contact Detail**

Type

Detail

Active

**Add Practitioner Contact Detail**

Type

Detail

Active

**Add Practitioner Contact Detail**

Qualification

Training Institution

Awarding Institution

Year Acquired

Extra

**Fig 4.25 Create Practitioner step 2 of 5 Form.**

Step three of creating a practitioner

Create Practitioner Step 3 of 5

Ec Number:

Date of Assumption of Duty

Station

Post

Employment Status

Department

Deployment Reason

Date of Appointment

Appointment Term

Employment Type

**Fig 4.26 Create Practitioner step 3 of 5 Form.**

At step 4 of creating a practitioner, the data capturer or HIO checks if the practitioners have submitted the supporting documents.

Create Practitioner Step 4 of 5

Qualification Proof Submitted

Experience Proof Submitted

Medical Certificate Submitted

Birth Date Proof Submitted

Prev Gvt Engagement Submitted

Satisfactory Medical Service

Assumption of Duty Submitted

Marriage Certificate Submitted

Tax Coding Advice Submitted

Security Vetting Submitted

**Fig 4.27 Create Practitioner step 4 of 5 Form.**





### 4.7.3 Output User Interface

A quality output meets the requirements of the end user and presents the information clearly. In output design it is determined how the information is to be displayed for immediate need also the hard copy output.

The overall Practitioner profile

**Confirm Details Create Practitioner Step 5 of 5**

Name	Prof Artwell Mamvura	Age	40	
Identifier	07-182879- B- 79	EC Number	1234567D	

Practitioner Overview	Contact Details	Qualification	Benefits	Operations	Leave
<b>Date of Birth</b>	03/04/1973	<b>Marital Status</b>	Single		
<b>Nationality</b>	Zimbabwean	<b>Citizenship</b>	Zimbabwean		
<b>Initial Employment Date</b>	09/09/1997	<b>Station</b>	Avondale Clinic		
<b>Department</b>	Paediatrics	<b>Post</b>	Doctor in Charge		
<b>Employment Type</b>	Part time	<b>Employment Status</b>	Employed		
<b>Years in Service</b>	16	<b>Notification</b>	Due for retirement after 25 Years		

**Fig 4.30 Advanced Practitioner Search Form**

Reports generated from the system for MoHCW. An example of the reports generated:

Logo	<b>Ministry Of Health and Child Welfare</b> All Practitioners in the Database	Logo					
Name	Gender	Nationality	EC Number	Station	Post	Department	Date Employed
1. Artwell Mamvura Prof	Male	Zimbabwean	1235647D	Avondale Clinic	Doctor in Charge	Paediatrics	09/09/1997
2. Artwell Mamvura Dr	Male	Zimbabwean	2312345T	Mutoko Clinic	Doctor in Charge	Casualties	09/09/2000
<b>Total 2</b>							
<b>NHPIS HITRAC Health Informatics Training and Research Advancement Centre</b>							

**Fig 4.31 Reports generated**

## 4.8 Security Design

The NHPIS system was design with security in mind. The diagram below shows how the researcher secured the NHPIS software from the browser up to the database.

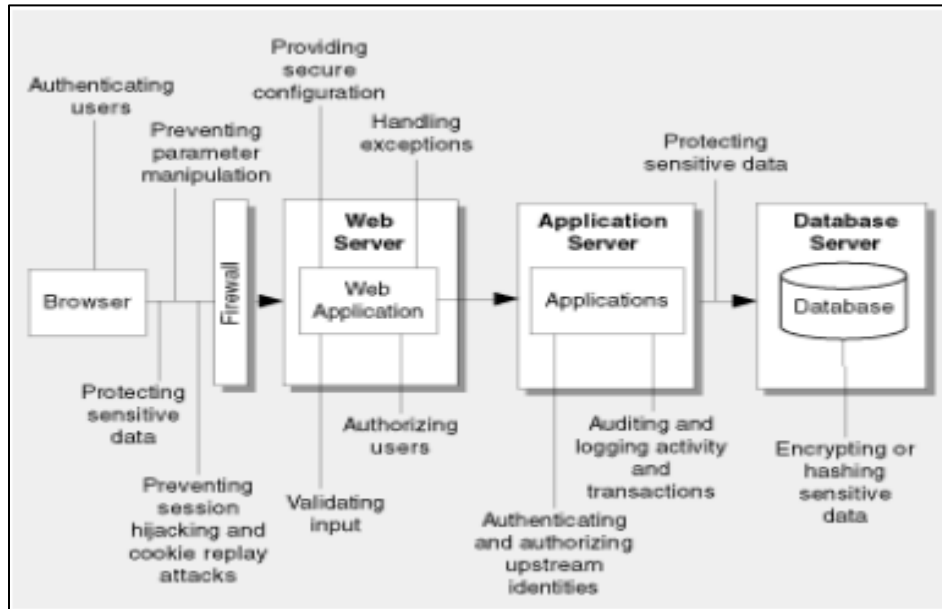


Figure 4.31 Security Design Issues

### 4.8.1 Web application design issues

The design guidelines are organized by application vulnerability category poor design in these areas, in particular, leads to security vulnerabilities (Cavusoglu, Mishra and Raghunathan, 2002). Below is a list of web application vulnerabilities and potential problem due to bad design:

- Input Validation - Attacks performed by embedding malicious strings in query strings, form fields, cookies, and HTTP headers. These include command execution, cross-site scripting (XSS), SQL injection, and buffer overflow attacks.
- Authentication - Identity spoofing, password cracking, elevation of privileges, and unauthorized access.
- Authorization - Access to confidential or restricted data, tampering, and execution of unauthorized operations.
- Configuration Management - Unauthorized access to administration interfaces, ability to update configuration data, and unauthorized access to user accounts and account profiles.

- Sensitive Data - Confidential information disclosure and data tampering.
- Session Management - Capture of session identifiers resulting in session hijacking and identity spoofing.
- Cryptography - Access to confidential data or account credentials, or both.
- Parameter Manipulation - Path traversal attacks, command execution, and bypass of access control mechanisms among others, leading to information disclosure, elevation of privileges, and denial of service.
- Exception Management - Denial of service and disclosure of sensitive system level details.
- Auditing and Logging - Failure to spot the signs of intrusion, inability to prove a user's actions, and difficulties

#### **4.9 Conclusion**

The design phase enabled the researcher to come up with the way the system operates. The researcher had a view of how the physical architecture, NHPIS software components are integrated. Inputs, processes and outputs of the new system were designed. Data flow, entity relationships and the database were designed as a whole. All this was done to prepare for the implementation of the proposed system.

Security design permeated every stage of the product development life cycle and it was a focal point of application design. Pay particular attention to the design of a solid authentication and authorization strategy. The majority of application level attacks rely on maliciously formed input data and poor application input validation. The guidance presented in this chapter helped the researcher solve challenging aspects of designing and building secure applications.

## CHAPTER FIVE: IMPLEMENTATION PHASE

### 5.1 Introduction

System implementation is the final phase in the SDLC. During this phase the system is developed, (Swanson, 1976). It is the longest, most expensive phase of the SDLC. The researcher built the system components from scratch. From design and analysis phase, the researcher developed exactly what he had proposed.

The implementation phase has three phases which are:

- System construction: Building and testing the system to ensure it performs as designed in the design phase and as per user requirements specifications
- System installation: Deploying the system for live production
- Post-Implementation review: a support plan for the system, as well as a systematic way for identifying changes in the system.

The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging. The end deliverable is the product itself. In this phase, the software product was installed, initial user training was completed and user documentation was delivered. When this phase is completed, the application is in steady-state production. The researcher established best development practices to detect and remove security and privacy threats using spring security.

Successful completion of the implementation phase comprises:

- System Testing
- System deployment or installation
- User training
- System maintenance

Once the system is in steady-state production, it is reviewed to ensure that all of the goals in the project plan of the planning phase were successfully implemented to the users' full satisfactory. The NHPIS system was developed with the users of the system at heart.

## 5.2 Coding

This is also called the programming phase in which the programmer converts the program specifications into computer instructions, which we refer to as programs. The programs coordinate the data movements and control the entire process in a system. It is generally felt that the programs must be modular in nature. This helps in fast development, maintenance and future changes, if required. The software developers take the design documents and development tools (editors, compilers, debuggers etc.) and start writing software. This is usually the longest phase in the product life cycle. A well-written code reduces the testing and maintenance effort. Hence, during coding simplicity and clarity should be strived for (Reutter, 1981).

### 5.2.1 Pseudo Code

According to Reutter (1981) pseudo-code is a non-formal language, a way to create a logical structure, describing the actions executed by the application. The researcher described the application logic using his native language, without applying the structural rules of a specific programming language.

#### **Creating a connection to the MySQL Database**

*If username and password are correct*

Connect to the Database

*Else*

Database Connection error

#### **Method to save a record**

Object method saveObject

Return the method get Hibernate Template method and use the merge method to save an entity

#### **Method to delete a record**

Return nothing and deleteObject by objectID

Get Hibernate Template method and the delete by objectID

### **Method to update a record**

Object get Object (Parameter ObjectID)

Object obj = (Object) getHibernateTemplate method and use the get method

*If obj equals null*

Log warning "Object not found"

Throw new Object Retrieval Failure Exception

*Else*

Return obj

### **Method to list records**

Suppress Warnings unchecked

List Object get All method

Returns get Hibernate Template method and load all method

### **Method to list distinct records**

Suppress Warnings unchecked

List<Object> getAllDistinctObject method

Collection result equals new Linked Hash Set getAll method

Return new Array List method and pass parameter equals result

### **Method to check duplication of records**

Object obj equals (Object) getHibernateTemplate and get method parameter equals objectID

*If obj is not equals null*

Return obj

### **Create Reports**

Get database connection

*If database username and password equals null or wrong*

Display Database Connection Failed

*Else*

Execute Report Query

Display Record in PDF format

### **Download and Export**

*If result has Errors*

For Object Error: Results get All Errors

Display "Error: " + errors get Code + " - " + errors get Default Message

Return uploads File

*Try*

Upload list and Return list

*Catch Exception*

Return toString method

## 5.3 Testing

Testing is the process of evaluating a system or application, to check whether the application meets all requirements of the client and to detect the errors. Testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements (Barlow, 1994). System testing was performed manually and functional tests were written in detail. The researcher used Apache Maven package to perform an automated JUnit testing on the methods used in the NHPIS application.

### 5.3.1 Methods of Testing

The testing methods that were used for testing the NHPIS software include black, white and grey box testing.

#### 5.3.1.1 Black Box Testing



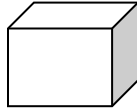
The technique of testing without having any knowledge of the interior workings of the application is Black Box testing. The tester was familiar with the system architecture and had no access to the source code. Typically, when performing a black box test, a tester interacts with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon, (Taggart, 1986).

The advantages of Black Box Testing are:

- Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems.
- Well-suited and efficient for large code segments and code access not required.

The disadvantages of Black Box Testing are:

- The test cases are difficult to design.
- The tester cannot target specific code segments or error prone areas.



### 5.3.1.2 White Box Testing

According to Taggart (1986), white box testing is the detailed investigation of internal logic and structure of the code. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.

The advantages of White Box Testing are:

- It helps in optimizing the code.
- Extra lines of code can be removed which can bring in hidden defects.

The disadvantages of White Box Testing are:

- Because a skilled tester is needed to perform white box testing, the costs are increased.
- It is difficult to maintain white box testing as the use of specialized tools like code analyzers and debugging tools are required.

### 5.3.2 Levels in Testing

There are different levels during the process of Testing. The first level in testing is functional testing.

- Functional Testing - This type of black box testing is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. here are different stages for functional testing; these include unit testing, integration testing, system testing and user acceptance testing.

#### 5.3.2.1 Unit Testing

The researcher performed this type of testing before the setup was handed over for test case to the testing team. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.



### **5.3.2.2 Integration Testing**

The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of integration testing and these are:

- Testing Bottom-up Integration testing: this testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
- Top-Down Integration testing: This testing, the highest-level modules are tested first and progressively lower-level.

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic those it will encounter in users' computers, systems and networks. The researcher tested the integrated modules of the system.

### **5.3.2.3 System Testing**

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets software quality standards. The NHPIS was tested and it met all the software quality defined in the Planning Phase. A specialized testing team performs this type of testing. System testing is so important because of the following reasons:

- The application is tested thoroughly to verify if meets the functional and technical specifications.
- The application is tested in an environment, which is very close to the production environment where the application will be deployed.
- System Testing enables us to test, verify and validate both the business requirements as well as the Applications Architecture.

### **5.3.2.4 Regression Testing**

Whenever a change in a software application is made, it is quite possible that other areas within the application have been affected by this change. To verify that a fixed bug has not resulted in another functionality or business rule violation is Regression testing. The intent of regression

testing is to ensure that a change, such as a bug fix did not result in another fault being uncovered in the application. Regression testing is so important because of the following reasons:

- Minimize the gaps in testing when an application with changes made has to be tested.
- Testing the new changes to verify that the change made did not affect any other area of the application.
- Mitigates Risks when regression testing is performed on the application.

The researcher performed a regression test whenever a change was made.

### **5.3.2.5 Acceptance Testing**

This is arguably the most important type of testing as it is conducted by the quality assurance (QA) team. Basically they were gauging whether the application met the intended specifications and satisfied the users' requirements. The QA team had a set of pre written scenarios and test cases that were to test the application. More ideas were shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors or interface gaps, but also to point out any bugs in the application that will result in system crashing or major errors in the application. By performing acceptance tests on an application the QA team can deduce how the application will perform in production. There are also legal and contractual requirements for acceptance of the system.

### **5.3.2.6 Alpha Testing**

This test was performed by the researcher and QA team. Unit testing, integration testing and system testing when combined are known as alpha testing. During this phase, the following was tested in the application:

- Spelling Mistakes
- Broken Links and Cloudy Directions
- The Application was tested on machines with the lowest specification to test loading times and any latency problems.

The spelling mistakes, broken links and cloudy directions were corrected before beta testing

### **5.3.2.7 Beta Testing**

This test was performed after alpha testing had been successfully performed. In beta testing, a sample of the intended audience tests the application. Beta testing is also known as pre-release testing. In this phase, the audience will be testing the following:

- Users will install, run the application and send their feedback to the project team.
- The project team can fix the problems before releasing the software to the actual users.

The second level in testing is non-Functional Testing.

- Non-Functional Testing: Based on testing the application from its non-functional attributes. Non-functional testing of software involves testing the software from the requirements, which are non-functional in nature related. It involves:

### **5.3.2.8 Performance Testing**

It was used to identify any bottlenecks or performance issues rather than finding the bugs in software. There are different causes that contribute in lowering the performance of software:

- Network delay.
- Client side processing.
- Database transaction processing.

The software passed the performance testing. Performance testing is considered as one of the important and mandatory testing type in terms of following aspects:

- Speed (i.e. Response Time, data rendering and accessing)
- Capacity, Stability, and Scalability of the NHPIS application.
- Performance testing can be either qualitative or quantitative testing activity and can be divided into different sub types such as load testing and stress testing.

### **5.3.2.9 Load Testing**

Is a process of testing the behavior of the software by applying the maximum load in terms of software accessing and manipulating large input data. It can be done at both normal and peak

load conditions. This type of testing identifies the maximum capacity of software and its behavior at peak time.

#### **5.3.2.10 Stress Testing**

Stress testing is testing of software behavior under abnormal conditions. Taking away the resources, applying load beyond the actual load limit is stress testing. The main intent is to test the Software by applying the load to the system and taking over the resources used by the software to identify the breaking point.

#### **5.3.2.11 Usability Testing**

It is a black box technique and was used to identify any errors and improvements in the NHPIS software by observing the users through their usage and operation. According to Nielsen, usability can be defined in terms of five factors i.e. efficiency of use, learn-ability, memory-ability, errors/safety, satisfaction. Usability is the quality requirement, which can be measured as the outcome of interactions with a computer system. This requirement can be fulfilled and the end user will be satisfied if the intended goals are achieved effectively with the use of proper resources. Molich in 2000 stated that user-friendly system should fulfill the following five goals i.e. easy to learn, easy to remember, efficient to use, satisfactory to use and easy to understand. In addition to different definitions of usability, there are some standards and quality models and methods, which define the usability in the form of attributes, and sub attributes such as ISO-9126, ISO-9241-11 and ISO-13407.

#### **5.3.2.12 UI vs. Usability Testing**

UI testing involves the testing of graphical user interface (GUI) of the Software. This testing ensures that the GUI should be according to requirements in terms of color, alignment, size and other properties. On the other hand Usability testing ensures that a good and user friendly GUI is designed and is easy to use for the end user. UI testing can be considered as a sub part of usability testing

### **5.3.2.13 Security Testing**

Security testing involves the testing of Software in order to identify any flaws and gaps from security and vulnerability point of view. Following are the main aspects, which Security testing should ensure:

- Confidentiality and integrity.
- Authentication and authorization
- Input checking and validation.
- SQL insertion attacks.
- Session management issues.
- Cross-site scripting attacks.

### **5.3.2.14 Portability Testing**

Portability testing includes the testing of Software with intent that it should be re-useable and can be moved from another Software as well. Following are the strategies that were used for portability testing:

- Transferred installed software from one computer to another.
- Building executable (.war) to run the software on different platforms.

## **5.3.3 System performance and objective evaluation**

NHPIS was tested to check if it met the entire project objectives identified in the system requirements specification stage. Screenshots and their relative messages were used to display the testing result. A test of the specified system objectives was carried and their results were displayed to show how the test procedures were conducted. Screenshots were used to display the testing result.

### **5.3.3.1 System objectives**

1. To enable easy access and manipulation of practitioners' data at provincial and national level.

**Refer to the system**

2. To promote accountability and responsibility on every act, event or use of the system.

created_by	date_created	date_modified	modified_by	voided	voided_by	date_voided	void_reason
ff80818135f342980135f3433b2c0001	2012-03-08 19:02:40	2012-03-08 19:03:06	ff80818135f342980135f3433b2c0001	0	NULL	NULL	NULL

**Fig 5.1 Audit trail on database**

3. To develop tool to track and monitor continuous professional development of the health care practitioners.

**Add Practitioner Qualification**

Qualification: --- Select One---

Training Institution: --- Select One---

Awarding Institution: --- Select One---

Year Acquired:

Extra:

Save Cancel

**Fig 5.2 Add Practitioner qualification**

4. To develop a system that uniquely identifies a practitioner.

D.O.B : 06/04/1994

Gender: Male:  Female:

Marital Status: Married

Nationality: Zimbabwean

Citizenship: Zimbabwean Citizenship

Identity Type: National Id

Identity Number: 07-182879-B83 Duplicate National ID

**Fig 5.3 Checks for duplication**

The system does not allow more than 1 practitioner to have the same National ID or EC Number.

5. To develop a system that enforces same naming standards for reporting consistency.

**Refer to the System Download module**

6. To establish connectivity, functional integration and interoperability of the databases at provincial level with the national instance

Refer when the system has been deployed

7. To enable flexible and accurate report generation

**Refer to the Reports module**

8. To enable automatic synchronization of a provincial data manipulation with the national instance database state.

**Refer to the System operation**

### **5.3.5 Verification and Validation**

In software project management, software testing, and software engineering, verification and validation is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It is referred to as software quality control. It is normally the responsibility of software testers as part of the SDLC.

Verification and validation is not the same thing, although they are often confused. Boehm succinctly expressed the difference between them:

- Validation: Are we building the right product?
- Verification: Are we building the product right?

There are two categories of validation these are

- Server-side validation – The data entered into the system is validated when a user saves into the database
- Client-side validation- The system validates the text box, select fields, and combo boxes using JavaScript.

The researcher performed server side validation in case a browser does not support Java Script.

### 5.3.5.1 Login Validation

The authentication was validated and if a user enters wrong login details the following error message appear:

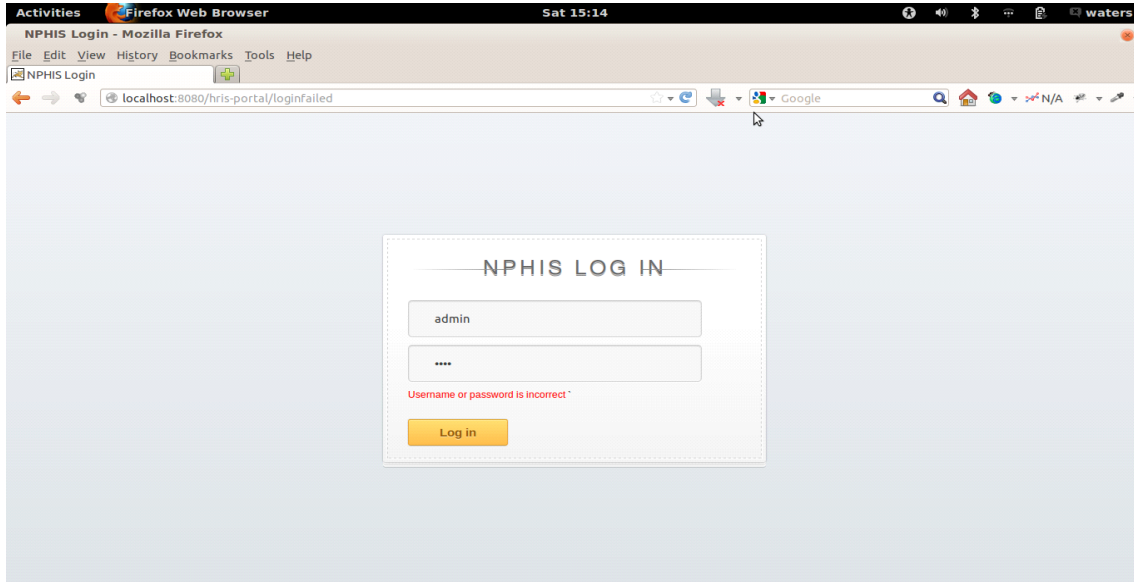


Fig 5.4 Validated Login form

### 5.3.5.2 Null Field Validation

The NHPIS system was validated, every input form was validated. In order to have clean data the inputs should be validated. If a user save a null field the following error messages appear

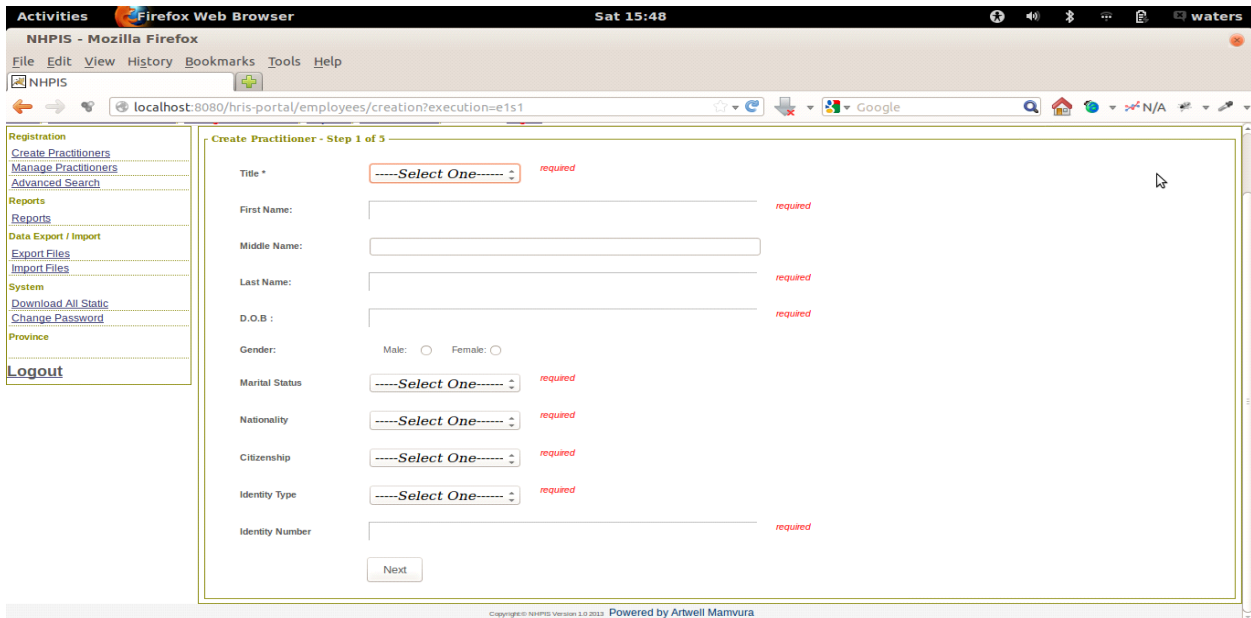


Fig 5.5 Null Field Validation



### 5.3.5.3 Zimbabwean National ID and Passport Number Validation

The system Zimbabwean National ID and Passport Number

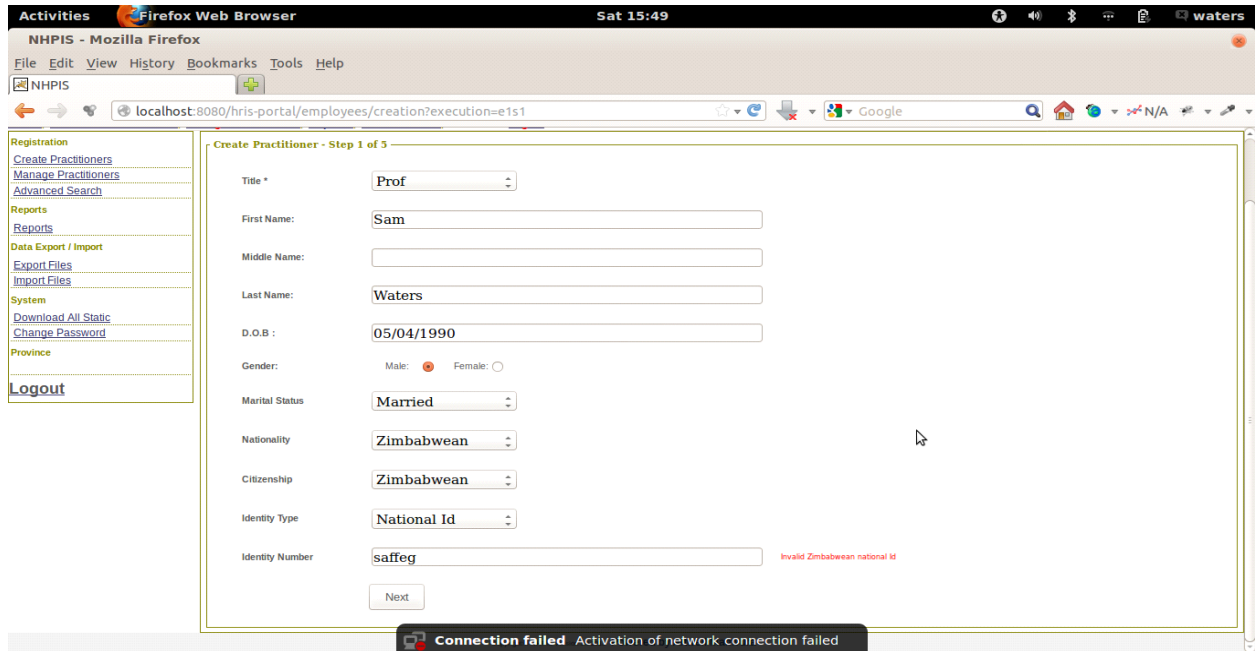


Fig 5.6 National ID and Passport Number Validation

### 5.3.5.4 EC Number validation

A practitioner's EC number is validated

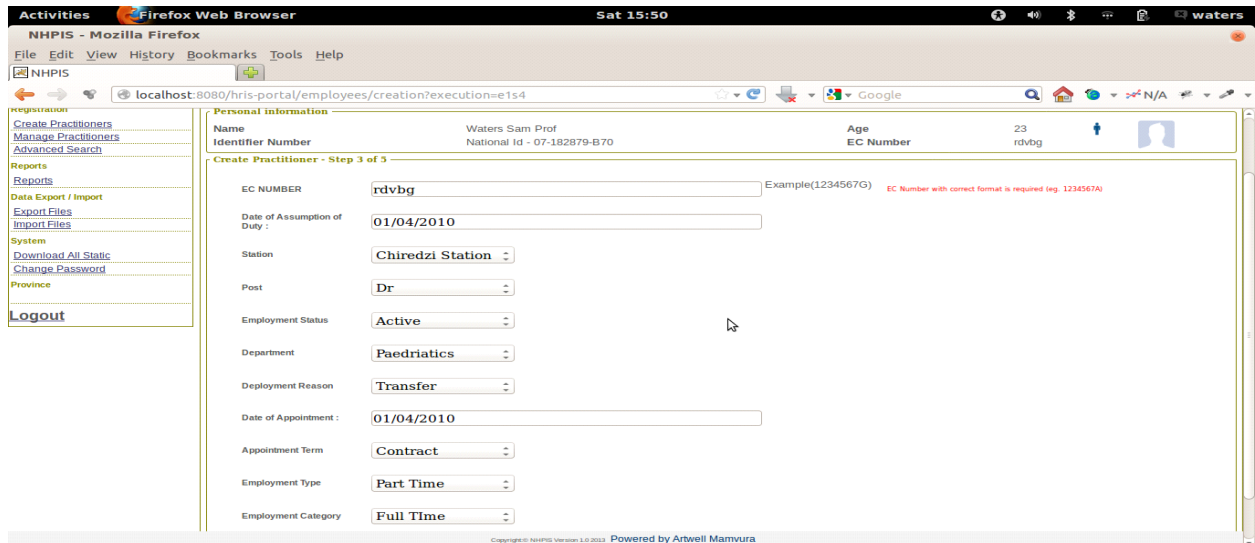


Fig 5.7 EC number validation

## 5.4 Installation

The NHPIS system was deployed in four provinces linked to the national instance, while other six provinces continue using the current manual system for a trial period. The national instance war file was deployed on HITRAC server as in the design phase. Apache Tomcat Server was used to deploy the NHPIS web portal (nhpis.war) and the NHPIS web services portal. The web services portal will allow the provincial systems to consume the services of the national instance. After successful installation at the national server, the system was installed at each of the selected four provinces. If a practitioner is registered, his details are pushed (or synchronized) to the national database.

### 5.4.1 Steps in software installation process

Software installation also encompasses the process of installing the software or application environment required to support the system being developed. In the case of the NHPIS software, installation involves building of the war file and deploying of the system for full production.

For Users who are non-developers of the software the following software need to be installed:

- Install Linux OS on the machine (Redhat, Ubuntu or Fedora latest versions)
- Open terminal and install apache2 and php5
- Install Apache Tomcat Server software on the web server. Latest version 7.0.39
- Install MySQL server and Phpmyadmin on the server as well for database management.
- Open the dispatcher-servlet.xml file, change the national instance to false when at province and to true when at national instance. Build the war file and deploy on Apache Tomcat
- Restore the database of the system in linux (if a user is in Linux type `mysql -uUsername -pPassword databaseName < /FolderWithSql File/databasename.sql`)
- Type the URL `http://serverip:8080/nhpis-portal` that is `http://192.168.1.22:8080/nhpis-portal` and `http:// 192.168.1.22:8080/nhpis-webservices-portal`
- Run the system and set the systems properties, when at provincial level set the host IP address to and `http://nationalServerIPAddress:8080/nhpis-webservices-portal` and set the minimum age of practitioners to 16 years and maximum retiring age to 65 years

- For the first time if the database is empty the default username is **admin** and the default password is **a**.
- Login and change the password.

For developers install:

- JDK 1.6
- Netbeans (7.2 or later)
- MySQL Workbench
- Configure Apache Tomcat Server as a service. To start the tomcat service just type in terminal (sudo tomcat service start) and to stop (sudo tomcat service stop)
- Redmine for tracking the NHPIS software versions
- GIT Server for pulling, cloning, pushing source code when working as a team, Just type sudo apt-get install git-server
- Build the project from Folder containing the source code and run the system in Netbeans.
- Restore the mysql database (Linux type mysql -uUsername -pPassword databaseName < /FolderWithSql File/databasename.sql)

## 5.4.2 User training

User training is the process of introducing the users to the new system highlighting on how to use the new system, Bentley et al (1994). For the NHPIS software, users that were trained include Health information Officers (HIO), MoHCW management, and Hitrac Administrator and maintenance programmers. **Refer to Appendix A** for User Manual

### 5.4.2.1 Training plan

The user training was scheduled for HITRAC staff members starting with the management and then with the other system developers since they would be responsible for troubleshooting or debugging and helping the users with challenges that may be faced during the system operation as well as maintenance and further development. The researcher of the NHPIS software attended all the training sessions. The following users were trained:

- HITRAC Administrators: The administrators were trained on how to use the system to upload health practitioners' data from provincial level into the system, adding static data into the system, managing practitioner details, backing up of the databases
- HIO also the Provincial administrators: The training was on how to import practitioner data from provinces in case its offline, how to download the static data, how manage the provincial databases and how to create a practitioner in the system, and how to

**Table 5.1 HITRAC Administrators training**

Venue	HITRAC Organization
Users	Administrators and System Developers
Training Scope	Administrator module, NHPIS functionality, NHPIS database backup and Report generation
Requirements	A single computer and an projector for screen elaboration
Trainer	System Developer (Artwell Mamvura)

**Table 5.2 Provincial Administrators**

Venue	Mandel Training Centre, Harare Zimbabwe
Users	Provincial Administrators
Training Scope	Health Practitioner Data capturing, Exporting of practitioner data, System functionality overview, System Trouble shooting, Handling errors, Documenting changes, downloading static data, exporting practitioner files, viewing reports, User creation, Database backup
Requirements	Four computers and, a projector.
Trainer	System Developer (Artwell Mamvura) and the HITRAC administrator and one System Developer and the respective Provincial Administrator

### **5.4.3 System Conversion**

System conversion is a technical process where a new system (NHPIS system) replaces the current system. After the training and operational environment is setup the new system can now replace the current or current system. There are several methods that can be used to install the new system and these include the following direct conversion, pilot conversion, phased conversion and parallel conversion.

#### **5.4.3.1 Direct Conversion**

This is the complete replacement of the current or current system by the new system. It is a risky approach and requires comprehensive system testing and training. The provinces that were using the current system will immediately function under the new system. A direct conversion may be the only option if the current and new systems cannot co-exist in any form. If used, it should not be done in a peak period where transition hiccups will upset the organization. This is a least expensive method among all four because it can occur in the quickest time. However, it involves high risk of data loss as the organization cannot revert to the current system as a backup option. The researcher did not use the direct conversion method.

#### **5.4.3.2 Pilot conversion**

In this type of run, the new system is run with the data from one or more of the previous periods for the whole or part of the system. The results are compared with the current system results. It is less expensive and risky than parallel run approach. This strategy builds the confidence and the errors are traced easily without affecting the operations, however, the system was not deployed using the pilot conversion method.

#### **5.4.3.3 Phased conversion**

In a phased conversion, a new system will be installed in phases rather than a complete transformation at one time. The phased approach takes the conversion one-step at a time. Moreover, at every milestone one has to instruct the employees and other users. The current system is taken over by the new system in predefined steps until it is totally abounded. The actual installation of the new system will be done in several ways, per module or per product and

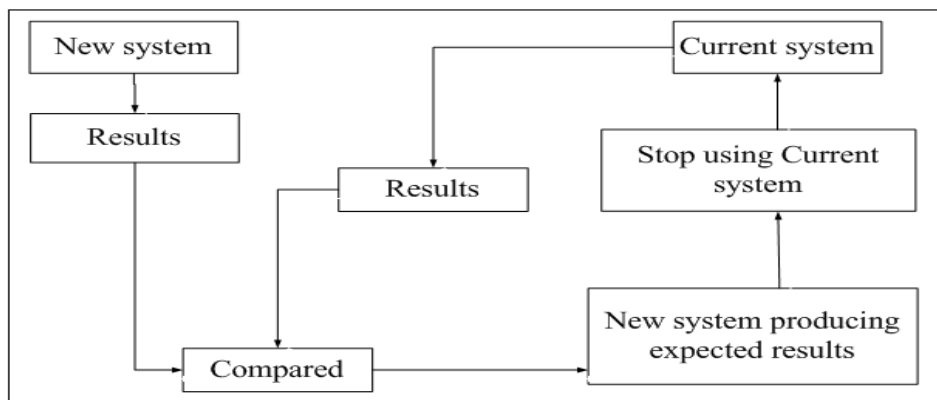
several instances can be carried out. This gives the users the time to cope with the changes caused by the system. The conversion will be done in parts. Time is available for adjustments. Negative influences that arise at the start are less critical and no ‘catch-up’ period is needed. The NHPIS system was not deployed using the phased system conversion method.

#### 5.4.3.4 Parallel conversion

In parallel conversion the both the systems are run or used that is computerized and manual, are executed simultaneously for certain defined period. The same data is processed by both the systems. This strategy is less risky but more expensive because of the following:

- Manual results can be compared with the results of the computerized system.
- The operational work is doubled.
- Failure of the computerized system at the early stage does not affect the working of the organization, because the manual system continues to work, as it used to do.

The NHPIS system (new system) was deployed in four provinces in parallel with the manual system (current system). According to the schedule by 2014 all the provinces would have been connected. The NHPIS system will basically work as per the diagram below:



**Fig 5.4 Parallel Conversion**

### 5.5 Maintenance

Maintenance is the last stage in the system development life cycle and, consequently, is affected by everything that happens in the previous stages. Errors made during the analysis and design

stages can significantly impact maintenance. It relies on the documentation created during the analysis, design, testing and implementation stages, and the system maintenance life cycle parallels the system development life cycle. Maintenance begins when the system is released and continues for the life of the system. Over a period of years, it is not unusual for the cost of maintaining a system to significantly exceed the cost of developing it, so a primary objective is controlling maintenance costs. When the cost of maintaining and operating an obsolete or inefficient system exceeds the cost of replacing it, the system life cycle ends and a new life cycle begins. The key to controlling maintenance costs is to design systems that are easy to change, so the link between development and maintenance is very strong.

Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environments. If a major change to a system is needed, a new project may have to be set up to carry out the change. The new project will then proceed through all the above life cycle phases. There is much more to maintenance than fixing bugs. The categories suggested by Swanson and extended by Reutter are widely accepted. The types of maintenance include:

- Corrective maintenance
- Adaptive maintenance
- Perfective maintenance
- Preventive maintenance

#### **5.5.1. Corrective maintenance**

Corrective maintenance refers to changes made to repair defects in the design, coding, or implementation of the system. Most corrective maintenance problems surface soon after installation. When corrective maintenance problems surface, they are typically urgent and need to be resolved to continue normal business activities. Corrective maintenance adds little or no value to an organization; it simply focuses on removing defects from an existing system without adding new functionality.

### **5.5.2 Adaptive maintenance**

Adaptive maintenance involves making changes to an information system to evolve its functionality to changing business needs or to migrate to a different operating environment. It adds enhancements to an operational system, such as new features, increased capability, or changes that improve efficiency or maintainability. An adaptive maintenance project is like a mini-SDLC project, and that adaptive maintenance can be even more difficult than new systems development because the enhancements must work within the constraints of an existing system. Adaptive maintenance is usually less urgent than corrective maintenance because business and technical changes typically occur over some period. Adaptive maintenance is generally a small part of an organization's maintenance effort, but does add Value to the organization.

### **5.5.3 Perfective maintenance**

Perfective maintenance involves making enhancements to improve processing performance, interface usability, or to add desired, but not necessarily required, system features. The objective of perfective maintenance is to improve response time, system efficiency, reliability, or maintainability. During system operation, changes in user activity or data pattern can cause a decline in efficiency, and perfective maintenance might be needed to restore performance. Usually, the IT department initiates the perfective maintenance work, while users normally request the corrective and adaptive maintenance work.

### **5.5.4 Preventive maintenance**

Preventive maintenance involves changes made to a system to reduce the chance of future system failure. An example of preventive maintenance might be to increase the number of records that a system can process far beyond what is currently needed or to generalize how a system sends report information to a printer so that so that the system can adapt to changes in printer technology. Preventive maintenance is less likely to occur as compared to corrective maintenance.

Finally note that over the life of a system, corrective maintenance is most likely to occur after system installation or after major system changes. This means that adaptive, perfective, and



preventive maintenance activities can lead to corrective maintenance activities if not carefully designed and implemented.

### **5.5.5 Managing maintenance**

Maintenance is expensive. The elements of a system often interact in unexpected ways, and ripple effects (unexpected bugs or new errors caused by a change intended to fix an initial problem) can be devastating. Sometimes, apparently unrelated maintenance problems are tightly linked. Consequently, maintenance must be carefully managed. Formal maintenance procedures are the key to managing maintenance.

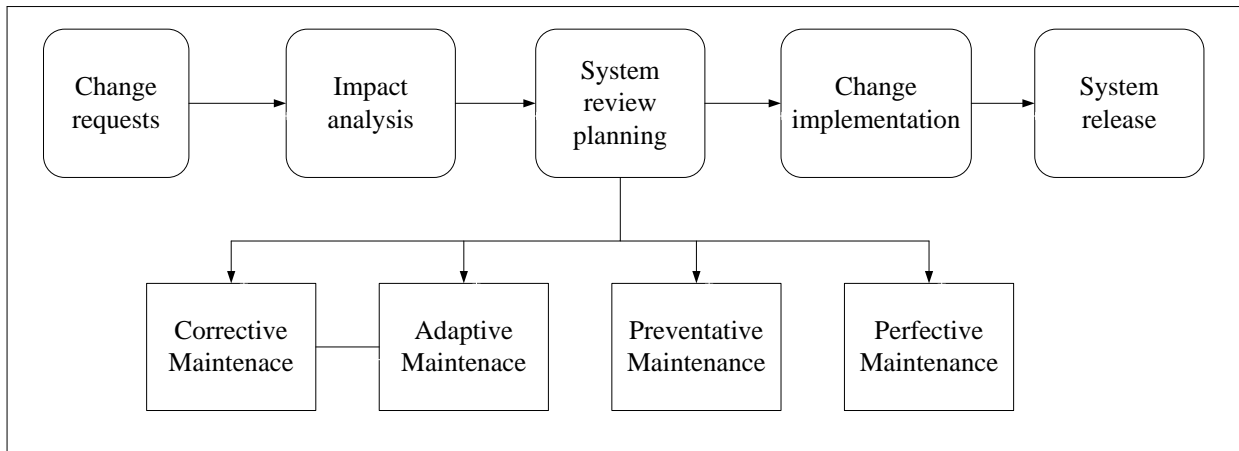
#### **5.5.5.1 Maintenance Team**

A maintenance team is a group of systems analysts and programmers with strong technical, interpersonal, business, communications, and problem-solving skills that use both analysis and synthesis. Analysis is the process of examining the whole in order to learn about the individual elements. Synthesis is the process of studying the parts to understand the overall system. ITS managers often divide systems analysts and programmers into two groups: one group performs new systems development and the other group handles all maintenance. The advantage is that the maintenance group develops strong support skills. Some organizations use a more flexible approach and assign IS staff members to various projects as they occur. By integrating development and support work, the people developing the system also assume responsibility of maintaining it.

### **5.5.6 The system maintenance life cycle**

The system maintenance life cycle is similar to the system development life cycle. Configuration management, the process of managing and controlling changes to a system, defines a context or methodology, including formal procedures for requesting, evaluating, and implementing changes. Evaluation analysis (a phase that parallels the information gathering and problem definition stage of the system development life cycle) is used to assess the impact of a proposed change. The objective is to identify and document the applications, programs, routines, and other components that must be modified, the likely impact of the change on normal operations, and the

time, cost, and other resources required to implement the change. Next, the change is analyzed, designed, coded, and tested. After the work is done, the proposed system or change is released.



**Fig 5.5 System Maintenance Life Cycle**

### **5.5.7 Prioritization**

Given the budgetary and resource constraints imposed on the system maintenance process, it is not unusual to have a backlog of change requests. Some organizations rely on simple first-in-first-out or (less frequently) last-in-first-out schemes to prioritize the requests. Other organizations prioritize the requests based on a preliminary evaluation analysis. Note that a sound evaluation analysis is the key to deciding if maintenance or new system development is needed to solve the problem.

### **5.5.8 Fire fighting**

Some maintenance problems require an immediate response. Following a system crash, a major integrity threat, such fire fighting activities are relatively rare, however, and emergency patches should be formally incorporated into a subsequent NHPIS release.

### **5.5.9 Standards and quality assurance**

Many system developers or system maintainers view all maintenance as fire fighting. Consequently, in their rush to get the job done, they sometimes ignore or disregard quality

assurance and other standards. It is important that all changes be made in a consistent manner. Consequently, such standards as code efficiency targets, fault tolerance rates, operational sequence optimization guidelines, and expected performance norms must be established and enforced, and even true fire fighting activities must be brought up to standards after the fact.

## **5.6 Conclusion**

When the implementation phase concludes, the system begins operating and continues to do so until the organization determines it has outlived its usefulness and starts planning for a replacement system. The approval of the Implementation Phase deliverables and the completion of the Implementation project status review, and the execution of project closeout activities, signify the end of the Implementation Phase.

## REFERENCES

### Books

1. Alain Abran, James W. Moore; (2005), Guide to the software engineering body of knowledge, Los Alamitos, CA: IEEE Computer Society Press
2. Bennatan, E. M., (1995), On Time, Within Budget Software Project Management Practices and Techniques, Wiley.
3. Birrell, N. D. and Ould, M.A., A Practical Handbook for Software Development, Cambridge University Press, Cambridge, UK, 1985.
4. Cavusoglu H, Mishra B and Raghunathan S, The Effect of Internet Security Breach, (2002) Univ.of Texas at Dallas.
5. Clifton, D. S. and Fyffe, Project Feasibility Analysis: A Guide to Profitable New Ventures, (1977), John Wiley & Sons, New York.
6. Conger, S Software Engineering (1994) Belmont California: Wadsworth, Inc
7. Davis C, Alan M, 201 Principles of Software Development, (1995), New York, NY: McGraw-Hill, Inc.
8. Fournier, R., Practical Guide to Structured System Development and Maintenance, Yourdon, Englewood Cliffs, NJ, 1991.
9. Garmus D, David H (1996), Measuring the Software Process: A Practical Guide to Functional Measurements, Upper Saddle River, NJ: Prentice-Hall Inc.
10. Hoffer, J., George, J., & Valacich, J. (2006), Modern systems analysis and design 6th. Prentice Hall: U.S.A.
11. Hoglund G and McGraw G, Exploiting Software, (2004), Addison-Wesley.
12. Howard M and LaBlanc D, Writing Secure Code, (2003), Microsoft Press.
13. Jeffrey L Whitten and Lonnie D Bentley, (2001), System Analysis and Design Methods (7<sup>th</sup> Edition), Prentice Hall of India
14. Jones, H Handbook of team design (1998). New York McGraw-Hill
15. Kendell K.E and Kendell J.E, (2002), Systems Analysis and Design, Pearson Education Asia

16. Kotonya, G. and Sommerville, I, (1998), Requirements Engineering: Processes and Techniques, UK: John Wiley and Sons.
17. Levin, Mark Sh., Composite Systems Decisions (2006), Springer, New York,
18. Maier, Mark W., and Rechtin, Eberhardt, The Art of Systems Architecting, Second Edition, (2000), CRC Press, Boca Raton.
19. McGraw G, "Software Security," IEEE Security & Privacy, (2004), Prentice Hall
20. Pierce, S Software System engineering: a first course (1992) Wilsonville, Oregon: Franklin, Beedle & Associates, Inc
21. Rajaraman V, (2004), Analysis and Design of Information Systems, Prentice Hall of India
22. Reutter, J., Maintenance is a management problem and a programmer's opportunity, AFIPS Conf. Proc. 1981 Natl. Comput.
23. Saltzer, J.H, et al., End-to-End arguments in Systems Design in: ACM Transactions in Computer Systems, (1984), Springer, New York,
24. Schach, S Classical and object oriented software engineering with UML & C++ 4<sup>th</sup> Edition (1999) New York McGraw- Hill.
25. Sindre G and Opdahl A.L, Technology of Object-Oriented Languages and Systems (2000), IEEE CS Press.
26. Steward, D V, (1987), Software engineering with systems analysis and design, Belmont California: Wadsworth, Inc
27. Swanson, E., (2004), The dimensions of maintenance, San Francisco.
28. Taggart W.M, and Silbey V, (1986), Information Systems: People and Computers in Organizations, Allyn and Bacon, Inc., Boston.
29. Tait, P. and Vessey, I. (1988), The Effect of User Involvement on System Success: A Contingency Approach. MIS Quarterly.
30. Ulrich, Karl T. and Eppinger, Steven D., Product Design and Development, Second Edition, (2000), Irwin McGraw-Hill, Boston,
31. Valacich, J.S., George, J. F., and Hoffer, J.A. (2009) Essentials of System Analysis and Design 4<sup>th</sup> Ed., Prentice Hall, Upper Saddle River, NJ.
32. Viega J and McGraw G, Building Secure Software, (2001), McGraw-Hill, Inc.

33. West Churchman C, the Design of Inquiring Systems: Basic Concepts of Systems and Organization (1971), Basic Books, New York.
34. Whitten, J. L., Bentley, L. D., and Barlow, V. M. (1994) Systems Analysis and Design Methods 3<sup>rd</sup> Ed. Richard D. Irwin, Inc., Burr Ridge, IL.
35. Whitten, J. L., Bentley, L. D., and Dittman, K. C. (2004) Systems Analysis and Design Methods 6<sup>th</sup> Ed., McGraw Hill Irwin, Boston.
36. Whitten, J. L., Bentley, L. D., and Ho, T.I.M. (1986) Systems Analysis and Design, Times Mirror/Mosby College Publishing, St. Louis.
37. Whitten, J.L., and Bentley, L.D. (2008) Introduction to Systems Analysis and Design 1<sup>st</sup> Ed. McGraw-Hill, Boston.
38. Wiegers, Karl E. (2003), Software Requirements (2nd ed.), Microsoft Press
39. William S. Davis, David C, (1998), The Information System Consultant's Handbook: Systems Analysis and Design by. Yen CRC Press, CRC Press LLC

### **Journals**

1. Available at: <http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1062&context=eispapers>. Accessed Feb 22, 2013.
2. Available at: <http://softwarefeasibilitystudy.blogspot.com/2009/07/feasibility-study-software-engineering.html> Accessed 25/02/13
3. Jones N. Telematics systems in primary care. Available at: <http://www.sjsu.edu/faculty/watkins/cba.htm>. Accessed Feb 22 2013.
4. Heard S, Grivel T, Schloeffel P and Doust J, The benefits and difficulties of introducing a national approach to electronic health records in Australia.
5. Software testing [http://www.tutorialspoint.com/software\\_testing/index.htm](http://www.tutorialspoint.com/software_testing/index.htm)  
[http://www.tutorialspoint.com/software\\_testing/testing\\_iso\\_standards.htm](http://www.tutorialspoint.com/software_testing/testing_iso_standards.htm) ACCESSED 2/04/2013,
6. Mohammad, R. (2006), Dilemma between the structured and object-oriented approaches to systems analysis and design. The Journal of computer information systems: 32-42. Accessed 15 March 2013
7. The Royal College of General Practitioners—information sheet, general practice computerization.

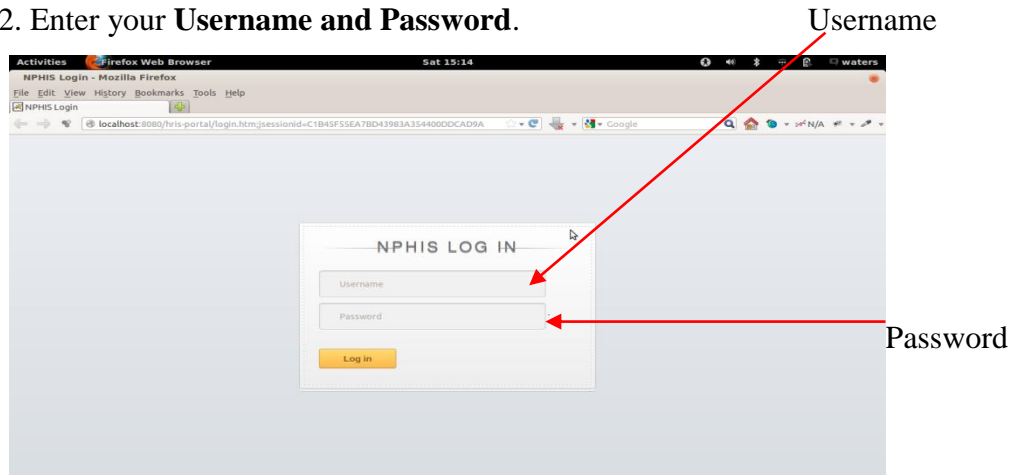
# APPENDICES

## Appendix A: User Manual

The user manual was designed to familiarize the users with the necessary knowledge of the NHPIS software without any problems.

### Getting Started

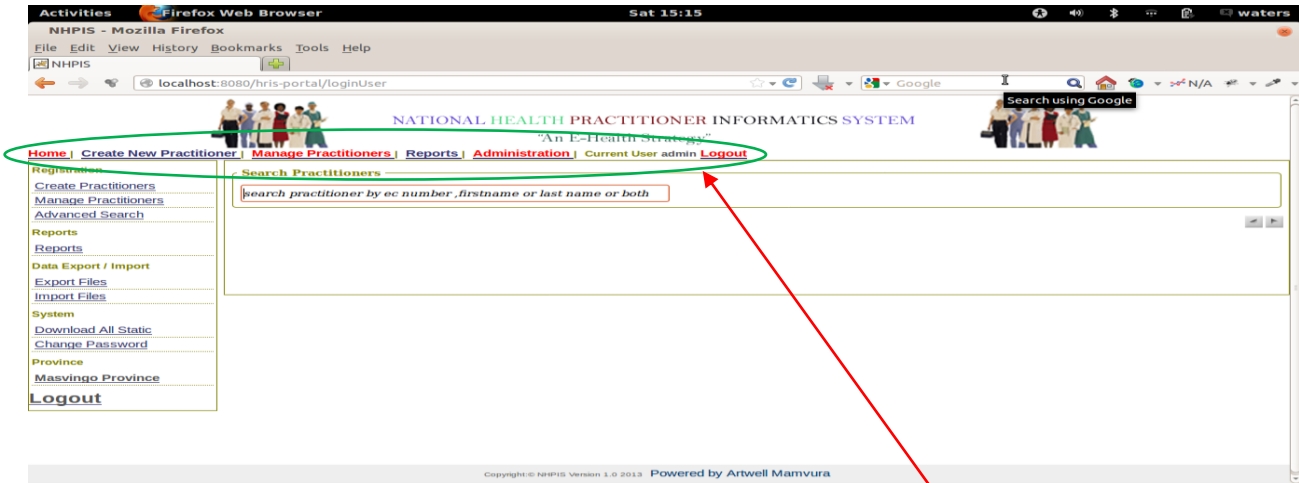
1. Go to <http://localhost:8080/hris-portal/>
2. Enter your Username and Password.



**Fig A1 Login Page**

3. Click Login Button

4. The following screen should appear if Login details are correct:



**Fig A2 Main Menu for the User at Province**

## 5. Searching for a Practitioner

Type the name or EC Number of the practitioner

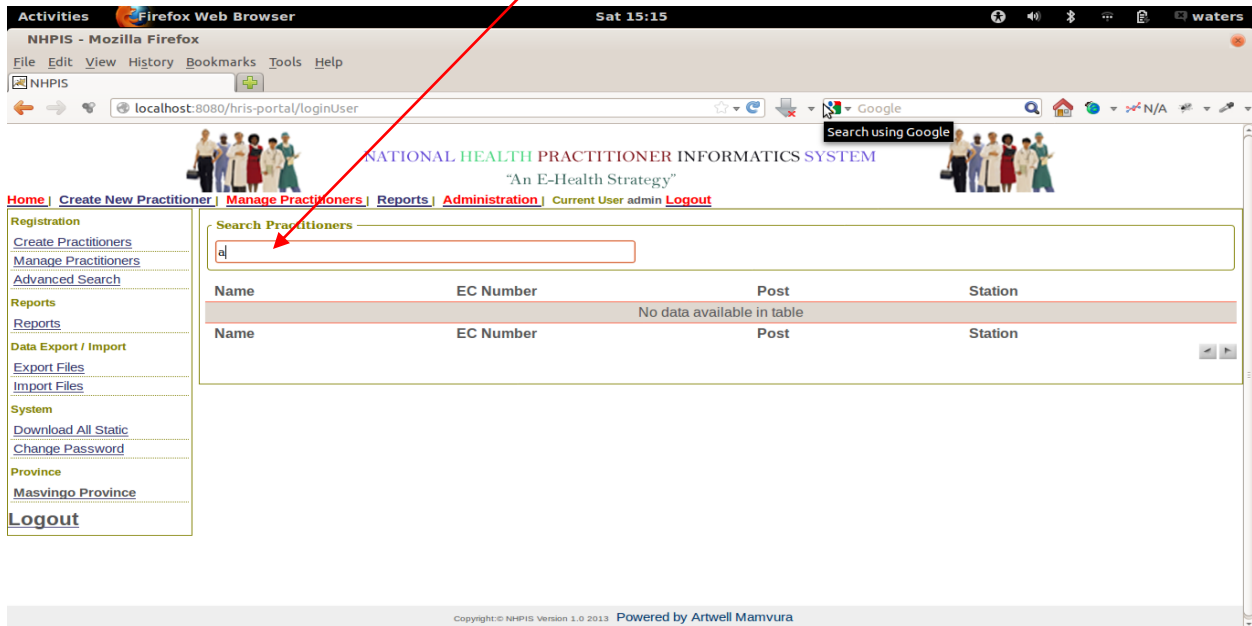


Fig A3 Search Practitioner

## 6. If a user Clicks on the Name of a practitioner the his or her summary is listed

Summary

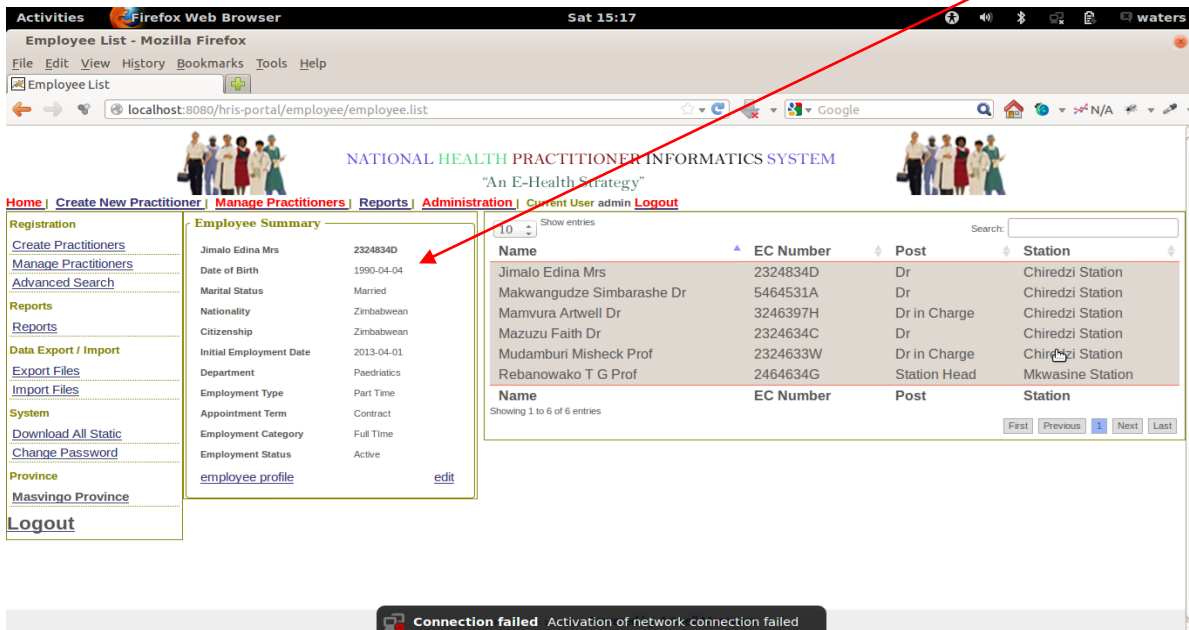
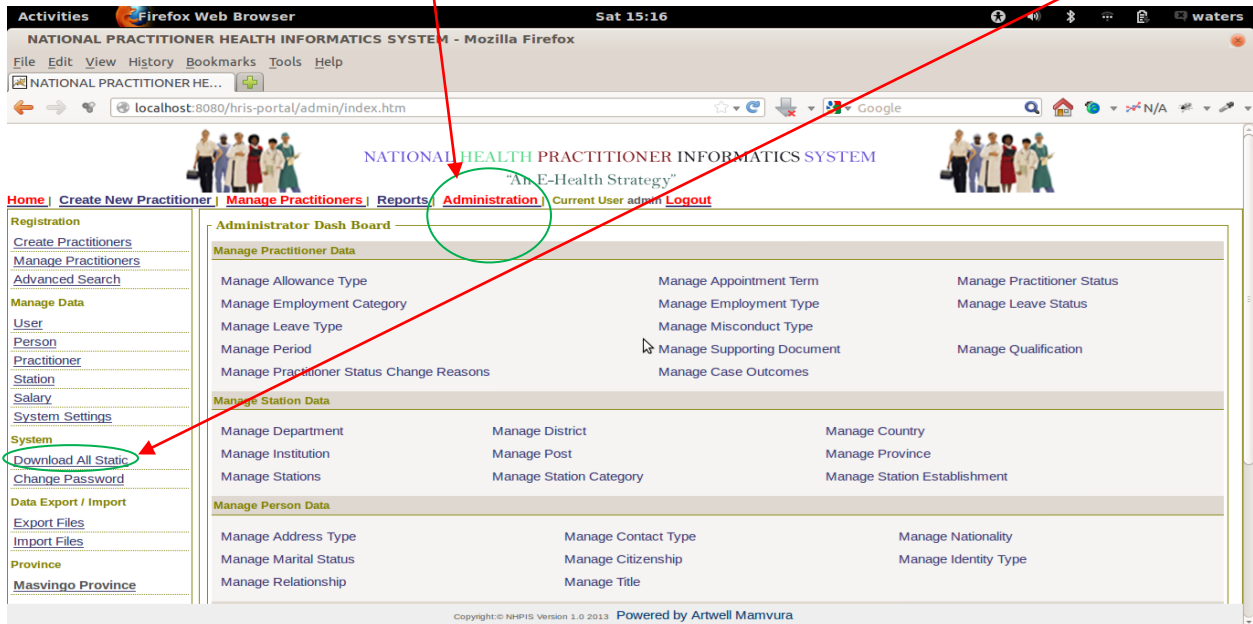


Fig A4 Summary Details



## 7. Downloading static data

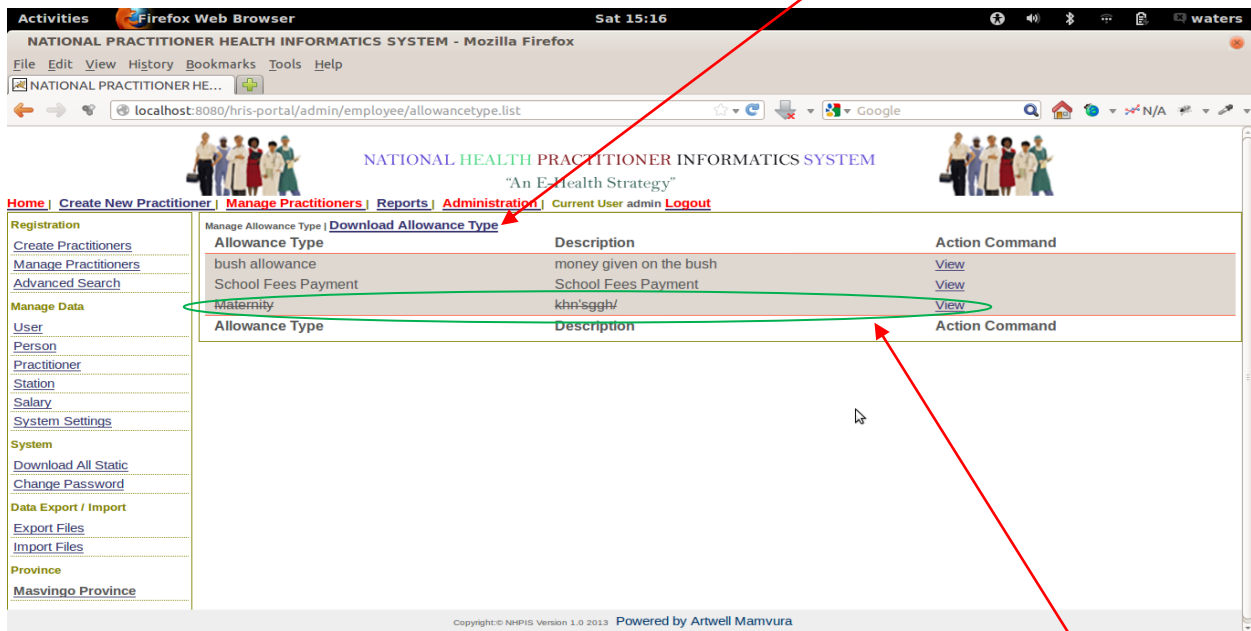
Click the Admin link to navigate to the administrator's panel and Download all static data



**Fig A5 Administrator page**

## 8. Download a specific static data

Download Link



**Fig A6 Download Static Data**

Retired Record

9. An administrator can download a practitioner's data

Download Practitioner File

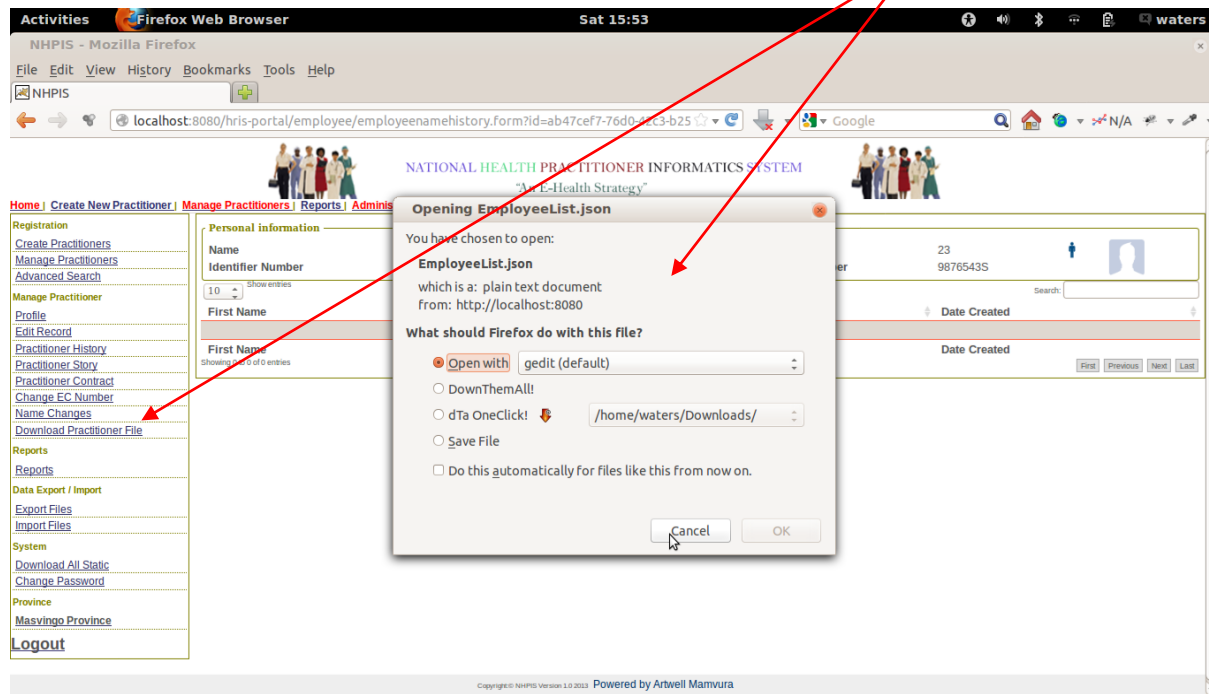


Fig A7 Download practitioner details

10. The HIO or data capturer adds a practitioner into the system by clicking the create practitioner link and the Form below is shown. Fill the details and click Next

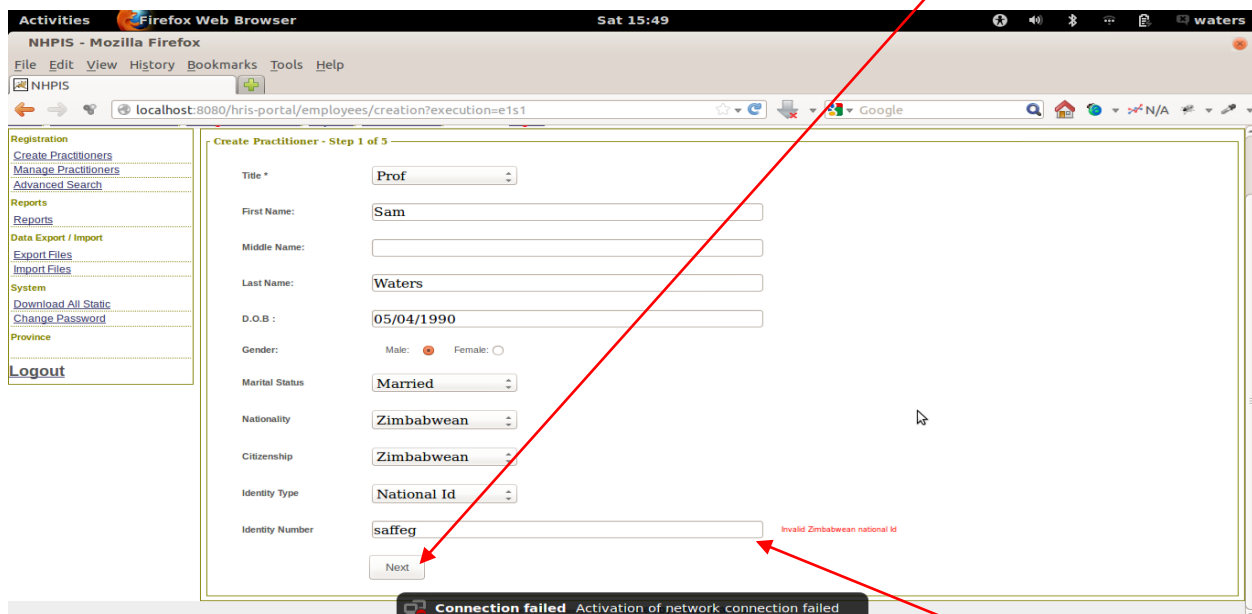


Fig A8 Create Practitioner Step 1 of 5 form

The Zimbabwe National ID is validated

11. Step 2 of 5 of Create Practitioner form Add Contact Form Add Address Add Qualification

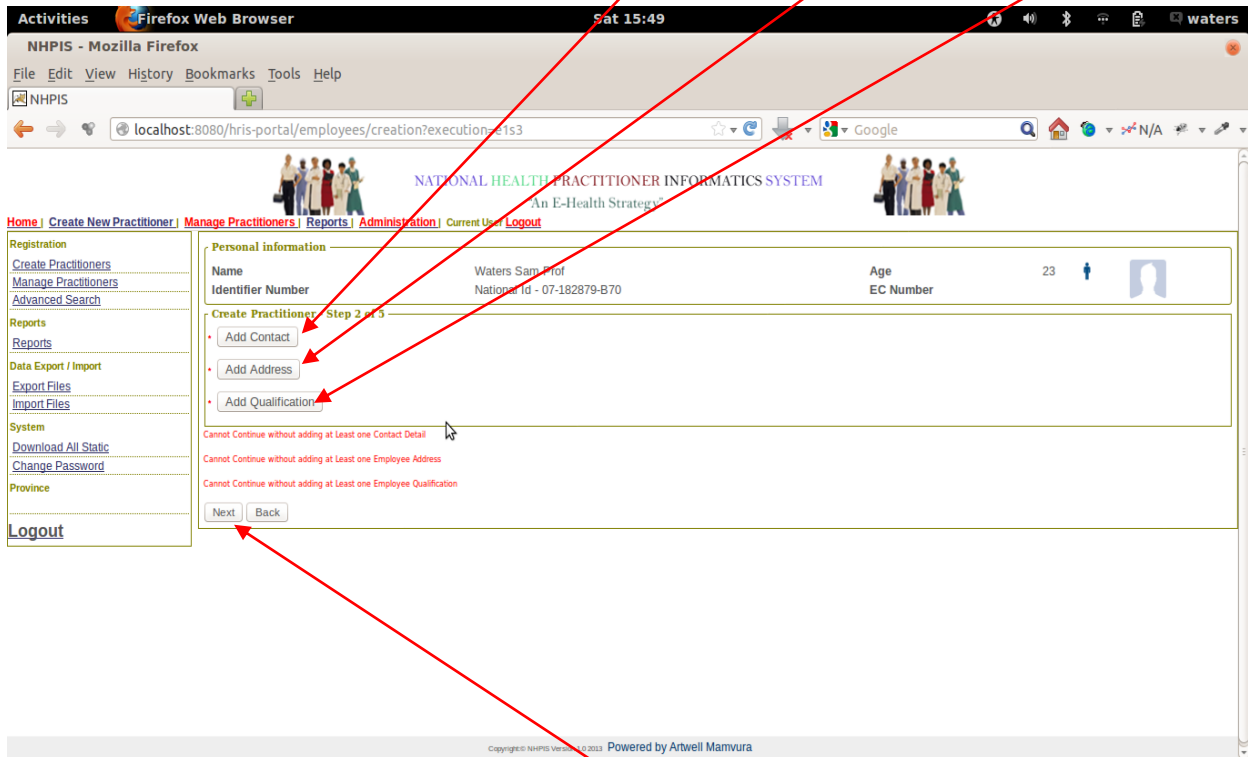


Fig A9 Step 2 of creating a Practitioner

12. Fill the Contact, Address and qualification and click the Next

Validated

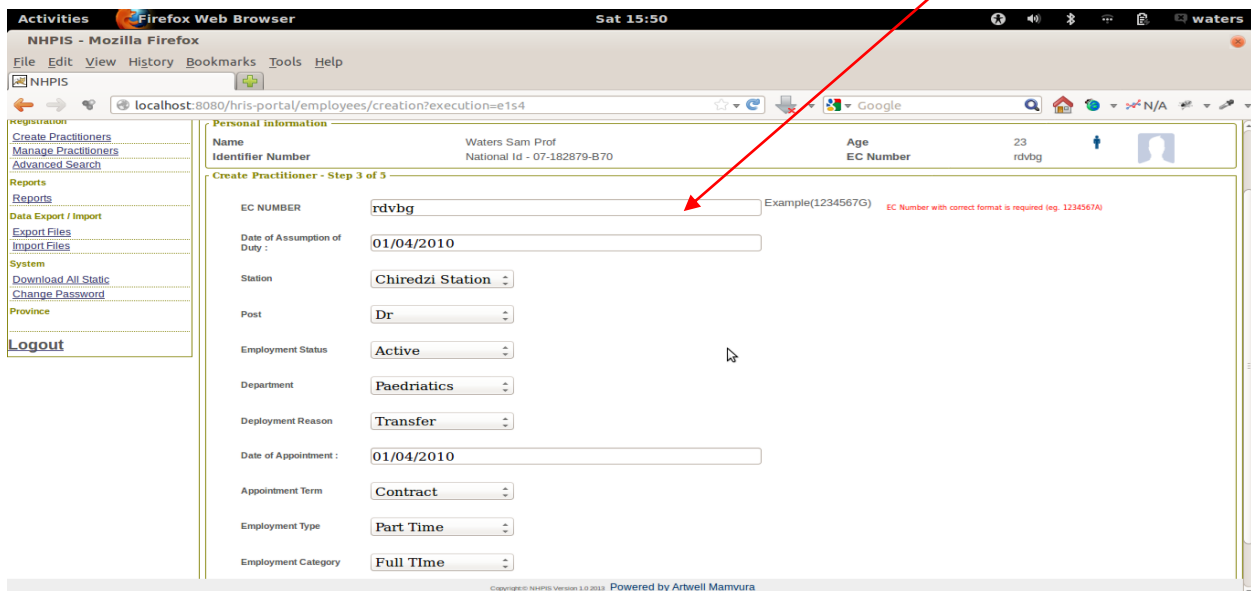


Fig A10 Step 3 of 5 of creating a practitioner

13. Step 4 of 5 of creating a practitioner Checklist box Tick the checklists and click Next

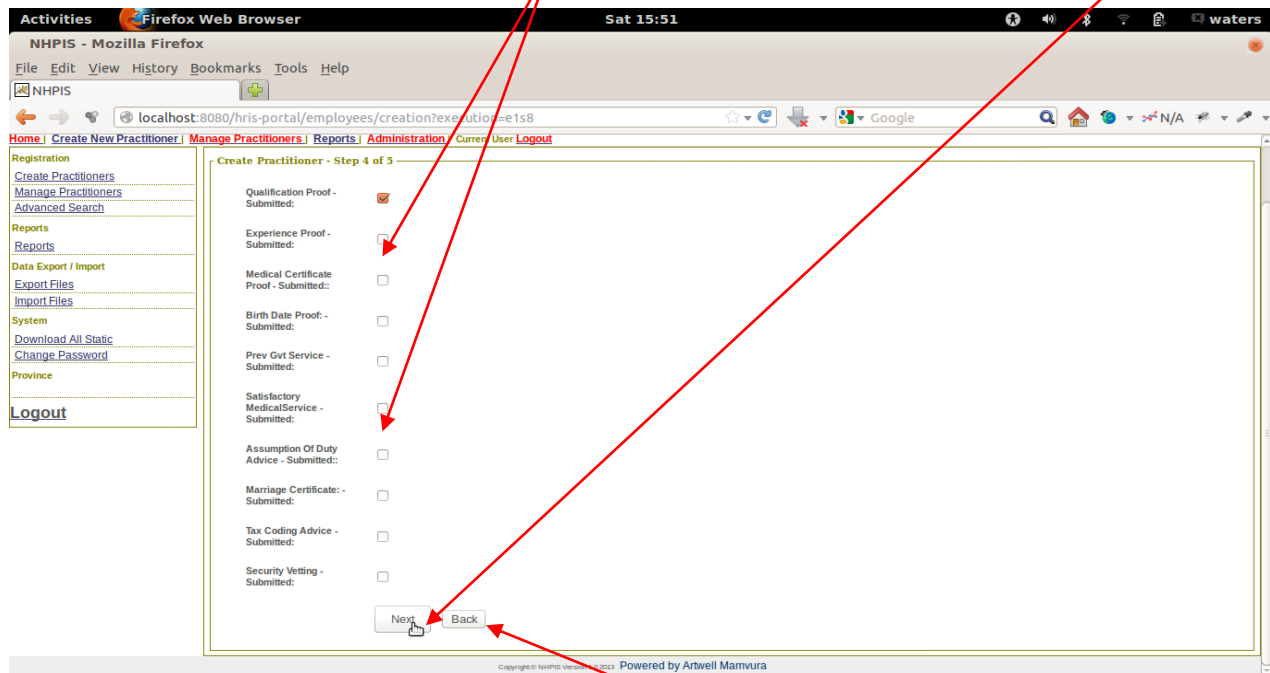


Fig A11 Step 4 of 5 Create Practitioner

14. If the data capturer feels he has made a mistake he should click the Back button

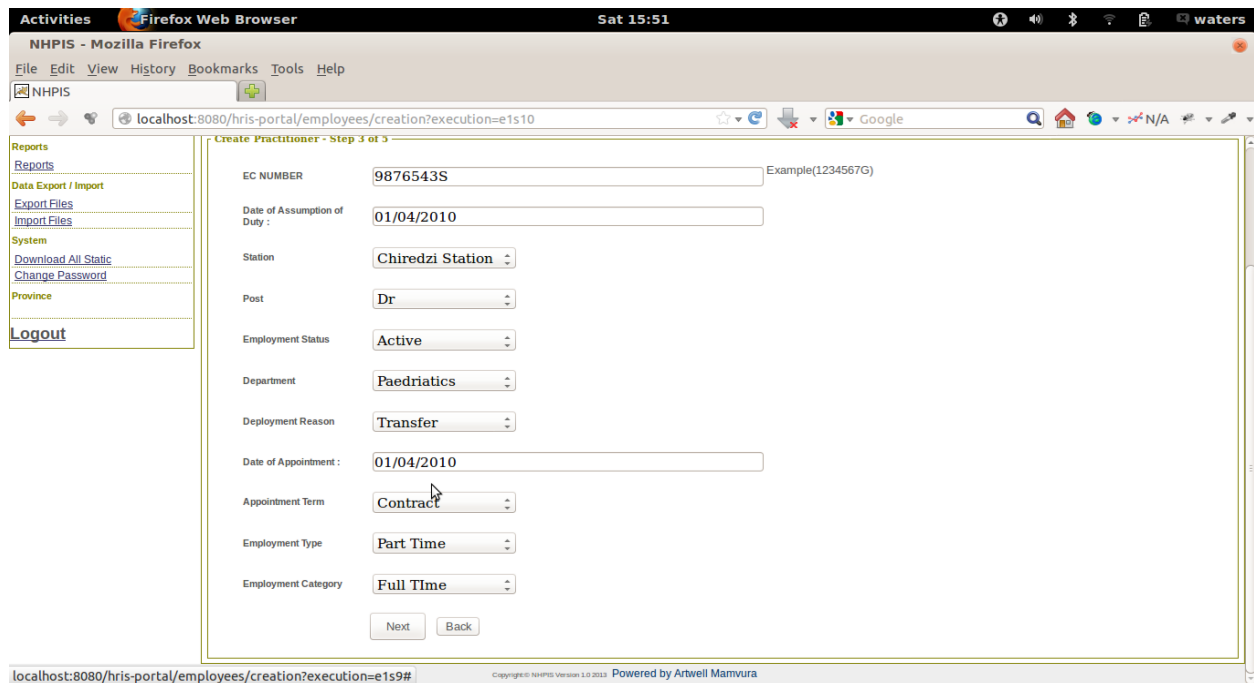


Fig A 12 Create Practitioner Step 3 of 5

15. Step 5 of 5 is Confirming what the user entered on the form before saving if there is need for a change just Update or click the Back Button or Cancel Button

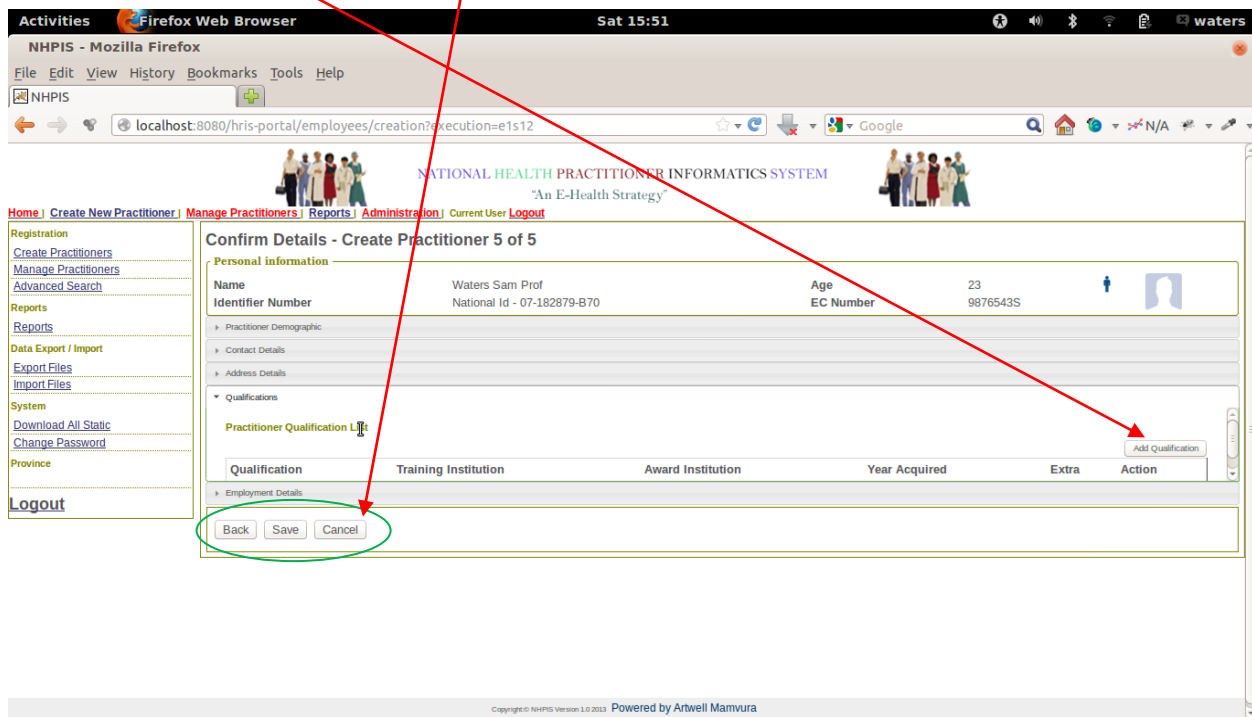


Fig A13 Create Practitioner Step 5 of 5

16. After saving a Practitioner's data the practitioner's profile is display as below:

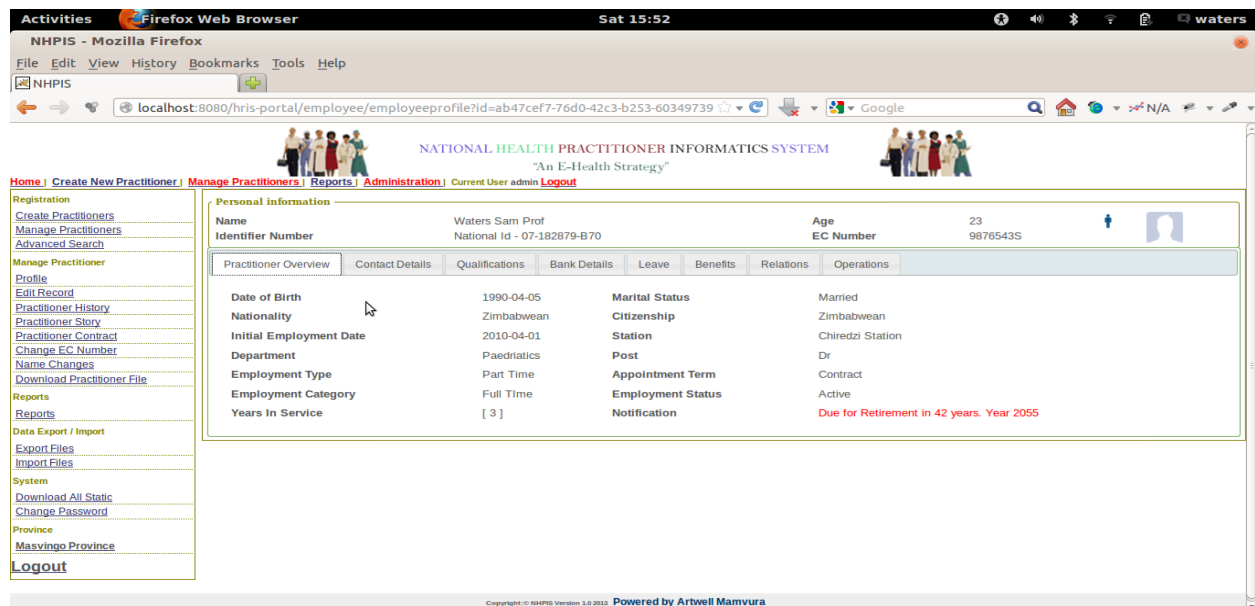


Fig A14 Practitioner Profile

17. Operations on practitioner data      Regrade, Transfer, Promote, add disciplinary e.tc

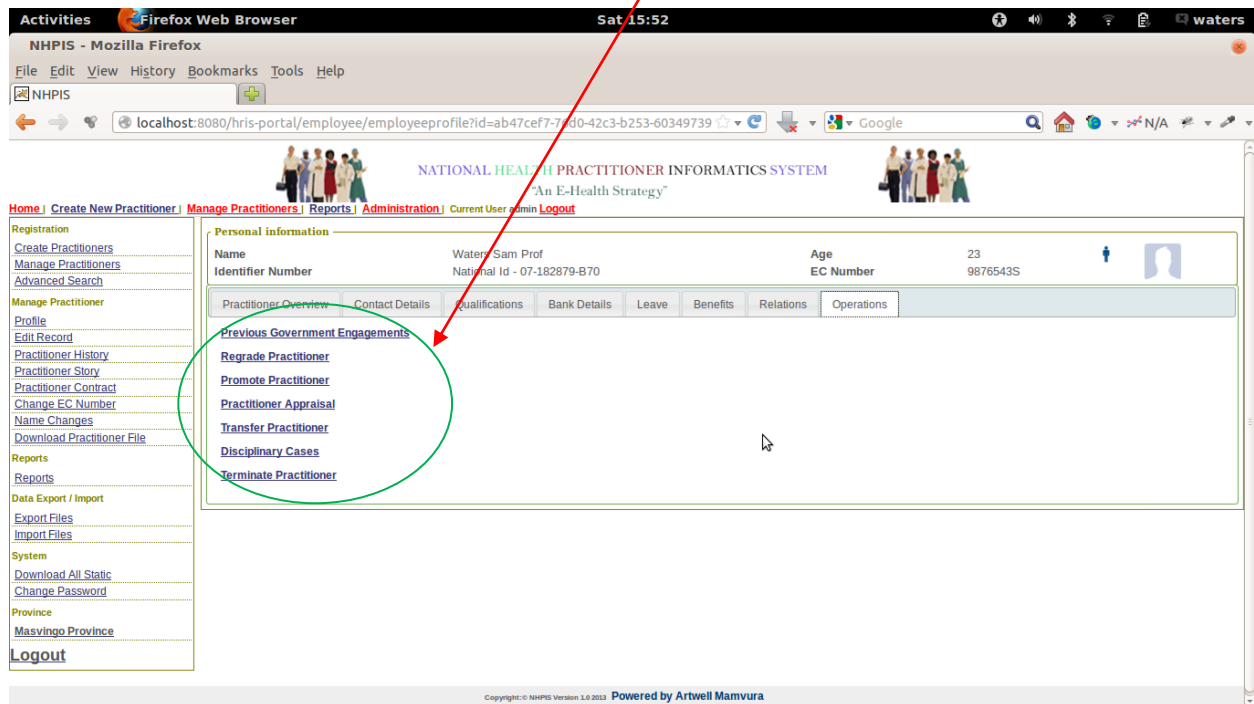


Fig A15 Operations on practitioner Data

18. Reports: User can click the Report link to view the reports

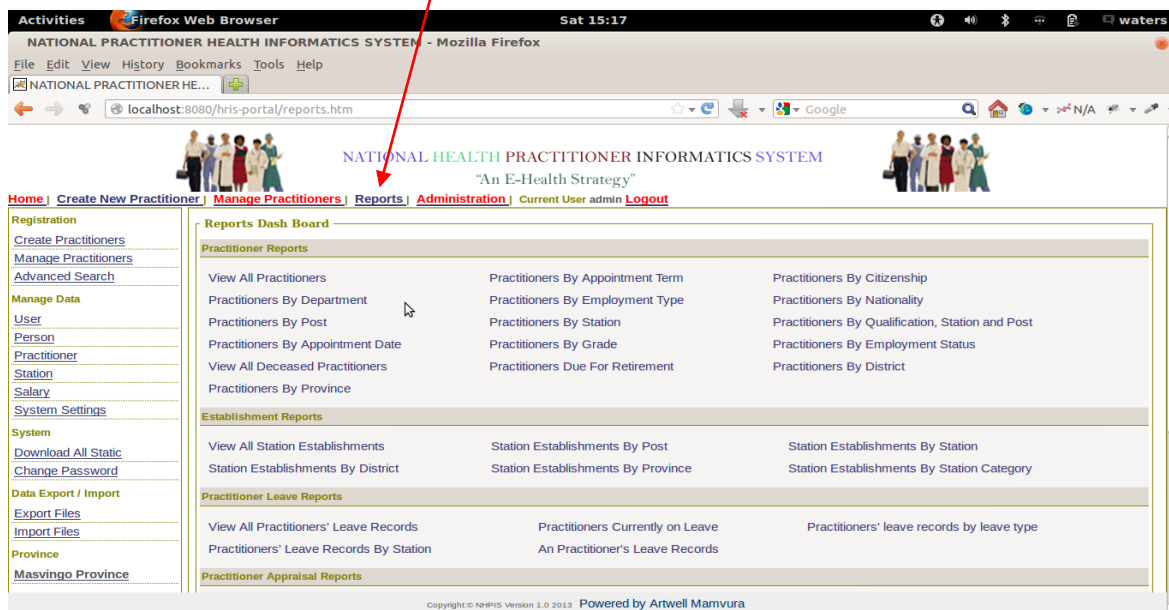


Fig A16 Reports

19. A user can upload a download File with practitioner details

Click Browse, navigate to the file location, and click import

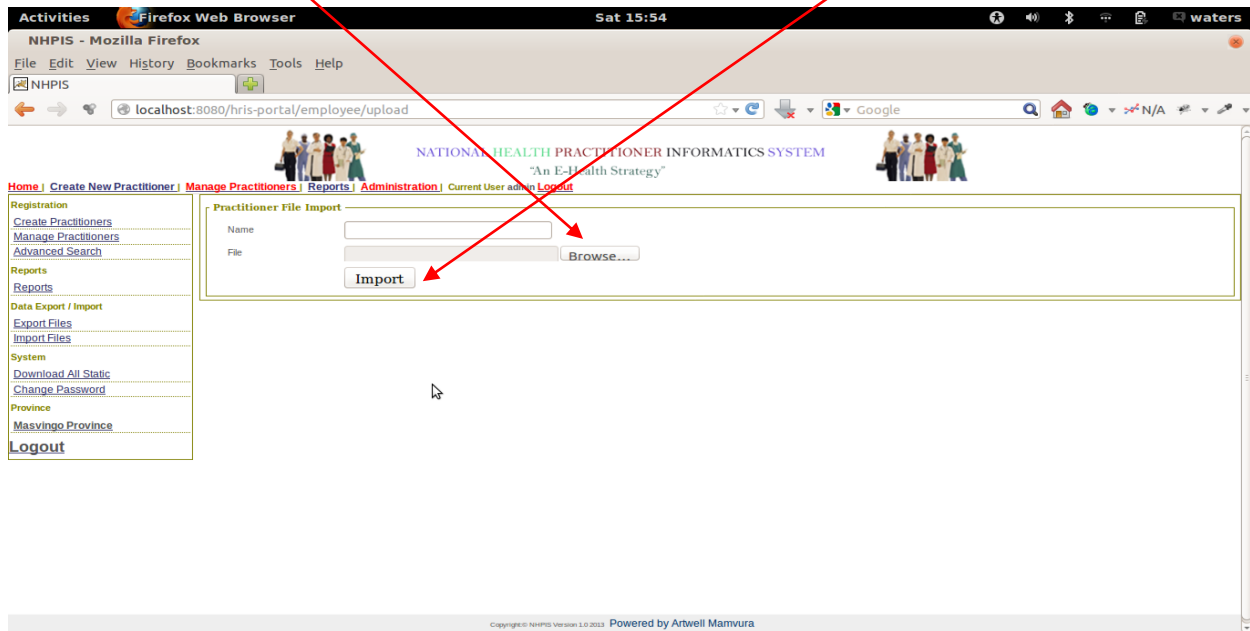


Fig A17 Upload Form

20. A user can download a practitioner files registered. Specify a period and click Download Button

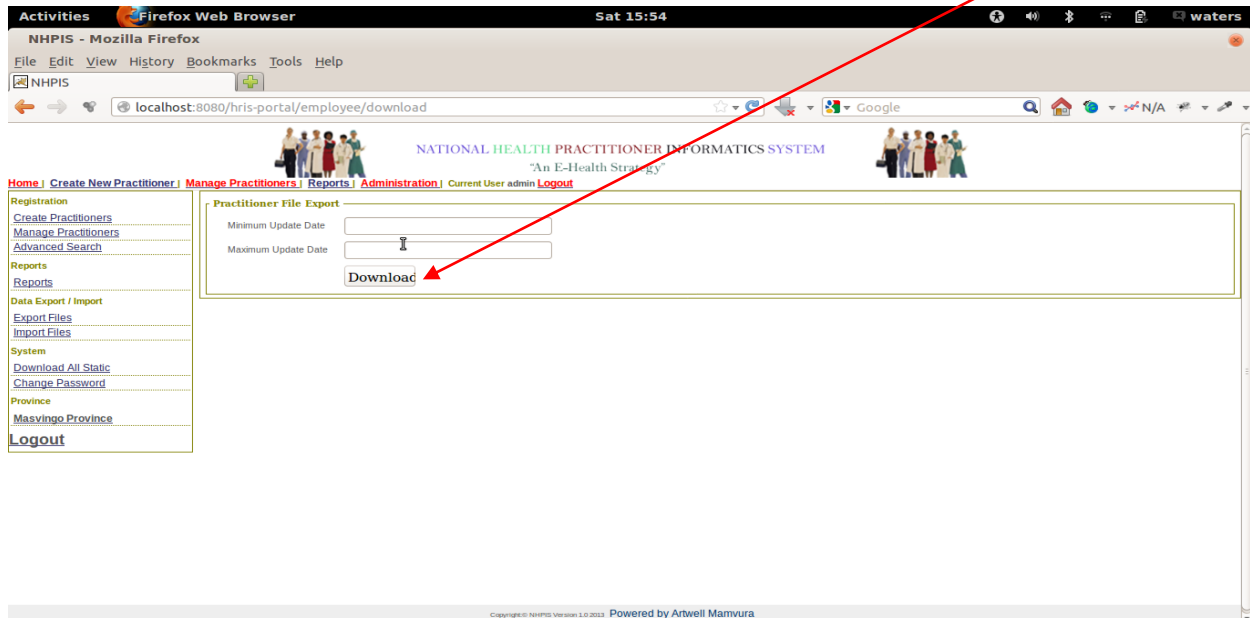


Fig A18 Download Form

21. A user can change his or her password. Click the change password link

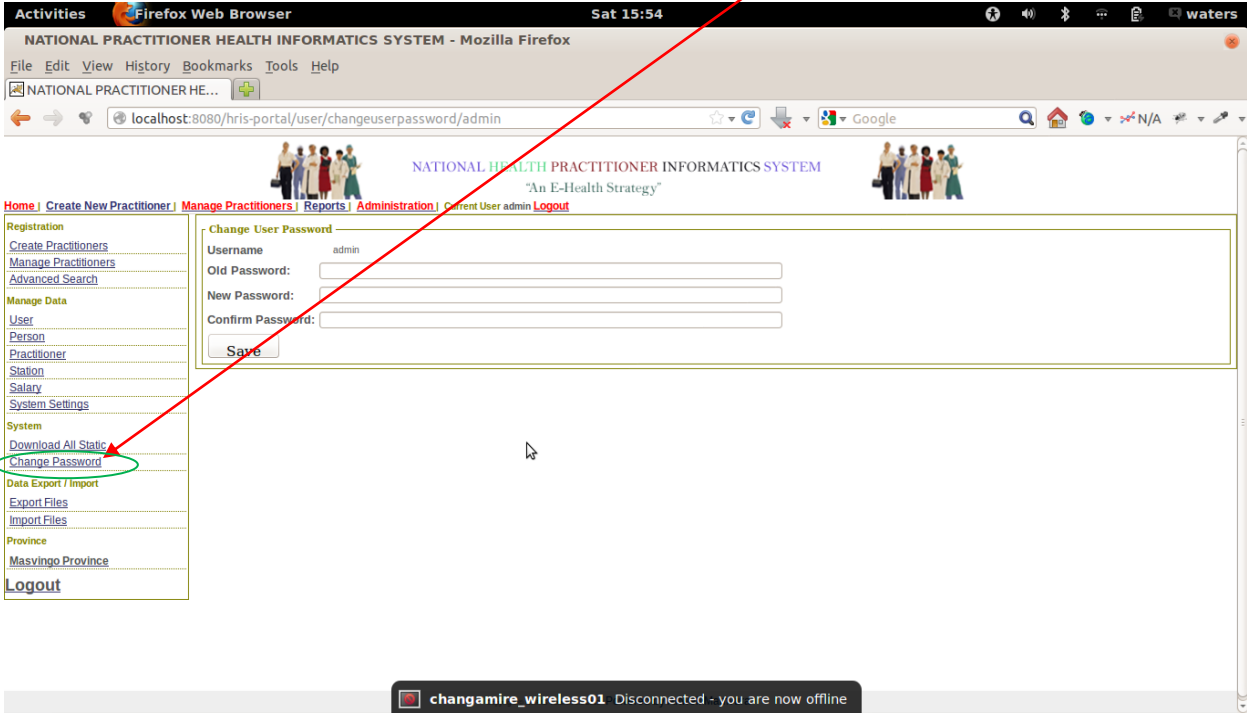


Fig A19 Change user password form

22. The administrator at the national instance can add static data

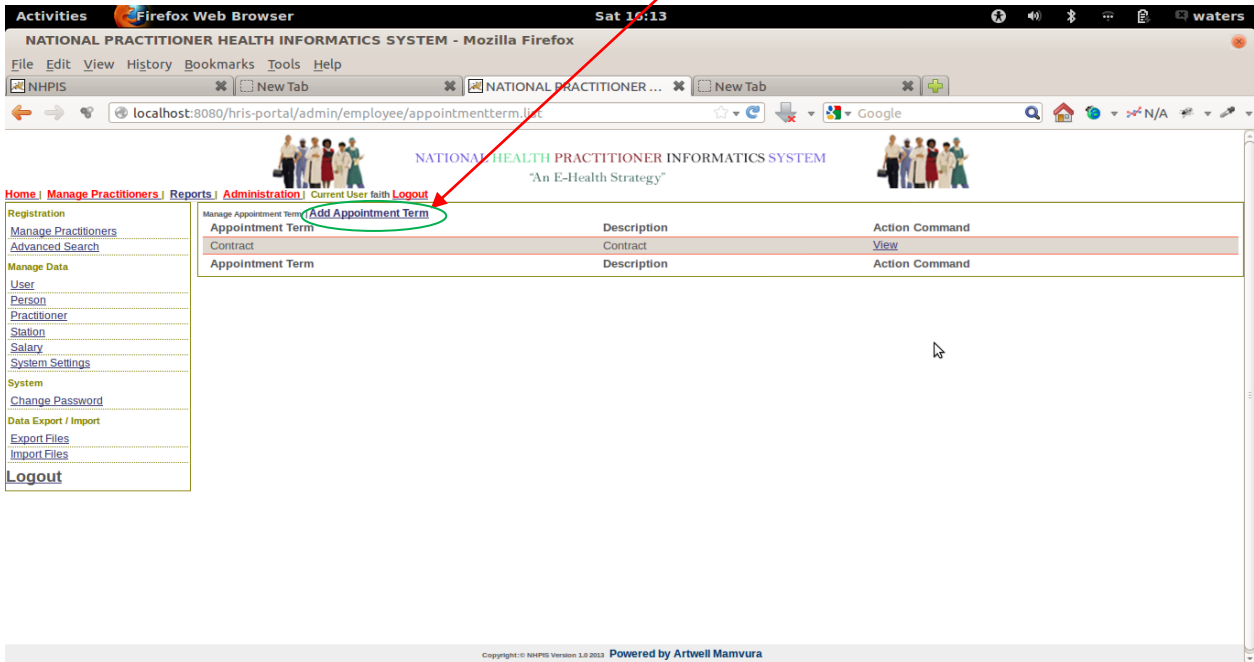


Fig A20 List Static data form



23. Add static data and click Save to save

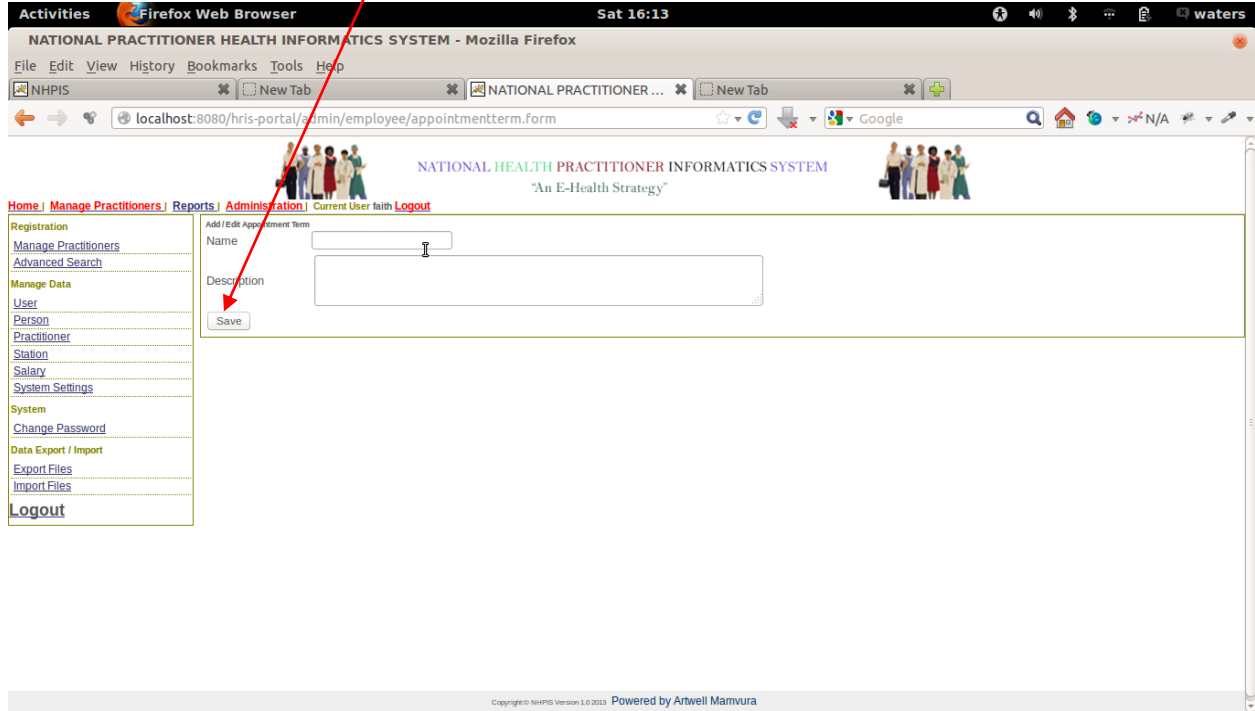


Fig A21 Add Static data form

24. Saved successfully alert message

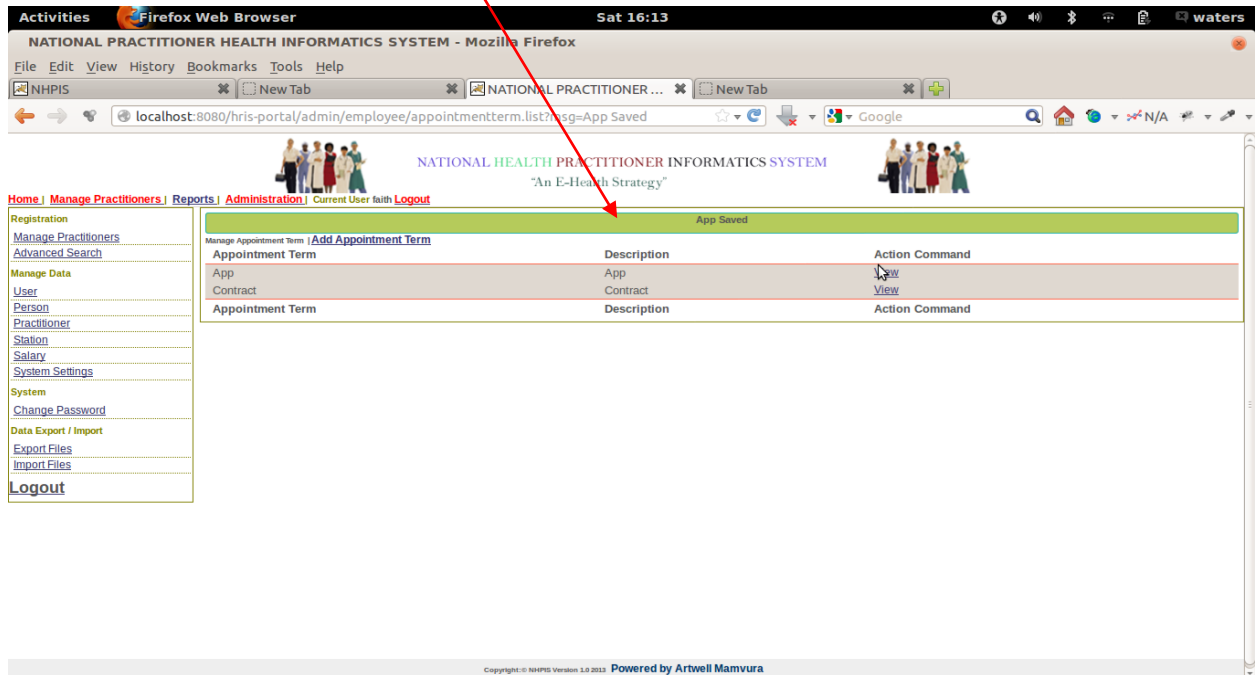


Fig A22 Save record Alert message

25. If a naming standard is no longer in use the system allows a user to retire it

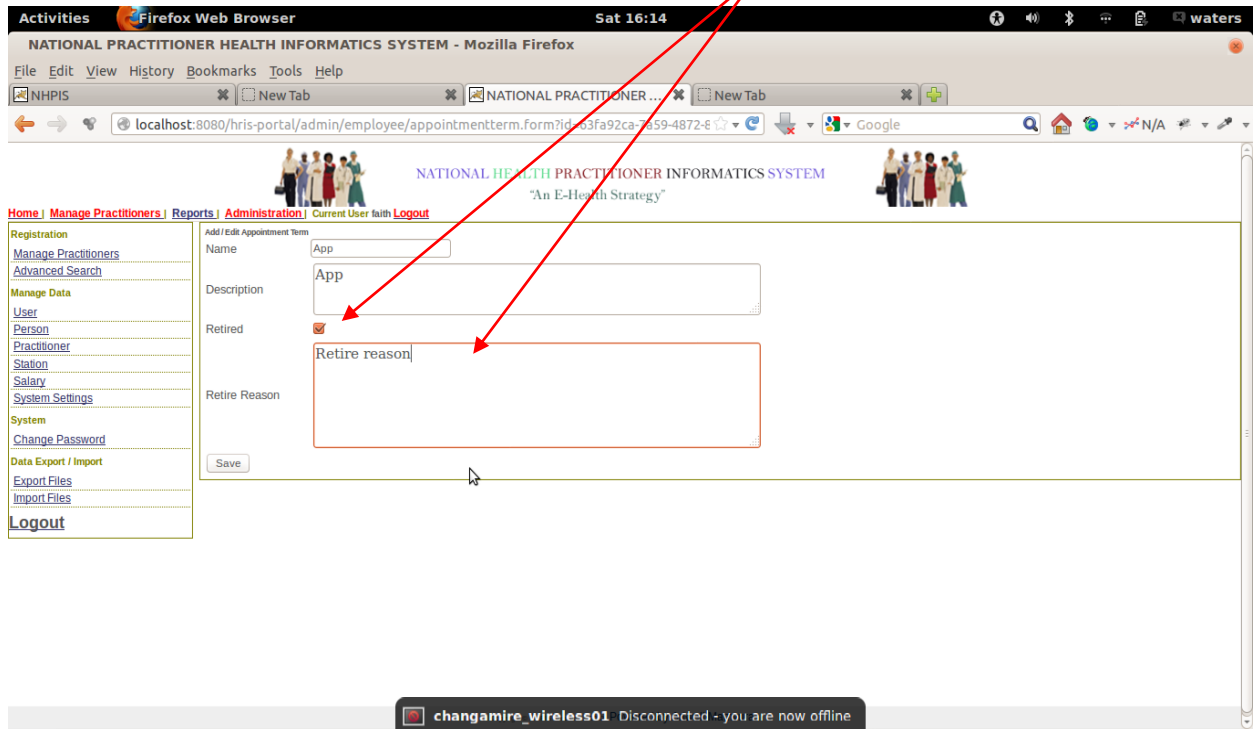


Fig A23 Retiring a record

26. Retired fields are shown with a strike through

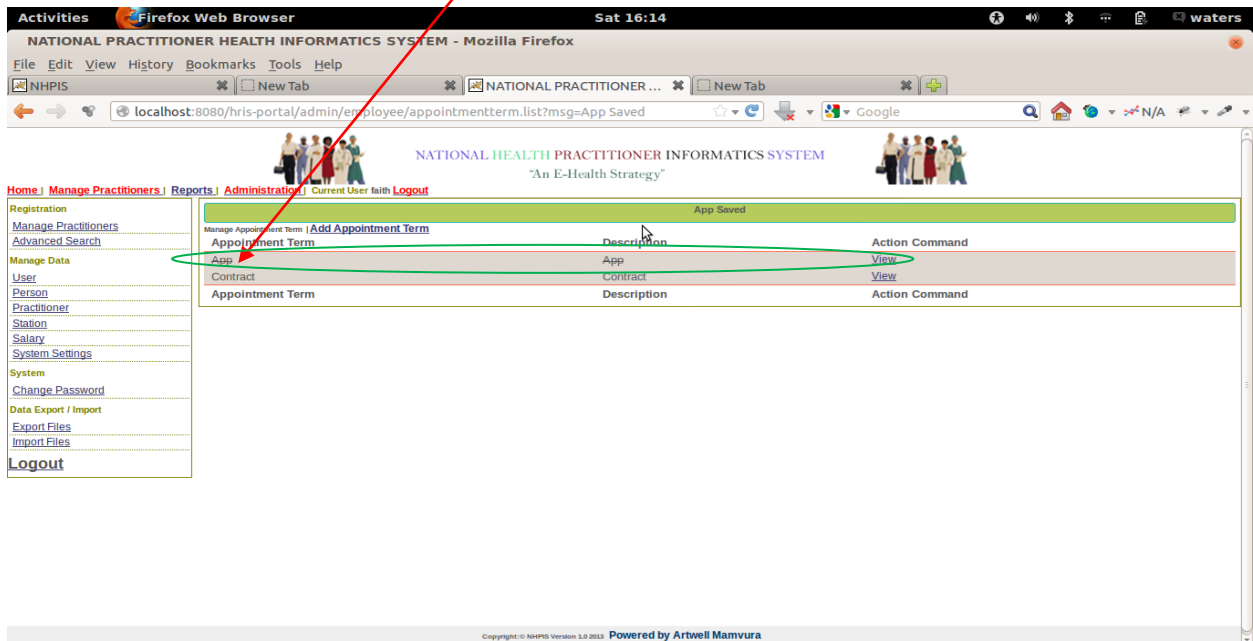


Fig A24 Retired records

27. A user can use the advanced search form to search a practitioner

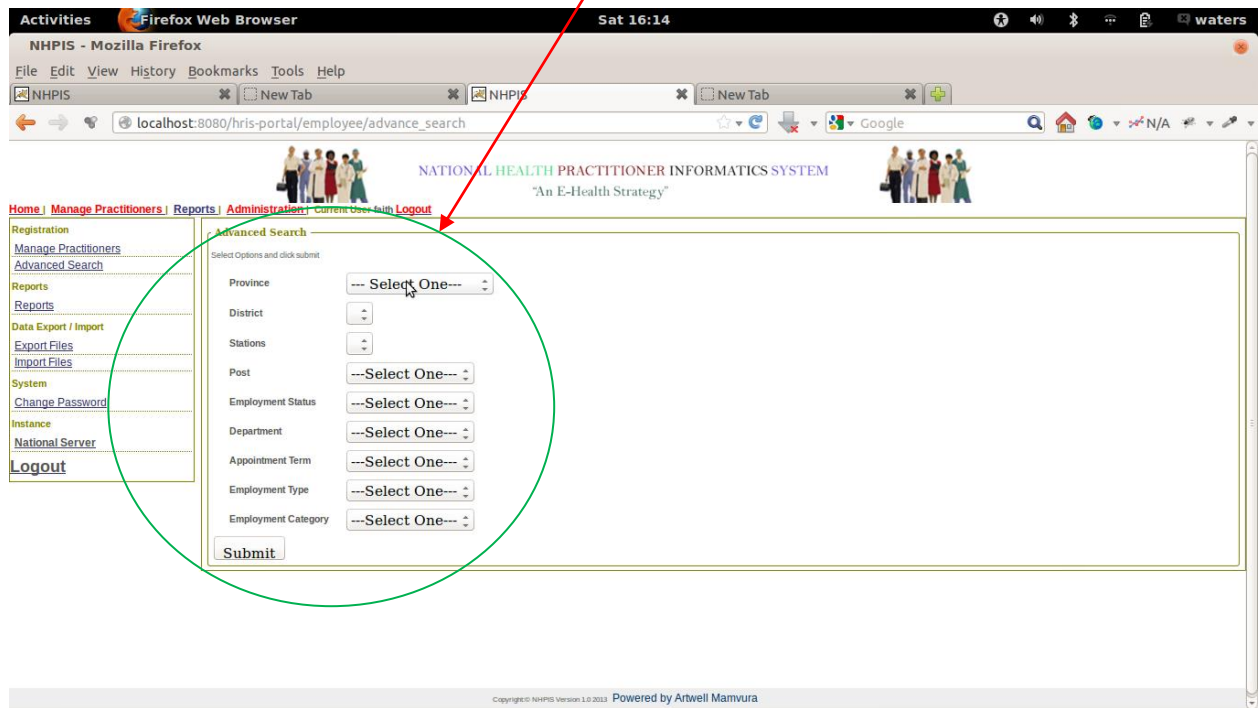


Fig A25 Advanced Practitioner form

28. Search Results      Export to other file formats      Search Criteria

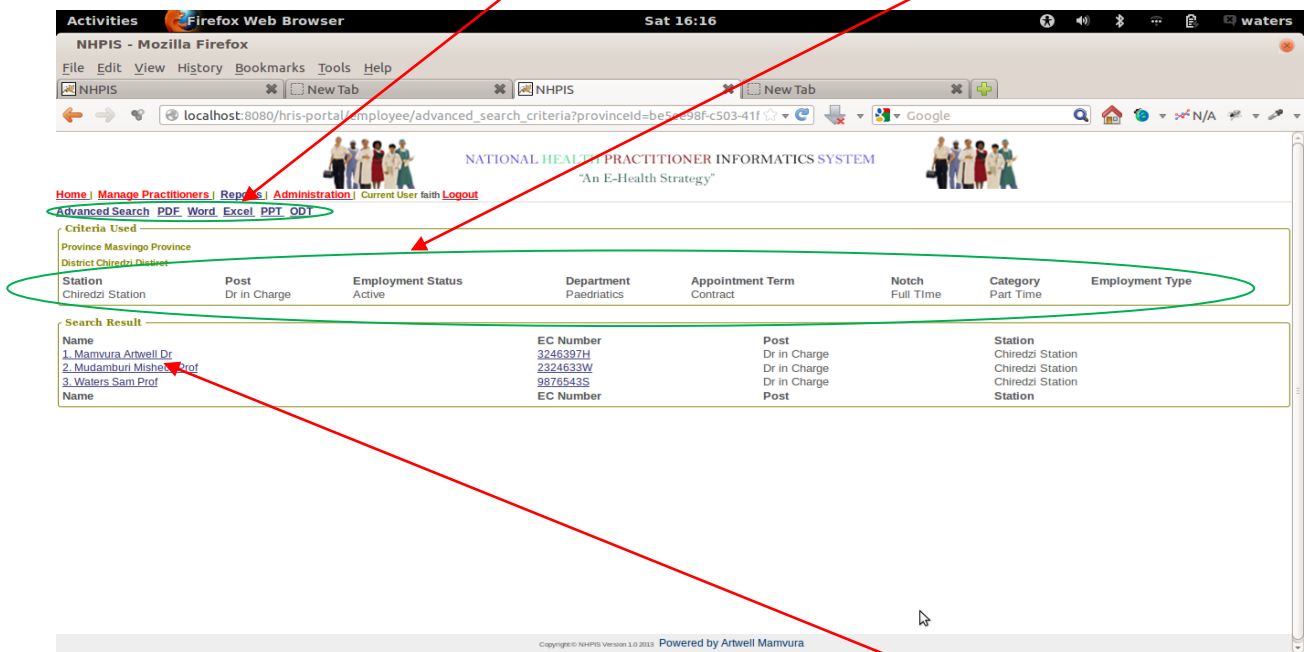


Fig A26 Search Results

Search Results

## **Appendix B: Interview Guide**

### **SECTION A**

**Appointment Date.....**

**Agenda.....**

### **SECTION B: Questions for Health Information Officers in Provinces**

1. What are the activities you carry out in you consider manual and need to be automated?
2. Can you give a brief description of the current system and what are the problems are you currently facing with manual registration of a Practitioner?
3. What do you understand by the term Practitioner Health Informatics Systems?
4. Do you understand the objectives and functionality of National Health Practitioner Informatics Systems (NHPIS)?
5. What are the key aspects of the proposed system usability or security issues?
6. Will the automation of the manual process make your work easier and how?
7. What kind of reports does the current system produce reports, which requires the reports?
8. Does the introduction of a NHPIS solve the problems of the current system?

### **SECTION C: Questions for MoHCW Practitioner**

1. How frequently do you receive data about practitioners from the Provinces?
2. Do you edit any practitioner data from the provinces?
3. What do you understand by the term Practitioner Health Informatics Systems?
4. What are the problems of the current system?
5. How often do you use the internet, will you be able to use the NHPIS system?
6. What are the types of the reports you need about a practitioner's data?
7. Will the adoption of NHPIS system solve the problems of the current system?
8. Do you think the new System will improve data quality?

## Appendix C: Questionnaire Checklist

I am Artwell Mamvura, a student at the Midlands State University in Zimbabwe pursuing a Degree in Computer Science. One of the requirements for this award is a project research and fully operating software application that can be implemented by an operating organization. This questionnaire is designed to find out the requirements for the National Health Practitioner Informatics System. I therefore kindly request you to assist me with the required information in this questionnaire. I promise to keep all the given information confidential and highly guard due rights. Thank you for your cooperation.

### Health Information Officer Questionnaire

**Qn1** What are the activities you carry out in you consider manual and need to be automated?

.....  
.....  
.....  
.....  
.....

**Qn2** Can you give a brief description of the current system and what are the problems are you currently facing with manual registration of a Practitioner?

.....  
.....  
.....  
.....

**Qn3** Do you understand the objectives and functionality of National Health Practitioner Informatics Systems (NHPIS)?

.....  
.....  
.....  
.....

**Qn4** What do you understand by the term Practitioner Health Informatics System?

.....  
.....  
.....  
.....  
.....  
.....

**Qn5** What are the key aspects of the proposed system usability or security issues?

.....  
.....  
.....  
.....

**Qn6** Will the automation of the manual process make your work easier and how?

.....  
.....  
.....  
.....

**Qn7** What kind of reports does the current system produce reports, which requires the reports?

.....  
.....  
.....  
.....

**Qn8** Does the introduction of a NHPIS solve the problems of the current system?

.....  
.....  
.....  
.....  
.....  
.....  
.....

*Thank you for this great contribution*

**MoHCW Practitioner Questionnaire**

**Qn1** How frequently do you receive data about practitioners from the Provinces?

.....  
.....  
.....

**Qn2** Do you edit any practitioner data from the provinces?

.....  
.....  
.....

**Qn3** What do you understand by the term Practitioner Health Informatics Systems?

.....  
.....  
.....

**Qn4** What are the problems of the current system?

.....  
.....  
.....  
.....

**Qn5** How often do you use the Internet and will you be able to use the proposed system?

**(Tick where appropriate)**

- Frequently and I will be able to use the system
- I will be able to use the system
- Sometimes and I will be able to use the system
- Rarely and I won't be able to use the system
- I won't be able to use the new system

**Qn6** What are the types of the reports you need about a practitioner's data?

**(Tick where appropriate)**

- Adhoc reports
- Weekly reports
- Monthly reports
- Quarterly reports
- Yearly reports

**Qn7** Do you think the introduction of Practitioner Health Informatics Systems will solve the problems of the current system?

.....  
.....  
.....  
.....  
.....  
.....

**Qn8** Do you think the new System will improve data quality?

**(Tick where appropriate)**

- Yes
- No
- Maybe

*Thank you for this great contribution*



**Appendix D: Observation Score Sheet**

Location: .....

Process Being Observed: .....

Date: .....

Time: .....

Objective of observation:

.....  
.....  
.....  
.....  
.....

Brief description of current system Process:

.....  
.....  
.....  
.....

Areas of strength:

.....  
.....  
.....  
.....  
.....

Areas that need development:

.....  
.....  
.....  
.....  
.....

Signed: .....

Date: .....

Signed: .....

Date: .....

## Appendix E: Snippet of Code

### Code to Connect to the Database

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
  <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
  <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/mohhris</property>
  <property name="hibernate.connection.username">root</property>
  <property name="hibernate.connection.password">hiu</property>
</hibernate-configuration>
```

### Java Database Connection to reference from the Application Context File

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/mohhris?autoReconnect=true&sessionVariables=storage_e
ngine=InnoDB&useUnicode=true&characterEncoding=UTF-8
jdbc.username=root
jdbc.password=hiu
```

### Application Context File for Dependency Injections

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" ">
  <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"
id="propertyConfigurer" p:location="classpath:org/hris/business/resources/jdbc.properties"/>
  <bean class="org.springframework.jdbc.datasource.DriverManagerDataSource" id="dataSource"
p:driverClassName="${jdbc.driverClassName}" p:password="${jdbc.password}"
p:url="${jdbc.url}" p:username="${jdbc.username}"/>
</beans>
```

## Code to create Objects from the Database

```
public class District extends BaseMetaData implements Serializable {
    private static final long serialVersionUID = 1L;
    private String districtId;
    private String districtCode;
    //Constructor
    public District() { }

    @Id
    @Basic(optional = false)
    @Column(name = "district_id", nullable = false, length = 36)
    public String getDistrictId() {
        return districtId;
    }
    public void setDistrictId(String districtId) {
        this.districtId = districtId;
    }
    Method that checks for duplication of records
    public boolean equals(Object obj) {
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final District other = (District) obj;
        if ((this.districtId == null) ? (other.districtId != null) : !this.districtId.equals(other.districtId)) {
            return false;
        }
        return true;
    }
}
```

### **Method that allows easy sorting of Records in the Database**

```
public int hashCode() {
    int hash = 7;
    hash = 79 * hash + (this.districtId != null ? this.districtId.hashCode() : 0);
    return hash;
}
}
```

### **Implementation of Generics with Methods to Save, List, Update, Delete, checkDuplicate**

#### **Method to List all records in the System**

```
public List<T> getAll() {
    return getHibernateTemplate().loadAll(this.persistentClass);
}
```

#### **Method to List all Distinct records from the Database**

```
public List<T> getAllDistinct() {
    Collection result = new LinkedHashSet(getAll());
    return new ArrayList(result);
}
```

#### **Method to Get an a single record from the Database**

```
public T get(PK id) {
    T entity = (T) this.getHibernateTemplate().get(this.persistentClass, id);
    if (entity == null) {
        log.warn("Uh oh, " + this.persistentClass + " object with id " + id + " not found...");
        throw new ObjectRetrievalFailureException(this.persistentClass, id);
    }
    return entity;
}
```

### **Method to check if a new record being saved is already in the Database**

```
public boolean exists(PK id) {
    T entity = (T) getHibernateTemplate().get(this.persistentClass, id);
    return entity != null;
}
```

### **Method to Save a record in the Database**

```
public T save(T object) {
    return (T) getHibernateTemplate().merge(object);
}
```

### **Method to Delete a record from the Database**

```
@Transactional(readOnly = false)
public void remove(PK id) {
    getHibernateTemplate().delete(this.get(id));
}
```

### **Method to search from the Database**

```
public List<T> findByNameQuery(String queryName, Map<String, Object> queryParams) {
    String[] params = new String[queryParams.size()];
    Object[] values = new Object[queryParams.size()];
    int index = 0;
    for (String s : queryParams.keySet()) {
        params[index] = s;
        values[index++] = queryParams.get(s);
    }
    return getHibernateTemplate().findByNameQueryAndNamedParam(queryName, params,
values);
}
```

## Method to List Deprecated Items in the Database

```
@Override
public List<T> getUnRetiredItems() {
    Criteria criteria =
getHibernateTemplate().getSessionFactory().getCurrentSession().createCriteria(this.persistentCl
ass);
    criteria.addOrder(Order.asc("name"));
    criteria.add(Expression.eq("retired", false));
    criteria.setResultTransformer(Criteria.DISTINCT_ROOT_ENTITY);
    return criteria.list();
}
}
```

## Method to Display an Admin Page

```
@RequestMapping(value = "/index", method = RequestMethod.GET)
public ModelAndView showIndex(ModelMap model) {
    logger.info("Showing index");
    if (contextUtil.userService.getCurrentUser() == null) {
        logger.info("User not logged in");
        model.addAttribute("notloggedOn", "true");
        return new ModelAndView("login");
    } else {
        logger.info("user logged in");
        contextUtil.getProperty(model);
        return new ModelAndView("index");
    }
}
}
```

### **Class For Connecting to the Database For Reports**

```
public class DBConnect {  
    public static Connection getConnection() {  
        Connection jdbcConnection = null;  
        try {  
            Class.forName("com.mysql.jdbc.Driver");  
            jdbcConnection = DriverManager.getConnection(  
                "jdbc:mysql://localhost:3306/mohhris", "root", "");  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
        return jdbcConnection;  
    }  
}
```

### **Class to Validate National ID**

```
public class NationalIdFormats {  
    public static String ZIMBABWE="^(\\d{2}-\\d{6,7}-\\w{1}-?\\d{2})$";  
}
```

### **Class to Validate Passport Number**

```
public class PassportNumberFormat {  
    public static String Zimbabwe = "^(\\w{2}\\d{6})$";  
}
```

### **Class to Validate EC Number**

```
public class ECNumberFormats implements Serializable{  
    public static String ZIMBABWE="^(\\d{7}[A-Z])$";  
}
```

### **Method to List unSynchronize Practitioner**

```
public List<Practitioner> getUnpushedPractitioners() {  
    return practitionerService.getUnpushedPractitioners();  
}
```

### **Method to get Practitioner Details from a Province**

```
public Practitioner getPractitioner(String practitionerId) {  
    return retrievePractitioner(practitionerId);  
}
```

### **Advanced Search for a Practitioner**

```
for (Station s : stationService.stationsInProvince(province)) {  
    practitionerList.addAll(practitionerService.getCriteriaSearch(s, practitionerStatus, department,  
employmentCartegory, post, employmentType, appointmentTerm));  
}  
} else if (employmentCartegory == null && practitionerStatus == null && post == null &&  
department == null && employmentType == null && appointmentTerm == null && station ==  
null && district != null && province != null) {  
    for (Station s : stationService.stationInDistrict(district)) {  
        practitionerList.addAll(practitionerService.getCriteriaSearch(s, practitionerStatus, department,  
employmentCartegory, post, employmentType, appointmentTerm));  
    }  
} else {  
    practitionerList.addAll (practitionerService.getCriteriaSearch (station, practitionerStatus,  
department, employmentCartegory, post, employmentType, appointmentTerm));  
}  
model.addAttribute("practitionerList", practitionerList);  
init();  
return "practitioner/search_criteria";  
}
```



### **Method to convert all Data types to String**

```
public Practitioner convert(String s) {  
    return service.get(s);  
}
```

### **Method to Manually Push Practitioner Files**

```
public String manualPush(List<Practitioner> practitioners) {  
    int totalPractitionerNo=0;  
    int totalStalePractitionerNo=0;  
    int totalErrorPractitionerNo=0;  
    int totalSavedPractitionerN0=0;  
    sb.append(totalEPractitionerNo-totalErrorPractitionerNo-totalStalePractitionerNo);  
    sb.append("<br />\n");  
    return sb.toString();  
}
```

### **Method to Login into the System for authenticating a user**

```
public boolean login(String username, String password) {  
    try {  
        Authentication authenticate = authenticationManager.authenticate(new  
UsernamePasswordAuthenticationToken(username, password));  
        if (authenticate.isAuthenticated()) {  
            SecurityContextHolder.getContext().setAuthentication(authenticate);  
            return true;  
        }  
    } catch (AuthenticationException e) {  
    }  
    return false;  
}
```

### **Method to Logout**

```
public void logout() {  
    SecurityContextHolder.getContext().setAuthentication(null);  
}
```

### **Code for Exception Handling**

```
public APIException(String message) {  
    super(message);  
}
```

### **Method to alter depending with instance whether Province or National**

```
<bean clas s= " org.hris.web.config.Hris Portal Instance Config Impl" id="hrisPortalInstance  
Config">  
    <property name="national" value="false"/>  
</bean>
```

### **Snippet of Code for calling User Interface Pages**

```
<bean id="viewResolver"  
    class="org.springframework.web.servlet.view.InternalResourceViewResolver"  
    p:prefix="/WEB-INF/jsp/"  
    p:suffix=".jsp" >  
</bean>
```

## User Login Page

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>HRIS Login</title>
  </head>

  <body onload='document.f.j_username.focus ();'>

    <form id="login" name='f' action="{pageContext.request.contextPath}/loginUser"
method='post'>
      <h1>HRIS Log In</h1>
      <fieldset id="inputs">
        <input id="username" name='username' placeholder="Username" autofocus=""
required="" type="text">
        <input id="password" name='password' placeholder="Password" required=""
type="password">
        <c:if test="{error}">
          <font color="red">Username or password is incorrect</font>
        </c:if>
        <c:if test="{notloggedOn}">
          <font color="red">You are not logged in. Please login.</font>
        </c:if>
      </fieldset>
      <fieldset id="actions">
        <input id="submit" value="Log in" type="submit">
      </fieldset>
    </form>
  </body>
</html>
```