



**André Filipe
Domingues Malta**

**Sistema open-source de registos clínicos de saúde
em doenças tropicais**

**Open-source electronic health record system for
neglected diseases**



**André Filipe
Domingues Malta**

**Sistema open-source de registos clínicos de saúde
em doenças tropicais**

**Open-source electronic health record system for
neglected diseases**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor José Luís Guimarães Oliveira, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais.

o júri / the jury

presidente / president

Prof. Doutor Joaquim Manuel Henriques de Sousa Pinto

Prof. Auxiliar do Dep. Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Joel Perdiz Arrais

Prof. Auxiliar Convidado do Dep. de Engenharia Informática da Fac. de Ciências e Tecnologia da Universidade de Coimbra

Prof. Doutor José Luís Guimarães Oliveira

Prof. Associado do Dep. Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

**agradecimentos /
acknowledgements**

Agradeço aos meus pais por me terem sempre apoiado, são eles a quem eu devo ser quem sou.

Ao meu orientador José Luis Guimarães Oliveira, pela sua direção, sabedoria e apoio.

Agradeço aos meus colegas e amigos que me foram acompanhando durante estes anos, bem como há paciência de todos nesta última etapa do meu percurso académico.

Palavras Chave

Registos Clinicos de Saude, Open-Source Software.

Resumo

Países sub-desenvolvidos são os principalmente afectados por um conjunto de doenças tropicais onde factores ambientais desempenham uma contribuição maior na sua origem. No geral estes países não dispõem de métodos para controlar estas doenças eficazmente, em parte devido à fraca implementação de Tecnologias da Informação em Saúde. Atualmente, com o avanço em standards e software na área da saúde, existem diversos sistemas *open-source* de registos clínicos que podem auxiliar centros de cuidados médicos na captura de informação útil à melhoria dos serviços prestados e há investigação de doenças negligenciadas. Nesta dissertação efectuámos uma revisão sistemática de tais soluções de maneira a escolher um candidato apropriado aos requisitos de uma cama de teste de um país sub-desenvolvido - Gondar, Etiópia. A implementação foi conduzida com ênfase à adaptação do fluxo de trabalho baseado em papel da instituição para o sistema proposto, assegurando que toda a informação gerada pelo centro pode ser capturada de forma digital. Como resultado final, um protótipo foi criado e algumas conclusões obtidas de todo este processo.

Keywords

Electronic Health Records, Open-Source Software.

Abstract

Low-resource countries are primarily the ones affected by tropical diseases where environmental factors play a major role. Means for controlling these diseases are often lacking in these countries in part due to their poor support of Health Information Technology. Nowadays, with the advances of standards and software in the health-field, several open-source electronic health record systems (EHR) exist which can assist facilities to capture of information, aiding to research and better health-care of neglected diseases in these countries. In this work, we performed a systematic review of several of such solutions to select the most appropriate candidate to satisfy the requirements of a test-bed in a low-resource country - Gondar in Ethiopia. The implementation was conducted with a strong focus on adapting the existing paper-based workflow of the institution to the proposed system, to assure that all the information generated in this center can be captured in a digital way. As a final result, a working prototype was deployed and some conclusions are obtained from all this process.

CONTENTS

CONTENTS	i
LIST OF FIGURES	iii
LIST OF TABLES	v
ACRONYMS	vii
1 INTRODUCTION	3
1.1 Motivations	4
1.2 Objectives	5
1.3 Thesis Outline	5
2 BACKGROUND	7
2.1 Electronic health record	7
2.2 Some challenges in switching	8
2.3 Free and open-source software	9
2.4 An open-source EHR system	10
2.5 A software model	10
2.6 Server and client software	11
2.7 Persistence support	12
2.8 Medical standards support	13
2.9 Operating-system support	13
3 RELATED WORK	15
3.1 Open-source EHR software	15
3.2 GNU Health	16
3.3 OpenEMR	20
3.4 FreeMED	25
3.5 OpenMRS	28
3.6 Bahmni	33
3.7 Software comparison	36
4 SYSTEM REQUIREMENTS	39
4.1 Scope	39
4.2 Functional requirements	40

5	SYSTEM ARCHITECTURE AND IMPLEMENTATION	45
5.1	Software Architecture	46
5.2	Technologies and communication	48
5.3	Data model	49
5.4	Bahmni EHR-OpenELIS interoperability	51
5.5	Network and security	54
5.6	Off-line support	58
5.7	Installation	58
5.8	Configuration and customization	62
5.8.1	Patients attributes and registration form	63
5.8.2	Address system	67
5.8.3	Appointments	70
5.8.4	Providing custom forms	71
5.8.5	Enabling off-line features	72
5.8.6	Security configuration	73
6	RESULTS	77
6.1	Users and roles	77
6.2	Registration extension	79
6.3	Patient management extension	81
6.4	Clinical extension	85
6.5	Validation	88
7	CONCLUSIONS	91
7.1	Future Work	92
	REFERENCES	93
	APPENDIX	99

LIST OF FIGURES

3.1	User-interface of GNU-Health's client application	19
3.2	User-interface of OpenEMR's platform	24
3.3	User-interface of FreeMED platform	27
3.4	OpenMRS's <i>legacy UI</i> user-interface	31
3.5	OpenMRS's <i>modern UI</i> user-interface	31
4.1	Use case diagram	41
5.1	Bahmni EHR architecture overview	47
5.2	Supporting technologies and intercommunication.	49
5.3	Overview of how feed producers and clients work	52
5.4	Representation of a feed and feed linkage	52
5.5	Relationship between user, roles and privileges in Bahmni EHR	55
5.6	List of access roles, some of which are custom.	55
5.7	VPN entry point with firewall, and server firewall	56
5.8	SELinux decision process logic	57
5.9	Bahmni components used on the solution implementation.	60
5.10	Adding a person custom attribute	65
5.11	Custom OpenMRS concept "Occupation"	66
5.12	Person attribute "Occupation", mapped to a custom OpenMRS concept	66
5.13	Customized person attributes	67
5.14	The common address hierarchy in Ethiopia	67
5.15	Custom XML address template for Ethiopia.	68
5.16	Hierarchy and mapping of an address field	68
5.17	Expanded mapped addressing system	69
5.18	Address hierarchy file created for Ethiopia addressing system	70
5.19	Alternative to the above method by using a Liquibase change set	71
5.20	Configuration for integrating the appointments feature with the Bahmni EHR user-interface	71
5.21	SELinux configuration file 'conf' in /etc/selinux/	74
5.22	SELinux configuration file 'jail.local' in /etc/fail2ban/	74
6.1	Gondar EHR main page, presenting all the available feature tiles.	78
6.2	List of users of Bahmni EHR and their associated roles. Includes the created users for doctor, nurse and registrant members.	78
6.3	Registration clerk available features in Gondar EHR	79
6.4	Customized patient registration page	80

6.5	Patient printable card, english version	81
6.6	Search page which is displayed before registration.	81
6.7	View and filter active patients in the clinic	82
6.8	Admit, Discharge, Transfer	82
6.9	Patient movement control, with additional notes support.	83
6.10	In-patient management area, showing a patient assigned to a bed in the "General Ward".	83
6.11	Appointment scheduling user-interface (modern UI).	84
6.12	Patient dashboard	85
6.13	Gondar EHR available tests in laboratory orders	86
6.14	Entering results for multiple ordered tests in OpenELIS.	86
6.15	Laboratory workflow between Gondar EHR and OpenELIS and different users on both platforms	87
1	Bahmni EHR "openmrs" database schema - all tables and their attributes	106
2	OpenELIS "clinlims" database schema - all tables and their attributes	107

LIST OF TABLES

3.1	Matrix with several aspects of the evaluated EHR software	37
5.1	Bahmni EHR-OpenELIS event fields	52
1	Login to system	99
2	Search a patient	99
3	Register a patient	100
4	Schedule patient appointment	100
5	Examine a patient	100
6	Manage a patient	101
7	Place an order	101
8	Receive order results	101
9	Make a diagnosis	102
10	Discharge a patient	102
11	Make a report	102
12	Receive an order	103
13	Enter order results	103
14	Validate order results	103
15	Send order results	104
16	Add a staff member	104
17	Edit a staff member	104
18	Delete a staff member	105

ACRONYMS

2FA	Two factor authentication	HL-7	Health Level-7
ACL	Access Control Lists	HTML	HyperText Markup Language
Ajax	Asynchronous JavaScript and XML	HTTPS	Hyper Text Transfer Protocol Secure
API	Application Programming Interface	HTTP	Hypertext Transfer Protocol
AtomPub	Atom Publishing Protocol	ICD	International Classification of Diseases
CPT	Current Procedural Terminology	ICT	Information and Communications Technology
CSV	Comma-separated values	ID	Identifier
DBMS	Database Management Systems	IHE	Integrating the Healthcare Enterprise
DICOM	Digital Imaging and Communications in Medicine	IRC	Internet Relay Chat
DSV	Delimiter-separated values	JavaEE	Java Platform, Enterprise Edition
DWR	Direct Web Remoting	JRE	JAVA Runtime-Environment
EDI	Electronic Data Interchange	JSON	JavaScript Object Notation
EHR	Electronic Health Record	JSP	JavaServer Pages
ERP	Enterprise Resource Planning	JS	JavaScript
FHIR	Fast Healthcare Interoperability Resources	LDAP	Lightweight Directory Access Protocol
FLOSS	Free/libre and open-source software	LIS	Laboratory Information System
FOSS	Free and open-source software	LRTC	Leishmaniasis Research and Treatment Center
FSF	Free Software Foundation	MPL	Mozilla Public License
GPL	GNU General Public License	MRI	Magnetic Resonance Imaging
GSP	Groovy Server Pages	MVC	Model-view-controller
GUI	Graphical User Interface	NGO	Non-governmental organization
HCPCS	Healthcare Common Procedure Coding System	NPO	Non-profit organization
HIPAA	Health Information Portability and Accountability Act	ODF	OpenDocument Format
HIS	Health Information System	ONC	Office of the National Coordinator for Health Information Technology
HIS	Health Information System		

ORDBMS	Object-Relational Database Management System	SNOMED CT	Systematized Nomenclature of Medicine - Clinical Terms
OSI	Open Source Initiative	SQL	Structured Query Language
OTP	One-time-password	SSH	Secure Shell
PACS	Picture Archiving and Communication System	TCP	Transmission Control Protocol
PDF	Portable Document Format	TLS	Transport Layer Security
PGP	Pretty Good Privacy	UDP	User Datagram Protocol
PHP	PHP Hypertext Preprocessor	UID	Unique identifier
PMS	Medical Practice Management Software	UI	User Interface
RBAC	Role-Based Access Control	URI	Uniform Resource Identifier
RDBMS	Relational Database Management Systems	URL	Uniform Resource Locator
REST	Representational State Transfer	VM	Virtual Machine
RPM	RPM Package Manager	VPN	Virtual Private Network
SMS	Short Message Service	WAN	Wide Area Network
		WYSIWYG	What You See Is What You Get
		XML	Extensible Markup Language

INTRODUCTION

The use of Health Information is often used at tertiary levels in the health-care field of low-resource countries. Nowadays, modern information systems are an important part in many fields but are generally unavailable in these countries as consequence of many adversary conditions. Poor infrastructure in health facilities and low education levels in these leads to the poor development and under-use of Health Information Technology.

Environmental factors are one of the main causes for the prevalence of several tropical diseases on these countries. Low-income and poor quality of living conditions plays a major role in treatment and spread of the disease, which affect specially the poorest in society.

Health facilities in these countries deal with diseases hard to control and the means to provide better understanding of these are often out of reach. These usually lack the means to efficiently store and share valuable information about the disease treatment or other important aspects of patients such as socio-economic and lifestyle data. As information is key to this process, stronger and modern health information systems need to be deployed in order to break this perpetuating cycle.

There is a need to build information out of many data produced in health-care. The use of paper-based work-flows hinders the management, occupies space and its prone to degeneration with time or subject to be lost in problematic events like floods or fires. Time is lost for staff during the information retrieval process, so storing and managing information in a digital manner is important for aiding the work-flow for health-care center staff for providing better health-care, and for better sharing of information with aiding organizations and researchers.

Health information systems store information in a system that uses electronic health records. Nowadays there is a great number of feature-rich electronic health records software developed, including open-source works. Characteristics like historical views, management of medications, appointment scheduling, integration with billing systems or laboratory can also be important to provide an efficient work-flow and better services to patients.

Today, there are several open-source electronic health record (EHR) software that provide a great number of features such as OpenEMR [1] or OpenMRS [2] which are used in health-care applications from small clinics to district hospitals. Understanding the challenges of low-resource settings provides helpful knowledge on what is needed for implementing such solutions. A number of these open-source works are designed from the ground with low-resource settings in mind, besides, their open-source nature is a beneficial aspect that enables adapting them to accommodate even more particular settings.

1.1 MOTIVATIONS

This work contributes to the objectives of an international project between the Institute of Tropical Medicine in Antwerp, Belgium, the BMD Software/University of Aveiro in Portugal and the University of Gondar in Ethiopia. One of the project's goals is to develop Information and Communications Technology (ICT) solutions to streamline the diagnosis and treatment of cutaneous leishmaniasis in under-served regions, with Gondar's LRTC serving a test bed. The project is part of the Euroleish-Net *programme* of Leishmaniasis control.

The Leishmaniasis Research and Treatment Center (LRTC) was established in 2004 and provides medical care for patients infected by leishmaniasis parasite, carries clinical trials and has its own laboratory and a bed ward for accommodating patients. The LRTC works under a low-resource environment where health-records and work-flow are paper-based and the supporting infrastructure is very basic. Maintenance, organization, efficient retrieval and integrity of the records is dependent on human-factors and involves itself a lot of care. Modernizing its health-records system and work-flow is not only necessary for the benefit of the health-care services provided, but also to the work of other organizations such as Drugs for Neglected Diseases initiative, which rely on this information for providing research on the aforementioned diseases. In return this can generate new opportunities for providing better health-care and disease control.

The main motivation of this work is to contribute to the advancement of the current health-record system and work-flow in place in similar health-centers, by providing a centralized health record database to link patient's data collection, clinical images and

lab test results for diagnosis and treatment monitoring.

1.2 OBJECTIVES

The focus of this dissertation is centered around the implementation of a solution for use in aforementioned scenarios using only open-source technologies. The scope of this dissertation does not include all the possible applications of the area, but focus mainly on a robust EHR implementation. The electronic health record be considered the most important part of the system since it gives support for other applications to be laid on its foundations.

The objectives are, on a first phase, to understand how a health-record is defined and how the model is used in EHR. After, to comprehend how EHR can be built today using open-source technologies. On a second phase, to identify current open-source software which implement EHRs and determine if one can be used for the main focus of the dissertation. On a final phase, to implement the necessary aspects of the software to satisfy the requirements gathered from our low-resource setting, and to validate the system.

1.3 THESIS OUTLINE

This document is organized in 7 chapters. With chapter 1 containing the introduction and this outline, the remaining chapters are:

- Chapter 2 gives a general overview of health records and depicts some supporting technologies and related standards for handling, storing or exchanging electronic health records.
- Chapter 3 presents an evaluation and comparison of several open-source software for EHRs and recommends one appropriate candidate for use in low-resource settings.
- Chapter 4 shows the system requirements for an electronic health records solution gathered from traditional paper-based workflows and use-cases of a health-care center in a low-resource country.
- Chapter 5 presents the architecture and implementation of the solution, and includes installation and configurations made to customize it according to the system requirements described in the previous chapter.
- Chapter 6 presents the results of the deployed demo and validation.

- Chapter 7 shows some conclusions obtained from the work done in this thesis, and future work is proposed.

BACKGROUND

In this chapter we describe what is a health data and how meaning is obtained from it, which includes how it is categorized and how relations to a subject are established. We describe former methods used for collecting, storing and handling health data and contrast with the electronic health record, highlighting some challenges which hinder transition to modernized methods. From there, we give an overview of how an EHR software can be build using only available open-source technologies.

2.1 ELECTRONIC HEALTH RECORD

Data is a frequently used word with different connotations. It can be described as source data, data which was not yet processed for meaningful use. When a meaningful interpretation can be taken from data, it becomes information. Categorizing a source data is one example of representing information.

In health care systems, data related to patients can be categorized as follows: information which makes one person differ from the rest, for example his name, address, date of birth and so on comes under the category of demographics; information obtained from a patient's visit to a health-care provider or regarding admittance of the patient in the hospital is recorded by heath-care professionals for the analysis of ones health comes under the clinical category; information regarding the bill payer, like his insurance company details, telephone number, comes under the category of the financial data. When we specify data under one of these categories, we are implicitly relating it to some subject. These examples portray data that is related to different subject types:

a person, a patient or a bill payer. Categorizing the data is not the only process for building information from it, establishing relations to subjects and between subjects also builds information. The design of how each piece of data is organized and related to one another is often regarded as a data model.

Health record is the collection of all the data elements related to a patient in a health system. A health record can be a collection of the patient's life long medical history that are maintained by a given health care provider. For simplicity, health records are often referred as the patient's information regarding a single visit to the health care facility or the collective experience.

In a health-record, a relation between patient and all of his data needs to be established. This is often represented by a unique identifier attributed to the patient, and associated directly or indirectly to all his related information. On a health facility it is called a patient index, which is uniquely taken from a *master patient index* and associated with the patient typically the first time he visits the health facility. Other indexes are often attributed to a patient in other services of a health-facility. For example, billing services identify a patient with an account number. Both these indexes are associated together through the master patient index, hence linking both subjects as one.

Identifying a patient uniquely in the system and linking this identity to the account number and hence linking the account number to the clinical or demographical data is the foundation of an health-record. Representing this model in an information system supported by computer technology leads to the creation of an EHR system [3].

2.2 SOME CHALLENGES IN SWITCHING

Computer technology and literacy

As we have advancing to the new era of technology there are still people out there that aren't computer literate or want to stick to the same ways of doing things which they learned while growing up. Sometimes it is the feeling of being uncomfortable that drags them away from using the electronic means while in other cases it is the lack of knowledge that hinders this process. In a health-facility it is more likely to have staff for nursing than other kind of staff that is computer literate. In such situation they may be unable to input any form of information into the system as they don't know how to communicate with the machine itself. To solve this issues there can be an arrangement of free computer classes and software formation given to the people who are already working in the facilities.

Funding and cost of change

Setting up an electronic system for storing data in big facilities comes with the burden of the cost of setting it up. Lack of funds for the health care is one of the major issues these facilities are dealing with. Health administrations and governments frequently consider if health-centers should invest their own money into such a setup as funding availability is low. The initial steps of setting up an EHR system are very important in the aspect of finance and in the aspect of upcoming time. As the data grows and the organizations change, the EHR system should handle these changes seamlessly and continue to provide desired results. Therefore the administrators who are in charge of planning the implementation of such system should take into the consideration requirement of the health organization and also their current clinical practice guidelines. In addition to this the administrator should calculate the current system cost plus the cost to change the system in the future to accommodate the changes and should determine the long term value of the EHR system.

Software adaptability

If a health-care institution has the means and reasons the change, functional features of the EHR system are not the only important characteristics for selection. An evaluation of different EHR systems should be done in order to identify if they are appropriate to replace whatever system the institution has currently in use and understanding which issues can appear from such transitions. The potential adaptability of such a system is an important aspect for knowing if these issues can be reduced. EHR systems are configurable up to some extent, but often a serious adaptation is needed for accommodating it to a specific setting. Software licensing may pose a problem to this as this control is generally in the hands of the software's author and not on the hands of the licensee [4].

Comprehending this enables the possibility to draw a line between proprietary software and free-software.

2.3 FREE AND OPEN-SOURCE SOFTWARE

While a universally agreed definition of free/libre and open-source software (FLOSS) software does not exist, it is commonly defined as software which is under a compatible license for this purpose, a FLOSS license. The *free* part refers to the licensing, which in this case has no legal restrictions enforcing commercial or proprietary aspects on the software, while the *open* part refers to the source-code. Software licensed in this way is expected to be freely obtainable and the source-code available publicly. Various groups like the Free Software Foundation (FSF) and Open Source Initiative (OSI),

which support FLOSS software, maintain lists of approved licenses [5].

For implementing a EHR system, free/libre and open-source software licensing is can be desirable for its permissive licenses and availability of code. Proprietary and FOSS licensed EHR have their differences and depending on the usage scenarios they can be more or less appropriate, but on the latter case, one can, if necessary, implement required features or adapt the software according to his requirements himself, without the need to deal with commercial license agreements that often do not provide or allow modifications to its source code. Being able to understand what the software is doing by looking at the source code is also an important factor as EHR deals with sensitive information. In low-resource countries FLOSS can be more appealing than proprietary software by looking at cost of software licensing, providing these funds to be available for use on other things.

2.4 AN OPEN-SOURCE EHR SYSTEM

With EHR described as such, an EHR system can be built. In these following sections, we try to find some software components and models which can support such system for low-resource settings, with examples of free and open-source technologies that can be used for its implementation. OpenEHR¹ provides open source specifications and reference implementations of future proof EHR systems. Despite this, OpenEHR proposes very advanced software models for EHR which branch out of the complexity and scope of this thesis, so a simpler model is presented.

2.5 A SOFTWARE MODEL

An EHR system is typically used at the same time by health-care professionals from different fields of activity. These sometimes need to access EHR platforms while away from their work-place, where typically they would not have access to it. Also, in low-resource settings, the available computer hardware is often outdated and not capable of running a EHR system. The architecture of the software should be designed with these aspects considered. One software model which is appropriate for this scenario, is the client-server model.

The client-server model is a distributed application structure where there is a division between a server, that provides the services or resources, and the clients that access these.

¹<http://www.openehr.org>

Typically servers await for client to communicate their requests through a protocol agreed upon, and process or provide accordingly. A server requires a program that listens for the communications, implements the protocol and handles the requests, and the client requires a program that implements the same protocol and initiate the requests. Typically, most of the logic and resources reside on the server side, so that the client can be very simple and generic, a so-called "thin-client", thus, enabling the user to request only what it needs when it is needed. It is advisable to have a EHR system using this architecture because it enables isolation of server functionality and data model into a a separate system, and having the access mediated through a thin client that only gets or controls what it can on the system. One obvious advantage of this is data privacy. In low-resource countries it is also advantageous to use software with this architecture because it makes possible the use of outdated, simple or more mobile devices to interact with the platform, like donated old computers, laptops or smart-phones.

2.6 SERVER AND CLIENT SOFTWARE

One popular application which implements the client-server model is a web server.

A web server is a computer program that speaks the HTTP protocol and serves clients requests, typically web content. The web server program can provide static content without applying any processing to it or provide dynamic content based on the client session or other conditions. At first, gateway interfaces that allow the web servers to interface with executable programs were implemented on the web servers. Later, new programming languages that enable the creation of runtime code to be run server-side were born and integration added to web servers. These can provide environments to run for instance Perl or PHP code server-side.

A popular open-source web-server is Apache [6], licensed under Apache License 2.0. There are currently two supported branches of the software, the former, version 2, and the latest, version 3, which diverged from version 2 run-time support compatibility and provides new features. Version 2 is however still actively maintained and frequently receives security updates.

A web application server is similar to a web-server but provides a given platform with a wide range of functionality frequently used in server software for web application developers to use or extend. These features usually include persistence APIs with access and connection management, managing of web server requests, web application crashes, mitigations and security policies. A useful advantage of a web application server is to serve several web applications under the same server. This is useful for example for a health-care provider to have different web applications providing services of different

domains.

A popular open-source software for this is Tomcat by the Apache Software Foundation which is licensed under the Apache License 2.0. It is a servlet container which supports web applications written for a given Java environment.

In order to interact with the server, a client application is required. In a EHR system implemented using one of the two types of servers depicted above, a client for interacting with the system would be a browser application. Although the openness nature of the client is not a requirement for the EHR system to be open-source, it is if the whole environment is considered. Examples of these are Firefox ² or Otter Browser ³.

2.7 PERSISTENCE SUPPORT

A EHR system needs persistence support for its data model. One example for providing persistence support to building robust applications is a Database Management Systems (DBMS).

DBMS were born for processing large amounts of information, and to provide access to this data in a standard way. EHR systems typically store and access large amount of data from different sources and with various degrees of sensitivity and preservation importance, so there is a need for having a system for management and error recovering. Popular and robust open-source implementations of these are MySQL and PostgreSQL, which EHR systems can use for the implementation of their data models.

MySQL [7] was initially released as open-source and licensed in GPL. MySQL is nowadays a proprietary RDBMS widely adopted in the area, under its new owner, Oracle. Created as a combination of open-source and proprietary effort from Oracle, this acquisition worried many as they feared a relicensing from Oracle to a more restricted license and so the open-source project was continued under a new fork called MariaDB. Under the hood, MySQL has the InnoDB storage engine and a new update from it, called XtraDB, with better multi-processor management, while still keeping its ACID-compliant design. MySQL is distributed in an enterprise edition, with a proprietary license, and in a community edition, licensed under GPL.

²<https://www.mozilla.org/en-US/firefox/products/>

³<https://otter-browser.org/>

2.8 MEDICAL STANDARDS SUPPORT

With the development of health-care information systems over the years, some medical standards for these types of systems were also created.

Some EHR systems can be lacking in features more commonly found in other health information system (HIS). Nonetheless, this can be solved by combining them with complementary systems so interoperability between them is often required.

HL-7 is a international standard from Health Level Seven International which defines a messaging standard for transmitting messages with clinical data between health information system. There are two currently popular versions of the standard, version 2 [8], which uses a specific encoding comprised of reserved words, delimiters and data, and version 3 [9], which uses XML format. There are several standards based on both versions, one of major importance is Fast Healthcare Interoperability Resources (FHIR) [10].

Open-source EHR software can benefit from the use of already established open-source projects that implement medical standards. For instance, providing HL-7 version 2 capabilities or support for FHIR can be done by using HAPI ⁴, a Java library dual licensed in MPL and GPL.

Digital Imaging and Communications in Medicine (DICOM) [11] is a standard created by the DICOM Standards Committee that defines a network protocol and a file format for medical imaging. In EHR systems it is commonly used for handling, storing, printing and transmitting information obtained from DICOM-compatible devices, like MRI scans from radiology equipment.

One implementation of the DICOM standard, is dcm4chee ⁵, developed in JAVA. It is part of the dcm4che project, a collection of open source tools for the healthcare enterprise. It is licensed under MPL/GPL/LGPL triple license. It can also be setup purely as a picture archiving and communication sSystem (PACS) service and supports the HL-7 standard for data exchange.

2.9 OPERATING-SYSTEM SUPPORT

FLOSS EHR software can be paired with a supported operating system based on the technologies it uses. This can include proprietary operating systems such as Microsoft Windows, however, in the non-proprietary world there are free and open-source operating systems such as GNU/Linux or BSD distributions which can be

⁴<http://hl7api.sourceforge.net>

⁵<http://www.dcm4che.org/>

used to keep the whole ecosystem open.

Using these open-source and cross-platforms technologies under the described software model, EHR software can be implemented with the advantage that a fully open system gives.

In conclusion, in the present there are open-source technologies which can leverage the creation of an EHR system. While with them the development of a new EHR system is possible, there are related works which bring other advantages with their use such as established communities, development progress or support.

RELATED WORK

This chapter presents a review of open-source software which implement electronic health records, based on public available information sourced from the Internet and on personal evaluation. The evaluation will focus on each tool features and standards supported, software design, development progress, popularity, community adoption and support. Although some of the software tools provide on-line demos for demonstration purposes, the evaluations depicted here were obtained through analysis of local installations of the software.

3.1 OPEN-SOURCE EHR SOFTWARE

On a first phase, EHR software was selected based on several aspects like openness (license), development progress, community support and popularity. This process reduced the candidates to the following: GNU Health, OpenEMR, FreeMED, OpenMRS and Bahmni. This chapter purpose is to choose one software which is capable of providing an appropriate solution for health-care applications in low-resource countries from evaluation and comparison of potential candidates.

3.2 GNU HEALTH

GNU Health ¹ is a electronic health records software and health information system developed by GNU Solidario ², a non-profit and non-governmental organization for health and education development. It was adopted by the Free Software Foundation in 2011, becoming official software of the GNU project with the goal to contribute to the improvement of lives of the underprivileged by providing a free system that optimizes health promotion and disease prevention [12].

The main areas GNU Health supports are: 1) individual management; 2) patient management; 3) health center management; 4) information management.

Individual management regards to a person's basic information, genealogy, household, domiciliary infrastructure and so on. For GNU Health project, an individual is a person first and then a patient. It encourages the health organization or institution to perform a census, at least on the habitants part of the operational sector it covers, so that it knows its population first-hand [13]. GNU Health project empathizes the importance of social and economic development, giving a strong focus to relevant factors like education level, occupation, living conditions and family relations. Patient management comprises the collection of different types of data related to a patient's health, for instance lifestyle parameters like diet and exercise, addictions, sexuality or safety. It also includes management of the patient's health-care like encounters and evaluations, medical procedures, laboratory test requests or results and so on.

Health center management is mainly comprised of enterprise resource planning (ERP), for instance financial and human resources management, sales, invoicing and accounting, inventory, stock and supply chain management. It also features stronger focus on pharmacy and laboratory management. The managed subject can either be the institution itself or individual departments within.

Information management covers the tasks that combine data produced in the platform. This includes building reports or presenting demographics with configurable parameters, which are useful for epidemiology studies or crossing different data for instance.

The areas explained above generally involve teams with different fields of activity, on which members generally only work with the domain of information corresponding to it. For example a team of social workers may perform individual interviews to people for demographic analysis, nurses or doctors may handle patients' health-related tasks and information, administrative staff may do health center management and so on. GNU Health's access management allows this separation of responsibilities by creating users with different access rights for respecting the information domains that different teams

¹<http://health.gnu.org/>

²<http://www.gnusolidario.org/>

work with. The platform provides an access control system that follows the Role-Based Access Control (RBAC) model. In this model, roles, known as groups in GNU Health, define a list of rights a user can have in the system. A user's access is defined by the roles he is associated with. The default access management is supported through a password-authentication method but it is also possible to authenticate via external methods like Lightweight Directory Access Protocol (LDAP).

For exchanging health-care information with different platforms, GNU Health implements FHIR. To assist this transfer it employs a separate server supporting the Representational State Transfer (REST) protocol. The access management of this server uses the same access model depicted above, and the exchanged data is encrypted on the application layer by using TLS.

There are two modes available for synchronizing data between multiple GNU Health servers, centralized or satellite. In the former, synchronization is made from independent servers to a central server that aggregates all the information, for example several institutions synchronizing their data with the Ministry of Health. In the latter, synchronization is made between each independent server with each server acting as a master and sharing data between themselves, for instance used in the case of several cooperating branches of the same institution. Synchronization is backed by a server-side message broker software on master servers, and a message broker client running on any server that pulls synchronization data. Secure communication is attained by the use of TLS in the communication between the previously designated systems.

As the information stored by GNU Health is highly sensitive, the platform is developed with a strong focus on data confidentiality and integrity. It provides serialization, hashing, signing and verification which are commonly used in features that handle or produce sensitive information such as reports, prescriptions or laboratory tests. It allows doctors to digitally sign death certificates or for users to verify the authenticity of reports compiled on other GNU Health systems. This is achieved via a bundled client-side plug-in for Pretty Good Privacy (PGP) that uses GNU Privacy Guard implementation.

The report sub-system makes heavy use of these cryptographic functionalities, supports creation of charts and outputs to ODF and PDF formats.

The support for medical imaging in the DICOM format is not one of GNU Health's strengths. It directs this support to other softwares such as Aeskulap or Ginkgo CADx [14], which are also open-source. GNU Health project releases public security advisories when they are aware of a vulnerability, allowing implementers to mitigate the issue earlier [15].

GNU Health is written in Python 2 and uses the Tryton framework, both licensed in GPL v3+. Tryton is a high-level general purpose application platform that provides GNU Health with an extensible, scalable, secure and modular framework. It also provides

specific software components generally suited for ERPs. The modular approach inherited from Tryton gives the possibility for implementers to choose only the components they need to satisfy their requirements.

The platform supports UTF-8 and is translated in many different languages, with official support for English, Chinese, French, Greek, Italian, Japanese, Portuguese and Spanish [16]. Support to a new language is made available through installation of a corresponding language module on the server.

Most of GNU Health's functionality is provided by the official add-on modules which extend the core module. These are separated into general domains or specialties, e.g. pediatrics, surgery or gynecology. Modules add new functionality to these domains by re-using models of given entities and creating specification classes designed in other modules. As it creates dependencies between modules, in order for a target module to work its parent modules need to be installed too.

In GNU Health, the way to provide new functionality such as custom forms for collecting patient information is through development of modules for it. Since this requires programming knowledge, building and adding custom forms becomes a difficult task for non-developers. GNU Health provides official modules for several neglected tropical diseases including visceral leishmaniasis. However, a more simple way for building custom forms should exist to empower health-organizations that do not have the required resources or staff with the skills to do so.

GNU Health user-interface contains a global menu where functionalities are presented in a tree-style structure (Figure 3.1). The menu supports filtering by text, providing easier and quicker ways to find the desired functionality. To the right of it is a content panel, dependent on the active menu item. Several items can be active at the same time, making use of tabs for arranging the multiple content panels. It is not uncommon for some features to present a multitude of functionalities separated into their own tabs, adding to navigation and display complexity.

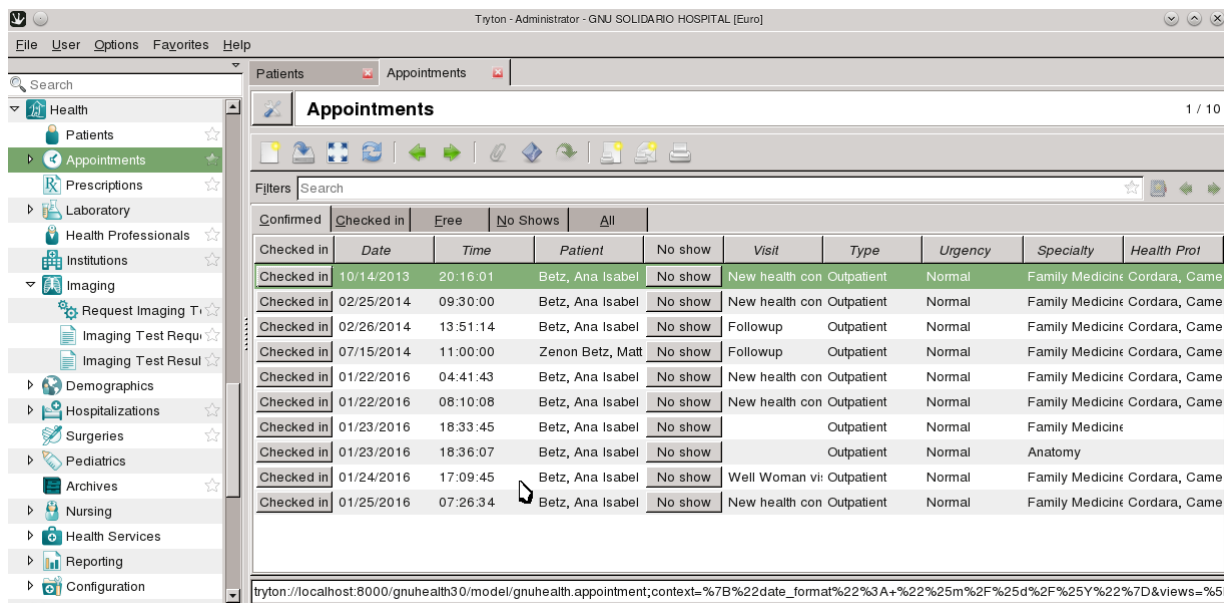


Figure 3.1: User-interface of GNU-Health's client application

GNU Health client does not provide off-line support, i.e., interactions made in the client while there is a disconnection to the server are lost. There is a workaround by deploying a GNU Health server on each client machine, synchronized in centralized mode with the main server. On the other hand, replicating sensitive data on to each client device would raise complexity, security and privacy issues and would also require different and more expensive hardware than the one needed for running just a GNU Health thin-client.

By using Tryton as its core, GNU Health software model is made alike to the Tryton three-tier application architecture composed of a client-server model supported by a DBMS. In practice this involves three main applications: 1) the Tryton-server, instanced as GNU Health server application; 2) the supporting DBMS, generally PostgreSQL; 3) the GNU Health client application, a generic Tryton thin-client.

The server application is cross-platform, meaning it can be run on different operating systems given that there is a compatible Python interpreter present. The Tryton framework provides a platform that enables Tryton thin-clients to fully interact with the server which means a generic Tryton client application can connect to Tryton server applications independently of the type of the server application. Therefore by connecting a generic Tryton client application to a GNU Health server, the client displays the associated GUI supplied by the latter, as it was designed and intended server-side. The communication between client and server uses the JSON-RPC protocol over HTTP, which does not provide a secure transmission of data, thus, an alternative method should be used to provide a secure channel between both parties.

The GNU Health project provides the official Tryton client, part of the Tryton project, as its own client application. It is also cross-platform, written using the GTK+

graphical toolkit for its graphical user interface (GUI) interface. The GTK+ toolkit libraries can be compiled for GNU/Linux, Mac OS and Microsoft Windows operating systems, thus the client application is generally more suited for office desktops. There is also a Tryton client application developed for Android, consequently enabling on-the-go and instant access to GNU Health servers for mobile users. A web-interface provided by the Tryton framework was recently integrated into GNU Health's server application. Consequently a greater number of different devices can now interact with it by using a compatible web-browser [17]. The web user-interface presentation is very similar to the one on the official GNU Health client application.

The development of the GNU Health project is very active, two major versions of the server were released in 2015, and another one in 2016. This success is owned in part to Tryton framework since its development directly contribute to GNU Health's improvement. It is also owned to the GNU Health's strong community which put an active effort to improve and translate the documentation and create third-party modules [18]. The documentation contains user, developer manuals and installation guides in many languages. There is also an open on-line process for translating its interface to other languages [19]. A mailing list is available for announcements and support in different languages, which can also be provided through the project's IRC channel [20].

3.3 OPENEMR

OpenEMR ³ is a electronic health records software, medical practice management software and enterprise resource planning software mainly supported by OEMR ⁴, an non-profit organization which has as a goal to ensure that all people have access to high-quality medical care through its software and services. The project accepts monetary and development contributions, which already attained OpenEMR with Office of the National Coordinator for Health Information Technology (ONC) certifications, the latest released version being certified as *2014 Modular Ambulatory EHR* [21].

It is licensed under GPL and is one of the most popular open-source software in its area, with many success stories around the world, ranging from small clinics to district hospitals [22].

OpenEMR comes with a numerous amount of functionality for covering a wide range of areas of a HIS besides patient records and management, including support for billing and claims management, ERP and information management.

Regarding patient records, clinical observations can be made through a variety of forms already provided. Captured observations on different visits of a patient can be

³<http://www.open-emr.org/>

⁴<http://www.oemr.org/>

presented later through graphical charts or tables showing changes in values along time. Free-text type notes can be attached to a patient's encounter, and medical issues registered in a flexible system of coding which supports CPT, HCPCS, ICD-9, ICD-10 and SNOMED CT codes. In the prescription management area, doctors can register prescriptions and their frequency, which can then be printed, faxed, or sent electronically to a few supported third-party platforms. This can be complemented with the pharmacy dispensary module which enables in-house drug dispensing with support for inventory, integration with drug databases and stocks tracking. In the area for patient's history, medical procedures, immunizations and laboratory tests made can be registered along with results.

Medical billing includes sales and patient ledgers with support for medical claims, insurance management and tracking.

OpenEMR provides a robust calendar sub-system which is capable of handling different kinds of events and assists many other sub-systems of the platform. Some examples of features assisted by it are: appointments and provider scheduling on patients management sub-system, or automated generation of reports on the clinical decision rules sub-system. The latter is a flexible sub-system on which rules execute actions based on different parameters, conditions or frequency. Subsequent actions can be notifications, alerts or report generation. With this sub-system it is possible, for example to automatically send an e-mail reminder to a patient each time he needs to take his medication, or, to automatically run reports of weight assessment for children and adolescents each month.

OpenEMR provides a native portal aimed at use from patients and other kinds of users and also integrates support with other external patient portals. In practice, this means that OpenEMR can provide external access to its information for instance to allow patients to view their medical history or even make payments or doctors to edit patients information, prescribe new medications and communicate with the patient through the portal, among many other uses [23].

The report sub-system comes with an extensible profile of reports that cover many of the platform's different areas. Some of these are configurable, for instance, patient reports can include medical history, encounters, billing, communications or other patient-related information. Adding custom reports to the OpenEMR platform is possible through the same method as the one used by the pre-loaded reports. These are built with PHP programming, for which a specific PHP template for coding reports is provided. There is not an alternative method to create custom reports and as a consequence, programming knowledge and other advanced computer skills are required for the creation of new reports.

OpenEMR comes with several forms designed for use during patient encounters. Creating custom forms is also possible and there are several methods to achieve this:

1) through layout based visit forms; 2) through nation notes; 3) through Extensible Markup Language (XML) forms [24].

Layout based visit forms are built by specifying each field and its data type directly. Forms built with this method can only be used in association to a patient's encounter. This feature automatically tries to lay the needed structures on the persistence side, i.e., create entities, fields and associations (e.g. with a patient's visit), however it is a crude way and is not well supported, as the project states that forms are lost between re-installation or upgrades of OpenEMR [25].

Nation notes brings a "What You See Is What You Get (WYSIWYG)" editor with a template engine built on top of the layout based forms module, thus inheriting its drawbacks [26].

XML forms is another way for writing forms using the XML format, following a given form specification. The forms are integrated with the user-interface and report sub-system by using a separate form generator application that takes the XML form as input and generates SQL code that creates the supporting structures on the database and PHP code that uses the Forms API to. This method is less straightforward than the previous ones, however, provides a way for reusing forms in other OpenEMR platforms by sharing the XML files that defines them [27].

For interoperability, OpenEMR supports the HL-7 standard which is used for exchanging patient data between other systems and the ANSI X12 Electronic Data Interchange (EDI) standard, mainly used on billing and invoicing in the ERP. It is also HIPAA compliant, which focus on ensuring the privacy and security [28] of this data. There is no native support for the DICOM standards, nevertheless, there is at least one project that demonstrates practical interoperability of DICOM in OpenEMR via Mirth Connect⁵ [29].

OpenEMR has had many security audits from independent groups which have found a great number of vulnerabilities [30]. OpenEMR project gives importance to security by releasing information and fixes about vulnerabilities and by having security recommendations and assessments [31]. OpenEMR provides encryption and decryption for external documents, which can then be associated to internal categories or patients [32]. However, support for digital signing of documents in the platform is not available.

Access control management in OpenEMR uses Access Control Lists (ACL), where each resource or feature has a list of permissions attached [33]. In ACL, access is granted for a subject to handle a particular object if the subject is associated to permissions of that object. In OpenEMR this subject is either a user or a group, on which users can be members of and inherit their permissions. If users are given access only based on these groups then access control follows a RBAC model [34]. Each resource features a list of permissions which can be assigned individually to groups or associated directly

⁵<https://www.nextgen.com/Interoperability/Mirth-Solutions/Connect-Overview>

with users. For instance, a user is allowed to create or edit notes if the *write* permission of that resource is associated with him, or, if the user has membership to any group containing that permission.

The OpenEMR platform has support for more than 20 languages and is compatible with UTF-8. The display language of the user interface is user-dependent, meaning that multiple users can use the platform simultaneously in their desired languages.

User interaction with OpenEMR is done via a web-interface. The user-interface arranges active content in a dual panel mode. The first panel is always shown and is used to display primary content of the active area or context. The second panel either shows a given global functionality of the active context such as past encounters and documents while browsing the patient's context or is used to show selected content in addition to the first panel. This proves useful for showing related information side-by-side, but quickly builds up complexity in the interface. Some functionality also opts to open a separate frame inside the content panel, or even a new pop-up window to display or gather information, again, adding complexity. The top section of the user-interface displays the current active patient (even if the active context is not the patient) and a menu with encounter operations for that patient which is redundant since these are also present on a global menu, under the visits name.

The content shown on both panels is defined by item selected on a global menu present on the left side. The menu aggregates features into a tree-style structure, color coding the parent and child in different background colors, which happen to have the same background color of items of upper levels, and presenting items that cannot be operated by the user in a lighter shade of gray. For the interface to enable this items or operations, the user is required to previously make a selection of a given entity in the content panel. This adds unnecessary complexity to the menu since these are dependent on the selected and shown content, but despite this, they are part of the global menu and always present (see Figure 3.2).

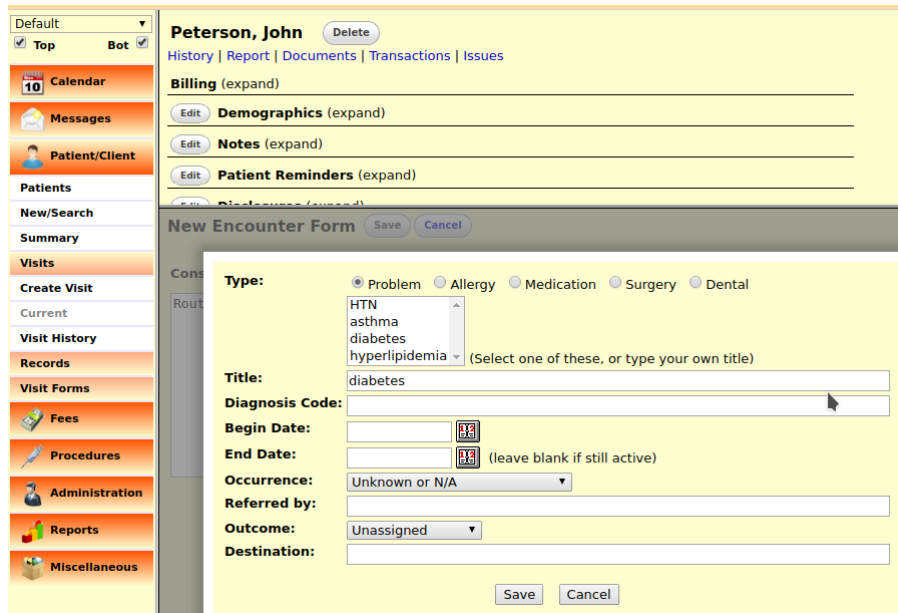


Figure 3.2: User-interface of OpenEMR’s platform

An alternative method to provide a more clear menu would be to move these operations from it to the context itself where they can be called from, e.g. move the patient visit operations from the menu to the content panel that shows the patient. The menu also does not have item filtering by a text query, so a user needs to manually navigate through the tree-structured items to find the desired one. These issues, inconsistencies and the overwhelming presentation of the user interface adds complexity to the user experience.

OpenEMR does not provide off-line support, for instance, users cannot navigate on the web-interface while a connection to the server is unavailable. This may pose a bigger problem on intermittent network connections since there are no warnings when the connection is off-line and can result in loss of entry data.

OpenEMR is written in PHP and runs in a compatible PHP runtime, PHP 5 as of the reviewed version (4.2.1), with the most recent version already supporting PHP 7. PHP provides OpenEMR with a scripting language that is widely supported and has cross-platform run-times. The use of web technologies makes the platform accessible through many different devices, provided they have a compatible web-browser. OpenEMR uses the ADOdb PHP library which allows transparent usage of many different DBMSs, however OpenEMR install manuals suggest using MySQL. The ACL is provided by the PHP library phpGACL, which is used in all OpenEMR subsystems.

Despite its software architecture not being designed in a truly modular way, new features can be integrated with OpenEMR by writing a native PHP module or by using the PHP framework Zend, which OpenEMR integrates support with [35]. OpenEMR provides an API for accessing its functionalities, however not all is documented or

accessible [36].

OpenEMR project development is active, with an average of two major releases per year. The project has a strong community of users and companies backing, with many OpenEMR modules developed by third-parties and contributed to OpenEMR which now make part of the project. The on-line documentation has extensive user and implementation guides, however the documentation for developers is lacking behind. There are a number of companies providing professional support in many different countries, some of which are certified by the OpenEMR project itself [37]. The OpenEMR project has an Internet Relay Chat (IRC) channel [38] and a community on-line forum⁶ for users or implementers to get help and support and for development discussions. Several on-line demos of the software are officially provided by the project [40].

3.4 FREEMED

FreeMED is an electronic health records software and medical practice management software by FreeMED Software Foundation⁷, a NPO whose primary goal is the betterment of open-source software community and the world in general through promoting development and adoption of FreeMED and other open-source medical software projects [41].

The software is licensed under GPL.

FreeMED is by default an EHR mainly designed from the health-care practice standpoint and generally aimed at health-care providers, registration secretaries and administrative professionals. Its offered features come under the categories: patient, documents, billing, system, reporting and utilities. Independent and inter-operable modules give support for both static and dynamic portions of these.

A patient record in FreeMED contains personal and demographics information, and information about insurance type and coverage. Patients can be assigned to multiple health-care professionals such as family doctors or other types of care. Pre-loaded form templates are provided which can be used for collecting clinical observations during patient visits. Medical issues support is provided by the diagnosis family module and can be coded using CPT and ICD codes. It is possible to assign medications to a patient and manage a registry with them. Patients can be arranged in groups which are assigned to doctors or referenced in programs. There is an area for triage and call-ins useful for registration secretaries. These can be complemented by the appointment scheduling sub-system. By default there are appointment templates for either patient consultations or other kind of services. Appointments can be scheduled for either an

⁶<https://sourceforge.net/p/openemr/discussion/>

⁷<http://freemedsoftware.org/>

individual or groups of patients and the interface has safeguards for double-booking, presenting warnings in case that two patients get booked for the same provider at the same time.

FreeMED offers document management support which can be also used for managing external documents, e.g. reports or medical images in formats not supported natively by FreeMED like DICOM [42]. FreeMED has an internal messaging system that allows different users to post global messages or communicate between themselves.

Regarding PMS functionality, FreeMED has integration with an extra application called REMITT ⁸, a billing and remittance package. REMITT is also from the FreeMED Software Foundation and under the GPL license. It has support for professional EDI standards like ANSI X12 and automatic insurance claims generation of HCFA-1500 forms, while being HIPAA compliant [43].

The intercommunication between FreeMED and REMITT is done via a remote procedure call protocol, in this case the XML-RPC, and is transported over HTTP. The protocol provides access management through *basic authentication*, but not secure communication. This can be attained by using HTTPS instead of HTTP as the transport protocol, which provides security by using Transport Layer Security (TLS) at the application layer.

FreeMED uses HL-7 for the exchange of medical data. This support is obtained via integration with Mirth Connect, a cross-platform HL-7 interface engine.

FreeMED comes already with some configured reports, and adding new ones requires related technical skills and also knowledge of the FreeMED data model. The reporting subsystem is dependent on the Java library JasperReports for report generation. JasperReports is accessible from FreeMED through a wrapper called JasperWrapper, which allows FreeMED to call JasperReports functionality from within the PHP runtime. There is no documentation about either the process or support for adding custom forms to FreeMED. An official module which claims support for custom forms exists, however a brief analysis of its source code did not reveal to which extent it supports them [44].

User interaction with the FreeMED platform is done through a web-interface. After language selection and user-authentication, different functionalities are displayed depending on the user's access clearance. Access is controlled with ACL in the same way described previously for OpenEMR, with implementation also making use of the phpGACL library.

The user-interface has an always-present menu which groups functionalities in categories. The active functionality is displayed on a content panel to the right of the menu, and while several items from the menu can be opened at the same time, only one is displayed at a moment, the remaining items being grouped into tabs that keep its current state in the background without interrupting the user experience. The

⁸<http://remitt.org>

content has a clean presentation, and while some more complex features may show different functionality grouped into their own tabs, there is an option for choosing the arrangement between tabbed and flat presentation.

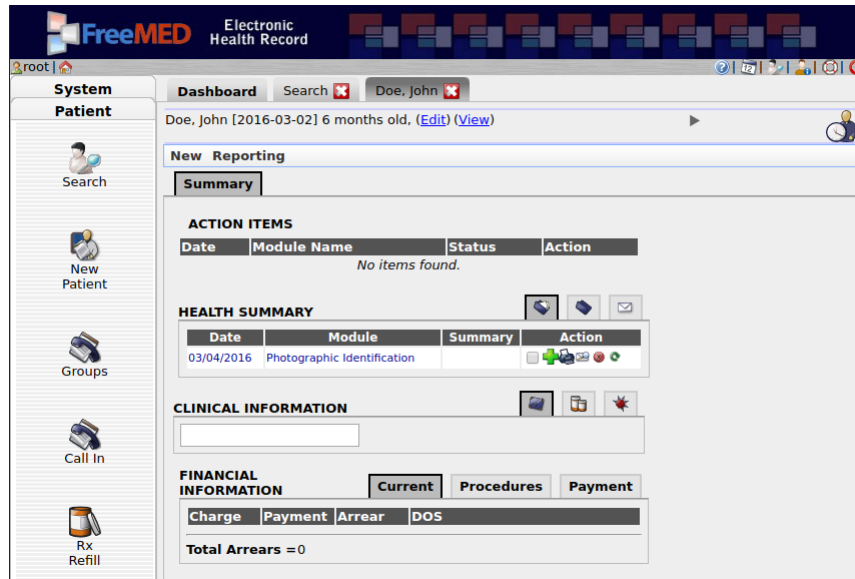


Figure 3.3: User-interface of FreeMED platform

FreeMED is written mostly in PHP and Perl. Additionally, it requires a JAVA Runtime-Environment (JRE) for the report subsystem and a supporting JavaEE web application server for REMITT as they are written in Java. Since all the technologies used in the software are cross-platform, FreeMED can be run in different operating systems, ranging from POSIX-compatible like Linux or BSD to Microsoft Windows and others. The software is designed in a client-server application model. The platform does not provide off-line support, so if the communication is interrupted between client and server, users cannot operate the platform from the client-interface and need to wait for the re-establishment of the connection. In the same way as OpenEMR, access to the database is made through the ADOdb PHP library, which allows transparent usage of many different DBMSs, nevertheless, there are parts of the code base that are not DBMSs agnostic and thus, the project recommends using a compatible MySQL DBMS.

FreeMED has a modular architecture, which is designed with functionality and interface separation. This allows the platform or the interface to be customized without having to rewrite core components of the system, and to group functionality in modules, allowing to bundle new functionality into new modules.

FreeMED supports localization and the project claims multiple languages support besides English, including German, French and Polish. There is also an on-line public project for contributing translations, which already attained complete localization in Portuguese and Russian languages [45]. We could not test other user-interface localizations besides English, probably as a result of a software bug. FreeMED supports

multiple ISO character-set formats and stores entry data with it, making it possible to maintain demographic database in one language and ISO set independently of the localization that displays.

The source code is available on an on-line public repository website, which also provides a brief installation guide and a Virtual Machine (VM) image of FreeMED for demoing. There are no development guides and the project lacks proper documentation overall. There is also a FreeMED demo available on-line, but there are no user guides neither in the official mediums neither in the FreeMED software itself. FreeMED advertises a commercial support from Foundation's support partners, although the URL is down as of this writing. ⁹.

There is a community support page for users and developers, although with very low activity on both of the previous subjects. ¹⁰.

3.5 OPENMRS

OpenMRS¹¹ is a electronic health records software lead by two collaborative non-profit organizations, Regenstrief Institute, a world-renowned leader in medical informatics research, and Partners In Health, a Boston-based philanthropic organization with a focus on improving the lives of underprivileged people worldwide through health care service and advocacy [46].

One of the main objectives of the project is to build a robust solution for health-care with special focus on resource constrained environments. There is a worldwide community of volunteers from many fields of activity including health-care, international development and technology, contributing on many levels to the software and project [2].

OpenMRS is designed in the perspective of health-workers, following a typical health-care work-flow: first, patients are registered in the system. A visit is opened for a walk-in patient, afterwards the patient goes for an encounter with health-care providers which register observations and manages the patient (e.g. admits the patient, schedules appointments), and the visit is closed with notes optionally attached.

At OpenMRS platform's core is a centralized dictionary of concepts, a customizable part of OpenMRS that strongly defines each implementation. In the dictionary, concepts are meta-data of given entities, defined by name, data-type, appropriate attributes and relationships to other concepts. These are used as models or latter instanced in some form. For example a value collected during a patient encounter is an instance of a

⁹http://freemedsoftware.org/commercial_support

¹⁰<https://groups.google.com/d/forum/freemed-support>

¹¹<http://openmrs.org/>

given concept that takes the form of an observation. As a model, concepts work as the building blocks that describe forms, orders, clinical summaries, reports and so on. The extent of these is directly related to the richness of the dictionary.

OpenMRS is a highly modular platform. Most of the official features it provides are external to the platform and added through add-on modules. A standardized API plays a major role in this by allowing access to OpenMRS platform functionality, which modules use or extend, providing their own functionality to other modules. The API is accessible directly to installed modules and is also made available via REST, through a complementary module that enables managed access to it [47].

OpenMRS is officially released in two different distributions, one which contains just OpenMRS platform, and another called the *Reference Application*, which packs selected modules that extend and add features to the platform.

The reference application includes a new framework for other modules to interface with the platform. The framework standardizes modules and defines their interface presentation, communication, behavior and security, enabling the creation of an OpenMRS application ecosystem. Through modules OpenMRS extends its platform features, providing appointment scheduling and management, a event system for notifications, a sub-system for allergies registration, representation of patient observations through charts and so on.

OpenMRS has a robust form sub-system for creation of custom forms. Any form relies on a form schema that defines the fields and concepts the form will use, which are supported by the concept dictionary. While custom forms can be created and their form schema can be designed from within OpenMRS legacy UI, the methods for designing the forms are independent of the platform. There are several modules that add these, each using different techniques to achieve the same goal.

One, the Infopath Form Entry or (also) simply Form Entry, allows the design and filling of forms using Microsoft Infopath templates. From a form schema, the module produces a XSN form template that the client downloads in order to design the intended form in an external application, for instance, Microsoft Infopath. The result can then be uploaded back through the module interface, completing the form creation process. A form of this kind is made available as a download in the *Forms* tab of the patient's dashboard (legacy UI), requiring again the use of Microsoft Infopath for data entry and submission [48]. Although the module itself is open source, this method is not favored since it deviates from a fully open source system by using proprietary formats or tools to manipulate it, consequently, other alternatives were sought.

Another alternative, the HTML Form Entry, provides enough flexibility for the form layout by using HyperText Markup Language (HTML) while requiring knowledge of the HTML language[49], which may not be suitable in some cases. There is a WYSIWYG module that produces HTML Forms for the HTML Form Entry (HTML Form Entry

Designer), but we could not get access to its source code and the latest available version for download is not up-to-date since it has hard dependencies on older modules which are not compatible with current OpenMRS versions [50].

An up-to-date alternative, the XForms module, allows building forms based on the XForms standard [51], [52]. The module bridges forms of this standard and OpenMRS forms, enabling the possibility of designing and using forms with external application. XForms Designer provides a WYSIWYG interface from within the OpenMRS user-interface itself for designing forms for XForms [53]. The form's model and layout are defined in XML and can be shared to other OpenMRS implementations while keeping the same presentation. An example of compatible external applications are the XForms Mobile Data Collection tools for OpenMRS. They enable off-line form filling provided that the necessary XForms and auxiliary data are previously downloaded to the device. Sending the data entered on the form back to OpenMRS can be done via HTTP/HTTPS, Bluetooth and SMS.

A robust reporting sub-system brings three ways for building reports: indicator reports, row-per-patient reports or custom reports. A report is built through associations to dataset definitions that acts as a key to values to include in the report. The dataset can generally refer person or patient properties and similar attributes that contain only one recorded value stored since its last entry, commonly used on row-per-person reports. Combination of values from multiple observations is possible, and it is used for building indicator reports. A report definition can be programmed to filter (cohort) a defined set of patients for use as the input of the report generation. Multiple dimensions can be added, e.g. only males, farmers. A report definition is finished through the use of indicators". These represent logic types or functions to be applied in the report generation process.

OpenMRS provides two user-interfaces with different feature set support, one called legacy UI (Figure 3.4), and another added with the reference application called modern UI (Figure 3.5). One obvious example of the difference in features is the administrative area of OpenMRS available only in the legacy UI. It is used for managing the dictionary, set providers schedules, add patient attributes or create forms, reports, users and so on. Nonetheless, the platform and reference application provide frameworks which enable the interfaces to be extended. The reference application UI framework has design guidelines and support for presenting clean and simple user-interfaces adopting a *one question-per-screen* approach, which the modern UI follows. Many external modules do this by adding features and functionality and extending the modern UI to support those.

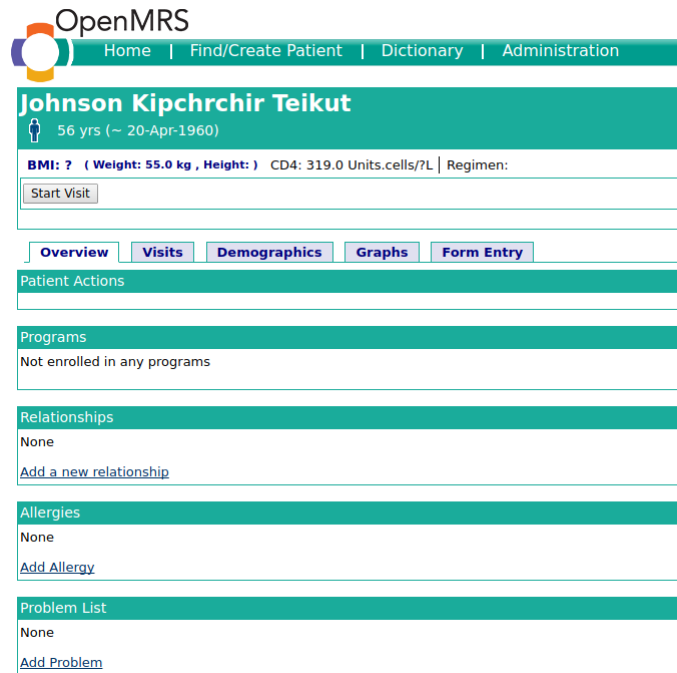


Figure 3.4: OpenMRS’s *legacy UI* user-interface

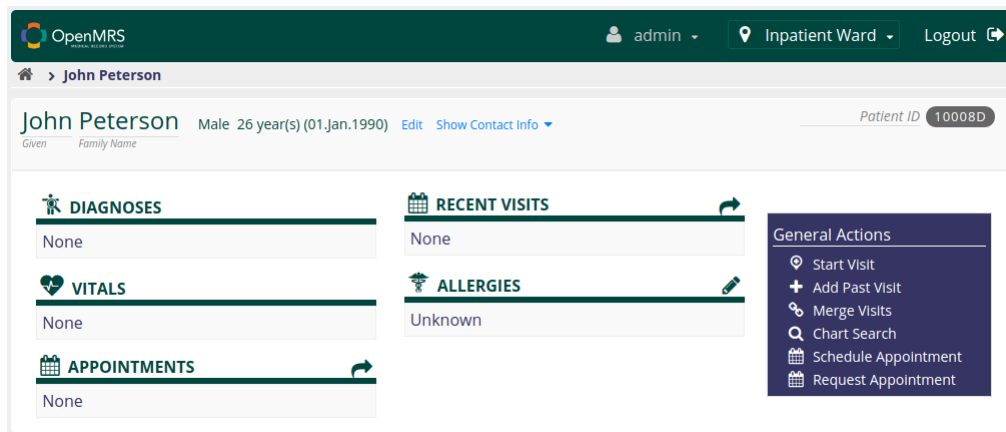


Figure 3.5: OpenMRS’s *modern UI* user-interface

The access management in OpenMRS follows RBAC. This system covers both the user-interface and the access to OpenMRS API via REST services.

Importing and exporting meta-data between platforms can be done either manually or automatically, using the meta-data sharing module. The module is advanced enough to provide automatic synchronization mechanisms, with support for merging and conflict resolution.

OpenMRS has native support for the HL-7 standard and can export its data directly to other applications [54]. There is support for DICOM and IHE radiology ¹²

¹²<https://www.ihe.net/Radiology/>

work-flow standards, via a third-party module called *Radiology Module* in conjunction with *dcm4chee*, a medical imaging archive and manager. Communication between the module and *dcm4chee* is a practical example of the use and support in OpenMRS of the HL-7 standard for exchanging of health-care information with external platforms. Alternatively it is possible to connect it to other systems using Mirth Connect which has native support for OpenMRS achieved via use of the OpenMRS API on Mirth Connect's side [55]. Native support for FHIR is also present natively in the platform since version 2.0 [56]. Other recent development assigned for this version was to decouple OpenMRS legacy user-interface from the core platform and moving the respective functionality to separate modules, which enables the possibility to distribute OpenMRS as a more *true* platform [57].

This version OpenMRS also added support for running on JRE 1.8, where previously it required JRE 1.7. OpenMRS runs on an compatible web application server, e.g. Tomcat or Jetty, alongside a compatible MySQL RDBMS for persistence support, cross-platform technologies which allow OpenMRS to be run in GNU/Linux, Microsoft Windows or other operating systems where they are supported.

Officially the project provides an extensive *wiki* which features content for developers, implementers, and end-users. Two on-line demos are provided, one of the reference application and another which is a customized implementation of OpenMRS at for management of drug-resistant tuberculosis can show the level of customization that OpenMRS offers. Currently more than 175 separate modules are available, although compatibility with OpenMRS is dependent on its version because of changes in the OpenMRS API. [58]

OpenMRS has full UTF-8 support and some extra localizations besides English, with possibility to add more. The project provides an on-line portal for collaborative translation of its user-interface, reference application and commonly used third-party modules [59].

All of this makes OpenMRS one of the most promising, customizable and used platforms in the area with many implementations around the world [60]. Other softwares exist which use OpenMRS at a lower-level and provide other implementations and features. Examples of these include Bahmni, Kenya EMR, an EHR officially maintained and administered by the government of Kenya, UgandaEMR, a custom implementation mandated by the Ministry of Health of Uganda and eSaude, developed by a regional community based in Mozambique and built meeting the requirements of MISAU (the country's Ministry of Health) for HIV Care & Treatment and Maternal and Child Health [61].

OpenMRS development is very active on three major branches: 1.9 which is still very popular among implementers, latest release on December 2015, 1.11 and the most recent, 2.0 [62]. The maintenance of older versions occurs mostly for security

vulnerabilities fixes, for which OpenMRS also announces through its security advisories [63]. OpenMRS is licensed under MPL v2.0 with an additional disclaimer of warranty and limitation of liability (for United States or other jurisdictions where they may apply) which does not alter its FLOSS nature [64].

3.6 BAHMNI

Bahmni ¹³ is an hospital system for low resource settings built on top of OpenMRS. It incorporates most of the OpenMRS platform functionality and extends it with support for PACS, laboratory information system (LIS) and ERP features. This is done via integration with three separate applications: dcm4chee, which provides the first, OpenELIS, which provides the second and OpenERP (currently named Odoo), which provides the third. Bahmni provides a new user-interface and features for EHR in a system called Bahmni EHR. All together these systems form Bahmni.

Bahmni is mainly developed by ThoughtWorks Global Health ¹⁴. ThoughtWorks Global Health's focus is to improve quality and expand access to care in low-resource settings and works alongside those who stand in solidarity with the poor and oppressed to eliminate health disparities.

In Bahmni EHR, features come distributed under several areas: patient registration, clinical services, inpatient management and reporting.

In a registration process, information like identity, demographics, photograph, contacts and other socio-economic details of a patient can be captured, and associated to a person via a new generated patient identifier. A sub-set of this collected information is used for printing patient ID cards, which can be handed to the patient for his next visits. Hand-outs also be printed containing details of the health-center like contacts, and can be customized to have also some life-style or useful recommendations. Different types of relationships can be registered, including genealogical or patient-provider relationships. When a patient does a registration in an institution, there is a strong possibility that he already has health-related documents stored on other institutions, like record scans, x-ray images or other types. Bahmni EHR gives support for storing these within the system and attaching them to a patient's profile. The profile of a patient is displayed through a modular dashboard, where customizable widgets display most recent information from several different categories of the patient's medical records. These can be for instance laboratory tests and results, programs enrolled, last visit observations and so on. From observations made in several encounters, history of changes and graphs can be presented.

¹³<http://www.bahmni.org/>

¹⁴<https://www.thoughtworks.com/global-health>

Clinical services in Bahmni EHR are centered around the patient's profile. These services become available in the consultation area of the dashboard after a patient is active in the system, i.e. he is in a given encounter with a provider. From here, several kinds of services are provided: capture observations, make laboratory or radiology orders, register diagnosis and medications, manage patient disposition or take consultation notes.

Regarding clinical observations, Bahmni already provides multiple forms by default. The forms sub-system is very customizable and can easily be extended with more forms. These can be built by creating and changing specific concepts of the dictionary. The form's layout and other parameters can be defined in JSON and, if necessary, there is support for integration with a server-side framework for advanced customization of a form's logic and other aspects. Current and past diagnosis can be associated to a patient and registered along some attributes like order, confidence and status. Diagnosis made can also be mapped to ICD-10 codes.

In regards to laboratory orders, Bahmni provides a panel of tests divided in several categories. The laboratory sub-system is flexible, allowing for choosing just the needed tests, categories or arrange them all together. New laboratory tests can be easily created and added just by using the concepts dictionary.

In regards to radiology orders, integration between Bahmni EHR and a PACS is assisted by dcm4chee. Radiology orders containing patient and the investigation details can be requested within Bahmni EHR, which are sent to a modality via acsHL-7. The results are integrated with the patient profile even if they come as native unsupported formats, on which references to them are shown. For instance, results that come as DICOM images are archived in an accessible PACS and referenced from the patient profile, from which they can be opened in compatible DICOM viewers.

Management of medications include registration of past and current medications from customizable drug registry which has route, frequency, dosage and instructions for each. The drug registry provides sensible defaults for medications wherever possible, nonetheless medication parameters can always be overridden. There is a drug dispensing service which is integrated with OpenERP and supports in-house dispensing with segregation of stocks. Patient disposition refers to a patient's state usually set at the end of a provider encounter. For instance a disposition of a patient that has a severe medical issue and needs to be hospitalized can be set as admitted or patients on that state but now recovered may be set as discharged and so on.

While using services from the consultation area, a background sub-system assists the user by automatically creating encounters whenever necessary. However, for starting the consultation itself, a visit needs to be opened previously, which makes sense but was not executed properly in Bahmni EHR. A user can't start a consultation from the patient's dashboard if an active visit was not opened previously, and needs to navigate

to the patient registration form to register a visit. A suggestion to avoid this issue is to enable patient's visit operations from its dashboard since it is from there that consultations are also started.

OpenERP is used predominantly for sales and purchase management including inventory and accounting. These are integrated to some extent in Bahmni EHR, for instance registration fees or payment of drugs dispensed under Bahmni EHR are registered under patient billing in OpenERP and track of stocks is kept.

The reporting area comes with many pre-loaded reports, which can be run within a given date period and be exported in CSV, PDF, HTML or Excel formats. Additionally the platform also provides integration with JasperReports, giving support for custom, flexible and more varied reports.

The administrative functionality in Bahmni EMR is very basic since it only allows for customizing an implementation by importing files with specific concepts of several domains to the dictionary. There is support for the reverse operation, exporting to a file, which gives at least a easy way to share customizations between instances of Bahmni EMR. Nevertheless, the remaining administrative functionality can be done from OpenMRS legacy UI.

The access control on Bahmni EHR is provided by the OpenMRS platform. However there are new access roles associated to the new functionalities and services of Bahmni EHR. Additionally, user access can now be set to use two-factor authentication.

Bahmni EHR supports off-line network alerts by displaying a notification in the user-interface when the connection is down. This is helpful in order to avoid loss of filled-in data by submitting it unknowingly when a connection to the server was down. True off-line support is also provided via a browser extension or a mobile application, offering the possibility of filling forms and other user-interface functionalities without an active connection to the server. The data produced with this functionality is saved locally on the device and can be uploaded at a latter time.

Bahmni EHR does not provide support for appointment scheduling or management. This functionality can be achieved by installing the appointment scheduling of OpenMRS which integrates with its own user-interfaces. However, this is a workaround and poses some problems like user-interface consistency and navigation issues since the UIs are not linked to each other.

Bahmni is built through separate OpenMRS modules and its user-interface communicates with these using the OpenMRS API via REST. The integration between platforms is done in order to share data and create a flow between them, resulting in less work than needed in creating a new platform. All these software are open source, Bahmni is AGPL licensed, includes a customized OpenELIS version that was forked from version 3.1 and is under Mozilla Public License 1.1, and the particular OpenERP version used is a Community edition which is under AGPL.

Bahmni is distributed through either a VM or via software packages for CentOS GNU/Linux distribution. Although CentOS is currently maintained under two branches, version 6 and 7, Bahmni requires other open-source technologies currently under version 6 and thus recommends the use of this branch instead [65] (n.b. development is progressing for making it compatible with version 7) [66]. The documentation is well organized, with different sections providing feature, installation and setup guides. There are manuals for implementers, users and developers. The community is closely tied with the OpenMRS community as both are under the same forum and are very active [67]. The same applies to the documentation, which is under the same *Wiki*. The project also provides an IRC channel. Bahmni EHR is multi-lingual, and has near-complete support for Spanish and Portuguese languages. There is an on-line platform that allows to collaborate in translating Bahmni to other languages and accepts community contributions [68] Diverse health-care institutions including district hospitals make use of Bahmni, which already counts with many implementations in India, Nepal and Bangladesh, the latter even adopting it for use in Ministry of Health [69].

On-line demos are provided from the official project.

3.7 SOFTWARE COMPARISON

From the software evaluated, we summarize some aspects in the following table from the perspective of EHR. There are more characteristics that could be shown but we mainly considered the ones who differ between the software.

Table 3.1: Matrix with several aspects of the evaluated EHR software

	GNU Health	OpenEMR	FreeMED	OpenMRS	Bahmni
Integrated applications	EHR,HIS	EHR,PMS,ERP	EHR,PMS	EHR	EHR,PMS,ERP,LIS,PACS
Configurable reports	YES	YES	NO	YES	YES
Custom reports	NO	NO	NO	YES	YES
Custom forms	NO	YES	NO	YES	YES
Data exchange	FHIR,custom	HL7	HL7	HL7,FHIR	HL7,FHIR
Coding systems	Few	Many	Few	Many	Many
External auth. methods	LDAP	LDAP,AD	-	-	-
Patient portal	NO	YES	NO	NO	NO
Access control model	RBAC	ACL	ACL	RBAC	RBAC
Cryptographic features	Sign, encrypt	Encrypt	-	-	-
Flexible data model	NO	NO	NO	YES	YES
Modularity rank	★★★	★	★★	★★★★	★★★★
Off-line support	YES	NO	NO	NO	YES
Native client	YES	NO	NO	NO	NO
Web client	YES	YES	YES	YES	YES
Other clients	YES	NO	NO	NO	YES
User-interface rank	★★	★	★★★★	★★	★★★★
Code-base language	Python	PHP	PHP	Java	Java
Development progress	Active	Active	Slow	Active	Active
Community support rank	★★★★	★★★★	★	★★★★	★★★★

User-interface refers to personal evaluation of the software primary interface made with other team members. Clean presentation, clear arrangement of navigation and issues present were taken in consideration for the rank. Modularity refers to the software modularity, i.e., if features are hard-coded; if it provides a rich API and well documented API. Community support is ranked according to forum activity and other contributions to the project, for instance translations or third party module availability.

From the evaluation and comparison of EHR software, Bahmni proved to be the most appropriate candidate for use in our low-resource setting. From its overall characteristics, the key aspects for choosing Bahmni were the solid facility it gives for creating observation and laboratory forms, report configuration, off-line support and the integration with a laboratory information system. Since Bahmni integrates other services with OpenMRS and is possible to integrate other OpenMRS features in a Bahmni solution, this made it a more advantageous choice than OpenMRS itself; hence, Bahmni is overall the solution with most potential for our scenarios.

SYSTEM REQUIREMENTS

This chapter describes the system requirements for the Gondar EHR system of Leishmaniasis Research and Treatment Center (LRTC). It covers overall description of LRTC, a routine work-flow, functional requirements as well as non-functional requirements, a use case diagram and a specification for each use case.

4.1 SCOPE

The LRTC is a part of Gondar University and was established in 2004. The center provides medical care for patients infected by leishmaniasis parasite, carries clinical trials and has its own laboratory. Moreover, the LRTC has a bed ward, which can accommodate up to 24 patients. The team consists of 15 employees, including physicians, nurses, pharmacists, and laboratory technologists.

The center was visited in early February 2016 with the main goal to gather user requirements, interview the clinical staff, check available equipment and verify infrastructure. A routine work-flow of LRTC was observed and analyzed in order to implement the Gondar EHR system. As a result, the functional requirements of the application were established by the end of the trip. The objectives of the Gondar EHR system are to provide clinical staff with cumulative health history of patients, integrate LRTC with its laboratory and generate reports.

During the observation process and interviewing it was identified that the present work-flow was paper-based and there was a need in the digitization of patient records. Another problem was faced in communication between laboratory and clinic, where

two LRTC divisions were transitioning their paperwork to each other. Moreover, the LRTC clinical staff prepare periodic reports for governmental and non-governmental organizations such as World Health Organization, Drugs for Neglected Diseases initiative and others. This manual process involves coping data from paper-based registration books to an excel spreadsheet, use an appropriate template and send by email. There is also some resource management done in assigning beds for inpatients.

4.2 FUNCTIONAL REQUIREMENTS

The identification of actors and use cases play an important role in establishing requirements. There are four actors which make use of the Gondar EHR System: a physician, a nurse, a lab technologist and a system administrator. A physician is the main actor who is in charge of patient treatment. He also examines a patient, makes a diagnose and prepares reports. The next actor is a nurse, who is responsible for patient registration, physical examination of vital signs and making orders for laboratory. A lab technologist tests samples such as blood and skin tissues, then provides results to the system. An administrator is non-clinical staff member, who maintains the system.

The use case diagram below (Figure 4.1) has been created to illustrate the interactions between a system and its environment, showing the functionality of the system from the perspective of each actor.

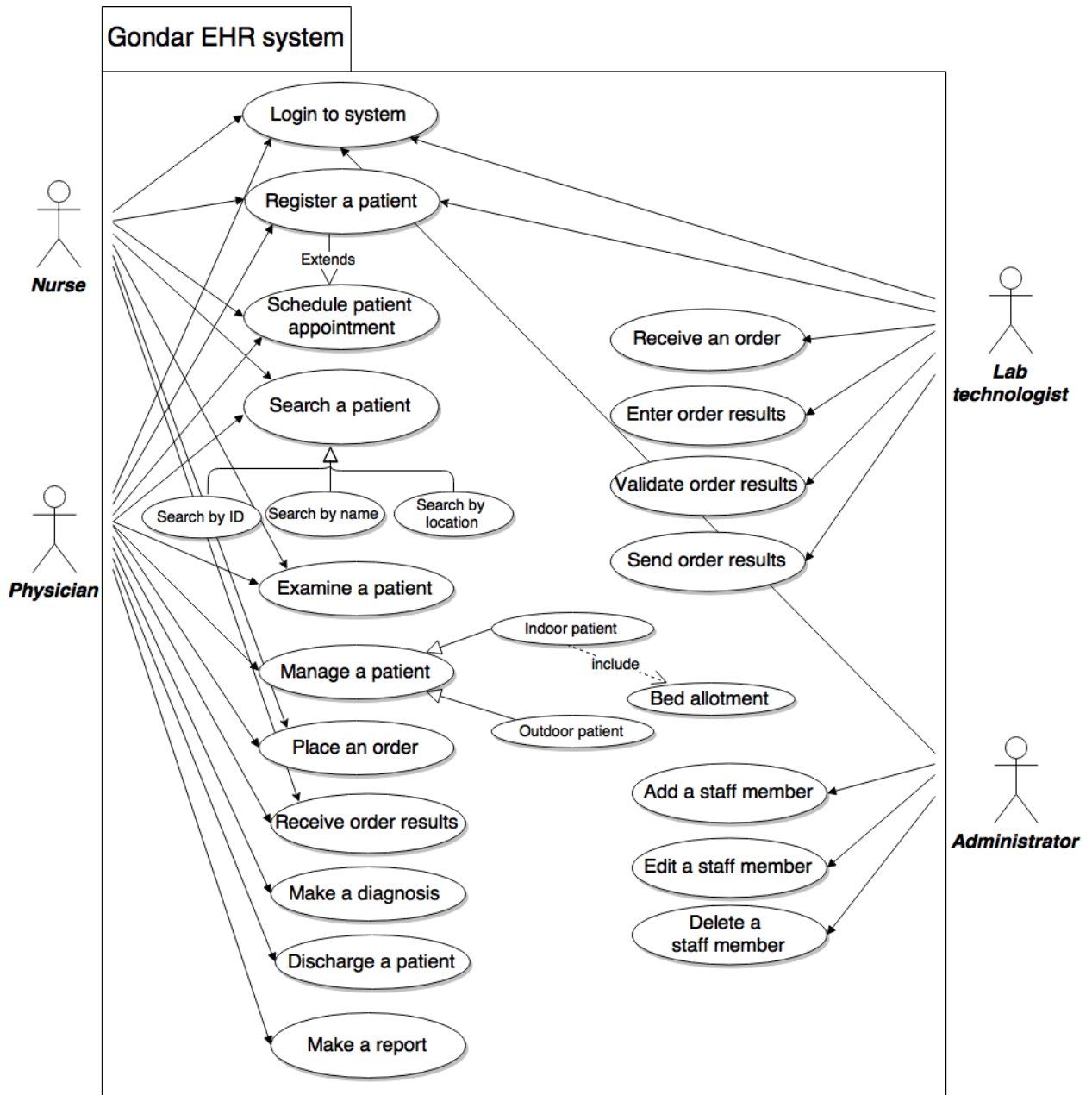


Figure 4.1: Use case diagram

The textual models that capture the requirements in context are depicted in the tables 1 to table 18 in the appendix.

From these, functional requirements were drafted. Summarizing, a system for Gondar LRTC must:

- A. Manage patients
 - A1. Register a new patient. The system should capture the following demographic information:
 - * A patient must have a unique medical number within the system

- * Patient's first name, surname, name in native language (not mandatory)
- * Sex, date of birth, place of birth, physical address, telephone contact
- * Biometric parameters as height, weight, blood pressure, BMI (calculated).
- * Allergies on medicine or products
- * Social history elements: marital status, occupation, socioeconomic status, and education.
- * Hospital visit dates, admission, discharge dates (if applicable), chief complaint, diagnosis, procedures performed.
- A2. Edit a patient information
- A3. Search a patient
 - * A3.1. Search patient by medical number, names.
- B. Out Patient Department (OPD)
 - B1. Consultation chambers
 - B2. Examination rooms
 - * B2.1. Send test request to the laboratory
 - * B2.1 Receive/have an access the laboratory test results.
 - B3. Diagnostics
 - * B3.1. Laboratory must receive requests from examination room
 - * B3.2. Lab assistant can create requests other than from LRTC
 - * B3.3. Lab assistant can enter test results into the system
 - * B3.4. Test result must be verified by lab staff member
 - * B3.5. When test results have been verified, they must be available
- C. In Patient Department (IPD)
 - C1. Set total number of beds grouping by children, men and women.
 - C2. Change patient status from outdoor to indoor and vice-versa.
 - C3. Show number of occupied and available beds.
- D. Alerts and appointments
 - D1. Create an appointment with a physician (up to 6-9 months)
 - D2. Send alert messages to a patient before 1-3 days of scheduled visit
 - D3. Edit an appointment
 - D4. Delete an appointment
- E. Reports
 - E.1. The system should build reports with set period. (The list of reports must be received by LRTC staff.)

- F. System logs
 - F1. The system must track dates and time stamps all entries.
- G. Confidentiality and Security
 - G1. System should support secure logon into the LRTC system.
 - G2. System should provide analysis of audit trails and unauthorized access attempts.

Non-functional requirements were also drafted:

- The user-interface should be simple to use.
- The system should be secure on the network level.

SYSTEM ARCHITECTURE AND IMPLEMENTATION

To address the system requirements present in the previous chapter, we implemented a solution which uses a subset of Bahmni software. We show the architecture, implementation, installation and configuration of the system.

We tried to satisfy the requirements by customizing and adapting the systems of Bahmni. Since there was no need to develop new functionality, this section will not focus on source code from the Bahmni project.

The features and model for EHR are implemented with Bahmni EHR. Three user-interfaces are available. Bahmni EHR provides its own, called Bahmni MRS. In addition to it, the two OpenMRS user interface (UI)s are also used: legacy UI, for the platform's administration and the *modern UI*, for the appointments management. The laboratory system requirements are satisfied by use of OpenELIS and through integration with Bahmni EHR.

The OpenERP platform, dcm4chee and PACS integration services, part of Bahmni, are not used in this implementation since they are not needed per system-requirements and therefore are not contemplated in the architecture overview. Nonetheless, the resulting solution is still compatible with these software components so they can be added on at any time if required. Since Bahmni EHR uses OpenMRS as its underlying platform this thesis will contemplate a in-depth analysis of OpenMRS architecture in this chapter.

The operating system chosen for server side is version 6.7 of GNU/Linux distribution CentOS¹, a community spin-off of RedHat Enterprise Linux² for maintenance, stability and security [70].

5.1 SOFTWARE ARCHITECTURE

The Bahmni EHR architecture can be represented as an extension of the OpenMRS three-layered architecture. The presentation layer has three separate user interfaces, written with different motives and needs of the OpenMRS and Bahmni projects development, each providing a overlapping and an exclusive set of features from the others. Success in satisfying the system requirements is obtained through the combination of the three user interfaces' resulting feature set, thus, for this reason, all the three presentation interfaces are used in our given solution. The diagram in Figure 5.1 shows the three-layered architecture and OpenMRS module architecture-layer integration. ³

- **Data Layer:** In the lower-layer is the Bahmni Data Model, a customization of the OpenMRS data model for Bahmni EHR. The data model is a relational model represented in SQL and supported by the MySQL RDBMS, in which it is instanced under the database name 'openmrs'. The OpenMRS platform and Bahmni EHR projects use the same technology for keeping the data model compatible between released versions, the Liquibase library. The integration of new changes on the data model is assisted by Liquibase that keeps the history track between schema versions, thus enabling older OpenMRS platform database instances to be updated for compatibility with a new OpenMRS platform, the same is applied between Bahmni EHR database schema versions.
- **Service Layer:** Present in the middle-layer is the logic of the OpenMRS platform' customized for Bahmni. It is supported by the Spring Framework⁴ and runs on a Java EE-compatible web application server. The middle-layer communicates with the database-layer in an object-oriented way through the Hibernate framework⁵. The Hibernate framework maps the relational model to an object oriented model that is then used on the Java code. The middle-layer also provides the OpenMRS API that enables usage or extension of the OpenMRS platform functionality.
- **Presentation Layer:** The presentation layer contains the three user-interfaces used. The OpenMRS classic interface, legacy UI, is assembled by interfacing

¹<https://www.centos.org>

²<https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux>

³<https://openmrs.github.io/openmrs-contrib-radar/>

⁴<https://spring.io>

⁵<http://hibernate.org>

JSP with Spring controllers. For background communication through Ajax, the DWR framework is used, by mapping certain Java objects and methods of the OpenMRS platform to JavaScript (JS), which are then used through jQuery⁶ or DOJO⁷ libraries. The OpenMRS modern interface, modern UI, is built through a set of new OpenMRS modules added to the top of the OpenMRS platform. A REST interface to the service layer is exposed by the OpenMRS REST API module. Among other possible uses, this enables asynchronous communication from the front end to the back end using Ajax techniques. The modern UI uses this method of interfacing through the JS library AngularJS 1.x⁸. Angular is also the framework used for the client-side MVC. In contrast with the legacy UI, the modern UI presentation is now assembled by using GSP instead of JSP, with Spring being replaced with a more recent version with Groovy support [71].

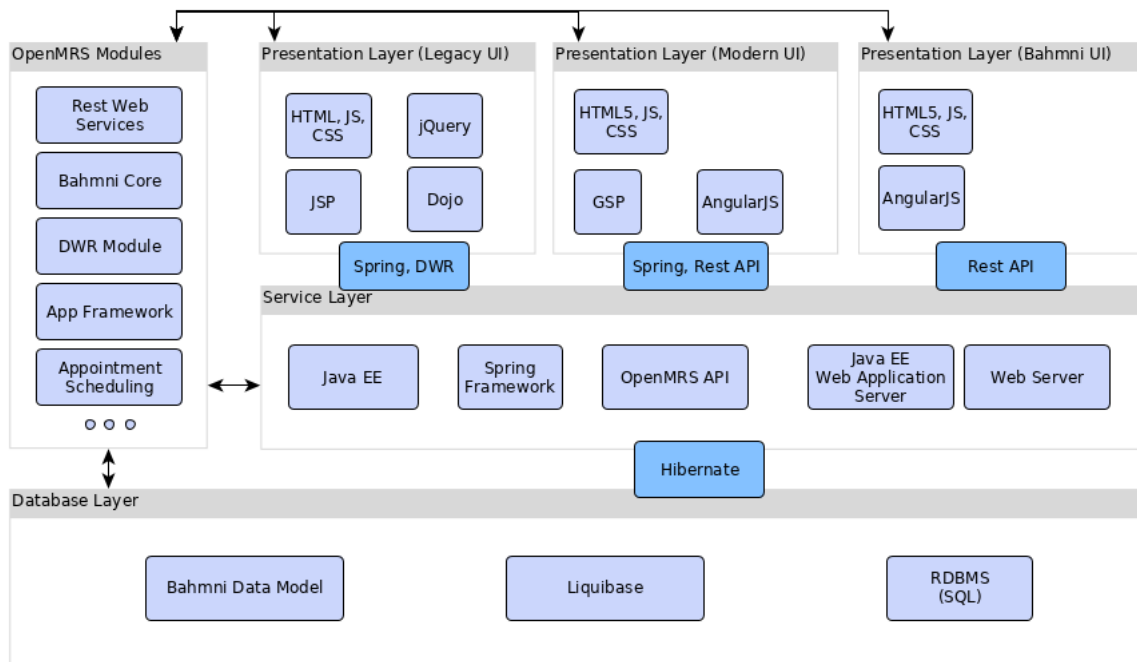


Figure 5.1: Bahmni EHR architecture overview

OpenMRS modules are transient to all layers, since they can interface with any layer directly. Bahmni EHR extends and adapts most of the OpenMRS platform functionality through an additional set of OpenMRS modules called *Bahmni Core*.

⁶<https://jquery.com>

⁷<https://dojotoolkit.org>

⁸<https://angularjs.org>

5.2 TECHNOLOGIES AND COMMUNICATION

The technologies which support the solution are distributed in a client-server model. The server-side architecture is comprised of three technology layers (see Figure 5.2): On the top layer is the Apache HTTP Server⁹ (colloquially Apache), handling all communication with the client side. Apache maps paths for the OpenMRS REST API, OpenELIS and OpenMRS server web content on their respective listening ports encapsulating all these services transparently under the same web server domain, effectively working as a reverse proxy. Separately, Apache directly serves Bahmni MRS front end to the client side, which uses the AngularJS library to interact asynchronously with the back end. The OpenMRS REST API module is the endpoint of this interactions, which are made via REST protocol.

Present on the middle layer are the Java EE web application servers, Apache Tomcat, running the customized Bahmni EHR version of OpenMRS and OpenELIS servers. The integration between OpenMRS and OpenELIS is made for the laboratory features of the latter, and is supported by two modules that feed and consume the information on both directions on each platform. The communication is done in the background through a protocol similar to HTTP, Atom Publishing Protocol (AtomPub).

The lower layer is comprised of DBMSs supporting the Bahmni EHR custom data model and the OpenELIS custom data model, accessed by their respective applications. The first one is supported by MySQL server, and the latter by PostgreSQL server. In the diagram of Figure 5.2, the dashed entities represent modules or extensions to the software that they are connected to (via dashed line).

As consequence of custom Apache URL rewrite rules, a client connection made to an HTTP URL will be redirected to the corresponding HTTPS URL. Client applications can only interface with the web server through the HTTPS protocol, resulting in more secure interactions than plain HTTP. This prevents snooping from third-party listeners observing the data flow between the two parties since the data is encrypted by TLS protocol at the application layer.

⁹<https://httpd.apache.org>

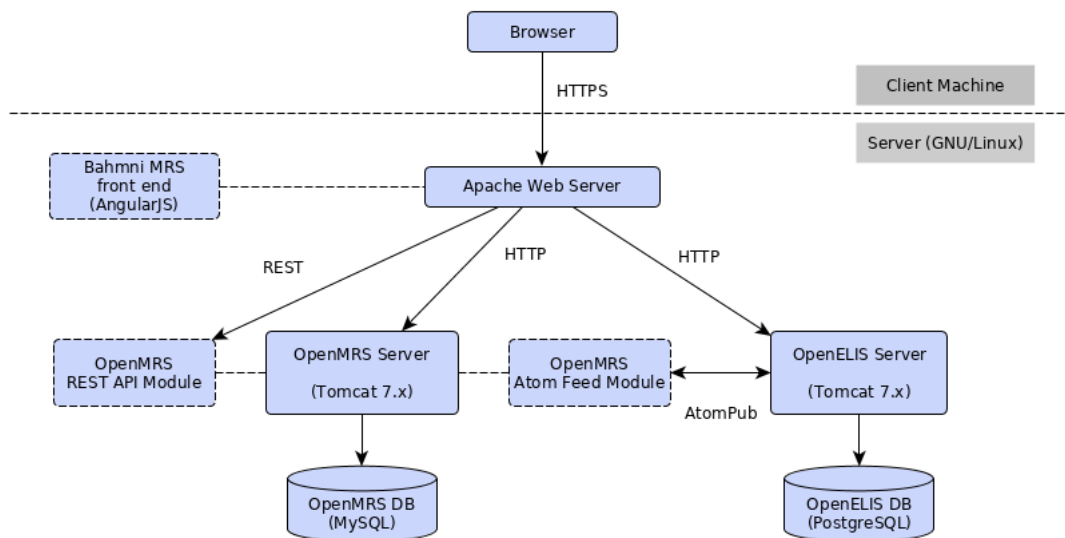


Figure 5.2: Supporting technologies and intercommunication.

The client accesses the desired platform by using its corresponding mapped URL where then Apache will either perform forwarding or resolve the request itself, giving the illusion that both platforms are present on the same web server. Requests directly aimed at OpenMRS legacy UI URL are rewritten to go to the Uniform Resource Locator (URL) and port where the web application server running OpenMRS is listening on, the same applies to the modern UI interface. Using Bahmni MRS front end, REST requests made through AngularJS are forwarded by Apache to the respective endpoint.

5.3 DATA MODEL

The Bahmni EHR data model is a more complex extension of the OpenMRS platform data model, represented in a relational model with 178 tables. For reference, the database schema of OpenMRS Platform version 1.11 has 99 and the database schema OpenMRS Reference Application 2.3.1 has 146[72]. The difference comes from extra modules that Bahmni EHR uses on top of the platform, some of which are part of the Reference Application while others are specific to Bahmni EHR. Although the number of tables is not directly comparable between the three projects since each use a different set of modules, it can still be revealing of Bahmni EHR' data model added complexity. In order to explain the data model without undergoing an exhaustive analysis, we divide it into eleven basic domains, specific to our system requirements:

- Person: Aggregation of the personal information about a person, like name, gender or current address.

- Patient: Basic information about a patient, like registration date and an UID which establish a connection to all of its related domains. A patient is always associated to the person domain.
- User: Information about a user that can interact with the system. A user can be associated to a person and have different privileges in the system.
- Encounter: Contains meta-data resultant of health-care provider meeting with a patient. An Encounter has observations or actions made besides the provider-patient association.
- Observation: Observations contain an actual record of data defined by a concept. Observations are recorded on a given Encounter.
- Concept: A strongly coded data definition for mapping concepts of the real-world into concepts in the system. An Observation can have an instanced Concept for recording its value. A form, for instance, is a concept that aggregates multiple observation concepts.
- Order: An Order is a request made by a health-care provider to a patient. Some order types inter-operate with a secondary platform, OpenELIS, as in the case of laboratory orders.
- Groups/Work-flow: Domain for patient programs, workflows and appointments and cohort data.
- Business: Domain that comprises OpenMRS' administrative part.
- System operation: Domain that refers to the system itself and is comprised of interoperability, system configuration and schema versioning, e.g. system events, quartz triggers, module management.
- Custom domains of OpenMRS Module: Independent domain created by a OpenMRS module. An OpenMRS module can change the database schema, generally extending it for supporting the additional functionality it will provide.

In a standard Bahmni installation, the data model represented by the database schema is combined with pre-constructed data that enables the Bahmni EHR standard feature set. The full data model of Bahmni EHR can be seen in Figure 1 of the appendix.

OpenELIS uses a object-relational data model, supported by the PostgreSQL Object-Relational Database Management System (ORDBMS). In the Bahmni project, PostgreSQL is also used for supporting OpenERP, dcm4chee and PACS Integration Service data models [73], nevertheless, these services are not used in this implementation. The data model of OpenELIS is represented in a object-relational manner, composed of 184

tables. For easier overview of the data model, we can segment it into 8 basic domains directly related to our requirements:

- Person: Domain of information that identifies a person.
- Patient: Domain that comprises patient information.
- Orders: Domain of laboratory orders. An order can exist either from a request coming from Bahmni EHR side or from local creation. Orders associate with registered samples, the tests made, their results and validation.
- Analysis: Domain that comprises the analysis of given tests and its results.
- Organizations: Domain of organizations that have orders on the laboratory system.
- User: Domain comprised of users that can interact with the laboratory system.
- Business: Domain that comprises OpenELIS' administrative part, like system roles and users clearance.
- System operation: Domain that refers to the system itself and is comprised of the interoperability, system configuration and schema versioning, e.g. events, quartz triggers.

In a standard Bahmni installation the database schema is combined with pre-constructed data that enables OpenELIS standard feature set. The Liquibase library is also used for management of different OpenELIS schema versions, in the same way as done for Bahmni EHR.

OpenELIS' full data model can be seen in Figure 2 of the appendix.

5.4 BAHMNI EHR-OPENELIS INTEROPERABILITY

The feature set of Bahmni EHR is complemented with a separate platform for the laboratory sub-system, called OpenELIS. Both platforms interact together, providing a seamless work-flow through them with separation of responsibilities. A doctor on the Bahmni EHR side can, for example request laboratory orders to be made to a patient, and these are transparently sent to the OpenELIS subsystem, where for example a laboratory clerk will view and process them. The data produced on each system that is needed on the other is sent through the Atom Publishing Protocol. In the designated protocol, a producer acts as a server, broadcasting events to whoever listens on them, and consumers act as clients, consuming these events (Figure 5.3). In our implementation, both OpenELIS and Bahmni EHR act as server and client, since both produce and consume information from one another.

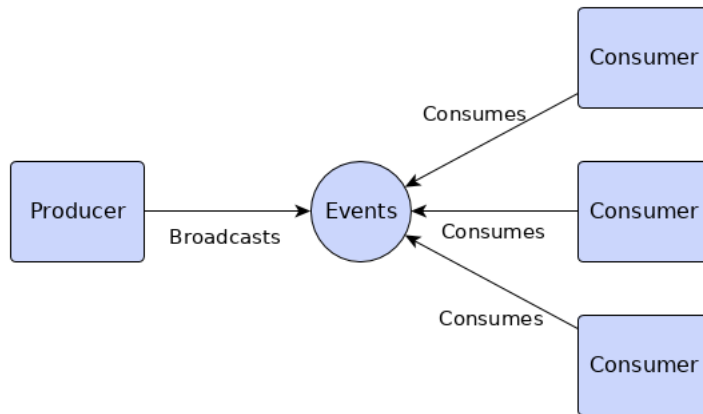


Figure 5.3: Overview of how feed producers and clients work

A feed holds multiple entries in series to a maximum of M entries, which is a configuration of the Atom publisher modules of Bahmni. In a feed, once this number of entries is reached, the feed is said to be full and another feed is created and linked using the previous 'next' field, and this new feed links to it using its 'previous' field (Figure 5.4).

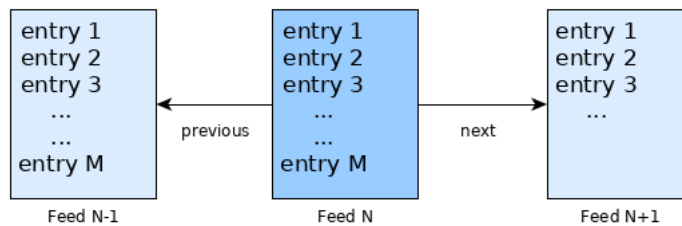


Figure 5.4: Representation of a feed and feed linkage

Each event is defined in XML using the ATOM Syndication Format, for the need of having syndicated events between this two different applications.

An event is an entity detailed as seen in Table 5.1:

Table 5.1: Bahmni EHR-OpenELIS event fields

Field	Description
UUID	Alpha-numeric identifier which uniquely identifies an event.
Title	Identifies the event in human readable form.
URI	URI pointing to an accessible location containing the content depicted by the event.
Timestamp	Time at which the event was created.
Contents	The data encapsulated by the event as string.

For some events, e.g. ones which has as category either Patient or Laboratory (results), URI references an HTTP obtainable resource through REST protocol holding the specific data the event designates.

Bahmni EHR uses AtomFeed ¹⁰, a open source implementation of the ATOM protocol in JAVA, licensed in Apache Version 2.0. This implementation is used by the Atom Feed OpenMRS module (openmrs-atomfeed), responsible for publishing events produced by Bahmni EHR. The OpenMRS module that consumes events from OpenELIS to Bahmni EHR side is OpenELIS Atom Feed Client (openelis-atomfeed-client).

Bahmni's OpenELIS fork includes an Atom server and client component integrated with the platform events, working in the same manner as depicted for Bahmni EHR. Also, a REST endpoint is available in order to give access to Bahmni EHR to resources present on OpenELIS side, only referenced on some specific event types.

The persistent nature of the events in Bahmni EHR is achieved through an extension of the OpenMRS data model, made by the Atom Feed OpenMRS modules on their initial run. The same extension was made for OpenELIS data model. Persistence is necessary since the clients don't need to be active at the time of a event broadcast to consume it, thus, in their respective databases, producers keep the information needed for generating events while consuming keep track of the last event received, processed or failed.

To familiarize the reader with the subsequent event logic processing without undergoing an exhaustive analysis, we present just the main extension to the data module for interoperability, represented by tables while explaining their need¹¹:

- event_records: This table holds the events to be published by Atom Feed for others to consume. Besides having the depicted entry fields, the table has an extra column named 'category', used to indicate the type of the event (e.g. patient, encounter, lab).
- markers: This table holds marker entries referencing feed URIs indicating the last processed records. This way the client module can know what he already consumed and get the next new entry in the feed or the next new feed when the previous is full.
- event_records_offset_marker: This table holds maker entries of the records cached for faster event process by the consumer.
- failed_events: This table holds events which could not be consumed and are consequently marked as failed. They are retried later by a different event handler.

An event represents its domain by a field named category. The categories used belong to a well-defined set, allowing consumers to fetch events of the domains they need by filtering them by the categories they want. The categories of the events created and

¹⁰<https://github.com/ICT4H/atomfeed>

¹¹<https://bahmni.atlassian.net/wiki/display/BAH/Atom+Feed+Based+Synchronization+in+Bahmni>

published by OpenMRS or OpenELIS for their counter-part consumption, concerning our implementation, are:

- Patient: For updating changes to the patients' basic information or address.
- Relationship: Event that carries inter-patient relations, e.g. father, sibling.
- Encounter: Event detailing an open or closed encounter where observations were collected.
- Lab: Events pertaining laboratory, like test results or validation, and new or updated definitions of tests.

5.5 NETWORK AND SECURITY

The security of the solution is comprised of different domains - the domain of service usage, comprehending the EHR interfaces provided to the end-users, including Bahmni EHR, OpenMRS and the OpenELIS platforms, and the domain of server administration, comprehending the system services accessible to server administrators.

The access management model in our EHR implementation follows RBAC, where a user is given access to interact with resources based on roles assigned to him which define access to resources [74]. In Bahmni EHR, this security model is made effective by the OpenMRS platform.

In Bahmni EHR one needs to be a user to interact with the system. A user in the system is either associated with a person or is an independent user used by software systems. In this security model, users access is defined based on the different system privileges associated with them.

A privilege is the basic unit that defines the access control to a particular feature, e.g. "Edit Observations" or "Get Orders". Privileges can be grouped under an umbrella called role, with a more generic name to what access it gives. New roles can be created as an extension of previous roles by a mechanism of inheritance, where the active privileges of the parent role are also active on the new role. Later modification to the list of active privileges on a parent role also propagate to its children inherited privileges list. By themselves, privileges and roles don't do anything on systems interaction access control, as they need to be associated to a user to complete the model. Multiple roles assigned to an user defines his clearance on the system. A OpenMRS module can add new privileges to the OpenMRS privileges list as another base for its functionality and resources' access control.

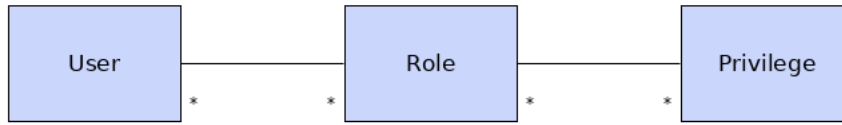


Figure 5.5: Relationship between user, roles and privileges in Bahmni EHR

A system administrator can assign a set of roles to a user to increase access to new resource domains. The Figure 5.6 shows the roles in Gondar EHR, where some are provided in a Bahmni standard installation while others are custom.

<input type="checkbox"/> Consultation-Orders	Will have access to consultation orders and save in both normal and retrospective mode	Consultation-Save	Add Orders , Edit Orders
<input type="checkbox"/> Consultation-Save	Will have basic access to save consultation	Clinical-Read-Only	Add Encounters , Get Encounter Roles ...
<input type="checkbox"/> Consultation-Treatment	Will have access to consultation treatment and save in both normal and retrospective mode	Consultation-Save	Add Orders , Edit Orders
<input type="checkbox"/> Doctor	Role for the doctor		Get Orders , app:clinical:diagnosisTab ...
<input type="checkbox"/> Emr-Reports	Ability to run reports	Patient-Listing	
<input type="checkbox"/> InPatient-Patient-Movement	Ability to admit, discharge and transfer the patient	InPatient-Read , Consultation-Save	Edit Admission Locations , Assign Beds
<input type="checkbox"/> InPatient-Read	Ability to view the Inpatient dashboard	Patient-Listing	Get Beds , Get Orders ...
<input type="checkbox"/> Nurse	Role for the nurse		Get Orders , View Encounter Roles ...
<input type="checkbox"/> Orders-Role	Ability to view and fulfill orders	Patient-Listing , Consultation-Save	Get Orders , Get Global Properties ...
<input type="checkbox"/> Patient-Documents-Upload	Ability to upload patient documents and radiology uploads	Patient-Listing , Consultation-Save	Get Global Properties , Get Patients ...
<input type="checkbox"/> Patient-Listing	Will have access to all patient queues		Get Providers , Get Users ...
<input type="checkbox"/> Privilege Level: Full	A role that has all API privileges		Provider Management Dashboard - View Patient
<input type="checkbox"/> Program-Enrollment	Can manage patient programs		Purge Program Attribute Types , Get Programs ...

Figure 5.6: List of access roles, some of which are custom.

OpenELIS also adopts the same access management model. In contrast with Bahmni EHR, roles in OpenELIS are not composed of smaller units, they are instead the single unit of access control. Despite OpenELIS not having a privilege grouping feature or role inheritance, its underlying security model remains the same as the Bahmni EHR model, by mapping roles in the first one to privileges in the latter.

Security also applies to the server system and what services it exposes to the network.

The implementation server exposes its services and functionality remotely on the network, e.g. for providing the Bahmni EHR or OpenELIS front ends to a remote location. For a production environment it would be more secure to only allow communication with the system services from the local network by adjusting the its architecture to not bridge access from outside or alternatively firewall it to the WAN side. Since the services will then be inaccessible from outside the server local network, this creates hindrances, e.g. a doctor traveling to another city would not be able to update a patients' information remotely.

This issue can be solved with a secure Virtual Private Network (VPN) that bridges an authorized client accessing from the Wide Area Network (WAN) with the server private network. The server services can be set to only be accessible from the private network and a firewall on the server machine can enforce this behavior, allowing only the required communications. In our solution, platforms are exposed to end-users through port 80 or 443 for either HTTP redirection or HTTPS communication respectively,

and port 23 is exposed for SSH for server administration. The server firewall blocks outbound communication to all the other remaining TCP or UDP ports of the server.

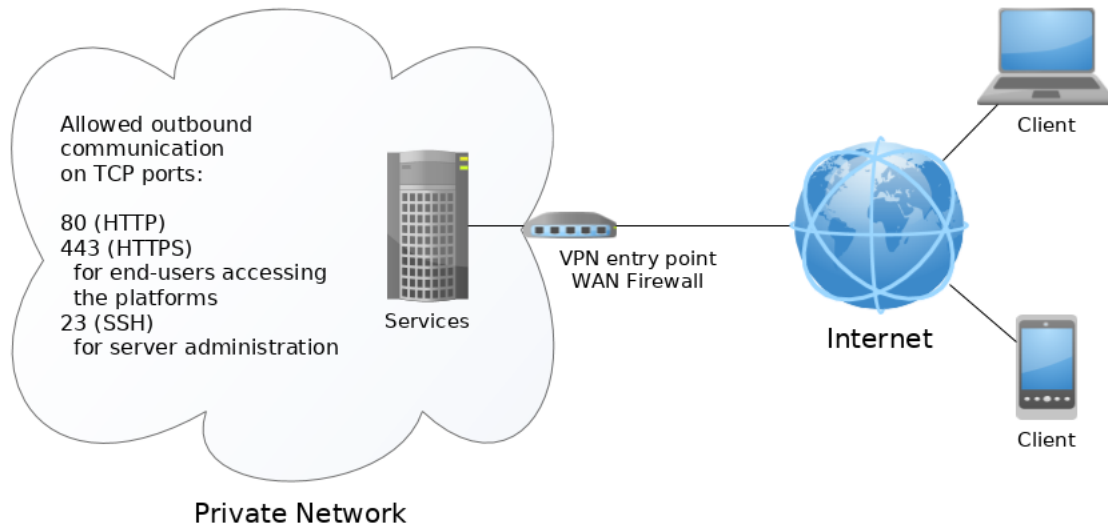


Figure 5.7: VPN entry point with firewall, and server firewall

Multiple user-space applications will be running in the system, each with a different purpose. A system administrator can improve his systems security better by giving just enough access to the applications running. This can be applied to the application layer by a sandbox layer that controls access from and to all resources in order to secure applications from other applications trying to tamper with them or with the system.

In our implementation this is achieved by using SELinux. SELinux adds a layer of security with policies in the Linux kernel by supervising access to the system resources. Figure 5.8 illustrates the decision process logic¹². SELinux comes already installed and enabled by default in a standard installation of CentOS. Specific policies needed to be added for our solution to work after installation, which can be seen in the configuration section.

¹²https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/ch-selinux.html

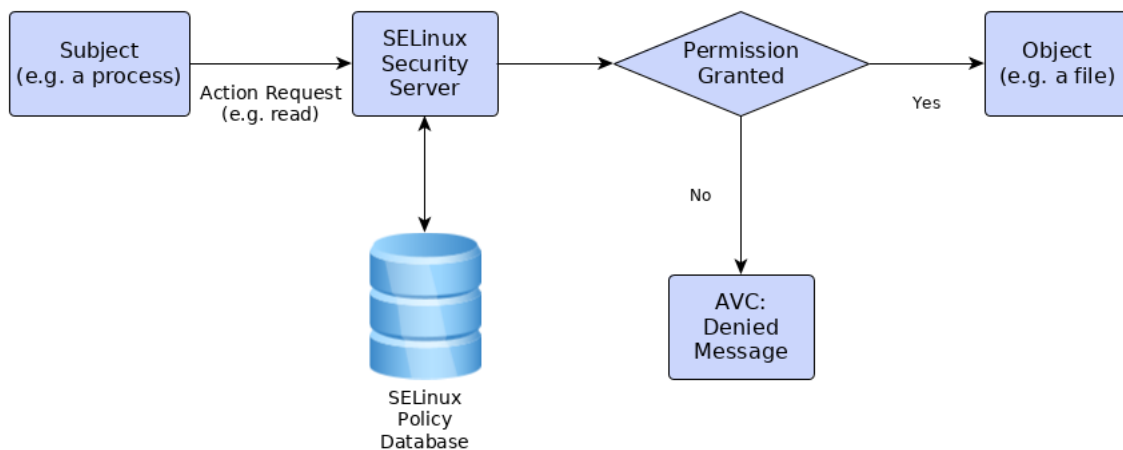


Figure 5.8: SELinux decision process logic

Remote access to a production environment is often needed for administration or maintenance tasks. Enabling remote access requires not only a secure authorization methods, but also policies for login in.

If the enabled authorization methods contain a password-based method, then, brute force or dictionary attacks are possible. The use of public-key cryptography for remote authentication to the SSH service is a proven secure method that greatly invalidates this kind of attacks.

Having a strong password, containing mixed letters, numbers and symbols is one way to invalidate plain dictionary attacks, but in the case of a brute forcing attack, an attacker might have success if the size of a users' password is small or if it is made of simple combination of characters without mixing types. The default SSH installation under CentOS is not secured against this kind of attack since an attacker can try as many passwords as the network communication and the SSH authentication processing overhead allow.

Fail2Ban is a free and open-source software (FOSS) software framework for protecting against brute force attacks. In order to refrain an attacker attempt to successfully login by exploiting password-based authentication methods that contain insecure passwords, Fail2Ban is one part-solution that was enabled for SSH authentication, by restricting the number of login attempts coming of a given IP address. With Fail2Ban set up, after a defined number of failed attempts, the IP address is banned for a duration, and for that duration the packets are dropped at the network layer by Netfilter subsystem, the second-part of the solution. This invalidates brute-force attacks from the same IP address, and despite distributed attacks (many organized login attempts from different IP addresses) being possible, they are greatly restrained.

5.6 OFF-LINE SUPPORT

The client-server model used in Bahmni requires a network for the interactions between both parties. At a time of an interaction by an authenticated user, presence of a network is required for making the changes or operations at that moment. In low-development countries interruptions to the network occur frequently, either by failure of the network itself or as result of external factors like power outages.

When failures occur, sometimes a user loses filled-in data by submitting it unknowingly while the connection to the server was down. In order to avoid this, Bahmni EHR can present an active notification in the user-interface with the status of the connection.

Support for using some of the Bahmni EHR functionality while off-line can also be achieved.

Bahmni provides two client applications for off-line support: a browser extension and a mobile application. These make some features available off-line by packaging in the client the corresponding user-interfaces and logic. This does not apply only to static content; dynamic content like for instance patient information can also be pre-loaded beforehand so that off-line features have the necessary data to work with. There is also support for caching custom form templates off-line. This can be added through configuration on the server-side, and can be seen below in the "Configuring off-line support" section.

While using the client application, interactions are queued in order. Periodically the client checks if the network recovered by trying to re-establish a connection with the server. When the network recovers, the client can send all the queued interactions to the server, effectively making the changes that were previously on hold.

5.7 INSTALLATION

Focusing on the server-side, the solution is installed on computer hardware satisfying the software implementation requirements. This comprises the operating system and required software installation, as well the configuration and deployment of the EHR software as described in the previous sections.

The version of Bahmni used in this implementation, is Bahmni version 0.80. This version brings and OpenMRS version 1.12-dev, an in-development version that adds some new features but which is not OpenMRS platform-stable yet. Several minimum hardware components are required for a successful implementation as described. A display monitor connected to the server hardware is not required, but can be useful for local installation or troubleshooting. The CPU must respect either the i386 or IA64 specification, and although the RAM required to run a full Bahmni installation is 8GB,

on our implementation 4GB is the minimum. Disk free-space should be at least 10GB for the installation, and be scaled accordingly to contain the expected produced data. For client-server interaction, network support is necessary and connections between both devices should be possible, at least the ones initiated from client to server.

The Bahmni project provides a repository with a set of RPM Package Manager (RPM)s for installing the different parts of the Bahmni software using a compatible RPM package manager. The following components were used in our solution:

- *bahmni-web* is a package that installs Apache HTTP Server version 2.x, the configuration that sets the default implementation behavior and installs the Bahmni MRS front end. The package files are placed under `"/var/www"`, the common directory for Apache HTTP Server data, and the service name associated with it is `httpd`, Apache. The *base default config* is a default configuration for Bahmni EHR, which includes customizable front end applications, internationalization files, document and report templates, all which can be adapted to suit specific use cases.
- *openmrs* RPM brings an embedded Tomcat server and a customized version of OpenMRS, and is already configured to deploy the OpenMRS platform on it.
- *bahmni-emr* RPM is the set of OpenMRS modules critical for bahmni functionality and OpenMRS integration, called Bahmni core modules, that installs under the modules folder in `"/opt/openmrs"`. Includes `openmrs-atomfeed.omod` (publisher for `openmrs` info).
- *bahmni-lab* RPM also brings an embedded Tomcat server already configured for deploying the OpenELIS laboratory system. This package places its files under `"/opt/bahmni-lab"`.
- *bahmni-lab-connect* is the package that provides an Atom Feed Client, responsible for providing and consuming background feeds between OpenELIS and OpenMRS. The component is an OpenMRS module, thus requiring the OpenMRS platform to work. It is used for interoperability between Bahmni EHR and OpenELIS and OpenELIS interacts with this module indirectly. The dashed line in Figure 5.9 represents this relationship.
- *bahmni-reports* RPM provides the reporting sub-system via a *webapp* which uses the DynamicReports library for reporting generation. The *webapp* runs on an embedded Tomcat server. The package places its content under `"/opt/bahmni-reports"`.
- *bahmni-event-log-service* RPM provides the event-log sub-system via a *webapp* for supporting Bahmni off-line synchronization. The *webapp* runs on an embedded

Tomcat server. The package places its content under `"/opt/bahmni-event-log-service"`.

Java Runtime Environment, MySQL and PostgreSQL are low-level dependencies of the project components. The first one is a direct dependency of Tomcat and the others are DBMS that give the persistence support for Bahmni EHR and OpenELIS respectively. The Bahmni project provides two database dumps which besides the database schema, contains filled data for a standard Bahmni EHR and OpenELIS configuration. Figure 5.9 shows the components used and the dependencies between them (represented by the connecting lines). In the figure, the arrow represents the direction of the dependency.

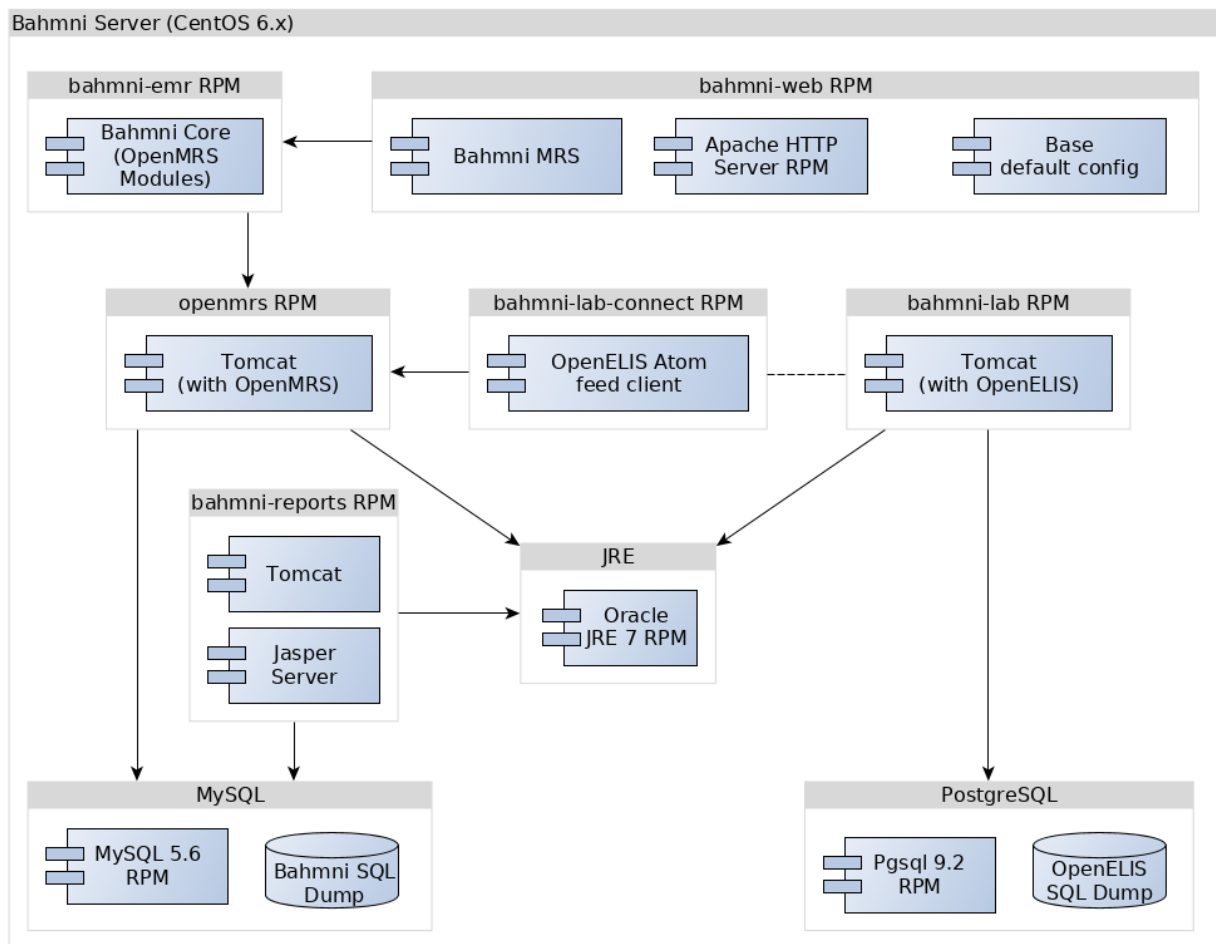


Figure 5.9: Bahmni components used on the solution implementation.

There are three provided ways to automate an installation of Bahmni and its environment: a bash script, Ansible setup, and Vagrant image. The first two installation methods require a working GNU/Linux distribution set up before their usage. The first is a bash script that relies only on the GNU tools of the operating system and a

compatible RPM package manager. The latter, used on the newer Bahmni versions, is built using the Ansible framework. Ansible is a FOSS platform that can be used for creating parametrized setup configurations and deploy those configurations on a wide array of different machines. The third being Vagrant, that uses either one of the first methods and gives as output a full virtual machine image.

The first method allows more flexible setups because one can choose the underlying RPM-based GNU/Linux distribution, or use an already deployed one. For the implementation of our solution, we chose CentOS 6.7 as the GNU/Linux distribution, also recommended by the Bahmni project. The provided bash script can be easily adapted to install only the desired components. The script is responsible for making some adjustments to PostgreSQL DBMS configuration and also fetches using the *wget* tool the database dumps provided by Bahmni project and installs them under the respective DBMS.

The second method is through the Ansible setup system, which is the method that Bahmni will adopt for the newer versions. In Ansible setup configuration made for Bahmni, it is possible to set well-documented configuration parameters of the installation process to desired values and deploy it easily on different machines, where the selected components will be installed and configured as it was previously defined. Existing configuration parameters are, for instance, listen ports to use for each platform, OpenMRS modules to install additionally and authentication parameters like passwords or certificates, among others. This new install method also lowers the differences between implementations so that implementers experience less issues resultant of installation or configuration methods.

The Bahmni project also provides a method of deployment automation that outputs a Vagrant image as result, using the Vagrant tool.

Vagrant gives a way to create a virtual machine image out of a well-defined machine environment. Vagrant is responsible for applying customizations to virtual machine parameters, e.g. specific network configurations, shared directories or authentication setup. In the Vagrant Bahmni installation, Vagrant relies on Packer. Packer is a FOSS tool that enables the automated creation of machine images and software provisioning out of a single configuration file. In the Vagrant Bahmni installation, Packer is used transparently for setting up the operating system by using the GNU/Linux CentOS 6.7 distribution, and for installing and configuring the software within. For the latter, it relies on a configured provisioner which, in the case of Bahmni software is either the bash installation script or the Ansible tool. After Packer finishes configuring the machine, the management of it is passed again to Vagrant which makes final adjustments before outputting a Vagrant-compatible virtual machine image. The resultant image is not only useful to developers, but also to implementers for the standard and well-configured environment, more easily achieved and less prone to miss-configurations.

In any of the above methods, after the Bahmni installation, there is a step where both MySQL and PostgreSQL databases are set under the respective DBMS. These databases, two dumps comprised of schema and data, are retrieved from a remote repository managed by the Bahmni project, and part of the official installation. At the moment (for Bahmni 0.80 version), there is not a clean database dump (without test data) provided. There is an undergoing efforts to change this in order to provide in the future a database dump with just the necessary information and no test data. Nonetheless, a custom bash script adapted from one distributed by Bahmni project was made, which interfaces with both MySQL and PostgreSQL DBMSs removes this extra test data from *openmrs* and *clinlims* databases [75] [76]. The custom script resets the auto-incrementing IDs and history of modifications, truncates the database tables for test data including patient's personal information, encounters, orders, samples, test results and so on and corrects the event sub-system correspondent to the interoperability between the platforms so that there is no data buffered for exchange and processing on/to the complementary platform. On our implementation, at the end of the installation process, the databases don't contain any test data.

The software package of each service brings a bash script file for the traditional SysVinit system service control of GNU/Linux that is installed under `"/etc/init.d/"`, and can be used for starting, stopping or enabling at boot the installed service.

On the first run of the OpenMRS platform, the Liquibase tool will check if the database schema of *openmrs* database is the latest available, by comparing the new schema version that the software installation brings with the schema version present in the database's DBMS. If the installation software provides a higher version of the schema, an update to the database schema of *openmrs* is issued and the database is migrated to the new version.

For providing Bahmni off-line support, manual installation of Bahmni event log service component is required if using installation via the official methods. It is present in the repository which is added on installation to the operating system's package manager. Therefore issuing an installation instruction with the component name as "bahmni-event-log-service" should be enough for installing it locally.

5.8 CONFIGURATION AND CUSTOMIZATION

After the task of software installation is done, configurations to each component are made in order to satisfy the system requirements for Gondar's LRTC. The results are depicted in the following chapter.

An extra OpenMRS module required besides the ones already provided by Bahmni EHR is the Appointments module, which is needed for adding appointment scheduling

functionality to the platform. The module depends on the modern UI and its supporting modules, therefore they also need to be installed.

5.8.1 PATIENTS ATTRIBUTES AND REGISTRATION FORM

The patient registration in Bahmni EHR has been adapted and extended to support Gondar's LRTC paper-based registration form.

A new patient Unique identifier (UID) was added to the platform and configured in such a way that it is automatically assigned to a new patient on a registration in Gondar. Creating a new UID can be done through the OpenMRS legacy UI by configuring an Identifier (ID) generator which specifies the ID format, followed by a definition of an identifier source, Gondar ID, which makes use of this generator. In our case, the UID is configured as a mix of letters and digits, starting with a prefix defined as the three letters *GLC* and followed by 6 minimum digits, which can go up to 9 in length. The UID is auto-incrementing, meaning the next patient registered will have the next UID in the sequence compared to the one attributed to the previous registered patient.

This ID references a patient through all the Bahmni systems. Exchanging patients' information to OpenELIS makes use of the same ID for referencing patients between both platforms. The patient ID is also required to be recreated on OpenELIS side, which can be done in OpenELIS administration area.

Gondar's LRTC registration clerk gives a *hospital chart number* to a patient on registration for associating multiple records with the same person. This ID is composed by 5 or 6 digits followed by back-slash and the year of registration. The former ID used in Gondar's, called *hospital chart number*, is also added to the system and kept for compatibility reasons for patients which have been previously registered through their registration books.

Customizing registration page or search features is done by modifying `app.json` in `default-config/openmrs/apps/clinical/app.json`.

Support for the *former ID* was made as a person attribute. If an attribute requires to be unique between patients, this can be done by setting the supporting structure to be a concept or a custom a database field that only accepts unique values. The attributes can be validated against a pattern, which can be defined by a regular expression in `app.json`, as can be seen in the listing.

```
"fieldValidation" : {
  "Telephone_Number" : {"pattern" : "[0-9]{9,14}", "errorMessage" : "Should
    be at least 9 numbers (country code is +251)"},
  "postalCode" : {"pattern" : "[0-9]{1,}", "errorMessage" : "Should contain
    only numbers"}
```

```
"glcidformer" : {"pattern" : "[0-9]{5,6}\\/[0-9]{2}", "errorMessage" :  
  "Should be 5 or 6 digits, slash, year of registration"}  
},
```

The mandatory registration fee field, enabled by default, was disabled by setting the *required* value to false as seen in the following code (*app.json*).

```
"conceptSetUI": {  
  "REGISTRATION_FEES": {  
    "required": false,  
    "label": "Fee"  
  }  
},
```

The search can be extended to support other fields as search parameters. was extended to support searching using a patient's name in his local language, and filtering by the village field of the addressing system, which can be done.

```
"patientSearch": {  
  "address": {  
    "label": "Village",  
    "placeholder": "Search by village",  
    "field": "address3"  
  }  
},  
"customAttributes": {  
  "label": "Amharic name",  
  "placeholder": "Amharic given, middle or family name",  
  "fields": ["givenNameLocal", "middleNameLocal", "familyNameLocal"]  
}
```

In the health area, registration forms generally have some questions for capturing more information regarding to a patient's health or demographics. Bahmni supports custom person attributes, where questions and the type of its data can be specified. A patient attribute always use the same field in the database for storing the most recent answer value, therefore there is no track of answer history in this feature. These questions are present to a patient during his registration, and answers should not be subjective to change with time, hence should only be used only for questions where only the most recent value matters.

Custom attributes can be defined data types or a OpenMRS concept. In the case of the latter, the "Foreign Key" field be the ID of the wanted concept to map. A privilege for editing this attribute can be set so that only users with assigned role matching that privilege can edit a given person attribute (Figure 5.10).

Person Attribute Type

Name*	<input type="text" value="hivpositive"/>	
Format	<input type="text" value="java.lang.Boolean"/>	<i>Name of a Java or OpenMRS class.</i>
Foreign Key	<input type="text"/>	<i>Integer id of the object specified by 'format'</i>
Searchable	<input type="checkbox"/>	<i>Whether this type can be searched on or not</i>
Description	<input type="text" value="Is HIV Positive?"/>	
Edit privilege	<input type="text"/>	<i>The privilege needed by a user to edit this person attribute</i>
Created By		
Changed By		
UUID		
<input type="button" value="Save Person A"/>		
Retire Person		
Reason	<input type="text"/>	
<input type="button" value="Retire Person"/>		
Delete Person		
<input type="button" value="Delete Person"/>		

- Add Allergies
- Add Cohorts
- Add Concept Proposals
- Add Drug Groups
- Add Drug Info
- Add Encounters
- Add HL7 Inbound Archive
- Add HL7 Inbound Exception
- Add HL7 Inbound Queue
- Add HL7 Source
- Add Observations
- Add Orders
- Add Patient Identifiers
- Add Patient Lists
- Add Patient Programs
- Add Patients
- Add People
- Add Problems
- Add Relationships

Figure 5.10: Adding a person custom attribute

Figure 5.11 shows the creation of the attribute "Occupation", with format of type "org.openmrs.concept", used for creating an attribute referencing an OpenMRS concept. The foreign key field in this case is a reference to an OpenMRS concept ID already created, pointing to the concept Occupation.

Person Attribute Type

Name*	<input type="text" value="occupation"/>	
Format	<input type="text" value="org.openmrs.Concept"/>	Name of a Java or OpenMRS class.
Foreign Key	<input type="text" value="109"/>	Integer id of the object specified by 'format'
Searchable	<input checked="" type="checkbox"/>	Whether this type can be searched on or not
Description	<input type="text" value="Occupation"/>	
Edit privilege	<input type="text" value=""/>	The privilege needed by a user to edit this person attribute
Created By	Super User - September 18, 2014 12:00:00 AM EAT	
Changed By	Super Man - May 20, 2016 4:49:02 PM EAT	
UUID	c1f7d1f1-3f10-11e4-adec-0800271c1b75	
<input type="button" value="Save Person Attribute Type"/>		

Figure 5.11: Custom OpenMRS concept "Occupation"

The concept which the attribute *Occupation* points to, can be created through the dictionary section (Figure 5.12). Other previously created concepts were assigned as the answers to this concept. This defines which values can be set on an instance of an attribute of this type.

Viewing Concept Occupation

[Previous](#) | [Edit](#) | [Stats](#) | [Next](#) | [New](#)

Id	109
UUID	c221869a-3f10-11e4-adec-0800271c1b75
Locale	English Spanish French Italian Portuguese
Fully Specified Name	Occupation
Synonyms	
Search Terms	
Short Name	occupation
Description	
Class	Misc
Datatype	Coded
Answers	Unemployed (110) Student (111) Government (112) Business (113) Housewife (114) Labour (115)
Mappings	<input type="checkbox"/>
Version	
Created By	Super User - September 18, 2014 2:19:53 PM EAT
Changed By	Super User - September 18, 2014 2:19:53 PM EAT

Figure 5.12: Person attribute "Occupation", mapped to a custom OpenMRS concept

A list with some of attributes configured for Gondar can be seen in Figure 5.13. Attributes which contain a strike line over their text are marked as disabled, meaning

they are not displayed on the registration form. Attributes can be marked as required, meaning that data entry is mandatory on registration, e.g. "Able to work".

Attribute Types				
Name	Format	Searchable	Description	Edit privilege
<input type="checkbox"/> givenNameLocal	java.lang.String		Name in local language	
<input type="checkbox"/> familyNameLocal	java.lang.String		familyNameLocal	
<input type="checkbox"/> middleNameLocal	java.lang.String		middleNameLocal	
<input type="checkbox"/> fathersname	java.lang.String	Yes	Father's Name	
<input type="checkbox"/> grandfathersname	java.lang.String	Yes	Grandfather's name	
<input type="checkbox"/> occupation	org.openmrs.Concept	Yes	Occupation	
<input type="checkbox"/> migrationstatus	java.lang.String	Yes	Migration Status	
<input type="checkbox"/> abletowork	java.lang.Boolean		Is able to work?	
<input type="checkbox"/> hivpositive	java.lang.Boolean		Is HIV Positive?	Add Observations
<input type="checkbox"/> art	java.lang.Boolean		Is on Antiretroviral Therapy?	
<input type="checkbox"/> Telephone Number	java.lang.String		Telephone Number	
<input type="checkbox"/> caste	java.lang.String	Yes	Caste	
<input type="checkbox"/> class	org.openmrs.Concept	Yes	Class	

Figure 5.13: Customized person attributes

The registration page allows printing of a card for handing out to the patient. The hand-out can be customized with patient recommendations in local languages and extended to support other fields and details. This was done by editing (accordingly) the files: `print.html` and `print_local.html` under "default-config/openmrs/apps/registration/registrationCardLayout".

5.8.2 ADDRESS SYSTEM

Each country has some governance system that defines how zones and boundaries are set in the country, leading to the creation of multiple addressing systems over different countries. For electronic health records, the addressing system supported should match the one of a patients' country. The address hierarchy in Ethiopia starts with regions. These are sub-divided into zones, which are sub-divided into districts. Under districts is the smallest administrative unit, the kebele. Kebeles delimit a group of people, similar to a ward. This hierarchy is represented on Figure 5.14.

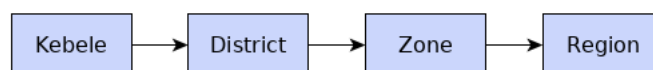


Figure 5.14: The common address hierarchy in Ethiopia

For providing support to this addressing system, an address template in the XML format is built for Bahmni EHR (Figure 5.15). This template defines each field name and

mapping to a corresponding database field which gives support to addresses' instances. The template can also define the layout of the fields, like order and size of the text fields for legacy and modern UI, however ignored in Bahmni EHR.

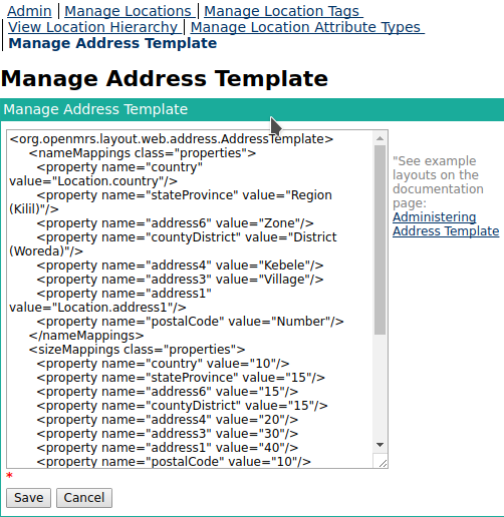


Figure 5.15: Custom XML address template for Ethiopia.

OpenMRS provides a user-interface in the legacy UI for building the address system. After the template is submitted to OpenMRS, the fields become defined but do not have a hierarchical relationship established between them. In the address system management of legacy UI, a user can specify the hierarchy between fields while OpenMRS deals with the internal entities associations automatically. Any address field can be marked as required, providing a minimum precision to addresses registered for a person. Fields down to kebele level are marked as required but not the house number field, since in Ethiopia not all addresses contain this field.

Address Hierarchy Levels			
Level	Name	Example Entry	Mapped Address Field
1	Country	Ethiopia (1 total entries)	Country (country)
2	Region (Kilil)	Addis Ababa (11 total entries)	Region (Kilil) (stateProvince)
3	Zone	Adama (special zone) (93 total entries)	Zone (address6)
4	District (Woreda)	Abala (749 total entries)	District (Woreda) (countyDistrict)
5	Kebele	(0 total entries)	Kebele (address4)
6	Village	(0 total entries)	Village (address3)
7	Address	(0 total entries)	Address (address1)
8	Number	(0 total entries)	Number (postalCode)

[Add New Address Hierarchy Level](#)

Figure 5.16: Hierarchy and mapping of an address field

Bahmni EHR supports auto-completion of address fields which are provided based on a user customized hierarchical list of addresses. A representation of this can be seen in Figure 5.17. Ethiopia contains 9 regions which are divided into a total of 68 zones, sub-divided into more than 600 districts. Some administrative divisions do not follow the depicted hierarchy in a straight manner.

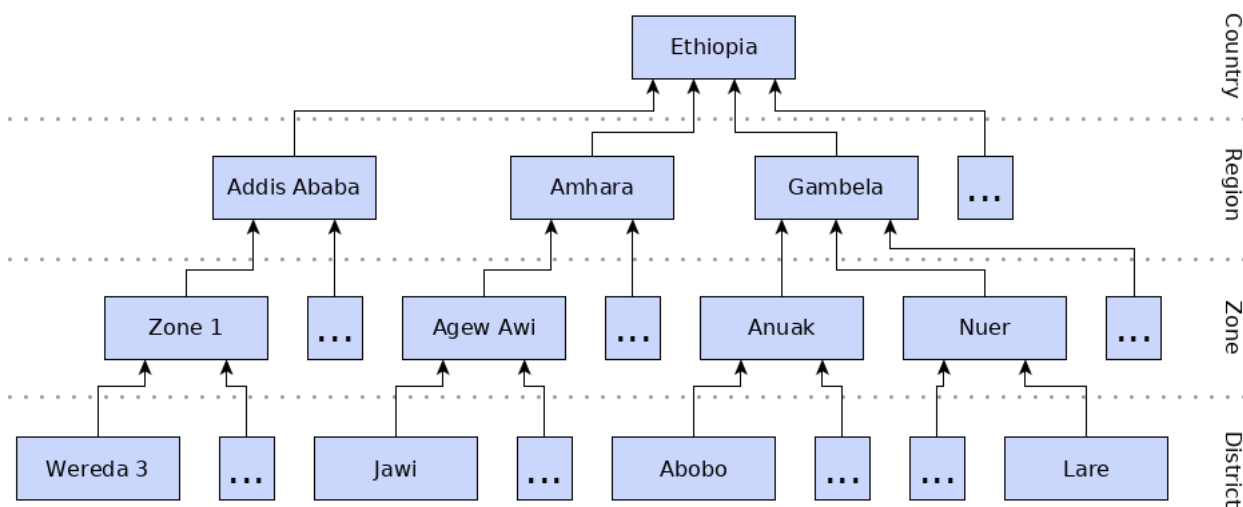


Figure 5.17: Expanded mapped addressing system

The support for providing address hierarchy data to the OpenMRS platform is also available via the legacy UI. Firstly, a delimiter-separated values file containing addresses represented in an hierarchical manner needs to be created. In this delimiter-separated values (DSV) file, the first row is the first and most outer-level, the second row is one level down under in the hierarchy where the first row value is its parent, and so on. In our implementation, the delimiter used for the DSV file is the special symbol '|', used through all the document since it is a unique character that is not used in any address value of Ethiopia. In our address data file, we have addressed hierarchically 11 regions, 93 zones and 749 districts of Ethiopia (Figure 5.18). This differs from the current official system as some former districts and zones were included for backwards-compatibility with the older address system used before 1991. The hierarchy of the addressing system in Ethiopia goes deeper but it would require too much effort to map and structure the remaining fields in the file.

1	Ethiopia	Addis Ababa	Addis Ababa (special woreda)	Addis Ababa
2	Ethiopia	Addis Ababa	Zone 1	Wereda 3
3	Ethiopia	Addis Ababa	Zone 1	Wereda 4
4	Ethiopia	Addis Ababa	Zone 1	Wereda 5
5	Ethiopia	Addis Ababa	Zone 1	Wereda 6
6	Ethiopia	Addis Ababa	Zone 2	Wereda 20
7	Ethiopia	Addis Ababa	Zone 2	Wereda 21
8	Ethiopia	Addis Ababa	Zone 2	Wereda 22
9	Ethiopia	Addis Ababa	Zone 2	Wereda 23
10	Ethiopia	Addis Ababa	Zone 2	Wereda 24
● ● ●				
26	Ethiopia	Addis Ababa	Zone 5	Wereda 14
27	Ethiopia	Addis Ababa	Zone 5	Wereda 25
28	Ethiopia	Addis Ababa	Zone 6	Wereda 26
29	Ethiopia	Addis Ababa	Zone 6	Wereda 27
30	Ethiopia	Afar	Argobba (special woreda)	Argobba
31	Ethiopia	Afar	Zone 1 (Awsi Rasu)	Afambo
32	Ethiopia	Afar	Zone 1 (Awsi Rasu)	Asayita
33	Ethiopia	Afar	Zone 1 (Awsi Rasu)	Chifra
34	Ethiopia	Afar	Zone 1 (Awsi Rasu)	Dubti
35	Ethiopia	Afar	Zone 1 (Awsi Rasu)	Elidar
36	Ethiopia	Afar	Zone 1 (Awsi Rasu)	Kori
37	Ethiopia	Afar	Zone 1 (Awsi Rasu)	Mille
38	Ethiopia	Afar	Zone 2 (Kilbet Rasu)	Abala
39	Ethiopia	Addis Ababa	Zone 5	Wereda 10

Figure 5.18: Address hierarchy file created for Ethiopia addressing system

5.8.3 APPOINTMENTS

In order to have working appointments functionality in Bahmni EHR, three steps need to be done.

The first step is to modify the Bahmni EHR data model, by adding two fields to the person attribute types: "Telephone Number" and "Unknown Patient", which are required by the appointments module. This can be done directly on the *openmrs* database by using SQL insert statements or alternatively by creating (and running) a Liquibase change set with this changes to be added, which is advantageous because it keeps a consistent and upgradeable data model whenever there are major changes to it e.g. upstream changes like in the case of a data model update (Figure 5.19).


```

1 <changeSet id="IMPL-PERSON-ATTRIBUTE-TELEPHONE-NUMBER" author="Swathi, Jaswanth" context="rel0.76">
2   <preConditions onFail="MARK_RAN">
3     <sqlCheck expectedResult="0">
4       SELECT COUNT(*) FROM person_attribute_type where name = 'Telephone Number';
5     </sqlCheck>
6   </preConditions>
7   <comment>Adding Telephone Number person attribute type</comment>
8   <sql>
9     INSERT INTO person_attribute_type (name, description, format, searchable, creator, date_created,
10    retired, sort_weight, uuid) VALUES ('Telephone Number', 'Telephone Number', 'java.lang.String',
11    '0', 1, now(), 0, 3, uuid());
12   </sql>
13 </changeSet>
14 <changeSet id="IMPL-PERSON-ATTRIBUTE-UNKNOWN-PATIENT" author="Swathi, Jaswanth" context="rel0.76">
15   <preConditions onFail="MARK_RAN">
16     <sqlCheck expectedResult="0">
17       SELECT COUNT(*) FROM person_attribute_type where name = 'Unkown patient';
18     </sqlCheck>
19   </preConditions>
20   <comment>Adding Unkown patient person attribute type</comment>
21   <sql>
22     INSERT INTO person_attribute_type (name, description, format, searchable, creator, date_created,
23    retired, sort_weight, uuid) VALUES ('Unkown patient', 'Unkown Patient', 'java.lang.String', '0',
24    1, now(), 0, 3, uuid());
25   </sql>
26 </changeSet>

```

Figure 5.19: Alternative to the above method by using a Liquibase change set

The second one is to enable managed access to the appointments module functionality on the Bahmni EHR user-interface, the following JSON code shown (Figure 5.20) needs to be added to the file extension.json (present in /default-config/openmrs/apps/home/).

```

{
  "id": "bahmni.appointments",
  "extensionPointId": "org.bahmni.home.dashboard",
  "type": "link",
  "label": "Appointments",
  "url": "../../openmrs/appointmentschedulingui/home.page",
  "icon": "fa-user",
  "order": 11,
  "requiredPrivilege": "App: appointmentschedulingui.home"
}

```

Figure 5.20: Configuration for integrating the appointments feature with the Bahmni EHR user-interface

Finally, in order to interact with the module, users that will require interaction with the appointment features need to be associated with the role "App:appointmentschedulingui.home". The module brings other roles for managing access to other appointments functionalities, which also need assignment to users that require them. The URL field in the code points to the root point of the appointments module user-interface, from which the functionality is mapped. This mapped URL is under the modern UI interface, which the appointments module depends on.

5.8.4 PROVIDING CUSTOM FORMS

In Bahmni EHR, the method with which the form-subsystem defines forms is supported by the OpenMRS concept dictionary. Both observations and laboratory tests can be defined by using the dictionary, by configuring the respective defining concepts. Once a new concept defining a form is created, adding it to the observations area is done

by making it an *answer* of the concept named "All Observation Templates". Concepts defining panel tests can be added as answer to the concepts relevant to laboratory tests, either by categorizing them under the concepts of "Panel concept sets" or "All tests and panels" adds it to the respective section laboratory orders' area.

The interoperability support between Bahmni EHR and OpenELIS not only synchronizes order requests and results, but also the laboratory tests themselves, i.e., the structures that define them. When a user defines a new or an existent laboratory test in Bahmni EHR, the definition is sent and replicated on OpenELIS and vice-versa.

5.8.5 ENABLING OFF-LINE FEATURES

In order to enable the active notification for network status on the client, modifications need to be done in the file `app.json` in `default-config/openmrs/apps/clinical/`:

```
{
  "config": {
    "networkConnectivity" : {
      "showNetworkStatusMessage": true,
      "networkStatusCheckInterval": 5000
    }
  }
}
```

This is achieved by setting the field "showNetworkStatusMessage" to the value "true". Optionally a interval for polling the connection status can also be set (in miliseconds) on the field "networkStatusCheckInterval".

Off-line support is enabled in the server by the installation of the event-log service.

Off-line support of some dynamic functionality on the client applications requires that given supporting structures and data is available client-side. These can include for instance the observation templates. The supporting structures to be exported are the concepts that define each observation form. The integration is automatic but requires making the server aware of which structures are to be exported. This can be achieved by running a python script containing this logic. ¹³.

After running the script, these concepts become marked in the database and are automatically exported via the event-log service to its clients when needed.

¹³<https://github.com/Bahmni/event-log-service/blob/master/event-log-service-webapp/src/main/resources/sql-scripts/copyOfflineConcepts.py>

5.8.6 SECURITY CONFIGURATION

Before entering production, some configurations for enhancing the server security and Bahmni login policies should be made.

Two-factor authentication

Bahmni EHR includes support for additional login protection. In Bahmni EHR, users and the system itself can benefit of added security mechanisms in case of vulnerable authentication credentials, e.g. in the case that given user credentials are known to an unwanted person. This can be done user-wide, by enabling two factor authentication on the accounts. In Bahmni EHR, the second factor is a one-time-password which is sent to the user through Short Message Service (SMS). The system requires configuration of a compatible SMS gateway for sending the messages, and each user account requires entry of a phone number. In the case that any of these fails, the user is unable to login.

The SMS gateway can be added to Bahmni through the creation of a plug-in for the Two factor authentication (2FA) sub-system. Bahmni project provides a plug-in that connects to a third-party service, <https://msgateway.me>, which leverages the sending of SMS to an actual phone with that capability. ¹⁴

The second step is to add a phone number for each user account, however since there is no way for configuring a telephone number through the user-interface, it needs to be added directly on the database. This can be done by executing, for example, the following Structured Query Language (SQL) statement in the 'openmrs' database, which adds a phone number and associates it with the user "doctor1":

```
INSERT INTO contact(user_name, country_code, mobile_number)
  values('doctor1', '251', '9123456');
```

Additionally, the role "bypass2fa" needs to be assigned to users which cannot use 2FA, for example, the *reports-user* used by the reports module.

Optionally, the One-time-password (OTP) generated by 2FA has some configurable parameters which can increase or relax its security, like code expiration time (minutes) and length and maximum number of accepted attempts and retries, with default values 15, 6, 3, 3 respectively.

User-space isolation

The example vagrant machine provided by the Bahmni project that deploys Bahmni's version 0.80 comes with SELinux security policy set with the permissive value, effectively not enforcing any security policy provided by this subsystem. However, the GNU/Linux

¹⁴sms-gate-way-me plug-in: <https://github.com/Bahmni/bahmni-sms-plugins/>

distribution used, CentOS, comes with SELinux enabled by default, with the security policy set to enforcing, which is preferred to keep. However, preserving this configuration value created some issues when running Bahmni EHR because some components could not communicate since SELinux would block the connection. The result was that Bahmni could not access the OpenMRS backend, making its web interface output an access error. Solving this issue was overcome by setting the role "httpd_can_network_connect" to enabled. This role is assigned to the Apache HTTP Server, so enabling it allowed Apache to communicate with the OpenMRS backend that is running on Tomcat, in our case, on the same machine.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - SELinux is fully disabled.
SELINUX=enforcing
# SELINUXTYPE= type of policy in use. Possible values are:
#   targeted - Only targeted network daemons are protected.
#   strict - Full SELinux protection.
SELINUXTYPE=targeted
```

Figure 5.21: SELinux configuration file 'conf' in /etc/selinux/

SSH authentication policy

In order to configure an authentication policy, Fail2Ban was installed in the server. Fail2Ban is configured by the file present in /etc/fail2ban/jail.local. It uses the Netfilter subsystem that is already configured in CentOS.

```
[DEFAULT]
# "ignoreip" can be an IP address, a CIDR mask or a DNS host. Fail2ban will not
# ban a host which matches an address in this list. Several addresses can be
# defined using space (and/or comma) separator.
ignoreip = 127.0.0.1/8
# (...)
# "bantime" is the number of seconds that a host is banned.
bantime = 3600
# A host is banned if it has generated "maxretry" during the last "findtime"
# seconds.
findtime = 600
# "maxretry" is the number of failures before a host get banned.
maxretry = 5
```

Figure 5.22: SELinux configuration file 'jail.local' in /etc/fail2ban/

The *bantime* configuration parameter is responsible for the period in which a client that tried authenticate himself during a *findtime* value and failed a number of times defined in the *maxretry* configuration, is banned from authenticating again. The figure

shows that the configuration was made so that the client will be banned for one hour, if he failed to authenticate 5 times in a duration of 10 minutes or less.

Server firewall

Firewalling the server at the operating system level can be provided by Netfilter. Netfilter is a networking subsystem of the Linux kernel that provides stateful or stateless packet filtering, as well as NAT and IP masquerading services¹⁵. This subsystem can be controlled by using the IPTables tool. It comes set by default in CentOS and with standard firewall rules. For the EHR solution, new rules were configured for white listing the ports necessary for the EHR systems.

Software updates

As security vulnerabilities are identified and fixed in the software used in the CentOS distribution, new packages are distributed through the respective repositories. The same happens for new releases of the Bahmni software, which when pushed through the respective repository, are then upgradeable using the same distribution's package manager. Administrative intervention is required in case of any technical difficulties or for issuing the update of system software for keeping the system secure.

¹⁵https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/ch-fw.html

RESULTS

In this chapter we present the deployed solution, Gondar EHR System, implemented as seen in the previous chapter. The results presented show what is changed for this implementation when compared to a standard Bahmni implementation. Some features that correspond to given functional requirements but that were not modified in some way are not presented (for instance, the reporting functionality or patient's document management). The implementation is set up on a managed server running CentOS 6.7 64-bit on a quad-core Intel Xeon X5650 @ 2.67GHz, with 8 GB of RAM assigned to the operating system and 100GB of disk space, divided in half to two *Extended-4* file system partitions, 50GB for the *root* directory and another 50GB partition for the *home* directory.

6.1 USERS AND ROLES

Users are created with associations to system-present roles that dictate what the user can do in that system. The roles list is exhaustive, divided in groups of features, subdivided in access conditions related to those features, enabling a system administrator to create different users with different fields of activity.

The web interface presents an unverified user with a login screen. A user specifies the credentials of his account in order to access the Gondar EHR system. The *location* parameter associates the selected location to actions that are made during the user's session, for instance, laboratory tests ordered from the consultation office.

The web interface presents the main panel to an authorized user where the different areas of the system are represented by feature tiles. These are available if they are

configured as so in the implementation and based on the user access permissions. The system limits the user access based on the roles associated with his account. Figure 6.1 shows all the feature tiles available, since all the existent system roles are associated with this user. A staff member from another field of activity should be assigned roles that match his work-counterpart. For instance, a user representing a registrant clerk should be given roles corresponding to registration functionality, which would also make available to him the feature tile "Registration".

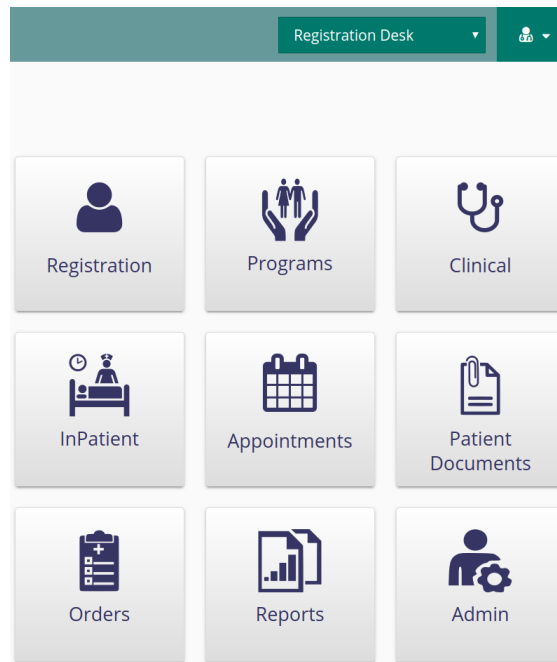


Figure 6.1: Gondar EHR main page, presenting all the available feature tiles.

The system administrator, or a user with the assigned roles "Edit User" or "Edit Privilege", is responsible for user creation and role assigning. Figure 6.2 shows some users with the different tasks that have been associated with them.

System Id	Username	Given	Family Name	Roles
7-5	automation	automation	automation	Privilege Level: Full
15-8	doctor1	Ermias	Diro	Appointments , Nurse ,
9-1	sahas	Sahas	Kumar	RegistrationClerk
11-7	suresh	Suresh	Kumar	bahmni-document-uploader
10-9	varsha	Varsha	Kumari	Nurse
16-6	nurse1	Tadele	Mulaw	RegistrationClerk , Appointments , Nurse ,
14-1	registrator	Registration	Staff	RegistrationClerk , Provider , InPatient-Patient-Movement , ...

Figure 6.2: List of users of Bahmni EHR and their associated roles. Includes the created users for doctor, nurse and registrant members.

The administration area of the system allows an administrator to create multiple users, and assign roles to them. This allows designing and mapping real-life staff roles to users within the system.

For Gondar EHR three new users were created: doctor, nurse and a registrant. These are representational of their work and can be used as models for creating other similar users. For instance, the registrant user models a staff member which is responsible for registering patients, managing appointment scheduling or in-house patient management. For this, the registrant user is only associated to roles which only provide access to this functionality. The main page (Figure 6.3).

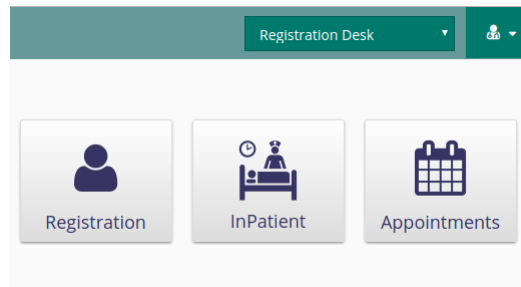


Figure 6.3: Registration clerk available features in Gondar EHR

6.2 REGISTRATION EXTENSION

The registration page in Gondar EHR is the entry point of patients in the system, analog to a patient registration sheet in Gondar LRTC. In this page, the patient's basic information like name, gender and birth date is entered. When the birth date is unknown, not an uncommon event in low-resource countries, an estimated age can be set. A photograph of the patient can be optionally attached or taken through a camera connected to the device that is presenting the web page.

The registration form allows the capture of patient names in their local language alphabet in the field "Name in local language", and a language or alphabet translation can be registered under "Patient Name". This gives foreigners not accustomed to the local language alphabet an easier way to read or search a patient's name. In Figure 6.4 we see an example of a patient being registered with his name in Amharic, the local language in Ethiopia, and his name in Latin alphabet. The system supports the Unicode encoding, thus multiple alphabets are supported in both fields.

Figure 6.4: Customized patient registration page

The registration page was also extended to support Ethiopia’s addressing system and person attributes that were captured in Gondar LRTC from their paper forms. One example of this is the *former ID* used in Gondar LRTC called "Hospital chart number". Attributes have a regular expression that validates its pattern, for instance in the case of the former ID the expression is: $[0-9]\{5,6\}\backslash/[0-9]\{2\}$. This validation is not only applied on registration, but at any time of modification of the given attributes.

The addressing system was complemented by providing input choices and auto-completion of real-world addresses in specific fields, e.g. zones, regions and districts. The auto-completion of Ethiopian addresses works down-to-top, so, if a lower-level address field is uniquely identified in the interface, the upper-levels are automatically filled in, a feature that prevents human typing mistakes and speeds up the inputting of an address. Each level can be selected independently, i.e., the user is not constrained to start filling the lower level, he can start from the most upper-level and work down, while the interface keeps filtering down based on his upper-level selections. It is also helpful for keeping address fields coherent, meaning that it is less likely to have two different textual representations of the same address.

The registration page allows printing of a card for handing out to the patient, which

contains a generated UID, basic personal information and/or recommendations. The patient can bring this card when he visits the health care center to aid the reception clerk to open his profile more quickly. The card information was extended to support some custom fields of Gondar LRTC, like *primary relative*, and can also be printed in the local language (Amharic). Figure 6.5 shows the English version of the card customized for patients of Gondar LRTC.


<p>Notice</p> <ul style="list-style-type: none"> • It is quite important to bring the card to the hospital. • Take medicines according to instructions of the doctor. • Registration Timings are from 8:30 am to 12:00 pm. • OPD days are on Monday, Wednesday, Friday. • Sundays are holidays. <p>Caution</p> <p>Smoking, Tobacco chewing can cause cancer, heart diseases and asthma.</p>	 <p>Leishmania Research and Treatment Center (Registered)</p> <p>Gondar, Semien Gondar - Amhara</p> <p>Dated : 02 Jun 16 Reg. ID : GLC200073 H.Chart No : 12345/16 Name : እትዮጵያ አፍሪቃ ውስጥ ናት Age : 22 y Gender : M Primary relative : PaiNome Village : Kala District : Gondar (town)</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Figure 6.5: Patient printable card, english version

6.3 PATIENT MANAGEMENT EXTENSION

In Gondar EHR, the system work-flow for creating a new patient goes through a patients search page. Making a search for a patient before registering it minimizes the chances of duplicate registrations.

The search uniquely identifies a patient if an UID is provided. The search page was adapted for Gondar LRTC through the addition of other fields that can be used to search patients, for instance by their local names (usually registered in Amharic), or by a given village of their address (Figure 6.6).



ID	Name	Given Name Local	Middle Name Local	Family Name Local	Gender	Age	Village	Registration Date
GLC200073	Teste Apellido	እትዮጵያ	አፍሪቃ	ውስጥ ናት	M	22	Kala	02 Jun 16

Figure 6.6: Search page which is displayed before registration.

While patients have a visit opened in Gondar LRTC, they become active in the clinical area in Gondar EHR. The search page of the clinical area can quickly provide a preview of active patients in the clinic along with their current hospitalization status while providing different types of filtering. An active patient can have a separate patient clinical status in the system which represents if he is an in-patient or out-patient. Figure 6.7 shows this:

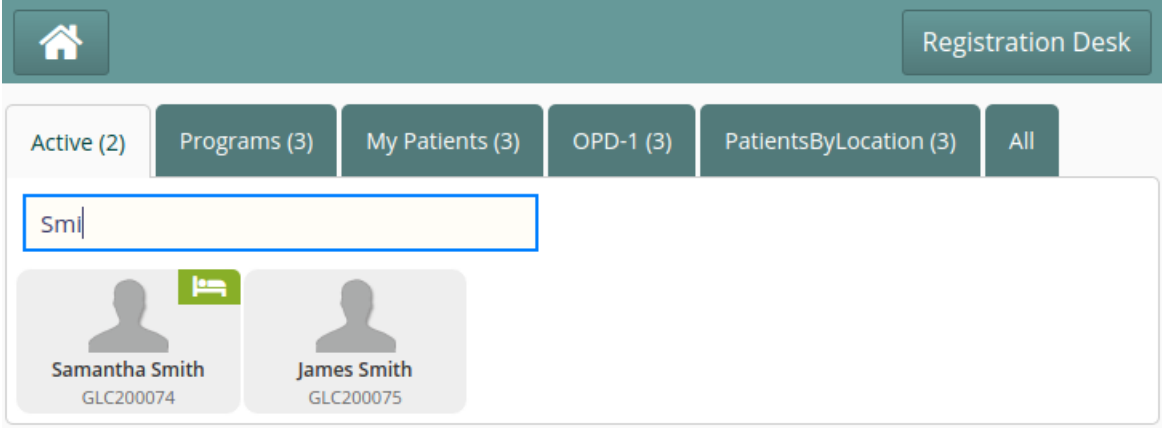


Figure 6.7: View and filter active patients in the clinic

Gondar EHR is equipped to handle hospitalization of patients, for instance, a out-patient under visitation can be admitted to the clinic under some conditions and become an in-patient. This flow between states is represented on the Figure 6.8.

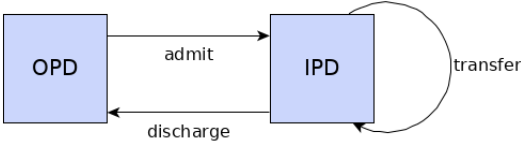


Figure 6.8: Admit, Discharge, Transfer

Managing a patient between these states is done under consultation by using admission, discharge and transfer controls (Figure 6.9). Discharging a patient will also release a bed resource automatically.

Figure 6.9: Patient movement control, with additional notes support.

Management of in-patients and available bed resources can also be done separately from consultation and may be available to other users since it is associated with a separate access role (Figure 6.10).

Bed	Name	Id	Age	District	Village	Admission By	Admission Time	Disposition By	Disposition Time	ADT Notes	Diagnosis
304-a	mogos tiruneh	GLC200084	35	Chilga		Ermias	24 Nov 16 11:39 AM	Ermias	24 Nov 16 03:21 PM		VL Confirmed Secondary Inactive

Figure 6.10: In-patient management area, showing a patient assigned to a bed in the "General Ward".

Support for editing the room layout and bed count in the user-interface is not provided by Bahmni EHR. Changes to these required direct modification of the corresponding entities in the database.

Bahmni EHR was extended with basic support for appointment management (Figure 6.11). Administrators can configure provider schedules, their duration and available

days, and other users with the right roles can schedule patients visits on provider schedules open slots. This integration is a workaround for presenting the feature in Bahmni EHR. It uses the OpenMRS modern UI for leveraging the appointment schedule features from the provided module. Its access control is integrated with Bahmni EHR and there is no need to re-login, however the user-interface is not linked back to Bahmni EHR, which can pose a navigation issue. This can be solved by opening the feature in a separate browser window.

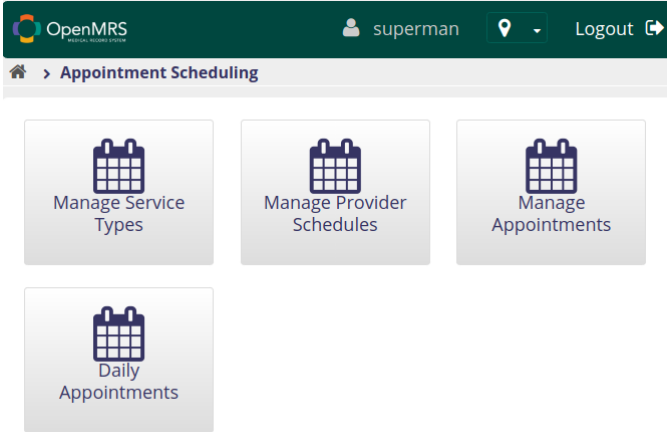


Figure 6.11: Appointment scheduling user-interface (modern UI).

6.4 CLINICAL EXTENSION

The clinical area in Gondar EHR provides most of medical practice management software features and is where integration with the other platforms come useful. The patient profile window presents an overview of patient clinical information, like latest observations, test results from the laboratory, medications prescribed, and so on (Figure 6.12). If the patient is active in the clinical area, meaning that a visit is opened for him, a provider can start a consultation and produce medical information associated with that encounter.

General | Registration Desk | Consultation

Samantha Smith (GLC200074) | Female | 45 Years 2 months 15 days

Samantha Smith (GLC200074) - Female, 45 Years 2 months 15 days (est.)
 Test Address
 Date of Birth: 30 Jun 71 (est.)

Disposition
 Admit Patient: 05 Jul 16
 Super Man 2:54 pm

Radiology
 No documents for this patient.
 No navigation links available.

Pacs
 No Radiology Order for this patient

Lab Results
 Accession at 05 Jul 16 2:54 pm
 ANC (Blood)

Vitals
 No Vitals for this patient

History and Examinations
 No History and Examinations for this patient

Obstetrics
 No Obstetrics for this patient

Admission Details
 Admission Date: 04 Jul 16
 Has become Inpatient: Super Man 10:06 am

Diagnosis
 Lymphadenitis, chronic CONFIRMED PRIMARY 05 Jul 16

Treatments
 No treatments for this patient.

Programs
 No active/inactive programs for this patient.

Radiology Orders
 No Radiology Orders for this patient

Lab Orders Display Control
 ANC (Blood) Super Man 05 Jul 16 2:54 pm
 No observations captured for this order.
 Hematology Super Man 04 Jul 16 10:04 am

Nutritional Values
 14 Sep 16 2:44 pm
 WEIGHT: 64
 HEIGHT: 173
 BMI: 21.38
 BMI STATUS: Normal
 04 Jul 16 9:51 am

Second Vitals
 No Second Vitals for this patient

Gynaecology
 No Gynaecology for this patient

Visits
 04 Jul 16 ★ IPD
 04 Jul 16 - 04 Jul 16 OPD

Bacteriology Results
 No specimen for this patient

Figure 6.12: Patient dashboard

One important feature in Gondar EHR is the management of laboratory orders. This feature shows the integration between Gondar EHR and OpenELIS for achieving a laboratory workflow encompassing several users across these two platforms. On Gondar EHR several tests are available organized under different categories and types. Single or multiple tests can make part of a laboratory order (Figure 6.13).

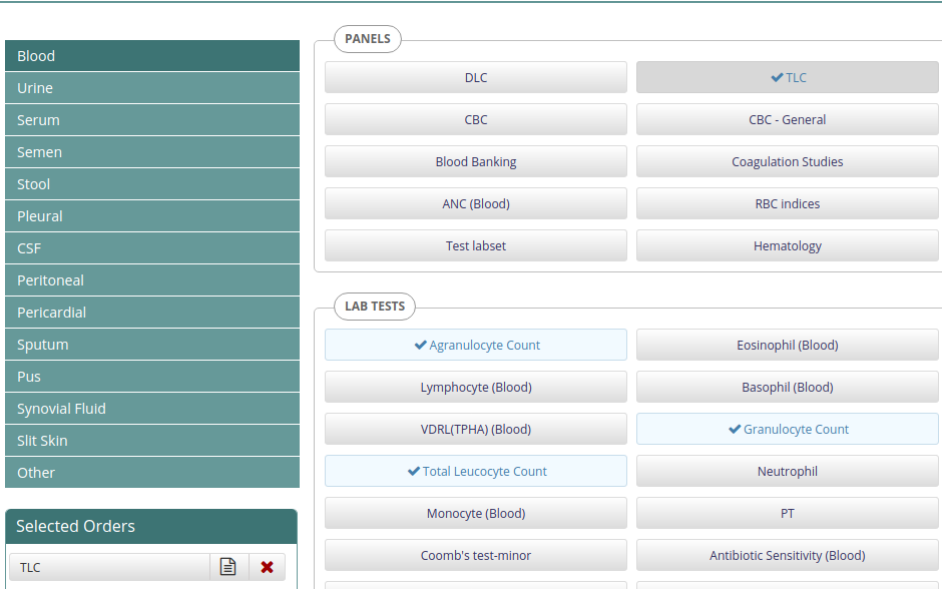


Figure 6.13: Gondar EHR available tests in laboratory orders

When in consultation, a doctor can order a set of tests which are sent automatically to the laboratory platform. A laboratory clerk can then collect samples, execute these tests and input the results on the respective orders in OpenELIS (Figure 6.14).

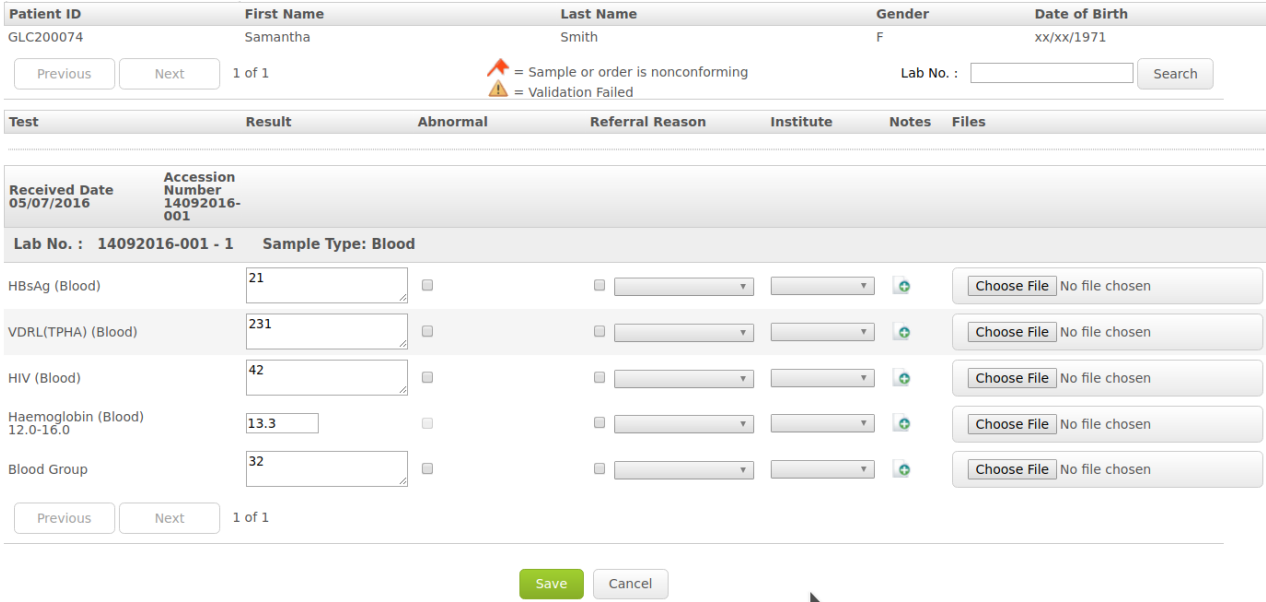


Figure 6.14: Entering results for multiple ordered tests in OpenELIS.

A separate member validates the test in OpenELIS, after which the results are sent automatically back to Gondar EHR. The doctor can view the test results on the patient profile without ever needing to access the laboratory platform, and laboratory staff can manage the orders, results and validation in their own platform without ever needing to access Gondar EHR. This enables segregation of responsibilities between the teams, which have access only to the domains of information needed on each work part. This is illustrated in Figure 6.15.

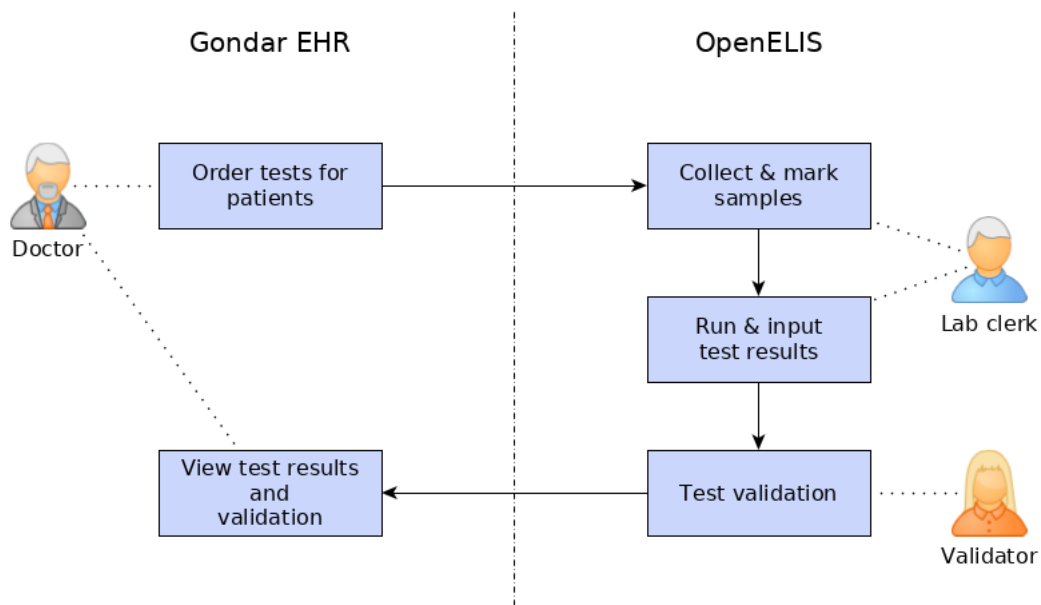


Figure 6.15: Laboratory workflow between Gondar EHR and OpenELIS and different users on both platforms

The laboratory orders has new available tests that were created for Gondar EHR following their laboratory forms.

The Possibility to create new laboratory tests or observation forms was one of the requirements for the LRTC. Our EHR solution offers a system to build new observation templates and laboratory tests just by using the concept dictionary management features from OpenMRS legacy UI. This method works the same way as in a Bahmni EHR standard installation and was explained briefly on the implementation section.

6.5 VALIDATION

Validating the Gondar EHR System can be done by demonstrating that the system requirements are satisfied.

On a first phase, the process of validation comprises collecting data from usage of the Gondar EHR System by staff members of Gondar's LRTC. For this, a user-manual and testing scenarios for Gondar EHR and OpenELIS were developed and sent to Gondar's LRTC staff members.

The Gondar EHR System was deployed by July, however at that time there was lack of availability of staff to test the system. The start of an internal war in Ethiopia and other external factors also contributed to the absence of required data for providing a validation at the time of this thesis presentation.

Internal testing was made between members of the team to test requirements in the prototype. The next list shows which ones are in a functional state, marked by ✓ and others which were not tested or unimplemented are unmarked.

- A. Manage patients
 - A1. Register a new patient. The system should capture the following demographic information: ✓
 - * A patient must have a unique medical number within the system ✓
 - * Patient's first name, surname, name in native language (not mandatory) ✓
 - * Sex, date of birth, place of birth, physical address, telephone contact ✓
 - * Biometric parameters as height, weight, blood pressure, BMI (calculated). ✓
 - * Allergies on medicine or products ✓
 - * Social history elements: marital status, occupation, socioeconomic status, and education. ✓
 - * Hospital visit dates, admission, discharge dates (if applicable), chief complaint, diagnosis, procedures performed. ✓
 - A2. Edit a patient information ✓
 - A3. Search a patient ✓
 - * A3.1. Search patient by medical number, names. ✓
- B. Out Patient Department (OPD)
 - B1. Consultation chambers
 - B2. Examination rooms
 - * B2.1. Send test request to the laboratory ✓
 - * B2.1 Receive/have an access the laboratory test results. ✓

- B3. Diagnostics
 - * B3.1. Laboratory must receive requests from examination room ✓
 - * B3.2. Lab assistant can create requests other than from LRTC ✓
 - * B3.3. Lab assistant can enter test results into the system ✓
 - * B3.4. Test result must be verified by lab staff member ✓
 - * B3.5. When test results have been verified, they must be available ✓
- C. In Patient Department (IPD)
 - C1. Set total number of beds grouping by children, men and women.
 - C2. Change patient status from outdoor to indoor and vice-versa. ✓
 - C3. Show number of occupied and available beds. ✓
- D. Alerts and appointments
 - D1. Create an appointment with a physician (up to 6-9 months) ✓
 - D2. Send alert messages to a patient before 1-3 days of scheduled visit
 - D3. Edit an appointment ✓
 - D4. Delete an appointment ✓
- E. Reports
 - E.1. The system should build reports with set period. (The list of reports must be received by LRTC staff.) ✓
- F. System logs
 - F1. The system must track dates and time stamps all entries. ✓
- G. Confidentiality and Security
 - G1. System should support secure logon into the LRTC system. ✓
 - G2. System should provide analysis of audit trails and unauthorized access attempts. ✓

The system lacks a proper validation of these requirements and also the remaining non-functional requirements. Despite this, we are confident that the implemented system will satisfy their requirements and be used in the future at Gondar's LRTC.

CONCLUSIONS

In this chapter we take a retrospective view of the work done, and propose opportunities to its continuation.

We started by defining what constitutes an electronic health record, which technological foundations can it use or lay on, and give practical open-source examples that can leverage its development, along with some medical standards for interoperability with other applications. This made possible to do a reflection on the possibility of building an EHR system for use in low-resource settings, and provided an insight into potential issues that happen on these scenarios. From this, research was done to identify open-source EHR software which could be used in a low-resource setting implementation, which gave a perspective on the overall state and development progress of open-source EHR projects. After evaluation of several software and a drawn comparison, one suitable solution was found.

From gathered use-case scenarios of a real health-care center of a developing country, the system requirements were defined for the low-resource setting. From these, and assisted by all the previous investigation and further analysis of the chosen software, the system architecture was defined along with its implementation, which gave direction for building a prototype. This prototype was built and deployed locally with success and provides a good basis for understanding strengths and weaknesses.

Test scenarios and manuals were developed in order to gather feedback for providing system validation, however there was not enough data for demonstrating fulfillment of all the system-requirements. Although we failed to provide a validation, many important conclusions were obtained in the process.

7.1 FUTURE WORK

First and foremost the most obvious work proposed to be done future is to do a system validation. Although the provided implementation is a working version of Bahmni software, there are still functionalities that it provides but that were not integrated and others which were not tested. There is a lot of potential in this solution to build an even more advanced EHR system.

Proposed future work can be:

- Notification system: implement a flexible notification sub-system for Bahmni EHR. The sub-system could be used by other services to provide notifications like system messages, users communications or external events like new patient documents being available in the PACS system or laboratory results.

- Events agenda: implement a flexible calendar sub-system for Bahmni EHR. This would be a flexible sub-system for other services to provide actions based on certain conditions, like generating reports and sending them by email on a set frequency.

- Development of native appointment feature for Bahmni EHR: as seen in the system implementation, the appointment feature was integrated through a workaround to the appointment scheduling provided in another user-interface. Providing a new appointment management would not only provide a better user-experience but could also provide integration with the notification sub-system above.

REFERENCES

- [1] OpenEMR. OpenEMR, [Online]. Available: <http://www.open-emr.org> (Accessed on 07/25/2016).
- [2] OpenMRS. OpenMRS, [Online]. Available: <http://openmrs.org/> (Accessed on 03/25/2016).
- [3] N. A. Davis and M. LaCour, *Health information technology*. Elsevier Health Sciences, 2014.
- [4] F. Trotter and D. Uhlman, *Meaningful use and beyond*. Farnham: O’Reilly, 2011, ISBN: 978-1-4493-0502-4 1-4493-0502-4.
- [5] Open Source Initiative. Licenses by name, [Online]. Available: <https://opensource.org/licenses/alphabetical> (Accessed on 03/01/2016).
- [6] Apache Software Foundation. Welcome! - the apache http server project, [Online]. Available: <https://httpd.apache.org/> (Accessed on 03/31/2016).
- [7] Oracle. Mysql, [Online]. Available: <http://www.mysql.com/> (Accessed on 03/31/2016).
- [8] T. Benson, “HL7 Version 2”, in *Principles of Health Interoperability HL7 and SNOMED*, London: Springer London, 2012, pp. 101–119, ISBN: 978-1-4471-2801-4. [Online]. Available: <http://dx.doi.org/10.1007/978-1-4471-2801-4%3Csub%3E7%3C/sub%3E>.
- [9] G. W. Beeler, “HL7 Version 3—An object-oriented methodology for collaborative standards development”, *International journal of medical informatics*, vol. 48, no. 1, pp. 151–161, 1998.
- [10] D. Bender and K. Sartipi, “HL7 FHIR: An Agile and RESTful approach to healthcare information exchange”, in *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, IEEE, 2013, pp. 326–331.
- [11] N. E. M. Association *et al.*, *Digital Imaging and Communications in Medicine (DICOM).: Parts 1-10*. The Association, 1993.
- [12] Luis Falcon. GNU Health - Summary [Blog post], [Online]. Available: <http://savannah.gnu.org/projects/health/> (Accessed on 04/10/2016).
- [13] WikiBooks. GNU Health/The core module, [Online]. Available: https://en.wikibooks.org/wiki/GNU_Health/The_core_module (Accessed on 04/10/2016).
- [14] Luis Falcon. Success of GNU Health goes beyond free software [Interview], [Online]. Available: <http://opensource.com/health/13/3/interview-luis-falcon-gnu-health> (Accessed on 10/10/2016).
- [15] WikiBooks. GNU Health/Security, [Online]. Available: https://en.wikibooks.org/wiki/GNU_Health/Security (Accessed on 10/10/2016).
- [16] Luis Falcon. (2015). GNU Health - News: GNU Health 2.8 series released [Blog post], [Online]. Available: http://savannah.gnu.org/forum/forum.php?forum_id=8199.

- [17] Tryton. New Tryton release 3.8 [Blog post], [Online]. Available: <http://www.tryton.org/posts/new-tryton-release-38.html> (Accessed on 10/10/2016).
- [18] WikiBooks. GNU Health/Modules, [Online]. Available: https://en.wikibooks.org/wiki/GNU_Health/Modules (Accessed on 04/10/2016).
- [19] Unknown. GNU Solidario: Translation Server, [Online]. Available: <http://translate.gnusolidario.org/projects/GNUHEALTH/> (Accessed on 10/11/2016).
- [20] Luis Falcon. GNU Health - News: GNU Health 3.0 released (Translation portal moves to Pootle) [Blog post], [Online]. Available: http://savannah.gnu.org/forum/forum.php?forum_id=8437 (Accessed on 10/11/2016).
- [21] ONC-Authorized Certification Body. OpenEMR Certification, [Online]. Available: <https://chpl.healthit.gov/#/product/43> (Accessed on 11/20/2016).
- [22] Unknown. OpenEMR Success Stories, [Online]. Available: http://www.open-emr.org/wiki/index.php/OpenEMR_Success_Stories (Accessed on 04/25/2016).
- [23] OpenEMR Wiki. Patient Portal - OpenEMR, [Online]. Available: http://www.open-emr.org/wiki/index.php/Patient_Portal (Accessed on 08/25/2016).
- [24] —, OpenEMR Form Creation Tools, [Online]. Available: http://www.open-emr.org/wiki/index.php/OpenEMR_Form_Creation_Tools (Accessed on 07/25/2016).
- [25] —, LBV Forms - OpenEMR, [Online]. Available: http://www.open-emr.org/wiki/index.php/LBV_Forms (Accessed on 07/25/2016).
- [26] —, Nation Notes - OpenEMR, [Online]. Available: http://www.open-emr.org/wiki/index.php/Nation_Notes (Accessed on 07/25/2016).
- [27] —, OpenEMR XML Form Generator, [Online]. Available: http://www.open-emr.org/wiki/index.php/OpenEMR_Xml_Form_Generator (Accessed on 07/25/2016).
- [28] Sreevidya Krishna. Taking medical records into the digital age, [Online]. Available: <https://www.ibm.com/developerworks/websphere/library/techarticles/ind-openemr/> (Accessed on 11/20/2016).
- [29] ViSolve, Inc. OpenEMR: Achieving DICOM Interoperability using Mirth, [Online]. Available: http://www.visolve.com/uploads/resources/vicareplus/OpenEMR_DICOM_Interoperability_using_Mirth.pdf (Accessed on 11/25/2016).
- [30] OpenEMR Wiki. Codebase Security - OpenEMR, [Online]. Available: http://www.open-emr.org/wiki/index.php/Codebase_Security (Accessed on 11/25/2016).
- [31] —, Security Assessment - OpenEMR, [Online]. Available: http://www.open-emr.org/wiki/index.php/Security_Assessment (Accessed on 11/25/2016).
- [32] —, Encryption and Decryption of Documents - OpenEMR, [Online]. Available: www.open-emr.org/wiki/index.php/Encryption_and_Decryption_of_Documents (Accessed on 11/25/2016).
- [33] —, ACL Fine Granular Control - OpenEMR, [Online]. Available: http://www.open-emr.org/wiki/index.php/ACL_Fine_Granular_Control (Accessed on 11/25/2016).
- [34] J. Barkley, “Comparing simple role based access control models and access control lists”, in *In Proceedings of the Second ACM Workshop on Role-Based Access Control*, ACM Press, 1997, pp. 127–132.
- [35] OpenEMR Wiki. Zend - OpenEMR, [Online]. Available: <http://www.open-emr.org/wiki/index.php/Zend> (Accessed on 11/25/2016).

- [36] —, The OpenEMR API - OpenEMR, [Online]. Available: http://www.open-emr.org/wiki/index.php/The_OpenEMR_API (Accessed on 11/25/2016).
- [37] —, OpenEMR Professional Support, [Online]. Available: http://www.open-emr.org/wiki/index.php/OpenEMR_Professional_Support (Accessed on 04/25/2016).
- [38] —, OpenEMR IRC, [Online]. Available: http://www.open-emr.org/wiki/index.php/OpenEMR_IRC (Accessed on 04/25/2016).
- [39] Unknown. OpenEMR / Discussion / Forum, [Online]. Available: <https://sourceforge.net/p/openemr/discussion/> (Accessed on 04/25/2016).
- [40] OpenEMR Wiki. Demos - OpenEMR, [Online]. Available: http://www.open-emr.org/wiki/index.php/OpenEMR_Wiki_Home_Page#Demos (Accessed on 04/25/2016).
- [41] FreeMED Software Foundation. FreeMED Software Foundation, [Online]. Available: <http://freemedsoftware.org/> (Accessed on 05/08/2016).
- [42] LinuxMedNews. (2003). FreeMED Stable Release!, [Online]. Available: <http://linuxmednews.com:8080/linuxmednews/1055991998/>.
- [43] FreeMED. REMITT Electronic Medical Information Translation and Transmission, [Online]. Available: <http://remitt.org> (Accessed on 05/30/2016).
- [44] —, `freemed-0.8.x/modules/forms.emr.module.php` at master - Github, [Online]. Available: <https://github.com/freemed/freemed-0.8.x/blob/master/modules/forms.emr.module.php> (Accessed on 05/30/2016).
- [45] —, FreeMED localization, [Online]. Available: <https://www.transifex.com/freemed/freemed/> (Accessed on 05/30/2016).
- [46] OpenMRS. About OpenMRS, [Online]. Available: <http://openmrs.org/about/> (Accessed on 03/25/2016).
- [47] —, REST Module - Documentation - OpenMRS Wiki, [Online]. Available: <https://wiki.openmrs.org/display/docs/REST+Module> (Accessed on 06/20/2016).
- [48] Unknown. Form entry module, [Online]. Available: <https://wiki.openmrs.org/display/docs/FormEntry+Module> (Accessed on 03/29/2016).
- [49] —, Html form entry module, [Online]. Available: <https://wiki.openmrs.org/display/docs/HTML+Form+Entry+Module> (Accessed on 03/29/2016).
- [50] —, Html form entry designer module, [Online]. Available: <https://wiki.openmrs.org/display/docs/HTML+Form+Entry+Designer+Module> (Accessed on 03/29/2016).
- [51] J. M. Boyer *et al.*, *Xforms 2.0, w3c, 2015*.
- [52] W3C. XForms 2.0 - XForms Users Community Group, [Online]. Available: https://www.w3.org/community/xformsusers/wiki/XForms_2.0 (Accessed on 11/01/2016).
- [53] Unknown. Xforms module, [Online]. Available: <https://wiki.openmrs.org/display/docs/XForms+Module> (Accessed on 03/29/2016).
- [54] OpenMRS. OpenMRS and Health Interoperability Standards, [Online]. Available: <https://wiki.openmrs.org/display/docs/OpenMRS+and+Health+Interoperability+Standards> (Accessed on 06/20/2016).
- [55] —, Connecting OpenMRS to Other Systems Using Mirth - Documentation - OpenMRS Wiki, [Online]. Available: <https://wiki.openmrs.org/display/docs/Connecting+OpenMRS+to+Other+Systems+Using+Mirth> (Accessed on 06/20/2016).

- [56] —, Platform Release Notes 2.0.0 - Resources - OpenMRS Wiki, [Online]. Available: <https://wiki.openmrs.org/display/RES/Platform+Release+Notes+2.0.0> (Accessed on 11/29/2016).
- [57] —, User Interface Modules - Documentation - OpenMRS Wiki, [Online]. Available: <https://wiki.openmrs.org/display/docs/User+Interface+Modules> (Accessed on 06/20/2016).
- [58] Unknown. Openmrs modules, [Online]. Available: <https://modules.openmrs.org> (Accessed on 03/30/2016).
- [59] OpenMRS. OpenMRS localization, [Online]. Available: <https://www.transifex.com/openmrs/OpenMRS/> (Accessed on 03/11/2016).
- [60] —, OpenMRS Atlas, [Online]. Available: <https://atlas.openmrs.org> (Accessed on 04/19/2016).
- [61] —, Service Providers - Documentation - OpenMRS Wiki, [Online]. Available: <https://wiki.openmrs.org/display/docs/Service+Providers> (Accessed on 11/10/2016).
- [62] —, Releases OpenMRS, [Online]. Available: <http://openmrs.org/category/releases/> (Accessed on 03/01/2016).
- [63] —, OpenMRS localization, [Online]. Available: <https://talk.openmrs.org/tags/security-advisory> (Accessed on 03/11/2016).
- [64] —, OpenMRS License Information, [Online]. Available: <http://openmrs.org/license/> (Accessed on 03/25/2016).
- [65] Bahmni Project. Install Bahmni on CentOS - Bahmni Documentation Wiki, [Online]. Available: <https://bahmni.atlassian.net/wiki/display/BAH/Install+Bahmni+on+CentOS> (Accessed on 04/01/2016).
- [66] Bahmni - Mingle. Support for CentOS v7.x - Bahmni-EMR - Mingle Card, [Online]. Available: https://bahmni.mingle.thoughtworks.com/projects/bahmni_emr/cards/1922 (Accessed on 07/11/2016).
- [67] Bahmni. Bahmni localization, [Online]. Available: <https://talk.openmrs.org/c/software/bahmni> (Accessed on 04/11/2016).
- [68] —, Bahmni localization, [Online]. Available: <https://www.transifex.com/openmrs/bahmni/> (Accessed on 11/01/2016).
- [69] —, Implementations - Bahmni, [Online]. Available: <http://www.bahmni.org/implementations/> (Accessed on 04/01/2016).
- [70] “Introducing CentOS”, in *The Definitive Guide to CentOS*, Berkeley, CA: Apress, 2009, pp. 3–11, ISBN: 978-1-4302-1931-6. [Online]. Available: <http://dx.doi.org/10.1007/978-1-4302-1931-6%3Csub%3E1%3C/sub%3E>.
- [71] Darius Jazayeri. (2015). AngularJS and REST - omrs15 tutorial, [Online]. Available: <http://www.slideshare.net/sunbiz/angularjs-and-rest-omrs15-tutorial> (Accessed on 04/20/2016).
- [72] OpenMRS. Data Model - Documentation - OpenMRS Wiki, [Online]. Available: <https://wiki.openmrs.org/display/docs/Data+Model> (Accessed on 04/16/2016).
- [73] Bahmni Project. Connecting to various databases - Bahmni Documentation Wiki, [Online]. Available: <https://bahmni.atlassian.net/wiki/display/BAH/Connecting+to+various+databases> (Accessed on 04/16/2016).

- [74] —, EMR Security and Access Control (OpenMRS) - Bahmni Documentation Wiki, [Online]. Available: <https://bahmni.atlassian.net/wiki/pages/viewpage.action?pageId=48889893> (Accessed on 04/20/2016).
- [75] —, bahmni-environment/scripts/deletePatientData at GitHub, [Online]. Available: <https://github.com/Bahmni/bahmni-environment/blob/master/scripts/deletePatientData/> (Accessed on 04/01/2016).
- [76] Bahmni Forums. How to start with a fresh / clean database for Bahmni?, [Online]. Available: <https://talk.openmrs.org/t/how-to-start-with-a-fresh-clean-database-for-bahmni/> (Accessed on 04/01/2016).

APPENDIX

Relevant information to the subject matter.

Table 1: Login to system

Use case title	Login to system
Brief description	A user logs into the system. This use case establishes access permissions for various categories of users.
Primary actors	Nurse, physician, lab technologist, administrator
Secondary actors	None
Preconditions	A user starts the application
Main flow	1. The system displays the login screen 2. A user enters his credentials (username and password) 3. The system verifies provided information and establishes permissions for a user Login failure: a) If a username or password is incorrect, the system displays an appropriate message b) If a username does not exist in the system, a user needs to register in the system
Postconditions	A user entered the system

Table 2: Search a patient

Use case title	Search a patient
Brief description	A user submits a search request
Primary actors	Nurse or physician
Secondary actors	None
Preconditions	A user is logged into the system
Main flow	1. A user submits a search request. Possible parameters to search a patient are: patient name, patient ID, region 2. System returns matching results
Postconditions	System displays results for a user

Table 3: Register a patient

Use case title	Register a patient
Brief description	A user creates a new patient in the system
Primary actors	Nurse, physician, lab technologist
Secondary actors	None
Preconditions	A user is logged into the system. A new patient arrives.
Main flow	1. A user enters personal data of a patient 2. A user saves patient information
Postconditions	A new patient information is successfully saved in the system

Table 4: Schedule patient appointment

Use case title	Schedule patient appointment
Brief description	A user schedules a new patient appointment with a physician
Primary actors	Nurse, physician
Secondary actors	None
Preconditions	A user is logged into the system. A patient is registered in the system and needs to schedule an appointment with a physician
Main flow	1. A user checks doctor's availability 2. A user finds a slot in a doctor's calendar and creates an appointment 3. A user saves an appointment
Postconditions	A new patient information is successfully saved in the system

Table 5: Examine a patient

Use case title	Examine a patient
Brief description	A nurse or physician examines a patient for vital signs and visual screening
Primary actors	Nurse, physician
Secondary actors	None
Preconditions	A user is logged into the system. A patient arrives to consultation
Main flow	1. A user finds a patient in the system 2. A user enters examination results into the system 3. A user saves examination information
Postconditions	A patient's physical examination information is saved in the system

Table 6: Manage a patient

Use case title	Manage a patient
Brief description	The use case describes indoor and outdoor patient management. An additional use case "Bed allotment" is required for indoor patients
Primary actors	Physician
Secondary actors	None
Preconditions	A user is logged into the system. An outdoor patient needs to be treated in the hospital
Main flow	<ol style="list-style-type: none"> 1. A user finds a patient 2. A user checks bed availability 3. A user allots a bed for a patient
Postconditions	An outdoor patient changes his status to indoor patient

Table 7: Place an order

Use case title	Place an order
Brief description	A user places an order for laboratory
Primary actors	Nurse, physician
Secondary actors	None
Preconditions	A user is logged into the system. A patient is under examination
Main flow	<ol style="list-style-type: none"> 1. A user chooses necessary lab tests for a patient 2. A user saves an order 3. After saving an order, the system sends request to laboratory
Postconditions	A test order is sent to laboratory

Table 8: Receive order results

Use case title	Receive order results
Brief description	After an order results are validated, a lab technologist sends results to a physician
Primary actors	Nurse, physician
Secondary actors	None
Preconditions	A user is logged into the system. An existing order has results and is validated by lab technologist
Main flow	The system displays lab test results to a user
Postconditions	Order results are available for physician and nurse

Table 9: Make a diagnosis

Use case title	Make a diagnosis
Brief description	After receiving physical examination and lab test results a doctor makes a diagnosis for a patient and identifies a disease
Primary actors	Physician
Secondary actors	None
Preconditions	A user is logged into the system. Clinical examination and lab test results of patient are received and seen by a physician
Main flow	<ol style="list-style-type: none"> 1. A user finds a patient in the system and opens his medical records 2. A user make a diagnosis according patient tests 3. A user saves a diagnosis
Postconditions	A patient diagnosis is successfully saved in the system

Table 10: Discharge a patient

Use case title	Discharge a patient
Brief description	An inpatient is discharged when a clinical treatment finishes or he leaves a hospital
Primary actors	Physician
Secondary actors	None
Preconditions	A user is logged into the system. An inpatient is ready to leave a hospital
Main flow	<ol style="list-style-type: none"> 1. A physician decides to discharge an indoor patient 2. A physician changes a patient status to discharged
Postconditions	An indoor patient is successfully discharged from hospital

Table 11: Make a report

Use case title	Make a report
Brief description	The system creates a report according to given parameters by a user
Primary actors	Physician
Secondary actors	None
Preconditions	A user is logged into the system
Main flow	<ol style="list-style-type: none"> 1. A user enters report parameters such as period, report type and exported document format 2. The system builds a report, opens the file in a new window and displays it to a user
Postconditions	A report is built and exported

Table 12: Receive an order

Use case title	Receive an order
Brief description	A lab order is requested by clinical staff
Primary actors	Lab technologist
Secondary actors	None
Preconditions	A user is logged into the system. A new order arrives.
Main flow	1. The system notifies a user about new order 2. A user accepts a new order
Postconditions	A new order is accepted in laboratory

Table 13: Enter order results

Use case title	Enter order results
Brief description	After making laboratory tests a user enters results of a requested order
Primary actors	Lab technologist
Secondary actors	None
Preconditions	A user is logged into the system. A laboratory tests are done by a lab technologist
Main flow	1. A user finds an order 2. A user enters results of laboratory tests 3. A user sends results for further validation
Postconditions	Test results are sent for validation

Table 14: Validate order results

Use case title	Validate order results
Brief description	Laboratory test results require validation by another lab technologist
Primary actors	Lab technologist
Secondary actors	None
Preconditions	A user is logged into the system. Lab test results are in the system
Main flow	1. A user double checks results a) If lab test results are correct, a user validates them b) If lab test results have mistakes, a user does not validate them and another lab test is required
Postconditions	Order results are checked and validated

Table 15: Send order results

Use case title	Lab technologist
Brief description	A user sends validated order results to clinical staff
Primary actors	Lab technologist
Secondary actors	Nurse, physician
Preconditions	A user is logged into the system. Order results are validated
Main flow	<ol style="list-style-type: none"> 1. A user sends validated results to clinical staff 2. The system notifies clinical staff that results are received 3. The system displays order results in patient's history
Postconditions	Order results successfully transferred

Table 16: Add a staff member

Use case title	Add a staff member
Brief description	A user creates a new staff member according to a request
Primary actors	Administrator
Secondary actors	None
Preconditions	A user is logged into the system. A new staff member arrives.
Main flow	<ol style="list-style-type: none"> 1. A user enters personal data of an employee and gives appropriate permissions 2. A user saves information
Postconditions	A new user is created in the system

Table 17: Edit a staff member

Use case title	Edit a staff member
Brief description	If an existing user needs changes in his account this use case is employed
Primary actors	Administrator
Secondary actors	None
Preconditions	A user is logged into the system. A change into account is requested
Main flow	<ol style="list-style-type: none"> 1. A user finds an account 2. A user changes account setting according to a request 3. A user saves changes
Postconditions	New settings are applied for a staff member in the system

Table 18: Delete a staff member

Use case title	Delete a staff member
Brief description	A user deactivates a staff member in the system. Removing users from the system is not allowed
Primary actors	Administrator
Secondary actors	None
Preconditions	A user is logged into the system. An account deactivation request is received
Main flow	<ol style="list-style-type: none"> 1. A user finds an account of a staff member 2. A user deactivates an account 3. A user saves changes
Postconditions	A staff member account is successfully deactivated in the system

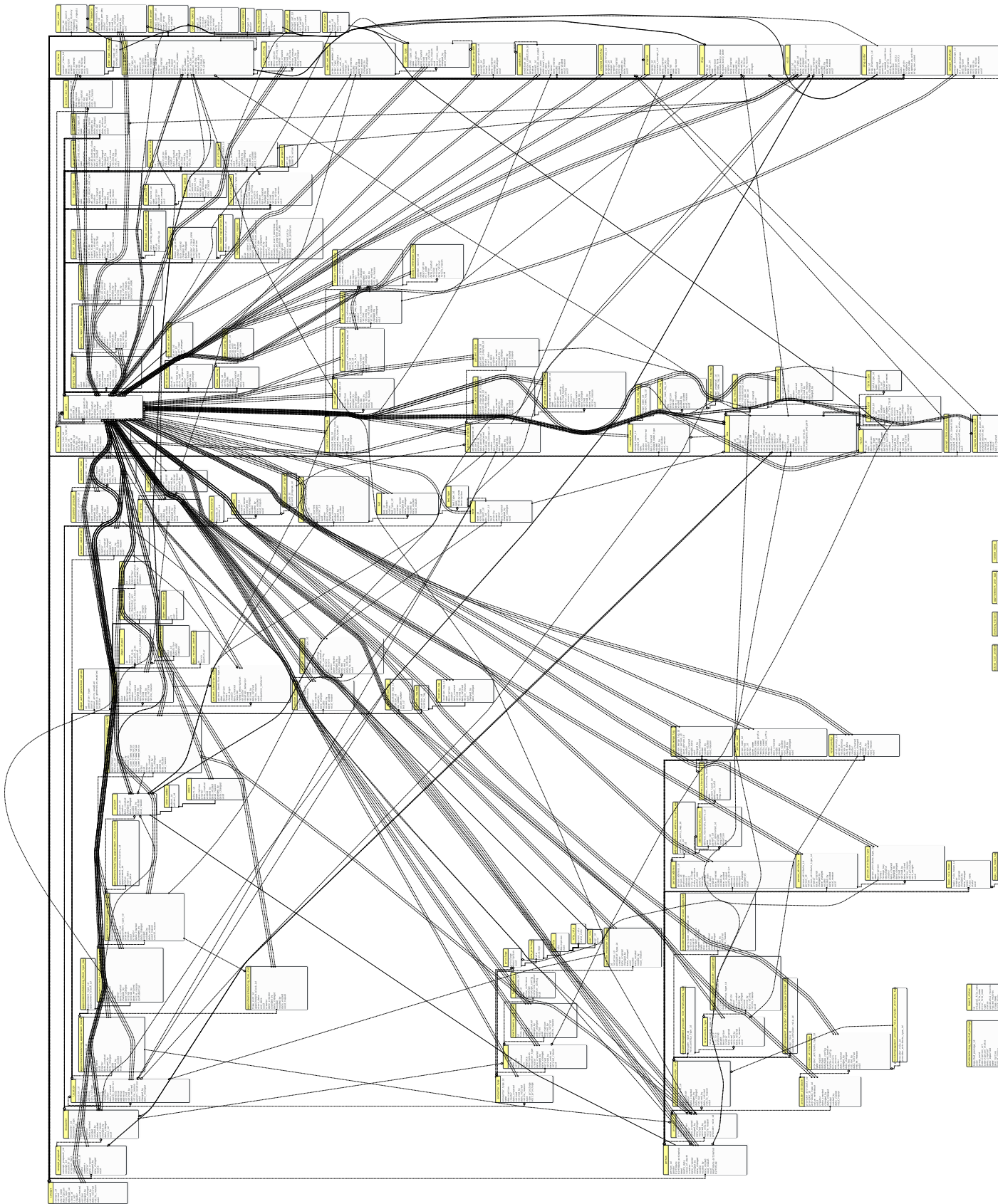


Figure 1: Bahmni EHR "openmrs" database schema - all tables and their attributes

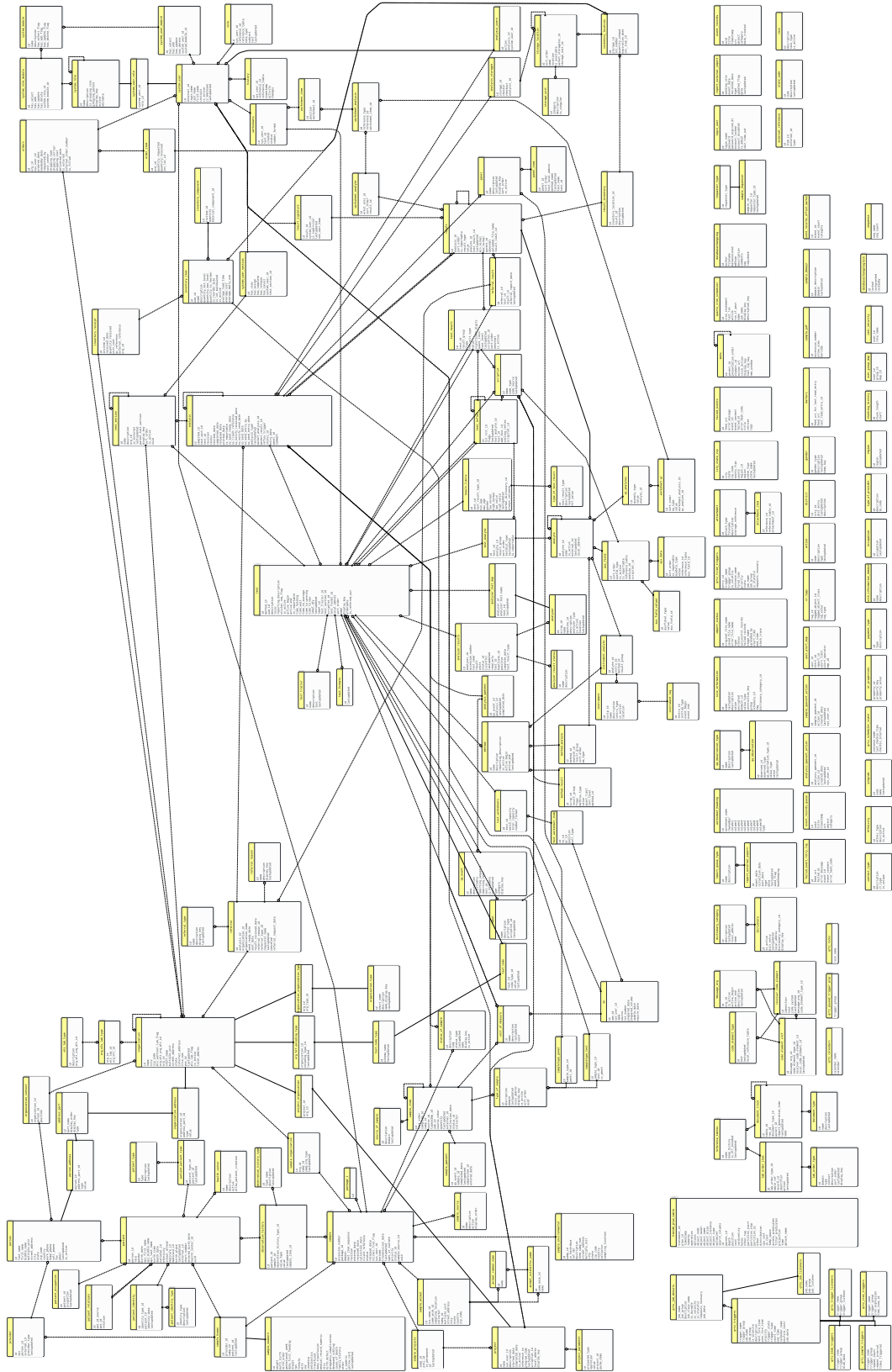


Figure 2: OpenELIS "clinlims" database schema - all tables and their attributes