
IDART DATA MART

By

ZUKILE RORO

A report submitted in partial fulfillment of the
requirements for the degree of BSc Honours
(Computer Science)

University of the Western Cape

2010

University of the Western Cape
Department of Computer Science
Supervisor: Dr William D. Tucker

ABSTRACT

IDART DATA MART

by Zukile Roro

Supervisor: Dr William D. Tucker
Department of Computer Science

The Intelligent Dispensing of ART (iDart) is the software solution designed by Cell-Life to support the dispensing of antiretroviral drugs in the public health sector.

The purpose of this project is to combine data from multiple instances of iDart into a single data mart that can be used by Cell-Life for analysis and reporting. The data mart design will use the star schema instead of snowflake schema. The advantage of using this schema is that it reduces the number of tables in the database.

A dashboard user interface will be used. Implementing a dashboard will allow Cell-Life to find an overall view of antiretroviral drug treatments. A High-Level Design provides an overview of the system, and includes a high-level architecture diagram depicting the components and interfaces that are needed. The low level design will contain: detailed functional logic of the module in pseudo code, database tables with all elements including their type and size, all interface details with complete API references(both requests and responses), complete input and outputs for a module(courtesy 'anonimas').

ACKNOWLEDGMENTS

First and foremost I would like to thank my family for their support, without them I wouldn't be where I am today. Then I wish to thank my supervisor Dr William D. Tucker for his kind supervision, advices and support.

Table of contents

Abstract	i
Acknowledgements	ii
Table of contents	iii
List of figures	iv
List of Tables	v
Glossary	vi
Chapter 1: Introduction	1
Chapter 2: User requirements	3
2.1 User's view of the problem	3
2.2 Expectations from a system	3
2.3 Not expected from a system.....	3
2.4 General constraints.....	3
Chapter 3: Requirements Analysis	5
3.2 User requirements interpretation	5
3.3 Suggested system	5
3.4 Testing the suggested solution	6
Chapter 4: User Interface Specification	7
4.1 What the user interface looks like to the user.....	7
4.2 How the user interface behaves	7
4.3 How the user interacts with the system.....	8
4.4 Suggested system	8
Chapter 5: High Level Design	11
5.1 Components	11
5.2 User interface design	11
5.3 Use case index	11
5.4 Class diagram.....	12
5.5 Schema.....	12
Chapter 6: Low Level Design	13
6.1 Details of class attributes	15
6.2 Details of class methods/functions	15
6.3 Pseudo-code	16

LIST OF FIGURES

Figure 1: IDART DATA MART CONCEPT	1
Figure 2: OVERVIEW OF THE SYSTEM	6
Figure 3: USER INTERFACE SPECIFICATION	8
Figure 4: KPI TOOLBAR	8
Figure 5: KPI EXAMPLE.....	9
Figure 6: KPI EXAMPLE CASE 1	9
Figure 7: KPI EXAMPLE CASE 2	10
Figure 8: USE CASE.....	12
Figure 9: CLASS DIAGRAM.....	13
Figure 9: DATA MART SCHEMA.....	13

LIST OF TABLES

Table 1: OBJECTS REQUIRED	11
Table 2: USE CASE INDEX TABLE	12
Table 3: A DESCRIPTION OF ATTRIBUTE	15
Table 4: A DESCRIPTION OF CLASS METHODS	15
Table 5: A DESCRIPTION OF FUNCTIONS/METHODS	16

GLOSSARY

ARV ó AntiRetroViral

iDart ó Intelligent Dispensing of ART

IDE - Integrated Development Environment is a software application that provides comprehensive facilities to computer programmers for software development.

Dashboard – A reporting tool that presents key indicators on a single screen, which includes measurements, metrics, and scorecards.

Data mart - It is a simple form of a data warehouse that is focused on a single functional area.

ETL - Extract, Transform, and Load is a process in database usage.

GUI - Graphical User Interface

HIV – Human Immunodeficiency Virus

KPI ó Key Performance Indicators

OOA – Object Oriented Analysis

OOD – Object Oriented Design

Pentaho – The Pentaho BI Project is open source application software for enterprise reporting, analysis, dashboard, data mining, workflow and ETL capabilities for business intelligence needs.

PostgreSQL– PostgreSQL, often simply Postgres, is an object-relational database management system (ORDBMS).

RA – Requirement Analysis

Star schema - is the simplest style of data warehouse schema.

Talend - is an open source data integration software vendor which produces several enterprise software products, including Talend Open Studio.

UIS - User Interface Specification

UR – User Requirements

INTRODUCTION

Any online transaction processing (OLTP) data contains information that can help in making informed decisions about businesses. For example, you can calculate your net profits for last quarter and compare them with the same quarter of the previous year. The process of analyzing your data for that type of information, and the data that results, are collectively called *business intelligence*. Because most operational databases are designed to store your data, not to help you analyze it, it's expensive and time consuming to extract business intelligence information from your database. The solution: an online analytical processing (OLAP) database, a specialized database designed to help you extract business intelligence information from your data.

In response to a request from the Desmond Tutu HIV Foundation to assist the management of ARV dispensing, the Intelligent Dispensing of ART (iDart) system was developed by Cell-life which in 2009 is in over 20 clinics dispensing drugs to more than 45,000 patients. This system is used by pharmacists to manage the supply of ARV stocks, print reports and manage collection of drugs by patients.

One of many iDart sites is the ARV pharmacy at the Tsepong Wellness Centre which became the third Elton Aids Foundation sponsored health care facility to receive the iDart system. The Tsepong Wellness Centre is currently servicing over 6000 HIV+ patients.

What is a data mart?: It is a simple form of a data warehouse that is focused on a single functional area. Data marts represent the retail level of the data warehouse, where data is accessed directly by end users.[3] The goal of this project is to combine data from multiple instances of iDart into a single data mart that can be used for reporting and analysis by Cell-life (see Figure 1).

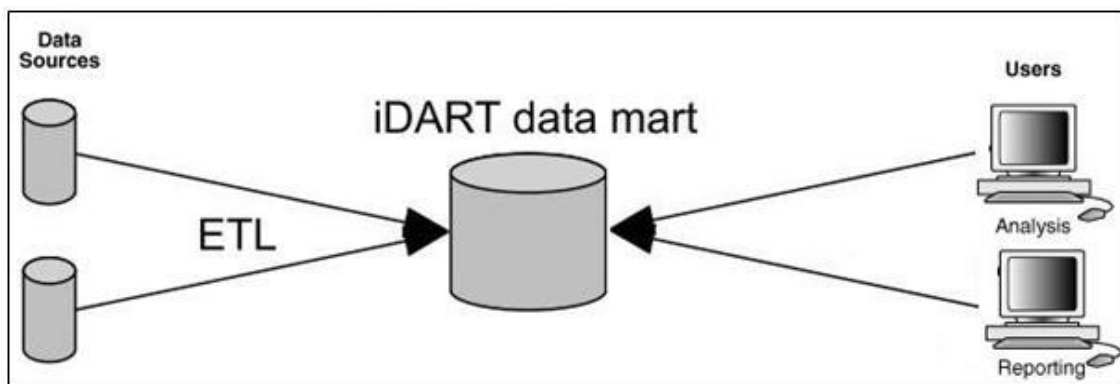


Figure 1: iDart data mart concept.

Concerning the data mart design, two commonly used schemas are the star and snowflake schema. In star schema the fact is denormalised, all dimension tables are normalise and there will be primary foreignkey relationship between fact and dimension tables. For better performance we use star schema when compare to snow flake schema where fact table and dimension tables are normalised every dimension table there will be a look table meaning that we have to dig from top to bottom in the snowflake schema. The main advantages in star schema: 1) it supports drilling and drill down options, 2) fewer tables, and 3) less database.

This document is intended to guide development of iDart data mart. It also will give overview of the project, including why it was conceived, what it will do when complete. Screenshots showing how the final product will look like and behave are provided.

The object oriented view of the system is presented, analysis of the high level design and describes the objects needed to implement the system is provided.

This document also presents the object oriented design of the system, analysis of the low level design and provides details for the object oriented analysis of the system.

The rest of this document is organized as follows. Chapter 2 specifies the user requirements, Chapter 3 provides the user requirement analysis, Chapter 4 provides the user interface specification, Chapter 5 specifies the high level design and Chapter 6 the low level design.

USER REQUIREMENTS

This chapter contains the user requirements of iDart data mart. These requirements have been derived from Cell-life's project specification. This chapter is intended to guide development of iDart data mart. This also will give overview of the project, including why it was conceived, what it will do when complete, and the types of people we expect will use it. Section 2.1 identifies the user's view of the problem, section 2.2 tells what is expected from the software solution, section 2.3 tells what is not expected from the software solution and section 2.4 identifies general constraints for this data mart design.

2.1 User's view of the problem

The time and expense involved in retrieving answers from databases means that a lot of business intelligence information often goes unused. Some organizations use a dozen different software packages to produce simple reports. Also, if the report doesn't have the proper information, its creators have to start over. Also, the cost of implementing a full Data warehouse is higher than that of implementing a data mart. The iDart data mart will help minimize cost of extracting business intelligence information from iDart instances around the country.

2.2 What is expected from a software solution?

The software system is expected to possess easy access to frequently needed data and creates collective view by a group of users.

Cell-Life expects a software solution that can be used for analysis and reporting purposes.

Cell-life would like to be able to generate the following statistics on a monthly/annual basis.

- Number of patients treated(based on packages created)
- Number of patients enroll on treatment
- Number of patients terminating treatment(including reason for termination)

by date, site, gender and age groups (see Appendix A).

2.3 What is not expected from a software solution?

The software solution is not expected to be deployed to all the Cell-Life branches and it is not expected to be able to function in times of power failure unless a backup power supply is in place.

Also the software solution is not expected to be used by multiple business units except what it's designed for.

2.4 General Constraints

We will work under a few number of constraints such as development environment which in this case has to be the integrated development environment (IDE). Also the database we will have to use is PostgreSQL, to make sure that our product (iDart data mart) is compatible with existing database which is currently in use.

REQUIREMENT ANALYSIS

Requirements analysis is critical to the success of a development project. [2] Requirements must be documented, actionable, measurable, testable, related to identified business needs, and defined to a level of detail sufficient for system design. Requirements can be functional and non-functional. Section 3.1 identifies the designer's interpretation of the user's requirements, section 3.2 describes suggested the software solution and section 3.3 identifies types of testing strategies to be used when testing the suggested software solution.

3.1 Designer's interpretation of the user's requirements

Cell-Life has clearly expressed the requirements for the iDart data mart in the previous chapter (Chapter 1). Now we will focus on the business and technical requirements needed to implement the given user requirements. Existing solutions will also be considered.

A basic desktop computer running Windows/Linux will work perfect and a PostgreSQL Database Management System with Java. For data integration, ETL (Extract, Transform, Load) tool (Talend) will be used. Pentaho server and Pentaho Dashboard Designer will form part of the system.

The basic building block I'll use in data mart design is the star schema. A star schema consist of one large central table called *fact table*, and a number of smaller tables called *dimension tables* which radiate out from the *fact table*.

After classifying data from the requirements in Chapter 1, I have the following:

- Date, location/site and patient are dimensions
- Number of patient treated, enrolled for treatment, terminating treatment are facts.

3.2 Suggested solution

The suggested solution will make use of a desktop personal computer (PC) running Windows/Linux and can be broken down into various parts. The first stage uses ETL (Extract, Transform, Load) tool Talend Open Studio to retrieve data from stand alone iDart databases to the iDart data mart. Second stage is accessing data in the data mart, analyzing it, creating reports, graphs, and charts using a Pentaho dashboard.

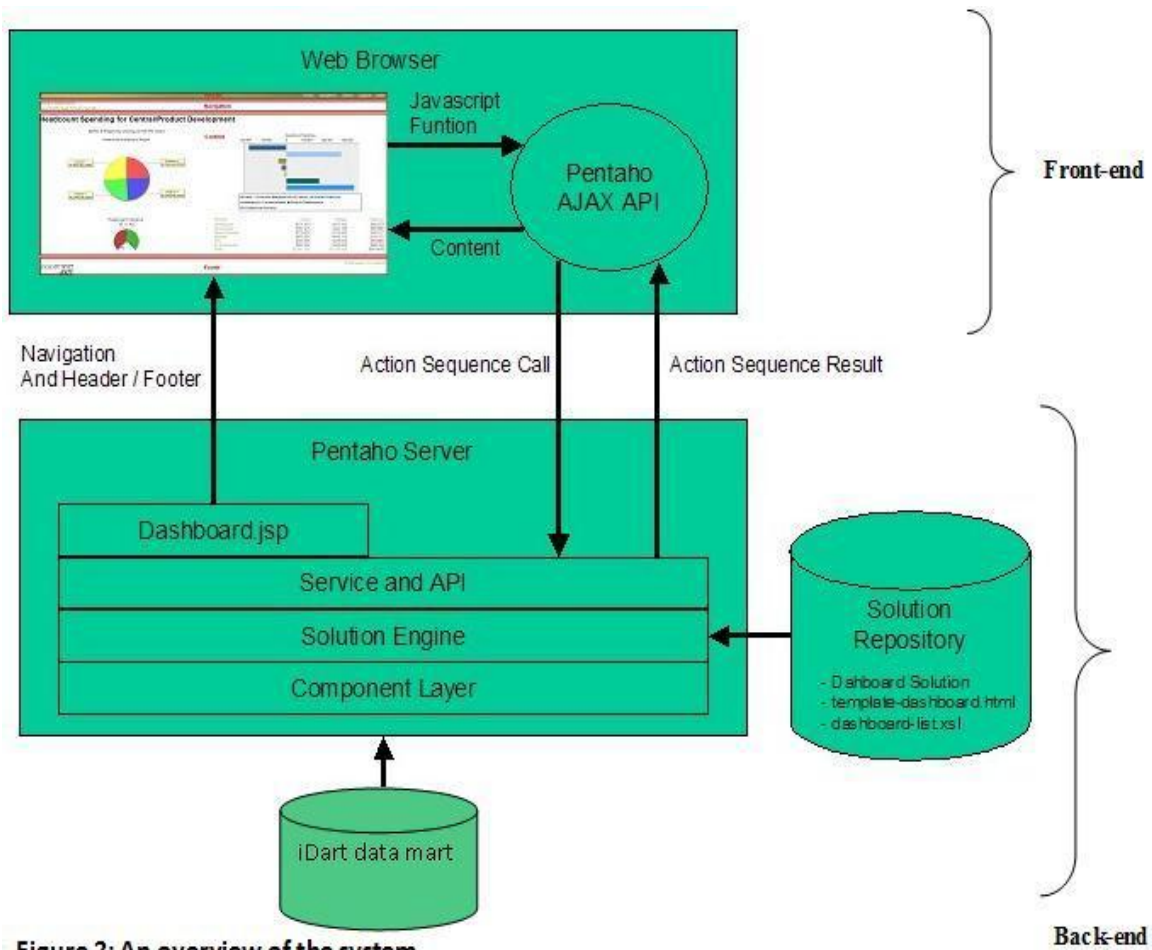


Figure 2: An overview of the system.

3.3 Testing the suggested solution

There are many different approaches to test software. For this project, functional and usability testing will be performed.

1. Functional Testing:

This is a new system and critical, so I must ensure its functional quality. All the features will be tested to ensure all functions provide the expected output.

2. Usability Testing:

Usability testing of this system will evaluate the potential for errors and difficulties involved in using the system for Cell-Life related activities.

Chapter 4

USER INTERFACE SPECIFICATION

The purpose of this document is to provide a detailed specification of the iDart Data Mart user interface. These requirements will detail the outwardly observable behavior of the program. The user interface provides the means for the user, to interact with the program. This User Interface Specification is intended to convey the general idea for the user interface design and the operational concept for the software. Many details have been omitted for both clarity and because they have not been addressed yet. This document will be updated with additional detail as our analysis and design activities progress.

Section 4.1 gives a description of the complete user interface, section 4.2 shows what the user interface looks like to the user, section 4.3 tells how the interface behaves and section 4.4 tells how the user interacts with the system.

4.1 Description of the complete user interface

The User Interface Specification (UIS) consists of one main graphical user interface (GUI), which consists with different operations enlisted in the options.

4.2 What the user interface looks like to the user

The Login page consists of two text boxes, namely Username and Password, and a Login command button allowing the users to log into the system. The login page helps the users to login as a user who visualizes and analyze data contained in the database, and as an Administrator (someone from the IT department) whose duty is to update, edit and modify the dashboard.

Once logged on, the user is presented with the dashboard.

Figure 3 shows the complete User Interface Specification (UIS). This is what a simple typical dashboard for any organization would look like.

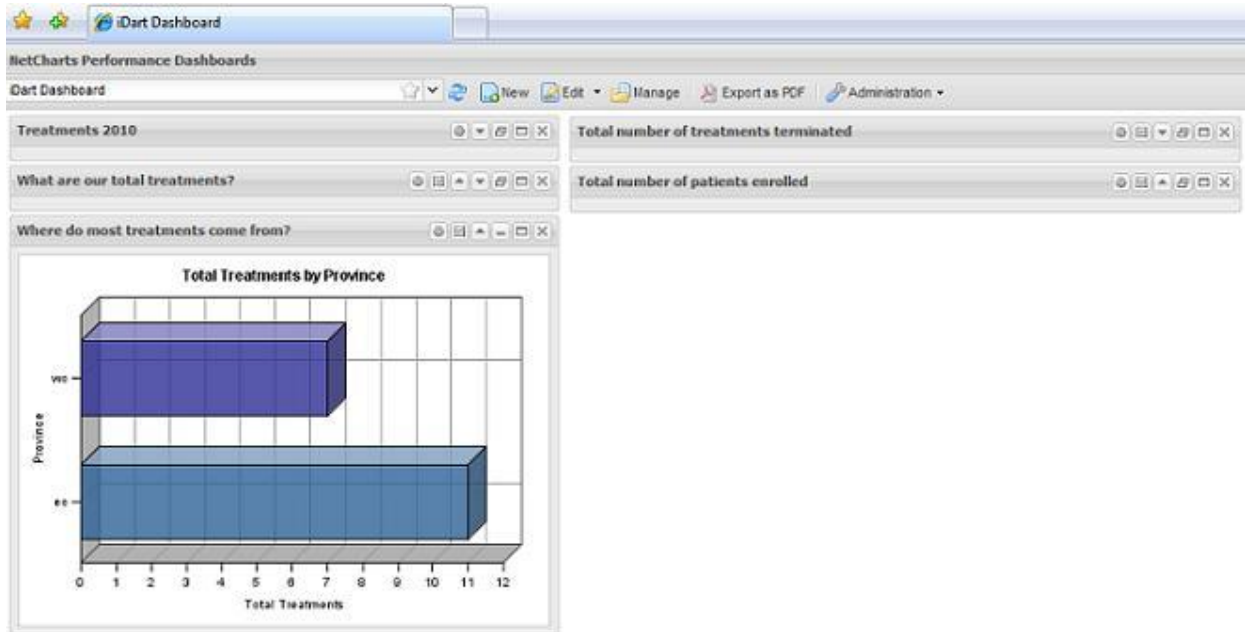


Figure 3: User Interface Specification (UIS).

4.3 How the user interface behaves

How the dashboard interface behaves during manipulation is interesting. Each Key Performance Indicator (KPI) on a page is contained with a portlet featuring up to 7 controls in the upper right corner (Figure 4) used telling the object how to move, resize or do anything else according to a certain user input.




Figure 4: Example of KPI

With these controls, the KPI can be deleted from the page, enlarged, repositioned over the one above it and so on. Such behavior provides the user will full control of how data represented appears in the dashboard.

4.4 How the user interacts with the system

A dashboard report is an important tool for any C-level executive and other business manager. While keeping them on top of vital statistics and Key Process Identifies (KPIs), dashboard reports help them visualize and track trends on every level of the business and to align activities with key goals. The user interface enables users to visualize and analyze data stored in the data mart database. The interface will enable users to choose what data they want to view (*measures*) and how they want to view it (*dimensions*). Figure 5: illustrates how this is achieved:

Consider a scenario where a user wants to see the total number of patients treated province/site name. By clicking the Open Preference menu  icon on the **Where do most treatments come from?** KPI, third in the left column Figure 4 will be shown.

Where do most treatments come from?

Measure: Total Treatments

Dimension: Province

Site Name: Province | All

Province: equals | All

Show: Top 10

Group Remaining Rows Enter Label For Remaining Rows

Visualization: Barchart (4)

Figure 5:

If the user chooses to view number of treatments by province the output would be as shown in Figure 6.

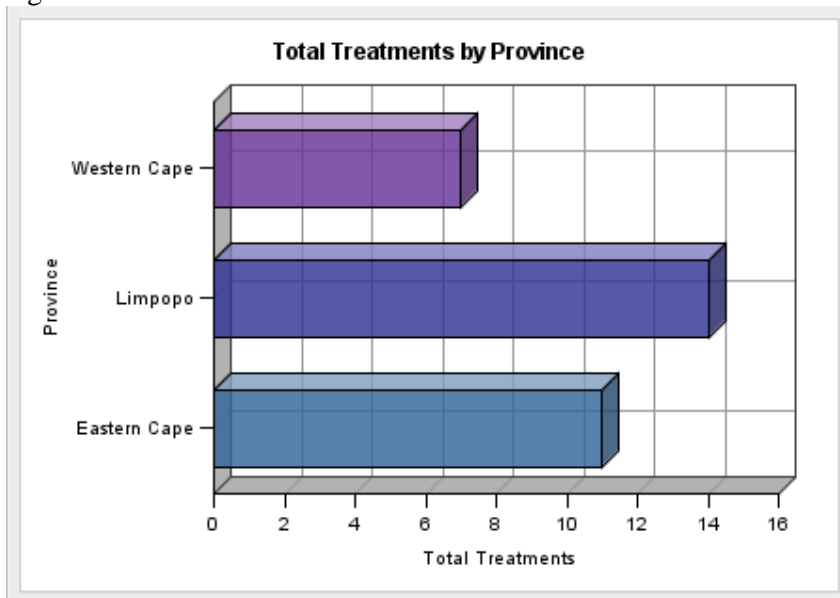


Figure 6:KPI example 1.

Where as if the user chooses to view number of treatments by site name the output would be as shown in Figure 7.

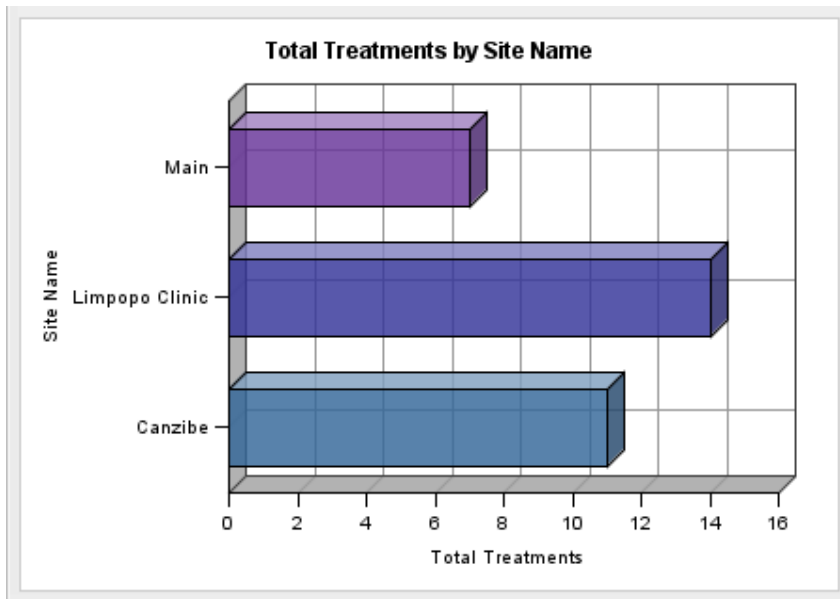


Figure 7: KPI example 2.

HIGH LEVEL DESIGN

This chapter presents the object oriented view of the system, analysis of the high level design and describes the objects needed to implement the system. Each one of these objects is described and documented, and a data dictionary providing details of each object is provided.

5.1 Components

Component name	Component description
Talend Open Studio	Talend Open Studio is an open source data integration product designed to combine, convert and update data in various locations across a business.
Pentaho BI Server	The BI Server is an enterprise-class Business Intelligence (BI) platform that supports Pentaho's end-user reporting, analysis, and dashboard capabilities.
Pentaho Dashboard Designer	Pentaho Dashboard Designer is within the Pentaho User Console. Self-service dashboard designer that lets business users easily create personalized dashboards with little to zero training

Table 1: Objects required.

5.2 User interface design (Use Case Diagram)

Optimized User Interface Design requires a systematic approach to the design process. The importance of good User Interface Design can be the difference between system acceptance and rejection in the marketplace. If end-users feel it is not easy to learn, not easy to use, an otherwise excellent product could fail. Good User Interface Design can make a product easy to understand and use, which results in greater user acceptance. The use case diagram below shows some functional activities of the system that a user can perform.

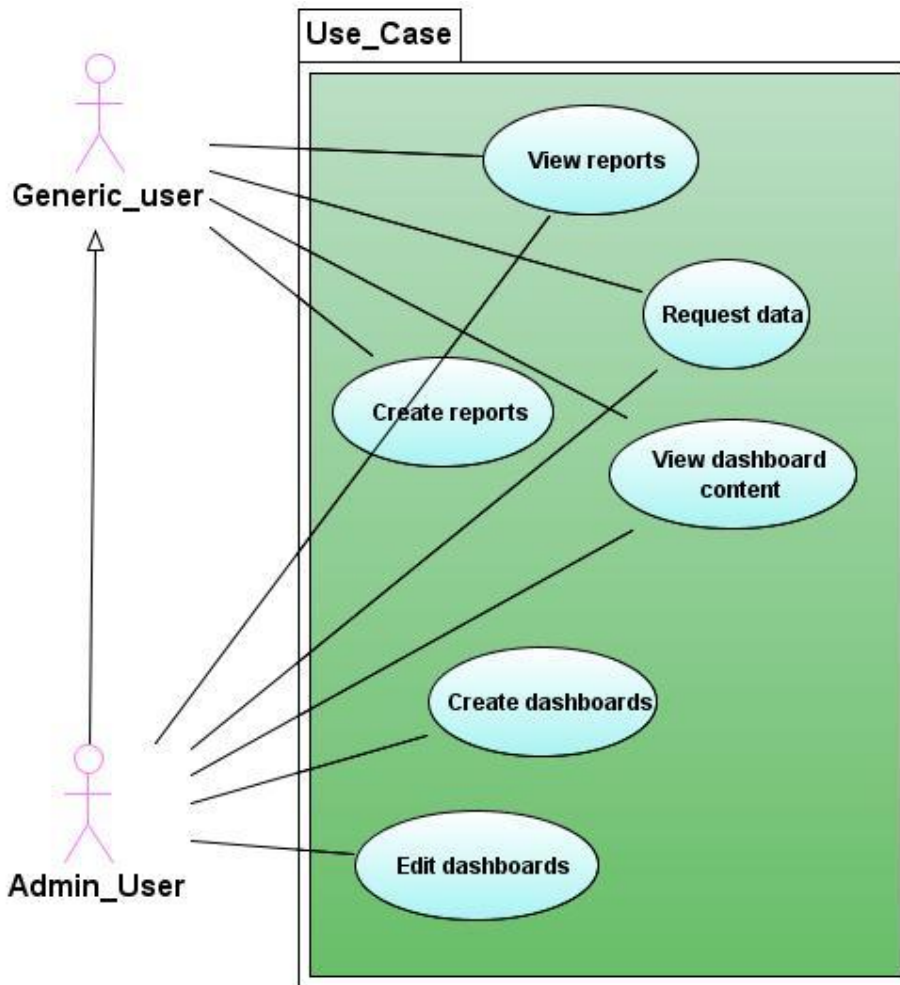


Figure 6: Use case diagram.

The above use case diagram illustrates that a generic user requests data from the data mart by dimension, creates and view reports and can view dashboards and that an administrator has its own behavior but also have the behavior of the generic user. The benefits of generalization eliminates duplicate behavior and attributes that will ultimately make the system more understandable and flexible.

5.3 Use case index

Use Case	Use Case Name	Primary Actor	Scope	Complexity
1	Request Data	Generic User	In	Low
2	Get Status	Generic User	In	Low
3	Create Reports	Generic User	In	Mid
4	Edit Dashboard	Administrator	In	High
5	Edit Dashboard Content	Administrator	In	High

Table 2: Use case index table.

5.4 Class Diagram

Structure diagrams are useful throughout the software lifecycle. Here we've used class diagrams to design and document the system's soon-to-be-coded classes. The purpose of the class diagrams is to show the types being modeled within the system. These types include:

- a class
- an interface
- a data type
- a component

Due to the nature of this project, we have a few number of classes and the reason for this is the fact that java script is mainly used. Figure 7 shows a more detailed description of the class diagrams.

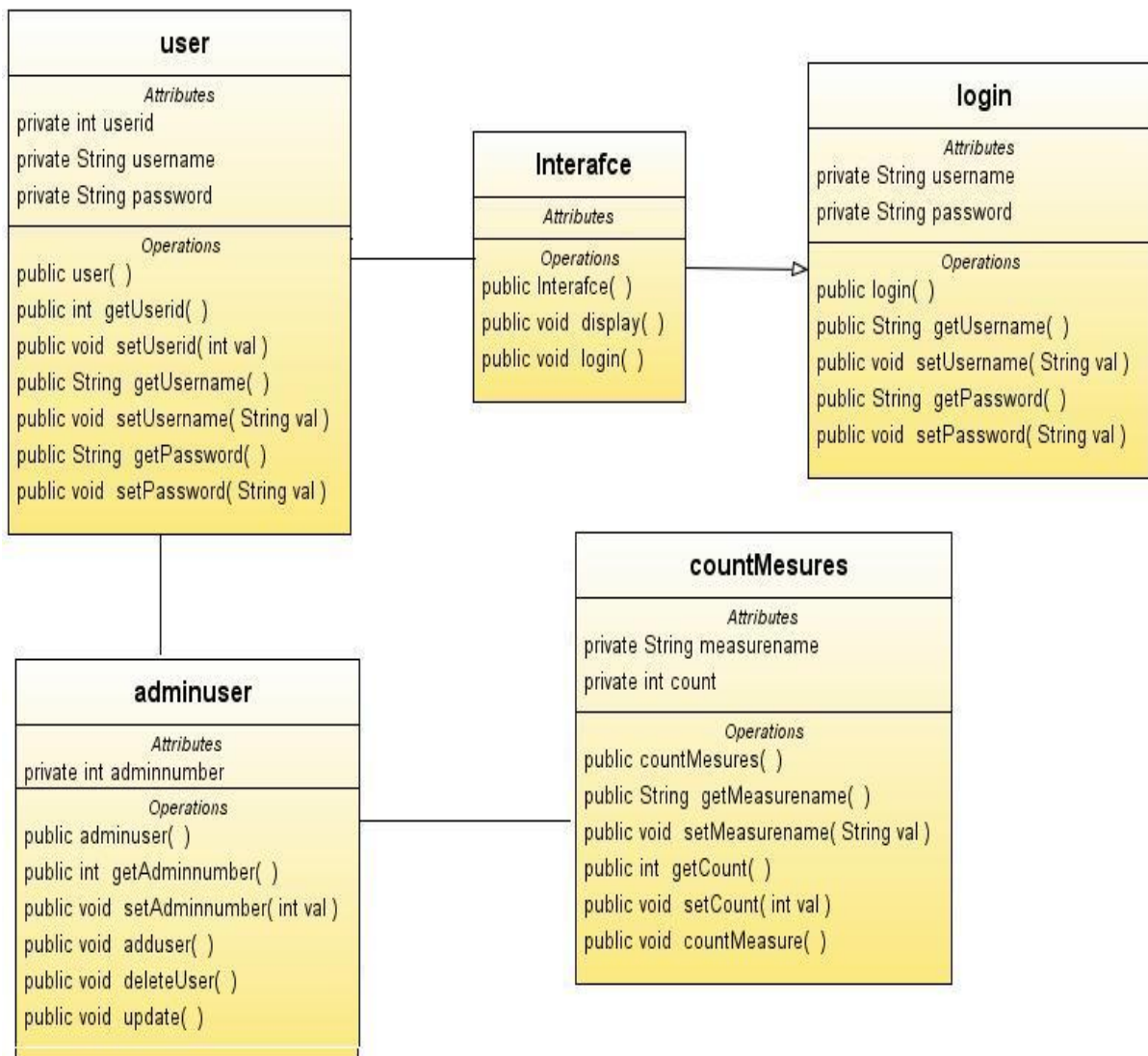


Figure 7. Class diagrams.

5.5 Data mart schema

The figure below (Figure 8) shows the data mart schema for the proposed system. It consists of the three dimension tables and one fact table. The three dimension tables are PatientDimension, SiteDimension and TimeDimension. Each of these tables contains a number of fields and a description of data types.

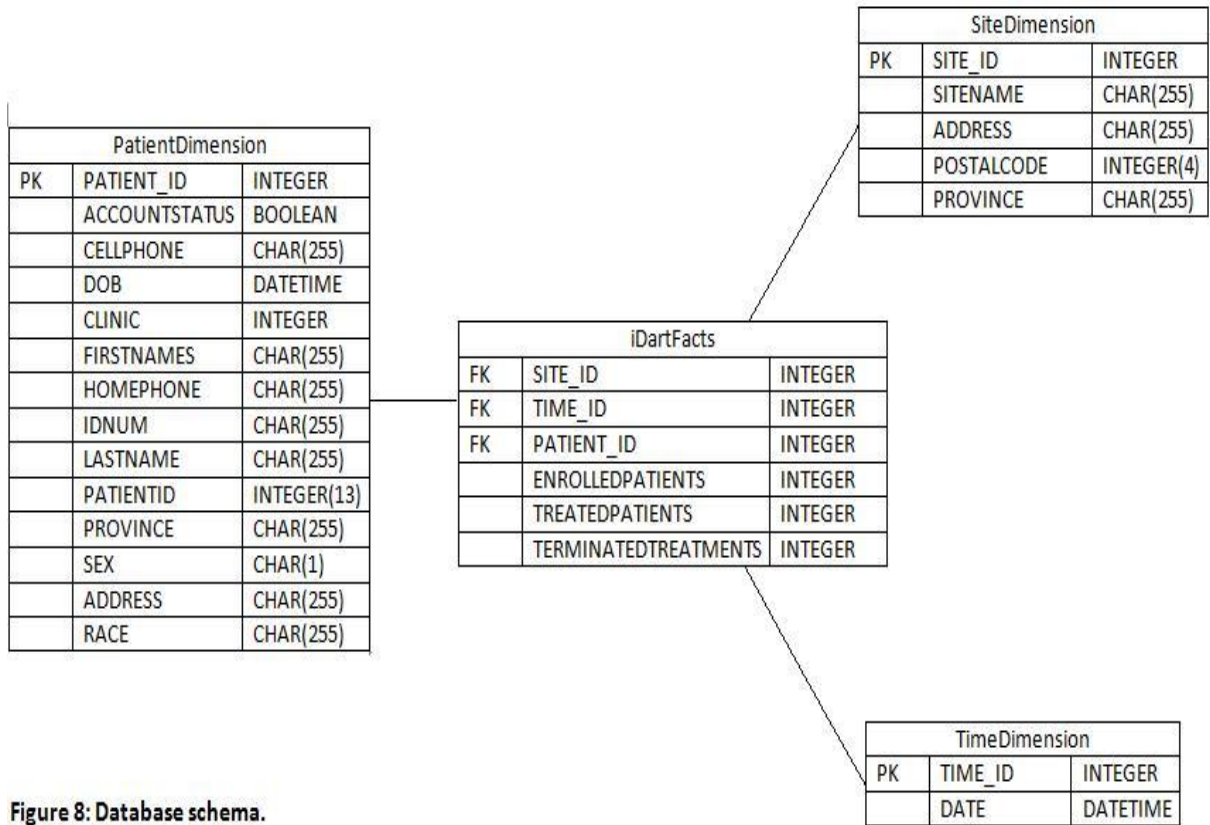


Figure 8: Database schema.

Chapter 6

LOW LEVEL DESIGN

This chapter presents the object oriented design of the system, analysis of the low level design and provides details for the object oriented analysis of the system.

6.1 Details of class attributes

Class	Attributes
User	Int Userid - uniquely identifies the user String Username - stores the username of the user String password - stores the user password
adminuser	Int adminnumber - uniquely identifies the admin user
countMeasures	String measurename - stores the name of the measure. Int count - stores the number of measures
login	String Username - stores the username of the user String password - stores the user password

Table 3. A description of attributes of each class.

6.2 Details of class methods

Class	Function
User	Public int setUserid() - sets the userid Public void setUsername() - sets the username of the user. Public void setPassword() - sets the user password Public int getUserid() - returns the user id when invoked. Public int getUsername() - returns the user name when invoked. Public int getPassword() - returns the user password when invoked.

adminuser	<p>Public int setAdminnumber()- sets the admin user number</p> <p>Public int getAdminnumber ()- returns the admin user number when invoked.</p> <p>Public void adduser ()- adds a new user when invoked.</p> <p>Public deleteUser ()- deletes a specified user when invoked.</p>
countMeasure	<p>Public void setMesurement()- sets the measure name.</p> <p>Public void setCount()- sets the count</p> <p>Public int getMesurement ()- returns the measure name.</p> <p>Public int getCount()- returns the measure count</p> <p>Public void countMeasure ()- returns the actual value of the specified when invoked.</p>
login	<p>Public void setUsername()- sets the username of the user.</p> <p>Public void setPassword()- sets the user password</p> <p>Public int getUsername()- returns the user name when invoked.</p> <p>Public int getPassword()- returns the user</p>

Table 5. A description of methods/functions of each class.

6.3 Pseudo code

```

public class User {

    public int Userid;
    public String password;
    public String password;

    /**
     * Constructor for User.
     * @param Userid int
     * @param Username String
     * @param password String
     */

    /**
     * Method getUserid.

```



```

    * @return int
    */
    public int getUserId() {
        return Userid;
    }

    /**
     * Method setUserid.
     * @param Userid int
     */
    public void setUserid(String Userid) {
        this.Userid = Userid;
    }

    /**
     * Method getUsername.
     * @return String
     */
    public String getUsername() {
        return Username;
    }

    /**
     * Method setUsername.
     * @param Username String
     */
    public void setUsername (String Username) {
        this. Username = Username;
    }

    /**
     * Method getPassword.
     * @return String
     */
    public String getPassword() {
        return Password;
    }

    /**
     * Method setPassword.
     * @param Password String
     */
    public void setPassword (String password) {
        this.password = password;
    }
}

public class admin_user extends User {

    public int adminnumber;

    /**
     * Constructor for admin_user.
     * @param adminnumber int
     */

```

```

/**
 * Method getAdminnumber.
 * @return int
 */
public int getAdminnumber () {
    return adminnumber;
}

/**
 * Method setUserid.
 * @param adminnumber int
 */
public void setAdminnumber (String adminnumber) {
    this.adminnumber = adminnumber;
}

public void adduser()          // delete & update are similar
{
    String name, password;
    Int userid;
    Connection      db;          // A connection to the database
    Statement       sql;        // Our statement to run queries with
    DatabaseMetaData dbmd;      // This is basically info the driver delivers
                                // about the DB it just connected to.

    Class.forName("org.postgresql.Driver"); //load the driver
    db = DriverManager.getConnection("jdbc:postgresql:"+database,
                                    username,
                                    password); //connect to the db
    dbmd = db.getMetaData(); //get MetaData to confirm connection
    System.out.println("Connection to "+dbmd.getDatabaseProductName()+" "+
                      dbmd.getDatabaseProductVersion()+" successful.\n");
    sql = db.createStatement(); //create a statement that we can use later

    String sqlText = "insert into usertable values (name,userid,password
etc)";

    sql.executeUpdate(sqlText);

    . . . . . //some exception handling code for invalid password, etc.
}

}

Public class countMeasure{

import java.sql.*; // Everything we need for JDBC
import java.text.*;
import java.io.*;

public void countMeasure()
{
    Int measure;
    Connection      db;          // A connection to the database
    Statement       sql;        // Our statement to run queries with
    DatabaseMetaData dbmd;      // This is basically info the driver delivers

```

```

// about the DB it just connected to.

Class.forName("org.postgresql.Driver"); //load the driver
db = DriverManager.getConnection("jdbc:postgresql:"+database,
                                username,
                                password); //connect to the db
dbmd = db.getMetaData(); //get MetaData to confirm connection
System.out.println("Connection to "+dbmd.getDatabaseProductName()+" "+
                  dbmd.getDatabaseProductVersion()+" successful.\n");
sql = db.createStatement(); //create a statement that we can use later

// Here will be a code that will actually count each of the measures
// This is tricky since on our data sources these measures aren't
// Counted.
String sqlText = "";

sql.executeUpdate(sqlText);
measure = sql.getUpdateCount();
}

```

Bibliography

[1] BATIN, C., SERI, S., AND NAVATE, S.B, (1994) *Conceptual Database Design: An Entity Relational Approach*, Redwood City, California

[2] Executive editors: Alain Abran, James W. Moore; editors Pierre Bourque, Robert Dupuis, ed (March 2005).

[3] InfoManagement Direct, November 1999. *Data Mart Does Not Equal Data Warehouse* [online]. Available <http://www.information-management.com/infodirect/19991120/1675-1.html> [accessed 7 March 2010]

[4] KIMBALL, R.,(1996): *The Data Warehouse Toolkit*, New York: J. Wiley & Sons.

[5] KIMBALL, R.,(1997): *DBMS Online*,A Dimensional Manifesto August, 1997.

APPENDIX A

iDART Consolidated Reporting Requirements

Reporting requirements

Cell-Life would like to be able to generate the following statistics on a monthly / annual basis. Each statistic should be broken down by date, site / location, gender and age groups.

- Number of patients treated (based on packages created)
- Number of patients enrolled on treatment (based on patient episodes)
- Number of patients terminating treatment (including reason for termination e.g. deceased)

Additional statistics are also required (broken down by site, gender and age groups):

- Total number of patients on treatment (based on patient episodes)

APPENDIX B

Project plan Term 1

Dates	28/02/10	07/03/10	14/03/10	21/03/10	22/03/10	23/03/10	24/03/10
Meetings	Meeting with supervisor	Meeting with supervisor		Meeting with supervisor	Meeting with supervisor	Meeting with supervisor	Meeting with supervisor
Thesis document			Writing document				Hand in final Document
UR	Get UR from Cell-Life						
RA		To analyze the UR given by Cell-Life					
Presentation & Deliverable	Download & install iDart			To do slides & send to supervisor	Present on BANG meeting		Presentation
Website			Design website				

Project plan Term 2

Dates	07/04/10	12/04/10	19/04/10	26/04/10	03/05/10	10/05/10	17/05/10	18/05/10
Meetings	Meeting with supervisor	Meeting with supervisor	Meeting with supervisor	Meeting with supervisor	Meeting with supervisor	Meeting with supervisor	Meeting with supervisor	BANG meeting
Thesis document					Writing document		Hand in final Document	
UIS	To Read the contents of UIS	To analyze how the UI will be		To build the interface				
OOA	To Read the contents of OOA		Drawing the diagrams					
OOD	To Read the contents of OOD		Download Eclipse to code in Java					
Prototype		Consult the websites about the codes	Trying to code		Test it			Present on BANG meeting
Deliverable							To do slides & send to supervisor	