**Prototype of a Radiation Dosage Tracking System**

Vaishnavi Konda

Department of Computer Science

University of Kentucky

Lexington, KY 40506

Advisory Committee

Dr. Zongming Fei

Dr. Jie Zhang

Dr.Tingting Yu

22nd May 2015

**Acknowledgement**

This project would not have been possible without the support of many people. I would like to express my gratitude to my supervisor Dr.Zongming Fei who was very helpful and offered invaluable assistance, support, and guidance and for the engagement through the learning process of this master’s project. Deepest gratitude are also due to the members of the supervisory committee, Dr. Jie Zhang without whose knowledge and assistance this project would not have been successful. Special thanks to Dr.Tingting Yu for being my committee member. I would also like to express my gratitude to all professors who have helped me throughout my MS program. Special thanks also to all my graduate friends for sharing the literature and invaluable assistance. Not forgetting my best friends who always been there. I would like to express my love and gratitude to my beloved families; for their understanding & endless love, through the duration of my studies.

**Table of Contents**

1 Introduction

1.1 Background ………………………………………………………………………….. 3

1.2 Problem Statement …………………………………………………………………... 3

1.3 Project Scope ………………………………………………………………………… 4

2 Requirements Definition

2.1 Objectives …………………………………………………………………………… 4

2.2 Critical Success Factors ……………………………………………………………... 4

2.2 Requirements ………………………………………………………………………... 4

3 Conceptual Design

3.1 Conceptual Data Model ……………………………………………………………… 7

3.1.1 System Context Diagram ……………………………………………………… 7

3.1.2. Entity-Relationship Diagram …………………………………………………. 8

3.2 System Architecture …………………………………………………………………. 9

3.3 Contingency Planning ……………………………………………………………….. 10

3.4 Product Delivery Plan ……………………………………………………………….. 11

3.4.1 Implementation Plan …………………………………………………………… 11

4 Functional Design

4.1 System Requirements ………………………………………………………………… 12

4.1.1 Hardware Requirements ………………………………………………….......... 12

4.1.2 Software Requirements ………………………………………………………… 13

4. 1.2.1 Operating System ……………………………………………………… 13

4.1.2.2 Additional Software ……………………………………………………. 13

4.2 Database design …………………………………………………………………….... 13

4.3 Test strategy …………………………………………………………………………. 14

4.4 Functional Architecture ……………………………………………………………… 14

4.5 Design

Iteration 1 ……………………………………………………………………….. 14

Iteration 2 ……………………………………………………………………….. 18

Iteration 3 …………………………………………………………………….…. 20

Iteration 4 ……………………………………………………………………….. 22

Iteration 5 ……………………………………………………………………….. 23

Iteration 6 …………………………………………………………………….…. 24

Iteration 7 ……………………………………………………………………….. 27

5 Implementation

5.1 Implementation Challenges ………………………………………………………….. 29

5.2 Implementation ………………………………………………………………………. 29

6 Change Requests ………………………………………………………………………………… 33

7 System Testing ………………………………………………………………………………….. 34

8 Technical Metric Collection ….…………………………………………………………………. 34

9 Learning Outcome ………….…………………………………………………………………… 34

10 Definition of Terms / Acronyms ………….…………………………………………………….. 34

11 References ……………………………………………………………………………………… 35

**Prototype of a Radiation Dosage Tracking System**

**1. Introduction**

**1.1 Background**

Patient radiation dose is a popular and controversial topic, due to both its risks and the increasing medical radiation exposure of the public. As a result, it is important that patient radiation dose is tracked to both prevent overexposure and to provide patients information about the risks associated with radiation-based tests and procedures. The main concern with medical radiation tests is cancer risk - i.e. that even a very small dose could induce a cancer - thus there is a need to track the radiation dose administered to the patients. Hence, a new system is in demand to meet these requirements.

This new system which is a web application simplifies the requirement by providing user interfaces to enter the doctor details and the physician details and thus plots the graphs from which the analysis can be done. This system is developed using PHP and MySQL web technology thereby bringing all the advantages of the relational database system and the PHP technology together to build a web-based database management system. This report walks the reader through the development of the basic relational database system developed using MySQL and PHP. The entire development life-cycle, including requirement analysis, design, implementation, and testing, is discussed.

* 1. **Problem Statement**

There is a need for a software which can be used to do analysis on the dosage pattern of the physician by simply entering the physician and patient details. This proposed system will perform the following functions like storing physician and patient information (Ex: first and last names, operation details, height, ID’s) and admin details (Ex: username, password, email)**,** getting the physician details, the patient details , plotting the graph by selecting physician and patients, getting the average of dosage. This new software will give graphs from which the analysis on the dosage patterns of the physicians can be performed.

**1.3 Project Scope**

This web application will be used to store patient and physician information. It will be used to add, update, delete their records, find the average of their dosages, retrieve their details, and plot the graph

This application ***does*** the following:

* Stores patient and physician records
* Retrieves appropriate physicians records and patients records
* Generates a list of patients and their operation details for selected physicians
* Generates a list of physicians and their details for selected operations
* Gives the average of reference dose for selected physicians
* Gives the average of reference dose for selected operations
* Generates a graph(bar chart) for selected physicians and operations in a particular range of dates
* Allows the admin to add, delete, update usernames and passwords for login purpose

This application ***does not*** *do* the following:

* Generating a report from the graph(Analysis is done manually)
* Validating information beyond the scope of the database like patients’ height and weight

**2. Requirements Definition**

**2.1 Objectives**

Provide all business requirements for the radiation dosage tracking system to ensure that the needs of the business are satisfied.

**2.2 Critical Success Factors**

The business requirements captured in this document must represent all the needs of the business in order to guarantee successful implementation of the radiation dosage tracking system at launch.

**2.3 Requirements**

The original business requirements specification for the system is listed below:

* The database should keep track of physicians and patients information.
* Admin should be able to keep track of login details
* Average of dosage for particular operations should be calculated
* Average of dosage for selected physicians should be calculated
* For selected operations and physicians in a given range of dates, graph (bar chart) needs to be plotted

After a series of interviews and review sessions, a detailed list of business requirements was captured (discussed in the following table). The nature of the project was analyzed and it was determined that the client requirements were generic. This would mean that there would be a lot of additions and changes in requirements as the system is built. Hence, it was agreed upon among the business clients and the analyst/developer that an iterative approach will be followed in order to give the system a chance to evolve and absorb changes with minimum possible risk of project failure.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requirement ID | Functional Area | Function | Sub-Function | Description | Required Values / Event |
| 001 | Login | User Authentication | Login | Accept user name and password. Compare the pair with the stored login information and allow/reject user into the system. | User name, password |
| 002 | Login | User Authentication | Logout | Terminate the user’s access to the system. Notify the user of the successful log off and direct the user how to log back in. | Clicking the “log out” button |
| 003 | Login | User Authentication | Logout | After the logout button is clicked, do not allow user to enter the application by clicking button “Back” or by book marking / manually typing a direct link to one of the pages of the application. | N/A |
| 004 | Login | User Authentication | Login | Accept user name and password and if its ‘admin’,redirect to admin page and if it’s not ‘admin’, redirect to other pages | Default view after logging in |
| 005 | Login | User Authentication | Login | During login , if they are privileged users ,they are redirected to page in which they can modify the database and if they are not, they will be redirected to another page  In which they cannot modify the database | Privilege field |
| 006 | User Details Maintenance | Users Maintenance | Users information | After the user logins in with admin privilege, they can insert, update or delete the users records | Clicking the “insert”,”edit”,”delete “buttons |
| 007 | Physician Details Maintenance | Users Maintenance | Physician information | After privileged users logins, they can insert the physicians records into the database  Physician ID number should be auto-generated | Click on the “insert into physicians table” link |
| 008 | Patient Details Maintenance | Users Maintenance | Patient information | After privileged users logins, it allows the user to enter the new patient’s information  Patient ID number should be auto-generated | Click on the “insert into patients table” link |
| 009 | Physicians Details Query | Retrieve physicians Information | Get all physicians information | Allow the user to get the all physician’s information  Operation Name and corresponding patient’s details should be automatically retrieved and displayed upon selecting their physician IDs.   The information entered needs to be routed to the corresponding tables namely patients Table, physicians Table and Operations table. | *Required event:* Clicking the “physician details “ link  *Required fields:* First Name Last Name Physician ID Patient ID ExperienceOperation nameDAPFlorence timeReference doseOperation date |
| 010 | Patient Details Query | Retrieve patient Information | Get all patients information | Allow the user to get the all patient’s information  Operation Name and corresponding physician details should be automatically retrieved and displayed upon selecting their patients IDs.   The information entered needs to be routed to the corresponding tables namely patients Table, physicians Table and Operations table. | *Required event:* Clicking the “patient details “ link  *Required fields:* First Name Last Name Patient ID HeightWeightOperation nameDAPFlorence timeReference doseOperation date |
| 011 | Physician’s operation Details Query | Retrieve operation Information that a physician has performed | Get operation details performed by a physician in that selected range of dates | *user can select a particular physician from the dropdown menu and he can select range of dates and can get all the operation details performed by that selected physician in those selected dates* | Clicking the ‘’ submit” button in pno.php page |
| 012 | Operation details | View operation and doctor Information | Operation information | User can select an operation from radio buttons and it displays the operation details and the doctors who performed it and his experience | Clicking the “operation details” link in the ope.php |
| 013 | Operation details | View the average of reference dosage | Average of reference dosage for operations | The average reference dosage for the selected operations in the checkboxes is calculated and is displayed and also the doctor who performed these operations and their individual reference dosage are retrieved and displayed in the form of a table | Selecting the operation names from the list of all operations in “avgop.html” page and clicking on “average “button |
| 014 | Physician details | View the average of reference dosage | Average of reference dosage for doctors | The average reference dosage for the selected physicians in the checkboxes is calculated and is displayed and also the operations performed these physicians and the reference dosage of these operations are retrieved an displayed in the form of a table | Selecting the physicians names from the list of all operations in “avgphy.html” page and clicking on “average “button |
| 015 | Physician and operation details | Graphical representation | Generate a graph | Allows the user to select the operations and physicians and the range of dates and this will be redirecting to main.php and then this will be redirecting to line.php which will display the graph | Clicking the “Submit” button in time.html will display table with all info and then clicking on “graph” button will give you graph |
| 016 | Admin details | Password security | Make a hash of password | The password should not be stored as it is in the database because it may became vulnerable to hacking and attacks and so password is hashed using “md5” hashing and then stored in the database | Having md5(”password”) in admindata.php |

**3. Conceptual Design**

In this section, an attempt has been made to create a model of the dosage tracking system that represents the real-world objects in the most realistic way possible, in software- as well as hardware independent manner. High-level diagrams capturing the system’s context, data model and technology architecture are illustrated in order to provide a big picture of the system.

**3.1 Conceptual Data Model**

**3.1.1 System Context Diagram**

Figure 3.1 shows the high-level system context diagram of the scope of the prototype of radiation dosage system that shows the system boundaries, external entities that interact with the system and the major information flows between the entities and the system.

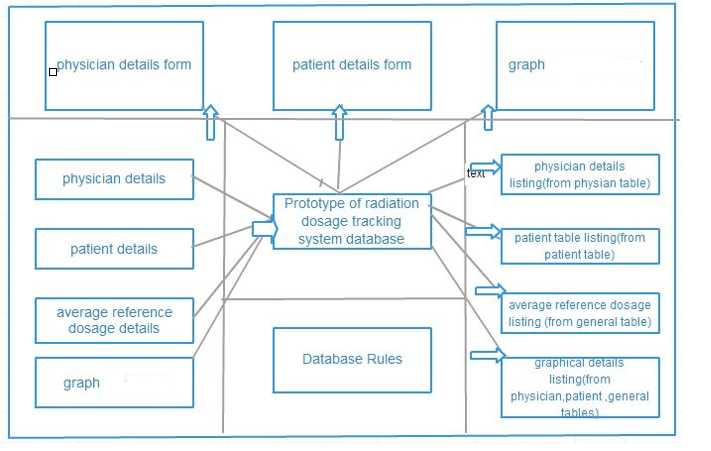


Figure 3.1- System Context Diagram

**3.1.2 ER Diagram**

Figure 3.2 shows a very high-level skeletal representation of the structure of each table involved in building this project, and the links between these tables. The purpose of this ER illustration is to only clearly identify and relate the entities only. Hence additional details like type of the entities, type of the relationships and cardinality have not been shown. For easy readability, attributes for each of these entities have also been omitted.

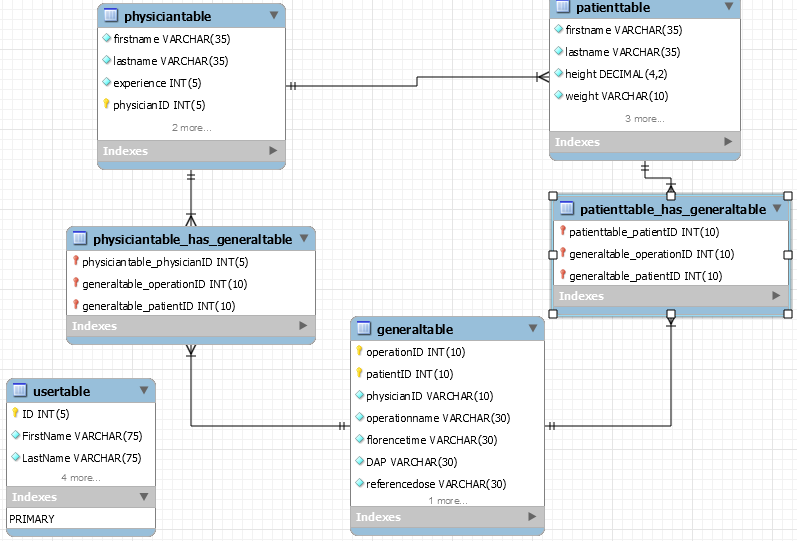


Figure 3.2 -ER Diagram

**3.2 System Architecture**

An Architecture Diagram is a logical diagram that shows how each of the components in a system is connected with each other and how are the data flowing between. The architecture diagram for our PHP based web application will be like the below diagram.

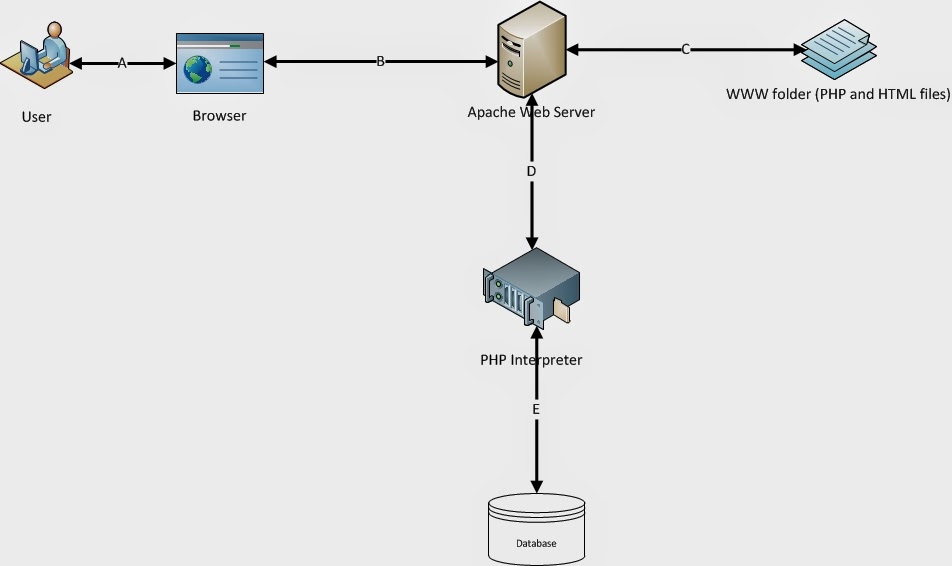


Figure 3.3 – System Architecture

**1**: First the user accesses the website through the browser. That means user types the URL of the website in browser and hit go.  
**2**: The page request on browser will reach to the Web Server (Apache).  
**3**: Web server will collect that requested page (HTML or PHP or Image file etc) from its document root. (Here it will be www folder in WAMP)   
**4**: Now if it is a static element like HTML , CSS , image file or Java Script file then Apache will send it directly to browser.  
**5**: And Browser will render it to the user on screen  
**6**: If it is a PHP file then Apache sends the content of the file to PHP Interpreter. PHP interpreter interprets the PHP code and executes it.   
**7**: PHP Interpreter generates output (if the PHP code is to generate any output) and sends to Apache  
**8**: Apache sends that content to browser  
**9**: Browser renders it to users' screen

**3.3 Contingency Planning**

A web-based front-end application will be used as the primary system to perform all the query and data modification operations. This will ensure that the application is available for remote operations and/or to multiple users for simultaneous read access. In order to cover for contingencies like network failures, web-server failure, etc., the front-end application will also be constructed using PHP and MySQL.

**3.4 Project Plan**

The following shows the proposed project plan:

Scope and Requirements Definition - 4 weeks

Conceptual Design - 1 week

Functional Design, Implementation, Unit & Integration Testing

Iteration 1 (Patient Details and Physician Details) - 3 weeks

Iteration 2 (Admin Details) - 2 weeks

Iteration 3 (Graph) - 2 weeks

System Test, Deployment, Training - 2 weeks

*Note:*

*Each iteration includes:*

* *Functional design of modules to be implemented*
* *Implementation*
* *Testing (Unit & Integration)*

This schedule is tentative for two different reasons:

* Several changes in requirements are anticipated.
* The developer is a novice to plot the graph.

**3.4.1 Implementation Plan**

Since this project employs incremental development, the requirements enlisted in Section 2.3 are packaged into separate modules. These modules will be designed and implemented sequentially in a predefined order, during each incremental cycle, and released for client testing in the same order. During each iteration, apart from implementation, unit and integration testing need to be performed. The errors identified need to be rectified before starting the next iteration.

The following list provides information about the various modules and the requirements they implement. Each incremental cycle is termed “Iteration x”, x showing the order in which the modules are being implemented.

**Iteration 1:**

In the first cycle, *User Authentication* is implemented

* *User Authentication:* Validates (authenticates) users logging into the portal and controls their access (authorizes) to the resources in the portal. This module covers requirements 001 – 005
* *User Details Maintenance Module:* Displays all information related to users, performs addition, editing and deletion of user information. This module covers requirements 006 and 016

The user authentication module will be implemented in the first iteration simultaneously with the User Details module.

**Iteration 2:**

*Physician and patient details maintenance module:* Privileged users are given access to enter the physician, patient and operation information. This module covers requirements 007 – 008

**Iteration 3:**

*Physician and patient details retrieval module:* Users are given access to get all the physician, patient and operation information in the form of tables. This module covers requirements 009 – 010

**Iteration 4:**

*Physician and operation**details in a given range of time module:* Users are given access to select a physician from a drop down menu and range of dates and this will give that physician details and the operations he /she has performed. This module covers requirements 011

**Iteration 5:**

*Operation details maintenance module:* Users are given access to select the operations and this will give all operation details and the doctors who performed this operation. This module covers requirements 012

**Iteration 6:**

*Average reference dosage module:* Users are given access to select operation names and physician names respectively and this will give average reference dosage for the selected operations and selected physicians respectively. This module covers requirements 013-014

**Iteration 7:**

*Graphical representation module:* Users are given access to select operation names and physician names and the range of dates and this will give a graph depicting the doctors and reference dosage over the selected period of time for the selected operations and selected physicians. This module covers requirement 016

**4. Functional Design**

“*Functional design is a design pattern which is used to simplify the design of computer software and this will assure that each modular part of a computer program has only one responsibility and performs that responsibility with the minimum of side effects on other parts”;* http://www.revolvy.com/main/index.php?s=Computer-aided%20industrial%20design. The functional design of the radiation dosage tracking is described in this section. Since an iterative approach is being adopted to build this application, common functional aspects like the system requirements, test strategies, etc. are described once – common to all the releases. The aspects that vary with each individual release like user interface design, application logic, etc. are discussed separately for each of the iterations.

**4.1 System Requirements**

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements.

**4.1.1 Hardware Requirements**

**Required processor :** Pentium 4 or AMD or Celeron Processor

**Required RAM :** RAM 512 MB or above

**4.1.2 Software Requirements**

**4. 1.2.1 Operating System**

Microsoft® Windows® 7 or higher

**4.1.2.2 Additional Software**

1. WAMP --- Web Application Server

2. PHP --- Server Side Scripting Language

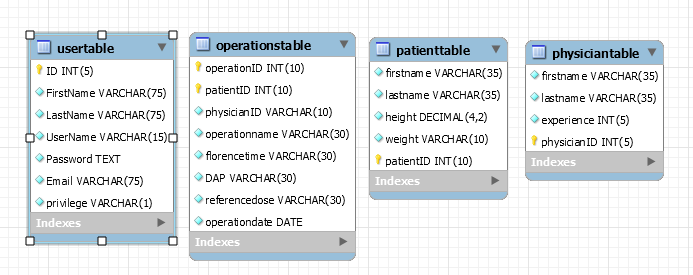
3. MySQL --- Database

4. XHTML --- Extendible Hypertext Markup Language

5. Notepad++ --- As a development tool

**4.2 Database Design**

The back-end of the radiation dosage tracking system will be designed as modeled in the ER diagram (Figure 2.0) using MySQL. Tables, Reports, and Forms will be created and the data that is currently available will be imported into the database. The data that is imported will serve as the test data model for developing the front-end applications.

**4.3 Test Strategy**

The following kinds of testing will be performed during the development of the Radiation Dosage Tracking System:

i) Unit testing (for every iteration)

Each module will be tested alone in an attempt to discover any errors in its code.

ii) Integration testing (after every iteration)

Combined parts of the application will be tested to determine if they function together correctly.

**4.4 Functional Architecture**

The high-level system design involved three layers, namely:

1. *User Interface:* Implements all the web forms that are presented to the user to allow the execution of system functionality. This layer coordinates all user inputs and actions, providing the business logic layer with the necessary data to execute the desired functionality. It is also responsible for formatting and presenting data and information received from the business logic.
2. *Application Logic:* Implements all the functional requirements of the system. This layer performs the required operations to respond to requests from the user layer. It is also responsible for identifying the appropriate data sources to pull data from, modification of data according to the user specifications, and adding new data to the database.
3. *Database:* Provides a storage repository for data, executes the data and schema modification operations requested by the application logic, extracts and returns requested data, organizes data to facilitate efficient access.

**4.5 Design**

In this section, the functional design conceived for each module, which includes user interfaces and application logic, is explained. The design of *user interfaces* includes deciding on the page layouts, controls to be used, formats/colors to be employed and the presentation logic to display the data on screen (GUI processing). The *application logic* describes the part of the prototype of radiation tracking application that performs the required business data processing. The *application logic* refers to the routines that perform the data entry, update, query and report processing, and more specifically to the processing that takes place behind the scenes.

*Note: Sections 4.1 through 4.4 show the functional design activities that will be carried out once - common for all the iterations of the system. The following sections (starting section 4.5) differ for each individual iteration, and hence are described individually for each iteration.*

**Iteration 1:****User Authentication** **Module**

Iteration one has two components:

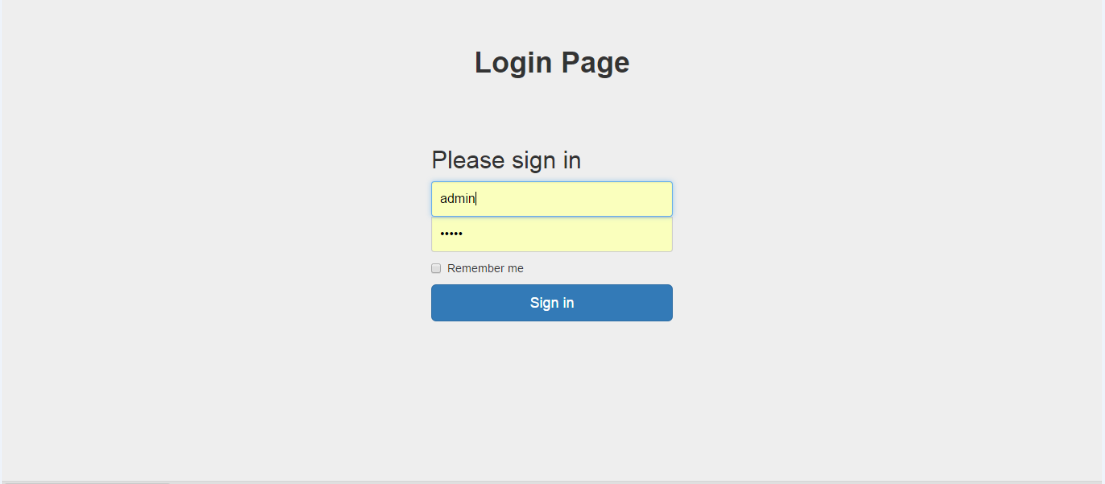
I1.1 – User Authentication Module

I1.2 – User Details Maintenance Module

**I1.1 User Authentication Module**

**I1.1.a - User Interface**

Below is the screen shot of the user interface designed for the Login screen. The user interface has text boxes for entering the user name and password. A checkbox is also placed on the page for the user to specify if the browser should remember the username and password entered by the user after the application browser had been closed and reopened.

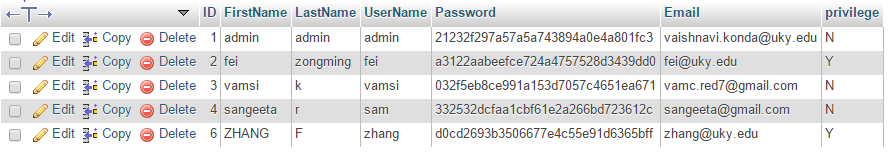


Login Screen

**I1.1.b - Application Logic**

The web application will be secured using forms authentication and authorization mechanisms supported by PHP. All the resources associated with this web application will be secured except for the images to be displayed on the login and logout pages. HTML forms authentication validates a legitimate user in order to allow them access to the application. Authorization controls the resources that the users are entitled to access. This application requires only a secure entry point to access the sensitive data since all the users are supposed to access all the resources in the application.

A list of users and their passwords are stored in “user table” which resides in the database on the web server. All the resources in the application are available for use by the users stored in “user table”.



The login page takes as input, the username and password of the user and validates it against the users listed in the “user table”. Once the user is validated, a cookie is returned to the user authenticating him/her to this application for a specific period of time. Thereafter, each time the user requests a page, the cookie is passed along with the request to indicate that the user is authenticated. If the user does not request for a page within a specified time interval, then the cookie will be invalidated and the user would need to login again.

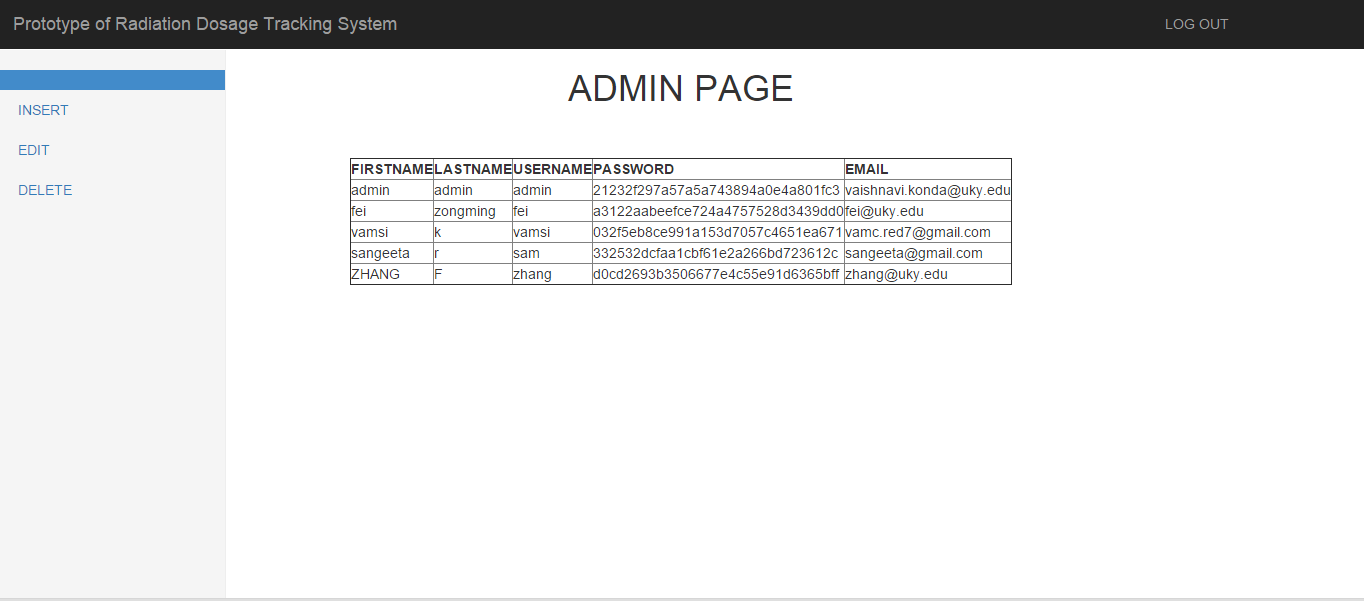
Once the user decides to exit the application, he/she would have to logout of the application, which implies that they manually invalidate their authentication cookie by pressing a “Log Out” button on the requested pages. The “Log Out” button invalidates the user’s cookie and redirects the user to the logout page. If the user does not logout of the application but closes the application browser, then if they had selected the “Remember me next time” option on the login page, the browser will be allowed to retain the cookie.

**I1.2 - User Details Maintenance Module**

The User Details Maintenance module is one of those modules through which we can enter the user details into the database and it require a form-like user interface design. The primary entity in this module is the user. All the information pertinent to the user is to be displayed as a table. The admin can view, edit or delete the information related to the users.

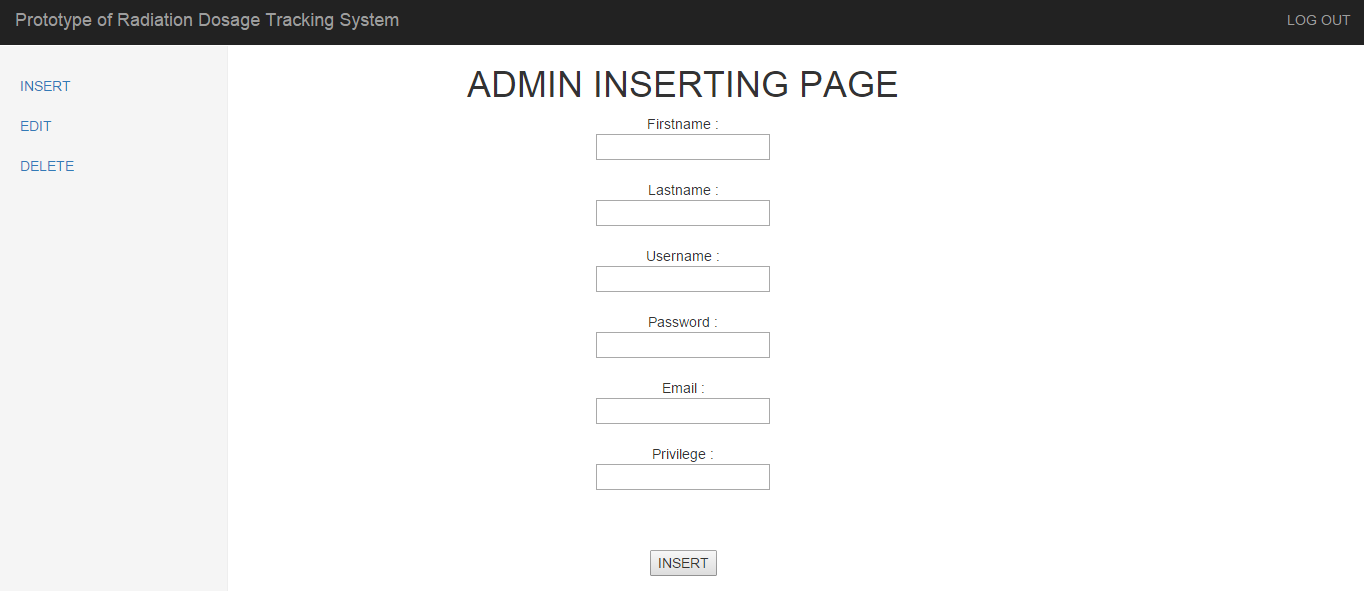
**I1.2.a - User Interface**

The user interface for this module (Figure I1.2) has *three* separate links through which all the form controls can be accessed. After the admin logins into the application, he/she will have the access to links through which they can insert, delete or edit the user information.Admin will have three links “Insert”, ”Edit”, ”Delete”



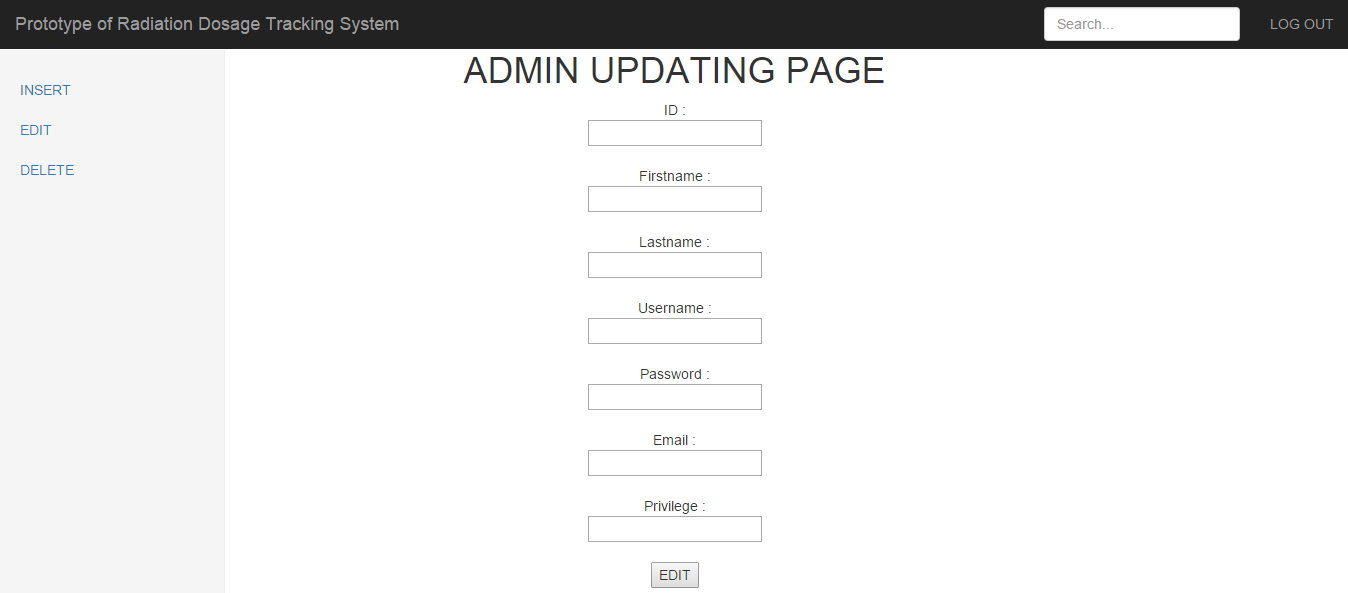
*“****On******Insert”*** contains the following controls:

* All the data fields managed by textboxes
* *‘Firstname’ / ‘Lastname’ / ‘Username’ / ‘password’ / ‘Email’ / ‘Privilege’* fields to enter the user information
* *‘INSERT’* button to submit the form



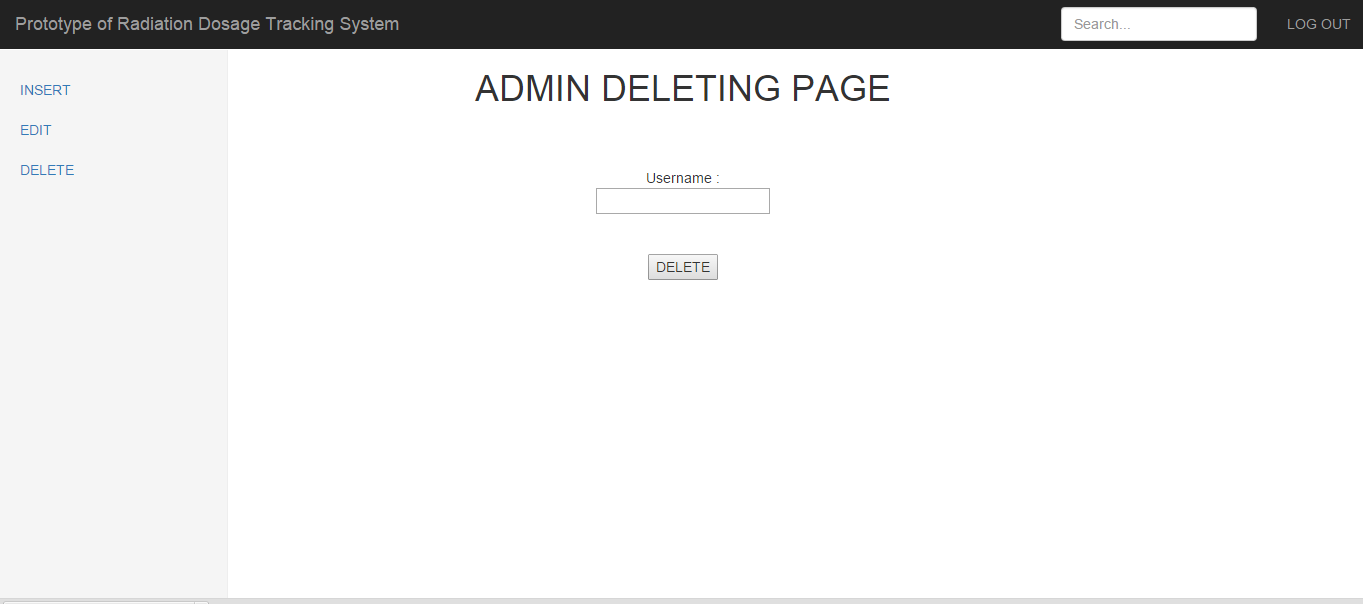
*‘****On Edit’*** contains the following controls:

* All the data fields managed by textboxes
* *‘ID’/‘Firstname’ / ‘Lastname’ / ‘Username’ / ‘password’ / ‘Email’ / ‘Privilege’* fields to enter the user information
* *‘EDIT’* button to submit the form



*‘****On******Delete’*** contains the following controls:

* All the data fields managed by textboxes
* *‘Username’ ‘Privilege’* field to enter the user information
* *‘DELETE’* button to submit the form



**I1.2.b - Application Logic:**

This section explains how the various events are implemented and also about the data access components used to perform data operations.

**On Page Load:**

As soon as the admin page loads, the following occur:

1. Table is generated with all user information
2. Three links to insert, delete or edit user information are loaded.

**On Insert:**

This operation is accomplished using action queries that take the values provided in the text boxes and post them to the database. Values entered into the textboxes are posted into the database and thus a record will be inserted.

**On Edit:**

This operation is also accomplished using action queries that take the values provided in the text boxes and post them to the database. Values entered into the textboxes are posted into the database and thus a record will be updated with these entered values.

**On Delete:**

This operation is also accomplished using action queries that take the values provided in the text boxes and post them to the database. In this case of delete operation determine the record to be deleted from the User ID and delete information related to the user from all the related tables. The order in which the database operations are performed is dictated by the Primary Key/Foreign Key relationships.

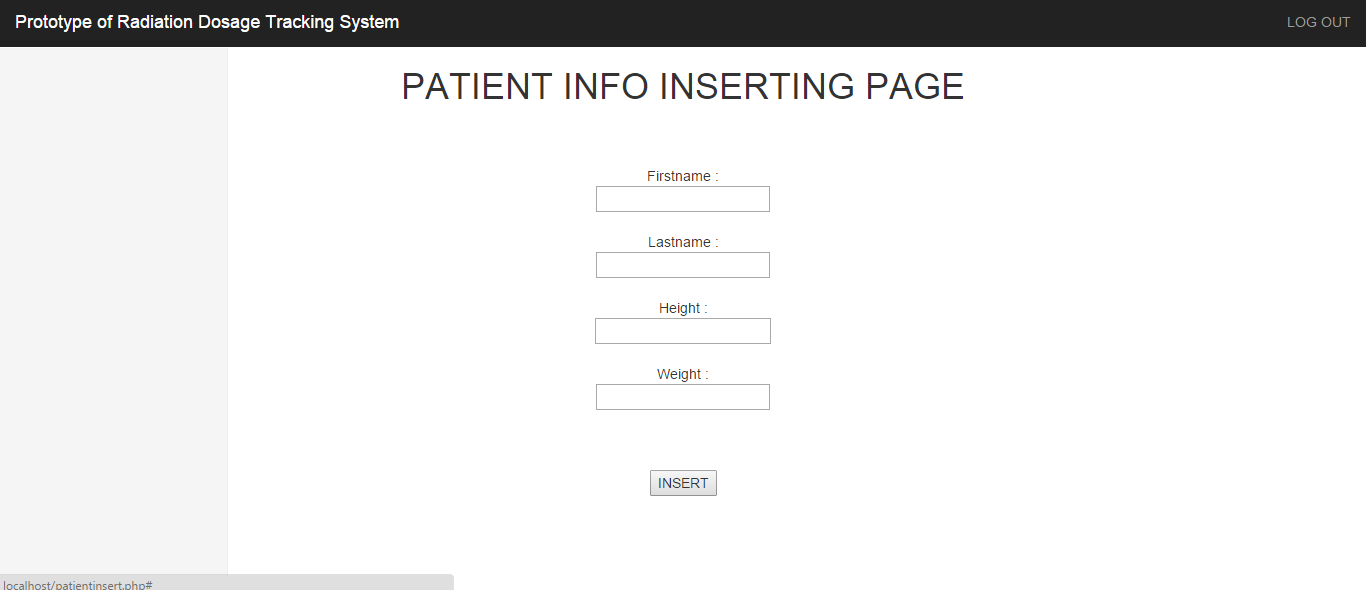
**Iteration 2: *Physician and patient details maintenance module***

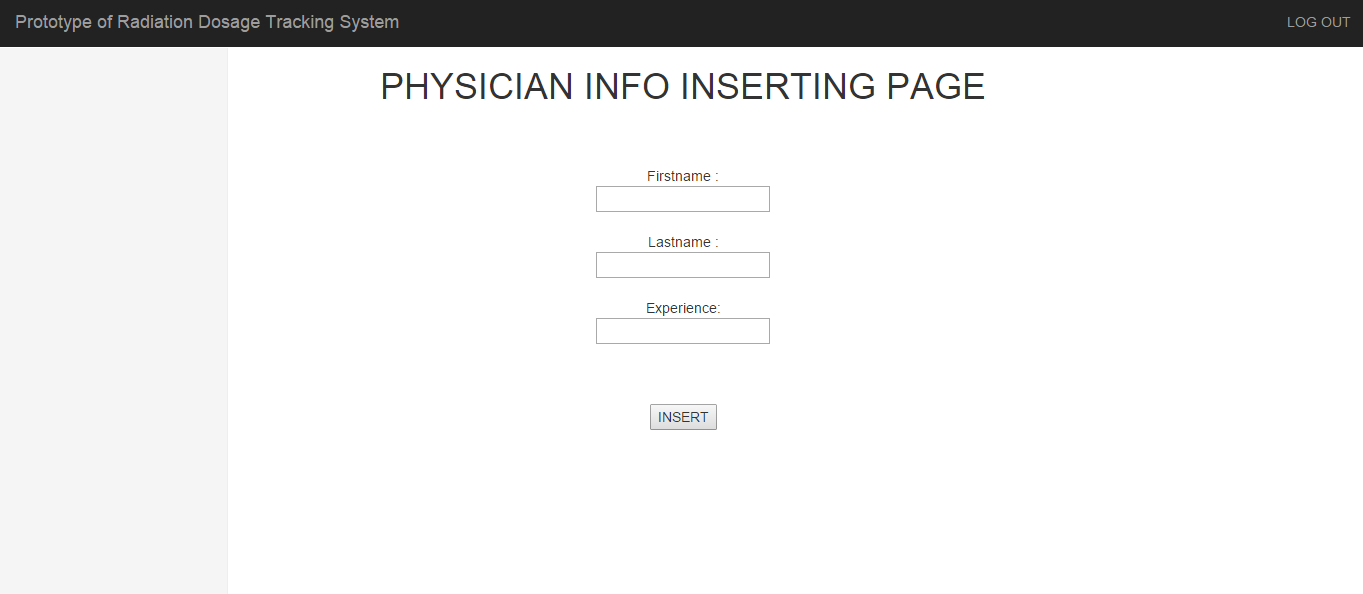
This module implements the functionality required to

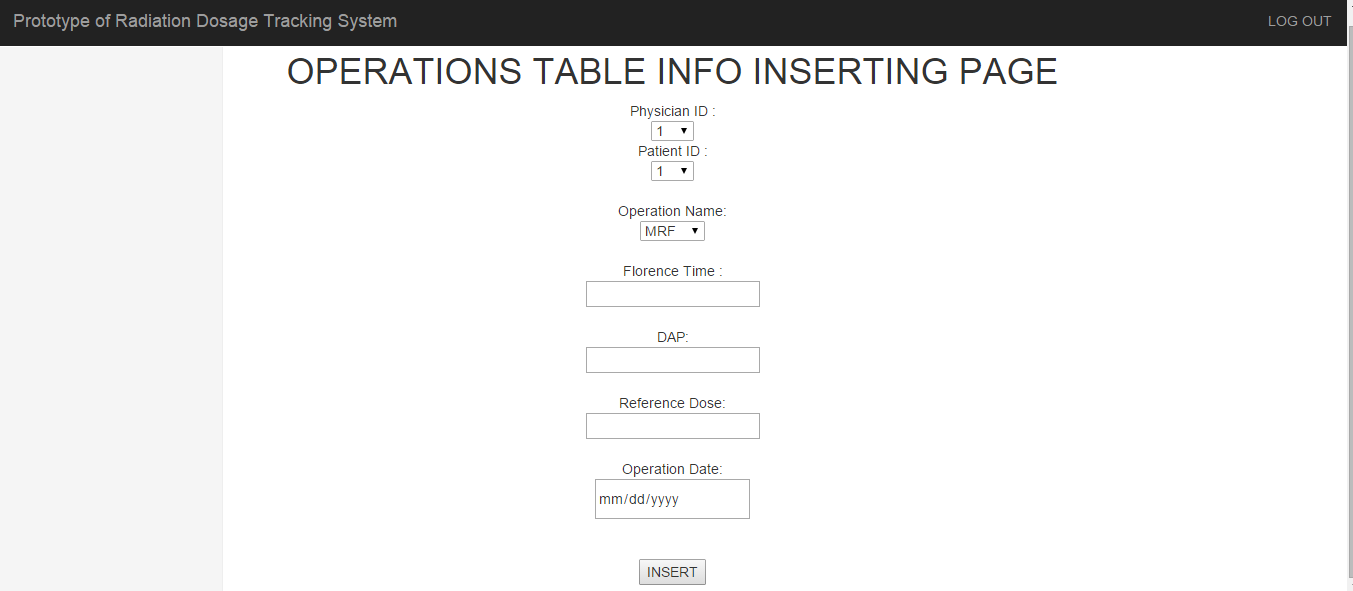
* Insert the data into the database tables “patient table, physician table and operations table”
* “INSERT” button to insert values into the database

**I2.a - User-Interface**

The user interface for this module are displayed below. This module has three links to pages “patient info insertion page”, “physician info insertion page”, “operations table insertion page” which will redirect to physician insertion page, patient insertion page and operation insertion page respectively which will have all fields to input the required values.







**I2.b - Application Logic**

These operations are accomplished using action queries that take the values provided in the text boxes and post them to the database. Values entered into the textboxes are posted into the database and thus records will be inserted. Here Physician ID, Patient ID will be auto generated for the physician table, patient table and for the insertion into the operations table, there is a dropdown menu to select Physician ID, Patient ID, operation name.

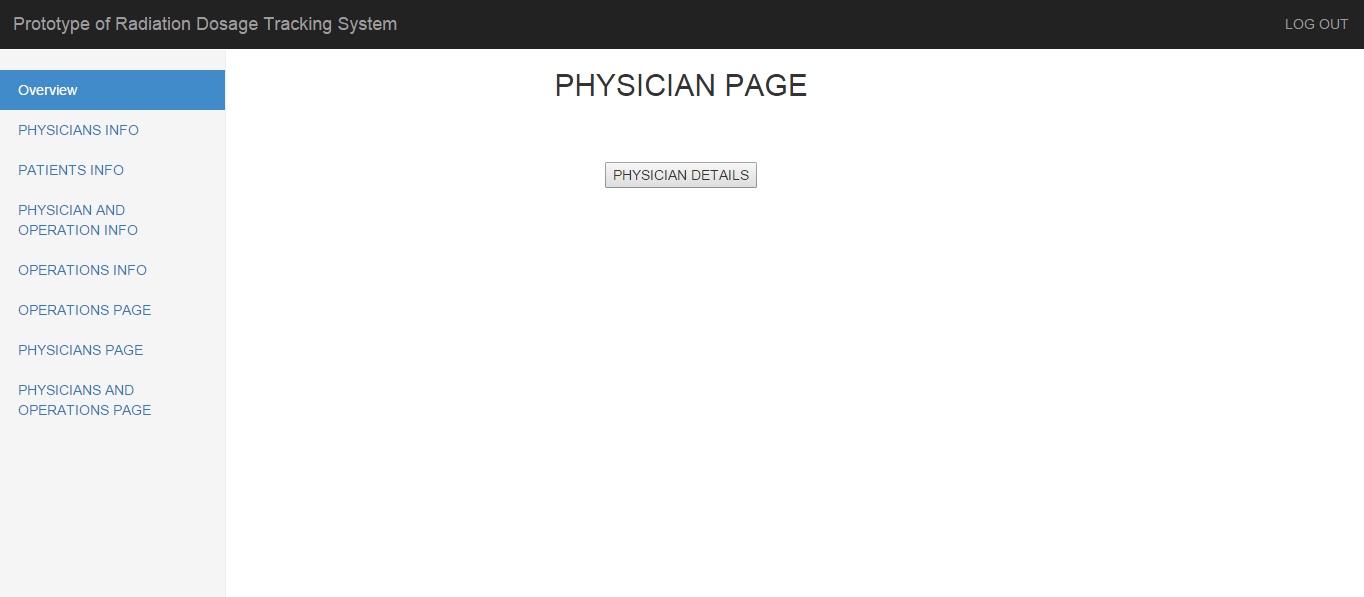
**Iteration 3: *Physician and patient details retrieval module***

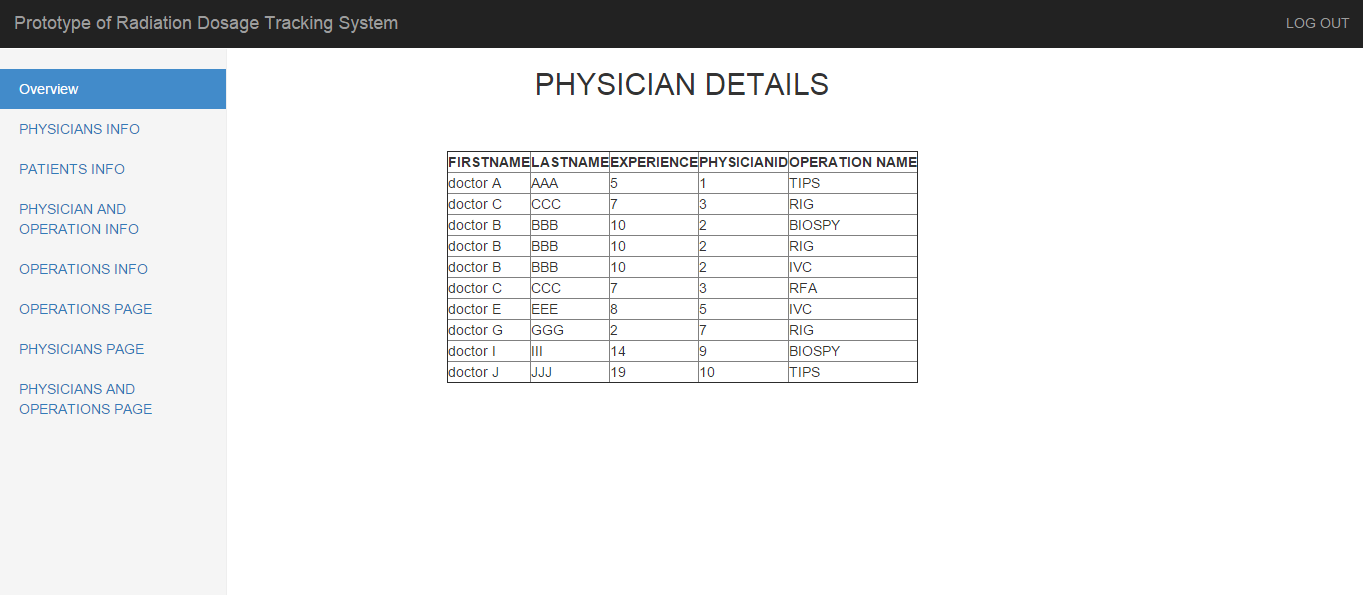
This module also uses a form driven database interface. The information from the patient table, physician table and the operations table are retrieved and displayed. All the physician details are retrieved from the database (“physician table”) and also the operations performed by him/her are also retrieved from the database (“operations table”) when the “PHYSICIANDETAILS” button and “PATIENTDETAILS” buttons are clicked. And all of these details are displayed in the form of a table.

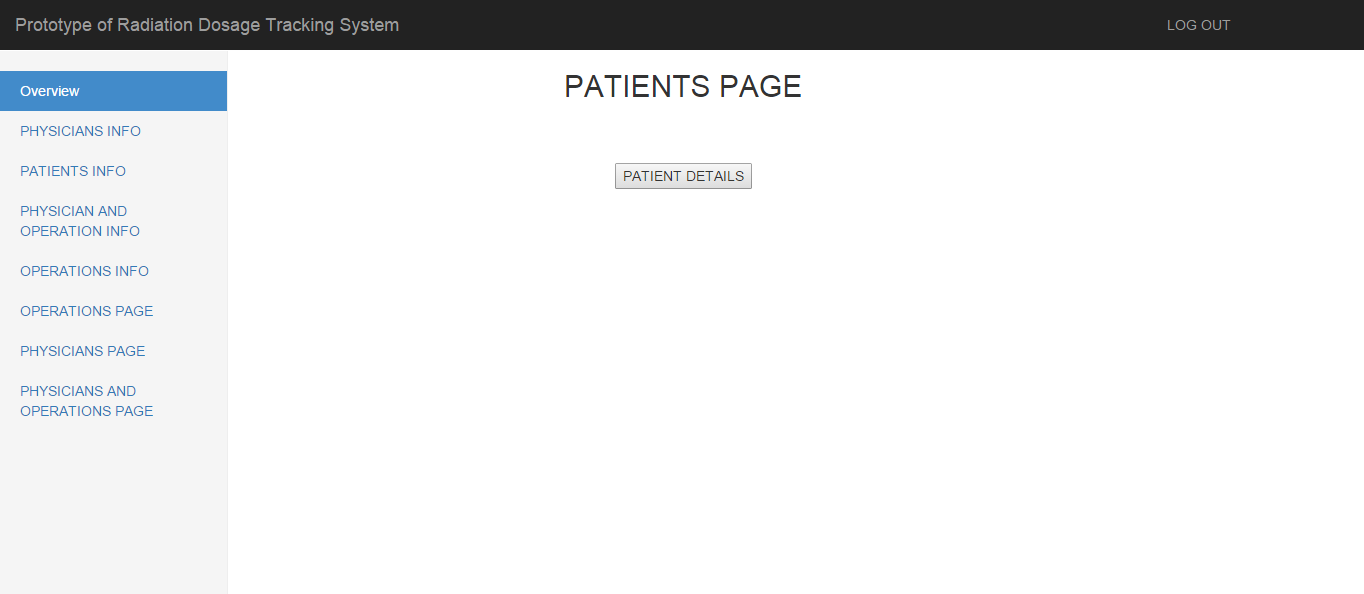
**I3.a - User-Interface**

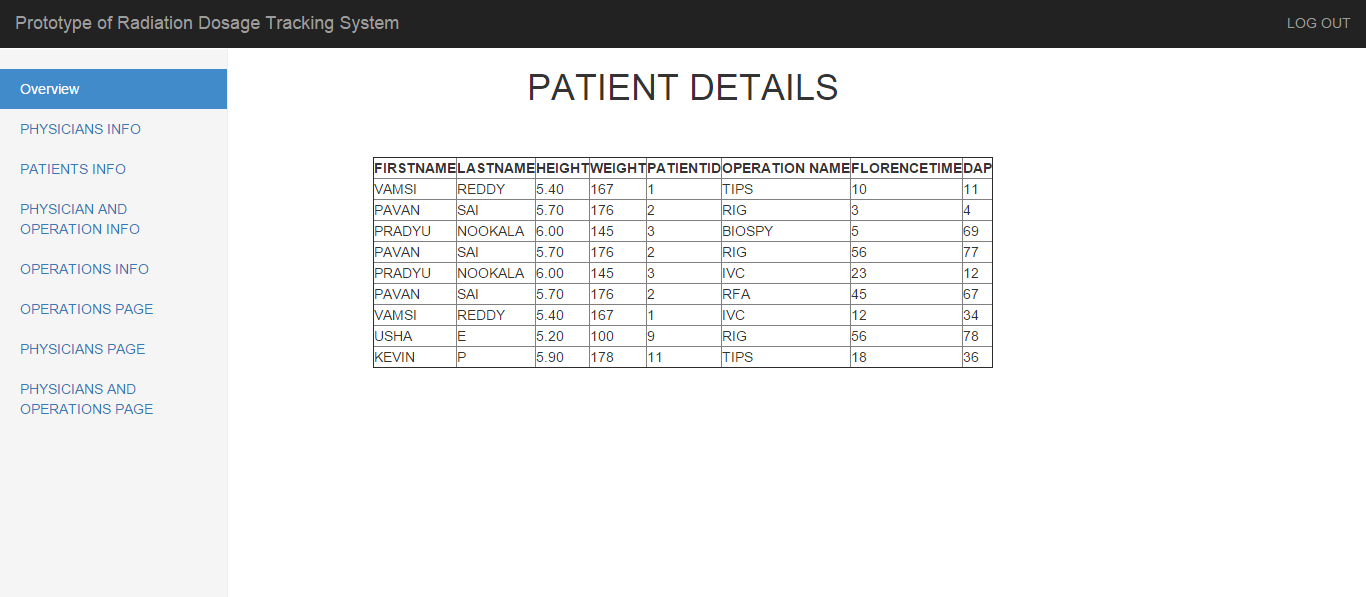
Figure I2.1 shows a screen shots of the user interface designed contains the following controls:

* + - “PATIENTDETAILS” button
    - “PHYSICIANDETAILS” button









**I3.b - Application Logic:**

In this module, the logic is clicking on “PHYSICIAN DETAILS” button invokes the SQL query "select p.firstname, p.lastname,p.experience,p.physicianID,g.operation name from physiciantable p,generaltable g where p.physicianID=g.physicianID" and clicking on “PATIENT DETAILS” button invokes the SQL query select p.firstname,p.lastname,p.height,p.weight,p.patientID,g.operationame,g.florencetime,g.dap from patienttable p,generaltable g where p.patientID=g.patientID.

*Average reference dosage module:* Users are given access to select operation names and physician names respectively and this will give average reference dosage for the selected operations and selected physicians respectively. This module covers requirements 013-014

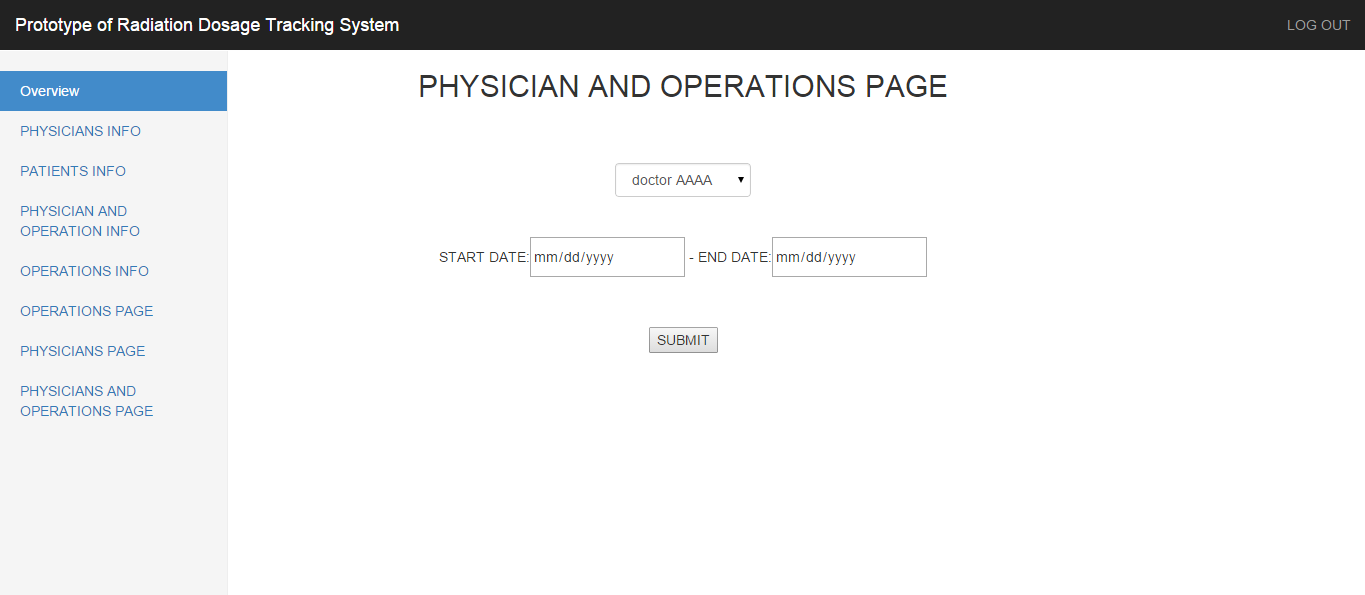
**Iteration 4: *Physician and operation* *details in a given range of time module***

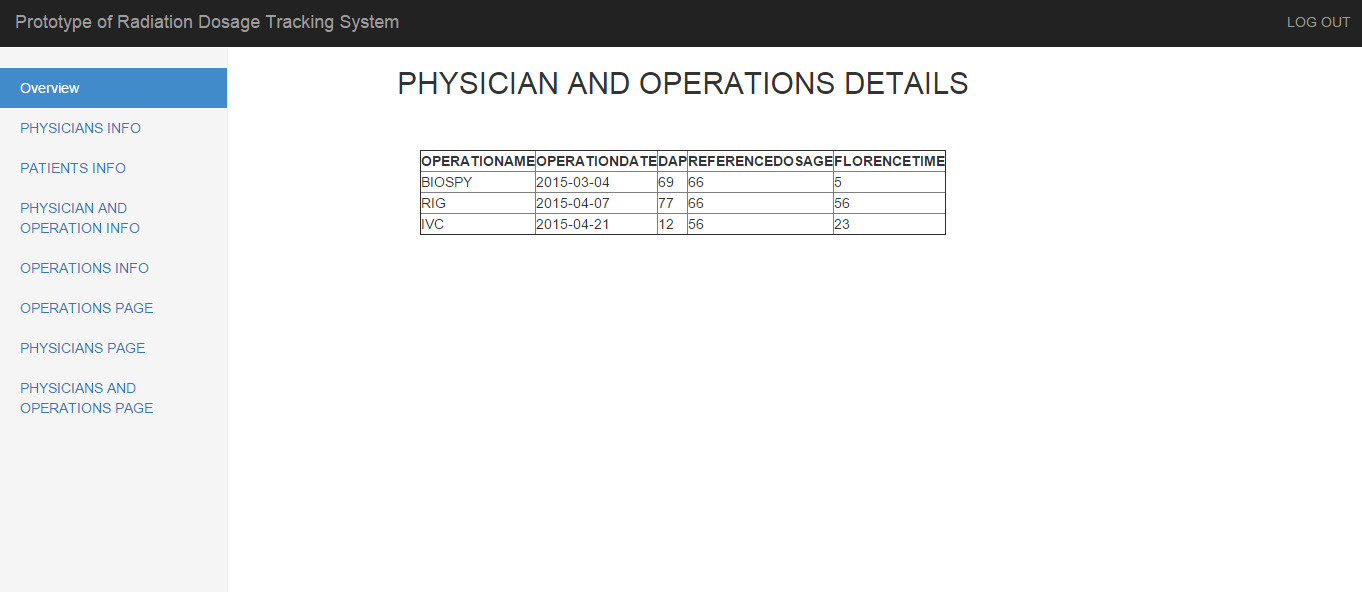
This module is a gateway to view all the operations the selected physician from checkbox has performed and also the reference dosage, DAP, florencetime, operation date of those operations

**I4.a- User-Interface**

This module has:

1. Drop down menu that displays doctors names
2. Text boxes to enter the range of dates
3. “Submit” button to submit the form





**I4.b - Application Logic**

In this module, the logic is clicking on “SUBMIT” button invokes the SQL query “select \* from generaltable g where g.operationdate between '".$\_POST['start\_date']."' and '".$\_POST['end\_date']."' and g.physicianid in (select physicianid from physiciantable p where p.physicianID='".$\_POST['doctor\_name']."')”

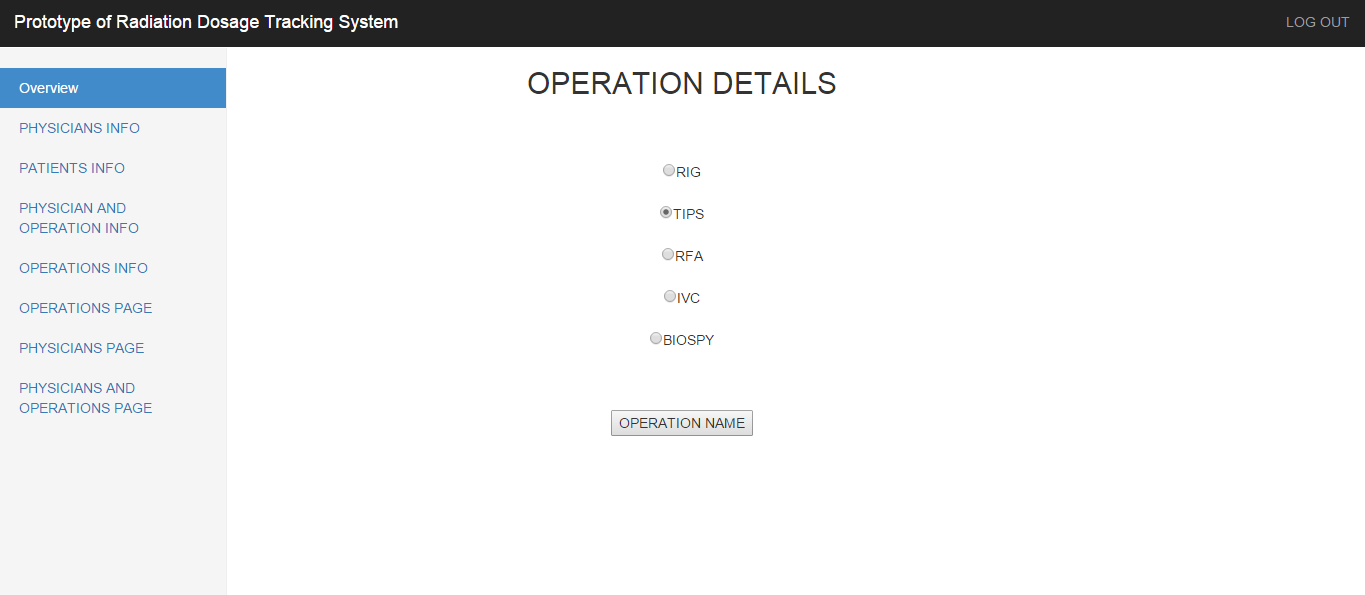
**Iteration 5: *Operation details maintenance module***

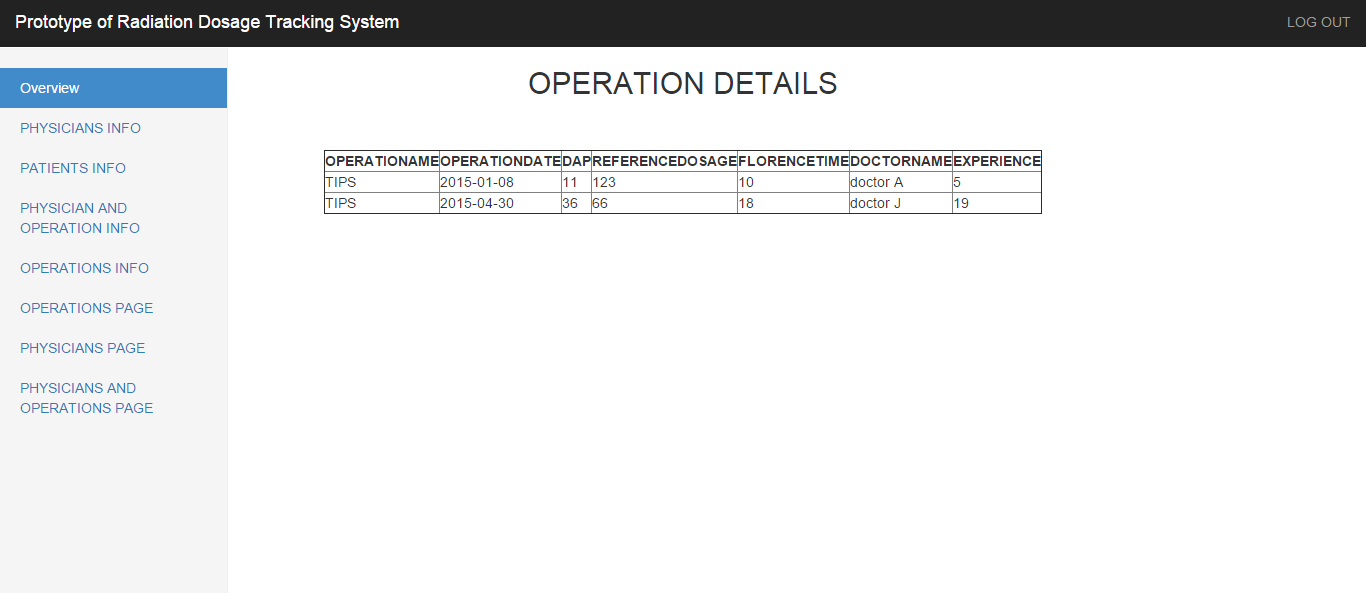
This module implements the functionality required to retrieve the operation details and also the doctor name who performed this operation and his experience

**I5.a- User-Interface**

The user interface for this module is displayed below. This module provides controls namely:

1. Radio buttons that lists the type of operations.
2. A button that is only enabled when the operation is selected in the radio box. This button is responsible for the SQL code formulate to generate operation details and doctor details.





**I5.b - Application Logic**

When an operation is selected from the radio buttons, the “OPERATION NAME” button is enabled. On clicking that button the following powerful and slick sql command is fired to generate operation details:

"select \* from generaltable g,physiciantable p where g.physicianid = p.physicianid and g.operationname = '".$\_POST['operation']."'" and the logic

while($row = mysqli\_fetch\_array($result)) {

echo "<tr><td>" . $row['operationname'] . "</td><td>" . $row['operationdate'] ."</td><td>" .$row['DAP']

."</td><td>".$row['referencedose']."</td><td>".$row['florencetime']."</td><td>".$row['firstname']."</td><td>".$row['experience']. "</td></tr>";

}

**Iteration 6: *Average reference dosage module***

This module implements the functionality required to

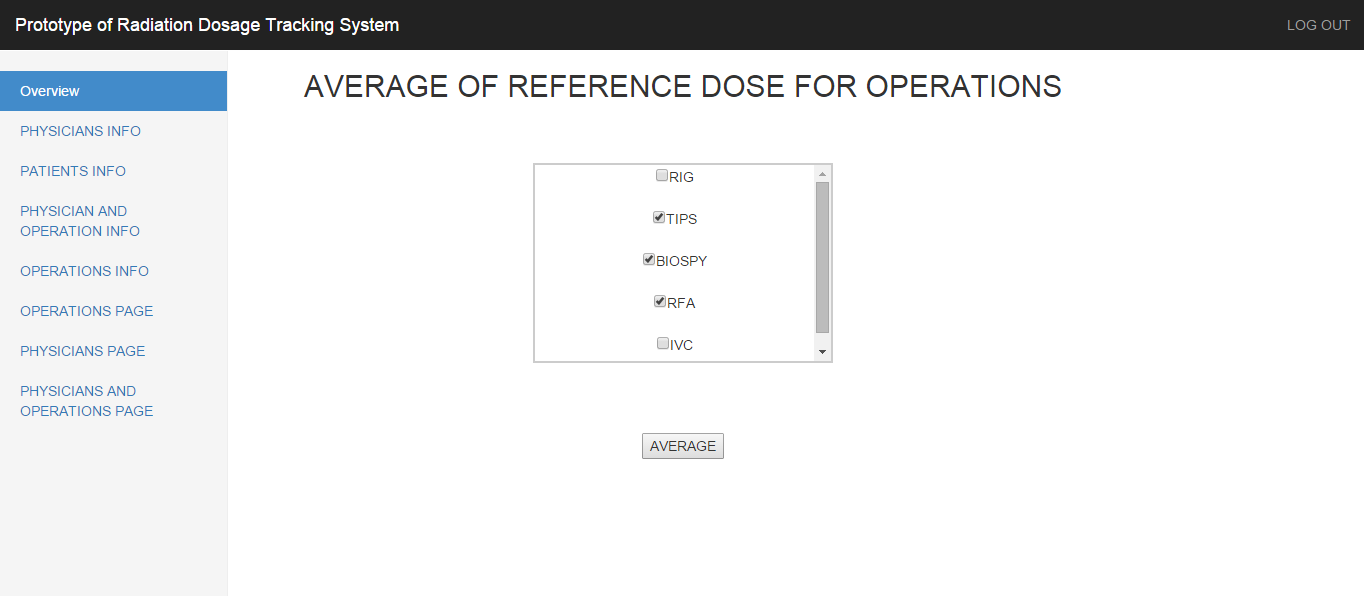
* Get the average reference dosage for the selected doctors
* Get the average reference dosage for the selected operations

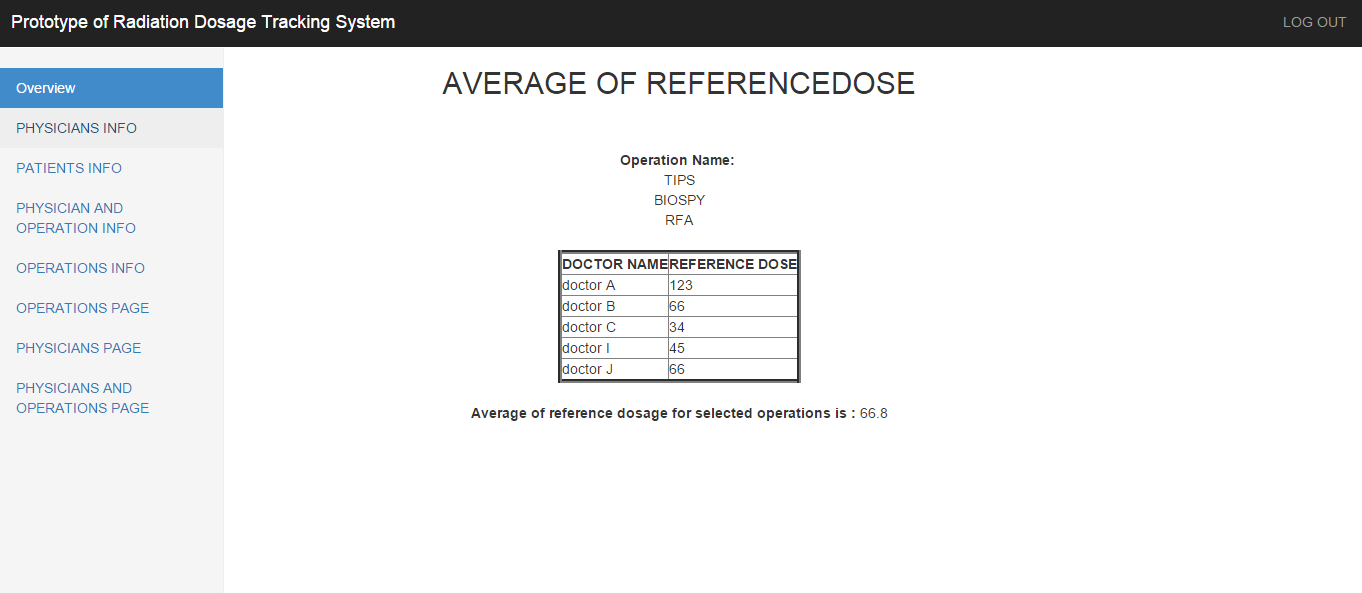
**I6.a - User-Interface**

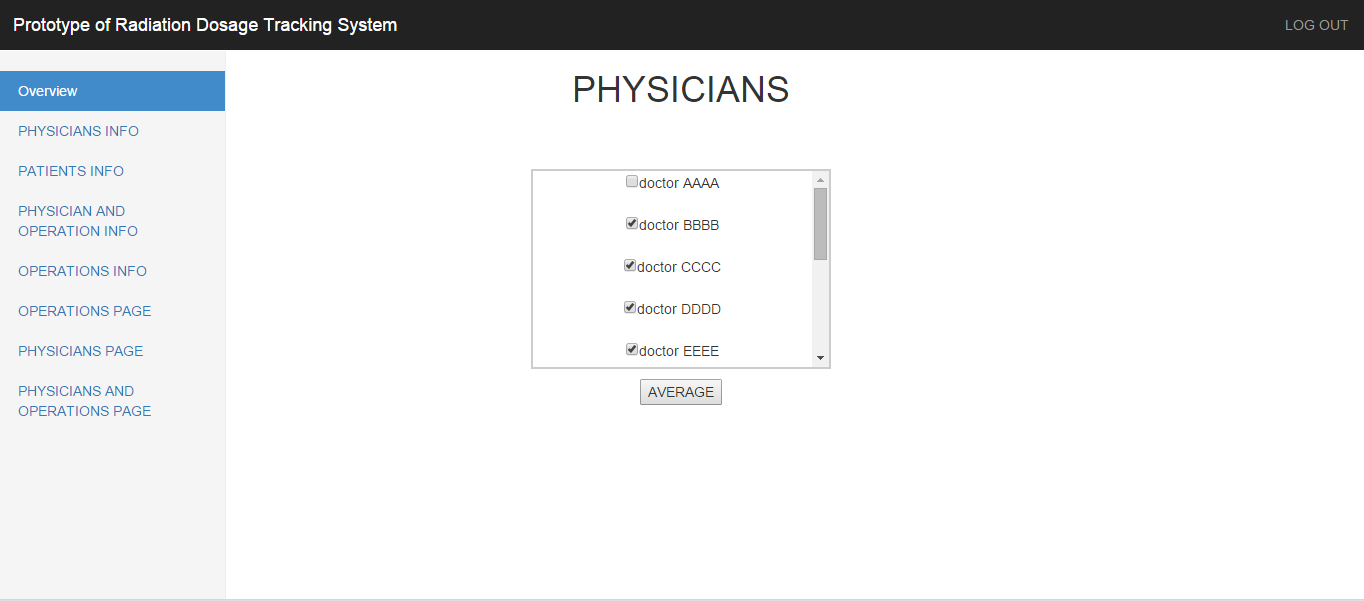
The user interfaces for these modules are displayed below. These modules provides controls namely:

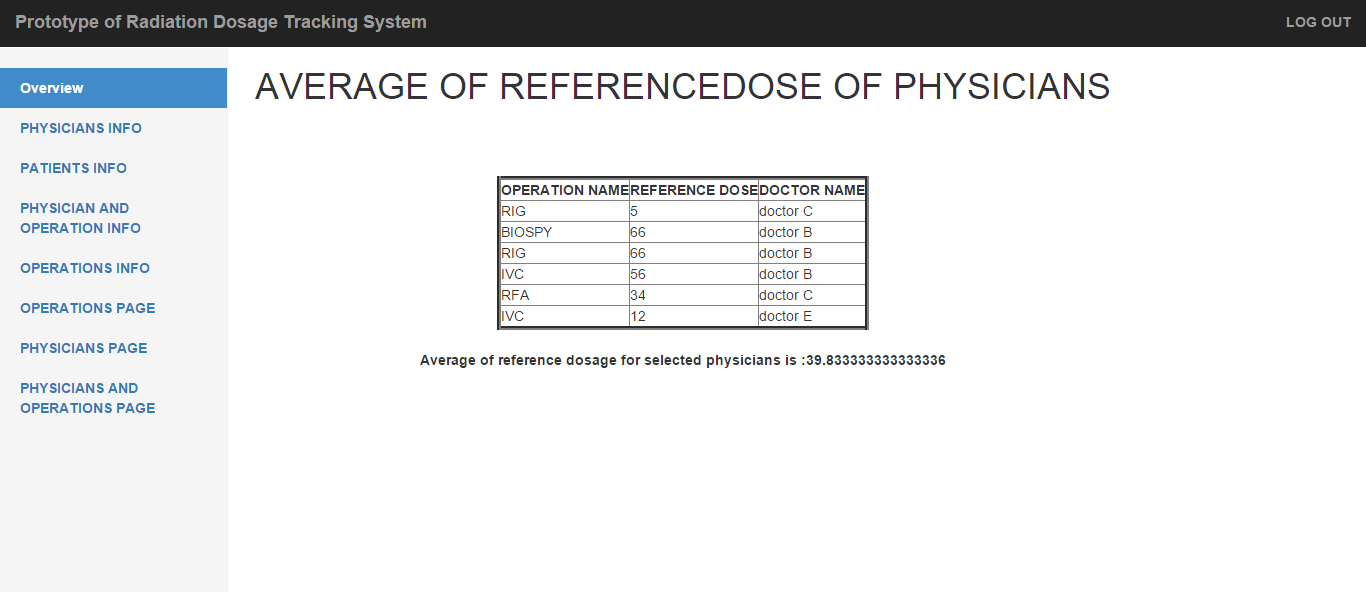
1. Check boxes that lists the type of operations, doctors separately in different pages.

2. Button that are only enabled when the test type is selected in the check boxes. This button is responsible for the sql code formulate to generate average reference dosage









**I6.b - Application Logic**

* When the operation names are selected from the check boxes, the “Average” button is enabled. On clicking that button the following powerful and slick sql command is fired to generate average reference dosage:

"select p.firstname,g.referencedose from generaltable g, physiciantable p where g.operationname in ('".$ope\_name[0]."','".$ope\_name[1]."','".$ope\_name[2]."','".$ope\_name[3]."','".$ope\_name[4]."') and p.physicianid = g.physicianid ")

* When the doctor names are selected from the check boxes, the “Average” button is enabled. On clicking that button the following powerful and slick sql command is fired to generate average reference dosage:

"select g.operationname,g.referencedose,p.firstname from generaltable g, physiciantable p where p.firstname in ("; for($x=0;$x<$arrlength-1;$x++) $query3 .= "'".$phy\_name[$x]."',"; $query3 .="'".$phy\_name[$arrlength-1]."') and p.physicianid = g.physicianid "

**Iteration 7: *Graphical details module***

This module implements the functionality required to

* select the desired doctors , operations and range of dates in a single page
* Display a list of all the selected doctors and the operations names and their details
* Displays a bar chart depicting the doctors and their reference dosage over the months of time

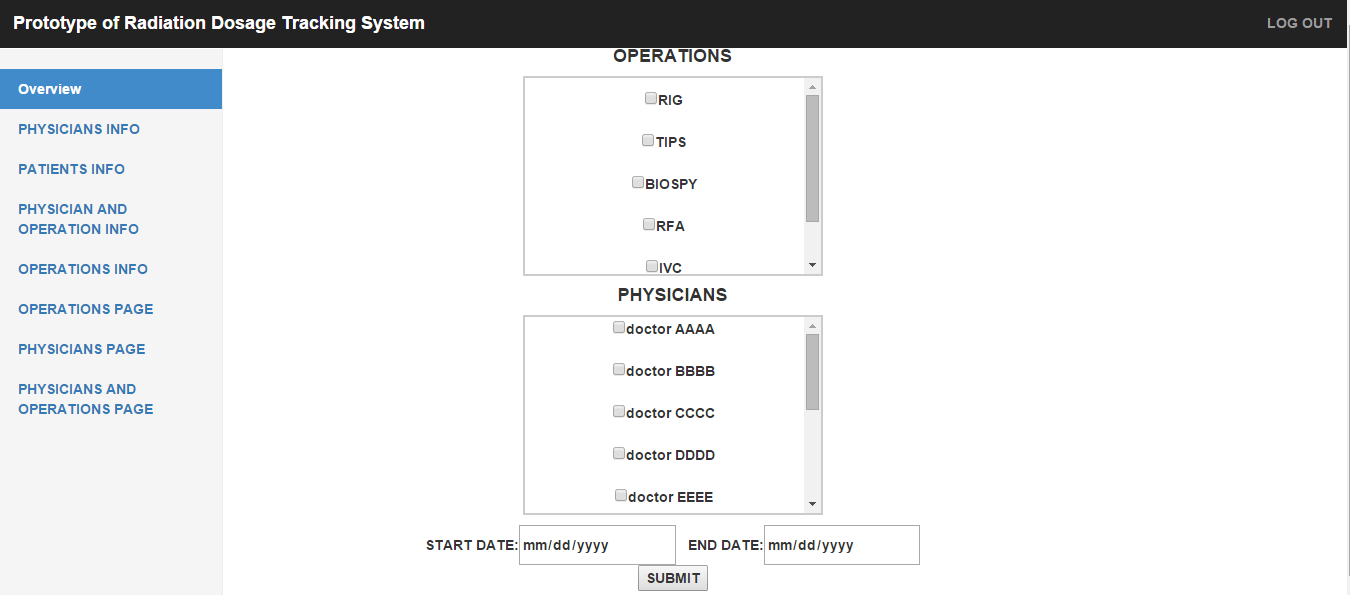
**I7.a - User-Interface**

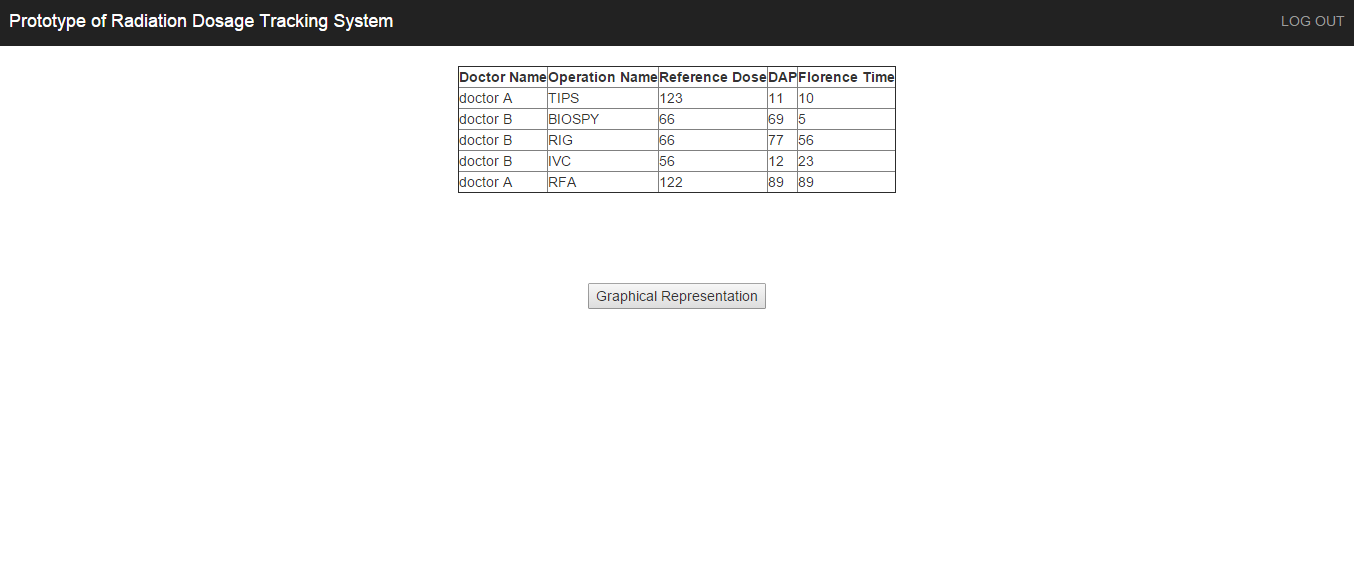
The user interface for this module is displayed below. These modules provides controls namely:

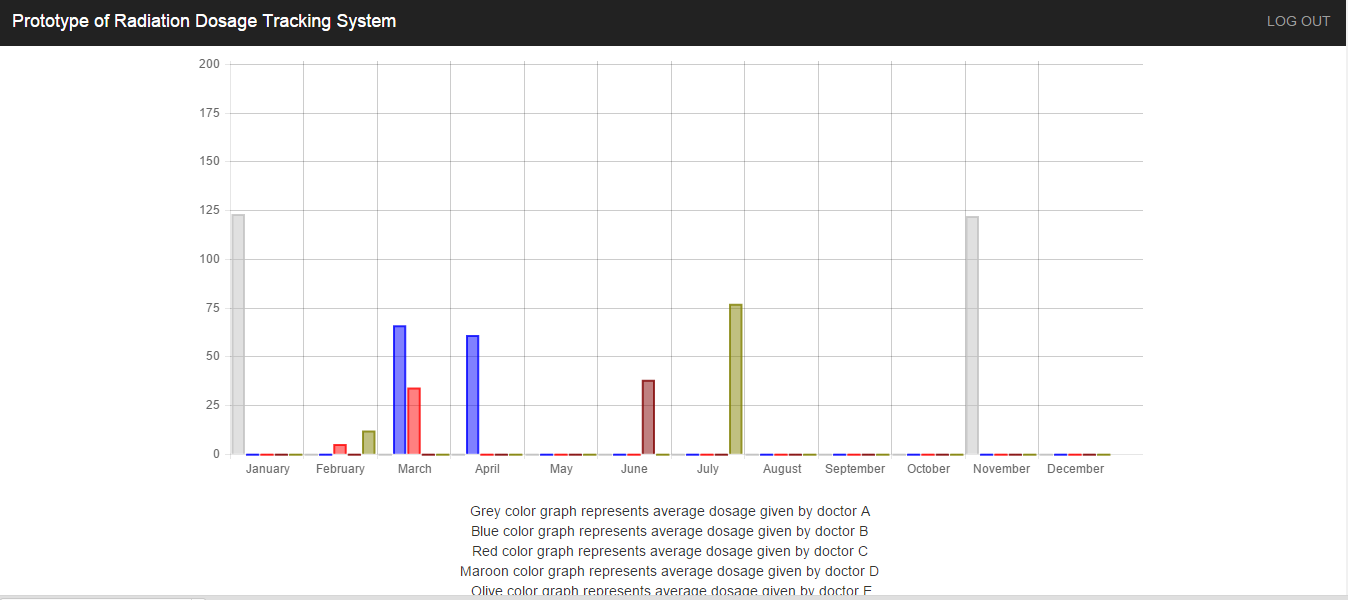
1. Check boxes that lists the type of operations, doctors separately in different pages.

2. Button that are only enabled when the test type is selected in the check boxes. This button is responsible for the sql code formulate to generate average reference dosage

3. Button “GRAPHICAL REPRESENTATION” to display the bar chart







**I7.b - Application Logic**

When the operation names, doctor names, range of dates are selected from the check boxes, text boxes for dates, the “SUBMIT” button is enabled. On clicking that button the following powerful and slick sql command is fired to generate the data:

"select firstname,operationname,referencedose,DAP,florencetime from generaltable g,physiciantable p where operationname in

('".$ope\_name[0]."','".$ope\_name[1]."','".$ope\_name[2]."','".$ope\_name[3]."','".$ope\_name[4]."') and p.firstname in (";

for($x=0;$x<$arrlength-1;$x++)

$query .= "'".$phy\_name[$x]."',";

$query .="'".$phy\_name[$arrlength-1]."')

and g.physicianID=p.physicianID and g.operationdate between '".$\_POST['start\_date']."' and '".$\_POST['end\_date']."'";

On clicking the “GRAPHICAL REPRESENTATION” button, a graph will be displayed.

**5. Implementation**

The implementation will be performed based on the functional design conceived for each module. A summary of Unit and Integration Tests performed during each release is also documented in this section .

**5.1 Implementation Challenges**

Some modules require a form-like user interface. This poses unique challenges to the way database operations are managed and implemented in a Web application environment. Web applications are stateless, meaning that the clients request a page from the server and the server provides the required information. After this set of transaction is completed the server would have no knowledge of the client. So, in this environment, it is a challenging task to develop system functionality that mimics stateful services that track the information sent to a client and remembers the current state. It requires complex functionalities like cookies and sessions to maintain information on the current state and determine the future states.

This is the reason why traditional web applications use tabular user interfaces, where in the data is all pre-fetched and stored in the page, thus relying on minimal amount of state information to be maintained. The tabular interface requires that the client request the server for data only once and all the subsequent operations could be managed with relative ease. In this project, tabular user interfaces are being used.

**5.2 Implementation:**

**Iteration 1:**

The Authorization/Authentication module was constructed first and hosted in the following pages:

1. zang\_login\_page.html - Implement user interface
2. admindata.php,admin-page.php - Implement application logic

The User Details Maintenance module was then constructed and hosted in the following pages:

1. admin\_insert.php, admin\_delete.php , admin\_edit.php - Implement user interface
2. insert.php,delete.php,edit.php - Implement application logic

The two modules were then integrated into a single application with the Authorization/Authentication module pointing to zang\_login\_page.html as the starting page for the application.

**I1 - Unit Test Summary**

The following errors were uncovered during unit testing and corrected immediately:

|  |  |
| --- | --- |
| **Errors** | **Corrections** |
| User ID should be auto generated in the user table | Completed using auto increment option in the database |
| Privilege should have values 1 or 0 | Completed |
| Indicate required fields, maximum allowed length and formats for the input fields along with the controls provided for input. | HTML labels have been added to display the required information |
| Only admin should be able to insert, delete or update the users information | Completed by providing those privileges to users who login as admin |
| Hide the inactive buttons on the page. | Completed |

**I1 - Integration Test Summary**

The following errors were uncovered during the integration of the Authentication/Authorization module and the User Details Maintenance module.

|  |  |
| --- | --- |
| **Errors** | **Corrections** |
| Pressing ‘Back’ after logout takes the user back to the secure page that was viewed before logging out. This is a major security concern as that secure page may contain confidential information. | As per the system design, each secure page that is requested from the server can be accessed only with valid credentials. So, after logging out when the user clicks the “back” button in the browser, they can only view the local cached copy of the page. However, if they try to refresh the page or navigate to another page or modify data being displayed on the page it warrants a request to the server, which will promptly be denied. Hence, by not allowing the client’s browser to cache the page locally, this problem can be resolved. The server-side caching should however be enabled in order to server the pages faster.  Each secure page developed thus far, has been updated such that on loading that page, the client’s browser would be prevented from caching the page. Each secure page that is to be developed further would also have to comply with this requirement. |

**Iteration 2:**

The Physician and patient details maintenancemodule was constructed and hosted in the following pages:

1. physicianinsert.php,patientinsert.php,generalinsert.php– Implements user interface
2. insertpy.php,insertpa.php,insertge.php– Implements application logic

**I2 - Unit Test Summary**

The following errors were uncovered during unit testing and corrected immediately:

|  |  |
| --- | --- |
| **Errors** | **Corrections** |
| Initially ID’s were also being inserted into the tables through frontend | This is corrected by enabling auto increment option in the database and by providing a dropdown menu |
| Make inactive buttons invisible. | Completed |
| Indicate maximum allowed length and formats for the input fields. | Completed |

**I2 - Integration Test Summary**

The Test Appointments module was integrated with the “Release 1” components. There were no errors accounted for.

**Iteration 3:**

The Physician and patient details retrieval module was constructed and hosted in the following pages:

1. patient.php, physician.php – Implements user interface
2. patient\_details.php, physician\_details.php – Implements application logic

**I3 - Unit Test Summary**

The following errors were uncovered during unit testing and corrected immediately:

|  |  |
| --- | --- |
| **Errors** | **Corrections** |
| Logout button was missing in patient.php page | .  Corrected by adding logout button in patient.php |

**I3 - Integration Test Summary**

The Physician and patient details retrieval module was integrated with the “Release 2” components. There were no errors accounted for.

**Iteration 4:**

The Physician and operationdetails in a given range of time module was constructed and hosted in the following pages:

1. pno.php– Implements user interface
2. doctorcheck.php– Implements application logic

**I4 - Unit Test Summary**

The following errors were uncovered during unit testing and corrected immediately:

|  |  |
| --- | --- |
| **Errors** | **Corrections** |
| When the Select Doctors page loads, the doctor name dropdown box, which is used for selecting the name of the doctor, is blank. It would be helpful to have a directive like “—select one--” or “any doctor name” in the dropdown box. | The option “—first doctor name--” was added in the dropdown box |

**I4 - Integration Test Summary**

The Physician and operationdetails module was integrated with the “Release 3” components. There were no errors accounted for.

**Iteration 5:**

The Operation details maintenance moduledetails in a given range of time module was constructed and hosted in the following pages:

1. ope.php– Implements user interface
2. type\_operation.php– Implements application logic

**I5 - Unit Test Summary**

The following errors were uncovered during unit testing and corrected immediately:

|  |  |
| --- | --- |
| **Errors** | **Corrections** |
| Initially checkbox was used to select operations in ope.php even though the requirement was to select just one operation at a time | This is corrected by changing checkbox to radio button in ope.php page |

**I5 - Integration Test Summary**

The Operation details maintenance module was integrated with the “Release 4” components. There were no errors accounted for.

**Iteration 6:**

The Average reference dosage moduledetails in a given range of time module was constructed and hosted in the following pages:

1. avgop.html,avgpy.html– Implements user interface
2. avgope.php,avgphy.php– Implements application logic

**I6 - Unit Test Summary**

The following errors were uncovered during unit testing and corrected immediately:

|  |  |
| --- | --- |
| **Errors** | **Corrections** |
| There was no scroll bar to move down if we add more operation or doctors | Scroll bar is added to the checkboxes for easy navigation |

**I6 - Integration Test Summary**

The Average reference dosage module was integrated with the “Release 5” components. There were no errors accounted for.

**Iteration 7:**

The Graphical details module details in a given range of time module was constructed and hosted in the following pages:

1. time.html– Implements user interface
2. main.php ,line.php– Implements application logic

**I7 - Unit Test Summary**

The following errors were uncovered during unit testing and corrected immediately:

|  |  |
| --- | --- |
| **Errors** | **Corrections** |
| There were errors trying to merge the code of graph(java script) into php code in main.php | Corrected |

**I7 - Integration Test Summary**

The Graphical details module was integrated with the “Release 6” components. There were some errors when trying to integrate this graph (line.php) into for www folder, where we have all php codes. This was rectified by adding some chart files in the same directory of php code.

**6. Change Requests**

Change requests may arise through changes in the business or issues in the project, which require changes to the requirement. The following is a list of the change requests that were received from the client.

|  |  |
| --- | --- |
| **Change Request** | **Solution Approach** |
| The physician and operations page, it has to redirect to graphical page. The current format is redirecting to tabular info page then to graphical info page. | In time.html, there is a button clicking on which, is redirecting to main.php because in action field of form there is main.php in time.html and then in main.php there is a button called graphical representation which will redirect to line.php which has the graph.  So, in main.php if the code of line.php is embedded , then we can directly get the graph from time.html |
| Physicians and operations records have to be inserted into the database through the user interface | Links are created in the html pages “patietinfo insert”, “operation info insert” which will have the fields and information inserted into these fields will get stored in the database |
| Only privileged users need to enter those details | A new column is added to “user table” which will have values 1 or 0 and if 1 means that corresponding users will have the privilege to enter the details into the patient and 0 means no privilege to enter the details |
| Have a bar chart for the doctors and their dosage with months for analysis purpose instead of a line graph | This requirement is met using High chart, which uses JavaScript |
| Retrieve the doctor names and their reference dosage for the selected operation names in “avgop.hml” page | The SQL query that fetches data from the database was modified to include doctor names and their reference dosage for the selected operation names |

**7. System Test**

As described earlier, the purpose of system testing is to identify defects that will only surface when a complete system is assembled. A thorough system test was conducted and it was made sure that the complete system that includes the code deliverables of all the iterations work together as a system.

**8. Technical Metric Collection**

* 1. **Estimated lines of code:**7000
  2. **Actual lines of code:**6000
  3. **Complexity :**

|  |
| --- |
| Quantitative Metrics |
| Files | | 40 |
| Logical Lines of Code LLOC | | 1335 |
| Single Line Comments | | 146 |
| Multi Line Comments | | 18 |
| High Quality Comments | | 54 |
| Strings | | 433 |
| Numeric Constants | | 92 |

**9. Learning Outcome**

This project helped me to d**evelop project management skills** like gathering date to identify customer requirements, develop and present design specifications, creating and refining preliminary design mockup and **to develop the ability to communicate effectively** verbally, in writing and electronically by documenting application/site changes, creating web site content, developing and implementing usability testing. This project helped me to enhance my **technical skills** required for Web Designers through use of W3C standards, PHP, MySQL, HTML5, CSS, Java script and highchart.

**10. Definition of Terms / Abbreviations / Acronyms**

|  |  |
| --- | --- |
| Term/Acronym | Definition |
| RIG | Radiologically inserted gastrostomy |
| IVC | Inferior Vena Cavae |
| TIPS | Transjugular Intrahepatic Porto-systemic Shunt |
| RFA | Radiofrequency ablation |

**11. References**

1. http://www.w3schools.com/php/

2. http://www.w3schools.com/sql/

3. http://getbootstrap.com/

4. http://en.wikipedia.org/wiki/X-ray

5. http://php.net/manual/en/function.error-reporting.php

6. http://stackoverflow.com/questions/1053424/how-do-i-get-php-errors-to-display

7. http://en.wikipedia.org/wiki/Entity%E2%80%93relationship\_model

8. http://www.scism.sbu.ac.uk/law/Section5/chap6/s5c6int.html

9. http://en.wikipedia.org/wiki/Interventional\_radiology#Procedures

10. http://letsdophp.blogspot.com/p/architecture-diagram-of-php-based-web.html

11. http://en.wikipedia.org/wiki/System\_requirements