# ONLINE DIAGNOSTIC LAB REPORTING SYSTEM

A Project and Thesis Report
Submitted in fulfillment of the requirements for the degree of
**Bachelor of Science in Computer Science and Engineering**

By

| | |
|---|---|
| Md. Rafiul Islam | 12.02.04.029 |
| Kanis Fatima | 12.02.04.034 |
| Sumaya Kabir | 12.02.04.049 |
| Rashidul Hasan | 11.01.04.047 |
| Mirazul Kabir Bhuiyan | 11.01.04.107 |

Supervised by
Shanjida Khatun
Assistant Professor
Department of Computer Science and Engineering

AHSANULLAH UNIVERSITY OF SCIENCE & TECHNOLOGY
DHAKA, BANGLADESH
DECEMBER 2016

# DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of Ms. Shanjida Khatun, Assistant Professor, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over the final year courses, CSE400 : Project & Thesis I and CSE450 : Project & Thesis II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

| | |
|---|---|
| ……………………….. | ……………………….. |
| ( Md. Rafiul Islam ) | ( Kanis Fatima ) |

| | | |
|---|---|---|
| ……………………….. | ……………………….. | ……………………….. |
| ( Sumaya Kabir ) | ( Rashidul Hasan ) | ( Mirazul Kabir Bhuiyan ) |

Countersign By

…………………………………………..…..

Shanjida Khatun

Assistant Professor

Ahsanullah University Of Science & Technology

# ABSTRACT

The purpose of the project entitled as "ONLINE DIAGNOSTIC LAB REPORTING SYSTEM" is to develop web application to computerize the Management of a Diagnostic System which is user friendly simple, fast, and cost-effective. It deals with the collection of patient's information, diagnosis details, etc. Traditionally, it was done manually. The main function of the system is register and store patient details and staff details and retrieve these details as and when required, and also to manipulate these details meaningfully. System input contains patient details, diagnosis details, while system output is to get these details on to the screen. The Online Diagnostic Lab Reporting System can be entered using a username and password. It is accessible by every level of user according to their role. Every user can view the data that they are accessible. Only that user can add data into the database who has editing permission. The data can be retrieved easily. The data are well protected for personal use and makes the data processing very fast.

# TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

# Introduction

## 1.1 Overview

**Online Diagnostic Lab Reporting System** is designed for Any Diagnostic Center to replace their existing manual paper based system. The new system has the control of the following information; patient information, staff information ant test information related to the diagnostic system. These services are to be provided in an efficient, cost effective manner, with the goal of reducing the time and resources currently required for such task.

## 1.2 What is Online Diagnostic Lab Reporting System

A significant part of the operation of any diagnostic center involves the acquisition, management and timely retrieval of great volumes of information. This information typically involves; patient personal information, staff information, test information including schedules and results and payment history. All of this information must be managed in an efficient and cost wise fashion so that an institution's resources may be effectively utilized. **Online Diagnostic Lab Reporting System** will automate the management of the diagnostic center making it more efficient and error free. It aims at standardizing data, consolidating data, ensuring data integrity and reducing inconsistencies.

## 1.3 Why Online Diagnostic Lab Reporting System

Currently in our country most of our Diagnostic Centers use a manual system for the management and maintenance of critical information. The current system requires numerous paper forms with data stores spread throughout the diagnostic center management infrastructure, Often information on forms is incomplete or does not follow management standards. Forms are often lost in transit between departments requiring a comprehensive auditing process to ensure that no vital information is lost. Multiple copies of the same information exists in the diagnostic center and may lead to inconsistencies in data in various data stores. And also efficiency of the hospital daily work hampers in that conventional way. As you introduce our **Online Diagnostic Lab Reporting System**, we can say that this structured system will ensure the perfection in daily work routine of any diagnostic center and also restore the consistent performance of the staff due to the systematic, organized approach of our system.

# CHAPTER 2

# SYSTEM OVERVIEW

# System Overview

## 2.1 System Study

The goal of any system development is to develop and implement the system cost effectively, user-friendly and most suited to the user's analysis is the heart of the process. Analysis the study of the various operations performed by the system and their relationship within and outside of the system. During analysis, data collected on files, decision points and transactions handled by the present system. Different kinds of tools are used in analysis of which interview is a common one

### 2.1.1 Problem specification

The sole purpose of our **Online Diagnostic Lab Reporting System** is to minimize the human effort in the basic diagnosis procedure like patient's information, test schedule, test information etc. The main function of the system is to register and store test details and retrieve these details as and when required and to manipulate these details meaningfully. System input contains patient details, diagnosis details while system output is to get these details on the computer screen.

**Need**

1. Efficiently maintains the details about the patient, staff.
2. Simultaneously updates the changes made to any data or item in the database.
3. It is faster than manual system

## 2.2 System Analysis

System analysis is a detailed study of the various operations performed by a system and their relationship within and outside of the system. Here the key question is – What all problems exist in the present system? What must be done to solve the problem? Analysis begins when a user or manager begins a study of the program using existing system.

During analysis, data collected on the various files, decision points and transactions handled by the present system. The commonly used tools in the system are Data Flow Diagram, interviews etc. training, experience and common sense are required for collection of relevant information needed to develop the system. The success of the system depends largely on how clearly the problem is defined, thoroughly investigated and properly carried out through the choice of solution. A good analysis model should provide not only the mechanisms of problem understanding but also the framework of the solution. Then the proposed system should be analyzed thoroughly in accordance with the needs.

System analysis can be categorized into following parts.

- System planning and initial investigation
- Information gathering
- Applying analysis tools for structured analysis
- Feasibility analysis
- Cost/benefit analysis

### 2.2.1 Initial Investigation

The first step in system development cycle is the identification of need of change to improve or enhance an existing system. An initial investigation on existing system was carried out. The present system of diagnostic center is completely manual. Many problem were identified during the initial study of the existing system.

#### 2.2.1.1 Existing Systems

A Diagnostic Centre is a place where Patients come up for various pathology test according to diseases. Diagnostic Centre facilities like-

- ➢ Consultation by Doctors on Diseases.
- ➢ Diagnosis for diseases.
- ➢ Providing treatment facility.

Various operational works that are done in a Diagnostic Center are:-

- ➢ Recording information about the Patients that come.
- ➢ Generating bills.
- ➢ Recording information related to diagnosis given to Patients.
- ➢ Keeping information about various diseases and medicines available to cure them.

These are the various jobs that need to be done in a Diagnostic Center by the operational staffs. All these works are done on papers.

The work is done as follows:-

- ➢ Information about Patients is done by just writing the Patients name, age and gender. Whenever the Patient comes up his information is stored freshly.
- ➢ Bills are generated by recording price for each facility provided to Patient on a separate sheet and at last they all are summed up.

- Diagnosis information to patients is generally recorded on the document, which contains Patient information. It is destroyed after some time period to decrease the paper load in the office.
- Information about various diseases is not kept as any document. Doctors themselves do this job by remembering various medicines.

All this work is done manually by the receptionist and other operational staff and lot of papers are needed to be handled and taken care of. Doctors have to remember various medicines available for diagnosis and sometimes miss better alternatives as they can't remember them at that time.

Often information is incomplete or does not follow management standards. Forms are often lost in transit between departments requiring a comprehensive auditing process to ensure that no vital information is lost. Multiple copies of the same information exist in the hospital and may lead to inconsistencies in data in various data stores. Moreover, there is no online system for the existing system.

#### 2.2.1.2 Proposed System

Diagnostic System is designed for any diagnostic center to replace their existing manual paper based system. The new system is to control the information of patients, testing capability and patient invoices. These services are to be provided in an efficient, cost effective manner, with the goal of reducing the time and resources currently required for such tasks.

This system provides online storage/updating and retrieval facility. This system promises very less or no paper work and also provides help to the operational staff.

In this system everything is stored electronically so very less amount of paper work is required and information can be retrieved very easily without searching here and there into registers. This system is been discussed here.

### 2.2.1.3 Objective of the system

The objective of the Online Diagnostic Lab Reporting System is as follows-

- Define Diagnostic System
- Recording information about the Patients that come.
- Generating bills.
- Recording information related to diagnosis given to Patients.

Keeping information about various diseases and medicines available to cure them.

## 2.2.2 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are:

### 2.2.2.1 Economic Feasibility

This study is carried out to check the economic impact will have on the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products have to be purchased.

#### 2.2.2.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes for the implementing this system.

#### 2.2.2.3 Operational Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

## 2.3 System Specifications

As our proposed system is designed for a small or a medium scale organization, it doesn't require much resource at the users end. Bangladesh is stepping towards the digitization day by day but still most of our country is deprived of basic computer facility and high speed internet connection as well as lack of skilled man power to operate this kind of sophisticated system. As our goal is to establish this efficient system not only in urban diagnostic center but also rural diagnostic center, we design a simple but elegant system that will require minimum resource and gives effective and insured results. Below we describe our requirement for the system to run.

## 2.3.1 Software

OS: Windows XP or higher

Wamp Server for localhost

.Net Framework 3.0

## 2.3.2 Hardware

Pentium Duel Core Processor (2.1 GHz or more)

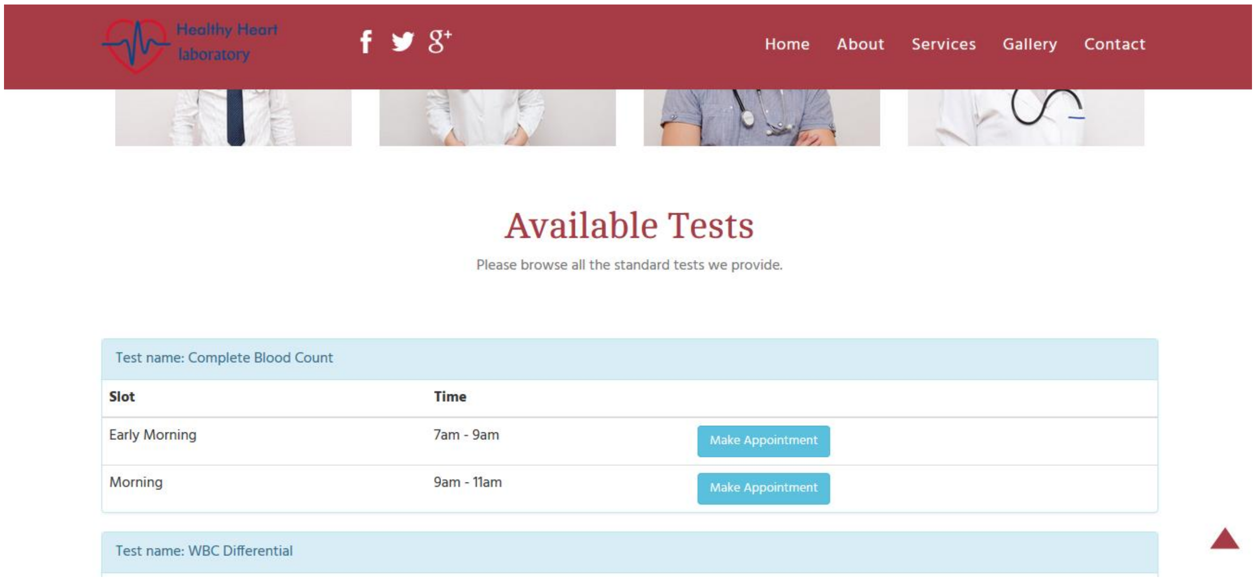
RAM (512 MB or more)

Modem or LAN for internet connection

## 2.4 Description of the System

The project Online Diagnostic Lab reporting System includes registration of patients, storing their details into the system, and also computerized billing in the pharmacy and labs. The software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. User can search availability of a doctor and the details of a patient using the id.
Online Diagnostic Lab reporting System can be entered using a username and password. It is accessible by every level of user according to their role. Every user can view the data that they are accessible. Only that user can add data into the database who has editing permission. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast.

At first, patient visits the website. There he see the free time slot for respective test. If he wants to test for his sickness, he has to make appointment. After clicking make appointment, he redirects to the register form.



Available Tests

Please browse all the standard tests we provide.

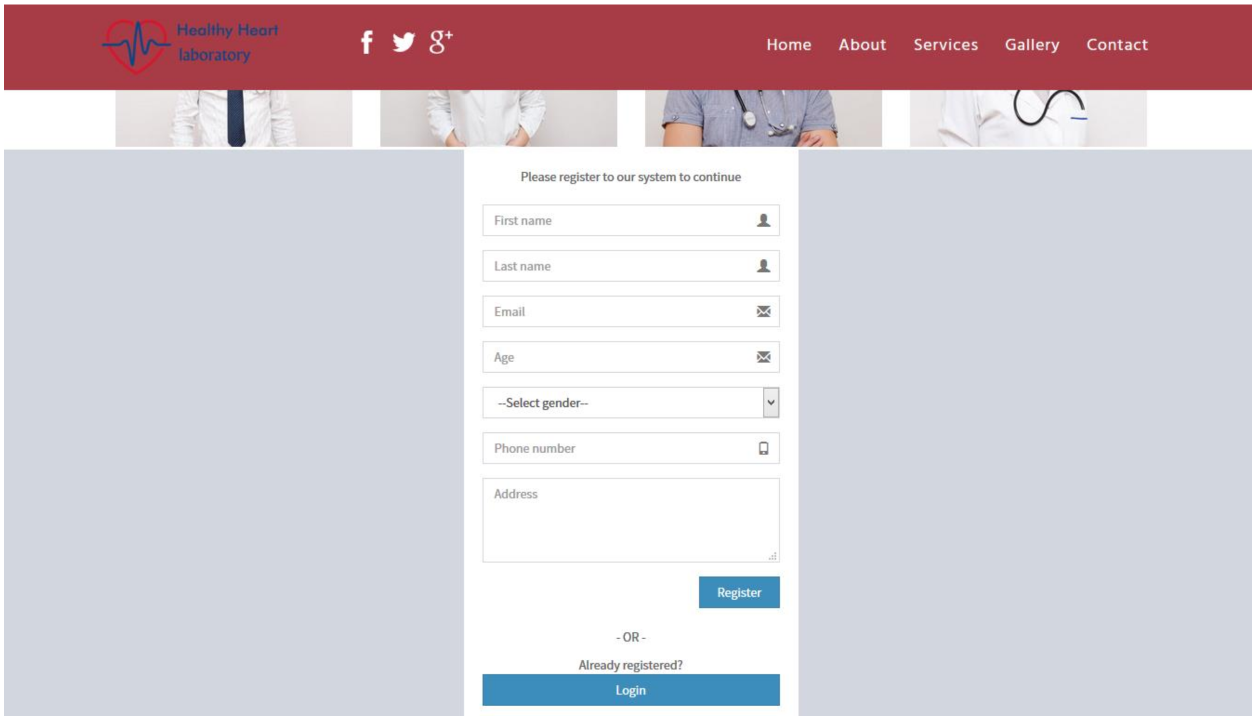
Test name: Complete Blood Count

| Slot | Time | |
|---|---|---|
| Early Morning | 7am - 9am | Make Appointment |
| Morning | 9am - 11am | Make Appointment |

Test name: WBC Differential

Figure 2.1: Website home page

In the resister form, he has to give proper information. After creating account, he redirects to his dashboard.



Please register to our system to continue

First name

Last name

Email

Age

--Select gender--

Phone number

Address

Register

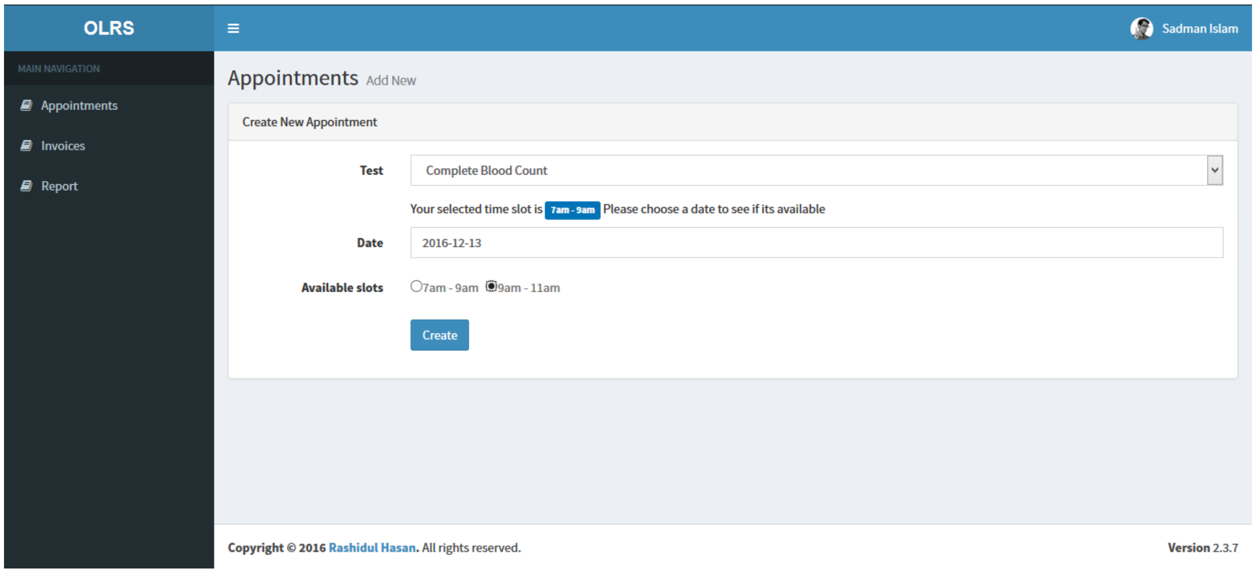
- OR -

Already registered?

Login

Figure 2.2: Patient registration

In the dashboard, the patient see the appointment form. After making appointment, he redirects to the payment page.



Appointments Add New

Create New Appointment

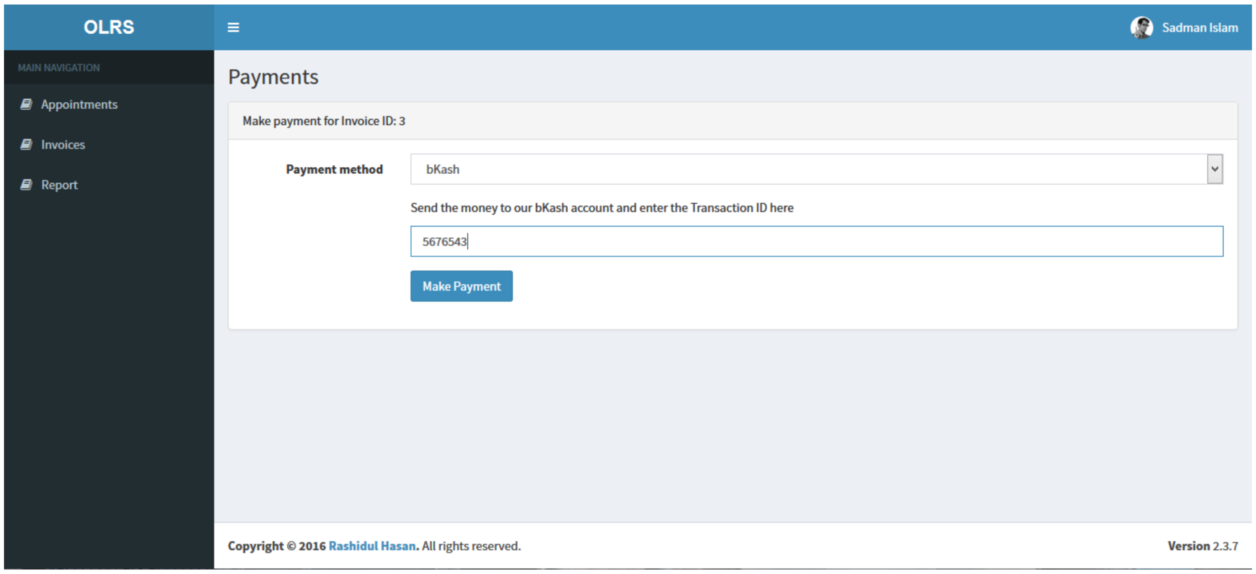
| | |
|---|---|
| Test | Complete Blood Count |
| | Your selected time slot is 7am - 9am Please choose a date to see if its available |
| Date | 2016-12-13 |
| Available slots | ○7am - 9am ●9am - 11am |
| | Create |

Copyright © 2016 Rashidul Hasan. All rights reserved.

Version 2.3.7

Figure 2.3: Patient appointment

There he has to pay the fees for the test through bkash payment method.


OLRS

Sadman Islam

MAIN NAVIGATION
- Appointments
- Invoices
- Report

Payments

Make payment for Invoice ID: 3

Payment method: bKash

Send the money to our bKash account and enter the Transaction ID here

5676543

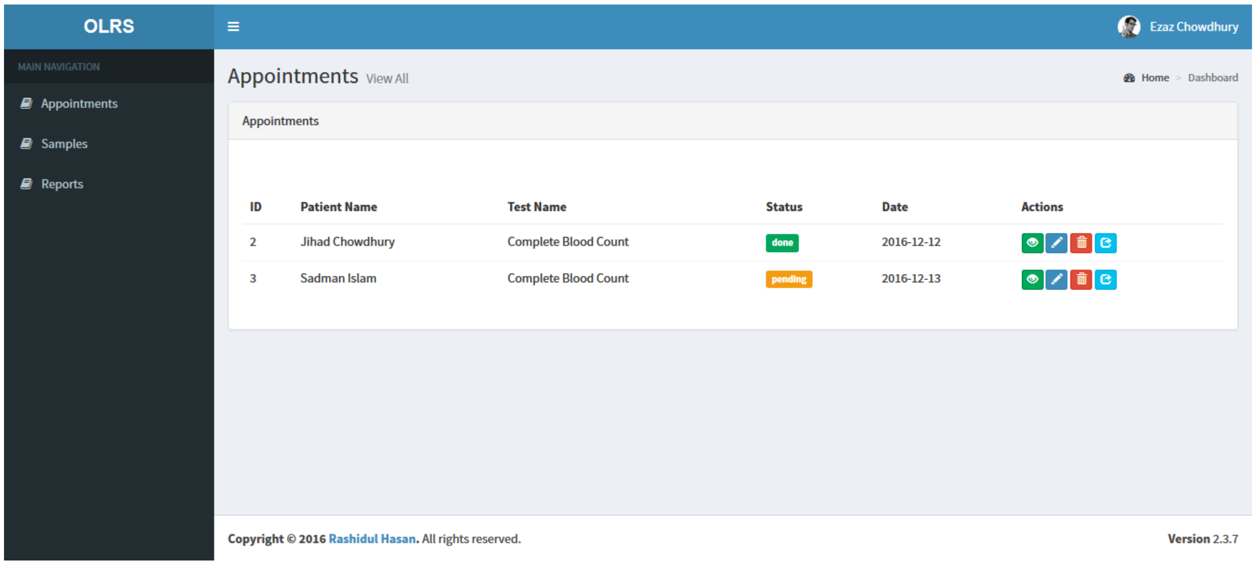
Make Payment

Copyright © 2016 Rashidul Hasan. All rights reserved.    Version 2.3.7

Figure 2.4: Patient payment by bkash

After payment, patient has to wait for giving the sample and after that for the report.

Now this part will begin for the laboratorist. After login, the laboratorist see the pending work. Then he collects the sample and generate report.


OLRS

Ezaz Chowdhury

MAIN NAVIGATION
- Appointments
- Samples
- Reports

Appointments View All    Home > Dashboard

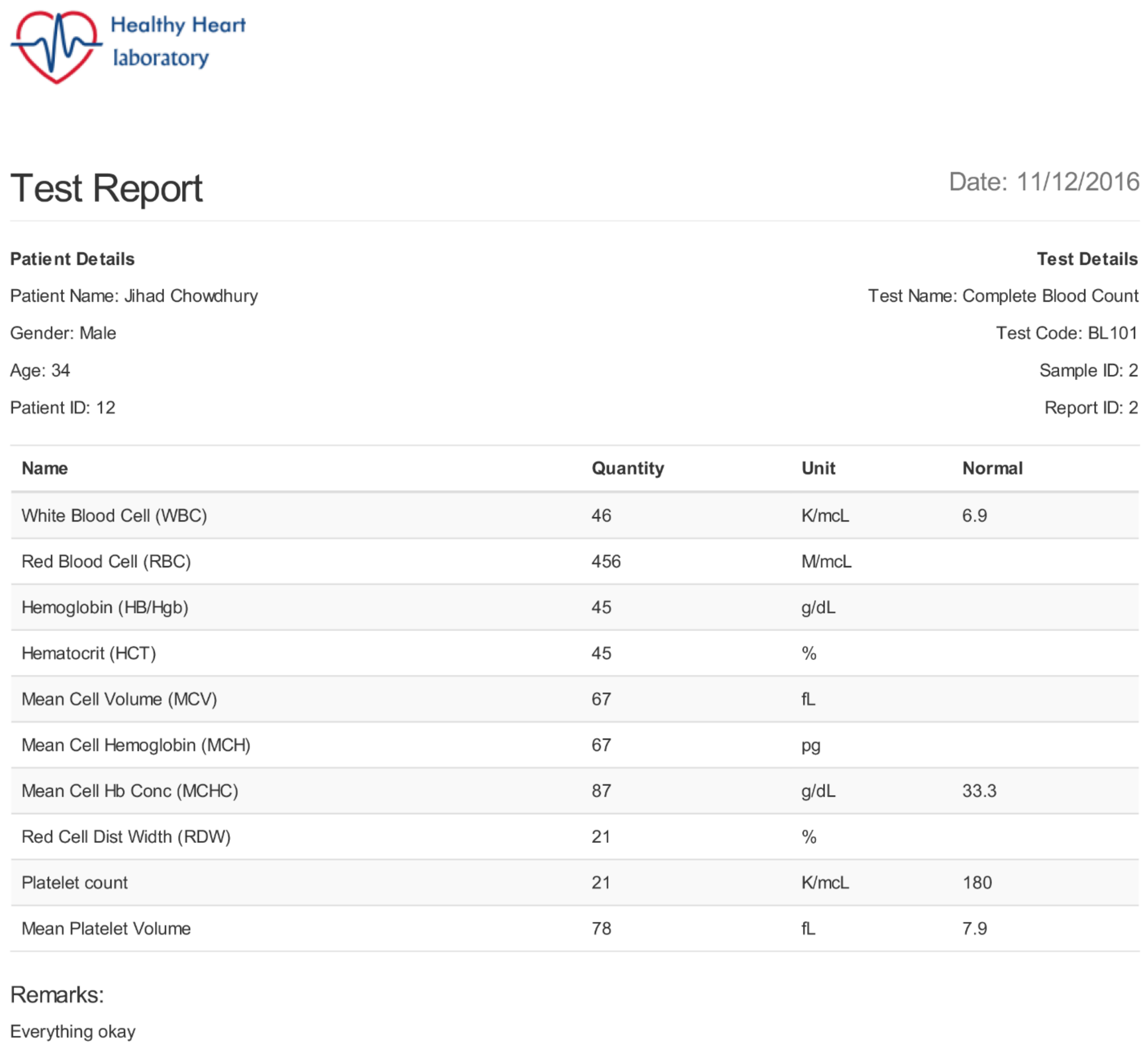
Appointments

| ID | Patient Name | Test Name | Status | Date | Actions |
|---|---|---|---|---|---|
| 2 | Jihad Chowdhury | Complete Blood Count | done | 2016-12-12 | |
| 3 | Sadman Islam | Complete Blood Count | pending | 2016-12-13 | |

Copyright © 2016 Rashidul Hasan. All rights reserved.    Version 2.3.7

Figure 2.5: Laboratorist dashboard

After generating report, the report looks like below this.



Healthy Heart laboratory

Test Report

Date: 11/12/2016

**Patient Details**

Patient Name: Jihad Chowdhury

Gender: Male

Age: 34

Patient ID: 12

**Test Details**

Test Name: Complete Blood Count

Test Code: BL101

Sample ID: 2

Report ID: 2

| Name | Quantity | Unit | Normal |
|---|---|---|---|
| White Blood Cell (WBC) | 46 | K/mcL | 6.9 |
| Red Blood Cell (RBC) | 456 | M/mcL | |
| Hemoglobin (HB/Hgb) | 45 | g/dL | |
| Hematocrit (HCT) | 45 | % | |
| Mean Cell Volume (MCV) | 67 | fL | |
| Mean Cell Hemoglobin (MCH) | 67 | pg | |
| Mean Cell Hb Conc (MCHC) | 87 | g/dL | 33.3 |
| Red Cell Dist Width (RDW) | 21 | % | |
| Platelet count | 21 | K/mcL | 180 |
| Mean Platelet Volume | 78 | fL | 7.9 |

Remarks:

Everything okay

Figure 2.6: Test report with details

Now the patient can download the report from the internet.

Now here we show the admin part. After login, admin see his dashboard.

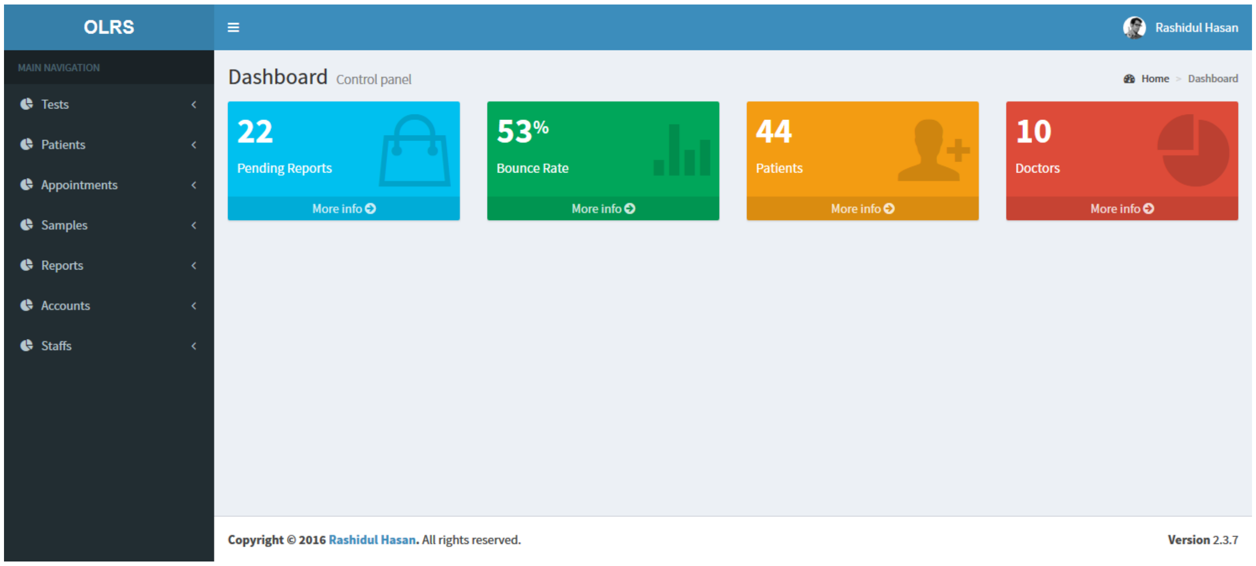
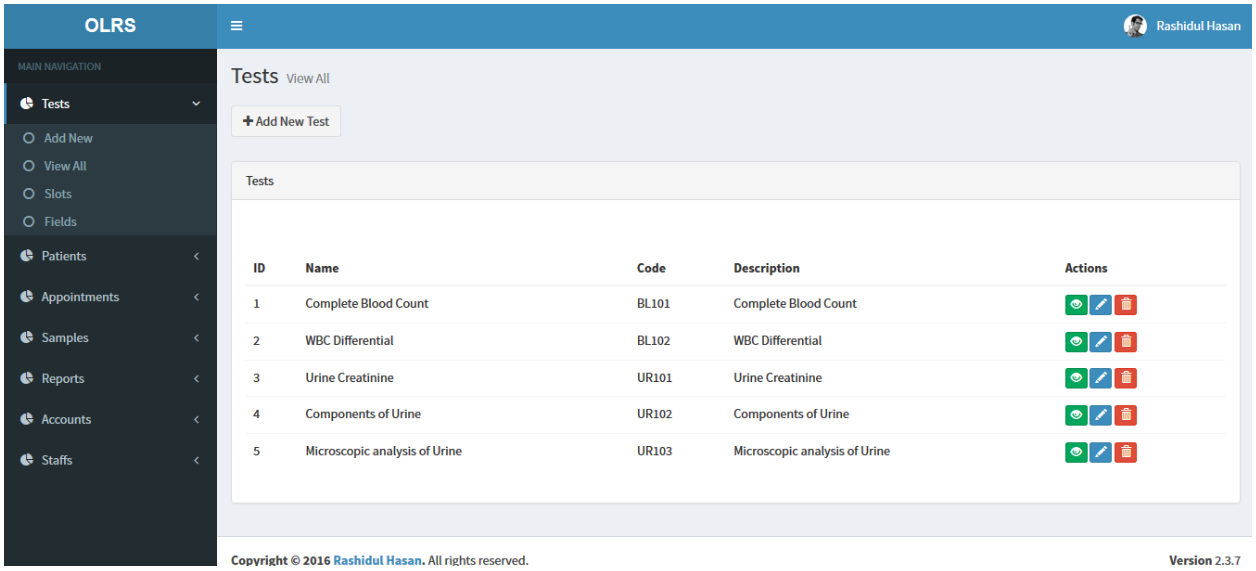


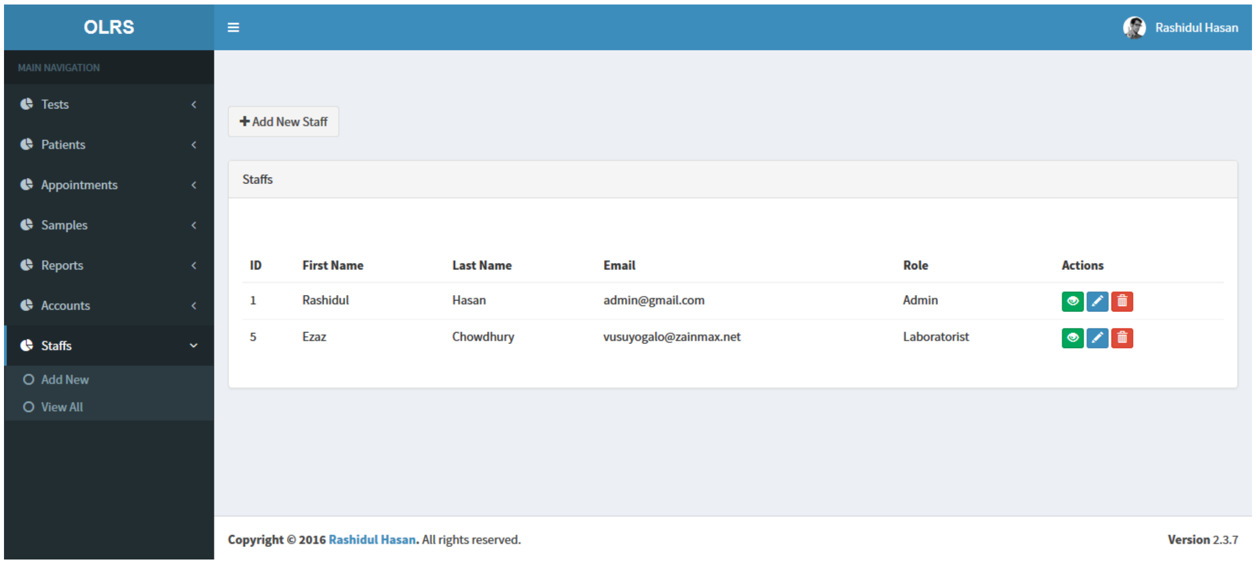
Figure 2.6: Admin dashboard

Here he can view patient list, appointment list, sample list, report list and transaction history. He also create new test item, new slot for test, test details for generating report, create new staff, manage them etc. Here we provide some screen.



| ID | Name | Code | Description | Actions |
|---|---|---|---|---|
| 1 | Complete Blood Count | BL101 | Complete Blood Count | |
| 2 | WBC Differential | BL102 | WBC Differential | |
| 3 | Urine Creatinine | UR101 | Urine Creatinine | |
| 4 | Components of Urine | UR102 | Components of Urine | |
| 5 | Microscopic analysis of Urine | UR103 | Microscopic analysis of Urine | |

Figure 2.7: Test details



Staffs

| ID | First Name | Last Name | Email | Role | Actions |
|---|---|---|---|---|---|
| 1 | Rashidul | Hasan | admin@gmail.com | Admin | |
| 5 | Ezaz | Chowdhury | vusuyogalo@zainmax.net | Laboratorist | |

Copyright © 2016 Rashidul Hasan. All rights reserved. Version 2.3.7

Figure 2.8: Staff details

Online Diagnostic Lab reporting System is powerful, flexible, and easy to use and is designed and developed to deliver real conceivable benefits to diagnostic system in hospital.

# CHAPTER 3

# DESIGN

# Design

## 3.1 Introduction

Design is the first step in the development phase for any techniques and principals for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.

Once the software requirements have been analyzed and specified, the software design involves three technical activities – design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its case of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer’s requirement into finished software or a system.

Design is the place where quality is fostered in development. Software design is a process through which requirement are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation or requirements into data.

## 3.2 Introduction To UML

### 3.2.1 UML Design

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the software system and its components. It is a graphical language, which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure, maintain, and control information about the systems.

The UML is a language for:

- Visualizing
- Specifying
- Constructing
- Documenting

**Visualizing**

Through UML we see or visualize an existing system and ultimately we visualize how the system is going to be after implementation. Unless we think, we cannot implement. UML helps to visualize, how the components of the system communicate and interact with each other.

**Specifying**

Specifying means building, models that are precise, unambiguous and complete UML addresses the specification of all the important analysis design, implementation decisions that must be made in developing and deploying a software system.

**Constructing**

UML models can be directly connected to a variety of programming language through mapping a model from UML to a programming language like JAVA or C++ or VB. Forward Engineering and Reverse Engineering is possible through UML.

**Documenting**

The Deliverables of a project apart from coding are some Artifacts, which are critical in controlling, measuring and communicating about a system during its developing requirements, architecture, desire, source code, project plans, tests, prototypes releasers, etc...

## 3.2.2 UML Approach

### 3.2.2.1 UML Diagram

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices and arcs. You draw diagram to visualize a system from different perspective, so a diagram is a projection into a system. For all but most trivial systems, a diagram represents an elided view of the elements that make up a system. The same element may appear in all diagrams, only a few diagrams, or in no diagrams at all. In theory, a diagram may contain any combination of things and relationships. In practice, however, a small number of common combinations arise, which are consistent with the five most useful views that comprise the architecture of a software-intensive system. For this reason, the UML includes nine such diagrams:

1. Class diagram
2. Object diagram
3. Use case diagram
4. Sequence diagram
5. Collaboration diagram
6. Entity Relationship diagram
7. Activity diagram
8. Component diagram
9. Deployment diagram

### 3.2.3 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

Use case diagrams are formally included in two modeling languages defined by the Unified modeling language (UML) and the systems modeling language (sysML)**.**

### 3.2.3.1 Use case diagram of our project:

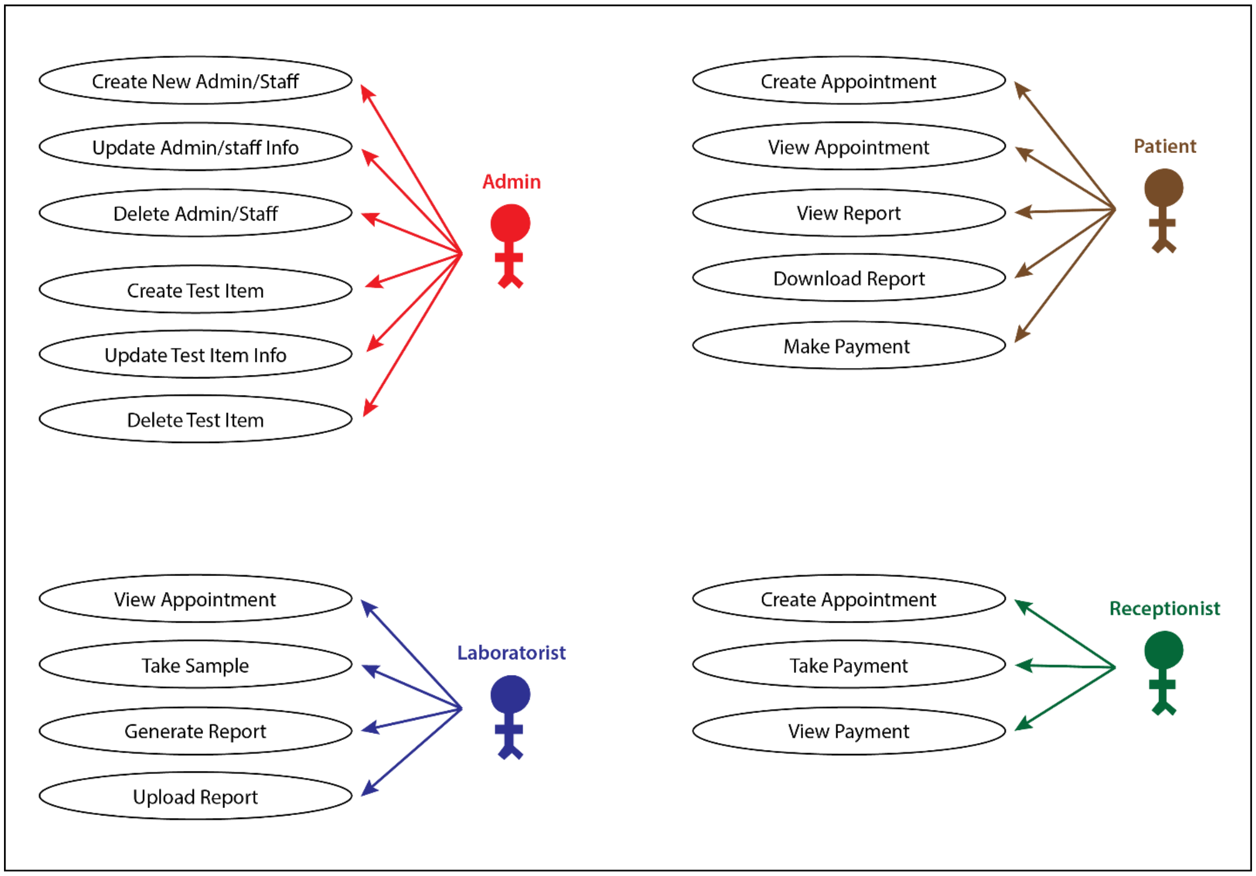


Figure 3.1: Use case Diagram

## 3.3 Data Flow Diagram

### 3.3.1 Definition

The DFD takes an input-process-output view of a system i.e. data objects flow into the software, are transformed by processing elements and resultant data objects flow out of the software. Data objects represented by labeled arrows and transformations are represented by circles also called as bubbles. DFD is presented in a hierarchical fashion i.e. the first data flow model represents the system as a whole. Subsequent DFD refine the context diagram (level 0 DFD), providing increasing details with each subsequent level. The DFD enables the software engineer to develop models of the information domain and functional domain at the same time. As the DFD is refined into greater levels of details, the analyst performs an implicit functional decomposition of the system. At the same time. DFD refinement results in a corresponding refinement of the data as it moves through the process that embody the application. A context-level DFD for the system the primary external entities produce information for use by the system and consume information generated by the system. The labeled arrows represents data objects or data hierarchy.

### 3.3.2 Rules for Data Flow Diagram

- Fix the scope of the system by means of context diagram.
- Organize the DFD so that the main sequence is in the actions.
- Reads left to right and top to bottom.
- Identify all inputs and outputs.
- Identify and label each process internal to the system with rounded circles
- A process is required for all the transformation and the transfers. Therefore, never connect a data store to a data source or the destinations or another data store with just a data flow arrow.
- DO not indicate hardware and ignore control information.

- Make sure the name of the process accurately convey everything the process the done.
- There must not be unnamed process.
- Indicate external sources and destination of the data with squares.
- Number each occurrence of repeated external entities.
- Identify all flows for each process steps, except simple record retrievals.
- Label data flow on each arrow.
- Use details flow on each arrow.
- Use the details flow arrow to indicate data movements.

### 3.3.3 Data Flow Diagram of the Project

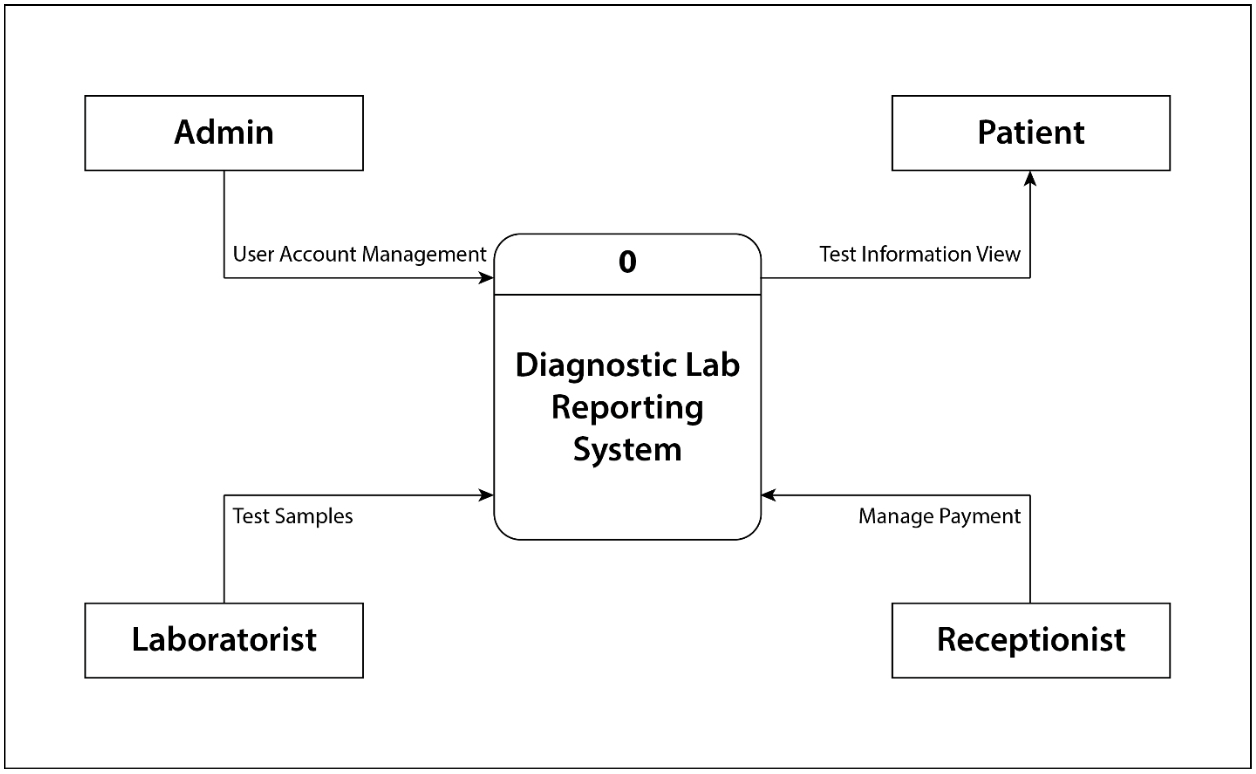


Figure 3.2: Level 0 context Data Flow Diagram

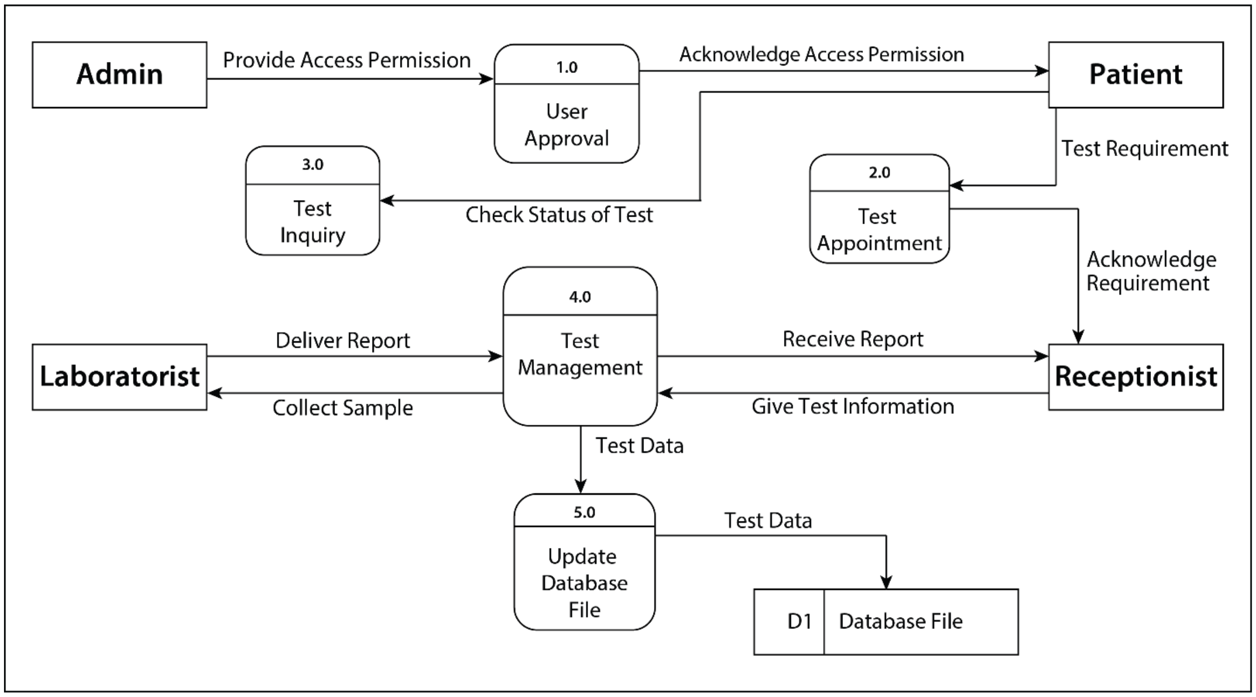


Figure 3.3: Level 1 context Data Flow Diagram

## 3.4 Entity Relationship Diagram

Entity-Relationship Diagram is a graphical representation of entities and their relationship to each other's. It describes how data is related to each other. An entity is a piece of data - an object or a concept about which data is stored. A relationship is how the data is shared between entities. In E-R Diagram, there are 3 main Components:

| **Symbol** | **Name** | **Description** |
|---|---|---|
| ▭ | Entity | An entity can be any object, place, person or anything. |
| ⬭ | Attribute | An Attribute Describes a property or characteristics of an entity. |
| ◇ | Relationship | A Relationship Describes relation between entities. |

## 3.4.1 E-R Diagram of the Project

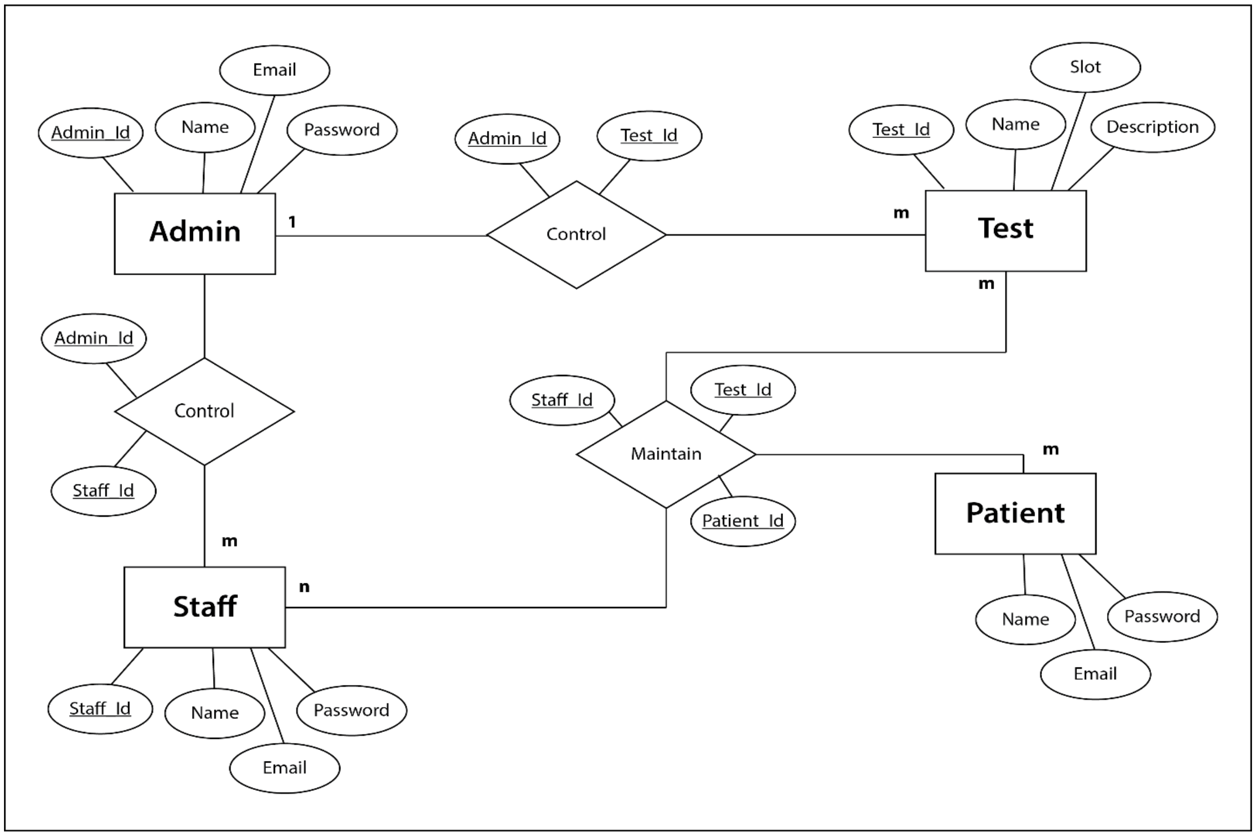


Figure 3.4: E-R Diagram

In this E-R Diagram, there are 4 entity and 5 relationship among the entities and so many attributes. Entities are –

- Admin
- Staff
- Test
- Patient

Relationships are –

| Entities | Relationship |
|---|---|
| **Admin to Staff** | 1 to many |
| **Admin to Test** | 1 to many |
| **Staff to Test** | many to many |
| **Staff to Patient** | many to many |
| **Patient to Test** | many to many |

## 3.5 Data Diagram of the Project

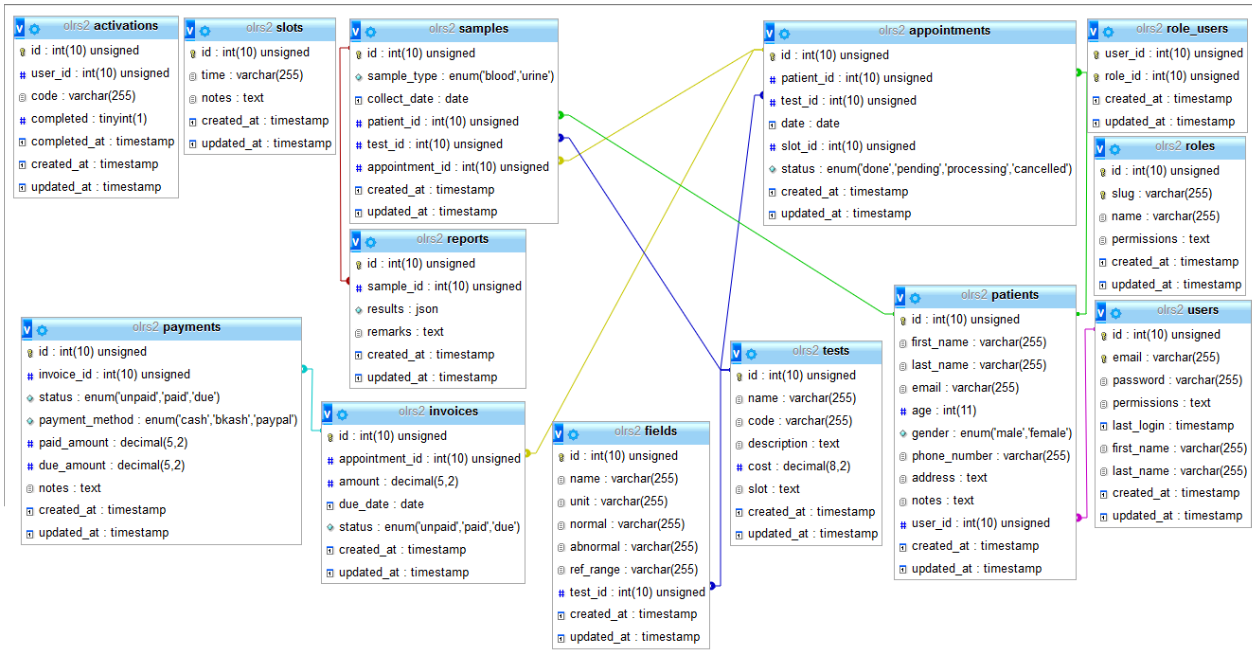


Figure 3.5: Data Diagram

# CHAPTER 4

# IMPLEMENTATION

# Implementation

## 4.1 Overview of the Technology Used

Online Diagnostic Lab Reporting System can be entered using a username and password as we discussed earlier. We also describe about the potential uses of this system. The interface of the software is user friendly.

## 4.2 Software

In our application, we use Laravel, MySQL server to execute a simple and effective model for the operation of the diagnostic center. We make our interface easily understandable and also make it simple as much as possible for future maintenance by the staff.

### 4.2.1 Front End Technology

Laravel is a free, open-source PHP web framework, created by Taylor Otwell and intended for the development of web applications following the model–view–controller (MVC) architectural pattern. Some of the features of Laravel are a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar.

As of March 2015, Laravel is regarded as one of the most popular PHP frameworks, together with Symfony2, Nette, CodeIgniter, Yii2 and other frameworks.
The following features serve as Laravel's key design points (where not specifically noted, descriptions refer to the features of Laravel 3).

- *Bundles* provide a modular packaging system since the release of Laravel 3, with bundled features already available for easy addition to applications. Furthermore, Laravel 4 uses Composer as a dependency manager to add framework-agnostic and Laravel-specific PHP packages available from the Packagist repository.

- *Eloquent ORM* (object-relational mapping) is an advanced PHP implementation of the active record pattern, providing at the same time internal methods for enforcing constraints on the relationships between database objects. Following the active record pattern, Eloquent ORM presents database tables as classes, with their object instances tied to single table rows.
- *Query builder*, available since Laravel 4, provides a more direct database access alternative to the Eloquent ORM. Instead of requiring SQL queries to be written directly, Laravel's query builder provides a set of classes and methods capable of building queries programmatically. It also allows selectable caching of the results of executed queries.
- *Application logic* is an integral part of developed applications, implemented either by using controllers or as part of the route declarations. The syntax used to define application logic is similar to the one used by Sinatra framework.
- *Reverse routing* defines a relationship between the links and routes, making it possible for later changes to routes to be automatically propagated into relevant links. When the links are created by using names of existing routes, the appropriate uniform resource identifiers (URIs) are automatically created by Laravel.
- *Restful controllers* provide an optional way for separating the logic behind serving HTTP GET and POST requests.
- *Class auto loading* provides automated loading of PHP classes without the need for manual maintenance of inclusion paths. On-demand loading prevents inclusion of unnecessary components, so only the actually used components are loaded.
- *View composers* serve as customizable logical code units that can be executed when a view is loaded.
- *Blade* templating engine combines one or more templates with a data model to produce resulting views, doing that by transpiring the templates into cached PHP

code for improved performance. Blade also provides a set of its own control structures such as conditional statements and loops, which are internally mapped to their PHP counterparts. Furthermore, Laravel services may be called from Blade templates, and the templating engine itself can be extended with custom directives.

- *IoC containers* make it possible for new objects to be generated by following the inversion of control (IoC) principle, in which the framework calls into the application- or task-specific code, with optional instantiating and referencing of new objects as singletons.
- *Migrations* provide a version control system for database schemas, making it possible to associate changes in the applications codebase and required changes in the database layout. As a result, this feature simplifies the deployment and updating of Laravel-based applications.
- *Database seeding* provides a way to populate database tables with selected default data that can be used for application testing or be performed as part of the initial application setup.
- *Unit testing* is provided as an integral part of Laravel, which itself contains unit tests that detect and prevent regressions in the framework. Unit tests can be run through the provided artisan command-line utility.
- *Automatic pagination* simplifies the task of implementing pagination, replacing the usual manual implementation approaches with automated methods integrated into Laravel.
- *Form request* is a feature of Laravel 5 that serves as the base for form input validation by internally binding event listeners, resulting in automated invoking of the form validation methods and generation of the actual form.
- *Homestead* - a Vagrant virtual machine that provides Laravel developers with all the tools nessessary to develop Laravel straight out of the box, including, Ubuntu, Gulp , Bower and other development tools that are useful in developing full scale web applications.

**Artisan CLI**

Laravel's command-line interface (CLI), called Artisan, was initially introduced in Laravel 3 with a limited set of capabilities. Laravel's later migration to a Composer-based architecture allowed Artisan to incorporate different components from the Symfony framework, resulting in the availability of additional Artisan features in Laravel 4.

The features of Artisan are mapped to different subcommands of the artisan command-line utility, providing functionality that aids in managing and building Laravel-based applications. Common uses of Artisan include managing database migrations and seeding, publishing package assets, and generating boilerplate code for new controllers and migrations; the latter frees the developer from creating proper code skeletons. The functionality and capabilities of Artisan can also be expanded by implementing new custom commands, which, for example, may be used to automate application-specific recurring tasks.

### 4.2.2 Back End Technology

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius' daughter and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

MySQL is a central component of the LAMP open-source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python". Applications that use the MySQL database include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, and Drupal. MySQL is also used in many high-profile, large-scale websites, including Google, Facebook, Twitter, Flickr and YouTube. MySQL is offered under two different editions: the open source MySQL Community Server and the proprietary Enterprise Server. MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and is built from the same code base.

Major features as available in MySQL 5.6:

- A broad subset of ANSI SQL 99, as well as extensions
- Cross-platform support
- Stored procedures, using a procedural language that closely adheres to SQL/PSM
- Triggers
- Cursors
- Updatable views
- Online DDL when using the InnoDB Storage Engine.
- Information schema
- Performance Schema that collects and aggregates statistics about server execution and query performance for monitoring purposes.
- A set of SQL Mode options to control runtime behavior, including a strict mode to better adhere to SQL standards.
- X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using the default InnoDB storage engine
- Transactions with savepoints when using the default InnoDB Storage Engine. The NDB Cluster Storage Engine also supports transactions.
- ACID compliance when using InnoDB and NDB Cluster Storage Engines.
- SSL support
- Query caching
- Sub-SELECTs (i.e. nested SELECTs)
- Built-in replication support (i.e., master-master replication and master-slave replication) with one master per slave, many slaves per master. Multi-master

replication is provided in MySQL Cluster and multi-master support can be added to unclustered configurations using Galera Cluster.

- Full-text indexing and searching
- Embedded database library
- Unicode support
- Partitioned tables with pruning of partitions in optimizer
- Shared-nothing clustering through MySQL Cluster
- Multiple storage engines, allowing one to choose the one that is most effective for each table in the application.
- Native storage engines InnoDB, MyISAM, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, NDB Cluster.
- Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second.

The developers release minor updates of the MySQL Server approximately every two months. The sources can be obtained from MySQL's website or from MySQL's GitHub repository, both under the GPL license.

# CHAPTER 5
# CONCULSIONS

# Conclusion

## 5.1 Discussion

By the grace of Almighty Allah, we have come to the end of our project report. It took us a year to complete. Our group members work hard to make this project efficient and viable.

Diagnostic services are critical to delivering high-quality patient care, but constantly changing reimbursement levels, rapidly evolving technology, and chronic shortages of skilled professionals can be daunting.

Since we are entering details of the patients and their diagnostic information electronically in the "Online Diagnostic Lab Reporting System", data will be secured. Using this application we can retrieve patient's history with a single click. Thus processing information will be faster. It guarantees accurate maintenance of Patient details. It easily reduces the book keeping task and thus reduces the human effort and increases accuracy speed.
The system also provides the facility of backup as per the requirement. It will be able to adequate scope for modification in future if it is necessary. It will be able to reduce potential errors in regulatory compliance, operational processes, and patient safety in the diagnostic system.

## 5.2 Future Opportunity

In this project “Online Diagnostic Lab Reporting System” we try to build an efficient, robust but yet very simple system by which thousands of patients in our country could be served. This model is made focusing on a particular diagnostic center. But the interesting part is this, this model can be expanded for several diagnostic centers. As it is a web application, it can be used from anywhere across the country. The database has designed in that way so that it can be connected with several diagnostic centers and they can be used in large scale.

# Appendix

**PatientsController.php**

```
<?php
namespace App\Http\Controllers;
use App\Http\Requests;
use App\Http\Controllers\Controller;
use App\Patient;
use Illuminate\Http\Request;
use Session;
use Sentinel;
use Mail;

class PatientsController extends Controller
{
    public function index()
    {
        $patients = Patient::paginate(25);
        return view('patients.index', compact('patients'));
    }

    public function create()
    {
        $gender = Patient::getPossibleEnumValues('gender');
        return view('patients.create', ['gender' => $gender]);
    }

    public function store(Request $request)
    {

        $requestData = $request->all();
        // create corresponding user for this patient
        $id = $this->createPatientUserAccount($requestData);
        $requestData['user_id'] = $id;
        // create patient
        Patient::create($requestData);
        Session::flash('flash_message', 'Patient added! A password has been sent to your
mail.Please use it to login');

        return redirect('patients');
    }


    public function show($id)
    {
        $patient = Patient::findOrFail($id);
        return view('patients.show', compact('patient'));
    }

    public function edit($id)
    {
        $gender = Patient::getPossibleEnumValues('gender');
        $patient = Patient::findOrFail($id);

        return view('patients.edit', ['patient' => $patient, 'gender' => $gender]);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  int  $id
     * @param \Illuminate\Http\Request $request
     *
     * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
     */
    public function update($id, Request $request)
    {

        $requestData = $request->all();
        //dd($requestData);

        $patient = Patient::findOrFail($id);
        $patient->update($requestData);
        Session::flash('flash_message', 'Patient updated!');
        return redirect('patients');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     *
     * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
     */
    public function destroy($id)
    {
        Patient::destroy($id);
        Session::flash('flash_message', 'Patient deleted!');
        return redirect('patients');
    }

    private function createPatientUserAccount($data){

        $role = Sentinel::findRoleBySlug('patient');
        $pass = substr(md5(rand()), 0, 8);
        $userData['email'] = $data['email'];
        $userData['first_name'] = $data['first_name'];
        $userData['last_name'] = $data['last_name'];
        $userData['password'] = $pass;
        $user = Sentinel::registerAndActivate($userData);
        $role->users()->attach($user);
        $this->sendMail($user, $pass);
        return $user->id;
    }

    private function sendMail($user, $pass){

        Mail::send('emails.password', ['pass' => $pass, 'user' => $user], function($message)
use ($user) {
```

```
            $message->to($user->email);
            $message->from('info@olrs.com', 'Healthy Heart Laboratory');
            $message->subject('Welcome to Healthy Heart Laboratory');
        });
    }
}
```

**AppointmentsController.php**

```
<?php
namespace App\Http\Controllers;
use App\Http\Requests;
use App\Http\Controllers\Controller;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Session;
use App\Appointment;
use App\Test;
use App\Patient;
use App\Invoice;
use App\Slot;

class AppointmentsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\View\View
     */
    public function index()
    {
        $appointments = Appointment::paginate(25);

        return view('appointments.index', compact('appointments'));
    }

    /**
```

```
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\View\View
 */
public function create()
{
    $tests = Test::all();
    $patients = Patient::all();
    $patient_data = [];
    $test_data = [];

    foreach ($patients as $patient) {
        $patient_data[$patient->id] = $patient->fullname;
    }
    foreach ($tests as $test) {
        $test_data[$test->id] = $test->name;
    }
    //dd($patient_data);
    //dd($test_data);
    return view('appointments.create', ['tests' => $test_data, 'patients' => $patient_data]);

}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 *
 * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
 */
public function store(Request $request)
{

    $requestData = $request->all();
    $requestData['status'] = 'pending';
    //dd($requestData);
```

```
        $appointment = Appointment::create($requestData);
        // create invoice for this appointment
        $id = $this->createInvoice($appointment);
        //dd($id);
        // send it to patient's mail
        Session::flash('flash_message', 'Appointment added!');
        return redirect('appointments');
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     *
     * @return \Illuminate\View\View
     */
    public function show($id)
    {
        $appointment = Appointment::findOrFail($id);
        $slot = Slot::findOrFail($appointment->slot_id);
        //dd($appointment->slot->time);

        return  view('appointments.show',  ['appointment'  =>  $appointment,  'slot'  =>  $slot-
>time]);
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     *
     * @return \Illuminate\View\View
     */
    public function edit($id)
    {
        $tests = Test::all();
        $patients = Patient::all();
    //dd($patients);
    foreach ($patients as $patient) {
    $patient_data[$patient->id] = $patient->name;
    }
    foreach ($tests as $test) {
        $test_data[$test->id] = $test->name;
    }

    $appointment = Appointment::findOrFail($id);
    $slot = Slot::findOrFail($appointment->slot_id);

    return view('appointments.edit', ['appointment' => $appointment,
            'patients' => $patient_data,
            'slot' => $slot->time,
            'tests' => $test_data]);
}

/**
 * Update the specified resource in storage.
 *
 * @param  int  $id
 * @param \Illuminate\Http\Request $request
 *
 * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
 */
public function update($id, Request $request)
{

    $requestData = $request->all();
    $appointment = Appointment::findOrFail($id);
    $appointment->update($requestData);
    Session::flash('flash_message', 'Appointment updated!');

    return redirect('appointments');
}

/**
```

```
 * Remove the specified resource from storage.
 *
 * @param  int  $id
 *
 * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
 */
public function destroy($id)
{
    Appointment::destroy($id);
    Session::flash('flash_message', 'Appointment deleted!');
    return redirect('appointments');
}


private function createInvoice($appointment){

    $invoice = new Invoice();
    $invoice->appointment_id = $appointment->id;
    $invoice->amount = $appointment->test->cost;
    $invoice->due_date = Carbon::today()->addDays(3);
    $invoice->status = 'unpaid';
    $invoice->save();

    return $invoice->id;
}

public function checkAvailableSlots(Request $request){

    $data = $request->all();
    //return response()->json($data);

    $appointments = Appointment::where('test_id', $data['test_id'])
                            ->where('status', 'pending')
                            ->where('date', $data['date'])
                            ->get();

    $test = Test::where('id', $data['test_id'])->first();;
```

```
        $slots = unserialize($test->slot);
        //return response()->json($appointment->toArray());

        if ( $appointments->isEmpty() ) {

            // all slots are available
            // return all slots for this test
            $slots = Slot::find($slots);
            return response()->json(['status' => 'found', 'slots' => $slots->toArray()]);
        } else {

            foreach ($appointments as $appointment) {

                if(($key = array_search($appointment->slot_id, $slots)) !== false) {
                    unset($slots[$key]);
                }

            }

            if ( empty($slots) ) {

                // all slots are occupied
                return response()->json(['status' => 'all_busy']);
            } else {
                // some slots are still available, return them
                $slots = Slot::find($slots);
                return response()->json(['status' => 'found', 'slots' => $slots->toArray()]);
            }
        }

        return response()->json(['status' => 'error']);
        //return response()->json(['slots' => $slots]);
    }
}
```

**create.blade.php**

```
<script>
    var dateToday = new Date();
    var dates = $(".my-datepicker").datepicker({
        startDate: new Date(),
        format: 'yyyy-mm-dd',
        todayHighlight: true,
        todayBtn: true,
        autoclose: true
    });
</script>

<script>
    $('#test_name').on('change', function (e) {

        if( !$('.my-datepicker').val() ) return;
        fetchSlots();
    });

    $('.my-datepicker').on('changeDate', function (e) {
        if( !$('#test_name').val() ) return;
        fetchSlots();
    });


    function fetchSlots(){

        $('#loader').fadeIn(100);

        var test_id = $('#test_name').val();
        var date = $('.my-datepicker').val();
        console.log('test id: ' + test_id);
        console.log('date: ' + date);

        $.ajax({
           url: '/appointments/check',
           type: 'GET',
           dataType: 'json',
           data: {
             test_id: test_id,
             date: date

           },

           error: function(data) {
             console.log(data);
             $('#loader').fadeOut(100);
             $('#loader').html('<p>Something went wrong!</p>');
           },

           success: function(data) {
             console.log(data);
             $('#loader').fadeOut(100);
             $('#slot_container > p').remove();
             showSlots(data);
           }
        });
    }

    function showSlots(data){

        if (data.status == 'all_busy') {
           $('#slot_container').html('<p  style="color:red;">All  slots  are  occupied!  Please
select another day.</p>');
        } else {

           // populate the radio input elements for slots
           var slots = data.slots;
           var html = ''
           for (var i = 0; i < slots.length; i++) {
              console.log(slots[i].time);
              //var el = '<input type="radio" name="slot" value="' + slots[i].id +'">' + slots[i].time + '<br>';
              html += '<span style="margin-right:5px;"><input type="radio" name="slot_id" value="' + slots[i].id +'">' + slots[i].time + '  </span>';
              //html.concat(el);
           }

           $('#slot_container').html(html);
        }
}
</script>
```

LoginController.php

```
<?php
namespace App\Http\Controllers\Auth;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Foundation\Auth\AuthenticatesUsers;
use Sentinel;

class LoginController extends Controller
{
   public function login(){
      return view('auth.login');
   }

   public function postLogin(Request $request)
   {
        Sentinel::authenticate($request->all());
        //dd($request->all());
      if ($user = Sentinel::check()){
        return redirect('/');
      } else {
        return redirect()->back()->withErrors(['login_failed' => 'Incorrect credentials!']);
      }
   }
```

```
    public function postLogout(Request $request){
            Sentinel::logout();
            return redirect('/login');
    }
}
```

**Appointment.php**

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
use App\Patient;
use App\Test;
use App\Slot;

class Appointment extends Model
{
    /**
     * The database table used by the model.
     *
     * @var string
     */
    protected $table = 'appointments';

    /**
    * The database primary key value.
    *
    * @var string
    */
    protected $primaryKey = 'id';

    /**
     * Attributes that should be mass-assignable.
     *
     * @var array
     */
    protected $fillable = ['patient_id', 'test_id', 'date', 'slot_id', 'status'];
    public function patient(){
        return $this->belongsTo('App\Patient');
    }

    public function test(){
        return $this->belongsTo('App\Test');
    }

    public function slot(){
        return $this->hasOne('App\Slot');
    }
}
```

**create_patients_table.php**

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
class CreatePatientsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('patients', function(Blueprint $table) {

            $table->engine = 'InnoDB';

            $table->increments('id');
            $table->string('first_name');
            $table->string('last_name');
            $table->string('email');
            $table->integer('age');
            $table->enum('gender', ['male', 'female']);
            $table->string('phone_number');
            $table->text('address')->nullable();
            $table->text('notes')->nullable();
            $table->integer('user_id')->unsigned()->nullable();
            $table->timestamps();
        });

        Schema::table('patients', function($table) {

           $table->foreign('user_id')
                  ->references('id')->on('users')
                  ->onDelete('set null');

       });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('patients');
    }
}
```

**web.php**

```
<?php
//Route::get('/home', 'HomeController@index');
// auth routes
Route::get('/login', [
        'uses' => 'Auth\LoginController@login'
]);
Route::post('/login', [
        'uses' => 'Auth\LoginController@postLogin'
]);
Route::post('/logout', [
	'uses' => 'Auth\LoginController@postLogout'
]);

//Route::get('/register', 'Auth\RegistrationController@register');
Route::group(['middleware' => 'admin'], function(){

	Route::get('/', [
		'uses' => 'DashboardController@index',
		'as' => 'dashboard.home'
	]);
	Route::resource('users', 'UsersController');
	Route::resource('patients', 'PatientsController');
	Route::resource('tests', 'TestsController');
	// appointments
	Route::get('/appointments/check',
'AppointmentsController@checkAvailableSlots');
	Route::resource('appointments', 'AppointmentsController');

	//Route::resource('posts', 'Admin\PostsController');
	Route::resource('posts', 'PostsController');
	Route::resource('slots', 'SlotsController');

	// samples
	// Route::get('samples/create/{a_id}', [
	//	'as' => 'samples.collect',
	//	'uses' => 'SamplesController@create'
	// ]);
	Route::resource('samples', 'SamplesController');

	Route::resource('fields', 'FieldsController');
	//Auth::routes();

	// report
	Route::get('reports/print/{id}', [
		'as' => 'reports.print',
		'uses' => 'ReportsController@printReport'
	]);
```

```
        Route::resource('reports', 'ReportsController');

        // invoices
        Route::get('invoices/print/{id}', [
                'as' => 'invoices.print',
                'uses' => 'InvoicesController@printInvoice'
        ]);
        Route::resource('invoices', 'InvoicesController');
        Route::resource('payments', 'PaymentsController');


        // seperate routes for patient dashboard
        //Route::get('/home/appointment', 'PatientDashboardController@appointment')

});


// route group for all routes of patient dashboard
Route::group(['prefix' => 'home', 'as' => 'home.', 'middleware' => 'admin', 'namespace' =>
'Patient'], function () {

   // appointments
        Route::get('/appointments/check',
'AppointmentsController@checkAvailableSlots');
        Route::resource('appointments', 'AppointmentsController');

        // report
        Route::get('/reports/print/{id}', [
                'as' => 'reports.print',
                'uses' => 'ReportsController@printReport'
        ]);
        Route::resource('reports', 'ReportsController');

        // invoices
        Route::get('/invoices/print/{id}', [
                'as' => 'invoices.print',
                'uses' => 'InvoicesController@printInvoice'
        ]);
        Route::resource('invoices', 'InvoicesController');

        Route::resource('payments', 'PaymentsController');
});


// route group for frontend
Route::group(['prefix' => 'front', 'as' => 'front.', 'namespace' => 'Front'], function () {

   // appointments
        Route::get('/index', ['uses' => 'HomeController@index', 'as' => 'index']);
        Route::get('/register', ['uses' => 'HomeController@register', 'as' => 'register']);
        Route::post('/register', ['uses' => 'HomeController@postRegister']);
});
```

**PatientsTableSeeder.php**

```
<?php
use Illuminate\Database\Seeder;
use Faker\Factory as Faker;
use Carbon\Carbon;

class PatientsTableSeeder extends Seeder
{
   /**
    * Run the database seeds.
    *
    * @return void
    */
   public function run()
   {
      $faker = Faker::create();
      $gender = ['male','female'];

      foreach (range(1,10) as $index) {
              DB::table('patients')->insert([
                 'first_name' => $faker->firstName,
           'last_name' => $faker->lastName,
           'email' => $faker->email,
                 'age' => $faker->numberBetween($min = 19, $max = 66),
                 'gender' => $faker->randomElement($gender),
                 'phone_number' => $faker->phoneNumber,
                 'address' => $faker->address,
                 'notes' => $faker->text($maxNbChars = 200) ,
                 'created_at' => Carbon::now()->format(('Y-m-d H:i:s'))
              ]);
      }
   }
}
```

**HomeController.php**

```
<?php
namespace App\Http\Controllers\Front;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Test;
use App\Slot;
use Sentinel;
use Mail;
use App\Patient;

class HomeController extends Controller
{
    public function index()
    {
          $data = [];
          $tests = Test::all()->toArray();
          foreach ($tests as $test) {
                    $item = [];
                    $item['test_name'] = $test['name'];
                    $item['test_id'] = $test['id'];
                    $slots = unserialize($test['slot']);
                    $test_slots = Slot::whereIn('id', $slots)->get()->toArray();
                    $item['slots'] = $test_slots;
                    array_push($data, $item);
          }

          //dd($data);
          return view('front.index', ['data' => $data]);
     }

     public function register()
     {
          if ($user = Sentinel::check()){
          return redirect('/home/appointments/create ');
        } else {
          return view('front.register');
        }
          //return view('front.register');
     }

     public function postRegister(Request $request)
     {
          $data = $request->all();
          //dd($data);
          $id = $this->createPatientUserAccount($data);
          //dd($id);

        $data['user_id'] = $id;

        // create patient
        Patient::create($data);

          return redirect('/home/appointments/create ');
     }

     private function createPatientUserAccount($data){

        $role = Sentinel::findRoleBySlug('patient');

        $pass = substr(md5(rand()), 0, 8);
        $userData['email'] = $data['email'];
        $userData['first_name'] = $data['first_name'];
        $userData['last_name'] = $data['last_name'];
        $userData['password'] = $pass;

        $user = Sentinel::registerAndActivate($userData);

        $role->users()->attach($user);

        // login the patient
        $login_data['email'] = $userData['email'];
        $login_data['password'] = $pass;
        Sentinel::authenticate($login_data);

        // send password via mail
        $this->sendMail($user, $pass);

        return $user->id;

    }

    private function sendMail($user, $pass){

        Mail::send('emails.password', ['pass' => $pass, 'user' => $user], function($message)
use ($user) {

            $message->to($user->email);
            $message->from('info@olrs.com', 'Healthy Heart Laboratory');
            $message->subject('Welcome to Healthy Heart Laboratory');
        });
    }
}
```

**ReportsController.php**

```
<?php
namespace App\Http\Controllers;
```

```
use App\Http\Requests;
use App\Http\Controllers\Controller;
use App\Report;
use App\Field;
use App\Appointment;
use App\Sample;
use Illuminate\Http\Request;
use Session;

class ReportsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\View\View
     */
    public function index()
    {
        $reports = Report::paginate(25);

        return view('reports.index', compact('reports'));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\View\View
     */
    public function create(Request $request)
    {
        $requestData = $request->all();
        $test_fields = Field::where('test_id', $requestData['test_id'])->get()->toArray();
        //dd($test_fields);
        return view('reports.create', ['data' => $requestData, 'test_fields' => $test_fields]);
    }
```

```
/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 *
 * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
 */
public function store(Request $request)
{

    $requestData = $request->all();
    //dd($requestData);
    $results = [];

    foreach ($requestData as $key => $value) {
        if (is_int($key)) {

            //dd($key);
            // get the field setails
            $field = Field::where('id', $key)->first()->toArray();

            $result = [];

            $result['name'] = $field['name'];
            $result['quantity'] = $value;
            $result['unit'] = $field['unit'];
            $result['normal'] = $field['normal'];

            array_push($results, $result);

            // remove from actual array
            unset($requestData[$key]);
        }
    }

    $requestData['results'] = json_encode($results);
    //dd($requestData);

    $report = Report::create($requestData);
    $sample = Sample::findOrFail($report->sample_id);
    $appointment = Appointment::where('id', $sample->appointment_id)->first();
    //dd($appointment);
    $appointment->status = 'done';
    $appointment->save();

    //dd($id);
    //$appointment = $report->appointment;

    //dd($appointment);
    //$appointment->status = 'done';
    //$appointment->save();

    //$report = Report::find($id);
    //$report->results = json_encode($results);
    //$report->save();
    Session::flash('flash_message', 'Report added!');
    return redirect('reports');
}

/**
 * Display the specified resource.
 *
 * @param  int  $id
 *
 * @return \Illuminate\View\View
 */
public function show($id)
{
    $report = Report::findOrFail($id);

    //dd(json_decode($report->results));
```

```
        return view('reports.show', ['report' => $report, 'results' => json_decode($report-
>results)]);
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     *
     * @return \Illuminate\View\View
     */
    public function edit($id)
    {
        $report = Report::findOrFail($id);

        return view('reports.edit', compact('report'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  int  $id
     * @param \Illuminate\Http\Request $request
     *
     * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
     */
    public function update($id, Request $request)
    {

        $requestData = $request->all();

        $report = Report::findOrFail($id);
        $report->update($requestData);

        Session::flash('flash_message', 'Report updated!');

        return redirect('reports');
```

```
}

/**
 * Remove the specified resource from storage.
 *
 * @param  int  $id
 *
 * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
 */
public function destroy($id)
{
    Report::destroy($id);

    Session::flash('flash_message', 'Report deleted!');

    return redirect('reports');
}


/**
 * Show the form for editing the specified resource.
 *
 * @param  int  $id
 *
 * @return \Illuminate\View\View
 */
public function printReport($id)
{
    //return view('pdf.reportv3');
   $report = Report::findOrFail($id)->first();
    $data['results'] = json_decode($report->results);
   //dd($data);
   $sample = $report->sample;
    $patient = $sample->patient;
    $test = $sample->test;

    //dd($sample->toArray());
```

```
        $data['report'] = $report->toArray();
        $data['sample'] = $sample->toArray();
        $data['patient'] = $patient->toArray();
        $data['test'] = $test->toArray();



        $pdf = \PDF::loadView('pdf.report', $data);
        //dd($pdf);
        return $pdf->inline();
        //dd($report);

        //return view('reports.edit', compact('report'));
    }
}
```

**InvoicesController.php**

```
<?php
namespace App\Http\Controllers;
use App\Http\Requests;
use App\Http\Controllers\Controller;
use App\Invoice;
use Illuminate\Http\Request;
use Session;

class InvoicesController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\View\View
     */
    public function index()
    {
        $invoices = Invoice::paginate(25);

        return view('invoices.index', compact('invoices'));
```

```
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\View\View
 */
public function create()
{
    return view('invoices.create');
}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 *
 * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
 */
public function store(Request $request)
{

    $requestData = $request->all();
    Invoice::create($requestData);
    Session::flash('flash_message', 'Invoice added!');
    return redirect('invoices');
}

/**
 * Display the specified resource.
 *
 * @param  int  $id
 *
 * @return \Illuminate\View\View
 */
public function show($id)
{
    $invoice = Invoice::findOrFail($id);
    return view('invoices.show', compact('invoice'));
}

/**
 * Show the form for editing the specified resource.
 *
 * @param  int  $id
 *
 * @return \Illuminate\View\View
 */
public function edit($id)
{
    $invoice = Invoice::findOrFail($id);
    return view('invoices.edit', compact('invoice'));
}

/**
 * Update the specified resource in storage.
 *
 * @param  int  $id
 * @param \Illuminate\Http\Request $request
 *
 * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
 */
public function update($id, Request $request)
{

    $requestData = $request->all();

    $invoice = Invoice::findOrFail($id);
    $invoice->update($requestData);
    Session::flash('flash_message', 'Invoice updated!');

    return redirect('invoices');
}
```

```
    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     *
     * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
     */
    public function destroy($id)
    {
        Invoice::destroy($id);
        Session::flash('flash_message', 'Invoice deleted!');
        return redirect('invoices');
    }

    public function printInvoice($id){
        $invoice = Invoice::findOrFail($id);
        $patient = $invoice->appointment->patient;
        $test = $invoice->appointment->test;
        $data = [];
        $data['invoice'] = $invoice->toArray();
        $data['patient'] = $patient->toArray();
        $data['test'] = $test->toArray();

        $pdf = \PDF::loadView('pdf.invoice', $data);
        //dd($pdf);
        return $pdf->inline();
    }
}
```

# Bibliography

1. Learning Laravel 5 : Building Practical Applications by Nathan Wu.
2. HTML, XHTML & CSS, 6th Edition by Elizabeth Castro.
3. High Performance MySQL by Baron Schwartz, Peter Ziatsev.
4. PHP and MySQL Web Development (4th Edition) by Luke Welling, Laura Thomson
5. https://www.youtube.com/
6. http://www.w3schools.com/php/default.asp
7. https://laravel.com/